Package ‘phylosim’

September 8, 2016

Version 3.0.2
Date 2016-09-04
Title Flexible Simulations of Biological Sequence Evolution
Author Botond Sipos <sbotond@gmail.com>, Gregory Jordan <greg@ebi.ac.uk>
Maintainer Botond Sipos <sbotond@gmail.com>
Depends R (>= 2.15.2), R.oo (>= 1.13.0), ape (>= 2.3), compoisson (>= 0.3), ggplot2 (>= 0.9.3)
Suggests
Imports R.methodsS3
Description An extensible object-oriented framework for the Monte Carlo simulation of sequence evolution written in 100 percent R. It is built on the top of the R.oo and ape packages and uses Gillespie’s direct method to simulate substitutions, insertions and deletions.
License GPL (>= 3)
URL https://github.com/bsipos/phylosim
LazyLoad TRUE
NeedsCompilation no
Repository CRAN
Date/Publication 2016-09-08 20:14:57

R topics documented:

== .Alphabet .................................................. 12
== .Process .................................................... 13
Alphabet ..................................................... 14
AminoAcidAlphabet ........................................ 16
AminoAcidSequence .......................................... 18
AminoAcidSubst ............................................. 20
AnyAlphabet .................................................. 22
areSynonymous.CodonAlphabet ................................. 23
as.character.Alphabet ....................................... 24
topics documented:

as.character.Event ........................................... 25
as.character.GeneralSubstitution ................................. 26
as.character.PhyloSim ......................................... 27
as.character.Process .......................................... 28
as.character.QMatrix ........................................... 29
as.character.Sequence ......................................... 30
as.character.Site .............................................. 31
as.character.ToleranceSubstitution ............................... 32
attachHookToNode.PhyloSim .................................... 33
attachProcess.Sequence ......................................... 34
attachProcess.Site .............................................. 35
attachSeqToNode.PhyloSim ...................................... 36
BrownianAlphabet .............................................. 37
BrownianSequence .............................................. 38
BinarySubst ..................................................... 40
BrownianInsertor .............................................. 42
BrownianPath.BrownianInsertor ................................ 45
buildFromPAML.AminoAcidSubst ................................ 46
buildFromPAML.CodonSubst ..................................... 47
checkConsistency.Alphabet .................................... 48
checkConsistency.AminoAcidSubst ............................... 49
checkConsistency.BinarySubst ................................ 49
checkConsistency.BrownianInsertor ............................. 50
checkConsistency.CodonAlphabet ................................ 51
checkConsistency.CodonSequence ................................ 51
checkConsistency.CodonSubst .................................. 52
checkConsistency.CodonUNREST ................................ 53
checkConsistency.ContinuousDeletor ............................ 53
checkConsistency.ContinuousInsertor ........................... 54
checkConsistency.DiscreteDeletor .............................. 55
checkConsistency.DiscreteInsertor ............................ 55
checkConsistency.Event ........................................ 56
checkConsistency.F81 .......................................... 57
checkConsistency.F84 .......................................... 57
checkConsistency.FastFieldDeletor ............................. 58
checkConsistency.GeneralDeletor .............................. 59
checkConsistency.GeneralInDel ................................ 59
checkConsistency.GeneralInsertor .............................. 60
checkConsistency.GeneralSubstitution ........................... 61
checkConsistency.GTR ........................................ 61
checkConsistency.GY94 ........................................ 62
checkConsistency.HKY ........................................ 63
checkConsistency.JC69 ........................................ 63
checkConsistency.K80 ........................................ 64
checkConsistency.K81 ........................................ 65
checkConsistency.PhyloSim .................................... 65
checkConsistency.Process ..................................... 66
checkConsistency.PSRoot ....................................... 67
topics documented:

checkConsistency.PSRootSummary ........................................ 67
checkConsistency.QMatrix .................................................. 68
checkConsistency.Sequence .................................................. 69
checkConsistency.Site ......................................................... 69
checkConsistency.T92 .......................................................... 70
checkConsistency.TN93 .......................................................... 71
checkConsistency.ToleranceSubstitution .................................. 71
checkConsistency.UNREST ..................................................... 72
clearStates.Sequence .......................................................... 73
clone.GeneralSubstitution .................................................... 74
close.Process ................................................................. 75
clone.Sequence ............................................................... 76
clone.ToleranceSubstitution .................................................. 77
CodonAlphabet ................................................................. 78
CodonSequence ............................................................... 80
CodonSubst ................................................................. 82
CodonUNREST ................................................................. 84
ContinuousDeleter ............................................................ 86
ContinuousInsertor ........................................................... 88
copySubSequence.Sequence .................................................. 91
cpREV ................................................................. 92
Debug.PhyloSim ............................................................... 94
deleteSubSequence.Sequence ................................................ 95
detachHookFromNode.PhyloSim ............................................. 96
detachProcess.Sequence ..................................................... 97
detachProcess.Site ............................................................ 98
detachSeqFromNode.PhyloSim .............................................. 99
DiscreteDeleter ............................................................. 100
DiscreteInsertor ............................................................ 102
ECMrest ................................................................. 105
ECMunrest ................................................................. 107
enableVirtual.PSRoot ....................................................... 109
Event ................................................................. 110
exportStatTree.PhyloSim .................................................... 113
F81 ................................................................. 114
F84 ................................................................. 117
FastFieldDeleter ............................................................ 119
flagTotalRate.Site ............................................................ 122
GeneralDeleter ............................................................. 123
GeneralInDel .............................................................. 126
GeneralInsertor ............................................................ 128
GeneralSubstitution .......................................................... 132
generateInsert.GeneralInsertor ............................................ 135
getAcceptBy.GeneralInDel .................................................. 136
getAcceptWin.GeneralInsertor ............................................. 137
getAlignment.PhyloSim ...................................................... 138
getAlignmentLength.PhyloSim ............................................. 139
getAlphabet.GeneralSubstitution .......................................... 140
getAlphabet.Process .......................................................... 141
getAlphabet.QMatrix ......................................................... 142
getAlphabet.Site ............................................................. 143
getAlphabet.ToleranceSubstitution ....................................... 144
getAlphabets.Sequence ........................................................ 145
getAncestral.Sequence ....................................................... 146
getAncestral.Site ............................................................. 147
getBaseFreqs.F81 .............................................................. 148
getBaseFreqs.F84 .............................................................. 149
getBaseFreqs.GTR .............................................................. 150
getBaseFreqs.HKY .............................................................. 151
getBaseFreqs.K80 .............................................................. 152
getBaseFreqs.K81 .............................................................. 153
getBaseFreqs.TN93 .............................................................. 154
getBigRate.Sequence .......................................................... 155
getBranchEvents.PhyloSim .................................................... 156
getCodonFreqs.GY94 ........................................................... 157
getComments.PSRoot .......................................................... 158
getCumulativeRates.Sequence ................................................ 159
getcumulativeRatesFromRange.Sequence ................................... 160
getDeletionTolerance.Sequence ............................................. 161
getDist.ContinuousDeletor .................................................... 162
gDist.ContinuousInsertor ..................................................... 163
getEdge.PhyloSim ............................................................. 164
getEdges.PhyloSim ............................................................ 165
getEquDist.GeneralSubstitution ............................................. 166
getEquDist.ToleranceSubstitution .......................................... 167
getEventRate.GeneralSubstitution .......................................... 168
getEventRate.QMatrix ........................................................ 169
gEventRate.ToleranceSubstitution .......................................... 170
gEventRateAtSite.GeneralSubstitution ..................................... 171
gEventRateAtSite.ToleranceSubstitution .................................. 172
gEvents.Sequence ............................................................. 173
gEvents.Site ................................................................. 174
gEventsAtSite.FastFieldDeletor ............................................. 175
gEventsAtSite.GeneralDeletor ............................................... 176
gEventsAtSite.GeneralInsertor .............................................. 177
gEventsAtSite.GeneralSubstitution ......................................... 178
gEventsAtSite.GY94 ........................................................... 179
gEventsAtSite.Process ........................................................ 180
gEventsAtSite.ToleranceSubstitution ...................................... 181
gGenerateBy.GeneralInsertor .................................................. 182
gHandler.Event ................................................................ 183
gGetHandler.Event ................................................................ 184
gGetInsertHook.GeneralInsertor .............................................. 185
gGetId.PhylomSim ............................................................... 186
gGetId.Process ................................................................ 187
gGetId.QMatrix ................................................................. 188
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getInsertionTolerance.Sequence</td>
<td>189</td>
</tr>
<tr>
<td>getKappa.F81</td>
<td>190</td>
</tr>
<tr>
<td>getKappa.GY94</td>
<td>191</td>
</tr>
<tr>
<td>getLength.Sequence</td>
<td>192</td>
</tr>
<tr>
<td>getLengthParam1.FastFieldDeletor</td>
<td>193</td>
</tr>
<tr>
<td>getLengthParam2.FastFieldDeletor</td>
<td>194</td>
</tr>
<tr>
<td>getLogFile.PhyloSim</td>
<td>195</td>
</tr>
<tr>
<td>getLogLevel.PhyloSim</td>
<td>196</td>
</tr>
<tr>
<td>getMatrix.QMatrix</td>
<td>197</td>
</tr>
<tr>
<td>getMaxLength.ContinuousDeletor</td>
<td>198</td>
</tr>
<tr>
<td>getMaxLength.ContinuousInsertor</td>
<td>199</td>
</tr>
<tr>
<td>getMethodsList.PSRoot</td>
<td>200</td>
</tr>
<tr>
<td>getName.Event</td>
<td>201</td>
</tr>
<tr>
<td>getName.PhyloSim</td>
<td>202</td>
</tr>
<tr>
<td>getName.Process</td>
<td>203</td>
</tr>
<tr>
<td>getName.QMatrix</td>
<td>204</td>
</tr>
<tr>
<td>getName.Sequence</td>
<td>205</td>
</tr>
<tr>
<td>getNedges.PhyloSim</td>
<td>206</td>
</tr>
<tr>
<td>getNodes.PhyloSim</td>
<td>207</td>
</tr>
<tr>
<td>getNtips.PhyloSim</td>
<td>208</td>
</tr>
<tr>
<td>getOmegas.CodonSequence</td>
<td>209</td>
</tr>
<tr>
<td>getOmegaScalingFactor.GY94</td>
<td>210</td>
</tr>
<tr>
<td>getParameterAtSite.Process</td>
<td>211</td>
</tr>
<tr>
<td>getParameterAtSites.Sequence</td>
<td>212</td>
</tr>
<tr>
<td>getPhylo.PhyloSim</td>
<td>213</td>
</tr>
<tr>
<td>getPosition.Event</td>
<td>214</td>
</tr>
<tr>
<td>getProbs.DiscreteDeletor</td>
<td>215</td>
</tr>
<tr>
<td>getProbs.DiscreteInsertor</td>
<td>216</td>
</tr>
<tr>
<td>getProcess.Event</td>
<td>217</td>
</tr>
<tr>
<td>getProcess.QMatrix</td>
<td>218</td>
</tr>
<tr>
<td>getProcesses.Sequence</td>
<td>219</td>
</tr>
<tr>
<td>getProcesses.Site</td>
<td>220</td>
</tr>
<tr>
<td>getProposeBy.GeneralInDel</td>
<td>221</td>
</tr>
<tr>
<td>getQMatrix.GenerallSubstitution</td>
<td>222</td>
</tr>
<tr>
<td>getQMatrix.ToleranceSubstitution</td>
<td>223</td>
</tr>
<tr>
<td>getRate.Event</td>
<td>224</td>
</tr>
<tr>
<td>getRate.GeneralInDel</td>
<td>225</td>
</tr>
<tr>
<td>getRate.GenerallSubstitution</td>
<td>226</td>
</tr>
<tr>
<td>getRate.QMatrix</td>
<td>227</td>
</tr>
<tr>
<td>getRate.ToleranceSubstitution</td>
<td>228</td>
</tr>
<tr>
<td>getRateList.GenerallSubstitution</td>
<td>229</td>
</tr>
<tr>
<td>getRateList.QMatrix</td>
<td>230</td>
</tr>
<tr>
<td>getRateList.ToleranceSubstitution</td>
<td>231</td>
</tr>
<tr>
<td>getRateMultipliers.Sequence</td>
<td>232</td>
</tr>
<tr>
<td>getRateParam.F81</td>
<td>233</td>
</tr>
<tr>
<td>getRateParam.F84</td>
<td>233</td>
</tr>
<tr>
<td>getRateParam.GTR</td>
<td>234</td>
</tr>
<tr>
<td>getRateParam.HKY</td>
<td>235</td>
</tr>
</tbody>
</table>
R topics documented:

getRateParam.K80 ........................................ 236
getRateParam.K81 ........................................ 237
getRateParam.T92 ........................................ 238
getRateParam.TN93 ....................................... 239
getRateParamList.F81 .................................. 240
getRateParamList.F84 .................................. 241
getRateParamList.GTR .................................. 242
getRateParamList.HKY .................................. 243
getRateParamList.K80 .................................. 244
getRateParamList.K81 .................................. 245
getRateParamList.T92 .................................. 246
getRateParamList.TN93 ................................ 247
getRootNode.PhyloSim ................................ 248
getRootSeq.PhyloSim .................................. 249
getScale.BrownianInsertor ............................ 250
getScaledMatrix.QMatrix ............................... 251
getSeqFromNode.PhyloSim .............................. 252
getSequence.Site ....................................... 253
getSequences.PhyloSim ................................ 254
getSite.Event ........................................... 255
getSites.Sequence ...................................... 256
getSiteSpecificParamIds.Process .................. 257
getSiteSpecificParamList.Process ................ 258
getSize.Alphabet ..................................... 259
getSizes.DiscreteDeleter ............................. 260
getSizes.DiscreteInsertor ............................ 261
getState.Site .......................................... 262
getStates.Sequence ................................... 263
getString.Sequence ................................... 264
getSymbolFreqs.Sequence ............................. 265
getSymbolLength.Alphabet ........................... 266
getSymbols.Alphabet .................................. 267
getTableId.CodonAlphabet ............................ 268
getTemplateSeq.GeneralInsertor .................... 269
getTheta.T92 ........................................... 270
getTipLabels.PhyloSim ................................ 271
getTips.PhyloSim ...................................... 272
getToleranceMargin.FastFieldDeleter .............. 273
getTotalRate.Site ..................................... 274
getTotalRates.Sequence ............................... 275
getTotalRatesFromRange.Sequence ................. 276
getTransTable.CodonAlphabet ....................... 277
getType.Alphabet ..................................... 278
type.BrownianInsertor ............................... 279
type.FastFieldDeleter ................................ 280
getUniqueAlphabets.Sequence ....................... 281
getUniqueProcesses.Sequence ....................... 282
**R topics documented:**

- `getWriteProtected.Alphabet` .................................................. 283
- `getWriteProtected.Event` ....................................................... 284
- `getWriteProtected.Process` ..................................................... 285
- `getWriteProtected.QMatrix` .................................................... 286
- `getWriteProtected.Sequence` ................................................... 287
- `globalConsistencyCheck.PSRoot` .............................................. 288
- `GTR` ................................................................................ 289
- `GY94` ............................................................................. 291
- `hasSiteSpecificParameter.Process` ............................................ 294
- `hasSymbols.Alphabet` ........................................................... 295
- `hasUndefinedRate.GeneralInDel` ............................................... 296
- `hasUndefinedRate.GeneralSubstitution` ..................................... 297
- `hasUndefinedRate.Process` ...................................................... 298
- `hasUndefinedRate.ToleranceSubstitution` ................................... 299
- `HKY` ............................................................................. 300
- `insertSequence.Sequence` ....................................................... 302
- `intersect.list.PSRoot` ........................................................... 303
- `is.Alphabet` ................................................................. 304
- `is.CodonAlphabet` ............................................................... 305
- `is.CodonUNREST` ................................................................. 306
- `is.Event` ......................................................................... 306
- `is.GeneralDeletor` ............................................................... 307
- `is.GeneralInDel` ................................................................. 308
- `is.GeneralInsertor` ............................................................... 309
- `is.GeneralSubstitution` ........................................................... 309
- `is.GY94` ......................................................................... 310
- `is.na.PSRoot` ................................................................. 311
- `is.phylo` ....................................................................... 312
- `is.Process` ................................................................. 313
- `is.PSRoot` ...................................................................... 314
- `is.QMatrix` ..................................................................... 315
- `is.Sequence` ................................................................. 316
- `is.Site` .......................................................................... 317
- `is.tip.PhyloSim` ................................................................. 318
- `is.ToleranceSubstitution` ..................................................... 319
- `is.Attached.Site` ............................................................... 320
- `isEmpty.Alphabet` .............................................................. 321
- `isStartCodon.CodonAlphabet` ............................................. 322
- `isStopCodon.CodonAlphabet` ................................................ 323
- `JC69` ........................................................................... 324
- `JTT` ............................................................................. 326
- `JTT.dcmut` .................................................................... 328
- `K80` ............................................................................. 330
- `K81` ............................................................................. 333
- `LG` ............................................................................... 336
- `li.PSRoot` .................................................................... 338
- `Log.PhyloSim` ................................................................. 339
- `mtArt` ........................................................................... 340
setAcceptBy.GeneralInDel
setAcceptWin.GeneralInsertor
setAlignment.PhyloSim
setAlphabet.GeneralSubstitution
setAlphabet.Process
setAlphabet.QMatrix
setAlphabet.Site
setAlphabet.ToleranceSubstitution
setAlphabets.Sequence
setAncestral.Sequence
setAncestral.Site
setBaseFreqs.F81
setBaseFreqs.F84
setBaseFreqs.GTR
setBaseFreqs.HKY
setBaseFreqs.K80
setBaseFreqs.K81
setBaseFreqs.TN93
setBigRate.Sequence
setBranchEvents.PhyloSim
setCodonFreqs.GY94
setComments.PSRoot
setCumulativeRates.Sequence
setDeletionTolerance.Sequence
setDist.ContinuousDeletor
setDist.ContinuousInsertor
setEdges.PhyloSim
setEquDist.GeneralSubstitution
setEquDist.ToleranceSubstitution
setEvents.Site
setGenerateBy.GeneralInsertor
setHandler.Event
setId.PhyloSim
setId.Process
setId.QMatrix
setId.Sequence
setInsertHook.GeneralInsertor
setInsertionTolerance.Sequence
setKappa.F84
setKappa.GY94
setLength.Sequence
setMaxLength.ContinuousDeletor
setMaxLength.ContinuousInsertor
setLogFile.PhyloSim
setLogLevel.PhyloSim
setMatrix.QMatrix
setMaxLength.ContinuousDeletor
setMaxLength.ContinuousInsertor
setMethodsList.PSRoot  ................................................. 458
setName.Event  ......................................................... 458
setName.PhyloSim ..................................................... 459
setName.Process ...................................................... 460
setName.QMatrix ....................................................... 461
setName.Sequence ..................................................... 462
setNedges.PhyloSim ................................................... 463
setNodes.PhyloSim ..................................................... 464
setNtips.PhyloSim ...................................................... 464
setOmegas.CodonSequence ........................................... 465
setParameterAtSite.Process ......................................... 466
setParameterAtSites.Sequence ....................................... 467
setPhylo.PhyloSim ..................................................... 468
setPosition.Event ..................................................... 469
setProbs.DiscreteDeleter ............................................ 470
setProbs.DiscreteInsertor ......................................... 471
setProcess.Event ....................................................... 472
setProcess.QMatrix ................................................... 473
setProcesses.Sequence .............................................. 474
setProcesses.Site .................................................... 475
setProposeBy.GeneralInDel ........................................ 476
setQMatrix.GeneralSubstitution .................................. 477
setQMatrix.ToleranceSubstitution ................................. 478
setRate.Event .......................................................... 480
setRate.GeneralInDel ................................................ 481
setRate.GeneralSubstitution ....................................... 482
setRate.GY94 .......................................................... 483
setRate.QMatrix ........................................................ 484
setRate.ToleranceSubstitution ..................................... 485
setRateList.GeneralSubstitution .................................. 486
setRateList.QMatrix .................................................. 487
setRateList.ToleranceSubstitution ................................. 488
setRateMultipliers.Sequence ....................................... 489
setRateParam.F81 ...................................................... 490
setRateParam.F84 ...................................................... 491
setRateParam.GTR ...................................................... 492
setRateParam.HKY ...................................................... 493
setRateParam.K80 ...................................................... 494
setRateParam.K81 ...................................................... 495
setRateParam.T92 ...................................................... 496
setRateParam.TN93 ..................................................... 497
setRateParamList.F81 ................................................ 498
setRateParamList.F84 ................................................ 499
setRateParamList.GTR ................................................ 500
setRateParamList.HKY ................................................ 501
setRateParamList.K80 ................................................ 502
setRateParamList.K81 ................................................ 503
setRateParamList.T92 ................................................ 504
topics documented:

setRateParamList.TN93 ......................................................... 505
setRootNode.PhyloSim .......................................................... 506
setRootSeq.PhyloSim ............................................................. 507
setScale.BrownianInsertor .................................................... 508
setScaledMatrix.QMatrix ....................................................... 509
setSequence.Site ............................................................... 510
setSequences.PhyloSim .......................................................... 511
setSite.Event ................................................................. 511
setSiteSpecificParamIds.Process ............................................. 512
setSiteSpecificParamList.Process .......................................... 513
setSize.Alphabet .............................................................. 514
setSizes.DiscreteDeleter ....................................................... 514
setSizes.DiscreteInsertor ...................................................... 515
setState.Site ................................................................. 516
setStates.Sequence ............................................................. 517
setString.Sequence ............................................................. 518
setSymbolLength.Alphabet .................................................... 519
setSymbols.Alphabet .......................................................... 520
setTableId.CodonAlphabet ..................................................... 521
setTemplateSeq.GeneralInsertor ............................................. 521
setTheta.T92 ................................................................. 522
setTipLabels.PhyloSim .......................................................... 523
setTips.PhyloSim ............................................................... 524
setToleranceMargin.FastFieldDeleter ...................................... 525
setTotalRate.Site .............................................................. 526
setTotalRates.Sequence ....................................................... 526
setTreeLength.PhyloSim ....................................................... 527
setType.Alphabet .............................................................. 529
setType.BrownianInsertor .................................................... 530
setType.FastFieldDeleter ...................................................... 530
setUniqueAlphabets.Sequence ................................................ 531
setUniqueProcesses.Sequence ................................................ 532
setWriteProtected.Alphabet ................................................ 533
setWriteProtected.Event ...................................................... 534
setWriteProtected.Process ................................................... 535
setWriteProtected.QMatrix ................................................... 536
setWriteProtected.Sequence ................................................ 536
Simulate.PhyloSim ............................................................. 537
Site ................................................................. 538
stringLength ................................................................. 541
stringLengthVector ........................................................... 542
summary.Alphabet ............................................................ 543
summary.AminoAcidSubst ..................................................... 544
summary.BinarySubst ........................................................ 545
summary.BrownianInsertor .................................................. 546
summary.CodonAlphabet ...................................................... 547
summary.CodonSubst .......................................................... 548
Check if two alphabet objects have the same symbol set.

Check if two alphabet objects have the same symbol set.
Usage

```r
## S3 method for class 'Alphabet'
e1 == e2, ...
```

Arguments

- `e1`: An Alphabet object.
- `e2`: An Alphabet object.
- `...`: Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Alphabet`.

Examples

```r
# create an alphabet object
a <- Alphabet(symbols = c("0", "1"));
# clone object
b <- clone(a)
# compare the two objects
print(a == b)
# modify symbol set in b
b$symbols <- c("AT", "GC");
print(a == b)
```

Description

Check whether the two supplied Process objects are identical.

Usage

```r
## S3 method for class 'Process'
e1 == e2, ...
```
Arguments

- e1: A Process object.
- e2: A Process object.
- ...: Not used.

Value

TRUE or FALSE;

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.

Examples

```ruby
# create some Process objects
p1<Process()
p2<clone(p1)
# check object equality
p1 == p1
p1 == p2
```

---

Alphabet  

The Alphabet class

---

Description

Class representing an alphabet (a set of symbols). Package: Class Alphabet

```
  Object
~~|~~
~~*---PSRoot
~~~~~~|
~~~~~~~~*---Alphabet
```

Directly known subclasses:
AminoAcidAlphabet, AnyAlphabet, BinaryAlphabet, CodonAlphabet, NucleotideAlphabet

public static class Alphabet
extends PSRoot
### Usage

Alphabet(symbols=NA, type="Generic", ...)

### Arguments

- **symbols**: A character vector containing the symbols for the alphabet. All elements must have the same length and no duplicates are allowed. No element of the vector may contain the dash symbol, which is reserved for gaps.
- **type**: An identifier for the Alphabet object.
- ... Not used.

### Fields and Methods

**Methods:**

- `!=`
- `==`
- `as.character`
- `checkConsistency`
- `getSize`
- `getSymbolLength`
- `getSymbols`
- `getType`
- `getWriteProtected`
- `hasSymbols`
- `is`
- `isEmpty`
- `setSize`
- `setSymbolLength`
- `setSymbols`
- `setType`
- `setWriteProtected`
- `summary`

**Methods inherited from PSRoot:**

- `checkConsistency`, `enableVirtual`, `getComments`, `getMethodsList`, `globalConsistencyCheck`, `intersect.list`, `is`, `is.na`, `ll`, `my.all.equal`, `plot`, `setComments`, `setMethodsList`, `summary`, `virtualAssignmentForbidden`

**Methods inherited from Object:**

- `$`, `$<-`, `[]`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstanceTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `names`, `objectSize`, `print`, `save`

### Author(s)

- Botond Sipos, Gregory Jordan
See Also

Site Process Event BinaryAlphabet NucleotideAlphabet AminoAcidAlphabet

Examples

```r
# create an alphabet object
a<-Alphabet(type="Binary",symbols=c("0","1"));
# print summary
summary(a);
# change the identifier
a$type<-"Nucleotide";
# change the symbol set
a$symbols<-c("A","T","G","C");
# print summary again
summary(a);
# clone the alphabet object
b<-clone(a);
# test the equality of the symbol sets
a == b;
```

---

**AminoAcidAlphabet**  
*The AminoAcidAlphabet class*

---

**Description**

Class of Alphabet objects representing amino acids, using the one-letter IUPAC amino acid codes as symbol set:

<table>
<thead>
<tr>
<th>IUPAC code</th>
<th>Amino acid</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alanine</td>
</tr>
<tr>
<td>C</td>
<td>Cysteine</td>
</tr>
<tr>
<td>D</td>
<td>Aspartic Acid</td>
</tr>
<tr>
<td>E</td>
<td>Glutamic Acid</td>
</tr>
<tr>
<td>F</td>
<td>Phenylalanine</td>
</tr>
<tr>
<td>G</td>
<td>Glycine</td>
</tr>
<tr>
<td>H</td>
<td>Histidine</td>
</tr>
<tr>
<td>I</td>
<td>Isoleucine</td>
</tr>
<tr>
<td>K</td>
<td>Lysine</td>
</tr>
<tr>
<td>L</td>
<td>Leucine</td>
</tr>
<tr>
<td>M</td>
<td>Methionine</td>
</tr>
<tr>
<td>N</td>
<td>Asparagine</td>
</tr>
<tr>
<td>P</td>
<td>Proline</td>
</tr>
<tr>
<td>Q</td>
<td>Glutamine</td>
</tr>
<tr>
<td>R</td>
<td>Arginine</td>
</tr>
<tr>
<td>S</td>
<td>Serine</td>
</tr>
<tr>
<td>T</td>
<td>Threonine</td>
</tr>
</tbody>
</table>
V Valine
W Tryptophan
Y Tyrosine

Package: Class AminoAcidAlphabet

Object
~~|~~
~~|PயRoot
~~~~~~~~|~~~~~~~~
~~~~~~~~~~|~~~~~~~~~~
~~~~~~~~~~~|~~~~~~~~~~~
~~~~~~~~~~~~|~~~~~~~~~~~~
~~~~~~~~~~~~KMMpsroot
~~~~~~~~~~~~|~~~~~~~~~~~~|~~~~~~~~~~~~
~~~~~~~~~~~~KMMalphabet
~~~~~~~~~~~~|~~~~~~~~~~~~KMMaminoacidalphabet

Directly known subclasses:

public static class AminoAcidAlphabet extends Alphabet

Usage

AminoAcidAlphabet(...)

Arguments

... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [>, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save
Author(s)

Botond Sipos, Gregory Jordan

See Also

Alphabet

Examples

```r
a <- AminoAcidAlphabet()
# get object summary
summary(a)
```

Description

Sequence objects aggregating Site objects having an AminoAcidAlphabet attached by default.

Package:

Class AminoAcidSequence

```r
Object
~| ~~~--PSRoot
~~~~~~| ~~~~~~~~~--Sequence
~~~~~~~~~~~~~| ~~~~~~~~~---AminoAcidSequence
```

Directly known subclasses:

public static class **AminoAcidSequence**

extends **Sequence**

Usage

```r
AminoAcidSequence(name = NA, string = NA, length = NA, processes = NA, ancestral.obj = NA, ...)
```
**AminoAcidSequence**

**Arguments**

<table>
<thead>
<tr>
<th>name</th>
<th>Name of the Sequence object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string specifying the length and the states of the Sequence object.</td>
</tr>
<tr>
<td>length</td>
<td>The length of the sequence. Mutually exclusive with &quot;string&quot;.</td>
</tr>
<tr>
<td>processes</td>
<td>A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence.</td>
</tr>
<tr>
<td>ancestral.obj</td>
<td>The ancestral object of the Sequence object (a valid Sequence or Process object).</td>
</tr>
</tbody>
</table>

**Fields and Methods**

**Methods:**

*No methods defined.*

**Methods inherited from Sequence:**

as.character, attachProcess, checkConsistency, clearStates, clone, copySubSequence, deleteSubSequence, detachProcess, getAlphabets, getAncestral, getBigRate, getCumulativeRates, getCumulativeRatesFromRange, getDeletionTolerance, getEvents, getId, getInsertionTolerance, getLength, getName, getOmegas, getParameterAtSites, getProcesses, getRateMultipliers, getSite, getStates, getStatistics, getSymbolFreqs, getRates, getTotalRates, getTotalRatesFromRange, getUniqueAlphabets, getUniqueProcesses, getWriteProtected, insertSequence, is, plot, plotParametersAtSites, plusGamma, plusInvGamma, sampleStates, setAlphabets, setAncestral, setBigRate, setCumulativeRates, setDeletionTolerance, setName, setInsertionTolerance, setLength, setOmegas, setParameterAtSites, setProcesses, setRateMultipliers, setSite, setState, setStatistics, setString, setTotalRates, setUniqueAlphabets, setUniqueProcesses, setWriteProtected, summary

**Methods inherited from PSRoot:**

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersectList, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**

$, $<-, [], [][-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

Sequence AminoAcidAlphabet

**Examples**

```r
# create an empty AminoAcidSequence object
s <- AminoAcidSequence(length=50)
s # set states
```
Create a sequence object by specifying a string:

```r
S <- AminoAcidSequence(string="CNNGYCCNGYYY")
```

---

### Description

This is a class implementing a continuous-time Markov process acting on the state-space defined by the `AminoAcidAlphabet` class. The rate matrix of this model is completely unrestricted. The rate matrix can be built from PAML files specified by the `paml.file` argument. Alternatively, the rates can be specified as a list through the `rate.list` parameter.

#### Package:

**Class AminoAcidSubst**

```r
~|--PSRoot
   ~|--Process
      ~|--GeneralSubstitution
         ~|--AminoAcidSubst
```

#### Directly known subclasses:

- JTT
- JTT.dcmut
- LG
- MtZoa
- PAM
- PAM.dcmut
- WAG
- cpREV
- mtArt
- mtMam
- mtREV24

#### Public static class `AminoAcidSubst` extends `GeneralSubstitution`

#### Usage

```r
AminoAcidSubst(name="Anonymous", paml.file=NA, rate.list=NA, equ.dist=NA, ...)
```

#### Arguments

- `name` The name of the object.
- `paml.file` The name of the PAML file used to construct the rate matrix.
- `rate.list` A list of unscaled substitution rates (see `setRateList.GeneralSubstitution`).
- `equ.dist` Equilibrium distribution.
- `...` Additional arguments.
Fields and Methods

Methods:

buildFromPAML
checkConsistency
newAAAMatrix
setEquDist
summary

Methods inherited from GeneralSubstitution:

as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:

!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:

$, $<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hash_code, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see AminoAcidSubst.

Examples

# create an object
p<-AminoAcidSubst()
# build rate matrix from paml file
# buildFromPAML(p,"path_to_paml_file") # do not run this
# set a rate
setRate(p,"A->D",2)
# get object summary
summary(p)
The AnyAlphabet class

Description

This is a special Alphabet class which matches any alphabet. The `==`.Alphabet method always returns TRUE when one of the compared objects inherits from AnyAlphabet. This behaviour is handy when creating processes that have no alphabet preference (like a deletion process).

Package:

Class AnyAlphabet

Object

```
| ~~~~
| ~~~~PSRoot
| ~~~~~~~| 
| ~~~~~~~~~~~Alphabet
| ~~~~~~~~~~~~| 
| ~~~~~~~~~~~~~~~~AnyAlphabet
```

Directly known subclasses:

public static class AnyAlphabet
extends Alphabet

Usage

AnyAlphabet(...)

Arguments

... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden
Methods inherited from Object:
$, $<-$, [., $[<-$, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet

Examples

# create some alphabet objects
a<-BinaryAlphabet()
b<-NucleotideAlphabet()
any<-AnyAlphabet()
# compare objects
a == b
any == a
any == b

areSynonymous.CodonAlphabet

Check whether two codons are synonymous

Description
Check whether two codons are synonymous.

Usage

## S3 method for class 'CodonAlphabet'
areSynonymous(this, codons, ...) 

Arguments

this A CodonAlphabet object.
codons A vector containing two codons.
... Not used.

Value

TRUE or FALSE.
as.character.Alphabet

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see CodonAlphabet.

Examples

```r
# create a CodonAlphabet object
a<-CodonAlphabet()
areSynonymous(a,c("TCC","TCT"))
areSynonymous(a,c("TCC","CCT"))
```

Description
Get the character representation of an Alphabet object.

Usage

```r
## S3 method for class 'Alphabet'
as.character(x, ...)
```

Arguments

- `x` An Alphabet object
- `...` Not used.

Value
Returns the character vector containing the symbol set.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Alphabet.
as.character.Event

Examples

```r
# create alphabet object
a <- Alphabet(symbols=c("A", "T", "G", "C", "N"))
# get character representation
as.character(a)
```

---

as.character.Event  
*Get the character representation of an Event object*

Description

Get the character representation of an Event object.

The character representation of an Event object has the following format: "event name" ("event rate") 
<= "generator process id", like "A->T (0.3333333333333333) <= JC69:Anonymous:44780832".

Usage

```r
## S3 method for class 'Event'
as.character(x, ...)
```

Arguments

- `x`: An Event object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create a sequence and attach a process
s <- NucleotideSequence(string="ATGC", processes=list(list(JC69())))
# get the first active event from the first site
e <- s$sites[[1]]$events[[1]]
# get the character representation of e
as.character(e)
# or more simply
e
```
as.character.GeneralSubstitution

Return the character representation of a GeneralSubstitution object

Description

Return the character representation of a GeneralSubstitution object. The character representation is the object id as returned by the getId.Process method defined in the parent class.

Usage

```r
## S3 method for class 'GeneralSubstitution'
as.character(x, ...)
```

Arguments

- `x` A GeneralSubstitution object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
p <- GeneralSubstitution(name = "MySubst")
# get character representation
as.character(p)
# the same implicitly
p
```
as.character.PhyloSim  

Return the character representation of a PhyloSim object

Description

Return the character representation of a PhyloSim object.

The character representation is the identifier of the PhyloSim object as returned by the getId method.

Usage

```r
## S3 method for class 'PhyloSim'
as.character(x, ...)
```

Arguments

- `x` A PhyloSim object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
o <- PhyloSim(name="MySim");
# get character representation
as.character(o)
```
as.character.Process  

Get the character representation of a Process object

Description

Get the character representation of a Process object. The string returned is the unique Process object identifier (class name + process name + object hash).

Usage

```r
## S3 method for class 'Process'
as.character(x, ...)
```

Arguments

- `x`  
  A Process object
- `...`  
  Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Process`.

Examples

```r
# create a Process object
p<-Process()
# get character representation
x<-as.character(p)
print(x)
```
as.character.QMatrix  

Return the character representation of a QMatrix object

Description

Return the character representation of a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
as.character(x, ...)
```

Arguments

- `x`: A QMatrix object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

```r
# create a QMatrix object
m<-QMatrix()
# get the character representation
as.character(m)
# the same, but implicitly
m
```
as.character.Sequence  
Get the string representation of a Sequence object

Description

Get the string representation of a Sequence object. The string representation is the concatenation of the states of the aggregated Site object. Undefined states (NA-s) are represented by question marks.

Usage

```r
## S3 method for class 'Sequence'
as.character(x, ...)
```

Arguments

- `x`  
  A Sequence object.
- `...`  
  Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create object
s <- Sequence(length=10)
# get character representation
as.character(s)
```
as.character.Site

Get the character representation of a Site object

Description

Get the character representation of a Site object.

Usage

```r
## S3 method for class 'Site'
as.character(x, ...)
```

Arguments

- `x` A Site object.
- `...` Not used.

Value

A character vector of length one containing the current state.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `site`.

Examples

```r
# create site object
s <- Site(alphabet = NucleotideAlphabet(), state = "A")
# get character representation
x <- as.character(s)
```

as.character.ToleranceSubstitution

*Return the character representation of a ToleranceSubstitution object*

**Description**

Return the character representation of a ToleranceSubstitution object. The character representation is the object id as returned by the `getid.Process` method defined in the parent class.

**Usage**

```r
## S3 method for class 'ToleranceSubstitution'
as.character(x, ...)
```

**Arguments**

- `x` A ToleranceSubstitution object.
- `...` Not used.

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ToleranceSubstitution`.

**Examples**

```r
# create a ToleranceSubstitution object
p <- ToleranceSubstitution(name="MySubst")
# get character representation
as.character(p)
# the same implicitly
p
```
attachHookToNode.PhyloSim

Attach a callback function to a given node of a phylo object aggregated by a PhyloSim object

Description

Attach a callback function to a given node of a phylo object aggregated by a PhyloSim object.

A "node hook" is a function which accepts a Sequence object through the named argument "seq" and returns a Sequence object. The node hook function must accept any object which inherits from the Sequence class!

After simulating the branch leading to the node, the resulting Sequence object is passed to the node hook and the returned object is used to simulate the downstream branches.

By using node hooks the attached processes can be replaced during simulation, hence enabling the simulation of non-homogeneous sequence evolution.

Usage

```r
## S3 method for class 'PhyloSim'
attachHookToNode(this, node=NA, fun=NA, ...)
```

Arguments

- `this`: A PhyloSim object.
- `node`: Node identifier.
- `fun`: A function (see above).
- `...`: Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.  

Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoral(3),
  root.seq=NucleotideSequence(string="ATGC", processes=list(JC69())))

# create a node hook function
hook<-function(seq=NA){
  # replace the substitution process with F84
  if(inherits(seq,"NucleotideSequence")){
    cat("Replacing JC69 with F84.\n");
    seq$processes<-list(F84(rate.params=list("Kappa" = 2))));
  }
  return(seq);
}
# attach hook function to node 5
attachHookToNode(sim, 5, hook);
# Run the simulation
Simulate(sim);
# Check if the processes have been truly replaced
lapply(sim$sequences, getUniqueProcesses.Sequence)
# Print the resulting alignment
sim$alignment

attachProcess.Sequence

Attach a Process object to a set of Site objects aggregated by a Sequence object.

Description

Attach a Process object to a set of Site objects aggregated by a Sequence object.

Usage

## S3 method for class 'Sequence'
attachProcess(this, process, index, ...)

Arguments

this A Sequence object.
process A Process object.
index An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
... Not used.
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

```r
# create a Sequence object of length 6
s <- Sequence(length=10, alphabets=list(NucleotideAlphabet()))
# attach a JC69 substitution process
attachProcess(s, JC69())
# get the list of attached processes
$s$processes
# attach the GTR substitution process to range 3:6
attachProcess(s, GTR(), 3:6)
# get the list of attached processes
$s$processes
```

---

attachProcess.Site  
*Attach a Process object to a Site object*

---

Description

Attach a Process object to a Site object.

The Alphabet objects associated with the Site and Process objects must have the same symbol set, or at least one of them should inherit from the class AnyAlphabet.

During the attachment, the site-process specific parameter templates are copied from the Process object and stored in the Site object. The Process objects are marked as write protected if the attachment was successful.

Usage

```r
## S3 method for class 'Site'
attachProcess(this, process, ...)
```

Arguments

- `this` A Site object.
- `process` A Process object.
- `...` Not used.
attachSeqToNode.PhyloSim

Value

The Site object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process detachProcess setProcesses getProcesses isAttached

Examples

# create a Site object and the associated NucleotideAlphabet object
s<-Site(alphabet=NucleotideAlphabet())
# create a K80 substitution process
p<-K80()
# attach p to s
attachProcess(s,p)
# get the list of attached processes
s$processes
# check write protection for p
p$writeProtected

attachSeqToNode.PhyloSim

Associate a Sequence object with a given node of a phylo object aggregated by a PhyloSim object

Description

Associate a Sequence object with a given node of a phylo object aggregated by a PhyloSim object. This method is mainly used internally.

Usage

## S3 method for class 'PhyloSim'
attachSeqToNode(this, node=NA, seq=NA, ...)

Arguments

this       A PhyloSim object.
node       Node identifier.
seq        A Sequence object.
...        Not used.
Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

---

**BinaryAlphabet**  
*The BinaryAlphabet class*

---

**Description**

Class of Alphabet objects with the c("0","1") symbol set.

Package:  
Class BinaryAlphabet

```
Object
~~|  
~~+-PSRoot
~~~~~~|  
~~~~~~~~+-Alphabet
~~~~~~~~~~|  
~~~~~~~~~~~~+-BinaryAlphabet
```

**Directly known subclasses:**

```
public static class BinaryAlphabet  
extends Alphabet
```

**Usage**

`BinaryAlphabet(...)`

**Arguments**

```
...  
Not used.
```
Fields and Methods

Methods:
No methods defined.

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [., [_[., as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet

Examples

# create a binary alphabet
b<-BinaryAlphabet()
# get alphabet summary
summary(b)

The BinarySequence class

Description
Sequence objects aggregating Site objects having a BinaryAlphabet attached by default.

Package:

Class BinarySequence

Object
~~|
~~+++PSRoot
~~~~~~|
~~~~~~~~~~+Sequence
~~~~~~~~~~~~|


**BinarySequence**

~~~~~~~~~~~~~~+---BinarySequence

**Directly known subclasses:**

public static class **BinarySequence** extends **Sequence**

**Usage**

BinarySequence(name=NA, string=NA, length=NA, processes=NA, ancestral.obj=NA, ...)

**Arguments**

- **name**  
  Name of the Sequence object.

- **string**  
  A string specifying the length and the states of the Sequence object.

- **length**  
  The length of the sequence. Mutually exclusive with "string".

- **processes**  
  A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence.

- **ancestral.obj**  
  The ancestral object of the Sequence object (a valid Sequence or Process object). Not used.

**Fields and Methods**

**Methods:**

No methods defined.

**Methods inherited from Sequence:**

as.character, attachProcess, checkConsistency, clearStates, clone, copySubSequence, deleteSubSequence, detachProcess, getAlphabets, getAncestral, getBigRate, getCumulativeRates, getCumulativeRatesFromRange, getDeletionTolerance, getEvents, getId, getInsertionTolerance, getLength, getName, getOmegas, getParameterAtSites, getProcesses, getRateMultipliers, getStates, getString, getSymbolFreqs, getTotalRates, getTotalRatesFromRange, getUniqueAlphabets, getUniqueProcesses, getWriteProtected, insertSequence, is, plot, plotParametersAtSites, plusGamma, plusInvGamma, sampleStates, setAlphabets, setAncestral, setBigRate, setCumulativeRates, setDeletionTolerance, setId, setInsertionTolerance, setName, setLength, setOmegas, setParameterAtSites, setProcesses, setRateMultipliers, setStates, setString, setTotalRates, setUniqueAlphabets, setUniqueProcesses, setWriteProtected, summary

**Methods inherited from PSRoot:**

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**

$, $<-, [, [<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save
Author(s)
Botond Sipos, Gregory Jordan

See Also
Sequence BinaryAlphabet

Examples

```r
# create an empty BinarySequence object
s <- BinarySequence(length=50)

# set states
s$states <- c(0, 0, 1, 0, 1, 1)

# create a sequence object by specifying a string
s <- BinarySequence(string="0000110010001111")
```

BinarySubst

The BinarySubst class

Description

This is a class implementing a continuous-time Markov process acting on the state space defined by the BinaryAlphabet class.

Package:
Class BinarySubst

Object

```text
|
-> PSRoot

|
-> Process

|
-> GeneralSubstitution

|
-> BinarySubst
```

Directly known subclasses:

public static class BinarySubst
extends GeneralSubstitution
BinarySubst

Usage

BinarySubst(name="Anonymous", rate.list=NA, equ.dist=NA, ...)

Arguments

name
   The name of the object.
rate.list
   A list of substitution rates (see setRateList.GeneralSubstitution).
equ.dist
   Equilibrium distribution.
...  
   Additional arguments.

Fields and Methods

Methods:

   checkConsistency  -
   summary            -

Methods inherited from GeneralSubstitution:
   as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
   !, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
   checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
   $, $<-, [, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralSubstitution GTR WAG
Examples

```r
# The following code demonstrates
# the use of the BinarySubst process
# during a simulation.
p <- BinarySubst(rate=0.25, name="Binary", rate.list=ist("0->1"=2,"1->0"=1))
# create a sequence object, attach process p
s <- BinarySequence(string="000000000000000", processes=list(list(p)));
# make the first five positions invariant
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
generateRateMultipliers(s,p)
# simulate
sim <- PhyloSim(root=seq,p,phylo=rcml(3))
Simulate(sim)
# print alignment
sim$alignment
```

---

**BrownianInsertor**

The BrownianInsertor class inherits from the DiscreteInsertor or ContinuousInsertor class depending on the type constructor argument ("discrete" or "continuous").

This process generates the insert sequence based on the sites flanking the insertions as follows:

- An insert length is sampled by calling the function stored in the `proposeBy` virtual field.
- A sequence object is constructed.
- The processes attached to both flanking sites are attached to the insert sequence. If there are no common processes, the processes from a randomly chosen site will be attached to the insert.
- The site-process specific parameters are sampled from Brownian paths with linear trends having the values from the flanking sites as endpoints.

The "noisiness" of the Brownian path can be controlled through the `scale` virtual field/constructor parameter.

**Description**

The BrownianInsertor class inherits from the DiscreteInsertor or ContinuousInsertor class depending on the type constructor argument ("discrete" or "continuous").

This process generates the insert sequence based on the sites flanking the insertions as follows:

- An insert length is sampled by calling the function stored in the `proposeBy` virtual field.
- A sequence object is constructed.
- The processes attached to both flanking sites are attached to the insert sequence. If there are no common processes, the processes from a randomly chosen site will be attached to the insert.
- The site-process specific parameters are sampled from Brownian paths with linear trends having the values from the flanking sites as endpoints.

The "noisiness" of the Brownian path can be controlled through the `scale` virtual field/constructor parameter.

**Package:**

**Class** BrownianInsertor

```
Object
| ~~~| ~~~--PSRoot
| ~~~~~| ~~~~~--Process
| ~~~~~~~~~| ~~~~~~~~~--GeneralInDel
```
BrownianInsertor

Directly known subclasses:

public static class BrownianInsertor
extends DiscreteInsertor

Usage
BrownianInsertor(name="Anonymous", type="discrete", scale=0.001, ...)

Arguments
name Object name.
type Process type (see above).
scale Brownian path scale parameter.
... Additional arguments.

Fields and Methods

Methods:
BrownianPath -
checkConsistency -
getScale -
getType -
setScale -
setType -
summary -

Methods inherited from DiscreteInsertor:
checkConsistency, getProbs, getSizes, plot, setProbs, setSizes, summary

Methods inherited from GeneralInsertor:
checkConsistency, generateInsert, getAcceptWin, getEventsAtSite, getGenerateBy, getInsertHook, getTemplateSeq, is, setAcceptWin, setGenerateBy, setInsertHook, setTemplateSeq, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary
Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [[, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
DiscreteInsertor ContinuousInsertor GeneralInsertor GeneralInDel

Examples

```r
# create a BrownianInsertor process, discrete type
p <- BrownianInsertor(
  type = "discrete",
  scale = 0.05,
  sizes = 1:4,
  probs = c(3/6,1/6,1/6,1/6),
  rate = 0.05
)

# get object summary
summary(p)
# plot insert length distribution
plot(p)
# create a nucleotide sequence, attach processes
s <- NucleotideSequence(string = "AAAAAAAAAAA", processes = list(list(p, JC69())))
# create simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
# simulate and show alignment
Simulate(sim)
sim$alignment
# check the rate multipliers and insertion tolerances in one of the sequences
res <- sim$sequences[[2]]
getRateMultipliers(res, p)
getInsertionTolerance(res, p)
```
Description

Generate a Brownian path.

This method generates a Brownian path given the scale parameter a (determining "noisiness") and the vector p describing the trends. More useful as a static method.

Usage

```r
## S3 method for class 'BrownianInsertor'
BrownianPath(this, p=NA, a=NA, ...)
```

Arguments

- `this`: A BrownianInsertor object.
- `p`: Path parameter (a numeric vector).
- `a`: Scale parameter (a numeric vector of length one).
- `...`: Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `BrownianInsertor`.

Examples

```r
path<-BrownianInsertor$BrownianPath(a=2,p=1:10);
path
plot(path)
```
buildFromPAML.AminoAcidSubst

Build rate matrix from PAML file

Description
Build rate matrix from PAML file.

Usage

## S3 method for class 'AminoAcidSubst'
buildFromPAML(this, paml.file, ...)

Arguments

this      An AminoAcidSubst object.
paml.file Path to the PAML file.
...      Not used.

Value

The AminoAcidSubst object (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see AminoAcidSubst.

Examples

# create an object
p<-AminoAcidSubst()
# build rate matrix from paml file
# buildFromPAML(p,"path_to_paml_file") # do not run this
# get object summary
summary(p)
buildFromPAML.CodonSubst

Build rate matrix from PAML file

Description

Build rate matrix from PAML file.

Usage

## S3 method for class 'CodonSubst'
buildFromPAML(this, paml.file, ...)

Arguments

- this: A CodonSubst object.
- paml.file: Path to the PAML file.
- ...: Not used.

Value

The CodonSubst object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSubst.

Examples

# create an object
p<-CodonSubst()
# build rate matrix from paml file
# buildFromPAML(p,"path_to_paml_file") # do not run this
# get object summary
summary(p)
checkConsistency.Alphabet

*Check object consistency*

### Description

Check object consistency.

### Usage

```r
## S3 method for class 'Alphabet'
checkConsistency(this, ...)
```

### Arguments

- **this**: An object.
- **...**: Not used.

### Value

Returns an invisible TRUE if no inconsistencies found, throws an error otherwise.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `alphabet`.

### Examples

```r
# create an alphabet object
a <- Alphabet(symbols = c(0, 1));
# check consistency
print(checkConsistency(a));
# mess up with the internals
a$symbols[1] <- "BAD";
# NOT run: consistency check now will throw an error
## Not run: print(checkConsistency(a));
```
checkConsistency.AminoAcidSubst

Check object consistency

Description

Check object consistency.

Usage

```r
## S3 method for class 'AminoAcidSubst'
checkConsistency(this, ...)
```

Arguments

- `this` : An object.
- `...` : Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `AminoAcidSubst`.

checkConsistency.BinarySubst

Check object consistency

Description

Check object consistency.

Usage

```r
## S3 method for class 'BinarySubst'
checkConsistency(this, ...)
```
checkConsistency.BrownianInsertor

Arguments

  this  An object.
  ...  Not used.

Value

  Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see BinarySubst.

---

checkConsistency.BrownianInsertor

*Check object consistency*

---

Description

  Check object consistency.

Usage

  ```r
  ## S3 method for class 'BrownianInsertor'
  checkConsistency(this, ...)
  ```

Arguments

  this  An object.
  ...  Not used.

Value

  Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see BrownianInsertor.
checkConsistency.CodonAlphabet

Description
Check object consistency.

Usage
## S3 method for class 'CodonAlphabet'
checkConsistency(this, ...)

Arguments
this An object.
... Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see CodonAlphabet.

checkConsistency.CodonSequence

Description
Check object consistency.

Usage
## S3 method for class 'CodonSequence'
checkConsistency(this, ...)

checkConsistency.CodonSubst

Arguments

- **this**: An object.
- **...**: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSequence.
**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'CodonUNREST'
checkConsistency(this, ...)
```

**Arguments**

- `this` An object.
- `...` Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonUNREST`.

---

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'ContinuousDeletor'
checkConsistency(this, ...)
```
checkConsistency.ContinuousInsertor

Arguments

this

An object.

...  Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ContinuousDeleter.

---

checkConsistency.ContinuousInsertor

*Check object consistency*

---

Description

Check object consistency.

Usage

```r
## S3 method for class 'ContinuousInsertor'
checkConsistency(this, ...)
```

Arguments

this

An object.

...  Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ContinuousInsertor.
checkConsistency.DiscreteDeletor

Check object consistency

Description
Check object consistency.

Usage
```r
## S3 method for class 'DiscreteDeletor'
checkConsistency(this, ...)
```

Arguments
- `this`: An object.
- `...`: Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `DiscreteDeletor`.

checkConsistency.DiscreteInsertor

Check object consistency

Description
Check object consistency.

Usage
```r
## S3 method for class 'DiscreteInsertor'
checkConsistency(this, ...)
```
checkConsistency.Event

Arguments

this  An object.
...  Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see DiscreteInsertor.

checkConsistency.Event

Check object consistency

Description

Check object consistency.

Usage

## S3 method for class 'Event'
checkConsistency(this, ...)

Arguments

this  An object.
...  Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.
Description
Check object consistency.

Usage

```r
## S3 method for class 'F81'
checkConsistency(this, ...)
```

Arguments

- `this` An object.
- `...` Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see F81.
Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

---

**checkConsistency**

Description

Check object consistency.

Usage

```r
## S3 method for class 'FastFieldDeletor'
checkConsistency(this, ...)
```

Arguments

- `this` An object.
- `...` Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.
checkConsistency. GeneralDeletor

Description
Check object consistency.

Usage
## S3 method for class 'GeneralDeletor'
checkConsistency(this, ...)

Arguments
- **this**: An object.
- **...**: Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralDeletor.

checkConsistency. GeneralInDel

Description
Check object consistency.

Usage
## S3 method for class 'GeneralInDel'
checkConsistency(this, ...)

---

**checkConsistency. GeneralDeletor**

*Check object consistency*

**Description**
Check object consistency.

**Usage**
## S3 method for class 'GeneralDeletor'
checkConsistency(this, ...)

**Arguments**
- **this**: An object.
- **...**: Not used.

**Value**
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
For more information see GeneralDeletor.

---

**checkConsistency. GeneralInDel**

*Check object consistency*

**Description**
Check object consistency.

**Usage**
## S3 method for class 'GeneralInDel'
checkConsistency(this, ...)

---
checkConsistency::GeneralInsertor

Arguments

- **this**: An object.
- **...**: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralInDel.
checkConsistency-GeneralSubstitution

Description
Check object consistency.

Usage
```r
## S3 method for class 'GeneralSubstitution'
checkConsistency(this, ...)
```

Arguments
- `this`: An object.
- `...`: Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `GeneralSubstitution`.

checkConsistency-GTR

Description
Check object consistency.

Usage
```r
## S3 method for class 'GTR'
checkConsistency(this, ...)
```

Arguments
- `this`: An object.
- `...`: Not used.
Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GTR.

---

**checkConsistency.GY94**  
*Check object consistency*

Description

Check object consistency.

Usage

```r
## S3 method for class 'GY94'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GY94.
Description
Check object consistency.

Usage
## S3 method for class 'HKY'
checkConsistency(this, ...)

Arguments
this An object.
... Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see HKY.

Description
Check object consistency.

Usage
## S3 method for class 'JC69'
checkConsistency(this, ...)

Arguments
this An object.
... Not used.
Check object consistency

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'K80'
checkConsistency(this, ...)
```

**Arguments**

- `this` An object.
- `...` Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `JC69`.

---

```r
checkConsistency.K80  Check object consistency
```

---

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `K80`. 
Description
Check object consistency.

Usage
```r
# S3 method for class 'K81'
checkConsistency(this, ...)
```

Arguments
- `this` An object.
- `...` Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `K81`.

Description
Check object consistency.

Usage
```r
# S3 method for class 'PhyloSim'
checkConsistency(this, ...)
```

Arguments
- `this` An object.
- `...` Not used.
**checkConsistency.Process**

*Check object consistency*

**Description**

Check object consistency.

**Usage**

```r
## S3 method for class 'Process'
checkConsistency(this, ...)  
```

**Arguments**

- `this` An object.
- `...` Not used.

**Value**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

---

**checkConsistency.Process**

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.
checkConsistency.PSRoot

Check object consistency

Description

Check object consistency. The consistency check is not implemented in plain PSRoot objects, the method prints out a warning about that.

Usage

```r
## S3 method for class 'PSRoot'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRoot.

checkConsistency.PSRootSummary

Check object consistency

Description

Check object consistency. The consistency check is not implemented for PSRootSummary objects, the method prints out a warning about that.

Usage

```r
## S3 method for class 'PSRootSummary'
checkConsistency(this, ...)
```
Arguments
this An object.
... Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PSRootSummary.

---

checkConsistency.QMatrix

*Check object consistency*

Description
Check object consistency.

Usage
```r
## S3 method for class 'QMatrix'
checkConsistency(this, check.process=TRUE, ...)
```

Arguments
- **this**: An object.
- **check.process**: Check the associated process object.
- ... Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see QMatrix.
checkConsistency.Sequence

Description
Check object consistency.

Usage
## S3 method for class 'Sequence'
checkConsistency(this, omit.sites=FALSE, ...)

Arguments
- **this**: An object.
- **omit.sites**: Do not check aggregated site objects.
- **...**: Not used.

Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Sequence.

checkConsistency.Site

Description
Check object consistency.

Usage
## S3 method for class 'Site'
checkConsistency(this, ...)
checkConsistency

Arguments

this       An object.
...        Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

checkConsistency

Check object consistency

Description

Check object consistency.

Usage

## S3 method for class 'T92'
checkConsistency(this, ...)

Arguments

this       An object.
...        Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see T92.
Description

Check object consistency.

Usage

```r
## S3 method for class 'TN93'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `TN93`.

Description

Check object consistency.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
checkConsistency(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.
Value
Returns an invisible TRUE if no inconsistencies found in the object, throws an error otherwise.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see ToleranceSubstitution.
clearStates.Sequence  

Set the states of a collection of Site objects aggregated by a Sequence object to undefined (NA)

Description

Set the states of a collection of Site objects aggregated by a Sequence object to undefined (NA).

Usage

```r
## S3 method for class 'Sequence'
clearStates(this, index, ...)
```

Arguments

- `this`: A Sequence object.
- `index`: An integer vector specifying a set of positions. It is set to `1:seq$length` if omitted.
- `...`: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a nucleotide sequence
s<-NucleotideSequence(string="ATGC")
s
# set states to NA in the range 2:3
clearStates(s,2:3)
s
# set all states to NA
clearStates(s)
s
```
clone.Generalsubstitution

Clone a GeneralSubstitution object

Description

Clone a GeneralSubstitution object.
This method also clones the aggregated QMatrix object, but not the aggregated Alphabet object, as that is a good target for recycling.

Usage

```r
## S3 method for class 'GeneralSubstitution'
clone(this, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `...`: Not used.

Value

A GeneralSubstitution object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Generalsubstitution`.

Examples

```r
# create a Generalsubstitution object
p <- Generalsubstitution(
  alphabet = BinaryAlphabet(),
  rate.list = list("0->1"=1,"1->0"=2),
  name = "MyBinary"
)

# clone p
pp <- clone(p)
# do some checks
p; pp
p == p
p == pp
equals(p$qMatrix, pp$qMatrix)
```
**clone.Process**

**Clone a process object**

**Description**
Clone a process object. Write protection is set to FALSE for the new Process object.

**Usage**

```r
## S3 method for class 'Process'
clone(this, ...)
```

**Arguments**

- `this`: A Process object.
- `...`: Not used.

**Value**

A Process object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create a process object
p <- Process()
# clone process object
pp <- clone(p)
# check identity
p == pp
```
clone.Sequence

Clone a Sequence object

Description

Clone a Sequence object.

The cloning of Sequence objects involves the cloning of all aggregated Site objects. Because of
that the cloning of long sequences is quite expensive. The cloned Site objects have the orginal Site
objects as ancestral. The new Sequence objects has the original object as ancestral.

Usage

```r
## S3 method for class 'Sequence'
clone(this, ...)
```

Arguments

- `this`: A Sequence object.
- `...`: Not used.

Value

A Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

Sequence clone.Object

Examples

```r
# create a nucleotide sequence
s <- NucleotideSequence(string = "ATG")
# clone the sequence
cs <- clone(s)
# get some properties
equals(s, s)
equals(s, cs)
cs$ancestral
cs$sites[[1]]$ancestral
```
clone.ToleranceSubstitution

Clone a ToleranceSubstitution object

Description

Clone a ToleranceSubstitution object.

This method also clones the aggregated QMatrix object, but not the aggregated Alphabet object, as that is a good target for recycling.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
clone(this, ...)
```

Arguments

- `this`: A ToleranceSubstitution object.
- `...`: Not used.

Value

A ToleranceSubstitution object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ToleranceSubstitution`.

Examples

```r
# create a ToleranceSubstitution object
p <- ToleranceSubstitution(
  alphabet = BinaryAlphabet(),
  rate.list = list("0->1" = 1, "1->0" = 2),
  name = "MyBinary"
)

# clone p
pp <- clone(p)
# do some checks
p; pp
p == p
p == pp
equals(p$qMatrix, pp$qMatrix)
```
The CodonAlphabet class

Description

This class implements codon alphabets and handles the translation of codons to their corresponding amino acids. Stop codons are excluded from the symbol set, thus the symbol set depends on the genetic code table. The genetic code table can be specified through the `table.id` constructor parameter.

The available genetic code tables:

1 Standard
2 Vertebrate Mitochondrial
3 Yeast Mitochondrial
4 Mold, Protozoan, and Coelenterate Mitochondrial and Mycoplasma/Spiroplasma
5 Invertebrate Mitochondrial
6 Ciliate, Dasycladacean and Hexamita Nuclear
7 Echinoderm Mitochondrial
8 Euplotid Nuclear
9 Bacterial
10 Alternative Yeast Nuclear
11 Ascidian Mitochondrial
12 Flatworm Mitochondrial
13 Blepharisma Nuclear
14 Chlorophycean Mitochondrial
15 Trematode Mitochondrial
16 Scenedesmus obliquus Mitochondrial
17 Thraustochytrium Mitochondrial

Package:
Class CodonAlphabet

Directly known subclasses:

public static class CodonAlphabet
extends Alphabet
Usage

CodonAlphabet(table.id=1, ...)

Arguments

table.id The identifier of the genetic code table.

... Not used.

Fields and Methods

Methods:

areSynonymous -
checkConsistency -
getTableId -
getTransTable -
is -
isStartCodon -
isStopCodon -
setTableId -
setTransTable -
summary -
translateCodon -

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldName, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonAlphabet.
### Examples

```r
# create a CodonAlphabet object
a <- CodonAlphabet(table.id=2)
# get object summary
summary(a)
```

---

**CodonSequence**

*The CodonSequence class*

---

**Description**

Sequence objects aggregating Site objects having a CodonAlphabet attached by default.

**Package:**

**Class CodonSequence**

**Object**

```r
~~|   
~~+-PSRoot
~~~~/|   
~~~~~~+-Sequence
~~~~~~~|   
~~~~~~~~+-CodonSequence
```

**Directly known subclasses:**

- `public static class CodonSequence`
- **extends** `Sequence`

**Usage**

```r
CodonSequence(name=NA, string=NA, length=NA, table.id=1, processes=NA, ancestral.obj=NA, 
...)
```

**Arguments**

- **name**
  - Name of the Sequence object.
- **string**
  - A string specifying the length and the states of the Sequence object.
- **length**
  - The length of the sequence. Mutually exclusive with "string".
- **table.id**
  - The genetic code table to use in the attached CodonAlphabet object ("Standard" by default).
- **processes**
  - A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence.
ancestral.obj  The ancestral object of the Sequence object (a valid Sequence or Process object).
...  Not used.

Fields and Methods

Methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Inherited From</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translate</td>
<td>-</td>
</tr>
<tr>
<td>checkConsistency</td>
<td>-</td>
</tr>
<tr>
<td>getOmegas</td>
<td>-</td>
</tr>
<tr>
<td>omegaHist</td>
<td>-</td>
</tr>
<tr>
<td>omegaVarM0</td>
<td>-</td>
</tr>
<tr>
<td>omegaVarM1</td>
<td>-</td>
</tr>
<tr>
<td>omegaVarM2</td>
<td>-</td>
</tr>
<tr>
<td>omegaVarM3</td>
<td>-</td>
</tr>
<tr>
<td>omegaVarM4</td>
<td>-</td>
</tr>
<tr>
<td>setOmegas</td>
<td>-</td>
</tr>
</tbody>
</table>

Methods inherited from Sequence:

as.character, attachProcess, checkConsistency, clearStates, clone, copySubSequence, deleteSubSequence, detachProcess, getAlphabets, getAncestral, getBigRate, getCumulativeRates, getCumulativeRatesFromRange, getDeletionTolerance, getEvents, getId, getInsertionTolerance, getLength, getName, getOmegas, getParameterAtSites, getProcesses, getRateMultipliers, getStates, getString, getSymbolFreqs, getTotalRates, getTotalRatesFromRange, getUniqueAlphabets, getUniqueProcesses, getWriteProtected, insertSequence, is, plot, plotParametersAtSites, plusGamma, plusInvGamma, sampleStates, setAlphabets, setAncestral, setBigRate, setCumulativeRates, setDeletionTolerance, setId, setInsertionTolerance, setLength, setName, setOmegas, setParameterAtSites, setProcesses, setRateMultipliers, setStates, setString, setTotalRates, setUniqueAlphabets, setUniqueProcesses, setWriteProtected, summary

Methods inherited from PSRoot:

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:

$, $<-, [], [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSequence.
Examples

```r
# create an empty CodonSequence object
s <- CodonSequence(length=50)
s
# set states
s$states <- c("ATG","CGA","TTT","CTA")
s
# create a sequence object by specifying a string
s <- CodonSequence(string="ATCTTCGAATG")
s
CodonSubst

<table>
<thead>
<tr>
<th>The CodonSubst class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Description

This is a class implementing a continuous-time Markov process acting on the state-space defined by the CodonAlphabet class.

The rate matrix can be built from PAML files specified by the `paml.file` argument. Alternatively the rates can be specified as a list through the `rate.list` parameter.

Package:

Class CodonSubst

```
Object
```
```
|~~|PSRoot
```
```
|~~~~~~|Process
```
```
|~~~~~~~~~|GeneralSubstitution
```
```
|~~~~~~~~~~~~~~~~|CodonSubst
```

Directly known subclasses:
ECMrest, ECMunrest

public static class **CodonSubst**
extends GeneralSubstitution

Usage

```r
CodonSubst(name="Anonymous", paml.file=NA, rate.list=NA, equ.dist=NA, ...)
```
Arguments

- **name**: The name of the object.
- **paml.file**: The name of the PAML file used to construct the rate matrix.
- **rate.list**: A list of unscaled substitution rates (see `setRateList.Generalsubstitution`).
- **equ.dist**: Equilibrium distribution.
- **...**: Additional arguments.

Fields and Methods

- **Methods**:
  - `buildFromPAML`
  - `checkConsistency`
  - `newMatrix`
  - `setEquDist`
  - `summary`

- **Methods inherited from Generalsubstitution**:
  - `as.character`, `checkConsistency`, `clone`, `getAlphabet`, `getEquDist`, `getEventRateAtSite`, `getEventsAtSite`, `getQMatrix`, `getRate`, `getRateList`, `hasUndefinedRate`, `is`, `plot`, `rescaleQMatrix`, `sampleState`, `setAlphabet`, `setEquDist`, `setQMatrix`, `setRate`, `setRateList`, `summary`

- **Methods inherited from Process**:
  - `!=`, `==`, `as.character`, `checkConsistency`, `clone`, `getAlphabet`, `getEventsAtSite`, `getId`, `getName`, `getEventRateAtSite`, `getSiteSpecificParamIds`, `getSiteSpecificParamList`, `getWriteProtected`, `hasSiteSpecificParameter`, `hasUndefinedRate`, `is`, `setAlphabet`, `setId`, `setName`, `setParameterAtSite`, `setSiteSpecificParamIds`, `setSiteSpecificParamList`, `setWriteProtected`, `summary`

- **Methods inherited from PSRoot**:
  - `checkConsistency`, `enableVirtual`, `getComments`, `getMethodsList`, `globalConsistencyCheck`, `intersect.list`, `is`, `is.na`, `ll`, `my.all.equal`, `plot`, `setComments`, `setMethodsList`, `summary`, `virtualAssignmentForbidden`

- **Methods inherited from Object**:
  - `S`, `$<-`, `[]`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clearLookupCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `getEnvironment`, `getFieldModifier`, `getFieldModifiers`, `getFields`, `getInstanceTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `names`, `objectSize`, `print`, `save`

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonSubst`. 
Examples

```c
# create an object
p<-CodonSubst()
# build rate matrix from paml file
# buildFromPAML(p, "path_to_paml_file") # do not run this
# get object summary
summary(p)
```

---

**CodonUNREST**

The CodonUNREST class

**Description**

This class implements a time-continuous Markov process acting on a state space defined by the symbol set of a CodonAlphabet object. The rate matrix of this model is unrestricted, so it can be used to implement empirical codon models or more restricted mechanistic models.

Package:

Class **CodonUNREST**

```
Object

|-- PSRoot
|   |-- Process
|       |-- GeneralSubstitution
|           |-- CodonUNREST
```

**Directly known subclasses:**

GY94

Public static class **CodonUNREST** extends **GeneralSubstitution**

**Usage**

`CodonUNREST(name="Anonymous", table.id=1, rate.list=NA, equ.dist=NA, ...)`

**Arguments**

- `name` The name of the object.
- `table.id` The identifier of the genetic code table (see CodonAlphabet).
- `rate.list` A list of unscaled substitution rates (see `setRateList.GeneraSubstitution`).
equ. dist  Equilibrium distribution.

...  Additional arguments.

Fields and Methods

Methods:

- checkConsistency
- is

Methods inherited from GeneralSubstitution:
- as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
- !=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
- checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
- $, $<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralSubstitution GY94

Examples

```r
# create a CodonUNREST object
p <- CodonUNREST(table.id=2)
# get object summary
summary(p)
```
ContinuousDeletor

The ContinuousDeletor class

Description

This class implements a process which performs deletions with lengths sampled from a user-specified R expression returning a numeric value. See GeneralDeletor for the how the deletion processes works.

Package:

Class ContinuousDeletor

Object
~~|~~
~~~|~~~PSRoot
~~~~~~|~~~~~~Process
~~~~~~~~~|~~~~~~~~~GeneralIndel
~~~~~~~~~~~~~|~~~~~~~~~~~~GeneralDeletor
~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~ContinuousDeletor

Directly known subclasses:

public static class ContinuousDeletor
extends GeneralDeletor

Usage

ContinuousDeletor(name="Anonymous", rate=NA, dist=NA, max.length=NA, ...)

Arguments

name The name of the object.
rate The general rate.
dist The length sampling expression.
max.length Maximum event length.
... Additional arguments.

Fields and Methods

Methods:
checkConsistency -
getDist -
getMaxLength -
plot -
setDist -
setMaxLength -
summary -

Methods inherited from GeneralDeletor:
checkConsistency, getEventsAtSite, is, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, setRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [., [.[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldName, getFields, getInitializationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralDeletor DiscreteDeletor GeneralInDel

Examples

# create a ContinuousDeletor process
C <- ContinuousDeletor(
  name = 'Conty',
  rate = 0.25,
  dist = expression(1),
  max.length = 2
)
# get object summary
summary(C)
# set/get length sampling expression

ContinuousInsertor

The ContinuousInsertor class

Description

This class implements a process which performs insertions with lengths sampled from a user-specified R expression returning a numeric value. See GeneralInsertor for the how the insertion process works.

Package:
Class ContinuousInsertor

Object
~~|
~~~PSRoot
~~~~~|
~~~~~~~~~Process
~~~~~~~~~~~|
~~~~~~~~~~~~~GeneralIndel
~~~~~~~~~~~~~~~|
~~~~~~~~~~~~~~~GeneralInsertor
~~~~~~~~~~~~~~~~~~~~|
~~~~~~~~~~~~~~~~~~~~ContinuousInsertor
Directly known subclasses:

public static class ContinuousInsertor
   extends GeneralInsertor

Usage

ContinuousInsertor(name="Anonymous", rate=NA, dist=NA, max.length=NA, ...)

Arguments
	name        The name of the object.
trate       The general rate.

dist        The length sampling expression.

max.length  Maximum event length.

...         Additional arguments.

Fields and Methods

Methods:

   checkConsistency -
   getDist         -
   getMaxLength    -
   plot            -
   setDist         -
   setMaxLength    -
   summary         -

Methods inherited from GeneralInsertor:
checkConsistency, generateInsert, getAcceptWin, getEventsAtSite, getGenerateBy, getInsertHook, getTemplateSeq, is, setAcceptWin, setGenerateBy, setInsertHook, setTemplateSeq, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, inter-
Methods inherited from Object:
S, S<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldName, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralInsertor DiscreteInsertor GeneralInDel

Examples

```r
# create a ContinuousInsertor process
o<-ContinuousInsertor(  
    name="Conty",  
    rate=0.1,  
    dist=expression(1),  
    max.length=2  
)

# set template sequence
o$templateSeq<-NucleotideSequence(string="CAC")

# get object summary
summary(o)

# set/get length sampling expression
o$dist<-expression(rnorm(1,mean=3,sd=3))
o$dist

# set/get maximum event length
o$maxLength<-4
o$maxLength

# plot length density
plot(o)

# The following code illustrates how to use
# a ContinuousInsertor process in a simulation

# create a sequence object, attach process o
s<-NucleotideSequence(string="AAAAAAAAAGGAAAAAA", processes=list(list(o)))

# set the insertion tolerance to zero in range 11:15
# creating a region rejecting all insertions
setInsertionTolerance(s,o,0,11:15)

# get insertion tolerances
getInsertionTolerance(s,o)

# create a simulation object
simc<-PhyloSim(root.seq=s, phylo=rcool(2))

# simulate
Simulate(sim)

# print resulting alignment
```
**Description**

Copy a collection of Site objects aggregated by a Sequence object into a new Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
copySubSequence(this, index, process=NA, ...)
```

**Arguments**

- `this`: A Sequence object.
- `index`: An index vector specifying a collection of sites to be copied. It is set to 1:seq$length if omitted.
- `process`: The Process object performing the copy (optional).
- `...`: Not used.

**Value**

A new Sequence object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create a nucleotide sequence
s <- NucleotideSequence(string = "ATATATATATATATA")
# copy sites in the range 3:8 in a new object
s2 <- copySubSequence(s, 3:8)
s2
# copy sites 1,3 and 5 from s2
s3 <- copySubSequence(s2, c(1, 3, 5))
s3
```
The cpREV empirical amino acid substitution model

Description

Package:
Class cpREV

Usage

cpREV(equ.dist=NA, ...)

Arguments

equ.dist  Equilibrium distribution.
...  Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary
Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [<-[, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples

# create substitution model object
p<-cpREV()

# get object summary
summary(p)

# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )

# sample states
sampleStates(s)
    # make the first three positions invariable
setRateMultipliers(s,p,0,1:3)
    # get rate multipliers
gRateMultipliers(s,p)
    # create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcol(2))
Save a debug message in the PhyloSim log file

Description
Save a debug message in the PhyloSim log file.

The debug message is written to the log file only if the log level is non-negative. You can use this method for logging debug messages in the case you write classes for PhyloSim.

Usage
```
## S3 method for class 'PhyloSim'
Debug(this, message, ...)  
```

Arguments
- `this` A PhyloSim object.
- `message` A character vector of length one.
- `...` Not used.

Value
The message (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `PhyloSim`.

Examples
```
# create a PhyloSim object,  
# with loglevel set to zero  
sim<-PhyloSim(log.level=0);  
# log a debug message  
Debug(sim,"Some useful detail...");  
# close log connection  
close(sim$.log.connection)  
# print out the log file
```
deleteSubSequence.Sequence

Delete a collection of sites aggregated by a Sequence object

Description
Delete a collection of sites aggregated by a Sequence object.

Usage
```
## S3 method for class 'Sequence'
deleteSubSequence(this, index, ...)
```

Arguments
- `this` A Sequence object.
- `index` An index vector specifying a collection of sites to be deleted. It is set to 1:seq$length if omitted.
- `...` Not used.

Value
Invisible TRUE or error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `Sequence`.

Examples
```
# create a nucleotide sequence
s <- NucleotideSequence(string = "ATATATATATATAT")
# delete sites 2, 4 and 6
deleteSubSequence(s, c(2, 4, 6))
s
# delete sites in the range 3:6
deleteSubSequence(s, 3:6)
s
```
detachHookFromNode.PhyloSim

Detach a node hook function from a given node of a phylo object aggregated by a PhyloSim object.

Description

Detach a node hook function from a given node of a phylo object aggregated by a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
detachHookFromNode(this, node = NA, ...)
```

Arguments

- `this`: A PhyloSim object.
- `node`: Node identifier.
- `...`: Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

attachHookToNode PhyloSim Simulate.PhyloSim

Examples

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim <- PhyloSim(
  name = "TinySim",
  phylo = rcom(3),
  root.seq = NucleotideSequence(string = "ATGC", processes = list(JC69())))

# create a node hook function
hook <- function(seq = NA){
  # replace the substitution process with F84
  if(inherits(seq, "NucleotideSequence")){
    cat("Replacing JC69 with F84.\n")
    seq$processes <- list(F84(rate.params = list("Kappa" = 2)))
  }

  # return the new node hook function
  hook
}
```
detachProcess.Sequence

Detach a Process object from a set of Site objects aggregated by a Sequence object

Description

Detach a Process object from a set of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
detachProcess(this, process, index, ...)
```

Arguments

- `this`: A Sequence object.
- `process`: A Process object.
- `index`: An integer vector specifying a set of positions. It is set to `1:seq$length` if omitted.
- `...`: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.
Examples

# create a sequence object with two processes attached
p<-JC69()
s<-Sequence(length=4,alphabets=list(NucleotideAlphabet()),processes=list(list(p,K80())))
# get the list of attached processes
s$processes
# detach JC69 from range c(1,4)
detachProcess(s,p,c(1,4))
s$processes

Description

Site. The site-process specific parameters stored in the Site object and belonging to the detached Process objects will be destroyed.

Usage

## S3 method for class 'Site'
detachProcess(this, process, ...)

Arguments

this       A Site object
process    A Process object
...        Not used.

Value

The Site object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.
**detachSeqFromNode.PhyloSim**

*Detach a Sequence object from a given node of a phylo object aggregated by a PhyloSim object*

**Description**

Detach a Sequence object from a given node of a phylo object aggregated by a PhyloSim object. This method is mainly used internally.

**Usage**

```r
## S3 method for class 'PhyloSim'
detachSeqFromNode(this, node=NA, ...)
```

**Arguments**

- `this` A PhyloSim object.
- `node` Node identifier.
- `...` Not used.

**Value**

The PhyloSim object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.  

---

**Examples**

```r
# create a Site objects and attach some processes
s<-Site()
s$alphabet<-NucleotideAlphabet()
p1<-JC69(); p2<-K80()
attachProcess(s,p1)
attachProcess(s,p2)
# get the list of attached processes
s$processes
# detach p1
detachProcess(s,p1)
s$processes
```
DiscreteDeletor  The DiscreteDeletor class

Description

This class implements a process which performs deletions with lengths sampled from a user-specified discrete distribution. See GeneralDeletor for how the deletion process works.

Package:

Class DiscreteDeletor

Object

~|KMMpsroot

~~~~~~~~|

~~~~~~~~|--Process

~~~~~~~~~|

~~~~~~~~~|--GeneralIndel

~~~~~~~~~~|

~~~~~~~~~~|--GeneralDeletor

~~~~~~~~~~~~~~~~|

~~~~~~~~~~~~~~~~|--DiscreteDeletor

Directly known subclasses:

public static class DiscreteDeletor
extends GeneralDeletor

Usage

DiscreteDeletor(name="Anonymous", rate=NA, sizes=NA, probs=NA, ...)

Arguments

name The name of the object.
rate The general rate.
sizes The deletion sizes to propose.
probs A vector with the probabilities of the deletion sizes.
...

Additional arguments.

Fields and Methods

Methods:
DiscreteDeletor

checkConsistency
getProbs
getSizes
plot
setProbs
setSizes
summary

Methods inherited from GeneralDeletor:
checkConsistency, getEventsAtSite, is, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralDeletor ContinuousDeletor GeneralInDel

Examples

# create a DiscreteDeletor process
d<-DiscreteDeletor(  
name="M.D.",  
rate=0.25,  
sizes=c(1,2),  
probs=c(1/2,1/2)  
)

# get object summary
summary(d)

# set/get deletions sizes
DiscreteInsertor

The DiscreteInsertor class

Description

This class implements a process which performs insertions with lengths sampled from a user-specified discrete distribution. See GeneralInsertor for how the insertion process works.

Package:

Class DiscreteInsertor

Object

```
++|--PSRoot
     ++|Process
         ++|--GeneralInDel
             ++|--GeneralInsertor
                 ++|--DiscreteInsertor
```
Directly known subclasses:
BrownianInsertor

public static class DiscreteInsertor
extends GeneralInsertor

Usage
DiscreteInsertor(name="Anonymous", rate=NA, sizes=NA, probs=NA, ...)

Arguments
name  The name of the object.
rates The general rate.
sizes  The insertion sizes to propose.
probs  A vector with the probabilities of the insertion sizes.
...  Additional arguments.

Fields and Methods
Methods:

checkConsistency  -
getProbs     -
getSizes     -
plot         -
setProbs     -
setSizes     -
summary      -

Methods inherited from GeneralInsertor:
checkConsistency, generateInsert, getEventsAtSite, getGenerateBy, getInsertHook, getTemplateSeq, is, setAcceptWin, setGenerateBy, setInsertHook, setTemplateSeq, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamLds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamLds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, inter-
DiscreteInsertor

Methods inherited from Object:
$ S <-, S[[ ], S[[ ]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralInsertor ContinuousInsertor GeneralInDel

Examples

# create a DiscreteInsertor process
i <- DiscreteInsertor(
  name = "Mii",
  rate = 0.25,
  sizes = c(1, 2),
  probs = c(1/2, 1/2)
)

# set template sequence
i$templateSeq <- NucleotideSequence(string = "C")

# get object summary
summary(i)

# set/get insertion sizes
i$sizes <- 1:3

i$sizes

# set/get length probabilities
i$probs <- c(3, 2, 1)/6

i$probs

# plot length distribution
plot(i)

# The following code illustrates how to use
# a DiscreteInsertor process in a simulation

# create a sequence object and attach process i to it
s <- NucleotideSequence(string = "AAAAAAAAAGGGGAAAAAAAA", processes = list(list(i)))

# set the insertion tolerance to zero in range 11:15
# creating a region rejecting all insertions
setInsertionTolerance(s, i, 0, 11:15)

# get insertion tolerances
getInsertionTolerance(s, i)

# create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))

# simulate
Simulate(sim)

# print resulting alignment
The ECMrest empirical codon substitution model

Description

Package:
Class ECMrest

Object
~|~
~|~~|PSRoot
|~|Process
|~|GeneralSubstitution
|~|GeneralSubstitution
|~|CodonSubst
|~|CodonSubst
|~|ECMrest

Directly known subclasses:

public static class ECMrest
extends CodonSubst

Usage

ECMrest(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
...
Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from CodonSubst:
buildFromPAML, checkConsistency, newMatrix, setEquDist, summary
Methods inherited from GeneralSubstitution:
  as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
  !=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
  checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
  $, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
  Botond Sipos, Gregory Jordan

References

See Also
  CodonSubst GeneralSubstitution CodonUNREST

Examples
  # create substitution model object
  p<ECMrest()
  # get object summary
  summary(p)
  # display a bubble plot
  plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.
  # create a sequence, attach process p
  s<CodonSequence(length=10,processes=list(list(p)))
  # sample states
  sampleStates(s)
  # make the first three positions invariable
  setRateMultipliers(s,p,0,1:3)
  # get rate multipliers
The ECMunrest empirical codon substitution model

Description

Package:

Class ECMunrest

Object

~| 
~~|--PSRoot
~~~~~| 
~~~~~~~~--Process
~~~~~~~~| 
~~~~~~~~~~--GeneralSubstitution
~~~~~~~~~~| 
~~~~~~~~~~~~--CodonSubst
~~~~~~~~~~~~| 
~~~~~~~~~~~~~~~~--ECMunrest

Directly known subclasses:

public static class ECMunrest
extends CodonSubst

Usage

ECMunrest(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
...
Not used.
Fields and Methods

**Methods:**

*No methods defined.*

**Methods inherited from CodonSubst:**

buildFromPAML, checkConsistency, newMatrix, setEquDist, summary

**Methods inherited from GeneralSubstitution:**

as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

**Methods inherited from Process:**

!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

**Methods inherited from PSRoot:**

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**

$, $<-, [, [[-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

**Author(s)**

Botond Sipos, Gregory Jordan

**References**


**See Also**

CodonSubst GeneralSubstitution CodonUNREST

**Examples**

```r
# create substitution model object
p<-ECMunrest()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.
```
enableVirtual.PSRoot

Description

Enable the use of virtual fields for a given object. R.oo disables the virtual field feature inside get/set methods. This method can be used to re-enable virtual fields.

Usage

```r
## S3 method for class 'PSRoot'
enableVirtual(this, ...)
```

Arguments

- `this` A PSRoot object.
- `...` Not used.

Value

The PSRoot object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

# create a sequence, attach process p
s<-CodonSequence(length=10,processes=list(list(p))
# sample states
sampleStates(s)
    # make the first three positions invariable
setRateMultipliers(s,p,0,1:3)
    # get rate multipliers
getRateMultipliers(s,p)
    # create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoal(2))
    # run simulation
Simulate(sim)
    # print alignment
sim$alignment
The Event class

Description

This is the class representing an event. Event objects usually are generated by the getEventsAtSite.* methods based on the state of attached Site object and the properties of the generating Process object.

The Perform method will refuse to modify target objects if the position field is not set. The Perform method can be called only once for any Event object.

Package:
Class Event

Object

---
---+PSRoot
~~~~~~|
~~~~~~~~+Event

Directly known subclasses:

public static class Event
extends PSRoot

Usage

Event(name=NA, rate=NA, site=NA, position=NA, process=NA, handler=NA, ...)

Arguments

name The name of the Event object. Often stores useful information.
rate The rate of the event.
site The associated Site object.
position The position of associated Site object in the enclosing Sequence object (if any).
process The generator Process object.
handler The handler function for the Event object. It will be called by Perform.Event with the Event object as an argument to make the changes corresponding to the event.

... Not used.
Event

Fields and Methods

Methods:
Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Site Process Sequence getEventsAtSite, GeneralSubstitution

Examples
# Note: the following examples are not very useful
# unless you want to implement your own process.

# create a Sequence object and a Process object
seq<-NucleotideSequence(string="X");
p<-Process(alphabet=NucleotideAlphabet())
# get the Site object from the Sequence object
s<-seq$sites[[1]]
# attach p to s
exportStatTree.PhyloSim

attachProcess(s,p)
  # create an Event object
e<-Event(name="A->G",rate=0.1,site=s,process=p,position=1)
  # get object summary
  summary(e)
  # get event name
e$name
  # set/get event rate
  e$rate<-0.2
  e$rate
  # get site
  e$site
  # set/get event handler
  e$.handler<-function(this)(this$.site$state<="G")
e$handler
  # perform the event
  Perform(e)
  # check the state of the target site
  s$state

exportStatTree.PhyloSim

Export the per-branch counts of an event as a phylo object

Description

Export the per-branch counts of an event as a phylo object.

During simulation the number of events performed on every branch is recorded. The recorded events can be "basic" events, like "insertion", "deletion" and "A->T" or events which are sums of basic events, like "substitution". The getBranchEvents method returns a character vector with the names of the events having per-branch statistics recorded. The method should be called after the simulation is finished.

The per-branch statistics can be exported as phylo objects by using the exportStatTree method. The branch lengths of the exported phylo objects are set to the value of the respective per-branch event count.

Usage

## S3 method for class 'PhyloSim'
exportStatTree(this, event, ...)

Arguments

this A PhyloSim object.

event The name of the event as returned by the getBranchEvents method.

... Not used.
Value

A phylo object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see phylosim.

Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.

# NOTE: this will be a little bit slow
sim<-PhyloSim(
phylorcoal(3),
root.seq= CodonSequence(
    string="ATGATTATT",
    processes=list(list(GY94(kappa=2,omega.default=0.5)))
)
);
# make the tree longer to have more events
scaleTree(sim,5)
# plot the tree
plot(sim)
# run simulation
Simulate(sim)
# get the list of recorded per-branch event counts
getBranchEvents(sim)
# export the number of substitutions as a phylo object
subst<-exportStatTree(sim,"substitution")
# plot the exported phylo object
plot(subst)
# export the number of synonymous substitutions as a phylo object
subst<-exportStatTree(sim,"nr.syn.subst")
# plot the exported phylo object
plot(subst)
Description
This class implements the F81 GTR-submodel.

Package:
Class F81

| Object ~~~ | ~~~~ | ~~~~~~~~ | ~~~~~~~~~~~~ | ~~~~~~~~~~~~~~~~~~ | ~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| KMMpsroot | KMMprocess | KMMgeneralsubstitution | KMMunrest | KMMgtr | KMMfXQ |

Directly known subclasses:

public static class **F81** extends GTR

Usage
F81(name="Anonymous", base.freqs=c(0.25, 0.25, 0.25, 0.25), ...)

Arguments

| name | Object name. |
| base.freqs | Base frequency parameters. |
| ... | Not used. |

Fields and Methods

Methods:

- checkConsistency
- getBaseFreqs
- getRateParam
- getRateParamList
- setBaseFreqs
- setRateParam
Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [. $<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR UNREST GeneralSubstitution HKY

Examples
# create substitution process object
p<-f81(base.freqs=c(1,2,3,4)/10)
# get a summary
summary(p)
# display a bubble plot
plot(p)
The following code demonstrates how to use the process in a simulation.

```r
# create a sequence, attach process p
s <- NucleotideSequence(length=20, processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s, p, 0, 1:5)
# get rate multipliers
getRateMultipliers(s, p)
# create a simulation object
sim <- PhyloSim(root.seq=s, phylo=rcoral(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**F84**

*The F84 class*

**Description**

This class implements the F84 GTR-submodel. The rate parameters are the following: Kappa. Package: Class F84

```
Object
~~|--PSRoot
       |~~|--Process
          |~~~~~~~~~~|--GeneralSubstitution
          |~~~~~~~~~~|----------|--UNREST
          |~~~~~~~~~~|-----------|--GTR
          |~~~~~~~~~~|---------------|--F84
```

**Directly known subclasses:**

public static class **F84**
extends GTR
Usage

F84(name="Anonymous", rate.params=list(Kappa = 0), base.freqs=c(0.25, 0.25, 0.25, 0.25))

Arguments

name Object name.
rate.params Rate parameters.
base.freqs Base frequency parameters.
... Not used.

Fields and Methods

Methods:

checkConsistency -
getBaseFreqs -
getKappa -
getRateParam -
getRateParamList -
setBaseFreqs -
setKappa -
setRateParam -
setRateParamList -
summary -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam,
setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite,
getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix,
sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, get
ParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSite
SpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSite
SpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, inter
sect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden
**FastFieldDeletor**

**Methods inherited from Object:**
$\$, $\ll$, $\lll$, $\llll$, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
GTR UNREST GeneralSubstitution HKY

**Examples**

```r
# create substitution process object
p<-F84(rate.params=list("Kappa"=2), base.freqs=c(1,2,3,4))
  # get a summary
 summary(p)
  # display a bubble plot
 plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
  # sample states
 sampleStates(s)
  # make the first five positions invariable
 setRateMultipliers(s,p,0,1:5)
  # get rate multipliers
 getRateMultipliers(s,p)
  # create a simulation object
 sim<-PhyloSim(root.seq=s,phylo=rcoal(2))
  # run simulation
 Simulate(sim)
  # print alignment
 sim$alignment
```

---

**FastFieldDeletor**

The `FastFieldDeletor` class

**Description**

The `DiscreteDeletor` and `ContinuousDeletor` processes generate deletion event objects with rates determined by the general rate of the process and the "rate.multiplier" parameter. The probability of rejecting an event is determined by the product of the "deletion.tolerance" parameters from
the affected sites. If the majority of the sites have low deletion tolerance most of the events are rejected, which slows down the simulation without performing much events.

The FastFieldDeletor process scales down the rate and length distribution of the proposed events based on the highest insertion tolerance value observed in the root sequence (see the package vignette for details), thus making the simulation more efficient.

The available length distributions are (see also the package vignette):

- Geometric (default) - lengthParam1 is $\Lambda$.
- Poisson+1 - lengthParam1 is $\Lambda$.
- Conway-Maxwell Poisson+1 - lengthParam1 is $\Lambda$, lengthParam2 is $\nu$
- Negative Binomial+1 - lengthParam1 is $\Lambda$, lengthParam2 is $\nu$

Insertion processes can insert sites with deletion tolerance higher than the largest deletion tolerance observed in the root sequence. The user can specify the largest expected tolerance value through the toleranceMargin virtual field. The process is then scaled by max(initial_highest_tolerance, toleranceMargin).

Package:

**Class FastFieldDeletor**

```java
public static class FastFieldDeletor
extends GeneralDeletor
```

**Usage**

```java
FastFieldDeletor(name="Anonymous", type="geometric", length.param.1=NA,
                 length.param.2=NA, tolerance.margin=0, ...)
```

**Arguments**

- name: Object name.
- type: Type of the process (see above).
length.param.1 Object name.
length.param.2 Object name.
tolerance.margin Object name.
... Additional arguments.

Fields and Methods

Methods:

checkConsistency -
getEventsAtSite -
getLengthParam1 -
getLengthParam2 -
getToleranceMargin -
getType -
setLengthParam1 -
setLengthParam2 -
setParameterAtSite -
setToleranceMargin -
setType -
summary -

Methods inherited from GeneralDeletor:
checkConsistency, getEventsAtSite, is, summary

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [[, ][<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan
See Also

For more information see \texttt{FastFieldDeletor}.

Examples

\begin{verbatim}
# create a FastFieldDeletor object, default (geometric) type
length.param.1 is "lambda"
createHlengthH1, length.param.1H0.9, tolerance.marginH0.8)
# get type
p$type
# get object summary
summary(p)
# set/get lambda
p$lengthParam1H-0.8
p$lengthParam1
# set/get tolerance margin
p$toleranceMarginH-0.5
p$toleranceMargin
# create a nucleotide sequence, attach process
sH-NucleotideSequence(lengthH30, processesHlist(list(p)))
# set state pattern
# sample deletion tolerances
setDeletionTolerance(s, p, sample(seq(fromH0, toH0.8, byH0.01), 30, replaceHTRUE))
# plot deletion tolerance
plotParametersAtSites(s, p, "deletion.tolerance")
# simulate
simH-PhyloSim(rootHseq, phyloHrcoal(2))
Simulate(sim)
# show alignment
sim$alignment
\end{verbatim}

---

\texttt{flagTotalRate.Site} \hspace{1cm} \textit{Flag the total event rate}

Description

Flag the total event rate. This method sets the cached total active event rate to NA, which will trigger its recalculation when next accessed via \texttt{getTotalRate}.

Usage

```r
## S3 method for class 'Site'
flagTotalRate(this, ...)
```

Arguments

- **this** \hspace{1cm} A Site object.
- **...** \hspace{1cm} Not used.
**GeneralDeletor**

**Value**

Invisible NA.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *Site*.

**Examples**

```r
# create a site object:
p <- x80(rate.params = list("Alpha"=2,"Beta"=0.5))
s <- Site(alphabet=NucleotideAlphabet(), state="G", processes=list(p))
# get site rate
s$Rate
# modifying site object in a dangerous way (do not do this under any circumstances!)
s$.processes = list() # site object is now inconsistent!
# get cached rate
s$Rate # incorrect value
# flag total rate
flagRate(s)
# get site rate
s$Rate # correct value
```

---

The GeneralDeletor class

**Description**

This is the class implementing a process generating deletion events. The rates of the deletion events are calculated as the product of the general rate of the process and the "rate.multiplier" site-process-specific parameter.

The simulation code calls the `perform` method on the selected deletion event objects, which call their handler function to perform the deletion.

The deletion lengths are proposed by the function stored in the `proposeBy` virtual field. The function must have the following arguments: process (the process object), sequence (the target sequence), position (the position of the site which generated the event).

The deletion randomly affects the sites from the left or from the right of the target position (but never both). Positions which are out of range are discarded.

The proposed deletions are accepted or rejected by the function stored in the `acceptBy` virtual field. The function must have the following arguments: process (the deletion process), sequence (the target sequence), range (a vector of positions affected by the deletion).
The probability of accepting a deletion is calculated as the product of the "deletion.tolerance" site-process-specific parameters from the sites affected by the deletion event.

Package:
Class GeneralDeletor

Object
~~| KMMpsroot
~~~~~~| KMMprocess
~~~~~~~~~~| KMMgeneralindel
~~~~~~~~~~~~~~~~~| KMMgeneraldeletor

Directly known subclasses:
ContinuousDeletor, DiscreteDeletor, FastFieldDeletor

public static class GeneralDeletor
extends GeneralInDel

Usage
GeneralDeletor(name="Anonymous", rate=NA, propose.by=NA, accept.by=NA, ...)

Arguments
name The name of the object.
rate The general rate of the object.
propose.by A function used to propose events.
accept.by A function used to accept/reject events.
... Additional arguments.

Fields and Methods
Methods:

checkConsistency -
getEventsAtSite -
is -
summary -

Methods inherited from GeneralInDel:
checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
    GeneralInDel DiscreteDeletor ContinuousDeletor FastFieldDeletor

Examples
    # create a GeneralDeletor object
    # proposed deletion length: 4, always accept
    d<-GeneralDeletor(
        name = "DEL",
        rate = 1,
        propose.by=function(process, sequence, position){ 4 },
        accept.by=function(process, sequence, range){ TRUE }
    )

    d
    # check if object inherits from GeneralDeletor
    is.GeneralDeletor(d)
    # get object summary
    summary(d)
    # set/get name
    d$name<-"Del Bosque"
    d$name
    # set/get rate
    d$rate<-0.5
    d$rate
    # set/get proposeBy
    # propose deletion lengths between 3 and 6
    d$proposeBy<-function(process, sequence, position){ sample(3:6,1) }
    d$proposeBy
    # set/get acceptBy
# reject half of the events
d$acceptBy<-function(process, sequence, range){ sample(c(TRUE, FALSE), 1)}
d$acceptBy

# create a sequence object, attach process
s<-NucleotideSequence(string="AATTGCCCCGTTAA", processes=list(list(d)))

# set the rate multiplier
setRateMultiplier(s,d,2)

# get the list of active events at site 6
events<-getEventsAtSite(d,s$sites[[6]])

# print sequence
s

# set the position for the event object
e<-events[[1]];
e$.position<-6;

# perform the deletion event
Perform(e)

# check the results
s

---

GeneralInDel

The GeneralInDel class

Description

This is a class implementing the methods which are needed by both the GeneralInsertor and GeneralDeletor process.

Package:
Class GeneralInDel

Object

```
|---PSRoot
|   |---Process
|   |   |---GeneralInDel
```

Directly known subclasses:
- BrownianInsertor, ContinuousDeletor, ContinuousInsertor, DiscreteDeletor, DiscreteInsertor, Fast-FieldDeletor, GeneralDeletor, GeneralInsertor

public static class GeneralInDel
extends Process
Usage

GeneralInDel(name="Anonymous", rate=NA, propose.by=NA, accept.by=NA, ...)

Arguments

- name: The name of the object.
- rate: The general rate of the object.
- propose.by: A function used to propose events.
- accept.by: A function used to accept/reject events.
- ...: Additional arguments.

Fields and Methods

Methods:

- checkConsistency
- getAcceptBy
- getProposeBy
- getRate
- hasUndefinedRate
- is
- proposeLength
- setAcceptBy
- setProposeBy
- setRate
- summary

Methods inherited from Process:

- !=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:

- checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:

- $, $<-, [], [[<-], as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldName, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan
GeneralInsertor

See Also

Process GeneralInsertor GeneralDeletor GeneralSubstitution

Examples

# create a GeneralInDel object
# rejecting half of the events
# and proposing sizes in the range 1:10
o<-GeneralInDel(
    rate=1,
    propose.by=function(process){sample(1:10,1)},
    accept.by=function(){sample(c(TRUE,FALSE),1)}
);
# check if inherits from GeneralInDel
is-GeneralInDel(o)
# check if it has undefined rates
hasUndefinedRate(o)
# get object summary
summary(o)
# set/get proposeBy function via virtual field
o$proposeBy<-function(process){return(3)} # fixed event length
o$proposeBy
# set/get acceptBy function via virtual field
o$acceptBy<-function(){return(TRUE)} # accept all events
o$acceptBy
# get/set general rate
o$rate
o$rate<-2 # double the rate
# propose event length
proposeLength(o)

GeneralInsertor

The GeneralInsertor class

Description

This is a class implementing a process generating insertion events. The rate of each event is calculated as the product of the general rate of the process and the "rate.multiplier" site-process specific parameter. The simulation code calls the Perform method on the selected insertion event objects, which call their insertion event handler to perform the insertion.

The insert lengths are proposed by the function stored in the proposeBy virtual field. The function must have the following arguments: process (the insertion process object).

The insertion events are accepted or rejected by the function stored in the acceptBy virtual field. The function must have the following arguments: process (the insertion process object), sequence (the target sequence object), window (a vector of positions affecting acceptance). The probability of accepting an insertion is calculated as the product of the site-process-specific "insertion.tolerance"
parameters of the sites neighboring the insertion. The number of sites considered is determined by
the acceptWin virtual field.

The insert is generated by the generateInsert method by calling the function stored in the generateBy
virtual field. The default generator function truncates/duplicates the sequence object stored in the
templateSeq virtual field to get a sequence having the sampled length. After constructing the Se-
quence object, it calls the sampleStates.Sequence method on the resulting object. That means
that if we start with a template sequence which has NA states, but has a substitution process at-
tached, then the resulting sequence will be different every time.

Before inserting the sequence returned by generateInsert, the handler function will pass the
object through the function stored in the insertHook virtual field. This allows to perform arbitrary
modifications on the inserted Sequence object.

The sequence is inserted randomly on the left or the right of the target position.

Package:

Class GeneralInsertor

Object
~~|  
~~|PSRoot
~~~~~~|  
~~~~~~~~|Process
~~~~~~~~|GeneralInDel
~~~~~~~~|GeneralInsertor

Directly known subclasses:
BrownianInsertor, ContinuousInsertor, DiscreteInsertor

public static class GeneralInsertor
extends GeneralInDel

Usage

GeneralInsertor(name="Anonymous", rate=NA, propose.by=NA, accept.by=NA, template.seq=NA,
insert.hook=NA, accept.win=NA, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the object.</td>
</tr>
<tr>
<td>rate</td>
<td>The general rate of the object (no default).</td>
</tr>
<tr>
<td>propose.by</td>
<td>A function used to propose events (no default).</td>
</tr>
<tr>
<td>accept.by</td>
<td>A function used to accept/reject events (no default).</td>
</tr>
<tr>
<td>template.seq</td>
<td>A Sequence object used as a template for generating insertions (no default).</td>
</tr>
<tr>
<td>insert.hook</td>
<td>A function object, see setInsertHook (no default).</td>
</tr>
</tbody>
</table>
accept.win A window of sites affecting the acceptance of insert events.
...
Additional arguments.

**Fields and Methods**

**Methods:**

- checkConsistency
- generateInsert
- getAcceptWin
- getEventsAtSite
- getGenerateBy
- getInsertHook
- getTemplateSeq
- is
- setAcceptWin
- setGenerateBy
- setInsertHook
- setTemplateSeq
- summary

**Methods inherited from GeneralInDel:**
- checkConsistency, getAcceptBy, getProposeBy, getRate, hasUndefinedRate, is, proposeLength, setAcceptBy, setProposeBy, setRate, summary

**Methods inherited from Process:**
- !=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

**Methods inherited from PSRoot:**
- checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**
- $, $<-, [ ], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

GeneralInDel DiscreteInsertor ContinuousInsertor BrownianInsertor
Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(
  name = "GIN",
  rate = 1,
  propose.by = function(process){4}, # fixed insert length
  acceptBy = function(process, sequence, window){TRUE}, # always accept insertions
  template.seq = NucleotideSequence(string = "A"), # a boring template sequence
  insert.hook = function(seq){ return(seq)}, # a boring insert hook
  accept.win = 2 # 4 sites affecting acceptance
)

# check if object inherits from GeneralInsertor
is.GeneralInsertor(i)

# get object summary
summary(i)

# set/get general rate
$i$rate <- 0.5
$i$rate

# set/get name
$i$name <- "Ins"
$i$name

# set/get proposeBy
# sample insertion length between 1 and 10
$i$proposeBy <- function(process){sample(1:10, 1)}
$i$proposeBy

# set/get acceptBy
# reject half of the insertions
$i$acceptBy <- function(process, sequence, window){ sample(c(TRUE, FALSE), 1) }
$i$acceptBy

# get generateBy
$i$generateBy

# set/get acceptWin
$i$acceptWin <- 1;

# set/get insert hook
$i$insertHook <- function(
  seq,
  target.seq,
  event.pos,
  insert.pos
)
  attachProcess(seq, GTR());seq)

$i$insertHook

# set/get template sequence
$i$templateSeq <- NucleotideSequence(
  length = 5,
  processes = list(list(JC69())))

$i$templateSeq

# generate an insert sequence
generateInsert(i)

# create a sequence object and attach the process i
s <- NucleotideSequence(string = "AAAA", processes = list(list(i)))
```
# set rate multiplier
setRateMultipliers(s,i,2)
# get the list of active events from site 2
events<-getEventsAtSite(i,s$sites[[2]])
 events
# set position for event
e<-events[[1]]
e$.position<-2
# print sequence
s
# perform event
Perform(e)
# check sequence again
s

---

### GeneralSubstitution

#### The GeneralSubstitution class

**Description**

This a class representing a continuous-time Markov process acting on the state space defined by the symbols stored in the Alphabet object passed to the object constructor.

The GeneralSubstitution objects generate Event objects corresponding to substitution events based on the state of the attached Site objects.

The GeneralSubstitution objects aggregate a QMatrix object, which stores the unscaled and scaled rate matrices. The scaled rate matrices, along with the site-process specific rate multiplier parameters define the rates of the generated Event objects.

**Package:**

**Class GeneralSubstitution**

```
Object
~~|
~~~~~~PSSRoot
~~~~~~~|
~~~~~~~~~~~~~Process
~~~~~~~~~~~~~~~|
~~~~~~~~~~~~~~~~~GeneralSubstitution
```

**Directly known subclasses:**

AminoAcidSubst, BinarySubst, CodonSubst, CodonUNREST, ECMrest, ECMunrest, F81, F84, GTR, GY94, HKY, JC69, JTT, JTT.dcmut, K80, K81, LG, MtZoa, PAM, PAM.dcmut, T92, TN93, UNREST, WAG, cpREV, mtREV, mtMam, mtREV24

public static class GeneralSubstitution
   extends Process
Usage

GeneralSubstitution(name="Anonymous", alphabet=NA, rate.list=NA, equ.dist=NA, ...)

Arguments

name The name of the object.
alphabet The alphabet on which the process acts (Alphabet object).
rate.list A list with the substitution rates. It will be passed to setRateList method.
equ.dist The equilibrium distribution (see setEquDist. GeneralSubstitution).

Fields and Methods

Methods:
sect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
S, $<-, [.[, [][<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objetoSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Process QMatrix Event Site GeneralIndel GTR WAG

Examples

# Create a GeneralSubstitution object
# also provide an Alphabet objects
# and the list of unscaled rates
a<-BinaryAlphabet()
gs<-GeneralSubstitution(
  name="MyBinarySubst",
  alphabet=a,
  rate.list=list("0->1"=2,"1->0"=1)
)
# check if inherits from GeneralSubstitution
is.GeneralSubstitution(gs)
# get an object summary
summary(gs)
# get a bubble plot
plot(gs)
# get unscaled rate for "0->1" by event name
getRate(gs,"0->1")
# get unscaled rate for "0->1" by states
getRate(gs,from="0", to="1")
# get scaled rate for "0->1" by name
getEventRate(gs,"0->1")
# get the list of unscaled event rates
gs$rateList
# set the \texttt{unscaled} rates
gs$rateList<-list("0->1"=1,"1->0"=1)
# reset equilibrium distribution
gs$equDist<- 5 * gs$equDist
# get the equilibrium distribution
gs$equDist
# sample a state form the equilibrium distribution
sampleState(gs)
# get the associated QMatrix object
gs$qMatrix
# create a site object
s<-Site(alphabet=a, state="0")
generateInsert. GeneralInsertor

Generate an insert

Description

Generate an insert.

This method uses the function stored in the proposeBy virtual field to sample the insert length and then calls the function stored in the generateBy field to generate the insert.

The default generateBy function set by the GeneralInsertor constructor truncates/repeats the template sequence stored in the templateSeq field to have the sequence with the right size and then calls the sampleStates method on the resulting object. That means that if we start with a template sequence which has NA states, but has a substitution process attached, then the resulting sequence will be different every time the generateInsert method is called.

The generateBy, proposeBy and templateSeq fields must be set in order to use this method.

Usage

```r
## S3 method for class 'GeneralInsertor'
generateInsert(this, length=NA, target.seq=NA, event.pos=NA, insert.pos=NA, ...)
```

Arguments

- `this`: A GeneralInsertor object.
- `length`: Generate an insert with the specified length if this argument is present.
- `target.seq`: The Sequence object targeted by the insertion (optional). This argument is passed to the generateBy method.
getAcceptBy.GeneralInDel

getAcceptBy.GeneralInDel

Get the function used for accepting/rejecting indel events

Description

Get the function used for accepting/rejecting indel events.

event.pos

The position of the site proposing the insertion (optional). This argument is passed to the generateBy method.

insert.pos

The position of the insertion in the target sequence (optional). This argument is passed to the generateBy method.

... Not used.

Value

A Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralInsertor.

Examples

# build the template sequence
ts<-NucleotideSequence(length = 10, processes=list(list(JC69())))
# fix some site states
setStates(ts,"A",c(1,2));
setStates(ts,"T",c(5,6));
setStates(ts,"C",c(9,10));
# print template sequence
ts
# create a GeneralInsertor object
i<-GeneralInsertor(
rate=0.5,
template.seq=ts,
propose.by=function(process)(sample(c(5:50),1)), # inserts between 5 and 50
)
# generate some inserts
generateInsert(i)
generateInsert(i)
generateInsert(i)
generateInsert(i)

getAcceptBy. GeneralInDel

Get the function used for accepting/rejecting indel events
Usage

```r
## S3 method for class 'GeneralInDel'
getAcceptBy(this, ...)
```

Arguments

- `this`:
  - A GeneralInDel object.
- `...`:
  - Not used.

Value

A function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.

Examples

```r
# create a GeneralInDel object
# rejecting half of the events
o<--GeneralInDel(
  rate=1,
  propose.by=function(process){return(5)},
  accept.by=function( ){sample(c(TRUE,FALSE),1)}
);

# set/get the acceptBy function
setAcceptBy(o,value=function(){return(FALSE)}) # reject all events
getAcceptBy(o)
# set/get acceptBy function via virtual field
o$acceptBy<-function(){return(TRUE)} # accept all events
o$acceptBy
```

Description

Get the size of the acceptance window

This parameter determines the number of sites neighbouring the position (from left and right) of the insertion considered when accepting/rejecting a proposed insertion. The "insertion.tolerance" parameter is retrieved from sites falling in the window specified by this parameter. The default value is 1, so the two neighbouring sites are considered by default.
Usage

```r
## S3 method for class 'GeneralInsertor'
getAcceptWin(this, ...)
```

Arguments

- `this`: A GeneralInsertor object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `generalinsertor`.

Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(rate = 0.5);
# get acceptance window size
getAcceptWin(i)
# get acceptance window size via virtual field
i$acceptWin
# set acceptance window size
setAcceptWin(i, 2)
# set acceptance window size via virtual field
i$acceptWin <- 3
i$acceptWin
```

---

**getAlignment.PhyloSim**  
Get the alignment stored in a PhyloSim object

---

**Description**

Get the alignment stored in a PhyloSim object.

**Usage**

```r
## S3 method for class 'PhyloSim'
getAlignment(this, ...)
```
getAlignmentLength.PhyloSim

Arguments

  this    A PhyloSim object.
  ...    Not used.

Value

  The alignment as a matrix. Gap are represented by strings composed of dashes.

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see PhyloSim.

Examples

  # Create a PhyloSim object.
  # Provide the phylo object
  # and the root sequence.
  sim<-PhyloSim(
    name="TinySim",
    phylo=rcoal(3),
    root.seq=NucleotideSequence(string="ATG",processes=list(list(JC69())))
  );
  # run the simulation
  Simulate(sim)
  # get the resulting alignment
  getAlignment(sim)
  # via virtual fileld:
  sim$alignment

getAlignmentLength.PhyloSim

  Get the alignment length from a PhyloSim object

Description

  Get the alignment length from a PhyloSim object.
  This method retruns the number of columns in the alignment stored in the PhyloSim object.

Usage

  ## S3 method for class 'PhyloSim'
  getAlignmentLength(this, ...)


Arguments

this A PhyloSim object.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `phylosim`.

Examples

```r
# create a PhyloSim object and run a simulation:
sim<-Simulate(  
    PhyloSim(phy=rc(3),  
        root=NucleotideSequence(string="ATGC", proc=list(list(JC69()))) )  
)
# get the alignment length
gerAlignmementLength(sim)
```

---

**getAlphabet.GeneralSubstitution**

*Get the Alphabet object aggregated by a GeneralSubstitution object*

Description

Get the Alphabet object aggregated by a GeneralSubstitution object. This method also sets the alphabet for the associated QMatrix object, which will set all rates to NA.

Usage

```r
## S3 method for class 'GeneralSubstitution'
gerAlphabet(this, ...)  
```

Arguments

this A GeneralSubstitution object.

... Not used.
getAlphabet.Process

Value
An Alphabet object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralSubstitution.

Examples

# create a GeneralSubstitution object with an attached BinaryAlphabet object
p <- GeneralSubstitution(alphabet = BinaryAlphabet())
# get object summary
summary(p)
# get alphabet
getAlphabet(p)
# get alphabet via virtual field
p$alphabet
# set a new alphabet
setAlphabet(p, NucleotideAlphabet())
summary(p)
# set alphabet via virtual field
p$alphabet <- BinaryAlphabet()
p$alphabet
getAlphabet.QMatrix

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

Examples

```r
# create a process object
p <- Process(alphabet=NucleotideAlphabet())
# get associated Alphabet object
getAlphabet(p)
# via virtual field
p$alphabet
```

getAlphabet.QMatrix  Get the Alphabet object associated with a QMatrix object

Description
Get the Alphabet object associated with a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
getAlphabet(this, ...)
```

Arguments

```r
this  A QMatrix object.
...
Not used.
```

Value
An Alphabet object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see QMatrix.
getAlphabet.Site

Examples

# create a QMatrix object
m<-QMatrix()
# set the alphabet
setAlphabet(m,NucleotideAlphabet())
# get the alphabet
getAlphabet(m)
# set alphabet via virtual field
m$alphabet<-BinaryAlphabet()
summary(m)

getAlphabet.Site  Get the Alphabet object attached to a Site object

Description

Get the Alphabet object attached to a Site object.

Usage

## S3 method for class 'Site'
getAlphabet(this, ...)

Arguments

this          A Site object.
...           Not used.

Value

An Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.
Examples

```r
# create a site object
s <- Site()
# create an Alphabet object
a <- Alphabet(c("A","T","G"))
# attach alphabet to site object
setAlphabet(s,a)
# get attached alphabet
getAlphabet(s)
# get attached alphabet via virtual field
s$alphabet
```

---

### getAlphabet.ToleranceSubstitution

*Get the Alphabet object aggregated by a ToleranceSubstitution object*

---

**Description**

Get the Alphabet object aggregated by a ToleranceSubstitution object.

This method also sets the alphabet for the associated QMatrix object, which will set all rates to NA.

**Usage**

```r
## S3 method for class 'ToleranceSubstitution'
getAlphabet(this, ...)
```

**Arguments**

- `this`: A ToleranceSubstitution object.
- `...`: Not used.

**Value**

An Alphabet object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ToleranceSubstitution`. 


Examples

```r
# create a ToleranceSubstitution object with an attached BinaryAlphabet object
p <- ToleranceSubstitution(alphabet=BinaryAlphabet())
# get object summary
summary(p)
# get alphabet
getAlphabet(p)
# get alphabet via virtual field
p$alphabet
# set a new alphabet
setAlphabet(p,NucleotideAlphabet())
summary(p)
# set alphabet via virtual field
p$alphabet <- BinaryAlphabet()
p$alphabet
```

---

**getAlphabets.Sequence**

*Get the list of the Alphabet objects attached to the Site objects aggregated by a Sequence object*

---

**Description**

Get the list of the Alphabet objects attached to the Site objects aggregated by a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
getAlphabets(this, index, ...)
```

**Arguments**

- `this`: A Sequence object.
- `index`: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...`: Not used.

**Value**

A list of Alphabet objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`. 
Examples

# create a Sequence object with NucleotideAlphabet
# and BinaryAlphabet objects attached
s<-'Sequence(alphabets=list(NucleotideAlphabet(),BinaryAlphabet()),length=5)
setStates(s,c("A","0"))
# get the list of attached Alphabet objects
getAlphabets(s)
# get Alphabets from a range
getAlphabets(s,c(2:3,5))
# get alphabets via virtual field
s$alphabets

getAncestral.Sequence  Get the ancestral object of a Sequence object

Description

Get the ancestral object of a Sequence object.

Usage

## S3 method for class 'Sequence'
getAncestral(this, ...)

Arguments

this       A Sequence object.
...
Not used.

Value

A Sequence object or a Process object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.
Examples

```r
# create a sequence object
s <- Sequence(length = 4)
# get ancestral object
gtAncestral(s) # newly created sequences have the "Root insertion process" as ancestral
# clone sequence
cs <- clone(s)
# get ancestral object id via virtual field
cs$ancestral$id
```

---

**getAncestral.Site**  
*Get the ancestral object of a Site object*

Description

Get the ancestral object of a Site object.

Usage

```r
## S3 method for class 'Site'
gtAncestral(this, ...)
```

Arguments

- `this`  
  A Site object.
- `...`  
  Not used.

Value

A Site object, a Process object or NA.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
a <- Alphabet();
# create some site object
s1 <- Site(alphabet = a)
s2 <- Site(ancestral = s1, alphabet = a)
# get ancestral objects
getAncestral(s1)
s2$ancestral
```
**getBaseFreqs.F81**  
*Get the base frequency parameters*

**Description**

Get the base frequency parameters.

**Usage**

```r
## S3 method for class 'F81'
getBaseFreqs(this, ...)
```

**Arguments**

- `this`  
  A F81 object.
- `...`  
  Not used.

**Value**

A matrix containing the base frequency parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see F81.

**Examples**

```r
# construct a F81 object
p<-F81()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)
```
getBaseFreqs.F84

Get the base frequency parameters

Description

Get the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

Usage

```r
## S3 method for class 'F84'
getBaseFreqs(this, ...)
```

Arguments

- `this` A F84 object.
- `...` Not used.

Value

A matrix containing the base frequency parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F84`.

Examples

```r
# construct object
p<-F84()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$sbaseFreqs<-c(4,4,1,1)/10
p$sbaseFreqs
# get object summary
summary(p)
```
**getBaseFreqs.GTR**

*Get the base frequency parameters*

---

**Description**

Get the base frequency parameters.

**Usage**

```r
## S3 method for class 'GTR'
getBaseFreqs(this, ...)
```

**Arguments**

- `this` A GTR object.
- `...` Not used.

**Value**

A matrix containing the base frequencies.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GTR`.

**Examples**

```r
# construct a GTR object
p <- GTR()
# set/get base frequency parameters
setBaseFreqs(p, c(2, 1, 2, 1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs <- c(4, 4, 1, 1)/10
p$baseFreqs
# get object summary
summary(p)
```
Description
Get the base frequency parameters.
The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

Usage
```r
## S3 method for class 'HKY'
getBaseFreqs(this, ...)  
```

Arguments
- `this`: An HKY object.
- `...`: Not used.

Value
A matrix containing the base frequency parameters.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `HKY`.

Examples
```r
# construct object
p<-HKY()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)
```
**getBaseFreqs.K80**  
*Get the base frequency parameters*

**Description**

Get the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects. The K80 model has equal base frequencies.

**Usage**

```
## S3 method for class 'K80'
getBaseFreqs(this, ...)
```

**Arguments**

- `this` A K80 object.
- `...` Not used.

**Value**

A matrix containing the base frequency parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [K80](#).

**Examples**

```r
# construct object
p <- K80()
# get base frequency parameters
getBaseFreqs(p) # uniform
# set/get base frequency parameters via virtual field
p$baseFreqs # uniform
```
getBaseFreqs.K81

Get the base frequency parameters

Description

Get the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects. The K81 model has equal base frequencies.

Usage

### S3 method for class 'K81'

getBaseFreqs(this, ...)

Arguments

this A K81 object.

... Not used.

Value

A matrix containing the base frequency parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see K81.

Examples

```r
# construct object
p<-K81()
# get base frequency parameters
getBaseFreqs(p) # uniform
# set/get base frequency parameters via virtual field
p$baseFreqs # uniform
```
**getBaseFreqs.TN93**  
*Get the base frequency parameters*

**Description**

Get the base frequency parameters.

**Usage**

```r
## S3 method for class 'TN93'
getBaseFreqs(this, ...)
```

**Arguments**

- `this` A TN93 object.
- `...` Not used.

**Value**

A matrix containing the base frequencies.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `TN93`.

**Examples**

```r
# construct a TN93 object
p<-TN93()
# set/get base frequency parameters
setBaseFreqs(p,c(2,1,2,1)/6)
geBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs<-c(4,4,1,1)/10
p$baseFreqs
# get object summary
summary(p)
```
getBigRate.Sequence  

*Get the sum of all active event rates from a Sequence object*

**Description**

Get the sum of all active event rates from a Sequence object. The sum of active event rates depends on all Site object states and on the attached Process objects. It basically returns the last element of the cumulative site rates vector.

**Usage**

```r
## S3 method for class 'Sequence'
getBigRate(this, ...)
```

**Arguments**

- `this` A Sequence object.
- `...` Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create a nucleotide sequence attach a process
s<-NucleotideSequence(length=5);
s$processes<-list(list(JC69()))
# get the sum of active event rates
getBigRate(s) # returns NA because site states are undefined
# set site states
s$states<-c("A","T")
# get big rate via virtual field
s$bigRate
```
getBranchEvents.PhyloSim

Get the list of events having per-branch statistics recorded

Description

Get the list of events having per-branch statistics recorded.

During simulation the number of events performed on every branch is recorded. The recorded events can be "basic" events, like "insertion", "deletion" and "A->T" or events which are sums of basic events, like "substitution". The getBranchEvents method returns a character vector with the names of the events having per-branch statistics recorded. The method should be called after the simulation is finished.

The per-branch statistics can be exported as phylo objects by using the exportStatTree method. The branch lengths of the exported phylo objects are set to the value of the respective per-branch event count.

Usage

### S3 method for class 'PhyloSim'

getBranchEvents(this, ...)

Arguments

- **this**: A PhyloSim object.
- **...**: Not used.

Value

A character vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.

# NOTE: this will be a little bit slow
sim<--PhyloSim(
phylo=rcoal(3),
getCodonFreqs.GY94

Description
Get codon frequencies.

Usage
```
## S3 method for class 'GY94'
getCodonFreqs(this, ...)  
```

Arguments
- **this**: A GY94 object.
- **...**: Not used.

Value
A matrix containing the codon frequencies.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GY94.
Examples

# create a GY94 object
p<-GY94()
# get codon frequencies
getCodeFreqs(p)
p$codonFreqs
# set codon frequencies
p$codonFreqs<-rep(c(1,2,3,4),length.out=61)
p$codonFreqs

getComments.PSRoot Get the comments associated with an object

Description

Get the comments associated with an object.

The comment field can contain any type of object.

Usage

## S3 method for class 'PSRoot'
getComments(this, ...)

Arguments

this A PSRoot object.

... Not used.

Value

The value of the comment field.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRoot.
Examples

# create an object
o<-PSRoot()
# add some comments
setComments(o,"Random comment")
# get the comment
getComments(o)
# get/set the comment via virtual files
o$comments<-"Second random comment"
o$comments

getCumulativeRates.Sequence

Get the total site rates from a Sequence object

Description

Get the total site rates from a Sequence object. This method simply calls getCumulativeRatesFromRange(this). See getCumulativeRates.Sequence for more details.

Usage

## S3 method for class 'Sequence'
getCumulativeRates(this, ...)

Arguments

this A Sequence object.

... Not used.

Value

A numeric vector containing the total site rates.

Author(s)

Botond Sipos, Gregory Jordan

See Also

gCumulativeRatesFromRange.Sequence
getCumulativeRatesFromRange.Sequence

Get the cumulative site rates for a collection of Site objects aggregated by a Sequence object

Description

Get the cumulative site rates for a collection of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getCumulativeRatesFromRange(this, index, ...)
```

Arguments

- `this`: A Sequence object.
- `index`: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...`: Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a sequence with some processes attached
s <- Sequence(
    string="ATGC",
    alphabets=list(NucleotideAlphabet()),
    processes=list(list(JC69()),list(JC69(),GTR()))
)

# get cumulative rates for positions 1 and 3
getCumulativeRatesFromRange(s,c(1,3))

# get all cumulative rates via virtual field
s$cumulativeRates # via the "getCumulativeRates.Sequence" method
```
getDeletionTolerance.Sequence

Get the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object.

Description

Get the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls getParameterAtSites(this=this, process=process, index=index). See getParameterAtSites.Sequence for more details.

Usage

## S3 method for class 'Sequence'
getDeletionTolerance(this, process, index, ...)

Arguments

- `this` A Sequence object.
- `process` A Process object.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

getParameterAtSites.Sequence
**getDist.ContinuousDeletor**

*Get the length sampling expression*

---

**Description**

Get the length sampling expression. The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

**Usage**

```r
## S3 method for class 'ContinuousDeletor'
getDist(this, ...)
```

**Arguments**

- `this`: A ContinuousDeletor object.
- `...`: Not used.

**Value**

An R expression object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ContinuousDeletor`.

**Examples**

```r
# create object
o <- ContinuousDeletor(rate=1)
# set/get length sampling expression
setDist(o, expression(rnorm(1, mean=3, sd=2)))
getDist(o)
# set/get length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean=6, sd=3))
o$dist
# set maxLength
o$maxLength <- 10
# propose a length
proposeLength(o)
```
getDist.ContinuousInsertor

Get the length sampling expression

Description
Get the length sampling expression.
The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

Usage
## S3 method for class 'ContinuousInsertor'
getDist(this, ...)

Arguments
this A ContinuousInsertor object.
... Not used.

Value
An R expression object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see ContinuousInsertor.

Examples
# create object
o<-ContinuousInsertor(rate=1)
# set/get length sampling expression
setDist(o, expression(rnorm(1,mean=3, sd=2)))
getDist(o)
# set/get length sampling expression via virtual field
o$dist<-expression(rnorm(1,mean=6, sd=3))
o$dist
# set maxLength
o$maxLength<-10
# propose a length
proposeLength(o)
**getEdge.PhyloSim**  
*Get and edge from the edge matrix*

**Description**

Get and edge from the edge matrix.

**Usage**

```r
## S3 method for class 'PhyloSim'
getEdge(this, number = NA, ...)
```

**Arguments**

- **this**: A PhyloSim object.
- **number**: The edge number.
- **...**: Not used.

**Value**

The edge as a matrix with a single row.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [PhyloSim](https://example.com/phylosim).

**Examples**

```r
# create a PhyloSim object
sim <- PhyloSim(phylo = rcoal(5));
# get edge number 3
getEdge(sim, 3)
```
getEdges.PhyloSim

Get the edge matrix from a phylo object aggregated by a PhyloSim object.

Description

Get the edge matrix from a phylo object aggregated by a PhyloSim object.

The rows of the edge matrix contain the nodes connected by the edge and the edge length.

Usage

```r
## S3 method for class 'PhyloSim'
getEdges(this, ...)  
```

Arguments

- `this` A PhyloSim object.
- `...` Not used.

Value

A matrix.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
sim<-PhyloSim(phylo=coal(5));
# get the edge matrix
getEdges(sim)
# get the edge matrix via virtual field
sim$edges
```
getEquDist.GeneralSubstitution

Get the equilibrium distribution from a GeneralSubstitution object

Description
Get the equilibrium distribution from a GeneralSubstitution object.

Usage
### S3 method for class 'GeneralSubstitution'
getEquDist(this, dummy=NA, ...)

Arguments
- **this**: A GeneralSubstitution object.
- **dummy**: Not used.
- **...**: Not used.

Value
The new equilibrium distribution (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `GeneralSubstitution`.

Examples
```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-.GeneralSubstitution(
  alphabet=BinaryAlphabet(),
  rate.list=list("1->0"=1,"0->1"=1)
)
# get equilibrium distribution
getEquDist(p)
# get equilibrium distribution via virtual field
p$equDist
# re-set the equilibrium distribution
dist<-p$equDist * 3
dist
setEquDist(p,dist)
p$equDist
```
Description

Get the equilibrium distribution from a ToleranceSubstitution object.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
getEquDist(this, dummy=NA, ...)
```

Arguments

- `this` A ToleranceSubstitution object.
- `dummy` Not used.
- `...` Not used.

Value

The new equilibrium distribution (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ToleranceSubstitution`.

Examples

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p<-ToleranceSubstitution(
  alphabet=BinaryAlphabet(),
  rate.list=list("1->0"=1,"0->1"=1)
)

# get equilibrium distribution
getEquDist(p)
# get equilibrium distribution via virtual field
p$equDist
```
# re-set the equilibrium distribution
dist<-p$equDist * 3
dist
setEquDist(p,dist)
p$equDist
# re-set equilibrium distribution via virtual field
p$equDist<-p$equDist * 2
p$equDist

getEventRate.Generalsubstitution

Get the scaled rate of an event from a GeneralSubstitution object

Description

Get the scaled rate of an event from a GeneralSubstitution object.

This method return the element from the scaled rate matrix stored in the associated QMatrix object corresponding to a given event. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

This method doesn’t take into account the site specific rate multipliers in any way.

Usage

## S3 method for class 'GeneralSubstitution'
getEventRate(this, name=NA, from=NA, to=NA, ...)

Arguments

- `this` A GeneralSubstitution object.
- `name` The name of the event.
- `from` The initial state.
- `to` Target state.
- `...` Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.
**Examples**

```r
c # create a GeneralSubstitution object
c # provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("U->A"=1,"A->U"=1))
c # get the scaled rate of "U->A" by name
cgetEventRate(p,"U->A")
c # get the scaled rate of "U->A" by states
cgetEventRate(p,from="U",to="A")
```

---

**Description**

Get the unscaled rate of an event from a QMatrix object.

This method returns the element corresponding to a given event from the scaled rate matrix stored in a QMatrix object. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

This method returns NA if the rescaling of the rates was not performed. The scaling is performed by the `rescaleQMatrix` method.

**Usage**

```r
## S3 method for class 'QMatrix'
cgetEventRate(this, name=NA, from=NA, to=NA, ...)
```

**Arguments**

- **this**: A QMatrix object.
- **name**: The name of the event.
- **from**: The initial state.
- **to**: Target state.
- **...**: Not used.

**Value**

A Numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `QMatrix`. 
getEventRate.ToleranceSubstitution

Get the scaled rate of an event from a ToleranceSubstitution object

Description

Get the scaled rate of an event from a ToleranceSubstitution object. This method return the element from the scaled rate matrix stored in the associated QMatrix object corresponding to a given event. The event can be specified by the inital and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

This method doesn’t take into account the site specific rate multipliers in any way.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
getEventRate(this, name=NA, from=NA, to=NA, ...)
```

Arguments

- `this` A ToleranceSubstitution object.
- `name` The name of the event.
- `from` The initial state.
- `to` Target state.
- `...` Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ToleranceSubstitution`.

Examples

```r
# create a QMatrix object
# provide an Alphabet object and the rates
m <- QMatrix(alphabet=BinaryAlphabet(), rate=list("0->1"=1,"1->0"=1))
# get the unscaled rate of "0->1" by name
g.getenvRate(m,"0->1") # returns NA
# get the unscaled rate of "0->1" by states
g.getenvRate(m,from="0",to="1") # returns NA
```
**getEventRateAtSite. GeneralSubstitution**

*Get the site specific rate of an event from a GeneralSubstitution object*

**Description**

Get the site specific rate of an event from a GeneralSubstitution object.

This method returns the element from the associated QMatrix object corresponding to a given event multiplied by the "rate.multiplier" site-process specific parameter stored in the specified site object. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

**Usage**

```r
## S3 method for class 'GeneralSubstitution'
getEventRateAtSite(this, site, name=NA, from=NA, to=NA, ...)
```

**Arguments**

- `this` A GeneralSubstitution object. It must be attached to the provided Site object.
- `site` A Site object.
- `name` The name of the event.
- `from` The initial state.
- `to` Target state.
- `...` Not used.

**Value**

A Numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneralSubstitution`.  

**Examples**

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p <- ToleranceSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get the scaled rate of "0->1" by name
getEventRate(p,"0->1")
# get the scaled rate of "0->1" by states
getEventRate(p,from="0",to="1")
```
**Examples**

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# create a Site object
s <- Site(alphabet=BinaryAlphabet())
# attach process p to site s
s$processes <- list(p)
# set the rate multiplier for s and p
setParameterAtSite(p,s,id="rate.multiplier",value=2)
# get the site specific rate of "0->1" by name
getEventRateAtSite(p,s,"0->1")
# get the site specific rate of "0->1" by states
getEventRateAtSite(p,s,from="0",to="1")
```

**getEventRateAtSite.ToleranceSubstitution**

*Get the site specific rate of an event from a ToleranceSubstitution object*

**Description**

Get the site specific rate of an event from a ToleranceSubstitution object.

This method returns the element from the associated QMatrix object corresponding to a given event multiplied by the "rate.multiplier" site-process specific parameter stored in the specified site object. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

**Usage**

```r
## S3 method for class 'ToleranceSubstitution'
getEventRateAtSite(this, site, name=NA, from=NA, to=NA, ...)
```

**Arguments**

- `this`: A ToleranceSubstitution object. It must be attached to the provided Site object.
- `site`: A Site object.
- `name`: The name of the event.
- `from`: The initial state.
- `to`: Target state.
- `...`: Not used.

**Value**

A Numeric vector of length one.
getEvents.Sequence

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ToleranceSubstitution.

Examples

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p <- ToleranceSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# create a Site object
s <- Site(alphabet=BinaryAlphabet())
# attach process p to site s
s$processes <- list(p)
# set the rate multiplier for s and p
    setParameterAtSite(p,s,id="rate.multiplier",value=2)
# get the site specific rate of "0->1" by name
getEventRateAtSite(p,s,"0->1")
# get the site specific rate of "0->1" by states
getEventRateAtSite(p,s,from="0",to="1")
```

---

getEvents.Sequence  
Get the list of active Event objects for a set of Site objects aggregated by a Sequence object

Description

Get the list of active Event objects for a set of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getEvents(this, index, ...)
```

Arguments

- **this**: A Sequence object.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

A list of Event objects.
getEvents.Site

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Sequence.

Examples

```r
# create a sequence with a process attached
s <- Sequence(
  string = "ATGC",
  alphabets = list(NucleotideAlphabet()),
  processes = list(list(JC69()))
)

# get the active events from range 1:3
getEvents(s, 1:3)

# get all active events via virtual field
s$events
```

---

**getEvents.Site**

*Get the list of active event objects given the current state of the Site object*

---

Description
Get the list of active event objects given the current state of the Site object. The list of active event object might change according to the state of the Site object.

Usage

```r
## S3 method for class 'Site'
getEvents(this, ...)
```

Arguments

- `this` A Site object.
- `...` Not used.

Value
A list of event objects.

Author(s)
Botond Sipos, Gregory Jordan
See Also

For more information see Site.

Examples

```r
# create a site object with a JC69 substitution process attached
s <- Site(alphabet=NucleotideAlphabet(), state="A", processes=list(JC69()))
# get the list of active event objects
getEvents(s)
# modify site state
s$state <- "T"
# get the list of active event objects via virtual field
s$events
```

---

getEventsAtSite.FastFieldDeletor

*Generate a deletion event object given the state of the target site*

Description

Generate a deletion event object given the state of the target site.

Usage

```r
## S3 method for class 'FastFieldDeletor'
getEventsAtSite(this, target.site, ...)
```

Arguments

- `this`: A FastFieldDeletor object.
- `target.site`: A Site object.
- `...`: Not used.

Value

A list of Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralDeletor getEventsAtSite.GeneralDeletor
getEventsAtSite.GeneraDeletor

Title

Description

Title.

This method generates a list containing a single deletion event object. The rate of the event is calculated as the product of the general rate of the process and the "rate.multiplier" site-process specific parameter. An empty list is returned if the rate is zero or NA.

Usage

```r
## S3 method for class 'GeneraDeletor'
getEventsAtSite(this, target.site, ...)
```

Arguments

- `this`: A GeneraDeletor object.
- `target.site`: The target Site object.
- `...`: Not used.

Value

A list of event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneraDeletor`.

Examples

```r
# create the Sequence object
s <- NucleotideSequence(string="ATGCCGGCGGATTTATTA");
# create a GeneralDeletor object
# proposed deletion length: 4, always accept
d <- GeneralDeletor(
  name = "Del Bosque",
  rate = 0.5,
  propose.by=function(process, sequence, position){ 4 },
  accept.by=function(process, sequence, range){ TRUE }
)
# attach process to site
attachProcess(s, d);
```
getEventsAtSite.GeneralInsertor

Generate insertion event object given the state of the target site

Description

Generate insertion event object given the state of the target site. This method generates a list with one insertion event. The rate of the event is calculated as the product of the general rate of the process and the "rate.multiplier" site-process specific parameter. An empty list is returned if the rate is zero or NA.

Usage

## S3 method for class 'GeneralInsertor'
getEventsAtSite(this, target.site, ...)

Arguments

this A GeneralInsertor object.
target.site A Site object.
... Not used.

Value

A list of Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

GeneralInsertor GeneralInDel Process Event
Examples

```r
# create a sequence object
s <- NucleotideSequence(string = "AAAA")
# create a GeneralInsertor process, provide template sequence.
# proposed insert lengths: 3, always accept.
i <- GeneralInsertor(
  rate = 0.5,
  template.seq = NucleotideSequence(string = "GGG"),
  propose.by = function(process)(3),
  accept.by = function(process, sequence, window){TRUE}
)
# attach process to site
s$processes <- list(list(i));
# set rate multiplier
setRateMultipliers(s, i, 2)
# get the list of active events from site 2
events <- getEventsAtSite(i, s$sites[[2]])
events
# set position for event
e <- events[[1]]
e$.position <- 2
# print sequence
s
# perform event
Perform(e)
# check sequence again
s
```

getEventsAtSite. GeneralSubstitution

*Generate the list of active Event objects for a given attached Site object*

Description

Generate the list of active Event objects for a given attached Site object.

This is the single most important method in the GeneralSubstitution class. It generates a list of the active Event objects given the transition rate matrix (Q matrix) and the "rate.multiplier" Site-Process specific parameter. It returns an empty list if the state of the site is "NA".

Usage

```r
## S3 method for class 'GeneralSubstitution'
getEventsAtSite(this, target.site, ...)
```
Arguments

this A GeneralSubstitution object.
target.site A Site object. The GeneralSubstitution object must be attached to the Site object.
... Not used.

Value

A list of the active Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

# create an Alphabet object
a<-BinaryAlphabet()
# create a Site object
s<-Site(alphabet=a);
# create a GeneralSubstitution object
p<-GeneralSubstitution(alphabet=a,rate.list=list("0->1"=1,"1->0"=1))
# attach process p to site object s
attachProcess(s,p)
# get the rate of active events
getEventsAtSite(p,s) # empty list
# set the state of s
s$state<-1;
# get the rate of active events
getEventsAtSite(p,s)
**getEventsAtSite.Process**

**Usage**

```r
## S3 method for class 'GY94'
getEventsAtSite(this, target.site, ...)
```

**Arguments**

- `this`: A GY94 object.
- `target.site`: A Site object. The GY94 object must be attached to the Site object.
- `...`: Not used.

**Value**

A list of the active Event objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

- `getEventsAtSite.GeneralSubstitution GeneralSubstitution`

---

**Description**

Generate the list of active Event objects given a Site object. The Process object must be attached to the specified Site object.

This method is crucial for the simulations. For the Process class it returns an empty list. Descendant classes should implement meaningful getEventsAtSite methods.

**Usage**

```r
## S3 method for class 'Process'
getEventsAtSite(this, site, position, ...)
```

**Arguments**

- `this`: A Process object.
- `site`: A valid Site object.
- `position`: The position of the site in the enclosing Sequence object (if any).
- `...`: Not used.
getEventsAtSite.ToleranceSubstitution

Value
An empty list.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

Examples

```r
# create objects
a<-NucleotideAlphabet()
p<-Process(alphabet=a)
s<-Site(alphabet=a)
# attach Process p to Site s
s$processes<-list(p)
# get active Event objects (empty list)
getEventsAtSite(p,s)
```

Description

Generate the list of active Event objects for a given attached Site object.

This is the single most important method in the ToleranceSubstitution class. It generates a list of the active Event objects given the transition rate matrix (Q matrix) and the "rate.multiplier" Site-Process specific parameter. It returns an empty list if the state of the site is "NA".

Usage

```r
## S3 method for class 'ToleranceSubstitution'
getEventsAtSite(this, target.site, ...)
```

Arguments

- **this**: A ToleranceSubstitution object.
- **target.site**: A Site object. The ToleranceSubstitution object must be attached to the Site object.
- **...**: Not used.
Value

A list of the active Event objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ToleranceSubstitution.

Examples

```r
# create an Alphabet object
a<-BinaryAlphabet()
# create a Site object
s<-Site(alphabet=a);
# create a ToleranceSubstitution object
p<-ToleranceSubstitution(alphabet=a,rate=list(list("0->1"=1,"1->0"=1)))
# attach process p to site object s
attachProcess(s,p)
# get the rate of active events
getEventsAtSite(p,s) # empty list
# set the state of s
s$state<-1;
# get the rate of active events
getEventsAtSite(p,s)
```

getGenerateBy.GeneralInsertor

_Get the function object used for generating inserts_

Description

Get the function object used for generating inserts.

Usage

```r
## S3 method for class 'GeneralInsertor'
getGenerateBy(this, ...)
```

Arguments

- `this` A GeneralInsertor object.
- `...` Not used.
getHandler.Event

Value

A function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralInsertor.

Examples

# create a GeneralInsertor object
i<-GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq=NucleotideSequence(string="AAAAAA")
)

# save insert generator
old.gen<-getGenerateBy(i)
# set a new insert generator
i$generateBy<-function(
  process,
  length,
  target.seq,
  event.pos,
  insert.pos
  ){
    return(NucleotideSequence(string="AATGGCC"))
  }

# get the generator function
i$generateBy
# generate insert
generateInsert(i)
# restore old generator
i$generateBy<-old.gen
# generate insert
generateInsert(i)

getHandler.Event  Get the handler function of an Event object

Description

Get the handler function of an Event object.
### Usage

```r
## S3 method for class 'Event'
getHandler(this, ...)
```

### Arguments

- `this`: An Event object.
- `...`: Not used.

### Value

A function object.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Event`.

### Examples

```r
# create a sequence and attach a process
s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
# get the first active event from the first site
# only Sequence methods set .position,
# so $sites[[1]]$events[[1]] wouldn't work.
e <- getEvents(s, 1)[[1]]
# get the handler of e
getHandler(e)
```

---

### Description

Get the unique identifier of a PhyloSim object. The unique identifier is the concatenation of the class, the object name as returned by `getName()` and the object hash as returned by `hashCode()`.

### Usage

```r
## S3 method for class 'PhyloSim'
getId(this, ...)
```
Arguments

this    A PhyloSim object.

...    Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# create a PhyloSim object
do<PhyloSim(name="MySim");
# get id
gtid(o);
# get id via virtual field
o$id;

get Id. Process  Get the unique identifier of a Process object

Description

Get the unique identifier of a Process object. The unique identifier is the concatenation of the class, the object name as returned by getName() and the object hash as returned by hashCode().

Usage

## S3 method for class 'Process'
gId(this, ...)

Arguments

this    A Process object.

...    Not used.

Value

A character vector of length one.
Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.

Examples

```r
# create a Process object
p <- Process()
# get unique id
getId(p)
# get unique id via virtual field
p$id
```

**getid.QMatrix**  
Get the unique identifier of a QMatrix object

Description

Get the unique identifier of a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
getId(this, ...)
```

Arguments

- `this` A QMatrix object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.
**getId.Sequence**

**Examples**

```r
# create a QMatrix object
m<-QMatrix()
# get object id
g getId(m)
# get object id via virtual field
m$id
```

---

**Descripcion**

Obtener el identificador único de un objeto Sequence. El identificador es la concatenación del nombre del objeto y el código hash del objeto como devuelto por hashCode().

**Usó**

```r
## S3 method for class 'Sequence'
g getId(this, ...)
```

**Arguments**

- `this` : Un objeto Sequence.
- `...` : No utilizado.

**Valor**

Un vector de caracteres de longitud uno.

**Autor(es)**

Botond Sipos, Gregory Jordan

**Véase tambié**

Para más información vea Sequence.

**Examples**

```r
# create a Sequence object.
s<-Sequence(length=5)
# get id
g getId(s)
# get id via virtual field
s$id
```
getInsertHook.GeneralInsertor

Get the insert hook function

Description

Get the insert hook function.

The insert hook allows to make various modifications on the insert before performing the insertion.

The insert hook function is called by the insertion event handler function. The insert hook takes the sequence generated by the generateInsert method through the "seq" argument. The function must return a Sequence object, which will be inserted in the target sequence.

Usage

```r
## S3 method for class 'GeneralInsertor'
getInsertHook(this, ...)
```

Arguments

- `this` A GeneralInsertor object.
- `...` Not used.

Value

A function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(  
  rate = 0.5,  
  propose.by = function(process){sample(c(5:10),1)},  
  # inserts between 5 and 10  
  template.seq = NucleotideSequence(length = 5, processes = list(JC69())))

# set a dummy insert hook
setInsertHook(i, function(seq){return(seq)})

# set a new insert hook via virtual field
i$insertHook <- function(seq){  
  seq$processes <- list(list(GTR()))  
  # replace the substitution process  
  return(seq)
```
getInsertionTolerance.Sequence

Get the insertion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object.

Description

Get the insertion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls `getParameterAtSites(this = this, process = process, index = index)`. See `getParameterAtSites.Sequence` for more details.

Usage

```r
## S3 method for class 'Sequence'
getInsertionTolerance(this, process, index, ...)
```

Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **index**: An integer vector specifying a set of positions. It is set to `1:seq$length` if omitted.
- **...**: Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

`getParameterAtSites.Sequence`
getKappa.F84

Get the transition transversion rate ratio

Description

Get the transition transversion rate ratio.

Usage

```r
## S3 method for class 'F84'
getKappa(this, ...)
```

Arguments

- `this` A F84 object.
- `...` Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `fXT`.

Examples

```r
# construct a F84 object
p<-F84();
# set/get Kappa
setKappa(p,2)
getKappa(p)
# set/get Kappa via virtual field
p$kappa<-4
p$kappa
# get object summary
summary(p)
```
Description

Get the transition/transversion rate ratio.

Usage

```r
## S3 method for class 'GY94'
getKappa(this, ...)
```

Arguments

- `this`: A GY94 object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GY94`.

Examples

```r
# create a GY94 object
p <- GY94()
# set/get kappa
setKappa(p, 2)
getKappa(p)
# set/get kappa via virtual field
p$kappa <- 3
p$kappa
```
getLength.Sequence

Get the number of Site objects aggregated in a Sequence object

Description

Get the number of Site objects aggregated in a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getLength(this, ...)
```

Arguments

- `this` A Sequence object.
- `...` Not used.

Value

An integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a Sequence object
s <- Sequence(length=5)
# get sequence length
getLength(s)
# get length via virtual field
s$length
```
**Description**

Get the first length parameter.

**Usage**

```r
## S3 method for class 'FastFieldDeletor'
getLengthParam1(this, ...)
```

**Arguments**

- `this` A `FastFieldDeletor` object.
- `...` Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `FastFieldDeletor`.

**Examples**

```r
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get the first length parameter
setLengthParam1(p, 0.5)
getLengthParam1(p)
# set/get the first length parameter via virtual field
p$lengthParam1 <- 0.2
p$lengthParam1
```
getLengthParam2.FastFieldDeletor

*Get the second length parameter*

### Description

Get the second length parameter.

### Usage

```r
## S3 method for class 'FastFieldDeletor'
getLengthParam2(this, ...)
```

### Arguments

- `this` A `FastFieldDeletor` object.
- `...` Not used.

### Value

A numeric vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `FastFieldDeletor`.

### Examples

```r
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get the second length parameter
setLengthParam2(p, 0.5)
getLengthParam2(p)
# set/get the second length parameter via virtual field
p$lengthParam2 <- 0.2
p$lengthParam2
```
getLogFile.PhyloSim

Get the name of the file used for logging

Description

Get the name of the file used for logging.

Usage

```r
## S3 method for class 'PhyloSim'
getLogFile(this, ...)
```

Arguments

- `this` A PhyloSim object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# Create a PhyloSim object
sim <- PhyloSim()
# get the name of the log file
getLogFile(sim)
# modify log file name
setLogFile(sim, "OldLog.txt")
# get/set log file name via virtual field
sim$logFile
sim$logFile <- "NewLog"
sim$logFile
```
getLogLevel.PhyloSim

Get the log level from a PhyloSim object

Description

Get the log level from a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getLogLevel(this, ...)
```

Arguments

- `this`: A PhyloSim object.
- `...`: Not used.

Value

The log level as an integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

setLogLevel.PhyloSim

Examples

```r
# Create a PhyloSim object
sim<-PhyloSim();
# get/set log level
getLogLevel(sim)
setLogLevel(sim,0)
# set/get log level via virtual field
sim$logLevel<- -1
sim$logLevel
# clean up
unlink(sim$logFile)
```
Get the unscaled rate matrix form a QMatrix object

**Description**

Get the unscaled rate matrix form a QMatrix object.

**Usage**

```r
## S3 method for class 'QMatrix'
getMatrix(this, ...)
```

**Arguments**

- `this`: A QMatrix object.
- `...`: Not used.

**Value**

A matrix object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `QMatrix`.

**Examples**

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate=list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m<-p$QMatrix
# get the unscaled rate matrix from m
m$matrix
```
getMaxLength.ContinuousDeletor

Get the maximum length

Description
Get the maximum length.

Usage

```r
## S3 method for class 'ContinuousDeletor'
getMaxLength(this, ...)
```

Arguments

- `this`: A ContinuousDeletor object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `ContinuousDeletor`.

Examples

```r
# create object
o <- ContinuousDeletor(rate=1)
# set length sampling expression via virtual field
do$dist <- expression(rnorm(1, mean=6, sd=3))
# set/get maxLength
setMaxLength(o, 3)
geMaxLength(o)
# set/get maxLength via virtual field
o$maxLength <- 10
o$maxLength
# propose a length
proposeLength(o)
```
getMaxLength.ContinuousInsertor

Get the maximum length

Description

Get the maximum length.

Usage

```r
## S3 method for class 'ContinuousInsertor'
getMaxLength(this, ...)
```

Arguments

- `this`: A ContinuousInsertor object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousInsertor`.

Examples

```r
# create object
o <- ContinuousInsertor(rate=1)
# set length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean=6, sd=3))
# set/get maxLength
setMaxLength(o, 3)
getMaxLength(o)
# set/get maxLength via virtual field
o$maxLength <- 10
o$maxLength
# propose a length
proposeLength(o)
```
getMethodsList.PSRoot  Get the list of applicable methods for an object

Description

Get the list of applicable methods for an object.

Usage

```r
## S3 method for class 'PSRoot'
getMethodList(this, ...)
```

Arguments

- `this`  A PSRoot object.
- `...`   Not used.

Value

The list of applicable methods.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

Examples

```r
# create an object
o<-'PSRoot()
# get the applicable methods
getMethodList(o)
# get methods via virtual field
o$methodsList
```
getNamEvent

Get the name of an Event object

Description

Get the name of an Event object.

Usage

```r
## S3 method for class 'Event'
getName(this, ...)
```

Arguments

- `this`: An event object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create an Event object
e <- Event(name = "MyEvent")
# get event name
getName(e)
# get name via virtual field
e$name
```
**getDescription.PhyloSim**

**getDescription:**

Get the name of a PhyloSim object.

**Usage:**

```r
## S3 method for class 'PhyloSim'
getName(this, ...)
```

**Arguments:**

- **this**: A PhyloSim object.
- **...**: Not used.

**Value:**

A character vector of length one.

**Author(s):**

Botond Sipos, Gregory Jordan

**See Also:**

For more information see `PhyloSim`.

**Examples:**

```r
# create a PhyloSim object
o<-PhyloSim();
# set/get name
setName(o,"MySim");
getName(o,"MySim");
# set/get name via virtual field
o$name<-"George";
o$name
```
**Description**

Get the name of a Process object.

**Usage**

```r
## S3 method for class 'Process'
getName(this, ...)
```

**Arguments**

- `this` A Process object.
- `...` Not used.

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create a Process object
p <- Process()
# get object name
getName(p)
# get name via virtual field
p$name
```
getName.QMatrix  

Get the name of a QMatrix object

Description

Get the name of a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
getName(this, ...)
```

Arguments

- `this`: A QMatrix object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create a QMatrix object
m <- QMatrix()
# set/get name
setName(m, "Susie Q")
getName(m)
# set/get name via virtual field
m$name <- "Q"
m$name
```
**getDescription.Sequence**

Get the name of a Sequence object.

### Usage

```r
## S3 method for class 'Sequence'
getName(this, ...)
```

### Arguments

- `this`: A Sequence object.
- `...`: Not used.

### Value

A character vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Sequence`.

### Examples

```r
# create object
s <- Sequence(length=10);
# get object name
getName(s)
# get name via virtual field
s$name
```
getNedges.PhyloSim

### Description

Get the number of edges from phylo object aggregated by a PhyloSim object.

### Usage

```r
## S3 method for class 'PhyloSim'
getNedges(this, ...)
```

### Arguments

- `this` A PhyloSim object.
- `...` Not used.

### Value

A numeric vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `phylosim`.

### Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo = rcoal(5));
# get the number of the edges
gtNedges(sim)
# get the number of the edges via virtual field
sim$nedges
```
getNodes.PhyloSim

Get the node identifiers from a PhyloSim object

Description

Get the node identifiers from a PhyloSim object.

Usage

```r
### S3 method for class 'PhyloSim'
getNodes(this, ...)
```

Arguments

- `this` A PhyloSim object.
- `...` Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo=rcoal(5));
# get the node IDs
getNodes(sim)
# get the node IDs via virtual field
sim$nodes
```
getNtips.PhlyoSim

Get the number of the tips form a phylo object aggregated by a PhyloSim object

Description

Get the number of the tips form a phylo object aggregated by a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getNtips(this, ...)
```

Arguments

- `this` A PhyloSim object
- `...` Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo = rcoal(5));
# get the number of tips
getNtips(sim)
# get the number of tips via virtual field
sim$ntips
```
getOmes.CodonSequence

Get the omegas from a collection of sites

Description

Get the omegas from a collection of sites.

Usage

```r
## S3 method for class 'CodonSequence'
getOmes(this, process, index, ...)
```

Arguments

- `this` A CodonSequence object.
- `process` A process object inheriting from GY94.
- `index` A vector of positions.
- `...` Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonSequence`.

Examples

```r
# create a GY94 process
p<-GY94()
# create a CodonSequence object,
# attach a process p
s<-CodonSequence(length=20, processes=list(list(p)))
# set omega values in range 1:5
setOmes(s,p,c(0.5,1,1.5,1:5))
# get omega values from sites 1, 2, 3, 10, and 20
getOmes(s,p,c(1:3,10,20))
```
getOmegaScalingFactor.GY94

*Get the omega scaling factor*

### Description

Get the omega scaling factor.

The rate matrix of the GY94 model is scaled in a way that the expected number of potential substitutions per site is equal to one at equilibrium. The *codeml* program from the PAML package scales the rate matrix in order to have the expected number of accepted substitutions per site equal to one.

This method calculates the branch length multiplier needed for switching to PAML-style scaling given a fixed omega.

### Usage

```
## S3 method for class 'GY94'
getOmegaScalingFactor(this, omega, ...)
```

### Arguments

- `this` A GY94 object.
- `omega` The value of omega.
- `...` Not used.

### Value

A numeric vector of length one.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see GY94.

### Examples

```
# construct a GY94 process object
p<-GY94(kappa=4)
# Calculate scaling factor for omega=2
getOmegaScalingFactor(p,omega=2)
```
**getParameterAtSite.Process**

*Get the value of a site-process specific parameter from a Site object attached to a Process object*

**Description**

Get the value of a site-process specific parameter from a Site object attached to a Process object. The Process object must be attached to the Site object.

**Usage**

```r
## S3 method for class 'Process'
getParameterAtSite(this, site, id, ...)
```

**Arguments**

- `this`: A Process object.
- `site`: A Site object.
- `id`: The identifier of the site-process specific parameter.
- `...`: Not used.

**Value**

The site-process specific parameter, which is a list containing the following keys: id, name, value, type.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create a Site and a Process object
a<-Alphabet()
s<-Site(alphabet=a)
p<-Process(alphabet=a)
# attach the process
attachProcess(s,p)
# get the value of the rate multiplier
getParameterAtSite(p,s,"rate.multiplier")
```
getParameterAtSites.Sequence

Get the values of a site-process specific parameter for a process and a collection of Site objects aggregated by a Sequence object.

Description

Get the values of a site-process specific parameter for a process and a collection of Site objects aggregated by a Sequence object.

Usage

## S3 method for class 'Sequence'
getParameterAtSites(this, process, id, index, ...)

Arguments

- **this**: A Sequence object.
- **process**: A valid Process object.
- **id**: The identifier of the site-process specific parameter.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

A list of site specific parameters. A site specific parameter is a list storing the id, the name, the value and the type of the parameter.

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process For more information see `Sequence`.

Examples

```r
# create a sequence, attach a process
p<-K80()
s<-Sequence(length=6, alphabets=list(NucleotideAlphabet()), processes=list(list(p)))
# set a new pattern of rate multipliers in the
# range 1:3, the default value is 1.0 by the way
setParameterAtSites(s,p,"rate.multiplier",c(2,3),1:3)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")
```
getPhylo.PhyloSim

```r
# set a new value for the whole sequence
setParameterAtSites(s,p,"rate.multiplier",0.5)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")
```

---

### Description

Get the phylo object aggregated in a PhyloSim object.

### Usage

```r
## S3 method for class 'PhyloSim'
getPhylo(this, ...)```

### Arguments

- **this**: A PhyloSim object.
- **...**: Not used.

### Value

A phylo object or NA.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

The PhyloSim class, the **ape** package.

### Examples

```r
# create a PhyloSim object
sim<-PhyloSim();
# create a phylo object
tree<-rcoal(3);
# get/set phylo object
setPhylo(sim,tree);
glypho(sim,tree);
# get/set phylo object via virtual field
sim$tree<-rcoal(5);
sim$tree;
```
getPosition.Event  

Get the position of the Site object associated to an Event object in the enclosing Sequence object

Description

Get the position of the Site object associated to an Event object in the enclosing Sequence object.

Usage

```r
## S3 method for class 'Event'
getPosition(this, ...)  
```

Arguments

- `this`  
  An Event object.
- `...`  
  Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Event`.

Examples

```r
# create a sequence and attach a process
s<-NucleotideSequence(string="ATGC",processes=list(list(JC69())))
# get the first active event from the first site
e<-getEvents(s,1)[[1]]
# get the position of the site associated with e
getPosition(e)
# get position via virtual field
e$position
```
getProbs.DiscreteDeletor

*Get the deletion length probabilities*

**Description**

Get the deletion length probabilities.

**Usage**

```r
## S3 method for class 'DiscreteDeletor'
getProbs(this, ...)```

**Arguments**

- `this` A DiscreteDeletor object.
- `...` Not used.

**Value**

A numeric vector with the deletion length probabilities.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteDeletor`.

**Examples**

```r
# create a DiscreteDeletor object
d<-DiscreteDeletor(rate=1, sizes=1:3)
# set/get length probabilities
setProbs(d,c(1/3,1/3,1/3)) # equal probabilities
getProbs(d)
# set/get length probabilities via virtual field
x<-c(2,2,1)
# normalize x
x<-x/sum(x)
d$probs<-x
d$probs```
getProbs.DiscreteInsertor

Get the insertion length probabilities

Description

Get the insertion length probabilities.

Usage

```r
## S3 method for class 'DiscreteInsertor'
getProbs(this, ...)
```

Arguments

- `this`: A DiscreteInsertor object.
- `...`: Not used.

Value

A numeric vector with the insertion length probabilities.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `DiscreteInsertor`.

Examples

```r
# create a DiscreteInsertor object
i <- DiscreteInsertor(rate = 1, sizes = 1:3)
# set/get length probabilities
setProbs(i, c(1/3, 1/3, 1/3))  # equal probabilities
getProbs(i)
# set/get length probabilities via virtual field
x <- c(2, 2, 1)
# normalize x
x <- x / sum(x)
i$probs <- x
i$probs
```
**getProcess.Event**  
*Get the Process object which generated an Event object*

**Description**

Get the Process object which generated an Event object.

**Usage**

```r
## S3 method for class 'Event'
getProcess(this, ...)
```

**Arguments**

- `this`  
  An Event object.

- `...`  
  Not used.

**Value**

A Process object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# create a sequence and attach a process
s<-NucleotideSequence(string="ATGC",processes=list(list(JC69())))
# get the first active event from the first site
e<-s$sites[[1]]$events[[1]]
# get the generator process for e
e$process
```
getProcess.QMatrix

Get the process object associated with a QMatrix object

Description

Get the process object associated with a QMatrix object.

Usage

## S3 method for class 'QMatrix'
getProcess(this, ...)

Arguments

this

A QMatrix object.

... 

Not used.

Value

A process object, most likely one which inherits from GeneralSubstitution.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

# Create a GeneralSubstitution object
p <- GeneralSubstitution(alphabet=BinaryAlphabet())

# get the associated QMatrix object from p
m <- p$qMatrix

summary(m)

# get the associated process from m
m$process

# clone p

pp <- clone(p)

# associate m with pp
pp$qMatrix <- m

# associate pp with m
m$process <- pp

m$process
getProcesses.Sequence

Get the Process objects attached to the Site objects aggregated by a Sequence object

Description

Get the Process objects attached to the Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getProcesses(this, index, ...)
```

Arguments

- `this`: A Sequence object.
- `index`: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...`: Not used.

Value

A list of lists of Process objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a sequence object with some processes attached
s <- Sequence(
  length=4,
  alphabets=list(NucleotideAlphabet()),
  processes=list(list(JC69(),K80()),list(GTR()))
)

# get the list of lists of attached processes from positions 1 and 3
getProcesses(s, c(1,3))

# get processes via virtual field
s$processes
```
getProcesses.Site  
*Get the list of Process objects attached to a Site object*

**Description**

Get the list of Process objects attached to a Site object.

**Usage**

```r
## S3 method for class 'Site'
getProcesses(this, ...)
```

**Arguments**

- `this`: A Site object.
- `...`: Not used.

**Value**

A list of Process objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Site`.

**Examples**

```r
# create a Site object with some processes attached
s <- Site(alphabet=NucleotideAlphabet(), processes=list(K80(), JC69()))
# get list of attached Process objects
getProcesses(s)
# get list of attached Process objects via virtual field
s$processes
```
getProposeBy.GeneralInDel

Get the function used for proposing indel lengths

Description

Get the function used for proposing indel lengths.

Usage

```r
## S3 method for class 'GeneralInDel'
getProposeBy(this, ...)
```

Arguments

- `this`: A GeneralInDel object.
- `...`: Not used.

Value

A function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.

Examples

```r
# create a GeneralInDel object
# proposing events with a constant length of 5
o <- GeneralInDel(rate = 1, propose.by = function(process) { return(5) });
# set/get the proposeBy function
setProposeBy(o, value = function(process) { return(6) })
getProposeBy(o)
# set/get proposeBy function via virtual field
o$proposeBy <- function(process) { return(3) }
o$proposeBy
```
getQMatrix.GeneralSubstitution

Get the QMatrix object aggregated by a GeneralSubstitution object

Description

Get the QMatrix object aggregated by a GeneralSubstitution object.
This method is mostly used internally.

Usage

```r
## S3 method for class 'GeneralSubstitution'
getQMatrix(this, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `...`: Not used.

Value

A QMatrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate=list(list("1->0"=1,"0->1"=1)))
# get the QMatrix object
getQMatrix(p)
# get the QMatrix object via virtual field
q <- p$qMatrix
# tweak with the QMatrix
setRate(q,"0->1",2)
# set a new QMatrix for p
setQMatrix(p,q)
summary(p)
# set new QMatrix via virtual field
setRate(q,"1->0",2)
p$qMatrix <- q
```
getQMatrix.ToleranceSubstitution

Get the QMatrix object aggregated by a ToleranceSubstitution object

Description

Get the QMatrix object aggregated by a ToleranceSubstitution object.
This method is mostly used internally.

Usage

## S3 method for class 'ToleranceSubstitution'
getQMatrix(this, ...)

Arguments

this A ToleranceSubstitution object.
...
Not used.

Value

A QMatrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ToleranceSubstitution.

Examples

# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p <- ToleranceSubstitution(alphabet=BinaryAlphabet(), rate=list(list("1->0"=1,"0->1"=1)))
# get the QMatrix object
getQMatrix(p)
# get the QMatrix object via virtual field
q <- p$qMatrix
# tweak with the QMatrix
setRate(q, "0->1", 2)
# set a new QMatrix for p
setQMatrix(p, q)
summary(p)
getRate.Event

Get the rate of an Event object

Description

Get the rate of an Event object.

Usage

## S3 method for class 'Event'
getRate(this, ...)

Arguments

this       An Event object.
...        Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

# create an Event object
e <- Event(rate = 0.1)
# get rate
getRate(e)
# get rate via virtual field
e$rate
**getRate.GeneraInDel**  
*Get the general rate*

**Description**

Get the general rate.

**Usage**

```r
## S3 method for class 'GeneraInDel'
getRate(this, ...)
```

**Arguments**

- `this` A `GeneraInDel` object.
- `...` Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneraInDel`.

**Examples**

```r
# create a GeneraInDel object
doGeneraInDel(rate=0.5)
# get/set general rate
getRate(do)
setRate(do, 1.5)
# get/set rate via virtual field
do$rate
do$rate<-0.3
o$rate
```
getRate.GeneralSubstitution

Get an unscaled rate of an event from a GeneralSubstitution object

Description

Get an unscaled rate of an event from a GeneralSubstitution object.

This method gets the element corresponding to a given event form the unscaled Q matrix. A given event. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments. The rescaled rates (used during simulations) are returned by the getEventRate method.

This method doesn’t take into account the site specific rate multipliers in any way.

Usage

```r
## S3 method for class 'GeneralSubstitution'
getRate(this, name=NA, from=NA, to=NA, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `name`: The name of the event.
- `from`: The initial state.
- `to`: Target state.
- `...`: Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get the unscaled rate of "0->1" by name
getRate(p,"0->1")
# get the unscaled rate of "0->1" by states
```
**getRate.QMatrix**

### Description

Get an unscaled rate of an event from a QMatrix object.

This method gets the element corresponding to a given event from the *unscaled* rate matrix. A given event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

### Usage

```r
## S3 method for class 'QMatrix'
getRate(this, name=NA, from=NA, to=NA, ...)
```

### Arguments

- **this**: A QMatrix object.
- **name**: The name of the event.
- **from**: The initial state.
- **to**: Target state.
- **...**: Not used.

### Value

A Numeric vector of length one.

### See Also

For more information see `QMatrix`.

### Examples

```r
# create a QMatrix object
# provide an Alphabet object and the rates
m<-QMatrix(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get the unscaled rate of "0->1" by name
getRate(m,"0->1")
# get the unscaled rate of "0->1" by states
getRate(m,from="0",to="1")
```
getRate.ToleranceSubstitution

*Get an unscaled rate of an event from a ToleranceSubstitution object*

**Description**

Get an unscaled rate of an event from a ToleranceSubstitution object.

This method gets the element corresponding to a given event from the unscaled Q matrix. A given event. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments. The rescaled rates (used during simulations) are returned by the getEventRate method.

This method doesn’t take into account the site specific rate multipliers in any way.

**Usage**

```r
## S3 method for class 'ToleranceSubstitution'
getRate(this, name=NA, from=NA, to=NA, ...)
```

**Arguments**

- `this` A ToleranceSubstitution object.
- `name` The name of the event.
- `from` The initial state.
- `to` Target state.
- `...` Not used.

**Value**

A Numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ToleranceSubstitution`.

**Examples**

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p<-ToleranceSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# get the unscaled rate of "0->1" by name
getRate(p,"0->1")
# get the unscaled rate of "0->1" by states
```
getRateList(GeneralSubstitution)

Get a list of events and their unscaled rates from a GeneralSubstitution object

Description

Get a list of events and their unscaled rates from a GeneralSubstitution object.
This method returns the list of event rates from the unscaled Q matrix (as returned by the getEventRate method). The returned list contains the rates associated with the corresponding event names.

Usage

## S3 method for class 'GeneralSubstitution'
getRateList(this, ...)

Arguments

this
A GeneralSubstitution object.

... Not used.

Value

A list of event rates.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the event rates from the unscaled Q matrix
getRateList(p)
# get rates from the unscaled rate matrix via virtual field
p$raterList
# set rates in the unscaled rate matrix
setRateList(p, list("0->1"=1,"1->0"=1))
p$raterList
getRateList.QMatrix

Description

Get a list of events and their unscaled rates from a QMatrix object.
This method returns the list of event rates from the unscaled rate matrix. The returned list contains the rates associated with the corresponding event names.

Usage

```r
## S3 method for class 'QMatrix'
getRateList(this, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `...`: Not used.

Value

A list of event rates.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet = BinaryAlphabet(), rate.list = list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m <- p$QMatrix

# get the event rates from the unscaled Q matrix
getRateList(m)
# get rates from the unscaled rate matrix via virtual field
m$ratematrix
# set rates in the unscaled rate matrix
```
getRateList.ToleranceSubstitution

Description

Get a list of events and their unscaled rates from a ToleranceSubstitution object.

This method returns the list of event rates from the unscaled Q matrix (as returned by the getEventRate method). The returned list contains the rates associated with the corresponding event names.

Usage

## S3 method for class 'ToleranceSubstitution'
getRateList(this, ...)

Arguments

- **this**: A ToleranceSubstitution object.
- **...**: Not used.

Value

A list of event rates.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ToleranceSubstitution.

Examples

# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p<-ToleranceSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the event rates from the unscaled Q matrix
getRateList(p)
# get rates from the unscaled rate matrix via virtual field

```r
setRateList(m, list("0->1"=1,"1->0"=1))
m$rateList
# set rates in the unscaled rate matrix via virtual field
m$rateList<-list("0->1"=3,"1->0"=1);
m$rateList
```
getRateMultipliers.Sequence

Get the values of the rate multiplier parameters for a given Process object and a collection of Site object aggregated by a Sequence object.

Description

Get the values of the rate multiplier parameters for a given Process object and a collection of Site object aggregated by a Sequence object. This method just calls getParameterAtSites(this=\this, process=\process, id="\ratemultiplier\index")

See getParameterAtSites.Sequence for details.

Usage

## S3 method for class 'Sequence'
getRateMultipliers(this, process, index, ...)

Arguments

- **this**: A Sequence object.
- **process**: A valid Process object.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- ...: Not used.

Value

A numeric vector with the current values of the rate multiplier in the specified range.

Author(s)

Botond Sipos, Gregory Jordan

See Also

setParameterAtSites.Sequence
getRateParam.F81

Forbidden action: the value of a rate parameters

Description

Forbidden action: the value of a rate parameters.
This model has no rate parameters.

Usage

```r
## S3 method for class 'F81'
getRateParam(this, name, ...)
```

Arguments

- `this`: An object.
- `name`: The name of the rate parameter.
- `...`: Not used.

Value

The object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see F81.

getRateParam.F84

Get the value of a rate parameter

Description

Get the value of a rate parameter.

The rate parameters are: Kappa.

Usage

```r
## S3 method for class 'F84'
getRateParam(this, name, ...)
```
getRateParam.GTR

**Arguments**

- `this`: A F84 object.
- `name`: The name of the rate parameter.
- `...`: Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see F84.

**Examples**

```r
# create F84 object
p<-F84()
# set/get rate parameters
setRateParamList(p,list("Kappa"=3))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list("Kappa"=2.5)
p$rateParamList
# get object summary
summary(p)
```

---

**getRateParam.GTR**

*Get the value of a rate parameter*

**Description**

Get the value of a rate parameter.

The rate parameters are named as in PAML (see PAML documentation: [http://bit.ly/9SQK2f](http://bit.ly/9SQK2f)).

**Usage**

```r
## S3 method for class 'GTR'
getRateParam(this, name, ...)
```
**Arguments**

- `this` A GTR object.
- `name` The name of the rate parameter.
- `...` Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GTR`.

**Examples**

```r
# construct a GTR object
p <- GTR();
# set/get a rate parameter
setRateParam(p, "a", 4)
getRateParam(p, "a")
# get object summary
summary(p)
```

---

**getRateParam.HKY**  
*Get the value of a rate parameter*

**Description**

Get the value of a rate parameter.

The rate parameters are: Alpha, Beta.

**Usage**

```r
## S3 method for class 'HKY'
getRateParam(this, name, ...)
```

**Arguments**

- `this` An HKY object.
- `name` The name of the rate parameter.
- `...` Not used.
Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see HKY.

Examples

```r
# construct HKY object
p<-HKY();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```

getRateParam.K80  

Get the value of a rate parameter

Description

Get the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'K80'
getRateParam(this, name, ...)
```

Arguments

- `this`  
  A K80 object.

- `name`  
  The name of the rate parameter.

- `...`  
  Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan
getRateParam.K81

See Also

For more information see K80.

Examples

```r
# construct a K80 object
p<-K80();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```

getRateParam.K81  
*Get the value of a rate parameter*

Description

Get the value of a rate parameter.

The rate parameters are: Alpha, Beta, Gamma.

Usage

```r
## S3 method for class 'K81'
getRateParam(this, name, ...)
```

Arguments

- `this` A K81 object.
- `name` The name of the rate parameter.
- `...` Not used.

Value

The value of the rate parameter.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see K81.
Examples

```r
# construct a K81 object
p<-K81();
# set/get a rate parameter
setRateParam(p, "Alpha", 4)
getRateParam(p, "Gamma")
# get object summary
summary(p)
```

---

**getRateParam.T92**  
*Get the value of a rate parameter*

Description

Get the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'T92'
getRateParam(this, name, ...)
```

Arguments

- `this`  
  A T92 object.
- `name`  
  The name of the rate parameter.
- `...`  
  Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see T92.
Examples

```r
# construct a T92 object
p <- T92();
# set/get a rate parameter
setRateParam(p, "Alpha", 4)
getRateParam(p, "Beta")
# get object summary
summary(p)
```

Description

Get the value of a rate parameter.

The rate parameters are: Alpha1, Alpha2, Beta.

Usage

```r
## S3 method for class 'TN93'
getRateParam(this, name, ...)
```

Arguments

- **this**: A TN93 object.
- **name**: The name of the rate parameter.
- **...**: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `TN93`.
Examples

```r
# construct a TN93 object
p <- TN93()
# set/get a rate parameter
setRateParam(p, "Beta", 4)
getRateParam(p, "Beta")
# get object summary
summary(p)
```

Forbidden action: getting the list of rate parameters

Description

Forbidden action: getting the list of rate parameters.

This model has no rate parameters.

Usage

```r
## S3 method for class 'F81'
getRateParamList(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

The object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F81`. 
**Description**

Get the rate parameters.

The rate parameters are: Kappa.

**Usage**

```r
## S3 method for class 'F84'
generateRateParamList(this, ...)
```

**Arguments**

- `this`: A `F84` object.
- `...`: Not used.

**Value**

The list of rate parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `F84`.

**Examples**

```r
# create F84 object
p <- F84()
# set/get rate parameters
setRateParamList(p, list("Kappa"=3))
generateRateParamList(p)
# set/get rate parameters via virtual field
p@rateParamList <- list("Kappa"=2.5)
p@rateParamList
# get object summary
summary(p)
```
getRateParamList.GTR  Get the rate parameters

Description

Get the rate parameters.

The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/9SJK2f).

Usage

```r
## S3 method for class 'GTR'
getRateParamList(this, ...)
```

Arguments

- `this` A GTR object.
- `...` Not used.

Value

A list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GTR.

Examples

```r
# create GTR object
p <- GTR()
# set/get rate parameters
setRateParamList(p, list(  
  "a"=1, "b"=2, "c"=3,  
  "d"=1, "e"=2, "f"=3
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(  
  "a"=4, "b"=1, "c"=4,  
  "d"=1, "e"=4, "f"=1
)
p$rateParamList
# get object summary
summary(p)
```
getRateParamList.HKY  Get the rate parameters

Description

Get the rate parameters.
The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'HKY'
getRateParamList(this, ...)
```

Arguments

- `this`  
  An HKY object.
- `...`  
  Not used.

Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `HKY`.

Examples

```r
# create HKY object
p<-HKY()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3
)
p$rateParamList
# get object summary
summary(p)
```
getRateParamList.K80  Get the rate parameters

Description

Get the rate parameters.
The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'K80'
getRateParamList(this, ...)
```

Arguments

- `this`  A K80 object.
- `...`  Not used.

Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see K80.

Examples

```r
# create K80 object
p<-K80()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5
  )
)  
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3
)
  
p$rateParamList
# get object summary
summary(p)
```
getRateParamList.K81  Get the rate parameters

Description
Get the rate parameters.
The rate parameters are: Alpha, Beta, Gamma.

Usage
```r
## S3 method for class 'K81'
getRateParamList(this, ...)
```

Arguments
- `this`: A K81 object.
- `...`: Not used.

Value
The list of rate parameters.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see K81.

Examples
```r
# create K81 object
p<-K81()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5,
  "Gamma"=2
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3,
  "Gamma"=2
)
p$rateParamList
```
**getRateParamList.T92**  
*Get the rate parameters*

## Description
Get the rate parameters.

The rate parameters are: Alpha, Beta.

## Usage

```r
## S3 method for class 'T92'
generateRateParamList(this, ...)
```

## Arguments

- `this`: A T92 object.
- `...`: Not used.

## Value
The list of rate parameters.

## Author(s)
Botond Sipos, Gregory Jordan

## See Also
For more information see `T92`.

## Examples

```r
# create a T92 object
p <- T92()
# set/get rate parameters
setRateParamList(p, list(
  "Alpha" = 1,
  "Beta" = 0.5
))
generateRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(
  "Alpha" = 1,
  "Beta" = 3
)
```
getRateParamList.TN93  

Get the rate parameters

Description

Get the rate parameters.

The rate parameters are: Alpha1, Alpha2, Beta.

Usage

```r
## S3 method for class 'TN93'
getRateParamList(this, ...)
```

Arguments

- `this`: A TN93 object.
- `...`: Not used.

Value

The list of rate parameters.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `TN93`.

Examples

```r
# create TN93 object
p<-TN93()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha1"=1,
  "Alpha2"=2,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha1"=1,
```
getRootNode.PhyloSim

Get the identifier of the root node from a PhyloSim object

Description

Get the identifier of the root node from a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getRootNode(this, ...)
```

Arguments

- `this`: A PhyloSim object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
sim<-PhyloSim(phylo=rcoal(5));
# get the root node ID
getRootNode(sim)
# get the root node ID via virtual field
sim$rootNode
```
Get the root sequence aggregated by a PhyloSim object

**Description**

Get the root sequence aggregated by a PhyloSim object.

**Usage**

```r
## S3 method for class 'PhyloSim'
getRootSeq(this, ...)
```

**Arguments**

- `this`: A PhyloSim object.
- `...`: Not used.

**Value**

The root Sequence object or NA.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `phylosim`.

**Examples**

```r
# create some objects
sim<-PhyloSim(phylo=rcoal(3));
seq<-NucleotideSequence(string="ATGCC");
# set/get root sequence
setRootSeq(sim, seq);
getRootSeq(sim, seq);
# set/get root sequence via virtual field
sim$rootSeq<-BinarySequence(string="11100011000");
sim$rootSeq;
```
getScale.BrownianInsertor

Get scale parameter

Description

Get scale parameter.

Usage

## S3 method for class 'BrownianInsertor'
getScale(this, ...)

Arguments

- `this` A BrownianInsertor object.
- `...` Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `BrownianInsertor`.

Examples

```r
# create a BrownianInsertor object
p <- BrownianInsertor(scale=0.002)
# set/get scale parameter
setScale(p, 0.1)
getScale(p)
# set/get scale parameter via virtual field
p$scale <- 0.1
p$scale
```
getScaledMatrix.QMatrix

Get the scaled rate matrix form a QMatrix object

Description

Get the scaled rate matrix form a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
getScaledMatrix(this, ...)
```

Arguments

- `this`: A QMatrix object.
- `...`: Not used.

Value

A matrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet = BinaryAlphabet(), rate.list = list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m <- p$QMatrix
# get the scaled rate matrix from m
m$scaledMatrix
```
getSeqFromNode.PhyloSim

Get the Sequence object associated with a given node of a phylo object aggregated by a PhyloSim object

Description

Get the Sequence object associated with a given node of a phylo object aggregated by a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
getSeqFromNode(this, node=NA, ...)

Arguments

- **this**: A PhyloSim object.
- **node**: Node identifier.
- **...**: Not used.

Value

A Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoral(3),
  root.seq=NucleotideSequence(string="ATG", processes=list(list(JC69())))
);
# get the sequence associated with node 5
getSeqFromNode(sim, 5) # Should be NA
# Run the simulation
Simulate(sim)
# try again
```
getSequence.Site

getSeqFromNode(sim, 5)

---

**getSequence.Site**  
*Get the Sequence object associated with a given Site object*

**Description**

Get the Sequence object associated with a given Site object.

**Usage**

```r
## S3 method for class 'Site'
getSequence(this, ...)
```

**Arguments**

- `this`  
  A Site object.
- `...`  
  Not used.

**Value**

A Sequence object or NA.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Site`.

**Examples**

```r
# create a site object
s <- Site(sequence = Sequence())
# get the associated Sequence object
class(s)
getSequence(s)
# get the associated Sequence object via virtual field
s$sequence
```
getSequences.PhyloSim  

**getSequences.PhyloSim**  

*Gets all the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object*

---

**Description**

Gets all the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object.

The order of the Sequence objects in the returned list reflects the identifiers of the associated nodes.

**Usage**

```r
## S3 method for class 'PhyloSim'
getSequences(this, ...)  
```

**Arguments**

- `this`  
  A PhyloSim object.

- `...`  
  Not used.

**Value**

A list of sequence objects.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

**Examples**

```r
# Create a PhyloSim object.  
# Provide the phylo object  
# and the root sequence.  
sim<-PhyloSim(  
  name="TinySim",  
  phylo=rcal(3),  
  root.seq=NucleotideSequence(string="ATG",processes=list(list(JC69())))  
);  
# run the simulation  
Simulate(sim)  
# get all the associated sequence objects  
getSequences(sim)  
# get the sequence associated with node 3  
# via virtual field  
```
**getSite.Event**

Get the Site object associated with an Event object.

### Description

Get the Site object associated with an Event object.

### Usage

```r
## S3 method for class 'Event'
getSite(this, ...)
```

### Arguments

- **this**: An Event object.
- **...**: Not used.

### Value

A Site object.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Event`.

### Examples

```r
# create a sequence and attach a process
s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
# get the first active event from the first site
e <- s$sites[[1]]$events[[1]]
# get the site associated with e
getSite(e)
# get site via virtual field
e$site
```
getSites.Sequence

Get the list of the Site object aggregated in a Sequence object

Description

Get the list of the Site object aggregated in a Sequence object. Warning: there is no setSites method!

Usage

```r
## S3 method for class 'Sequence'
getSites(this, ...)
```

Arguments

- `this` A Sequence object.
- `...` Not used.

Value

A list of Site objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a sequence object
s <- Sequence(alphabets = list(NucleotideAlphabet()), string = "AATTGCCC")
# get the list of aggregated Site objects
getSites(s)
# get Site objects via virtual field
s$sites
```
**Description**

Get the site specific parameter identifiers from a Process object.

**Usage**

```r
## S3 method for class 'Process'
getSiteSpecificParamIds(this, ...)
```

**Arguments**

- `this`: A Process object.
- `...`: Not used.

**Value**

A character vector.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create process object
p <- Process()
# get site specific parameter identifiers
gSiteSpecificParamIds(p)
# via virtual field
p$siteSpecificParamIds
```
getSiteSpecificParamList.Process

*Get the list of site specific parameters of a Process object*

**Description**

Get the list of site specific parameters of a Process object. Every site specific parameter is a list storing the name, the identifier and the type of the given parameter.

**Usage**

```r
## S3 method for class 'Process'
getSiteSpecificParamList(this, ...)  
```

**Arguments**

- `this`: A Process object.
- `...`: Not used.

**Value**

A list of site specific parameters.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create a process object
p <- Process()
# get the list of site specific parameters
getSiteSpecificParamList(p)
# get it via virtual field
p$siteSpecificParamList
```
getSize.Alphabet

Get the number of symbols in an Alphabet object

Description

Get the number of symbols in an Alphabet object.

Usage

```r
## S3 method for class 'Alphabet'
getSize(this, ...)
```

Arguments

- `this`: An Alphabet object.
- `...`: Not used.

Value

An integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `alphabet`.

Examples

```r
# create an alphabet object
a <- Alphabet(symbols = c(0, 1, 2, 3, 4, 5))
# get alphabet size
getSize(a)
```
getSizes.DiscreteDeletor

Get the sizes of the proposed deletions

Description

Get the sizes of the proposed deletions.

Usage

```r
## S3 method for class 'DiscreteDeletor'
getSizes(this, ...)
```

Arguments

- `this` A DiscreteDeletor object.
- `...` Not used.

Value

A vector of integers.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `DiscreteDeletor`.

Examples

```r
# create a DiscreteDeletor object
d <- DiscreteDeletor(rate=1)
# set deletion sizes
setSizes(d, c(1, 2, 3))
# get deletion sizes
getSizes(d)
# set/get sizes via virtual field
d$sizes <- 1:10
d$sizes
```
getSizes.DiscreteInsertor

Get the sizes of the proposed insertions

Description

Get the sizes of the proposed insertions.

Usage

```r
## S3 method for class 'DiscreteInsertor'
getSizes(this, ...)
```

Arguments

- `this`: A DiscreteInsertor object.
- `...`: Not used.

Value

A vector of integers.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `DiscreteInsertor`.

Examples

```r
# create a DiscreteInsertor object
i<-DiscreteInsertor(rate=1)
# set insertion sizes
setSizes(i,c(1,2,3))
# get insertion sizes
getSizes(i)
# set/get sizes via virtual field
i$sizes<-1:10
i$sizes
```
**getSite**

Get the current state of a Site object.

### Description

Get the current state of a Site object.

### Usage

```r
## S3 method for class 'Site'
getSite(this, ...)
```

### Arguments

- `this`: A Site object.
- `...`: Not used.

### Value

A character vector of length one containing the state (a symbol belonging to the attached Alphabet object).

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Site`.

### Examples

```r
# create a Site object with an Alphabet object attached
s <- Site(alphabet = Alphabet(symbols = c(0, 1)), state = 1);
# get current state
getSite(s)
# get state via virtual field
s$state
```
getStates.Sequence  

*Get the states of a set of Site objects aggregated by a Sequence object*

**Description**

Get the states of a set of Site objects aggregated by a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
getStates(this, index, ...)
```

**Arguments**

- `this`  
  A Sequence object.
- `index`  
  An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...`  
  Not used.

**Value**

A character vector.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create a sequence object
s <- Sequence(alphabets=list(NucleotideAlphabet()), string="AATTGCCCCCTTGG")
# get all Site states
getStates(s)
# get the states for a collection of sites
getStates(s, c(1:3, 5, 8))
# get states via virtual field
s$states
```
**getString.Sequence**

Get the string representation of a Sequence object

**Description**

Get the string representation of a Sequence object. The string representation is the concatenation of the states of the aggregated Site object. Undefined states (NA-s) are represented by question marks.

**Usage**

```r
## S3 method for class 'Sequence'
getString(this, ...)
```

**Arguments**

- `this` A Sequence object.
- `...` Not used.

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

**Examples**

```r
# create object
s<-Sequence(length=10)
# get character representation
getString(s) # a bunch of '?'-s
# get string representation via virtual field
s$string
```
getSymbolFreqs.Sequence

Get a table with the frequencies of the states of a collection of Site objects aggregated by a Sequence object.

Description

Get a table with the frequencies of the states of a collection of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
getSymbolFreqs(this, index, ...)
```

Arguments

- `this` A Sequence object.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.

Value

A table.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a nucleotide sequence
s <- NucleotideSequence(length = 30, processes = list(list(JC69())))
# sample states
sampleStates(s)
# get state frequencies from ranges 1:10 and 20:30
getSymbolFreqs(s, c(1:10, 20:30))
# get symbol frequencies for the full sequence
getSymbolFreqs(s)
```
getSymbolLength.Alphabet

Get the length of the symbols in a given alphabet object

Description

Get the length of the symbols in a given alphabet object.

Usage

```r
## S3 method for class 'Alphabet'
getSymbolLength(this, ...)
```

Arguments

- `this` An Alphabet object.
- `...` Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `alphabet`.

Examples

```r
# create an alphabet object
a<-Alphabet(symbols=c("AAA","AAC"));
# get symbol length
getSymbolLength(a);
# get symbol length via virtual field
a$symbolLength
```
getSymbols.Alphabet

Get the symbol set from an Alphabet object

Description

Get the symbol set from an Alphabet object.

Usage

## S3 method for class 'Alphabet'
getSymbols(this, ...)

Arguments

this An Alphabet object.

... Not used.

Value

A character vector containing the symbol set of the Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Alphabet.

Examples

# create a new alphabet object
a<-Alphabet(symbols=c("AC","GT"));
# get the symbols
getSymbols(a)
# get the symbols by using the virtual field
a$symbols
getTableId.CodonAlphabet

Get the genetic code id

Description

Get the genetic code id.

Usage

```r
## S3 method for class 'CodonAlphabet'
getableid(this, ...)
```

Arguments

- `this`: A CodonAlphabet object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonAlphabet`.

Examples

```r
# create CodonAlphabet object
a <- CodonAlphabet()
# get genetic code id
getableid(a)
# get genetic code id via virtual field
a$tableId
```
getTemplateSeq.GeneralInsertor

Get the template sequence object

Description

Get the template sequence object. The template sequence object is used by the default \texttt{generateBy} function to generate insert sequences.

Usage

\begin{verbatim}
## S3 method for class 'GeneralInsertor'
getTemplateSeq(this, ...)
\end{verbatim}

Arguments

\begin{verbatim}
this A GeneralInsertor object.
...
Not used.
\end{verbatim}

Value

A Sequence object or NA.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see \texttt{GeneralInsertor}.

Examples

\begin{verbatim}
# create a GeneralInsertor object
i<-GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq=NucleotideSequence(string="AAAAAA")
)
# get template sequence
getTemplateSeq(i)
# get template sequence via virtual field
i$templateSeq
# set template sequence
setTemplateSeq(i, NucleotideSequence(string="C"));
# generate insert
generateInsert(i)
# set template sequence via virtual field
i$templateSeq<-NucleotideSequence(string="G")
\end{verbatim}
getTheta.T92

---

**Description**

Get the GC content.

**Usage**

```r
## S3 method for class 'T92'
getTheta(this, ...)
```

**Arguments**

- `this`: A T92 object.
- `...`: Not used.

**Value**

A numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `T92`.

**Examples**

```r
# construct a T92 object
p<-T92()
# set/get GC content
setTheta(p,0.6)
getTheta(p)
# set/get GC content via virtual field
p$theta<-0.3
p$theta
# get object summary
summary(p)
```
getTipLabels.PhyloSim

Get the tip labels from a phylo object aggregated by a PhyloSim object

Description

Get the tip labels from a phylo object aggregated by a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
getTipLabels(this, ...)
```

Arguments

- `this`: A PhyloSim object.
- `...`: Not used.

Value

A matrix containing the tip labels.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo = rcoal(5));
# get the tip labels
getTipLabels(sim)
# get the tip labels via virtual field
sim$tipLabels
```
getTips.PhyloSim

Get the node identifiers of the tip nodes from a PhyloSim object

Description

Get the node identifiers of the tip nodes from a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
getTips(this, ...)

Arguments

this

A PhyloSim object

... 

Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

# create a PhyloSim object
sim <- PhyloSim(phylo=coal(5));
# get the tip IDs
getTips(sim)
# get the tip IDs via virtual field
sim$tips
Description

Get the tolerance margin.

Usage

## S3 method for class 'FastFieldDeletor'
getToleranceMargin(this, ...) 

Arguments

- **this**: A FastFieldDeletor object.
- **...**: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

Examples

```r
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get tolerance margin
setToleranceMargin(p, 0.8)
getToleranceMargin(p)
# set/get tolerance margin via virtual field
p$toleranceMargin <- 0.75
p$toleranceMargin
```
getTotalRate.Site

Get the total active event rate

Description

Get the total active event rate. The total rate is the sum of the rates of all active events given the current state of the Site object.

Usage

```r
## S3 method for class 'Site'
getTotalRate(this, ...)
```

Arguments

- `this`: A Site object.
- `...`: Not used.

Value

A numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```r
# create a nucleotide site with a JC69 substitution process attached
s <- Site(state="A", alphabet=NucleotideAlphabet(), processes=list(JC69()))
# get the total rate
getTotalRate(s)
# add a new process
attachProcess(s,K80(rate.params=list("Alpha"=1,"Beta"=0.5)))
# get the total rate via virtual field
s$totalRate
```
getTotalRates.Sequence

Get the total site rates from a Sequence object

Description

Get the total site rates from a Sequence object. This method simply calls getTotalRatesFromRange(this). See getTotalRatesFromRange.Sequence for more details.

Usage

```r
## S3 method for class 'Sequence'
getTotalRates(this, ...)
```

Arguments

- `this` A Sequence object.
- `...` Not used.

Value

A numeric vector containing the total site rates.

Author(s)

Botond Sipos, Gregory Jordan

See Also

getTotalRatesFromRange.Sequence

---

generateRatesFromRange.Sequence

Get the vector of total site rates for a collection of Site objects aggregated by a Sequence object

Description

Get the vector of total site rates for a collection of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
generateRatesFromRange(this, index, ...)
```
getTransTable.CodonAlphabet

Arguments

this A Sequence object.
index An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
... Not used.

Value

A numeric vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

# create a sequence with some processes attached
s<-Sequence(
  string="ATGC",
  alphabets=list(NucleotideAlphabet()),
  processes=list(list(JC69()),list(JC69()),GTR())
)

# get total rates for positions 1 and 3
getTotalRatesFromRange(s,c(1,3))
# get all total rates via virtual field
s$totalRates # via the " getTotalRates.Sequence" method

getTransTable.CodonAlphabet

Get the list storing the genetic code table

Description

Get the list storing the genetic code table.

Usage

## S3 method for class 'CodonAlphabet'
getTransTable(this, ...)
Arguments

this          A CodonAlphabet object.
...           Not used.

Value

A list.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonAlphabet.

Examples

# create object
a<-CodonAlphabet()
# get genetic code table
getTransTable(a)
# get genetic code table via virtual field
a$transTable

getTreeLength.PhyloSim

Get the tree length from a PhyloSim object

Description

Get the tree length from a PhyloSim object.
This method returns the sum of the edge lengths stored in the aggregated phylo object.

Usage

## S3 method for class 'PhyloSim'
getTreeLength(this, ...)

Arguments

this          A PhyloSim object.
...           Not used.

Value

A numeric vector of length one.
**Description**

Get Alphabet object type.

**Usage**

```r
## S3 method for class 'Alphabet'
getType(this, ...)
```

**Arguments**

- `this`: An Alphabet object.
- `...`: Not used.

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Alphabet`.
Examples

```r
# create alphabet object
a <- Alphabet(symbols = c(0, 1), type = "Binary");
# get alphabet type
getType(a)
a$type
```

Description

Get the type of the BrownianInsertor process.

If type is discrete, than the process inherits from `DiscreteInsertor`. If the type is continuous, then the object inherits from `ContinuousInsertor`.

Usage

```r
## S3 method for class 'BrownianInsertor'
getType(this, ...)
```

Arguments

- `this` A `BrownianInsertor` object.
- `...` Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `BrownianInsertor`.
Examples

```r
p <- BrownianInsertor(type = "discrete")
getType(p)  # get type
class(p)    # get upstream classes

p <- BrownianInsertor(type = "continuous")
getType(p)  # get type via virtual field
class(p)
```

Description

Get the type of a FastFieldDeletor object.

Usage

```r
## S3 method for class 'FastFieldDeletor'
getType(this, ...)
```

Arguments

- `this`: A FastFieldDeletor object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`. 
getUniqueAlphabets.Sequence

Examples

```r
# create a FastFieldDeletor, default type (geometric)
p <- FastFieldDeletor()
# get type
type(p)
# create a FastFieldDeletor, poisson type
p <- FastFieldDeletor(type = "poisson")
p$type
```

Description

Get the list of unique Alphabet objects associated to Site objects aggregated by a Sequence object.

The returned list contains unique instances of the Alphabet class. The symbol sets are not compared, so two instances of the same class are considered to be different.

Usage

```r
## S3 method for class 'Sequence'
getUniqueAlphabets(this, ...)
```

Arguments

- `this`: A Sequence object.
- `...`: Not used.

Value

A list of Alphabet objects.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.  

Examples

```r
# create a Sequence object with some Alphabet objects attached
s <- Sequence(
  alphabets = list(NucleotideAlphabet(),
                    BinaryAlphabet(),
                    NucleotideAlphabet()),
  length = 10
)

# get the list of attached alphabets
s$alphabets

# get the unique list of attached Alphabet objects
getUniqueAlphabet(s)
```

### Description

Get the list of unique Process instances attached to the Site objects aggregated by a Sequence object.

### Usage

```r
## S3 method for class 'Sequence'
getUniqueProcesses(this, ...)  # A Sequence object.
...
```

### Arguments

- **this**: A Sequence object.
- **...**: Not used.

### Value

A list of Process objects.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `Sequence`. 

getWriteProtected.Alphabet

Examples

# create a sequence object and attach processes
p<-JC69()
s<-Sequence(
  length=4,
  alphabets=list(NucleotideAlphabet()),
  processes=list(list(p,K&O()),list(p))
)
# get the unique list of attached Process instances
getUniqueProcesses(s)
# via virtual field
s$uniqueProcesses

getWriteProtected.Alphabet

Check if the object is write protected

Description

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

## S3 method for class ‘Alphabet’
getWriteProtected(this, ...)

Arguments

this An object.

... Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Alphabet.
Examples

```r
# create an object
o <- Alphabet()
# toggle write protection
o$writeProtected <- TRUE
# check if it's write protected
getWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```

getWriteProtected.Event

*Check if the object is write protected*

**Description**

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

**Usage**

```r
## S3 method for class 'Event'
getWriteProtected(this, ...)
```

**Arguments**

- **this**: An object.
- **...**: Not used.

**Value**

TRUE or FALSE

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see Event.
getWriteProtected.Process

Examples

```r
# create an object
o <- Event()
# toggle write protection
o$writeProtected <- TRUE
# check if it's write protected
getWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```

getWriteProtected.Process

*Check if the object is write protected*

Description

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

```r
## S3 method for class 'Process'
getWriteProtected(this, ...)
```

Arguments

- `this` A Process object.
- `...` Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.
Examples

```r
# create an object
o <- Process()
# toggle write protection
o$writeProtected <- TRUE
# check if it's write protected
getWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```

**getWriteProtected.QMatrix**

*Check if the object is write protected*

**Description**

Check if the object is write protected.

QMatrix objects do not have a write protection flag of their own, but they use the one from the associated Process object. Write protected objects cannot be modified through get/set methods and virtual fields.

**Usage**

```r
## S3 method for class 'QMatrix'
getWriteProtected(this, ...)
```

**Arguments**

- `this` A QMatrix object.
- `...` Not used.

**Value**

TRUE or FALSE

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

getWriteProtected.Process
getWriteProtected.Sequence

Check if the object is write protected

Description

Check if the object is write protected. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

```r
## S3 method for class 'Sequence'
getWriteProtected(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create an object
o<-Sequence()
# toggle write protection
o$writeProtected<-TRUE
# check if it's write protected
getWriteProtected(o)
# check write protection via virtual field
o$writeProtected
```
Description

Check the consistency of all objects inheriting form PSRoot in the current environment.
This method searches for objects which inherit from PSRoot and calls checkConsistency() for all of them, which can take a lots of time. Prints the results of the checks as text.

Usage

```r
## S3 method for class 'PSRoot'
globalConsistencyCheck(...)
```

Arguments

```r
...
```
Not used.

Value

Returns invisible TRUE if all checks are successful.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRoot.

Examples

```r
# create some objects
a<-NucleotideAlphabet()
s<-Site()
p<-Process()
# ask for a global consistency check
PSRoot$globalConsistencyCheck();
```
The GTR class

Description

This class implements the general time-reversible nucleotide substitution model (GTR, REV). The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/95QK2f).

The default value for the rate parameters is 1 and the default value for the base frequencies is 0.25. So the GTR objects are equivalent to JC69 objects by default.

Package:
Class GTR

Object

Directly known subclasses:
F81, F84, HKY, K80, K81, T92, TN93

public static class GTR
extends UNREST

Usage

GTR(name="Anonymous", rate.params=list(a = 1, b = 1, c = 1, d = 1, e = 1, f = 1),
    base.freqs=rep(0.25, times = 4), ...)

Arguments

name Object name.
rate.params A list of unscaled rates (see setRateList.GrandSubstitution).
base.freqs Equilibrium distribution.
... Additional arguments.
Fields and Methods

Methods:

- checkConsistency
- getBaseFreqs
- getRateParam
- getRateParamList
- setBaseFreqs
- setRateParam
- setRateParamList
- summary

Methods inherited from UNREST:
- checkConsistency, summary

Methods inherited from GeneralSubstitution:
- as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
- !=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, setSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
- checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
- $, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

GeneralSubstitution UNREST HKY
Examples

```r
# create substitution process object
p <- GTR(
  rate.params = list(
    "a" = 1, "b" = 2, "c" = 3,
    "d" = 1, "e" = 2, "f" = 3
  ),
  base.freqs = c(2, 2, 1, 1) / 6
)
  # get a summary
  summary(p)
  # display a bubble plot
  plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.

  # create a sequence, attach process p
  s <- NucleotideSequence(length = 20, processes = list(list(p)))
  # sample states
  sampleStates(s)
  # make the first five positions invariable
  setRateMultipliers(s, p, 0, 1:5)
  # get rate multipliers
  getRateMultipliers(s, p)
  # create a simulation object and run simulation
  sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
  Simulate(sim)
  # print alignment
  sim$alignment
```

GY94

The GY94 class

Description

This class implements the codon substitution model of Goldman and Yang (1994). The transition/transversion rate ratio is stored in the kappa virtual field. The nonsynonymous/synonymous substitution rate ratio (omega) is a site-process specific parameter with a default value of one. Hence, after the attachment of the process the variation of omega ratios among sites follows the M0 model (see Yang et al. 2000).

The rate matrix of the GY94 model is scaled in a way that the expected number of potential substitutions per site is equal to one at equilibrium. The codeml program from the PAML package scales the rate matrix in order to have the expected number of accepted substitutions per site equal to one. Use the getOmegaScalingFactor.GY94 method to calculate a branch length scaling factor which allows to switch to a PAML-style scaling given an average omega.

If the scale.nuc constructor argument is TRUE, the rates of the returned Event objects will be multiplied by 3 to obtain a process which has the expected number of nucleotide substitutions (not
codon substitutions) equal to one at equilibrium. This is useful when simulating mixed sequences. This option doesn’t affect the rate matrix in any way.

The M1-M4 models are implemented in the omegaVarM[1-4].CodonSequence methods. Simulations under more complex models (M5-M13) can be achieved by first discretizing them using the M5-13 tool from the INDELible software package (http://abacus.gene.ucl.ac.uk/software/indelible/). After discretization, the M5-M13 models can be simulated through the M3 (discrete) model.

Package:
Class GY94

```
Object
|--
    |--PSRoot
    |   |--Process
    |      |--GeneralSubstitution
    |      |--CodonUNREST
    |         |--GY94

Directly known subclasses:

public static class GY94
extends CodonUNREST

Usage

GY94(name="Anonymous", table.id=1, kappa=1, omega.default=1, codon.freqs=NA,
      scale.nuc=FALSE, ...)

Arguments

name          Name of the object.
table.id      The identifier of the genetic code table to use (1 by default).
kappa        The transition/transversion rate ratio (1 by default).
omega.default The default value of the omega site-process specific parameter (1 by default).
codon.freqs   A vector of codon frequencies.
scale.nuc     Scale to nucleotide substitutions if TRUE (see above).
...           Additional arguments.
Fields and Methods

Methods:

- checkConsistency
- getCodonFreqs
- getEventsAtSite
- getKappa
- getOmegaScalingFactor
- is
- setCodonFreqs
- setKappa
- setRate
- summary

Methods inherited from CodonUNREST:
- checkConsistency, is

Methods inherited from GeneralSubstitution:
- as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
- !, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
- checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
- $, $<-, [,. ,[[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References


hasSiteSpecificParameter.Process

Check if a Process object has the site-process specific parameter with the given id.

Description

Check if a Process object has the site-process specific parameter with the given id.

Usage

```r
## S3 method for class 'Process'
hasSiteSpecificParameter(this, id, ...)
```
hasSymbols.Alphabet

Arguments

- `this` A Process object.
- `id` The identifier of the site-process specific parameter of interest.
- ... Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Process`.

Examples

```r
# create a process object
p <- Process()
# check whether it has the "rate.multiplier" site-process specific parameter
hasSiteSpecificParameter(p, "rate.multiplier"); # TRUE
# check whether it has the "omega" site-process specific parameter
hasSiteSpecificParameter(p, "omega"); # FALSE
```

hasSymbols.Alphabet Check if an Alphabet object has a given set of symbols

Description

Check if an Alphabet object has a given set of symbols. Returns true if the class of the object is "AnyAlphabet".

Usage

```r
## S3 method for class 'Alphabet'
hasSymbols(this, sym, ...)
```

Arguments

- `this` An Alphabet object.
- `sym` A character vector.
- ... Not used.
hasUndefinedRate. GeneralInDel

Check whether the general rate of a GeneralInDel object is undefined

Description

Check whether the general rate of a GeneralInDel object is undefined.

Usage

```r
## S3 method for class 'GeneralInDel'
hasUndefinedRate(this, ...)
```

Arguments

- **this**: A GeneralInDel object.
- **...**: Not used.

Value

TRUE or FALSE.
Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralInDel.

Examples
# create a GeneralInDel object
o<GeneralInDel()
# check if the general rate is undefined
hasUndefinedRate(o)
# set general rate
o$rate<-1
# check rate again
hasUndefinedRate(o)

Description
Check if a GeneralSubstitution object has undefined rates.

Usage
## S3 method for class 'GeneralSubstitution'
hasUndefinedRate(this, ...)

Arguments
this A GeneralSubstitution object.
... Not used.

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see GeneralSubstitution.
Examples

# create a GeneralSubstitution object
p<-GeneralSubstitution(alphabet=BinaryAlphabet())
# check if it has undefined rates
hasUndefinedRate(p) # TRUE
# set the missing rates
p$dRateList<-list("0->1"=1,"1->0"=2)
# check for undefined rates again
hasUndefinedRate(p) # FALSE

Description

Check if the Process object has undefined rate parameters.

For the instances of the Process class this method always returns FALSE. Descendant classes should implement more meaningful methods.

Usage

## S3 method for class 'Process'
hasUndefinedRate(this, ...)

Arguments

this A Process object.

... Not used.

Value

FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.
Examples

```r
# create object
p <- Process()
# check if has undefined rates
hasUndefinedRate(p) # return FALSE
```

### hasUndefinedRate.ToleranceSubstitution

*Check if a ToleranceSubstitution object has undefined rates*

#### Description
Check if a ToleranceSubstitution object has undefined rates.

#### Usage

```r
## S3 method for class 'ToleranceSubstitution'
hasUndefinedRate(this, ...)
```

#### Arguments

- `this`: A ToleranceSubstitution object.
- `...`: Not used.

#### Value
TRUE or FALSE.

#### Author(s)
Botond Sipos, Gregory Jordan

#### See Also
For more information see `ToleranceSubstitution`.

#### Examples

```r
# create a ToleranceSubstitution object
p <- ToleranceSubstitution(alphabet=BinaryAlphabet())
# check if it has undefined rates
hasUndefinedRate(p) # TRUE
# set the missing rates
p$rateList <- list("0->1"="1","1->0"="2")
# check for undefined rates again
hasUndefinedRate(p) # FALSE
```
The HKY class

Description

This class implements the HKY GTR-submodel. The rate parameters are the following: "Alpha", "Beta". Package: 

Class HKY

Usage

```r
HKY(name="Anonymous", rate.params=list(Alpha = 1, Beta = 1), 
    base.freqs=c(0.25, 0.25, 0.25, 0.25), ...)
```

Arguments

- `name` Object name.
- `rate.params` Rate parameters.
- `base.freqs` Base frequency parameters.
- ... Not used.

Fields and Methods

Methods:
Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR UNREST GeneralSubstitution TN93
Examples

# create substitution process object
p<-HKY(rate.params=list( "Alpha"=10,"Beta"=2),
base.freqs=c(4,3,2,1)/10)
  # get a summary
  summary(p)
# display a bubble plot
  plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.
# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
  sampleStates(s)
  # make the first five positions invariable
  setRateMultipliers(s,p,0,1:5)
  # get rate multipliers
  getRateMultipliers(s,p)
  # create a simulation object
  sim<-PhyloSim(root.seq=s,phylo=rcoal(2))
  # run simulation
  Simulate(sim)
  # print alignment
  sim$alignment

insertSequence.Sequence

Insert a Sequence object into another Sequence object after a specified position

Description

Insert a Sequence object into another Sequence object after a specified position.

Usage

## S3 method for class 'Sequence'
insertSequence(this, insert, position, process=NA, paranoid=FALSE, ...)

Arguments

this: The target Sequence object.
insert: The Sequence object to be inserted.
position: The position after the Sequence object will be inserted.
process The Process object performing the insertion (optional).
paranoid If TRUE, then the consistency of the target objects is checked more rigorously after insertion.
... Not used.

Value
The Sequence object (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Sequence.

Examples
# create some sequence objects.
target<-NucleotideSequence(string="AAAAAAAAAA")
insert<-NucleotideSequence(string="GGGGGGGGGG")
# insert after position 5
insertSequence(target,insert,5)
# print the target sequence
target

Description
Utility method returning the intersection of two lists. Duplicated elements are not considered.

Usage
### S3 method for class 'PSRoot'
intersect.list(this, one, two, ...)

Arguments
- **this** A PSRoot object.
- **one** A list of objects.
- **two** A list of objects.
- ... Not used.
Value
The list containing the intersection.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see psroot.

Examples
# create some lists
a<-list(1,2,3);
b<-c(a,list("a","b","c"))
# get the intersection of a and b
PSRoot$intersect.list(a,b)

---

is.Alphabet

Check if an object is an instance of the Alphabet class

Description
Check if an object is an instance of the Alphabet class.

Usage
```r
## Default S3 method:
is.Alphabet(this, ...)
```

Arguments
- **this**: An object.
- **...**: Not used.

Value
TRUE or FALSE

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Alphabet.
Examples

```r
# create an alphabet object
a <<- Alphabet()
# create a PSRoot object
o <<- PSRoot()
# check if they are alphabet objects
is.Alphabet(a)
is.Alphabet(o)
```

Description

Check if an object inherits from `CodonAlphabet`.

Usage

```r
## Default S3 method:
is.CodonAlphabet(this, ...)
```

Arguments

- `this`  
  An object.
- `...`  
  Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

Examples

```r
# create some objects
a <<- CodonAlphabet()
p <<- Process()
# check if they inherit from CodonAlphabet
is.CodonAlphabet(a)
is.CodonAlphabet(p)
```
is.CodonUNREST  
*Check whether an object inherits from CodonUNREST*

**Description**
Check whether an object inherits from CodonUNREST.

**Usage**
```r
## Default S3 method:
is.CodonUNREST(this, ...)
```

**Arguments**
- `this` An object.
- `...` Not used.

**Value**
TRUE or FALSE.

**Author(s)**
Botond Sipos, Gregory Jordan

**Examples**
```r
# create some objects
p<-CodonUNREST()
pp<-GTR()
# check if they inherit from CodonUNREST
is.CodonUNREST(p)
is.CodonUNREST(pp)
```

---

is.Event  
*Check whether an object inherits from the class Event*

**Description**
Check whether an object inherits from the class Event.

**Usage**
```r
## Default S3 method:
is.Event(this, ...)
```
is.GeneralDeletor

Arguments

this      An object.
...      Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

# create some objects
e<-Event(); a<-Alphabet()
# check if they inherit from Event
is.Event(e)
is.Event(a)

is.GeneralDeletor  Check whether an object inherits from GeneralDeletor

Description

Check whether an object inherits from GeneralDeletor.

Usage

## Default S3 method:
is.GeneralDeletor(this, ...)

Arguments

this      An object.
...      Not used.

Value

TRUE or FALSE.
Author(s)

Botond Sipos, Gregory Jordan

Examples

```r
# create some objects
d <- GeneralDeleter()
i <- GeneralInsertor()
# check if they inherit from GeneralDeleter
is.GeneralDeleter(d)
is.GeneralDeleter(i)
```

is.GeneralInDel | Check if an object inherits from the GeneralInDel class

Description

Check if an object inherits from the GeneralInDel class.

Usage

```r
## Default S3 method:
is.GeneralInDel(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

Examples

```r
# create some objects
o <- GeneralInDel(rate=1, propose.by=function(process){sample(c(1:10),1)});
x <- GTR()
# check if they inherit from GeneralInDel
is.GeneralInDel(o)
is.GeneralInDel(x)
```
is GeneralInsertor

Check whether an object inherits from GeneralInsertor

Description

Check whether an object inherits from GeneralInsertor.

Usage

```r
## Default S3 method:
is.GeneralInsertor(this, ...)
```

Arguments

this An object.
...
Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

Examples

```r
# create some objects
d <- GeneralDeleter()
i <- GeneralInsertor()
# check if they inherit from GeneralInsertor
is.GeneralInsertor(i)
is.GeneralInsertor(d)
```

is GeneralSubstitution

Check if an object is an instance of the GeneralSubstitution class

Description

Check if an object is an instance of the GeneralSubstitution class.

Usage

```r
## Default S3 method:
is.GeneralSubstitution(this, ...)
```
Arguments

this  An object.
...  Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

# create some objects
p<-GeneralSubstitution()
pp<-Process()
# check if they inherit from GeneralSubstitution
is.GeneralSubstitution(p)
is.GeneralSubstitution(pp)

---

is.GY94  Check whether an object inherits from GY94

Description

Check whether an object inherits from GY94.

Usage

## Default S3 method:
is.GY94(this, ...)

Arguments

this  An object.
...  Not used.

Value

TRUE or FALSE.
is.na.PSRoot

Author(s)
Botond Sipos, Gregory Jordan

Examples

# create some objects
p <- CodonUNREST()
pp <- GY94()
# check if they inherit from CodonUNREST
is.GY94(p)
is.GY94(pp)

is.na.PSRoot  Check if a PSRoot object is NA

Description
Check if a PSRoot object is NA. PSRoot objects accanot be NA, so this method always returns FALSE.

Usage

## S3 method for class 'PSRoot'
is.na(x, ...)

Arguments

x A PSRoot object.
...
Not used.

Value
FALSE

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PSRoot.

Examples

is.na(PSRoot());
is.phylo

Check if an object is an instance of the phylo class

Description

Check if an object is an instance of the phylo class. Phylo objects are created by the APE package. This method just return the value of inherits(this,"phylo").

Usage

## Default S3 method:
is.phylo(this, ...)

Arguments

this An object.

... Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

The ape package.

Examples

# load APE
library(ape);
# create some objects
o1<-Object();
o2<-rcoal(3);
# check if they are phylo objects
is.phylo(o1);
is.phylo(o2);
is.Process  

**Check if an object is an instance of the Process class**

**Description**

Check if an object is an instance of the Process class.

**Usage**

```r
## Default S3 method:
is.Process(this, ...)
```

**Arguments**

- `this`: An object.
- `...`: Not used.

**Value**

TRUE or FALSE

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Process`.

**Examples**

```r
# create some objects
a <- Site();
p <- Process()
# check if they inherit from Process
is.Process(a)
is.Process(p)
```
is.PSRoot

Check if an object inherits from PSRoot

Description

Check if an object inherits from PSRoot.

Usage

```r
## Default S3 method:
is.PSRoot(this, ...)
```

Arguments

- `this` An object.
- `...` Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

Examples

```r
# create some objects
o <- PSRoot()
a <- Alphabet()
x <- Object()
# check if they inherit form PSRoot
is.PSRoot(o)
is.PSRoot(a)
is.PSRoot(x)
```
is.QMatrix

Description

Check if an object is an instance of the QMatrix class.

Usage

## Default S3 method:
is.QMatrix(this, ...)

Arguments

this  An object.
...
Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.

Examples

# create some objects
m<-QMatrix()
p<-Process()
# check if they inherit from QMatrix
is.GeneralSubstitution(m)
is.GeneralSubstitution(p)
is.Sequence  

Check whether an object inherits from the Sequence class

Description

Check whether an object inherits from the Sequence class.

Usage

```r
## Default S3 method:
is.Sequence(this, ...)
```

Arguments

- `this` An object.
- `...` Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `sequence`.

Examples

```r
# create some objects
seq<-Sequence(length=10)
a<-Alphabet()
# check if they inherit from Sequence
is.Sequence(seq)
is.Sequence(a)
```
is.Site

Check if an object is an instance of the Site class

Description

Check if an object is an instance of the Site class.

Usage

```r
## Default S3 method:
is.Site(this, ...)
```

Arguments

- `this` An object.
- `...` Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

Examples

```r
# create an object
s<-Site();
# check whether is a Site object
is.Site(s)
# the same with an Event object
is.Site(Event());
```
is.tip.PhyloSim

Check if a node is a tip

Description

Check if a node is a tip.

Usage

```r
## S3 method for class 'PhyloSim'
is.tip(this, node=NA, ...)
```

Arguments

- `this` A PhyloSim object.
- `node` A node identifier (integer vector of length one).
- `...` Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `phylosim`.

Examples

```r
# create a PhyloSim object
sim<-PhyloSim(phylo=rcoal(5));
# check if node 4 is a tip
is.tip(sim,4)
# check if node 6 is a tip
is.tip(sim,6)
```
is.ToleranceSubstitution

Check if an object is an instance of the ToleranceSubstitution class

Description

Check if an object is an instance of the ToleranceSubstitution class.

Usage

```r
## Default S3 method:
is.ToleranceSubstitution(this, ...)
```

Arguments

- `this`: An object.
- `...`: Not used.

Value

TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ToleranceSubstitution`.

Examples

```r
# create some objects
p <- ToleranceSubstitution()
pp <- Process()
# check if they inherit from ToleranceSubstitution
is.ToleranceSubstitution(p)
is.ToleranceSubstitution(pp)
```
isAttached.Site

Check whether a Process object is attached to a Site object

Description
Check whether a Process object is attached to a Site object.

Usage
## S3 method for class 'Site'
isAttached(this, process, ...)

Arguments
this
A Site object.
process
A Process object.
...
Not used.

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
Site Process attachProcess detachProcess getProcesses setProcesses

Examples
# create a Site object
s<-Site(alphabet=NucleotideAlphabet())
# create a Process object
p<-JC69()
# check if p is attached to s
isAttached(s,p)
# attach p to s
s$processes<-list(p)
isAttached(s,p)
isEmpty.Alphabet

Check if the symbol set is empty

Description

Check if the symbol set is empty.

Usage

```r
# S3 method for class 'Alphabet'
isEmpty(this, ...)
```

Arguments

- `this`: An Alphabet object.
- `...`: Not used.

Value

TRUE or FALSE

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Alphabet`.

Examples

```r
# create an empty alphabet
a <- Alphabet()
# check whether it is empty
isEmpty(a)
# specify a new symbol set
a$symbols <- c(0, 1)
isEmpty(a)
```
isStartCodon.CodonAlphabet

*Check if a codon is a start codon*

**Description**

Check if a codon is a start codon.

**Usage**

```r
## S3 method for class 'CodonAlphabet'
isStartCodon(this, codon, ...)
```

**Arguments**

- `this`: A CodonAlphabet object.
- `codon`: The codon to be checked.
- `...`: Not used.

**Value**

TRUE or FALSE.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonAlphabet`.

**Examples**

```r
# create a CodonAlphabet object
a<-CodonAlphabet()
# check some codons
isStartCodon(a,"ATG")
isStartCodon(a,"TGA")
```
isStopCodon(CodonAlphabet)

*Check if a codon is a stop codon*

**Description**

Check if a codon is a stop codon.

**Usage**

```r
## S3 method for class 'CodonAlphabet'
isStopCodon(this, codon, ...)
```

**Arguments**

- `this`: A CodonAlphabet object.
- `codon`: The codon to be checked.
- `...`: Not used.

**Value**

TRUE or FALSE.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonAlphabet`.

**Examples**

```r
# create a CodonAlphabet object
a<-CodonAlphabet()
# check some codons
isStopCodon(a,"ATG")
isStopCodon(a,"TGA")
```
The JC69 class

Description

This class implements Jukes-Cantor nucleotide substitution model.

Package:

Class JC69

Object

~|~
~|---PSRoot

~~~~~~~|~
~~~~~~~~~|---Process

~~~~~~~~||
~~~~~~~~~~~~~|---GeneralSubstitution

~~~~~~~~~~~~~~~~|---UNREST

~~~~~~~~~~~~~~~~||
~~~~~~~~~~~~~~~~~~~|---JC69

Directly known subclasses:

public static class JC69
extends UNREST

Usage

JC69(name="Anonymous", ...)

Arguments

name Object name.

... Additional arguments.

Fields and Methods

Methods:

checkConsistency -
summary -
Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GeneralSubstitution UNREST GTR

Examples
p<-JC69()
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
The JTT empirical amino acid substitution model

Description

Package: Class JTT

Object

```
| ~--PSRoot
| ~~~~~~|--Process
| ~~~~~~~~~|--GeneralSubstitution
| ~~~~~~~~~~|--AminoAcidSubst
| ~~~~~~~~~~~~~|--JTT
```

Directly known subclasses:

- public static class JTT
- extends AminoAcidSubst

Usage

```
JTT(equ.dist=NA, ...)
```

Arguments

- `equ.dist` Equilibrium distribution.
- `...` Not used.
Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldName, getFieldNameModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples

# create substitution model object
p<-JTT()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.
# create a sequence, attach process p
s <- AminoAcidSequence(length=10, processes=list(list(p)))

# sample states
sampleStates(s)
    # make the first three positions invariable
    setRateMultipliers(s, p, 0, 1:3)
# get rate multipliers
getRateMultipliers(s, p)
# create a simulation object
sim <- PhyloSim(root.seq=s, phylo=rcoal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment

---

**JTT.dcmut**  
*The JTT.dcmut empirical amino acid substitution model*

---

**Description**

Package:

Class **JTT.dcmut**

---

**Object**

```
~~|
~~---PSRoot
~~~~~~|  
~~~~~~~~---Process
~~~~~~~~~~~~~~|    
~~~~~~~~~~~~~~~---GeneralSubstitution
~~~~~~~~~~~~~~~|  
~~~~~~~~~~~~~~~~____--AminoAcidSubst
~~~~~~~~~~~~~~~~|  
~~~~~~~~~~~~~~~~~~~---JTT.dcmut

```

**Directly known subclasses:**

```
public static class JTT.dcmut
extends AminoAcidSubst
```

**Usage**

```
JTT.dcmut(equ.dist=NA, ...
```
Arguments

equ.dist  Equilibrium distribution.
...  Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

AminoAcidSubst GeneralSubstitution UNREST
Examples

```r
# create substitution model object
p<-JTT.dcmut()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first threee positions invariable
setRateMultipliers(s,p,0,1:3)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcol(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**K80**  
*The K80 class*

**Description**

This class implements the K80 (Kimura 2-parameter) GTR-submodel. The rate parameters are the following: "Alpha", "Beta". Package:

Class K80

```r
Object
  ~~~|
  ~~~~PSRoot
  ~~~~~|
  ~~~~~~~~~~Process
  ~~~~~~~~~~~~~|
  ~~~~~~~~~~~~~~GeneralSubstitution
  ~~~~~~~~~~~~~~~~~|
  ~~~~~~~~~~~~~~~~~~UNREST
  ~~~~~~~~~~~~~~~~~~|
  ~~~~~~~~~~~~~~~~~~~GTR
  ~~~~~~~~~~~~~~~~~~~|
```
Directly known subclasses:

public static class K80
extends GTR

Usage

K80(name="Anonymous", rate.params=list(Alpha = 1, Beta = 1), ...)

Arguments

name Object name.
rate.params Rate parameters.
... Not used.

Fields and Methods

Methods:

checkConsistency -
getBaseFreqs -
getRateParam -
getRateParamList -
setBaseFreqs -
setRateParam -
setRateParamList -
summary -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSite-
SpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [[-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR UNREST GeneralSubstitution TN93

Examples

# create substitution process object
p<-K80(rate.params=list("Alpha"=6,"Beta"=2),
base.freqs=c(4,3,2,1)/10
)
   # get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
gRateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoal(2))
# run simulation
Simulate(sim)
The K81 class

Description

This class implements the K81 (Kimura 3-parameter) GTR-submodel.

The rate parameters are the following: "Alpha", "Beta", "Gamma". Package: Class K81

Object
  ~|  
  ~--PSRoot
  ~~~~~|  
  ~~~~~~~~~Process
  ~~~~~~~~~|
  ~~~~~~~~~~GeneralSubstitution
  ~~~~~~~~~|
  ~~~~~~~~~~UNREST
  ~~~~~~~~~|
  ~~~~~~~~~~GTR
  ~~~~~~~~~|
  ~~~~~~~~~~~K81

Directly known subclasses:

public static class K81
extends GTR

Usage

K81(name="Anonymous", rate.params=list(Alpha = 1, Beta = 1, Gamma = 1), ...)

Arguments

name          Object name.
rate.params   Rate parameters.
...           Not used.
Fields and Methods

   Methods:
checkConsistency -
getBaseFreqs -
getRateParam -
getRateParamList -
setBaseFreqs -
setRateParam -
setRateParamList -
summary -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [, [<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR UNREST GeneralSubstitution HKY
Examples

```r
# create substitution process object
p <- K81(rate.params=list("Alpha"=10,"Beta"=2,"Gamma"=5))
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- NucleotideSequence(length=20, processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim <- PhyloSim(root.seq=s, phylo=rcoal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**LG**

The LG empirical amino acid substitution model

---

Description

Package: Class LG

```
Object
~| ~--PSStructure
~~| ~--Process
~~~~| ~--GeneralSubstitution
~~~~~~| ~--AminoAcidSubstitution
~~~~~~~| ~--LG
```
Directly known subclasses:

public static class LG
extends AminoAcidSubst

Usage

LG(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
...

Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldValue, getFieldValueModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

References

See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p <- LG()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length=10, processes=list(list(p)))
# sample states
sampleStates(s)

# make the first three positions invariable
setRateMultipliers(s, p, 0, 1:3)
# get rate multipliers
getRateMultipliers(s, p)
# create a simulation object
sim <- PhyloSim(root.seq=s, phylo=coal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**ll.PSRoot**

Display detailed information about the virtual fields and methods defined for a given object

---

**Description**

Display detailed information about the virtual fields and methods defined for a given object. The method prints the class of the object, all the parent classes, and the virtual fields and methods defined in the immediate class.

This method provides a "quick and minimal" documentation for PhyloSim classes.

**Usage**

```r
## S3 method for class 'PSRoot'
ll(this, quiet=FALSE, ...)
```
**Arguments**

- **this**  
  A PSRoot object.
- **quiet**  
  Do not print out methods list.
- **...**  
  Not used.

**Value**

Text.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [PSRoot](#).

**Examples**

```r
# create a Site object
s <- Site()
ll(s)
# get information about the Process class
ll(Process())
```

---

**Log.PhyloSim**  
*Save a message in the PhyloSim log file*

**Description**

Save a message in the PhyloSim log file.

The message is written to the log file only if the log level is non-negative. You can use this method for logging in the case you write classes for PhyloSim.

**Usage**

```r
## S3 method for class 'PhyloSim'
Log(this, message, ...)
```

**Arguments**

- **this**  
  A PhyloSim object.
- **message**  
  A character vector of length one.
- **...**  
  Not used.
Value

The message (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

Examples

```r
# create a PhyloSim object,
# with logLevel set to zero
sim <- PhyloSim(log.level = 0);
# log a message
Log(sim, "Hiya there!");
# close log connection
close(sim$log.connection)
# print out the log file
cat(paste(scan(file = sim$LogFile, what = character(), sep = "\n"), collapse = "\n"));
cat("\n");
# clean up
unlink(sim$LogFile)
```

---

**mtArt**

The mtArt empirical amino acid substitution model

Description

Package:

Class mtArt

```
Object
---
---<---PSRoot
--------|---
----------<---Process
-------------|---
----------------<---GeneralSubstitution
---------------------------|---
-------------------------------<---AminoAcidSubst
---------------------------------|---
-----------------------------------------------<---mtArt
```
Directly known subclasses:

public static class mtArt
extends AminoAcidSubst

Usage

mtArt(equ.dist=NA, ...)

Arguments

equ.dist    Equilibrium distribution.
...         Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

References

The mtMam empirical amino acid substitution model

See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p <- mtArt()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length=10, processes=list(list(p)))
# sample states
sampleStates(s)
    # make the first three positions invariant
    setRateMultipliers(s, p, 0:3)
    # get rate multipliers
    getRateMultipliers(s, p)
    # create a simulation object
    sim <- PhyloSim(root_seq=s, phylo=rc Oakland)
    # run simulation
    Simulate(sim)
    # print alignment
    sim$alignment
```

Description

Package:

Class mtMam

Object

```
| PSRoot
| ~~~~
| GeneralSubstitution
| ~~~~~~~~~~~~~
| AminoAcidSubst
```
Directly known subclasses:

public static class mtMam
extends AminoAcidSubst

Usage

mtMam(equ.dist=NA, ...)

Arguments

equ.dist            Equilibrium distribution.
...                  Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan
References


See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```
# create substitution model object
p<-mtMam()
# get object summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )
# sample states
sampleStates(s)
  # make the first three positions invariable
  setRateMultipliers(s,p,0,1:3)
  # get rate multipliers
  getRateMultipliers(s,p)
  # create a simulation object
  sim<-PhyloSim(root.seq=s,phylo=coal(2))
  # run simulation
  Simulate(sim)
  # print alignment
  sim$alignment
```

mtREV24

*The mtREV24 empirical amino acid substitution model*

Description

Package: mtREV24

Class mtREV24

Object

```---}
```
Directly known subclasses:

public static class **mtREV24**
extends AminoAcidSubst

Usage

```r
mtREV24(equ.dist=NA, ...)
```

Arguments

equ.dist Equilibrium distribution.

... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, setRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventRateAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$S, S<-, [I, [I<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach,
equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples
# create substitution model object
p<-mtREV24()
# get object summary
summary(p)
# display a bubble plot
plot(p)

   # The following code demonstrates how to use
   # the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )
# sample states
sampleStates(s)
   # make the first three positions invariable
setRateMultipliers(s,p,0:1:3)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcal(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
The MtZoa empirical amino acid substitution model

Description

Package:
Class MtZoa

Object
~~|  
~~+-PSRoot  
~~~~~~|  
~~~~~~~~--Process  
~~~~~~~~~~~~~|  
~~~~~~~~~~~~++--GeneralSubstitution  
~~~~~~~~~~~~~~|  
~~~~~~~~~~~~~~++--AminoAcidSubst  
~~~~~~~~~~~~~~~~|  
~~~~~~~~~~~~~~~~++--MtZoa

Directly known subclasses:

public static class MtZoa  
extends AminoAcidSubst

Usage

MtZoa(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary
Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [., [.[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples

# create substitution model object
p<-MtZoa()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )
# sample states
sampleStates(s)
    # make the first three positions invariable
setRateMultipliers(s,p,0,1:3)
    # get rate multipliers
getRateMultipliers(s,p)
    # create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoal(2))
my.all.equal.PSroot | Test if two objects are nearly equal

Description
Test if two objects are nearly equal.

This method simply calls `all.equal.default` with the tolerance parameter set to `.Machine$double.eps ^ 0.5`. More useful as a static method.

Usage
```r
## S3 method for class 'PSRoot'
my.all.equal(static, target, current, ...)
```

Arguments
- `static` A PSRoot object.
- `target` R object.
- `current` Other R object, to be compared with target.
- `...` Not used.

Value
TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `PSRoot`.

Examples
```r
PSRoot$my.all.equal(0.0, 0.0001)
PSRoot$my.all.equal(0.0, 0.000000001)
```
newAAMatrix.AminoAcidSubst

Undocumented method

Description

Undocumented method.

Usage

```r
## S3 method for class 'AminoAcidSubst'
newAAMatrix(name=NA, paml.file=NA, equ.dist=NA, ...)
```

Arguments

- `name`: Object name
- `paml.file`: PAML file.
- `equ.dist`: Equilibrium distribution.
- `...`: Not used.

Value

A process object inheriting from AminoAcidSubst.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `AminoAcidSubst`.

---

newMatrix.CodonSubst

New codon substitution matrix from PAML file

Description

New codon substitution matrix from PAML file.

Usage

```r
## S3 method for class 'CodonSubst'
newMatrix(name=NA, paml.file=NA, equ.dist=NA, ...)
```
**NucleotideAlphabet**

**Arguments**

- **name**: Object name
- **paml.file**: PAML file.
- **equ.dist**: Equilibrium distribution.
- **...**: Not used.

**Value**

A process object inheriting from CodonSubst.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonSubst`.

---

**NucleotideAlphabet**  
*The NucleotideAlphabet class*

**Description**

Class of Alphabet objects with the c("T","C","A","G") symbol set, representing nucleotides.

**Package:**  
Class NucleotideAlphabet

```
Object
|   ~
|   ~---PSRoot
|   ~~~~~~|
|   ~~~~~~~---Alphabet
|   ~~~~~~~~~|
|   ~~~~~~~~~---NucleotideAlphabet
```

**Directly known subclasses:**

public static class NucleotideAlphabet extends Alphabet

**Usage**

NucleotideAlphabet(...)

Arguments

... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from Alphabet:
!=, ==, as.character, checkConsistency, getSize, getSymbolLength, getSymbols, getType, getWriteProtected, hasSymbols, is, isEmpty, setSize, setSymbolLength, setSymbols, setType, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, ][<-., as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

See Also

Alphabet

Examples

# create a nucleotide alphabet
b<-NucleotideAlphabet()
# get alphabet summary
summary(b)

---

NucleotideSequence The NucleotideSequence class

Description

Sequence objects aggregating Site objects having a NucleotideAlphabet attached by default.

Package:

Class NucleotideSequence

Object ~|
Directly known subclasses:

```java
public static class NucleotideSequence extends Sequence
```

Usage

```java
NucleotideSequence(name=NA, string=NA, length=NA, processes=NA, ancestral.obj=NA, ...)
```

Arguments

- **name**: Name of the Sequence object.
- **string**: A string specifying the length and the states of the Sequence object.
- **length**: The length of the sequence. Mutually exclusive with "string".
- **processes**: A list of lists of Process objects, to be attached to the aggregated Site objects. Recycled if shorter than the length of the sequence.
- **ancestral.obj**: The ancestral object of the Sequence object (a valid Sequence or Process object).
- **...**: Not used.

Fields and Methods

Methods:

- `revComp` -

Methods inherited from `Sequence`:
- `as.character`, `attachProcess`, `checkConsistency`, `clearStates`, `clone`, `copySubSequence`, `deleteSubSequence`, `detachProcess`, `getAlphabets`, `getAncestral`, `getBigRate`, `getCumulativeRates`, `getCumulativeRatesFromRange`, `getDeletionTolerance`, `getEvents`, `getId`, `getInsertionTolerance`, `getLength`, `getName`, `getOmegas`, `getParameterAtSites`, `getProcesses`, `getRateMultipliers`, `getSites`, `getStates`, `getString`, `getSymbolFreqs`, `getTotalRates`, `getTotalRatesFromRange`, `getUniqueAlphabets`, `getUniqueProcesses`, `getWriteProtected`, `insertSequence`, `is`, `plot`, `plotParametersAtSites`, `plusGamma`, `plusInvGamma`, `sampleStates`, `setAlphabets`, `setAncestral`, `setBigRate`, `setCumulativeRates`, `setDeletionTolerance`, `setId`, `setInsertionTolerance`, `setLength`, `setName`, `setOmegas`, `setParameterAtSites`, `setProcesses`, `setRateMultipliers`, `setStates`, `setString`, `setTotalRates`, `setUniqueAlphabets`, `setUniqueProcesses`, `setWriteProtected`, `summary`

Methods inherited from `PSRoot`: 

checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Sequence NucleotideAlphabet

Examples

`# create an empty NucleotideSequence object`
`s<-NucleotideSequence(length=50)`
`s`
`# set states`
`s$states<-c("A","A","G","T")`
`s`
`# create a sequence object by specifying a string`
`s<-NucleotideSequence(string="ATGCCGATTAGAAA")`
`s`

---

omegaHist.CodonSequence

*Plot a histogram of omega values from a range*

Description

Plot a histogram of omega values from a range.

Usage

```r
## S3 method for class 'CodonSequence'
omegaHist(this, process, breaks, index, ...)
```

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **breaks**: breaks parameter for hist().
- **index**: A vector of positions.
- **...**: Not used.
Value

The CodonSequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSequence.

Examples

```r
# create a GY94 process
p<-GY94()
# create a CodonSequence object,
# attach a process p
s<-CodonSequence(length=20,processes=list(list(p)))
  # set omega values through omegaVarM0.CodonSequence
  omegaVarM0(s,p)$theta[1]$omega$theta[2]$
  # get a histogram of omega values from the range 1:15
  omegaHist(s,p,breaks=10,1:15)
```

Description

The M0 (one-ratio) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M0 (one-ratio) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega 1</td>
<td></td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM0(this, process, omega, index, ...)
```
Arguments

- **this**: A `CodonSequence` object.
- **process**: A process object inheriting from GY94.
- **omega**: The fixed omega value.
- **index**: A vector of positions.
- **...**: Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see `CodonSequence`.

Examples

```r
# create a GY94 object
p <- GY94(kappa=2)
# create a CodonSequence object, attach process p
s <- CodonSequence(length=20, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:5 from model M0
omegaVarM0(s, p, omega=2, 1:5)
# get omega values
getOmegavs(s, p)
# get a histogram of omega values in range 1:5
omegaHist(s, p, breaks=50, 1:5)
```
The $M1$ (neutral) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the $M1$ (neutral) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega$_0$</td>
<td>$p$</td>
</tr>
<tr>
<td>omega$_1$</td>
<td>$1-p$</td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM1(this, process, p0, index, ...)
```

Arguments

- `this` A CodonSequence object.
- `process` A process object inheriting from GY94.
- `p0` See above.
- `index` A vector of positions.
- `...` Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see CodonSequence.

Examples

```r
# create a GY94 object
p <- GY94(kappa=2)
# create a CodonSequence object, attach process p
s <- CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M1
omegavarm1(s, p, p0=0.5, 1:20)
# get omega values
getOmegas(s, p)
# get a histogram of omega values in range 1:20
omegahist(s, p, breaks=50, 1:20)
```

omegaVarM2.CodonSequence

The M2 (selection) model of variable omega ratios among sites

Description

The M2 (selection) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M2 (selection) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_0</td>
<td>p0</td>
</tr>
<tr>
<td>omega_1</td>
<td>1-p0-p1</td>
</tr>
<tr>
<td>omega_2</td>
<td>p1</td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegavarm2(this, process, p0, p1, omega_2, index, ...)
```

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **p0**: See above.
### omegaVarM2.CodonSequence

<table>
<thead>
<tr>
<th>p1</th>
<th>See above.</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_2</td>
<td>See above.</td>
</tr>
<tr>
<td>index</td>
<td>A vector of positions.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

#### Value

Invisible TRUE.

#### Author(s)

Botond Sipos, Gregory Jordan

#### References


#### See Also

For more information see [CodonSequence](#).

#### Examples

```r
# create a GY94 object
p<-GY94(kappa=2)
# create a CodonSequence object, attach process p
s<-CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M2
omegaVarM2(s,p,p0=0.2,p1=0.3,omega_2=4,1:20)
# get omega values
getOmeas(s,p)
# get a histogram of omega values in range 1:20
omegaHist(s,p,breaks=50,1:20)
```
The M3 (discrete) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M3 (discrete) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_0</td>
<td>p0</td>
</tr>
<tr>
<td>omega_1</td>
<td>p1</td>
</tr>
<tr>
<td>omega_2</td>
<td>p2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>omega_k</td>
<td>pk</td>
</tr>
</tbody>
</table>

Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM3(this, process, omegas, probs, index, ...)
```

Arguments

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **omegas**: A vector of omega values (omega_0 ... omega_k).
- **probs**: A vector of probabilities (p0 ... pk).
- **index**: A vector of positions.
- **...**: Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan
omegaVarM4.CodonSequence

References


See Also

For more information see CodonSequence.

Examples

# create a GY94 object
p<-GY94(kappa=2)
# create a CodonSequence object, attach process p
s<-CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M3
omegaVarM3(s,p,omegas=c(0,2,4),probs=c(1/3,1/3,1/3),1:20)
# get omega values
getOmegas(s,p)
# get a histogram of omega values in range 1:20
omegahist(s,p,breaks=50,1:20)

omegaVarM4.CodonSequence

The M4 (freqs) model of variable omega ratios among sites

Description

The M4 (freqs) model of variable omega ratios among sites.

This method sets the omega site-process specific parameter in the specified range to values sampled from the M4 (freqs) model of variable omega ratios among sites.

Distribution of omega values:

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>omega_0  = 0</td>
<td>p0</td>
</tr>
<tr>
<td>omega_1  = 1/3</td>
<td>p1</td>
</tr>
<tr>
<td>omega_2  = 2/3</td>
<td>p2</td>
</tr>
<tr>
<td>omega_3  = 1</td>
<td>p3</td>
</tr>
<tr>
<td>omega_4  = 3</td>
<td>p4</td>
</tr>
</tbody>
</table>
Usage

```r
## S3 method for class 'CodonSequence'
omegaVarM4(this, process, probs, index, ...)
```

Arguments

- `this`: A CodonSequence object.
- `process`: A process object inheriting from GY94.
- `probs`: A vector of probabilities (p0 ... p4).
- `index`: A vector of positions.
- `...`: Not used.

Value

`Invisible TRUE.`

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see `CodonSequence`.

Examples

```r
# create a GY94 object
p <- GY94(kappa=2)
# create a CodonSequence object, attach process p
s <- CodonSequence(length=25, processes=list(list(p)))
# sample states
sampleStates(s)
# sample omegas in range 1:20 from model M4
omegaVarM4(s, p, probs=c(2/5, 1/5, 1/5, 1/10, 1/10), 1:20)
# get omega values
getOmegas(s, p)
# get a histogram of omega values in range 1:20
omegaHist(s, p, breaks=50, 1:20)
```
Description

Package:
Class PAM

Object
||
|---PSRoot
|---Process
|---GeneralSubstitution
|---AminoAcidSubst
|---PAM

Directly known subclasses:

public static class PAM
extends AminoAcidSubst

Usage

PAM(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
...
Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary
Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[]<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p<-PAM()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s<-AminoAcidSequence(length=10,processes=list(list(p)) )
# sample states
sampleStates(s)
    # make the first three positions invariable
setRateMultipliers(s,p,0,1:3)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoral(2))
# run simulation
Simulate(sim)
```
The PAM.dcmut empirical amino acid substitution model

Description

Package:
Class PAM.dcmut

Object

| ~~~~~| ~~~--PSSRoot
| ~~~~~| ~~~~~~~~~--Process
| ~~~~~~~~~--GeneralSubstitution
| ~~~~~~~~~--AminoAcidSubst
| ~~~~~~~~~--PAM.dcmut

Directly known subclasses:

class PAM.dcmut extends AminoAcidSubst

Usage

PAM.dcmut(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
builtFromPAML, checkConsistency, newAAMatrix, setEquDist, summary
Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[], as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
AminoAcidSubst GeneralSubstitution UNREST

Examples
# create substitution model object
p<-PAM.dcmut()
# get object summary
summary(p)
# display a bubble plot
plot(p)

    # The following code demonstrates how to use
    # the process in a simulation.

# create a sequence, attach process p
s<-Amino AcidSequence(length=10,processes=list(list(p)))
# sample states
sampleStates(s)
    # make the first three invariable
setRateMultipliers(s,p,0,1:3)
# get rate multipliers
Perform an event

**Description**

Performing an event means that the modifications described by the Event object are actually made by calling the event handler function as returned by `getHandler` with the Event object as the first argument.

The event won’t be performed if the handler function is invalid, if there is no associated Site object, if the site position is undefined, if the rate is undefined, or if the generator process is invalid.

The handler function will be overwritten after performing an event, so the Perform method should be called only once for every Event object.

**Usage**

```r
## S3 method for class 'Event'
Perform(this, ...)```

**Arguments**

- `this` An Event object.
- `...` Not used.

**Value**

The value returned by the handler function.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.
Examples

```r
# create a sequence and attach a process
s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
# get the first active event from the first site
# only Sequence methods set .position,
# so s$sites[[1]]$events[[1]] wouldn't work.
sequences <- getEvents(s, 1)[[1]]
# perform e
Perform(e)
# check the effect of the event on s
s
```

---

**PhyloSim**

**The PhyloSim class**

**Description**

PhyloSim is an extensible object-oriented framework for the Monte Carlo simulation of sequence evolution written in 100 percent R. It is built on the top of the `ape` and `rNoo` packages and uses Gillespie’s direct method to simulate substitutions, insertions and deletions.

Key features offered by the framework:

- Simulation of the evolution of a set of discrete characters with arbitrary states evolving by a continuous-time Markov process with an arbitrary rate matrix.
- Explicit implementations of the most popular substitution models (for nucleotides, amino acids and codons).
- Simulation under the popular models of among-sites rate variation, like the gamma (+G) and invariants plus gamma (+I+G) models.
- The possibility to simulate with arbitrarily complex patterns of among-sites rate variation by setting the site specific rates according to any R expression.
- Simulation with one or more separate insertion and/or deletion processes acting on the sequences, which sample the insertion/deletion length from an arbitrary discrete distribution or an R expression (so all the probability distributions implemented in R are readily available for this purpose).
- Simulation of the effects of variable functional constraints over the sites by site-process-specific insertion and deletion tolerance parameters, which determine the rejection probability of a proposed insertion/deletion.
- The possibility of having a different set of processes and site-process-specific parameters for every site, which allow for an arbitrary number of partitions in the simulated data.
- Simulation of heterotachy and other cases of non-homogeneous evolution by allowing the user to set "node hook" functions altering the site properties at internal nodes of the phylogeny.
- The possibility to export the counts of various events ("branch statistics") as phylo objects (see `exportStatTree.PhylSim`).
General notes:

- The Sequence objects have no "immortal links". The simulation is aborted if the sequence length shrinks to zero. It is up to the user to choose sensible indel rates and sequence lengths to prevent that.

- The sites near the beginning and end of the sequences have less sites proposing insertion and deletion events around the so the insertion and deletion processes have an "edge effect". The user can simulate realistic flanking sequences to alleviate the edge effect in the simulation settings where it may be an issue.

Notes on performance:

- The pure R implementation offers flexibility, but also comes with a slower simulation speed. If the PSIM_FAST object is present in the environment, a "fast & careless" mode is enabled. In this mode most of the error checking is skipped, increasing the speed. It is recommended that simulations are only run in fast mode if you are sure that the simulation settings are free from errors. It is probably a good practice to set up the simulations in normal mode with short sequences and enable fast mode when running the actual simulation with long sequences.

- Please note, that no "branch statistics" are saved in fast mode.

- Logging also has a negative impact on performance, so it’s not a good idea to run large simulations with the logging enabled.

- The time needed to run a simulation depends not only on the number of the sites, but also on the length of the tree.

- Constructing Sequence objects with large number of sites is expensive. Avoid doing that inside a cycle.

- In the case of Sequence objects with a large number of sites (more than 10 000) the amount of available memory can be limiting as well.

The examples below demonstrate only some more common simulation settings, the framework offers much more flexibility. See the package vignette (vignette("PhyloSim",package="phylosim")) and the examples directory (http://github.com/bsipos/phylosim/tree/master/examples/) for additional examples.

Package:

**Class PhyloSim**

```
Object
| ~| |
| -=<PSRoot
| ~~~~~| |
| ~~~~~~~~~=PhyloSim
```

Directly known subclasses:

- public static class **PhyloSim**
- extends PSRoot
Usage

PhyloSim(phylo=NA, root.seq=NA, name=NA, log.file=NA, log.level=-1, ...)

Arguments

phylo       A rooted phylo object, constructed by the APE package.
root.seq    A valid Sequence object with Process objects attached. Used as the starting
            sequence during simulation.
name        The name of the object (a character vector of length one).
log.file    Name of the file used for logging.
log.level   An integer specifying the verbosity of logging (see setLogLevel.PhyloSim).
...         Not used.

Fields and Methods

Methods:

  Debug       -
  Log         -
  Simulate    -
  Undocumented -
  as.character -
  attachHookToNode -
  attachSeqToNode -
  checkConsistency -
  detachHookFromNode -
  detachSeqFromNode -
  exportStatTree -
  getAlignment -
  getAlignmentLength -
  getBranchEvents -
  getEdge      -
  getEdges     -
  getId        -
  getLogFile   -
  getLogLevel  -
  getName      -
  getNedges    -
  getNodes     -
  getNtips     -
  getPhylo     -
  getRootNode  -
  getRootSeq   -
  getSeqFromNode -
  getSequences -
  getTipLabels -
Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$,.<-, [. [, [<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References
See Also

PSRoot Alphabet AminoAcidAlphabet Amino AcidSequence Amino AcidSubst Any Alphabet Binary Alphabet BinarySubst BrownianInsertor Codon Alphabet Codon Sequence Codon UNREST Continuous Deleter Continuous Insertor cpRE V Discrete Deleter Discrete Insertor Event F81 F84 Fast Field Deleter General Deleter General In Del General Insertor General Substitution GTR GY 94 HK Y JC 69 JTT JTT dcmut K 80 K 81 LG m t Art m t Mam m t REV 24 Mt Zo a Nucleotide Alphabet Nucleotide Sequence PAM PAM dcmut PhyloSim Process QMatrix Sequence Site T92 TN93 UNREST WAG

Examples

```r
set.seed(1)
## The following examples demonstrate
## the typical use of the framework.
## See the package vignette and
## "url(\text{http://github.com/bsi pos/phylosim/tree/master/examples/})

## The ll() method gives information about the methods defined
## in the immediate class of an object.
## Useful when exploring the framework.

s<-Sequence()
ll(s)
ll(PhyloSim())
ll(GTR())

## Example 1 - A short simulation:
## simulate nucleotide sequences and display
## the resulting alignment matrix.

Simulate(PhyloSim(phy=rcoul(3),
        root=NucleotideSequence(string="ATGC", proc=list(list(JC69()))) ) )
$alignment

## Construct a phylo object for the following
## simulations, scale total tree length to 1:

tmp<-PhyloSim(phy=rcoul(3))
scaleTree(tmp,1/tmp$treeLength)
tmp$treeLength

t<-tmp$phylo

## Example 3 - simulating rate variation,
## insertions and deletions.
## See the examples/example_3_clean.R file
## in the phylosim GitHub repository.

## Construct a GTR process object
   gtr<-GTR(
        name="MyGTR",
        rate.params=list(
            "a"=1, "b"=2, "c"=3,
            "d"=5, "e"=6
        ),
        delta="none",
        epsilon="none",
        kappa="none",
        omega="none"
    )
```
PhyloSim

"d"=1, "e"=2, "f"=3
base.freqs=c(2,2,1,1)/6

# get object summary
summary(gtr)
# display a bubble plot
plot(gtr)

# construct root sequence object
s<-NucleotideSequence(length=20)
# attach process via virtual field
s$processes<-list(list(gtr))
# sample states from the equilibrium
# distribution of the attached processes
sampleStates(s)
# create among-site rate variation by sampling
# the "rate.multiplier" site-process-specific parameter
# from a discrete gamma distribution (GTR+G).
plusGamma(s,gtr,shape=0.1)
# make the range 11:12 invariable
setRateMultipliers(s,gtr,0,11:12)
# get the rate multipliers for s and gtr
getRateMultipliers(s,gtr)

# proposing lengths in the range 1:3
d<-DiscreteDeletor(
  rate=0.1,
  name="MyDel",
  sizes=c(1:3),
  probs=c(3/6,2/6,1/6)
)
# get object
summary(d)
# plot deletion length distribution
plot(d)
# attach deletion process d to sequence s
attachProcess(s,d)
  # create a region rejecting all deletions
  setDeletionTolerance(s,d,0,11:12)

# construct an insertion process object
# proposing lengths in the range 1:3
i<-DiscreteInsertor(
  rate=0.1,
  name="MyDel",
  sizes=c(1:2),
  probs=c(1/2,1/2),
  template.seq=NucleotideSequence(length=1,processes=list(list(JC69())))
)
# states will be sampled from the JC69 equilibrium distribution
# get object
summary(i)
# plot insertion length distribution
plot(i)
# attach insertion process i to sequence s
attachProcess(s, i)
    # create a region rejecting all insertions
    setInsertionTolerance(s, i, 0, 11:12)

# plot total site rates
plot(s)
# construct simulation object
sim <- PhyloSim(root.seq = s, phylo = t)
# get object summary
summary(sim)
# plot tree
plot(sim)
# run simulation
Simulate(sim)
    # get the list of recorded per-branch event counts
    getBranchEvents(sim)
# export the number of substitutions as a phylo object
subst <- exportStatTree(sim, "substitution")
# plot the exported phylo object
plot(subst)
# plot tree and alignment
plot(sim)
# save and display alingment
file <- paste("PhyloSim_dummy.fasta", Sys.getpid(), ".fas", sep = "")
saveAlignment(sim, file = file);
# print out the Fasta file
cat(paste(scan(file = file, what = character(), sep = "\n"), collapse = "\n"));
cat("\n");
# delete Fasta file
unlink(file)

    # See \url{http://github.com/bsiophyssim/tree/master/examples/examples_class.R}
    # for the full list of PhyloSim constructor examples.

## See the package vignette and
## the GitHub repository for even more examples.

plot.ContinuousDeleter

*Plot the density of proposed lengths*

**Description**

Plot the density of proposed lengths.
Usage

```r
## S3 method for class 'ContinuousDeletor'
plot(x, sample.size = NA, ...)
```

Arguments

- `x` A `ContinuousDeletor` object.
- `sample.size` Number of lengths sampled for the plot.
- `...` Not used.

Value

The process object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousDeletor`.

Examples

```r
# create object
o <- ContinuousDeletor(rate = 1)
# set length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean = 10, sd = 4))
# set maxLength
setMaxLength(o, 30)
# plot density
plot(o)
```

---

**plot.ContinuousInsertor**

*Plot the density of proposed lengths*

Description

Plot the density of proposed lengths.

Usage

```r
## S3 method for class 'ContinuousInsertor'
plot(x, sample.size = NA, ...)
```
Arguments

x A ContinuousInsertor object.

sample.size Number of lengths sampled for the plot.

... Not used.

Value

The process object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ContinuousInsertor.

Examples

# create object
c <- ContinuousInsertor(rate=1)
# set length sampling expression via virtual field
c$dist <- expression(rnorm(1, mean=10, sd=4))
# set maxLength
setMaxLength(c, 30)
# plot density
plot(c)

plot.DiscreteDeleter  Plot the deletion length distribution

Description

Plot the deletion length distribution.

Usage

## S3 method for class 'DiscreteDeleter'
plot(x, ...)

Arguments

x A DiscreteDeleter object.

... Not used.
Value

The DiscreteDeletor object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see DiscreteDeletor.

Examples

d <- DiscreteDeletor(
  name = "MyDiscDel",
  sizes = 1:6,
  probs = c(0.25000000, 0.16666667, 0.08333333, 0.08333333, 0.16666667, 0.25000000)
)
# plot the deletion length distribution
plot(d)

Description

Plot the insertion length distribution.

Usage

## S3 method for class 'DiscreteInsertor'
plot(x, ...)

Arguments

  x  A DiscreteInsertor object.
  ... Not used.

Value

The DiscreteInsertor object (invisible).

Author(s)

Botond Sipos, Gregory Jordan
### Description

Create a bubble plot of the substitution process.

Bubble plots visualize the characteristics of the substitution process. The area of the circles is proportional to the rates/probabilities. The plot is not produced if the rate matrix or the equilibrium distribution has undefined elements.

### Usage

```r
## S3 method for class 'GeneralSubstitution'
plot(x, scale = 1, ...)
```

### Arguments

- `x` An object inheriting from GeneralSubstitution.
- `scale` A scale factor affecting the area of the circles.
- `...` Not used.

### Value

The process object (invisible).

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `GeneralSubstitution`. 

---

Examples

```r
i <- DiscreteInsertor(
  name = "MyDiscIns",
  sizes = 1:6,
  probs = c(0.2500000, 0.1666667, 0.0833333, 0.0833333, 0.1666667, 0.2500000)
)
# plot the insertion length distribution
plot(i)
```
plot.PhyloSim

Examples

```r
plot(BinarySubst(rate.list=list("0->1"=1,"1->0"=1.5)))
plot(JC69())
# get smaller circles
plot(JC69(),scale=0.5)
plot(F84(base.freqs=c(3/6,1/6,1/6,1/6)))
plot(WAG())
```

Description

Plot a PhyloSim object.

This method plots the aggregated alignment alongside the tree used for simulation. Various options allow for control over the plot style.

Usage

```r
## S3 method for class 'PhyloSim'
plot(x, plot.tree, plot.ancestors, plot.chars, plot.legend, plot.labels, aspect.ratio, num.pages, char.text.size, axis.text.size, color.scheme, color.branches, tree.xlim, aln.xlim, tracks, aln.length.tolerance, plot.nongap.bl, ...)
```

Arguments

- `x` A PhyloSim object.
- `plot.tree` Whether to plot the tree alongside the alignment. TRUE or FALSE; defaults to TRUE.
- `plot.ancestors` Whether to plot the ancestral sequences. TRUE or FALSE; defaults to TRUE.
- `plot.chars` Whether to plot the actual text of the characters.
- `plot.legend` Whether to plot the legend showing the character-to-color mapping.
- `plot.labels` Whether to plot the sequence labels along the y-axis.
- `aspect.ratio` (Experimental; when set, this option forces the num.pages value to 1) Constrains the alignment residues to have a certain aspect ratio; values above 1 cause vertically-stretched blocks. FALSE disables aspect ratio control, numerical values set the aspect ratio; defaults to FALSE.
- `num.pages` Optionally split the alignment over a number of vertically-stacked pages. This is useful for long alignments. 'auto' chooses a sensible number of pages, numerical values specify a number; defaults to 'auto'.
- `char.text.size` Text size for the aligned characters. This may require tweaking depending on the DPI and output format. Defaults to 'auto'.

Plot a PhyloSim object
axis.text.size  Text size for the sequence labels along the y-axis. This may require tweaking depending on the DPI and output format. Defaults to 'auto'.

color.scheme  Color scheme to use ('auto', 'binary', 'dna', 'protein', 'codon', 'combined', 'combined_codon'). Defaults to 'auto'. When set to 'auto', the function will choose an appropriate coloring scheme based on the alignment content.

color.branches  The event count used to color the branches ('substitutions' by default). See getBranchEvents.PhyloSim.

tree.xlim  The x-axis limits of the tree panel.
aln.xlim  The x-axis limits of the alignment panel (in alignment column coordinates).

tracks  Tracks to display above or below the alignment as colored blocks.

The input format for tracks is a list of data frames with the following possible fields, all of which are optional and can be omitted:

- pos - the sequence position (starting with 1) of the feature. Defaults to NULL.
- score - the score (between 0 and 1) of the feature. Scores above 1 or below zero will be truncated. Defaults to 1.
- y_lo - the lower Y offset (between 0 and 1) of the feature. Defaults to 0. Use a y_lo and y_hi value for each row in the track data frame to create a wiggle plot like effect.
- y_hi - the upper Y offset (between 0 and 1) of the feature. Defaults to 1. Use just a y_hi value for each row in the track data frame to create a bar plot like effect.

- id - the display ID for the track. Defaults to 'Track'.
- layout - set to 'above' to put the track above the alignment, 'below' for below.
- height - the number of alignment rows for the track to span in height. Defaults to 3.
- color.gradient - a comma-separated list of colors to interpolate between when coloring the blocks. Examples: 'white,red' 'blue,gray,red' '#FF00FF,FFFFFF'. Defaults to 'white,black'.
- color - a single color to use when coloring the blocks. Mutually exclusive with color.gradient, and if a color.gradient value exists then this value will be ignored. Defaults to black.
- background - a color for the background of the track. Defaults to white.

aln.length.tolerance  The desired alignment/sequence length ratio (A/S ratio) to trim the alignment to. The A/S ratio is defined as the ratio between the alignment length and the mean ungapped sequence length, and the alignment trimming procedure will remove blocks of indel-containing columns (in a sensible order) until either (a) the desired indel tolerance is reached, or (b) no more columns can be removed without yielding an empty alignment. A track is added below the alignment to indicate how many indels each resulting alignment column used used to harbor, and black squares are overlaid onto the alignment where extant sequence data has been trimmed. Defaults to NULL (no trimming); values in the range of 0.9 to 1.3 tend to work well at improving the legibility of very gappy alignments.
plot.nongap.bl  If set to TRUE, plots the non-gap branch length (defined as the branch length of
the subtree of non-gapped sequences) as a track below the alignment. Defaults
to FALSE.

...  Not used.

Value
The PhyloSim object (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PhyloSim.

Examples
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
  name="TinySim",
  phylo=rcoal(3),
  root.seq=NucleotideSequence(string="ATGCTAGCTAGG",processes=list(list(JC69())))
);
  # plot the aggregated phylo object
  plot(sim)
# run simulation
Simulate(sim)
  # Plot the alignment without the tree or ancestral sequences.
  plot(sim, plot.ancestors=FALSE, plot.tree=FALSE)
# Force a DNA-based color scheme
# (default is 'auto' to auto-detect based on the sequence composition)
  plot(sim, color.scheme='dna', plot.legend=TRUE)

plot.Sequence  
Plot the total site rates for a collection of Site objects aggregated by a
Sequence object

Description
Plot the total site rates for a collection of Site objects aggregated by a Sequence object.

Usage
## S3 method for class 'Sequence'
plot(x, index=NA, ...)
Arguments

`x` A Sequence object.

`index` An integer vector specifying a set of positions. It is set to `1:seq$length` if omitted.

`...` Not used.

Value

Invisible TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create a nucleotide sequence with a process attached
s <- NucleotideSequence(string = "ATGGCCA", processes = list(list(JC69())))
# plot total rates in range 1:4
plot(s, 1:4)
# plot all total rates
plot(s)
```

---

`plot.ToleranceSubstitution`

Create a bubble plot of the substitution process

Description

Create a bubble plot of the substitution process.

Bubble plots visualize the characteristics of the substitution process. The area of the circles is proportional to the rates/probabilities. The plot is not produced if the rate matrix or the equilibrium distribution has undefined elements.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
plot(x, scale = 1, ...)
```
Arguments

x  An object inheriting from ToleranceSubstitution.
scale  A scale factor affecting the area of the circles.
...  Not used.

Value

The process object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ToleranceSubstitution.

Examples

plot(BinarySubst(rate.list=list("0->1"=1,"1->0"=1.5)))
plot(JC69())
  # get smaller circles
plot(JC69(),scale=0.5)
plot(F84(base.freqs=c(3/6,1/6,1/6,1/6)))
plot(WAG())
plusGamma.Sequence

Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **id**: The identifier of the site-process specific parameter.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

Invisible TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process Sequence

Examples

```r
# create a nucleotide sequence with a process attached
p<-JC69()
s<-NucleotideSequence(string="ATGGCCA",processes=list(list(p)))
# plot rate multipliers in range 1:4
plotParametersAtSites(s,p,"rate.multiplier",1:4)
# plot rate multiplier for the full sequence
plotParametersAtSites(s,p,"rate.multiplier")
```

Description

Sample the rate multiplier parameters of a Process from a Gamma distribution for a collection of Site objects aggregated by a Sequence object.

The gamma distribution is discretized by calculating the means of the portions corresponding to the categories having equal probabilities. If the `ncat` argument is not numeric, the rates are sampled from the continuous gamma distribution.
Usage

### S3 method for class 'Sequence'

plusGamma(this, process, shape, index, ncat=4, ...)

Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **shape**: The shape parameter of the gamma distribution.
- **index**: A vector of positions.
- **ncat**: Number of categories in the discretized gamma distribution (4 by default).
- **...**: Not used.

Value

The Sequence object (invisible);

Author(s)

Botond Sipos, Gregory Jordan

References


See Also

For more information see `Sequence`.

Examples

```r
# create a sequence
s <- NucleotideSequence(length=20)
# attach a process
p <- JC69()
attachProcess(s, p)
# get rate multipliers
getRateMultipliers(s, p) # the default value is 1.0
# sample rate multipliers in range 1:5 from a discrete gamma distribution with shape parameter 0.5
plusGamma(s, p, 0.5, 1:5)
# get rate multipliers
getRateMultipliers(s, p) # the default value is 1.0
# sample rates from a continuous gamma distribution with shape parameter 0.5
plusGamma(s, p, 0.5, ncat="cont")
# get rate multipliers
```
Sample the rate multiplier parameters of a Process from an I+Gamma distribution for a collection of Site objects aggregated by a Sequence object.

Sample the rate multiplier parameters of a Process from an I+Gamma distribution for a collection of Site objects aggregated by a Sequence object. The gamma distribution is discretized by calculating the means of the portions corresponding to the categories having equal probabilities. If the ncat argument is not numeric, the rates are sampled from the continuous gamma distribution.

## Usage

```r
## S3 method for class 'Sequence'
plusInvGamma(this, process, pinv, shape, index, ncat=4, ...)
```

## Arguments

- `this`: A Sequence object.
- `process`: A Process object.
- `pinv`: The proportion of invariant sites.
- `shape`: The shape parameter of the gamma distribution.
- `index`: A vector of positions.
- `ncat`: Number of categories in the discretized gamma distribution (4 by default).
- `...`: Not used.

## Value

The Sequence object (invisible);

## Author(s)

Botond Sipos, Gregory Jordan

## References


See Also

For more information see Sequence.

Examples

# create a sequence
s <- NucleotideSequence(length=20)
# attach a process
p <- JC69()
attachProcess(s,p)
# get rate multipliers
gRateMultipliers(s,p) # the default value is 1.0
# sample rate multipliers in range 1:5 from I+discrete Gamma
plusInvGamma(s,p, pinv=0.5, shape=0.5,1:5)
# get rate multipliers
gRateMultipliers(s,p) # the default value is 1.0
# sample rates from an I+continuos Gamma model
plusInvGamma(s,p, pinv=0.5, shape=0.5,1:5, ncat="cont")
# get rate multipliers
gRateMultipliers(s,p)
print.QMatrix

Print the character representation of a QMatrix object

Description

Print the character representation of a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
print(x, ...)
```

Arguments

- `x` A QMatrix object.
- `...` Not used.

Value

The character representation of the QMatrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create a QMatrix object
m <- QMatrix()
# print the character representation
print(m)
# the same, but implicitly
m
```
**Process**

---

**The Process class**

**Description**

This is the class representing a generic process acting on Site and Sequence objects. Process objects can be attached to Site objects if the associated Alphabet objects match.

The processes can have site-process-specific parameters. The templates for site-process-specific parameters and their default values are stored in the Process objects and copied into the Site object when the process is attached. See the documentation of the Site class for more details.

The rate multiplier parameter (id="rate.multiplier") is present in the Process class and is inherited by all descendant classes.

**Package:**

**Class Process**

~~|  
~~|--PSRoot  
~~~~|  
~~~~|--Process  

**Directly known subclasses:**

AminoAcidSubst, BinarySubst, BrownianInsertor, CodonSubst, CodonUNREST, ContinuousDele- 
tor, ContinuousInsertor, DiscreteDeleter, DiscreteInsertor, ECMrest, ECMUnrest, F81, F84, Fast-
FieldDeleter, GTR, GY94, GeneralDeleter, GeneralInDel, GeneralInsertor, GeneralSubstitution,  
HKY, JC69, JTT, JTT.dcmut, K80, K81, LG, MtZoa, PAM, PAM.dcmut, T92, TN93, Tolerance-
Substitution, UNREST, WAG, cpREV, mtArt, mtMam, mtREV24

**public static class Process**

extends PSRoot

**Usage**

Process(name=NA, alphabet=NA, ...)

**Arguments**

- **name**
  The name of the Process object: a character vector of length one.

- **alphabet**
  The associated Alphabet object.

- **...**
  Not used.
Fields and Methods

Methods:
Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Process.

Examples

# create a Process object
p<-Process(name="MyProc",alphabet=AminoAcidAlphabet())
# check if it's a Process object
is.Process(p)
# check object consistency
checkConsistency(p)
# set process name
p$name="Ppppprocce"
# get process name
p$name
# get unique process identifier
p$id
# get the list of site specific parameters and parameter IDs
p$siteSpecificParamList
proposeLength. GeneralInDel

Propose indel length

Description

Propose indel length.
This method simply calls the function returned by the getProposeBy method.

Usage

```r
# S3 method for class 'GeneralInDel'
proposeLength(this, ...)```

Arguments

- `this` A GeneralInDel object.
- `...` Not used.

Value

A numeric vector of length one (the indel length).
**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
For more information see `GeneralInDel`.

**Examples**

```r
# create a GeneralInDel object
# proposing event lengths in the range 1:10
o <- GeneralInDel(rate = 1, propose.by = function(process)(sample(c(1:10),1)));
# propose indel length
proposeLength(o)
```

---

**PSRoot**

*The root class for all phylosim objects*

---

**Description**

The root class for all phylosim objects containing some utility methods. Package:
Class `PSRoot`

```
Object
```
```
  ~ ~ ~ ~ PSRoot
```

**Directly known subclasses:**

- `Alphabet`, `AminoAcidAlphabet`, `AminoAcidSequence`, `AminoAcidSubst`, `AnyAlphabet`, `BinaryAlphabet`, `BinarySequence`, `BinarySubst`, `BrownianInsertor`, `CodonAlphabet`, `CodonSequence`, `CodonSubst`, `CodonUNREST`, `ContinuousDeletor`, `ContinuousInsertor`, `DiscreteDeletor`, `DiscreteInsertor`, `ECMrest`, `ECMunrest`, `Event`, `F81`, `F84`, `FastFieldDeletor`, `GTR`, `GY94`, `GeneralDeletor`, `GeneralInDel`, `GeneralInsertor`, `GeneralSubstitution`, `HKY`, `JC69`, `JTT`, `JTT.dcmut`, `K80`, `K81`, `LG`, `MtZoa`, `NucleotideAlphabet`, `NucleotideSequence`, `PAM`, `PAM.dcmut`, `PhyloSim`, `Process`, `QMatrix`, `Sequence`, `Site`, `T92`, `TN93`, `ToleranceSubstitution`, `UNREST`, `WAG`, `cpREV`, `mtArt`, `mtMam`, `mtREV24`

public static class `PSRoot`
extends `Object`

**Usage**

`PSRoot(...)`
Arguments

... Not used.

Fields and Methods

Methods:

checkConsistency -
enableVirtual -
geComments -
geMethodsList -
globalConsistencyCheck -
intersect.list -
is -
is.na -
ll -
my.all.equal -
plot -
setComments -
setMethodsList -
summary -
virtualAssignmentForbidden -

Methods inherited from Object:
$; $<-, []; [[<-; as.character; attach; attachLocally; clearCache; clearLookupCache; clone; detach; equals; extend; finalize; getEnvironment; getFieldModifier; getFieldModifiers; getFields; getInstantiationTime; getStaticInstance; hasField; hashCode; ll; load; names; objectSize; print; save

Author(s)

Botond Sipos, Gregory Jordan

See Also

Object

Examples

obj<-'PSRoot()';
ll(obj);
The PSRootSummary class

Description

PSRootSummary objects are blessed lists containing summary entries created by summary.* methods.

Package:

Class PSRootSummary

++-PSRootSummary

Directly known subclasses:

public class PSRootSummary

Usage

PSRootSummary(summary=list(), ...)

Arguments

summary A list.
... Not used.

Fields and Methods

Methods:

    checkConsistency -
    print -

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PSRootSummary.
QMatrix

The QMatrix class

Description

This is the class representing rate matrices. The QMatrix objects store two rate matrices: one contains the rates provided by the user (unscaled rate matrix), the other matrix (scaled rate matrix) is rescaled so the expected number of substitutions per unit time is equal to one when the process is at equilibrium. The scaled rate matrices, along with the site-process-specific rate multiplier parameters define the rates of the Event objects generated by the associated GeneralSubstitution objects.

The QMatrix objects have a bidirectional relationship with the GeneralSubstitution object as they store a reference to the associated process objects. QMatrix objects do not have a write-protection field, but they use the one from the associated GeneralSubstitution object.

Usually there is no need to interact with the QMatrix objects directly, so this class is mainly used internally.

Package:

Class QMatrix

Object

```java
~~| ~~~++--PSRoot
```

```
```

Directly known subclasses:

public static class QMatrix
extends PSRoot

Usage

QMatrix(name="Anonymous", alphabet=NA, rate.list=NA, process=NA, ...)

Arguments

name The name of the QMatrix object (a character vector of length one).
alphabet An Alphabet object.
rate.list A list of unscaled event rates and the associated event names. It will be passed to the setRateList.QMatrix method.
process A GeneralSubstitution object to be associated with the QMatrix object.
... Not used.
Fields and Methods

Methods:

Scale
as.character
checkConsistency
getAlphabet
getEventRate
getId
getMatrix
getName
getProcess
getRate
getRateList
getScaledMatrix
getWriteProtected
is
print
setAlphabet
setId
setMatrix
setName
setProcess
setRate
setRateList
setScaledMatrix
setWriteProtected
summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [], [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
GeneralSubstitution Alphabet Process
Examples

# create a QMatrix object by providing an Alphabet object and rates
m<-QMatrix(name="Susie Q", alphabet=BinaryAlphabet(), rate.list=list("1->0"=2,"0->1"=3))
# get object summary
summary(m)
# create a GeneralSubstitution object by
# providing an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m<-p$QMatrix
# get various object properties
m
is.QMatrix(m)
m$name
m$id
m$alphabet
# get the associated process
m$process
# get the unscaled rate of "0->1"
getRate(m,"0->1")
# get the scaled rate of "0->1"
getEventRate(m,"0->1")
# get the list of unscaled rates
m$rateList
# get unscaled rate matrix
m$matrix
# get scaled rate matrix
m$scaledMatrix

readAlignment.PhyloSim

Read alignment from file

Description

Read alignment from file.

This method reads an alignment by using the read.dna function from the ape package and stores in the PhyloSim object. If a tree is already attached to the PhyloSim object, the alignment must at least contain the sequences corresponding to tip nodes (but it may also contain additional ancestral sequences).

Usage

## S3 method for class 'PhyloSim'
readAlignment(this, file, format="fasta", ...)
Arguments

this A PhyloSim object.
file A file name specified by either a variable of mode character, or a double-quoted string.
format a character string specifying the format of the DNA sequences. Four choices are possible: "interleaved", "sequential", "clustal", or "fasta", or any unambiguous abbreviation of these.
... Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see \texttt{phylosim}.

Examples

\begin{verbatim}
# get a safe file name
fname<-paste("PhyloSim_dummy_fas_",Sys.getpid(),sep="")
# write out a fasta alignment
cat("T\nATCTTT-CG\n",file=fname);
cat("A\nG--TC-TCGG\n",file=fname,append=TRUE);
cat("G\nG--TC-TCGG\n",file=fname,append=TRUE);
cat("G\nTC-TCGG",file=fname,append=TRUE);
# construct a PhyloSim object,
# set the phylo object
sim<-PhyloSim(phylr=coal(4))
# read the alignment
readAlignment(sim,fname)
# remove alignment file
unlink(fname)
# plot the tree & alignment
plot(sim)
\end{verbatim}
Description

Read tree from file.

This method reads a tree by using the \texttt{read.tree} function from the \texttt{ape} package and stores in the \texttt{PhyloSim} object. If an alignment is already attached to the \texttt{PhyloSim} object, it must contain all sequences corresponding to tip nodes.

Usage

\begin{verbatim}
## S3 method for class 'PhyloSim'
readTree(this, file, ...)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{this} A \texttt{PhyloSim} object.
\item \texttt{file} A file name specified by either a variable of mode character, or a double-quoted string.
\item \texttt{...} Not used.
\end{itemize}

Value

The \texttt{PhyloSim} object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see \texttt{PhyloSim}.

Examples

\begin{verbatim}
# get a safe file name
fname<-paste("PhyloSim_dummy_fas_", Sys.getpid(), sep="")
# write out a fasta alignment
cat("(a,(b,c));", file=fname);
# construct a PhyloSim object:
sim<-PhyloSim()
# read the alignment
readTree(sim, fname)
# remove alignment file
unlink(fname)
# plot the tree
plot(sim)
\end{verbatim}
Rescale the scaled rate matrix of a QMatrix object aggregated by a GeneralSubstitution object.

The QMatrix objects aggregated by the GeneralSubstitution objects store two rate matrices: one contains the rates provided by the user (unscaled rate matrix), the other matrix (scaled rate matrix) is rescaled to have the expected number of substitutions per unit time equal to one when the process is at equilibrium. This method performs the re-scaling of the scaled rate matrix in the associated QMatrix object based on the equilibrium distribution and the unscaled rate matrix.

This method is mainly used internally as the scaled matrix is rescaled every time the unscaled matrix is modified.

## Usage

```r
## S3 method for class 'GeneralSubstitution'
rescaleQMatrix(this, ...)  
```

## Arguments

**this**  
A GeneralSubstitution object.

**...**  
Not used.

## Value

Invisible TRUE.

## Author(s)

Botond Sipos, Gregory Jordan

## See Also

For more information see `GeneralSubstitution`.

## Examples

```r
# create a GeneralSubstitution object  
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate=list("1->0"=1,"0->1"=3))
# rescale rate matrix
rescaleQMatrix(p)
```
rescaleQMatrix.ToleranceSubstitution

Rescale the scaled rate matrix of a QMatrix object aggregated by a ToleranceSubstitution object

Description
Rescale the scaled rate matrix of a QMatrix object aggregated by a ToleranceSubstitution object. The QMatrix objects aggregated by the ToleranceSubstitution objects store two rate matrices: one contains the rates provided by the user (unscaled rate matrix), the other matrix (scaled rate matrix) is rescaled to have the expected number of substitutions per unit time equal to one when the process is at equilibrium. This method performs the re-scaling of the scaled rate matrix in the associated QMatrix object based on the equilibrium distribution and the unscaled rate matrix. This method is mainly used internally as the scaled matrix is rescaled every time the unscaled matrix is modified.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
rescaleQMatrix(this, ...)
```

Arguments

- `this` A ToleranceSubstitution object.
- `...` Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ToleranceSubstitution`.

Examples

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p <- ToleranceSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# rescale rate matrix
rescaleQMatrix(p)
```
Reverse complement a NucleotideSequence object

Description

Reverse complement a NucleotideSequence object.

The method reverse complements the sequence "in place", no object cloning is performed.

Usage

## S3 method for class 'NucleotideSequence'
revComp(this, ...)

Arguments

- this: A NucleotideSequence object
- ...: Not used.

Value

Invisible TRUE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see NucleotideSequence.

Examples

s<-NucleotideSequence(string="ATGC")
s revComp(s) s
SampleState.GeneralSubstitution

Sample a state from the equilibrium distribution of a GeneralSubstitution object.

Description

Sample a state from the equilibrium distribution of a GeneralSubstitution object.

Usage

```r
## S3 method for class 'GeneralSubstitution'
sampleState(this, ...)
```

Arguments

- `this`: A GeneralSubstitution object.
- `...`: Not used.

Value

A character vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet = BinaryAlphabet(), rate.list = list("1->0"=1,"0->1"=1))
# get equilibrium distribution
getEquDist(p)
# get equilibrium distribution via virtual field
p$equDist
# sample from equilibrium distribution
sampleState(p)
```
Sample a state from the equilibrium distribution of a ToleranceSubstitution object.

**Description**

Sample a state from the equilibrium distribution of a ToleranceSubstitution object.

**Usage**

```r
## S3 method for class 'ToleranceSubstitution'
sampleState(this, ...)  
```

**Arguments**

- `this`  
  A ToleranceSubstitution object.

- `...`  
  Not used.

**Value**

A character vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ToleranceSubstitution`.

**Examples**

```r
# create a ToleranceSubstitution object  
# provide an Alphabet object and the rates  
p <- ToleranceSubstitution(alphabet = BinaryAlphabet(), rate = list("1->0" = 1, "0->1" = 1))  
# get equilibrium distribution  
ged <- getEquiDist(p)  
# get equilibrium distribution via virtual field  
p$equiDist  
# sample from equilibrium distribution  
sampleState(p)
```
**sampleStates.Sequence**  
Sample the states for a collection of Site objects aggregated by a Sequence object

**Description**

Sample the states for a collection of Site objects aggregated by a Sequence object. This method samples new states from the equilibrium distribution of the attached process(es) for sites having undefined states (NA). If a site has more than one substitution process attached, then the method samples the new state from the mixture of equilibrium distributions. The weight of each equilibrium distribution is proportional to the site-process specific rate multiplier of the corresponding process at the given site.

Sites having defined states are not touched. All sites with undefined states must have at least one substitution process (object inheriting from GeneralSubstitution) attached.

**Usage**

```r
## S3 method for class 'Sequence'
sampleStates(this, index, ...)
```

**Arguments**

- `this`  
  A Sequence object.

- `index`  
  An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.

- `...`  
  Not used.

**Value**

The Sequence object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

sampleState.GeneralSubstitution GeneralSubstitution

**Examples**

```r
# create a nucleotide sequence
s<-NucleotideSequence(length=80)
# create some processes
jc69<-JC69(); # Jukes-Cantor
hky<-HKY(base.freqs=c(0.1,0.4,0.1,0.4)) # "GC-rich" HKY
# attach the processes
```
saveAlignment.PhyloSim

s$processes<-list(list(jc69)) # jc is attached to all sites
attachProcess(s,hky,60:80) # hky is attached to range 60:80
# tweak rate multiplier for hky
setRateMultipliers(s,hky,10,60:80)
# set states in range 1:20
setStates(s,"A",1:20)
# sample remaining states
sampleStates(s)
# print sequence
s

description

Save the alignment stored in a PhyloSim object in a Fasta file.

Usage

## S3 method for class 'PhyloSim'
saveAlignment(this, file="phylosim.fas", skip.internal=FALSE, paranoid=FALSE, ...)

Arguments

this A PhyloSim object.
file The name of the output file.
skip.internal Do not save sequences corresponding to internal nodes.
paranoid Check the consistency of the alignment.
... Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
Scale.QMatrix

Scale the scaled rate matrix stored in a QMatrix object by the provided factor.

Description

Scale the scaled rate matrix stored in a QMatrix object by the provided factor. This method sets the scaled rate matrix to \( \text{unscaled_matrix} \times \text{constant} \).

Usage

```r
## S3 method for class 'QMatrix'
Scale(this, constant=NA, ...)
```

Arguments

- **this**: A QMatrix object.
- **constant**: The scaling factor (a numeric vector of length one).
- **...**: Not used.

Value

The QMatrix object (invisible).

Author(s)

Botond Sipos, Gregory Jordan
scaleTree.PhyloSim

See Also
For more information see QMatrix.

Examples

```r
# create a QMatrix object
# , provide Alphabet object and rates
m <- QMatrix(name = "Susie Q", alphabet = BinaryAlphabet(), rate.list = list("1->0" = 2, "0->1" = 3))
# get object summary
summary(m)
# perform scaling
Scale(m, 1/0.666)
# get object summary
summary(m)
```

scaleTree.PhyloSim  Scale the branch lengths of a phylo object aggregated by a PhyloSim object

Description
Scale the branch lengths of a phylo object aggregated by a PhyloSim object. This method multiples all the edge lengths by the specified factor.

Usage

```r
## S3 method for class 'PhyloSim'
scaleTree(this, factor, ...)
```

Arguments

- `this`  A PhyloSim object.
- `factor`  A numeric vector of length one.
- `...`  Not used.

Value
The PhyloSim object (invisible).

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PhyloSim.
Examples

```r
# create a PhyloSim object
sim <- PhyloSim(phylo = coal(5));
# get the tree length
sim$treeLength
# scale tree
scaleTree(sim, 10)
# get the scaled tree length
sim$treeLength
```

Sequence

**The Sequence class**

Description

This is the class representing a sequence. The backbone of the Sequence objects are lists aggregating Site objects. The class has fields for keeping track of cumulative site rates, the sum of all active event rates and methods for performing actions on a collection of sites (positions).

The Sequence objects have a field specifying an ancestral object, which can be a Sequence object (when the object is obtained through clone() ) or the "Root insertion process" object (for newly created objects).

Package:

**Class Sequence**

```r
public static class Sequence
extends PSRoot
```

Directly known subclasses:

*AminoAcidSequence, BinarySequence, CodonSequence, NucleotideSequence*

Usage

```r
Sequence(name=NA, string=NA, length=NA, alphabets=NA, processes=NA, ancestral.obj=NA, ...)
```
Arguments

name
The name of the Sequence object.

string
A string containing symbols belonging to the associated Alphabet object. It can be used to set the initial states of the aggregated Site objects. It also specifies the length of the sequence.

length
The length of the sequence. It cannot be used when 'string' is specified.

alphabets
A list of Alphabet objects to be associated with the Site objects. The list is recycled in the case it is shorter than the sequence length.

processes
A list of lists of Process objects to be attached (recycled if shorter than sequence length).

ancestral.obj
The ancestral object (Sequence or Process).

...Not used.

Fields and Methods

Methods:

- as.character
- attachProcess
- checkConsistency
- clearStates
- clone
- copySubSequence
- deleteSubSequence
- detachProcess
- getAlphabets
- getAncestral
- getBigRate
- getCumulativeRates
- getCumulativeRatesFromRange
- getDeletionTolerance
- getEvents
- getId
- getInsertionTolerance
- getLength
- getName
- getOmegs
- getParameterAtSites
- getProcesses
- getRateMultipliers
- getSites
- getStates
- getString
- getSymbolFreqs
- getTotalRates
- getTotalRatesFromRange
Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$,$<-, [ ], [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstanceationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet Site Process Event
Examples

# create a sequence object by providing alphabets, processes and states
s.one<-Sequence(
  name="Seq",
  string="AATTGGCCTTAAGGCCTTAA",
  alphabets=list(NucleotideAlphabet()),
  processes=list(list(JC69()))
)
s.one
# check if inherits from Sequence
is.Sequence(s.one)
# get object summary
summary(s.one)
# create a sequence object,
# specifying length, alphabets
# and ancestral object
s<-Sequence(
  name="Seq",
  length=20,
  ancestral.obj=s.one
)
# get sequence string
s$string
# get the list of site objects
s$sites
# get object id
s$id
# set and get name
s$name<-"SeqSeq"

s$seq
# get length
s$length
# get and set ancestral object
s$ancestral
s$ancestral<-Sequence();
# set alphabets
setAlphabets(s,list(NucleotideAlphabet()))
# set states
# "A"-s in the range 1:10
setStates(s,"A",1:10)
# a pattern of "ATGC"-s in the range 11:20
setStates(s,c("A","T","G","C"),11:20)
# get states from range 10:12
getStates(s,10:12)
# attach a JC69 process to range 1:10
jc69<-JC69()
attachProcess(s,jc69,1:10)
# set the rate multiplier site-process specific parameter for jc69
setParameterAtSites(s,jc69,"rate.multiplier",2,1:10)
# get "rate.multiplier" for jc69 from range 1:2
getParameterAtSites(s, jc69, "rate.multiplier",1:2)
# attach a GTR process to range 11:20
gtr<-GTR()
attachProcess(s, gtr, 11:20)
# set and get rate multipliers for gtr
setRateMultipliers(s, gtr, c(0, 5, 1), 11:20)
getRateMultipliers(s, gtr, 11:20)
# get processes from range 1:5
getProcesses(s, 1:5)
# replace the processes with a (GTR, JC69), JC69 pattern
s$processes<-list(list(GTR()), JC69()), list(JC69()))
# get processes from range 1:2
getProcesses(s, 1:2)
# get unique processes
s$uniqueProcesses
# get unique alphabets
s$uniqueAlphabets
# get symbol frequencies
getSymbolFreqs(s)
# get the big rate
s$bigRate
# get a vector of total rates from range 1:4
getTotalRatesFromRange(s, 1:4)
# get a vector of cumulative rates from range 1:4
getCumulativeRatesFromRange(s, 1:4)
# reset all site states
clearStates(s)
# sample states from the equilibrium distributions
# of the attached substitution processes
sampleStates(s)
# clone s
s.clone<-clone(s)
# insert a sequence in s.clone after position 2
insertSequence(s.clone, NucleotideSequence(string="AAAAA"), 2)
s.clone
# delete positions 1, 2, 3 and 10
deleteSubSequence(s.clone, c(1:3, 10))
s.clone
# copy positions 7:10 into a new sequence object
sc<-copySubSequence(s.clone, 7:10)
sc
# get events from sc in the range 1:2
getEvents(sc, 1:2)

setAcceptBy.GeneralInDel

Set the function used for accepting/rejecting indel events
Description

Set the function used for accepting/rejecting indel events.

The function object must have the following arguments: process (the caller process), sequence (the target sequence), window (a vector containing the positions affecting acceptance).

Usage

```r
## S3 method for class 'GeneralInDel'
setAcceptBy(this, value, ...)
```

Arguments

- `this`: A GeneralInDel object.
- `value`: A function object.
- `...`: Not used.

Value

The function object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.

Examples

```r
# create a GeneralInDel object
# rejecting half of the events
o <- GeneralInDel(
  rate=1,
  propose.by=function(process){return(5)},
  accept.by=function(){sample(c(TRUE,FALSE),1)}
);

# set/get the acceptBy function
setAcceptBy(o, value=function(){return(FALSE)}) # reject all events
getAcceptBy(o)

# set/get acceptBy function via virtual field
o$acceptBy <- function(){return(TRUE)} # accept all events
o$acceptBy
```
Description

Set the size of the acceptance window
This parameter determines the number of sites neighbouring the position (from left and right) of
the insertion considered when accepting/rejecting a proposed insertion. The "insertion.tolerance"
parameter is retrieved from sites falling in the window specified by this parameter. The default value
is 1, so the two neighbouring sites are considered by default.

Usage

```r
## S3 method for class 'GeneralInsertor'
setAcceptWin(this, value, ...)
```

Arguments

- `this` A GeneralInsertor object.
- `value` An integer vector of length one.
- `...` Not used.

Value

The new value.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(rate=0.5);
# get acceptance window size
getAcceptWin(i)
# get acceptance window size via virtual field
i$acceptWin
# set acceptance window size
setAcceptWin(i, 2)
# set acceptance window size via virtual field
i$acceptWin <- 3
i$acceptWin
```
setAlignment.PhyloSim  Forbidden action: setting the alignment stored in a PhyloSim object

Description
Forbidden action: setting the alignment stored in a PhyloSim object.

Usage
```r
## S3 method for class 'PhyloSim'
setAlignment(this, value, ...)
```

Arguments
- `this`: An object.
- `value`: Not used.
- `...`: Not used.

Value
Throws an error.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `PhyloSim`.

setAlphabet.Generalsubstitution

Set the Alphabet object aggregated by a GeneralSubstitution object

Description
Set the Alphabet object aggregated by a GeneralSubstitution object.
This method also sets the alphabet for the associated QMatrix object, which will set all rates to NA. This method will also re-initialize the equilibrium distribution by setting all frequencies to NA.

Usage
```r
## S3 method for class 'Generalsubstitution'
setAlphabet(this, value, ...)
```
Arguments

- **this**: A GeneralSubstitution object.
- **value**: An Alphabet object.
- **...**: Not used.

Value

The Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralSubstitution`.

Examples

```r
# create a GeneralSubstitution object with an attached BinaryAlphabet object
p <- GeneralSubstitution(alphabet=BinaryAlphabet())
# get object summary
summary(p)
# get alphabet
getAlphabet(p)
# get alphabet via virtual field
p$alphabet
# set a new alphabet
setAlphabet(p, NucleotideAlphabet())
summary(p)
# set alphabet via virtual field
p$alphabet <- BinaryAlphabet()
p$alphabet
```

---

`setAlphabet.Process` *Associate an Alphabet object with a Process object*

Description

Associate an Alphabet object with a Process object.

Usage

```r
## S3 method for class 'Process'
setAlphabet(this, new.alphabet, ...)
```
Arguments

- `this` A Process object.
- `new.alphabet` A valid Alphabet object.
- `...` Not used.

Value

The Alphabet object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Process`.

Examples

```r
# create objects
a<-AminoAcidAlphabet()
p<-Process()
# associate p with Alphabet object a
setAlphabet(p,a)
p$alphabet
# associate p with a new NucleotideAlphabet via virtual field
p$alphabet<-NucleotideAlphabet()
p$alphabet
```

---

**setAlphabet.QMatrix**  
*Set the Alphabet object for a QMatrix object*

Description

Set the Alphabet object for a QMatrix object.

This method rebuilds the scaled and unscaled rate matrices and so sets all rates to NA.

Usage

```r
## S3 method for class 'QMatrix'
setAlphabet(this, value, ...)
```

Arguments

- `this` A QMatrix object.
- `value` An Alphabet object.
- `...` Not used.
setAlphabet.Site

Value

The Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create a QMatrix object
m<-QMatrix()
# set the alphabet
defaultAlphabet(m,
# get the alphabet
getAlphabet(m)
# set alphabet via virtual field
m$alphabet<-BinaryAlphabet()
summary(m)
```

---

setAlphabet.Site Attach an Alphabet object to a Site object

Description

Attach an Alphabet object to a Site object. If the ancestral site is not NA, then the symbol set of the ancestral Alphabet object and the new Alphabet object must be the same. The current state must be in the symbol set of the new Alphabet object, unless it is NA.

Usage

```r
## S3 method for class 'Site'
setAlphabet(this, new.alphabet, ...)
```

Arguments

- `this`: A Site object.
- `new.alphabet`: A valid Alphabet object.
- `...`: Not used.

Value

Returns the new Alphabet object (invisible).
setAlphabet.ToleranceSubstitution

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Site.

Examples

```r
# create a site object
s<-'Site()
# create an Alphabet object
a<-'Alphabet(c("A","T","G"))
# attach alphabet to site object
setAlphabet(s,a)
# set site state
s$state<-'A'
# clone the alphabet object
b<-'clone(a)
# modify symbol set in b
b$symbols<-'c(b$symbols,"C")
# attach b to s via virtual field
s$alphabet<b
s$alphabet
```

setAlphabet.ToleranceSubstitution

Set the Alphabet object aggregated by a ToleranceSubstitution object

Description
Set the Alphabet object aggregated by a ToleranceSubstitution object.
This method also sets the alphabet for the associated QMatrix object, which will set all rates to NA.
This method will also re-initialize the equilibrium distribution by setting all frequencies to NA.

Usage
```r
## S3 method for class 'ToleranceSubstitution'
setAlphabet(this, value, ...)
```

Arguments
- **this** A ToleranceSubstitution object.
- **value** An Alphabet object.
- **...** Not used.
Value

The Alphabet object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see ToleranceSubstitution.

Examples

# create a ToleranceSubstitution object with an attached BinaryAlphabet object
p <- ToleranceSubstitution(alphabet = BinaryAlphabet())
# get object summary
summary(p)
# get alphabet
getAlphabet(p)
# get alphabet via virtual field
p$alphabet
# set a new alphabet
setAlphabet(p, NucleotideAlphabet())
summary(p)
# set alphabet via virtual field
p$alphabet <- BinaryAlphabet()
p$alphabet

setAlphabet.Sequence

Associate Alphabet objects to a set of Site objects aggregated by a Sequence object.

Description

Associate Alphabet objects to a set of Site objects aggregated by a Sequence object.

Usage

## S3 method for class 'Sequence'
setAlphabet(this, value, index, ...)

Arguments

this  A Sequence object.
value A list of Alphabet objects, recycled if shorter than the index vector.
index An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
...   Not used.
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

```r
# create a Sequence object
s <- Sequence(length=10)
# set the alphabets for range 1:5
setAlphabets(s, list(NucleotideAlphabet(), BinaryAlphabet()), 1:5)
# set the alphabets for range 6:10
setAlphabets(s, list(AminoAcidAlphabet()), 6:10)
# get the list of attached Alphabet objects
getAlphabets(s)
# get Alphabets from a range
getAlphabets(s, c(2:3, 5))
# set alphabets via virtual field
s$alphabets <- list(BinaryAlphabet(), NucleotideAlphabet())
# get alphabets via virtual field
s$alphabets
```

Description

Set the ancestral object of a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setAncestral(this, value, ...)
```

Arguments

- `this` A Sequence object.
- `value` A Sequence or a Process object.
- `...` Not used.
**Value**

The new ancestral object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see Sequence.

**Examples**

```r
# create a nucleotide sequence and a process object
s <- NucleotideSequence(string = "AGCT")
p <- Process(name = "MyProcess")
# set the p as the ancestral of s
setAncestral(s, p)
s$ancestral
# clone s
cs <- clone(s)
# set cs as ancestral of s via virtual field
s$ancestral <- cs
# get ancestral ids
s$ancestral$id
cs$ancestral$id
```

---

**setAncestral.Site**  
*Forbidden action: setting the ancestral object for a Site object*

**Description**

Forbidden action: setting the ancestral object for a Site object.

**Usage**

```r
## S3 method for class 'Site'
setAncestral(this, value, ...)
```

**Arguments**

- **this**  
  A Site object.
- **value**  
  Not used.
- **...**  
  Not used.

**Value**

Throws an error.
**setBaseFreqs.F81**

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see Site.

---

setBaseFreqs.F81  *Set the base frequency parameters*

**Description**

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'F81'
setBaseFreqs(this, value, ...)
```

**Arguments**

- `this`  
  A F81 object.
- `value`  
  A vector of base frequencies.
- `...`  
  Not used.

**Value**

value (invisible)

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see F81.
Examples

```r
# construct a F81 object
p <- F81()
# set/get base frequency parameters
setBaseFreqs(p, c(2, 1, 2, 1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs <- c(4, 4, 1, 1)/10
p$baseFreqs
# get object summary
summary(p)
```

---

```r
setBaseFreqs.F84 Set the base frequency parameters
```

Description

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

Usage

```r
## S3 method for class 'F84'
setBaseFreqs(this, value, ...)
```

Arguments

- **this**  
  A F84 object.
- **value**  
  A vector of base frequencies.
- **...**  
  Not used.

Value

- **value** (invisible)

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see [F84](#).
## setBaseFreqs.GTR

### Set the base frequency parameters

**Description**

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'GTR'
setBaseFreqs(this, value, ...)
```

**Arguments**

- **this**: A GTR object.
- **value**: A vector of base frequencies.
- **...**: Not used.

**Value**

value (invisible)

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see GTR.
Examples

```r
# construct a GTR object
p <- GTR()
# set/get base frequency parameters
setBaseFreqs(p, c(2, 1, 2, 1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs <- c(4, 4, 1, 1)/10
p$baseFreqs
# get object summary
summary(p)
```

---

**setBaseFreqs.HKY**  
*Set the base frequency parameters*

**Description**

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'HKY'
setBaseFreqs(this, value, ...)
```

**Arguments**

- `this`: An HKY object.
- `value`: A vector of base frequencies.
- `...`: Not used.

**Value**

`value` (invisible)

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `hky`.  

Examples

```r
# construct object
p <- K8Y()
# set/get base frequency parameters
setBaseFreqs(p, c(2, 1, 2, 1)/6)
getBaseFreqs(p)
# set/get base frequency parameters via virtual field
p$baseFreqs <- c(4, 4, 1, 1)/10
p$baseFreqs
# get object summary
summary(p)
```

---

setBaseFreqs.K80  
*Forbidden action: setting the base frequency parameters for a K80 object*

---

Description

Forbidden action: setting the base frequency parameters for a K80 object.

Usage

```r
## S3 method for class 'K80'
setBaseFreqs(this, value, ...)
```

Arguments

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `K80`.
**setBaseFreqs.K81**

*Forbidden action: setting the base frequency parameters for a K81 model*

---

**setBaseFreqs.TN93**

*Set the base frequency parameters*

---

**Description**

Forbidden action: setting the base frequency parameters for a K81 model.

**Usage**

```r
## S3 method for class 'K81'
setBaseFreqs(this, value, ...)
```

**Arguments**

- `this` An object.
- `value` Not used.
- `...` Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see K81.

---

**Description**

Set the base frequency parameters.

The order of the frequency parameters must match with the order of symbols in the NucleotideAlphabet objects.

**Usage**

```r
## S3 method for class 'TN93'
setBaseFreqs(this, value, ...)
```
Arguments

    this  A TN93 object.
    value A vector of base frequencies.
    ... Not used.

Value

    value (invisible)

Author(s)

    Botond Sipos, Gregory Jordan

See Also

    For more information see TN93.

Examples

    # construct a TN93 object
    p<-TN93()
    # set/get base frequency parameters
    setBaseFreqs(p,c(2,1,2,1)/6)
    getBaseFreqs(p)
    # set/get base frequency parameters via virtual field
    p$baseFreqs<-c(4,4,1,1)/10
    p$baseFreqs
    # get object summary
    summary(p)

setBigRate.Sequence  Forbidden action: setting the sum of total active event rates for a Sequence object

Description

    Forbidden action: setting the sum of total active event rates for a Sequence object.

Usage

    ## S3 method for class 'Sequence'
    setBigRate(this, value, ...)

Arguments

    this  An object.
    value Not used.
    ... Not used.
Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

---

**setBranchEvents.PhyloSim**

Forbidden action: setting the list of events having per-branch statistics recorded

Description

Forbidden action: setting the list of events having per-branch statistics recorded.

Usage

```r
## S3 method for class 'PhyloSim'
setBranchEvents(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- ... Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
setCodonFreqs.GY94

**Get codon frequencies**

**Description**

Get codon frequencies.

**Usage**

```r
## S3 method for class 'GY94'
setCodonFreqs(this, value, ...)
```

**Arguments**

- `this` A GY94 object.
- `value` A vector of codon frequencies.
- `...` Not used.

**Value**

value (invisible)

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GY94`.

**Examples**

```r
# create a GY94 object
p<-GY94()
# get codon frequencies
getCodeonFreqs(p)
p$codonFreqs
# set codon frequencies
p$codonFreqs<-rep(c(1,2,3,4),length.out=61)
p$codonFreqs
```
setComments.PSRoot  Set the comments associated with an object

Description

Set the comments associated with an object.
The comment field can contain any type of object.

Usage

```r
## S3 method for class 'PSRoot'
setComments(this, new_value, ...)
```

Arguments

- **this**: A PSRoot object.
- **new_value**: An object.
- **...**: Not used.

Value

The new value of the comment field (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

Examples

```r
# create an object
o <- PSRoot()
# add some comments
setComments(o, "Random comment")
# get the comment
getComments(o)
# get/set the comment via virtual fields
o$comments <- "Second random comment"
o$comments
```
**setCumulativeRates.Sequence**

*Forbidden action: setting the cumulative rates for the sites aggregated by a Sequence object*

---

**Description**

Forbidden action: setting the cumulative rates for the sites aggregated by a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
setCumulativeRates(this, value, ...)
```

**Arguments**

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

---

**setDeletionTolerance.Sequence**

*Set the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object*

---

**Description**

Set the deletion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls `setParameterAtSites(this=this, process=process, id=id, value=value, index=index)`.

See `setParameterAtSites.Sequence` for more details.
setDist.ContinuousDeletor

Usage

```r
## S3 method for class 'Sequence'
setDeletionTolerance(this, process, value, index, ...)
```

Arguments

- `this`: A Sequence object.
- `process`: A Process object.
- `value`: A numeric vector, recycled if shorter than the index vector.
- `index`: An integer vector specifying a set of positions. It is set to `1:seq$length` if omitted.
- `...`: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

`setParameterAtSites.Sequence`

Description

Set the length sampling expression.

The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

Usage

```r
## S3 method for class 'ContinuousDeletor'
setDist(this, value, ...)
```

Arguments

- `this`: A ContinuousDeletor object.
- `value`: An R expression.
- `...`: Not used.
Value
An R expression object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see ContinuousDeleter.

Examples
# create object
o<-ContinuousDeleter(rate=1)
# set/get length sampling expression
setDist(o, expression(rnorm(1,mean=3, sd=2)))
getDist(o)
# set/get length sampling expression via virtual field
o$dist<-expression(rnorm(1,mean=6, sd=3))
o$dist
# set maxLength
o$maxLength<-10
# propose a length
proposeLength(o)

Description
Set the length sampling expression.
The length sampling expression can be any valid R expression returning a numeric vector of length one. The value returned by the expression will be rounded.

Usage
## S3 method for class 'ContinuousInsertor'
setDist(this, value, ...)

Arguments
   this A ContinuousInsertor object.
   value An R expression.
   ...  Not used.
Value

An R expression object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousInsertor`.

Examples

```r
# create object
o <- ContinuousInsertor(rate = 1)
# set/get length sampling expression
setDist(o, expression(rnorm(1, mean = 3, sd = 2)))
getDist(o)
# set/get length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean = 6, sd = 3))
o$dist
# set maxLength
o$maxLength <- 10
# propose a length
proposeLength(o)
```

---

**setEdges.PhyloSim**  
*Forbidden action: setting the edge matrix for a phylo object aggregated by a PhyloSim object*

Description

Forbidden action: setting the edge matrix for a phylo object aggregated by a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
setEdges(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.
setEquDist.GeneralSubstitution

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PhyloSim.

---

**setEquDist.GeneralSubstitution**

*Set the equilibrium distribution for a GeneralSubstitution object*

**Description**
Set the equilibrium distribution for a GeneralSubstitution object.

In the case the equilibrium distribution cannot be guessed from the rate matrix one should provide a valid equilibrium distribution. The equilibrium distribution must be compatible with the rate matrix. The provided numeric vector will be rescaled in the case the sum of the elements is not one.

**Usage**

```r
## S3 method for class 'GeneralSubstitution'
setEquDist(this, value, force=FALSE, silent=FALSE, ...)  
```

**Arguments**

- **this**
  A GeneralSubstitution object.

- **value**
  A numeric vector containing the equilibrium symbol frequencies. The order of the frequencies must be the same as in the symbol vector of the attached Alphabet object.

- **force**
  Do not check compatibility with the rate matrix.

- **silent**
  Do not print out warnings.

- **...**
  Not used.

**Value**

The new equilibrium distribution (invisible).

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
For more information see GeneralSubstitution.
Examples

```r
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(
  alphabet = BinaryAlphabet(),
  rate.list = list("1->0"=1,"0->1"=1)
)

# get equilibrium distribution
getEquDist(p)
# get equilibrium distribution via virtual field
p$equDist
# re-set the equilibrium distribution
dist <- p$equDist * 3
dist
setEquDist(p, dist)
p$equDist
# re-set equilibrium distribution via virtual field
p$equDist <- p$equDist * 2
p$equDist
```

---

setEquDist.ToleranceSubstitution

*Set the equilibrium distribution for a ToleranceSubstitution object*

Description

Set the equilibrium distribution for a ToleranceSubstitution object.

In the case the equilibrium distribution cannot be guessed from the rate matrix one should provide a valid equilibrium distribution. The equilibrium distribution must be compatible with the rate matrix. The provided numeric vector will be rescaled in the case the sum of the elements is not one.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
setEquDist(this, value, force=FALSE, silent=FALSE, ...)
```

Arguments

- **this**: A ToleranceSubstitution object.
- **value**: A numeric vector containing the equilibrium symbol frequencies. The order of the frequencies must be the same as in the symbol vector of the attached Alphabet object.
- **force**: Do not check compatibility with the rate matrix.
- **silent**: Do not print out warnings.
- **...**: Not used.
**Value**

The new equilibrium distribution (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ToleranceSubstitution`.

**Examples**

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p<-ToleranceSubstitution(
  alphabet=BinaryAlphabet(),
  rate.list=list("1->0"=1,"0->1"=1)
)

# get equilibrium distribution
getEquDist(p)

# get equilibrium distribution via virtual field
p$equDist

# re-set the equilibrium distribution
dist<-p$equDist * 3

dist

setEquDist(p,dist)
p$equDist

# re-set equilibrium distribution via virtual field
p$equDist<-p$equDist * 2

p$equDist
```

---

**setEvents.Site**

Forbidden action: setting the list of active events for a Site object

**Description**

Forbidden action: setting the list of active events for a Site object.

**Usage**

```r
## S3 method for class 'Site'
setEvents(this, value, ...)
```

**Arguments**

- `this`: An object.
- `value`: Not used.
- `...`: Not used.
setGenerateBy.GeneralInsertor

Description

Set the function object used for generating inserts. The provided function must return a Sequence object when called and must have the following arguments: process, length, target.seq, event.pos, insert.pos (see `generateInsert.GeneralInsertor`).

Usage

```r
## S3 method for class 'GeneralInsertor'
setGenerateBy(this, value, ...)
```

Arguments

- `this`: A GeneralInsertor object.
- `value`: A function object.
- `...`: Not used.

Value

The function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`. 
Examples

```r
# create a GeneralInsertor object
i<-GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq=NucleotideSequence(string="AAAAAAA")
)

# save insert generator
old.gen<-getGenerateBy(i)
# set a new insert generator
i$generateBy<-function(
  process,
  length,
  target.seq,
  event.pos,
  insert.pos){
  return(NucleotideSequence(string="AATGGCC"))
}

# get the generator function
i$generateBy
# generate insert
generateInsert(i)
# restore old generator
i$generateBy<-old.gen
# generate insert
generateInsert(i)
```

---

**setHandler.Event**

*Forbidden action: setting the handler function of an Event object*

---

**Description**

Forbidden action: setting the handler function of an Event object. The handler function is typically set by a `getEventsAtSite.*` method generating the Event object by directly modifying the `this$handler` field or by the `.setHandler()` method.

**Usage**

```r
## S3 method for class 'Event'
setHandler(this, value, ...)
```

**Arguments**

- **this**: An object.
- **value**: Not used.
- **...**: Not used.
setld.PhyloSim

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

---

setld.PhyloSim  
*Forbidden action: setting the unique identifier of a PhyloSim object*

Description

Forbidden action: setting the unique identifier of a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
setld(this, value, ...)
```

Arguments

- `this`  
  An object.
- `value`  
  Not used.
- `...`  
  Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.
setId.Process

Forbidden action: setting the unique Process object identifier

Description
Forbidden action: setting the unique Process object identifier.

Usage

```r
## S3 method for class 'Process'
setId(this, value, ...)
```

Arguments

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Process`.


setId.QMatrix

Forbidden action: setting the unique identifier for a QMatrix object

Description

Forbidden action: setting the unique identifier for a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
setId(this, value, ...)
```
Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Thows an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

---

### `setId.Sequence` **Forbidden action: setting the unique identifier of a Sequence object**

Description

Forbidden action: setting the unique identifier of a Sequence object.

Usage

```
## S3 method for class 'Sequence'
setId(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`. 
Description

Set the insert hook function.

The insert hook allows to make various modifications on the insert before performing the insertion. The function must have the following arguments: seq (the sequence object to insert), target.seq (the target Sequence object), event.pos (the position of the site which generated the insertion event), insert.pos (the position of the insertion).

The insert hook function is called by the insertion event handler function. The insert hook takes the sequence generated by the generateInsert method through the "seq" argument. The function must return a Sequence object, which will be inserted in the target sequence.

Usage

```r
## S3 method for class 'GeneralInsertor'
setInsertHook(this, value, ...)
```

Arguments

- **this**: A GeneralInsertor object.
- **value**: A function object.
- **...**: Not used.

Value

The function object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create a GeneralInsertor object
i <- GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}, # inserts between 5 and 10
  template.seq=NucleotideSequence(length=5, processes=list(list(JC69())))
)
# set a dummy insert hook
```
setInsertHook(i, function(seq)(return(seq)))
# set a new insert hook via virtual field
i$insertHook <- function(seq)(
  seq$processes <- list(list(GTR()))  # replace the substitution process
  return(seq)
)
# get the insert hook via virtual field
i$insertHook
# get the insert hook
getInsertHook(i)

Description

Set the insertion tolerance site-process specific parameter for a collection of Site objects aggregated by a Sequence object. This method does some error checking and the calls setParameterAtSites(this = this, process = process, index = index, value = value).

See `setParameterAtSites.Sequence` for more details.

Usage

```r
## S3 method for class 'Sequence'
setInsertionTolerance(this, process, value, index, ...)
```

Arguments

- **this**: A Sequence object.
- **process**: A Process object.
- **value**: A numeric vector, recycled if shorter than the index vector.
- **index**: An integer vector specifying a set of positions. It is set to 1:length if omitted.
- **...**: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

`setParameterAtSites.Sequence`
Get the transition transversion rate ratio

Description

Get the transition transversion rate ratio.

Usage

```r
## S3 method for class 'F84'
setKappa(this, value, ...)
```

Arguments

- `this`  
  A F84 object.
- `value`  
  A numeric vector of length one.
- `...`  
  Not used.

Value

The new value of Kappa (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F84`.

Examples

```r
# construct a F84 object
p<-F84();
# set/get Kappa
setKappa(p,2)
getKappa(p)
# set/get Kappa via virtual field
p$kappa<-4
p$kappa
# get object summary
summary(p)
```
Description

Set the transition/transversion rate ratio.

Usage

```r
## S3 method for class 'GY94'
setKappa(this, value, ...)
```

Arguments

- `this` A GY94 object.
- `value` A numeric vector of length one.
- `...` Not used.

Value

The new value of kappa.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GY94`.

Examples

```r
# create a GY94 object
p <- GY94()
# set/get kappa
setKappa(p, 2)
getKappa(p)
# set/get kappa via virtual field
p$kappa <- 3
p$kappa
```
**setLength.Sequence**

Forbidden action: setting the length of a Sequence object. The length of the Sequence objects can be specified when the object is constructed, or modified later by the "insertSequence" and "deleteSubSequence" methods.

**Usage**

```r
## S3 method for class 'Sequence'
setLength(this, value, ...)
```

**Arguments**

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Sequence`.

---

**setLengthParam1.FastFieldDeleter**

*Set the first length parameter*

**Description**

Set the first length parameter.

**Usage**

```r
## S3 method for class 'FastFieldDeleter'
setLengthParam1(this, value, ...)
```
setLengthParam2.FastFieldDeletor

Arguments

- this: A FastFieldDeletor object.
- value: A numeric vector of length one.
- ...: Not used.

Value

value (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

Examples

```r
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get the first length parameter
setLengthParam1(p, 0.5)
getLengthParam1(p)
# set/get the first length parameter via virtual field
p$lengthParam1 <- 0.2
p$lengthParam1
```

Description

Set the second length parameter.

Usage

```r
## S3 method for class 'FastFieldDeletor'
setLengthParam2(this, value, ...)
```

Arguments

- this: A FastFieldDeletor object.
- value: A numeric vector of length one.
- ...: Not used.
**setLogFile.PhyloSim**

**Value**

value (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see FastFieldDeletor.

**Examples**

```r
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get the second length parameter
setLengthParam2(p, 0.5)
ge_lengthParam2(p)
# set/get the second length parameter via virtual field
p$lengthParam2 <- 0.2
p$lengthParam2
```

---

**setLogFile.PhyloSim**  *Set the name of the file used for logging*

**Description**

Set the name of the file used for logging.

**Usage**

```r
## S3 method for class 'PhyloSim'
setLogFile(this, value, ...)
```

**Arguments**

- **this**: A PhyloSim object.
- **value**: The name of the file used for logging.
- **...**: Not used.

**Value**

The new logfile.

**Author(s)**

Botond Sipos, Gregory Jordan
See Also

For more information see `PhyloSim`.

Examples

```r
# Create a PhyloSim object
sim<-PhyloSim();
# get the name of the log file
getLogFile(sim)
# modify log file name
setLogFile(sim,"OldLog.txt")
# get/set log file name via virtual field
sim$logFile
sim$logFile<-"NewLog"
sim$logFile
```

Description

Set the log level for a given PhyloSim object.

No logging is performed if the log level is negative. If the log level is zero, the messages passed to the `log` method will be written in the log file. If the log level is positive, the messages passed to the `debug` method are saved as well.

The default log level is -1. The specified file will be truncated in the case it already exists.

Usage

```r
## S3 method for class 'PhyloSim'
setLogLevel(this, value, ...)
```

Arguments

- **this**: A PhyloSim object.
- **value**: The new log level as an integer.
- **...**: Not used.

Value

The new level as an integer vector of length one.

Author(s)

Botond Sipos, Gregory Jordan
setMatrix.QMatrix

See Also

getLogLevel PhyloSim

Examples

```r
# Create a PhyloSim object
sim<-PhyloSim();
# get/set log level
gLogLevel(sim)
sLogLevel(sim,0)
# set/get log level via virtual field
sim$logLevel<- -1
sim$logLevel
```

Description

Forbidden action: setting the unscaled rate matrix for a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
setMatrix(this, value, ...)
```

Arguments

- `this` An object.
- `value` Not used.
- `...` Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.
setMaxLength.ContinuousDeletor

*Set the maximum length*

**Description**

Set the maximum length.

**Usage**

```r
## S3 method for class 'ContinuousDeletor'
setMaxLength(this, value, ...)
```

**Arguments**

- `this` : A ContinuousDeletor object.
- `value` : A numeric (integer) vector of length one.
- `...` : Not used.

**Value**

The new maximum length.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ContinuousDeletor`.

**Examples**

```r
# create object
do<-'ContinuousDeletor'(rate=1)
# set length sampling expression via virtual field
do$dist<-expression(rnorm(1,mean=6,sd=3))
# set/get maxLength
setMaxLength(o, 3)
getMaxLength(o)
# set/get maxLength via virtual field
o$maxLength<-10
o$maxLength
# propose a length
proposeLength(o)
```
setMaxLength.ContinuousInsertor

Set the maximum length

Description

Set the maximum length.

Usage

```r
## S3 method for class 'ContinuousInsertor'
setMaxLength(this, value, ...)
```

Arguments

- `this` A ContinuousInsertor object.
- `value` A numeric (integer) vector of length one.
- `...` Not used.

Value

The new maximum length.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ContinuousInsertor`.

Examples

```r
# create object
o <- ContinuousInsertor(rate = 1)
# set length sampling expression via virtual field
o$dist <- expression(rnorm(1, mean = 6, sd = 3))
# set/get maxLength
setMaxLength(o, 3)
getMaxLength(o)
# set/get maxLength via virtual field
o$maxLength <- 10
o$maxLength
# propose a length
proposeLength(o)
```
**setMethodsList**. **PSRoot**  
*Forbidden action: setting the list of applicable methods for an object*

**Description**  
Forbidden action: setting the list of applicable methods for an object.

**Usage**  
```r  
## S3 method for class 'PSRoot'  
setMethodsList(this, value, ...)
```

**Arguments**  
- **this**: An object.  
- **value**: Not used.  
- **...**: Not used.

**Value**  
Throws an error.

**Author(s)**  
Botond Sipos, Gregory Jordan

**See Also**  
For more information see **PSRoot**.

---

**setName**. **Event**  
*Set the name of an Event object*

**Description**  
Set the name of an Event object.

**Usage**  
```r  
## S3 method for class 'Event'  
setName(this, new.name, ...)
```
setName.PhyloSim

Arguments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>An Event object.</td>
</tr>
<tr>
<td>new.name</td>
<td>A character vector of length one.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

Value

The new name (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

```r
# create an Event object
e <- Event()
# set event name
setName(e, "Insertion")
# get event name
e$name
# set name via virtual field
e$name <- "Deletion"
e$name
```

Description

Set the name of a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
setName(this, new.name, ...)
```

Arguments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>A PhyloSim object.</td>
</tr>
<tr>
<td>new.name</td>
<td>A character vector of length one.</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>
**setName.Process**

**Value**

The new name.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see *PhyloSim*.

**Examples**

```r
# create a PhyloSim object
o <- PhyloSim();
# set/get name
setName(o, "MySim");
getName(o, "MySim");
# set/get name via virtual field
o$name <- "George";
o$name
```

---

### setDescription.Process

**Set the name of a Process object**

**Description**

Set the name of a Process object.

This method also updates the unique identifier of the Process object.

**Usage**

```r
## S3 method for class 'Process'
setName(this, new.name, ...)
```

**Arguments**

- **this**: A Process object.
- **new.name**: A character vector of length one.
- **...**: Not used.

**Value**

Returns the new name (invisible).
setName.QMatrix

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

Examples
```
# create object
p<-Process()
# get name
p$name
# set new name
g GetName(p)
# get name and id
p$name
p$id
# set new name via virtual field
p$name<-"Procy"
p$name
```

Description
Set the name of a QMatrix object.

Usage
```
## S3 method for class 'QMatrix'
setName(this, value, ...)
```

Arguments
- this A QMatrix object.
- value A character vector of length one.
- ... Not used.

Value
The new object name.

Author(s)
Botond Sipos, Gregory Jordan
See Also

For more information see QMatrix.

Examples

```r
# create a QMatrix object
m <- QMatrix()
# set/get name
setName(m, "Susie Q")
getName(m)
# set/get name via virtual field
m$name <- "Q"
m$name
```

Description

Set the name of a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setName(this, new.name, ...)
```

Arguments

- **this**: A Sequence object.
- **new.name**: A character vector of length one.
- **...**: Not used.

Value

The new object name (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.
Examples

```r
# create a Sequence object
s <- Sequence(name = "MySeq", length = 4)
# get sequence name
s$name
# rename object
setName(s, "50")
# rename via virtual field
s$name
s$name <- "SeqSeq"
s$name
```

Description

Forbidden action: setting the number of edges for phylo object aggregated by a PhyloSim object.

Usage

```r
## S3 method for class 'PhyloSim'
setNedges(this, value, ...)
```

Arguments

- `this` An object.
- `value` Not used.
- `...` Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.
setNodes.PhyloSim

Forbidden action: setting the node identifiers for a PhyloSim object

Description
Forbidden action: setting the node identifiers for a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
setNodes(this, value, ...)

Arguments

- this: An object.
- value: Not used.
- ...: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see PhyloSim.

setNtips.PhyloSim

Forbidden action: setting the number of the tips for a phylo object aggregated by a PhyloSim object

Description
Forbidden action: setting the number of the tips for a phylo object aggregated by a PhyloSim object.

Usage

## S3 method for class 'PhyloSim'
setNtips(this, value, ...)


**Arguments**

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see **PhyloSim**.

---

**setOmegas.CodonSequence**

*Set the omegas for a collection of sites*

**Description**

Set the omegas for a collection of sites.

**Usage**

```r
## S3 method for class 'CodonSequence'
setOmegas(this, process, value, index, ...)
```

**Arguments**

- **this**: A CodonSequence object.
- **process**: A process object inheriting from GY94.
- **value**: A vector containing the new values of the site-process specific parameter, recycled if shorter than the index vector.
- **index**: A vector of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

**Value**

A numeric vector.

**Author(s)**

Botond Sipos, Gregory Jordan
setParameterAtSite.Process

Set the value of a site-process specific parameter in a Site object attached to a Process object.

Description

Set the value of a site-process specific parameter in a Site object attached to a Process object. The Process object must be attached to the Site object. The new value must be compatible with the type of the site-process specific parameter.

Usage

```r
## S3 method for class 'Process'
setParameterAtSite(this, site, id, value, ...)
```

Arguments

- `this`: A Process object.
- `site`: A Site object.
- `id`: The identifier of the site-process specific parameter.
- `value`: The new value for the parameter.
- `...`: Not used.

Value

The site-process specific parameter, which is a list containing the following keys: id, name, value, type.

Examples

```r
# create a GY94 process
p<-GY94()
# create a CodonSequence object,
# attach a process p
s<-CodonSequence(length=20, processes=list(list(p)))
# set omega values in range 1:5
setOmegas(s,p,c(0.5,1,1.5),1:5)
# get omega values from siutes 1,2,3,10, and 20
getOmegas(s,p,c(1:3,10,20))
```
setParameterAtSites.Sequence

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

Examples

```r
# create a Site and a Process object
a <- BinaryAlphabet()
s <- Site(alphabet = a)
p <- Process(alphabet = a)
# attach the process
attachProcess(s, p)
# set the value of the rate multiplier
setParameterAtSite(p, s, "rate.multiplier", 2)
# get the value of the rate multiplier
getParameterAtSite(p, s, "rate.multiplier")
```

Description
Set the values of a site-process specific parameter for a process and a collection of Site objects aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setParameterAtSites(this, process, id, value, index, ...)
```

Arguments

- `this` A Sequence object.
- `process` A valid Process object.
- `id` The identifier of the site-process specific parameter.
- `value` A vector containing the new values of the site-process specific parameter, recycled if shorter than the index vector. It should be consistent with the type of the parameter.
- `index` An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- `...` Not used.
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

Site Process For more information see Sequence.

Examples

# create a sequence, attach a process
p<-k80()
s<-Sequence(length=6,alphabets=list(NucleotideAlphabet()),processes=list(list(p)))
# set a new pattern of rate multipliers in the range 1:3,
  # the default value is 1.0 by the way
setParameterAtSites(s,p,"rate.multiplier",c(2,3),1:3)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")
# set a new value for the whole sequence
setParameterAtSites(s,p,"rate.multiplier",0.5)
# get rate multipliers
getParameterAtSites(s,p,"rate.multiplier")

Description

Set the phylo object for a PhyloSim object.

The internal structure of the provided phylo object is reordered in a cladewise fashion.

Usage

```
## S3 method for class 'PhyloSim'
setPhylo(this, value, ...)
```

Arguments

- **this**: A PhyloSim object.
- **value**: A phylo object created by the ape package.
- **...**: Not used.
setPosition.Event

Value

A phylo object or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

The PhyloSim class, the ape package.

Examples

# create a PhyloSim object
sim <- PhyloSim();
# create a phylo object
tree <- rcoal(3);
# get/set phylo object
setPhylo(sim, tree);
getPhylo(sim, tree);
# get/set phylo object via virtual field
sim$tree <- rcoal(5);
sim$tree;

description

Set the position of the Site object associated to an Event object.

The position field is usually not modified directly, but set by the getEvents.Sequence method.

The position is *not* set by Site methods as getEventsAtSite.Site.

Usage

## S3 method for class 'Event'
setPosition(this, value, ...)

Arguments

this An Event object.
value The position.
... Not used.

Value

The new position (invisible).
Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

# Note: the following example is not too useful

# create a sequence and attach a process
s <- NucleotideSequence(string = "ATGC", processes = list(list(JC69())))
# get the first active event from the first site
e <- getEvents(s, 1)[[1]]
# get event position
e$position
# set the position of the site associated with e
setPosition(e, 2)
# get position via virtual field
e$position
# set position via virtual field
e$position <- 1
e$position

setProbs.DiscreteDeletor

Set the deletion length probabilities

Description

Set the deletion length probabilities.

The sizes virtual field must be set before setting the length probabilities. The length of the provided numeric vector must match with the length of the vector stored in the sizes virtual field. The vector is rescaled if the values do not sum to one and a warning is issued.

Usage

## S3 method for class 'DiscreteDeletor'
setProbs(this, value, ...)

Arguments

this A DiscreteDeletor object.
value A numeric vector containing the length probabilities.
... Not used.
**Value**

The vector of probabilities.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteDeletor`.

**Examples**

```r
# create a DiscreteDeletor object
d <- DiscreteDeletor(rate = 1, sizes = 1:3)
# set/get length probabilities
setProbs(d, c(1/3, 1/3, 1/3))  # equal probabilities
getProbs(d)
# set/get length probabilities via virtual field
x <- c(2, 2, 1)
# normalize x
x <- x / sum(x)
d$d$probs <- x
d$d$probs
```

---

**Description**

Set the insertion length probabilities.

The `sizes` virtual field must be set before setting the length probabilities. The length of the provided numeric vector must match with the length of the vector stored in the `sizes` virtual field. The vector is rescaled if the values do not sum to one and a warning is issued.

**Usage**

```r
## S3 method for class 'DiscreteInsertor'
setProbs(this, value, ...)
```

**Arguments**

- **this**
  A DiscreteInsertor object.

- **value**
  A numeric vector containing the length probabilities.

- **...**
  Not used.
The vector of probabilities.

Botond Sipos, Gregory Jordan

For more information see `DiscreteInsertor`.

```
# create a DiscreteInsertor object
i<-DiscreteInsertor(rate=1, sizes=1:3)
# set/get length probabilities
setProbs(i,c(1/3,1/3,1/3)) # equal probabilities
getProbs(i)
# set/get length probabilities via virtual field
x<-c(2,2,1)
# normalize x
x<-x/sum(x)
i$probs<-x
i$probs
```

---

**setProcess.Event**  
*Set the generator process for an Event object*

Set the generator process for an Event object.

```r
## S3 method for class 'Event'
setProcess(this, new.proc, ...)
```

**Arguments**

- `this`  
  An Event object.

- `new.proc`  
  A valid Process object.

- `...`  
  Not used.

**Value**

A Process object.
**setProcess.QMatrix**  

**Author(s)**  
Botond Sipos, Gregory Jordan

**See Also**  
For more information see Event.

**Examples**

```r
# create an Event object
e <- Event()
# set a generator process for e
setProcess(e, Process())
# get generator process
e$process
# set process via virtual field
e$process <- QMatrix()
e$process
```

---

**Description**

Associate a process object with a QMatrix object.

**Usage**

```r
## S3 method for class 'QMatrix'
setProcess(this, value, ...)
```

**Arguments**

- **this**: A QMatrix object.
- **value**: A Process object.
- **...**: Not used.

**Value**

The Process object.

**Author(s)**

Botond Sipos, Gregory Jordan
setProcesses.Sequence

Specify a set of Process objects to be attached to a set of Site objects aggregated by a Sequence object

Description

Specify a set of Process objects to be attached to a set of Site objects aggregated by a Sequence object. The Process objects in a given inner list correspond to the set of processes to be attached to one Site object. Process objects already attached to a given Site are skipped. Attached processes which are not members of the list are detached, so specifying an empty list will detach all processes.

Usage

```r
## S3 method for class 'Sequence'
setProcesses(this, value, index, ...)
```

Arguments

- **this**: A Sequence object.
- **value**: A list of list of Process objects, recycled if shorter than the index vector.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Examples

```r
# Create a GeneralSubstitution object
p <- GeneralSubstitution(alphabet=BinaryAlphabet())
p
# get the associated QMatrix object from p
m <- p$qMatrix
summary(m)
# get the associated process from m
m$process
# clone p
pp <- clone(p)
# associate m with pp
pp$qMatrix <- m
# associate pp with m
m$process <- pp
m$process
```

See Also

For more information see QMatrix.
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

```r
# create a sequence of length 5
s <- Sequence(length=5, alphabets=list(NucleotideAlphabet()));
# set a pattern of processes
setProcesses(s, list(list(JC69(), K81())));
# get attached processes
s$processes
# detach all processes from range 1:3
setProcesses(s, list(list()), 1:3);
# detach all processes via virtual field
s$processes <- list(list());
# create a process pattern in the full sequence via virtual field
s$processes <- list(list(JC69()), list(GTR(), K80()));
s$processes
```
Arguments

- **this**: A Site object.
- **value**: A list of valid Process objects.
- **...**: Not used.

Value

The Site object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

Examples

```r
# create a Site object
s<-Site(alphabet=NucleotideAlphabet())
# create some processes
p1<-JC69(); p2<-K80(); p3<-DiscreteInsertor(rate=1);
# attach the processes
setProcesses(s,list(p1,p2,p3))
# attach one more process via virtual field
s$processes<-c(s$processes,list(GTR()))
# get the list of attached processes
s$processes
# detach all processes via virtual field
s$processes<-list()
```

---

**setProposeBy GeneralInDel**

*Set the function used for proposing indel lengths*

Description

Set the function used for proposing indel lengths.

The function must return a numeric vector of length one. The function must have an argument named "process" which will hold the calling process object.

Usage

```r
## S3 method for class 'GeneralInDel'
setProposeBy(this, value, ...)
```
Arguments

this  A GeneralInDel object.
value A function object returning a numeric vector of length one.
...  Not used.

Value

The function object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralInDel.

Examples

```r
# create a GeneralInDel object
# proposing events with a constant length of 5
o<-GeneralInDel(rate=1, propose.by=function(process){return(5)});
# set/get the proposeBy function
setProposeBy(o,value=function(process){return(6)})
getProposeBy(o)
# set/get proposeBy function via virtual field
o$proposeBy<-function(process){return(3)}
o$proposeBy
```
Arguments

this
value
... 

A GeneralSubstitution object.
A QMatrix object.
Not used.

Value

The QMatrix object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.

Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p<-GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("A->B"=1,"B->A"=1))
# get the QMatrix object
getQMatrix(p)
# get the QMatrix object via virtual field
q<-p$qMatrix
# tweak with the QMatrix
setRate(q,"A->B",2)
# set a new QMatrix for p
setQMatrix(p,q)
summary(p)
# set new QMatrix via virtual field
setRate(q,"A->B",2)
p$qMatrix<-q
summary(p)

---

setQMatrix.ToleranceSubstitution

Set the QMatrix object aggregated by a ToleranceSubstitution object

Description

Set the QMatrix object aggregated by a ToleranceSubstitution object.
This method is mostly used internally.
Usage

```r
## S3 method for class 'ToleranceSubstitution'
setQMatrix(this, value, ...)
```

Arguments

- `this`: A `ToleranceSubstitution` object.
- `value`: A `QMatrix` object.
- `...`: Not used.

Value

The `QMatrix` object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ToleranceSubstitution`.

Examples

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p <- ToleranceSubstitution(alphabet = BinaryAlphabet(), rate.list = list("1->0" = 1, "0->1" = 1))
# get the QMatrix object
getQMatrix(p)
# get the QMatrix object via virtual field
q <- p$qMatrix
# tweak with the QMatrix
setRate(q, "0->1", 2)
# set a new QMatrix for p
setQMatrix(p, q)
summary(p)
# set new QMatrix via virtual field
setRate(q, "1->0", 2)
p$qMatrix <- q
summary(p)
```
setRate.Event  

*Set the rate of an Event object*

**Description**

Set the rate of an Event object.

**Usage**

```r
t # S3 method for class 'Event'
setRate(this, value, ...)
```

**Arguments**

- `this`  : An Event object.
- `value` : The event rate.
- `...`  : Not used.

**Value**

The new value of the rate (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# create an Event object
e <- Event(rate=0.1)
# set a new rate
setRate(e, 0.2)
# get rate via virtual field
e$rate
# set rate via virtual field
e$rate <- 0.5
e$rate
```
setRate.GeneraInDel  

Description

Set the general rate.

Usage

```r
## S3 method for class 'GeneralInDel'
setRate(this, value, ...)
```

Arguments

- `this`  
  A GeneralInDel object.

- `value`  
  The new general rate (a numeric vector of length one).

- `...`  
  Not used.

Value

The new general rate.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.

Examples

```r
# create a GeneralInDel object
o <- GeneralInDel(rate = 0.5)
# get/set general rate
gRate(o)
setRate(o, 1.5)
# get/set rate via virtual field
o$rate
o$rate <- 0.3
o$rate
```
setRate.Gene

Set an unscaled rate for an event from a GeneralSubstitution object

Description

Set an unscaled rate for an event from a GeneralSubstitution object.

This method sets the element corresponding to a given event in the unscaled Q matrix. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

Modifying any rate in the unscaled Q matrix will trigger the re-scaling of the whole matrix. The rescaled rates (used during simulations) are returned by the getEventRate method.

This method doesn’t modify the site specific rate multipliers.

Usage

### S3 method for class 'GeneralSubstitution'

setRate(this, name=NA, value, from=NA, to=NA, ...)

Arguments

- **this** A GeneralSubstitution object.
- **name** The name of the event.
- **from** The initial state.
- **value** The new value of the rate.
- **to** Target state.
- **...** Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralSubstitution.
Examples

# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# set the unscaled rate by event name
setRate(p,"0->1",2)
# get the unscaled rate of "0->1" by name
getRate(p,"0->1")
# set the unscaled rate by states
setRate(p,"0->1",0.5)
# get the unscaled rate of "0->1" by states
getRate(p,from="0",to="1")

setRate.GY94

Set an unscaled rate for an event from a GY94 object

Description

Set an unscaled rate for an event from a GY94 object.
See setRate.GeneralSubstitution.

Usage

## S3 method for class 'GY94'
setRate(this, name=NA, value, from=NA, to=NA, ...)

Arguments

this A GeneralSubstitution object.
name The name of the event.
from The initial state.
value The new value of the rate.
to Target state.
... Not used.

Value

A Numeric vector of length one.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GY94.
**setRate.QMatrix**

Set an unscaled rate in a QMatrix object.

This method sets the element corresponding to a given event in the unscaled rate matrix. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

**Usage**

```r
## S3 method for class 'QMatrix'
setRate(this, name=NA, value, from=NA, to=NA, scale=TRUE, diag=TRUE, guess.equ=TRUE, ...)
```

**Arguments**

- `this`: A QMatrix object.
- `name`: The name of the event.
- `value`: The new value of the rate.
- `from`: The initial state.
- `to`: Target state.
- `scale`: Call rate rescaling.
- `diag`: Calculate diagonal elements.
- `guess.equ`: Guess equilibrium distribution.
- `...`: Not used.

**Value**

A Numeric vector of length one.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [QMatrix](#).
setRate.ToleranceSubstitution

Examples

```r
# create a QMatrix object
# provide an Alphabet object and the rates
m <- QMatrix(alphabet=BinaryAlphabet(), rate.list=list("1->0"="1","0->1"="1"))
# set the unscaled rate by event name
setRate(m,"0->1",2)
# get the unscaled rate of "0->1" by name
getRate(m,"0->1")
# set the unscaled rate by states
setRate(m,"0->1",0.5)
# get the unscaled rate of "0->1" by states
getRate(m,from="0",to="1")
```

setRate.ToleranceSubstitution

*Set an unscaled rate for an event from a ToleranceSubstitution object*

Description

Set an unscaled rate for an event from a ToleranceSubstitution object.

This method sets the element corresponding to a given event in the unscaled Q matrix. The event can be specified by the initial and target states ("from" and "to" arguments), or by the event name ("from->to"). The event name takes precedence over the "from" and "to" arguments.

Modifying any rate in the unscaled Q matrix will trigger the re-scaling of the whole matrix. The rescaled rates (used during simulations) are returned by the `getEventRate` method.

This method doesn’t modify the site specific rate multipliers.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
setRate(this, name=NA, value, from=NA, to=NA, ...)
```

Arguments

- `this` A ToleranceSubstitution object.
- `name` The name of the event.
- `from` The initial state.
- `value` The new value of the rate.
- `to` Target state.
- `...` Not used.
**Value**
A Numeric vector of length one.

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
For more information see `ToleranceSubstitution`

**Examples**

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p <- ToleranceSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=1))
# set the unscaled rate by event name
setRate(p,"0->1",2)
# get the unscaled rate of "0->1" by name
getRate(p,"0->1")
# set the unscaled rate by states
setRate(p,"0->1",0.5)
# get the unscaled rate of "0->1" by states
getRate(p,from="0",to="1")
```

---

**Description**
Setting the rates for a GeneralSubstitution object.

This method set the rates in the *unscaled* \(Q\) matrix based on the provided list containing even names and the associated rates. The rate must be specified for every event!

**Usage**

```r
## S3 method for class 'GeneralSubstitution'
setRateList(this, value, ...)
```

**Arguments**

- **this**
  A GeneralSubstitution object.

- **value**
  A list with the events names and the associated rates.

- **...**
  Not used.
**Value**

The GeneralSubstitution object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneralSubstitution`.

**Examples**

```
# create a GeneralSubstitution object
# provide an Alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the event rates from the unscaled Q matrix
getRateList(p)
# get rates from the unscaled rate matrix via virtual field
p$pageRateList
# set rates in the unscaled rate matrix
setRateList(p, list("0->1"=1,"1->0"=1))
p$pageRateList
# set rates in the unscaled rate matrix via virtual field
p$pageRateList<-list("0->1"=3,"1->0"=1);
# check the content of the associated QMatrix object
summary(p$QMatrix)
```

---

**setRateList.QMatrix**

Setting the unscaled rates stored in a QMatrix object

**Description**

Setting the unscaled rates stored in a QMatrix object.

This method set the rates in the *unscaled* Q matrix based on the provided list containing even names and the associated rates. The rate must be specified for every event!

**Usage**

```
## S3 method for class 'QMatrix'
setRateList(this, value, ...)
```

**Arguments**

- `this`: A QMatrix object.
- `value`: A list with the events names and the associated rates.
- `...`: Not used.
Value

The QMatrix object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see **QMatrix**.

Examples

```r
# create a GeneralSubstitution object
# provide an alphabet object and the rates
p <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("1->0"=1,"0->1"=3))
# get the QMatrix object from p
m <- p$QMatrix
# get the event rates from the unscaled Q matrix
getRateList(m)
# get rates from the unscaled rate matrix via virtual field
m$rateList
# set rates in the unscaled rate matrix
setRateList(m, list("0->1"=1,"1->0"=1))
m$rateList
# set rates in the unscaled rate matrix via virtual field
m$rateList <- list("0->1"=3,"1->0"=1);
m$rateList
```

---

**setRateList.ToleranceSubstitution**

*Setting the rates for a ToleranceSubstitution object*

Description

Setting the rates for a ToleranceSubstitution object.

This method set the rates in the unscaled Q matrix based on the provided list containing even names and the associated rates. The rate must be specified for every event!

Usage

```r
# S3 method for class 'ToleranceSubstitution'
setRateList(this, value, ...)
```
**setRateMultipliers.Sequence**

Set the values of the rate multiplier parameters for a given Process object and a collection of Site object aggregated by a Sequence object. This method just calls `setParameterAtSites(this=this, process=process, id="rateMultiplier")`. See `setParameterAtSites.Sequence` for details.

---

**Arguments**

- **this**
  A ToleranceSubstitution object.

- **value**
  A list with the events names and the associated rates.

- **...**
  Not used.

**Value**

The ToleranceSubstitution object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ToleranceSubstitution`.

**Examples**

```r
# create a ToleranceSubstitution object
# provide an Alphabet object and the rates
p <- ToleranceSubstitution(alphabet=BinaryAlphabet(), rate=list(list("1->0"=1,"0->1"=3)))
# get the event rates from the unscaled Q matrix
getRateList(p)
# get rates from the unscaled rate matrix via virtual field
getRateList(p)
# set rates in the unscaled rate matrix
setParameterAtSites(p, list("0->1"=1,"1->0"=1))
setParameterAtSites(p, list("0->1"=3,"1->0"=1));
# check the content of the associated QMatrix object
summary(p$QMatrix)
```
Usage

## S3 method for class 'Sequence'
setRateMultipliers(this, process, value, index, ...)

Arguments

- **this**: A Sequence object.
- **process**: A valid Process object.
- **value**: A numeric vector containing the new values of the site-process specific parameter, recycled if shorter than the index vector.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.

Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

setParameterAtSites.Sequence

---

Description

Forbidden action: getting the list of rate parameters.

This model has no rate parameters.

Usage

## S3 method for class 'F81'
setRateParam(this, name, value, ...)

Arguments

- **this**: An object.
- **name**: Not used.
- **value**: Not used.
- **...**: Not used.
setRateParam.F84

**Value**

The object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see F81.

---

**setRateParam.F84**  
*Set the value of a rate parameter*

---

**Description**

Set the value of a rate parameter.

The rate parameters are: Kappa.

**Usage**

```r
## S3 method for class 'F84'
setRateParam(this, name, value, ...)
```

**Arguments**

- **this**: A F84 object.
- **name**: The name of the rate parameter.
- **value**: A numeric vector of length one.
- **...**: Not used.

**Value**

The new value of the rate parameter (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see F84.
Examples

```r
# construct a F84 object
p<-F84();
# set/get a rate parameter
setRateParam(p,"Kappa",4)
getRateParam(p,"Kappa")
# get object summary
summary(p)
```

Description

Set the value of a rate parameter.

The rate parameters are named as in PAML (see PAML documentation: [http://bit.ly/9SQK2f](http://bit.ly/9SQK2f)).

Usage

```r
## S3 method for class 'GTR'
setRateParam(this, name, value, ...)
```

Arguments

- **this**: A GTR object.
- **name**: The name of the rate parameter.
- **value**: A numeric vector of length one.
- **...**: Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GTR`. 
Examples

```r
# construct a GTR object
p <- GTR();
# set/get a rate parameter
setRateParam(p,"a",4)
generateParam(p,"a")
# get object summary
summary(p)
```

>Description

Set the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'HKY'
setRateParam(this, name, value, ...)
```

Arguments

- `this` : An HKY object.
- `name` : The name of the rate parameter.
- `value` : A numeric vector of length one.
- `...` : Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see HKY.
Examples

```r
# construct HKY object
p<-HKY();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```

Description

Set the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'K80'
setRateParam(this, name, value, ...)
```

Arguments

- `this` A K80 object.
- `name` The name of the rate parameter.
- `value` A numeric vector of length one.
- `...` Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `K80`. 
Examples

```
# construct a K80 object
p<-k80();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```

Description

Set the value of a rate parameter.

The rate parameters are: Alpha, Beta, Gamma.

Usage

```
## S3 method for class 'K81'
setRateParam(this, name, value, ...)
```

Arguments

- `this`: A K81 object.
- `name`: The name of the rate parameter.
- `value`: A numeric vector of length one.
- `...`: Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `K81`. 
Examples

```r
# construct a K80 object
p<-K80();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Gamma")
# get object summary
summary(p)
```

Description

Set the value of a rate parameter.

The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'T92'
setRateParam(this, name, value, ...)
```

Arguments

- `this` A T92 object.
- `name` The name of the rate parameter.
- `value` A numeric vector of length one.
- `...` Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see T92.
Examples

# construct a T92 object
p<-T92();
# set/get a rate parameter
setRateParam(p,"Alpha",4)
getRateParam(p,"Beta")
# get object summary
summary(p)

Description

Set the value of a rate parameter.

The rate parameters are: Alpha1, Alpha2, Beta.

Usage

## S3 method for class 'TN93'
setRateParam(this, name, value, ...)

Arguments

this A TN93 object.
name The name of the rate parameter.
value A numeric vector of length one.
...
Not used.

Value

The new value of the rate parameter (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see TN93.
Examples

```r
# construct a TN93 object
p<-TN93();
# set/get a rate parameter
setRateParam(p,"Beta",4)
getRateParam(p,"Beta")
# get object summary
summary(p)
```

Description

Forbidden action: getting the list of rate parameters.
This model has no rate parameters.

Usage

```r
## S3 method for class 'F81'
setRateParamList(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

Value

The object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see [F81](#).
setRateParamList.F84

Set the rate parameters

Description

Set the rate parameters.

The rate parameters are: Kappa.

Usage

```r
## S3 method for class 'F84'
setRateParamList(this, value, ...)
```

Arguments

- `this`: A F84 object.
- `value`: A list containing the rate parameters.
- `...`: Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F84`.

Examples

```r
# create F84 object
p<-F84()
# set/get rate parameters
setRateParamList(p, list("Kappa"=3))
gRateParamList(p)
# set/get rate parameters via virtual field
p$rRateParamList<-list("Kappa"=2.5)
p$rRateParamList
# get object summary
summary(p)
```
Description

Set the rate parameters.

The rate parameters are named as in PAML (see PAML documentation: http://bit.ly/95QK2f).

Usage

```r
## S3 method for class 'GTR'
setRateParamList(this, value, ...)
```

Arguments

- **this**: A GTR object.
- **value**: A list containing the rate parameters.
- **...**: Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GTR.

Examples

```r
# create GTR object
p <- GTR()
# set/get rate parameters
setRateParamList(p, list(
  "a" = 1, "b" = 2, "c" = 3,
  "d" = 1, "e" = 2, "f" = 3
))
gRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(
  "a" = 4, "b" = 1, "c" = 4,
  "d" = 1, "e" = 4, "f" = 1
)
p$rateParamList
# get object summary
```
**setRateParamList.HKY**  

Set the rate parameters

**Description**

Set the rate parameters.

The rate parameters are: Alpha, Beta.

**Usage**

```r
## S3 method for class 'HKY'
setRateParamList(this, value, ...)
```

**Arguments**

- `this`  
  An HKY object.

- `value`  
  A list containing the rate parameters.

- `...`  
  Not used.

**Value**

The list of rate parameters (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see [HKY](#).

**Examples**

```r
# create HKY object
p<-HKY()
# set/get rate parameters
setRateParamList(p, list(  
  "Alpha"=1,  
  "Beta"=0.5  
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(  
  "Alpha"=1,  
  "Beta"=3
)```
Description

Set the rate parameters.
The rate parameters are: Alpha, Beta.

Usage

```r
## S3 method for class 'K80'
setRateParamList(this, value, ...)
```

Arguments

- `this` A K80 object.
- `value` A list containing the rate parameters.
- `...` Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see K80.

Examples

```r
# create K80 object
p<-K80()
# set/get rate parameters
setRateParamList(p,list(
    "Alpha"=1,
    "Beta"=0.5
  ))
getRateParamList(p)
# set/get rate parameters via virtual field
setRateParamList.K81

```
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3
)
p$rateParamList
# get object summary
summary(p)
```

---

**setRateParamList.K81**  Set the rate parameters

**Description**

Set the rate parameters.

The rate parameters are: Alpha, Beta, Gamma.

**Usage**

```
## S3 method for class 'K81'
setRateParamList(this, value, ...)
```

**Arguments**

- **this**  A K81 object.
- **value**  A list containing the rate parameters.
- **...**  Not used.

**Value**

The list of rate parameters (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see K81.

**Examples**

```
# create K81 object
p<-K81()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5,
```
"Gamma"=2
)
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
 "Alpha"=1,
 "Beta"=3,
 "Gamma"=2
)
p$rateParamList
# get object summary
summary(p)

setRateParamList.T92  Set the rate parameters

Description

Set the rate parameters.
The rate parameters are: Alpha, Beta.

Usage

## S3 method for class 'T92'
setRateParamList(this, value, ...)

Arguments

this A T92 object.
value A list containing the rate parameters.
... Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see T92.
setRateParamList.TN93

Examples

```r
# create a T92 object
p<-T92()
# set/get rate parameters
setRateParamList(p,list(
  "Alpha"=1,
  "Beta"=0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList<-list(
  "Alpha"=1,
  "Beta"=3
)
p$rateParamList
# get object summary
summary(p)
```

Description

Set the rate parameters.

The rate parameters are: Alpha1, Alpha2, Beta.

Usage

```r
## S3 method for class 'TN93'
setRateParamList(this, value, ...)
```

Arguments

- **this**: A TN93 object.
- **value**: A list containing the rate parameters.
- **...**: Not used.

Value

The list of rate parameters (invisible).

Author(s)

Botond Sipos, Gregory Jordan
See Also
For more information see TN93.

Examples

```r
# create TN93 object
p <- TN93()
# set/get rate parameters
setRateParamList(p, list(
  "Alpha1" = 1,
  "Alpha2" = 2,
  "Beta" = 0.5
))
getRateParamList(p)
# set/get rate parameters via virtual field
p$rateParamList <- list(
  "Alpha1" = 1,
  "Alpha2" = 1,
  "Beta" = 3
)
p$rateParamList
# get object summary
summary(p)
```

---

**setRootNode.PhyloSim**  
*Forbidden action: setting the identifier of the root node for a PhyloSim object*

---

**Description**
Forbidden action: setting the identifier of the root node for a PhyloSim object.

**Usage**
```
## S3 method for class 'PhyloSim'
setRootNode(this, value, ...)
```

**Arguments**
- **this**: An object.
- **value**: Not used.
- **...**: Not used.

**Value**
Throws an error.
Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see PhyloSim.

---

**setRootSeq.PhyloSim**  
*Set the root sequence for a PhyloSim object*

### Description
Set the root sequence for a PhyloSim object.

The root sequence will be used as a starting point for the simulation. The phylo object must be set before trying to set the root sequence object.

### Usage
```
## S3 method for class 'PhyloSim'
setRootSeq(this, value, ...)
```

### Arguments
- **this**: A PhyloSim object.
- **value**: A valid Sequence object.
- **...**: Not used.

### Value
The root Sequence object if succesfull, FALSE otherwise.

### Author(s)
Botond Sipos, Gregory Jordan

### See Also
For more information see PhyloSim. Sequence Process
Examples

```r
# create some objects
sim<-PhyloSim(phylr=rcoral(3));
seq<-NucleotideSequence(string="ATGCC");
# set/get root sequence
setRootSeq(sim, seq);
getRootSeq(sim, seq);
# set/get root sequence via virtual field
sim$rootSeq<-BinarySequence(string="11100011000");
sim$rootSeq;
```

setScale.BrownianInsertor

*Set scale parameter*

Description

Set scale parameter.

Usage

```r
## S3 method for class 'BrownianInsertor'
setScale(this, value, ...)
```

Arguments

- `this` A BrownianInsertor object.
- `value` A numeric vector of length one.
- `...` Not used.

Value

`value` (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `BrownianInsertor`. 
Examples

```r
# create a BrownianInsertor object
p <- BrownianInsertor(scale=0.002)
# set/get scale parameter
setScale(p, 0.1)
getScale(p)
# set/get scale parameter via virtual field
p$scale <- 0.1
p$scale
```

---

setScaledMatrix.QMatrix

Forbidden action: setting the scaled rate matrix for a QMatrix object

---

Description

Forbidden action: setting the scaled rate matrix for a QMatrix object.

Usage

```r
## S3 method for class 'QMatrix'
setScaledMatrix(this, value, ...)
```

Arguments

- `this`: An object.
- `value`: Not used.
- `...`: Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see QMatrix.
setSequence.Site  

**Assotiate a Sequence object with a Site object**

**Description**

Assotiate a Sequence object with a Site object.

**Usage**

```r
## S3 method for class 'Site'
setSequence(this, new.seq, ...)
```

**Arguments**

- **this**: A Site object.
- **new.seq**: A valid Sequence object.
- **...**: Not used.

**Value**

The Sequence object (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Site`.

**Examples**

```r
# create a site object
s<-Site()
# get associated Sequence object
s$sequence
# set associated Sequence object
setSequence(s,Sequence())
s$sequence
```
**setSequences.PhyloSim**

Forbidden action: setting the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object.

**Description**

Forbidden action: setting the Sequence objects associated with the nodes of a phylo object aggregated by a PhyloSim object.

**Usage**

```r
## S3 method for class 'PhyloSim'
setSequences(this, value, ...)
```

**Arguments**

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PhyloSim`.

---

**setSite.Event**

Associate an Event object with a Site object

**Description**

Associate an Event object with a Site object.

**Usage**

```r
## S3 method for class 'Event'
setSite(this, new.site, ...)
```
Arguments

this  An Event object.
new.site  A valid Site object.
...  Not used.

Value

The new associated Site object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Event.

Examples

# create an Event object
e<-Event()
# create some Site objects
s1<-Site(alphabet=NucleotideAlphabet(),state="A")
s2<-clone(s1); s2$state<="T"
# associate s1 with e
setSite(e,s1)
e$site
# associate s2 with e via virtual field
e$site<-s2
e$site

setSiteSpecificParamIds.Process

Forbidden action: setting the parameter identifiers of the site specific parameters from a Process object

Description

Forbidden action: setting the parameter identifiers of the site specific parameters from a Process object.

Usage

## S3 method for class 'Process'
setSiteSpecificParamIds(this, value, ...)

Forbidden action: setting the parameter identifiers of the site specific parameters from a Process object.
Forbidden action: setting the site specific parameter list for a Process object. Use .addSiteSpecificParameter to add new site specific parameters when implementing new processes.
See Also

For more information see Process.

---

**setSize.Alphabet**  
*Forbidden action: setting the symbol set size of an Alphabet object

**Description**

Forbidden action: setting the symbol set size of an Alphabet object.

**Usage**

```r
## S3 method for class 'Alphabet'
setSize(this, value, ...)
```

**Arguments**

- **this**: An Alphabet object.
- **value**: Not used.
- **...**: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

See Also

For more information see Alphabet.

---

**setSizes.DiscreteDeleter**  
*Set the sizes of the proposed deletions

**Description**

Set the sizes of the proposed deletions.

The provided numeric vector is rounded.
setSizes.DiscreteInsertor

Usage

```r
# S3 method for class 'DiscreteDeletor'
setSizes(this, value, ...)
```

Arguments

- `this`: A DiscreteDeletor object.
- `value`: A numeric vector.
- `...`: Not used.

Value

A vector of integers (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `DiscreteDeletor`.

Examples

```r
# create a DiscreteDeletor object
d <- DiscreteDeletor(rate = 1)
# set deletion sizes
setSizes(d, c(1, 2, 3))
# get deletion sizes
getSizes(d)
# set/get sizes via virtual field
d$sizes <- 1:10
d$sizes
```

---

**setSizes.DiscreteInsertor**

*Set the sizes of the proposed insertions*

---

Description

Set the sizes of the proposed insertions.

The provided numeric vector is rounded.

Usage

```r
# S3 method for class 'DiscreteInsertor'
setSizes(this, value, ...)
```
Arguments

- **this**: A DiscreteInsertor object.
- **value**: A numeric vector.
- **...**: Not used.

Value

A vector of integers (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `DiscreteInsertor`.

Examples

```r
# create a DiscreteInsertor object
i <- DiscreteInsertor(rate=1)
# set insertion sizes
setSizes(i, c(1, 2, 3))
# get insertion sizes
getSizes(i)
# set/get sizes via virtual field
i$sizes <- 1:10
i$sizes
```

Description

Set the state of a Site object.

Usage

```r
## S3 method for class 'Site'
setState(this, new.state, ...)  # S3 method for class 'Site'
ssetState(this, new.state, ...)  # S3 method for class 'Site'
```

Arguments

- **this**: A Site object.
- **new.state**: A character vector of length one, containing a symbol belonging to the attached Alphabet object.
- **...**: Not used.
Value

Returns the new state (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Site.

Examples

```r
# create a Site object with an Alphabet object attached
s <- Site(alphabet=Alphabet(symbols=c(0,1), state=1));
# set a new state
setState(s,"0")
# get state via virtual field
s$state
# set a new state via virtual field
s$state<-1
s$state
```

---

**setStates.Sequence**

*Set the states for a set of Site objects aggregated by a Sequence object*

Description

Set the states for a set of Site objects aggregated by a Sequence object. The value vector is recycled, which is useful when creating repeated patterns.

Usage

```r
## S3 method for class 'Sequence'
setStates(this, value, index, ...)
```

Arguments

- **this**: A Sequence object.
- **value**: A character vector containing the states (recycled if shorter than the index vector). The states must be compatible with the corresponding Alphabet object.
- **index**: An integer vector specifying a set of positions. It is set to 1:seq$length if omitted.
- **...**: Not used.
Value

The Sequence object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.

Examples

```r
# create a sequence object of length 10
s <- Sequence(alphabets = list(NucleotideAlphabet()), length = 10)
# set the states in some ranges
setStates(s, c("A", "T", "A"), index = 1:5)
setStates(s, c("G", "C"), index = 6:10)
# display sequence
s
# set the states for the whole Sequence object
setStates(s, c("A", "T", "T", "A"))
s
# set states via virtual field
s$states <- c("A", "T")
s
setString.Sequence

Forbidden action: setting the string representation of a Sequence object.

Description

Forbidden action: setting the string representation of a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setString(this, value, ...)
```

Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.
Forbidden action: setting the symbol length for an Alphabet object

For more information see `alphabet`.

## Description
Forbidden action: setting the symbol length for an Alphabet object.

## Usage
```r
## S3 method for class 'Alphabet'
setSymbolLength(this, value, ...)
```

## Arguments
- `this` An Alphabet object.
- `value` Not used.
- `...` Not used.

## Value
Throws an error.

## Author(s)
Botond Sipos, Gregory Jordan

## See Also
For more information see `alphabet`.
setSymbols.Alphabet  Specify a new symbol set for an Alphabet object

Description

Specify a new symbol set for an Alphabet object.

Usage

```r
## S3 method for class 'Alphabet'
setSymbols(this, set, ...)
```

Arguments

- `this`: An Alphabet object.
- `set`: The vector containing the new symbols set, automatically converted into a character vector.
- `...`: Not used.

Value

The new symbol set as a character vector.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `alphabet`.

Examples

```r
# create a new alphabet object
a <- Alphabet()
a
# specify a new symbol set
setSymbols(a, c(0, 1))
a
# the same, but now use the virtual field
a$symbols <- c("A", "T", "G", "C")
a
```
Forbidden action: setting the genetic code id

Use the `table.id` constructor argument to set the genetic code.

Usage

```r
## S3 method for class 'CodonAlphabet'
setTableId(this, value, ...)
```

Arguments

this  An object.
value Not used.
...   Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `CodonAlphabet`.

Set the template sequence object.

Usage

```r
## S3 method for class 'GeneralInsertor'
setTemplateSeq(this, value, ...)
```
Arguments

this          A GeneralInsertor object.
value         A Sequence object.
...           Not used.

Value

The Sequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see GeneralInsertor.

Examples

# create a GeneralInsertor object
i<-GeneralInsertor(
  rate=0.5,
  propose.by=function(process){sample(c(5:10),1)}), # inserts between 5 and 10
  template.seq=NucleotideSequence(string="AAAAAA")
)
# get template sequence
getTemplateSeq(i)
# get template sequence via virtual field
i$templateSeq
# set template sequence
setTemplateSeq(i, NucleotideSequence(string="C"));
# generate insert
generateInsert(i)
# set template sequence via virtual field
i$templateSeq<-NucleotideSequence(string="G")
# generate insert
generateInsert(i)

---

setTheta.T92  Set the GC content

Description

Set the GC content.
## setTipLabels.PhyloSim

### Usage

```r
## S3 method for class 'T92'
setTheta(this, value, ...)
```

### Arguments

- `this` : A T92 object.
- `value` : A numeric vector of length one.
- `...` : Not used.

### Value

The new value of theta (invisible).

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see `T92`.

### Examples

```r
# construct a T92 object
p <- T92()
# set/get GC content
setTheta(p, 0.6)
getTheta(p)
# set/get GC content via virtual field
p$theta <- 0.3
p$theta
# get object summary
summary(p)
```

---

### Description

Forbidden action: setting the tip labels for a phylo object aggregated by a PhyloSim object.

### Usage

```r
## S3 method for class 'PhyloSim'
setTipLabels(this, value, ...)
```
setTips.PhyloSim

Arguments

  this     An object.
  value    Not used.
  ...      Not used.

Value

  Throws an error.

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see PhyloSim.

Description

  Forbidden action: setting the node identifiers of the tip nodes for a PhyloSim object.

Usage

  # S3 method for class 'PhyloSim'
  setTips(this, value, ...)

Arguments

  this     An object.
  value    Not used.
  ...      Not used.

Value

  Throws an error.

Author(s)

  Botond Sipos, Gregory Jordan

See Also

  For more information see PhyloSim.
Description

Set the tolerance margin.

Usage

## S3 method for class 'FastFieldDeletor'
setToleranceMargin(this, value, ...)

Arguments

- **this**: A FastFieldDeletor object.
- **value**: A numeric vector of length one.
- **...**: Not used.

Value

value (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

Examples

```
# create a geometric FastFieldDeletor
p <- FastFieldDeletor()
# set/get tolerance margin
setToleranceMargin(p, 0.8)
gtoleranceMargin(p)
# set/get tolerance margin via virtual field
p$toleranceMargin <- 0.75
p$toleranceMargin
```
Description

Forbidden action: setting the total active event rate for a Site object.

Usage

```r
## S3 method for class 'Site'
setTotalRate(this, value, ...)
```

Arguments

- `this` An object.
- `value` Not used.
- `...` Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

---

Description

Forbidden action: setting the list of total site rates for a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setTotalRates(this, value, ...)
```
Forbidden action: setting the genetic code table for a CodonAlphabet object.

Arguments

this An object.

value Not used.

... Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see Sequence.
For more information see CodonAlphabet.

Forbidden action: setting the tree length for a PhyloSim object.

## S3 method for class 'PhyloSim'
setTreeLength(this, value, ...)  

### Arguments

- **this**: An object.
- **value**: Not used.
- **...**: Not used.

### Value

Throws an error.

### Author(s)

Botond Sipos, Gregory Jordan

### See Also

For more information see PhyloSim.
**setType.Alphabet**

*Set Alphabet object type*

**Description**

Set Alphabet object type.

**Usage**

```r
## S3 method for class 'Alphabet'
setType(this, new_type, ...)
```

**Arguments**

- `this` An Alphabet object.
- `new_type` A character vector of length one.
- `...` Not used.

**Value**

The new type (invisible).

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Alphabet`.

**Examples**

```r
# create an alphabet object
a <- Alphabet(symbols = c(1, 2, 3))
# set a new type
setType(a, "MyAlphabet")
a$type
# set type via virtual field
a$type <- "MorphChars"
```
### `setType.FastFieldDeletor`

**Forbidden action: setting the type of a FastFieldDeletor object**

#### Description
Forbidden action: setting the type of a FastFieldDeletor object.
The type can be set only through the type constructor argument.

#### Usage
```r
## S3 method for class 'FastFieldDeletor'
setType(this, value, ...)
```

#### Arguments
- `this`: An object.
- `value`: Not used.
- `...`: Not used.

#### Value
Throws an error.

#### Author(s)
Botond Sipos, Gregory Jordan

#### See Also
For more information see `BrownianInsertor`.

### `setType.BrownianInsertor`

**Forbidden action: setting the type of a BrownianInsertor object**

#### Description
Forbidden action: setting the type of a BrownianInsertor object.
The type can be set only from the type constructor argument and cannot be changed later.

#### Usage
```r
## S3 method for class 'BrownianInsertor'
setType(this, value, ...)
```

#### Arguments
- `this`: An object.
- `value`: Not used.
- `...`: Not used.

#### Value
Throws an error.

#### Author(s)
Botond Sipos, Gregory Jordan

#### See Also
For more information see `BrownianInsertor`. 
setUniqueAlphabets.Sequence

Arguments

this  An object.
value  Not used.
...  Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see FastFieldDeletor.

---

setUniqueAlphabets.Sequence

Forbidden action: setting the list of unique Alphabet objects attached to the Site object aggregated by a Sequence object

Description

Forbidden action: setting the list of unique Alphabet objects attached to the Site object aggregated by a Sequence object.

Usage

```r
## S3 method for class 'Sequence'
setUniqueAlphabets(this, value, ...)
```

Arguments

this  An object.
value  Not used.
...  Not used.

Value

Throws an error.

Author(s)

Botond Sipos, Gregory Jordan
See Also

For more information see Sequence.

---

**setUniqueProcesses.Sequence**

*Forbidden action: setting the list of unique Process instances attached to the sites of a Sequence object*

---

**Description**

Forbidden action: setting the list of unique Process instances attached to the sites of a Sequence object.

**Usage**

```r
## S3 method for class 'Sequence'
setUniqueProcesses(this, value, ...)
```

**Arguments**

- `this`  
  An object.
- `value`  
  Not used.
- `...`  
  Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see Sequence.
**setWriteProtected.Alphabet**

*Set the write protection field for an object*

**Description**

Set the write protection field for an object. Write protected objects cannot be modified through get/set methods and virtual fields.

**Usage**

```r
## S3 method for class 'Alphabet'
setWriteProtected(this, value, ...)
```

**Arguments**

- `this`: An object.
- `value`: A logical vector of size one.
- `...`: Not used.

**Value**

Invisible TRUE or FALSE.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Alphabet`.

**Examples**

```r
# create an object
o<-Alphabet()
# toggle write protection
setWriteProtected(o, TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected<-FALSE
o$writeProtected
```
**Description**

Set the write protection field for an object. Write protected objects cannot be modified through get/set methods and virtual fields.

**Usage**

```r
## S3 method for class 'Event'
setWriteProtected(this, value, ...)
```

**Arguments**

- `this` An object.
- `value` A logical vector of size one.
- `...` Not used.

**Value**

Invisible TRUE or FALSE.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# create an object
o <- Event()
# toggle write protection
setWriteProtected(o, TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected <- FALSE
o$writeProtected
```
setWriteProtected.Process

Set the write protection field for an object

Description
Set the write protection field for an object. Write protected objects cannot be modified through get/set methods and virtual fields.

Usage

```r
## S3 method for class 'Process'
setWriteProtected(this, value, ...)
```

Arguments

- `this`: A Process object.
- `value`: A logical vector of size one.
- `...`: Not used.

Value
Invisible TRUE or FALSE.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see `Process`.

Examples

```r
# create an object
o <- Process()
# toggle write protection
setWriteProtected(o, TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected <- FALSE
o$writeProtected
```
setWriteProtected.QMatrix

Set the write protection field for a QMatrix object

Description
Set the write protection field for a QMatrix object.
QMatrix object do not have a write protection flag of their own, but they use the one from the
associated Process object. Write protected objects cannot be modified through get/set methods and
virtual fields.

Usage
```r
## S3 method for class 'QMatrix'
setWriteProtected(this, value, ...)
```

Arguments
- **this**: A QMatrix object.
- **value**: A logical vector of size one.
- **...**: Not used.

Value
TRUE or FALSE

Author(s)
Botond Sipos, Gregory Jordan

See Also
setWriteProtected.Process

setWriteProtected.Sequence

Set the write protection field for an object

Description
Set the write protection field for an object. Write protected objects cannot be modified through
get/set methods and virtual fields.
Simulate.PhyloSim

Usage

```r
## S3 method for class 'Sequence'
setWriteProtected(this, value, ...)
```

Arguments

- `this`: An object.
- `value`: A logical vector of size one.
- `...`: Not used.

Value

Invisible TRUE or FALSE.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create an object
o <- Sequence()
# toggle write protection
setWriteProtected(o, TRUE)
# check write protection
o$writeProtected
# set write protection via virtual field
o$writeProtected <- FALSE
o$writeProtected
```
Usage

```r
## S3 method for class 'PhyloSim'
Simulate(this, quiet=FALSE, ...)
```

Arguments

- `this` A PhyloSim object.
- `quiet` TRUE or FALSE (default).
- `...` Not used.

Value

The PhyloSim object (invisible).

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`.

Examples

```r
# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim<-PhyloSim(
name="TinySim",
phylo=rcoal(3),
root.seq=NucleotideSequence(string="ATGC",processes=list(list(JC69()))))

# Run the simulation
Simulate(sim);
# Print the resulting sequences
sim$sequences
# Print the resulting alignment
sim$alignment
```
Description

This is the class representing a site. Site objects can have one associated Alphabet object and one or more Process objects that act on their states. The associated Process and Site objects must have associated Alphabet objects with the same symbols set, or at least one of the Alphabet objects should inherit from the class AnyAlphabet.

Site objects store the site-process-specific parameters of the attached Process objects. A site-process-specific parameter is a list containing: the identifier, the name, the value and type of the parameter. For example the ubiquitous rate multiplier site-process-specific parameter looks like

```
listHid="rate.multiplier",name="Rate multiplier",value=1,type="numeric"L
```

Templates for site-process-specific parameters and their default values are stored in the Process objects and copied into the Site object when the process is attached.

Site objects have fields for associated ancestral Site objects and Sequence objects.

Package:

Class Site

```
Object
|---PSRoot
|------Site

Directly known subclasses:

```
public static class Site
extends PSRoot
```

Usage

```
Site(state=NA, alphabet=NA, ancestral=NA, processes=NA, sequence=NA, ...)
```

Arguments

- **state**: A symbol belonging to the specified alphabet.
- **alphabet**: An alphabet object.
- **ancestral**: The ancestral Site object.
- **processes**: A list of Process objects.
- **sequence**: The Sequence object to which the Site object belongs.
- **...**: Not used.

Fields and Methods

Methods:
Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$. $<-, [, [[-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

See Also
Alphabet, Process, Event

Examples
# create a site object
site<-Site();
# print the character representation (state)
print(site);
# get a summary
summary(site);
# create a site with a nucleotide alphabet
site<-Site(state="A",alphabet=NucleotideAlphabet());
# set site state
site$state<-"G"
# manipulate the associated Sequence object
site$sequence<-Sequence()
# attach a substitution process
site$processes<-list(JC69())
# add one more substitution process
attachProcess(site,K80())
# get a list of active Event objects
site$events
# get a summary of the Site object
summary(site);

## Description

Returns the string length of the character representation of an object. More useful as a static method.

## Usage

```r
## Default S3 method:
stringLength(this, ...)
```

## Arguments

- `this` An object.
- `...` Not used.

## Value

An integer vector of length one.

## Author(s)

Botond Sipos, Gregory Jordan

## See Also

For more information see `PSRoot`. 
Examples

```r
x <- "character representation"
# get the string length of x
StringLength(x)
```

### `StringLengthVector`  
*Returns the string lengths of the character representations of a collection of objects*

#### Description

Returns the string lengths of the character representations of a collection of objects. More useful as a static method.

#### Usage

```r
## Default S3 method:
StringLengthVector(this, ...)
```

#### Arguments

- `this`: An array or a list of object.
- `...`: Not used.

#### Value

An integer vector with the corresponding lengths.

#### Author(s)

Botond Sipos, Gregory Jordan

#### See Also

For more information see `PSRoot`.

#### Examples

```r
x <- c("character representation", "other string");
# get the string length of x
StringLengthVector(x)
```
summary.Alphabet

summary.Alphabet  Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'Alphabet'
summary(object, ...)
```

Arguments

- `object`  An object
- `...`     Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Alphabet`.

Examples

```r
# create an object
a<-NucleotideAlphabet()
# get a summary
summary(a)
```
summary.AminoAcidSubst

*Summarize the properties of an object*

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'AminoAcidSubst'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `AminoAcidSubst`.

Examples

```r
# create an object
a<-AminoAcidSubst()
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Usage

```r
## S3 method for class 'BinarySubst'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `BinarySubst`.

Examples

```r
# create an object
p <- BinarySubst(rate = 0.25, name = "Binary", rate.list = list("0->1" = 2, "1->0" = 1))
# get a summary
summary(p)
```
summarize.BrownianInsertor

Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'BrownianInsertor'
summary(object, ...)
```

Arguments

- `object` An object
- `...` Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `BrownianInsertor`.

Examples

```r
# create an object
p <- BrownianInsertor(
  type = "discrete",
  scale = 0.05,
  sizes = 1:4,
  probs = c(3/6, 1/6, 1/6, 1/6),
  rate = 0.05
)

# get a summary
summary(p)
```
**Summary.CodonAlphabet**  
*Summarize the properties of an object*

---

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'CodonAlphabet'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonAlphabet`.

**Examples**

```r
# create an object
a <- CodonAlphabet()
# get a summary
summary(a)
```
summary.CodonSubst

---

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'CodonSubst'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `CodonSubst`.

**Examples**

```r
# create an object
a <- CodonSubst()
# get a summary
summary(a)
```
**summary.ContinuousDeleter**

*Summarize the properties of an object*

---

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'ContinuousDeleter'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ContinuousDeleter`.

**Examples**

```r
# create an object
da <- ContinuousDeleter(rate=1, dist=expression(rnorm(1, mean=5, sd=3)), max.length=10)
# get a summary
summary(da)
```
**summary.ContinuousInsertor**

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'ContinuousInsertor'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `ContinuousInsertor`.

**Examples**

```r
# create an object
a <- ContinuousInsertor(rate=1, dist=expression(rnorm(1, mean=5, sd=2)), max.length=7)
# get a summary
summary(a)
```
**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'DiscreteDeletor'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteDeletor`.

**Examples**

```r
# create an object
a <- DiscreteDeletor(rate=1, sizes=c(1,2), probs=c(1/2,1/2))
# get a summary
summary(a)
```
**summary.DiscreteInsertor**

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'DiscreteInsertor'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `DiscreteInsertor`.

**Examples**

```r
# create an object
a<-DiscreteInsertor(rate=1,sizes=c(1,2),probs=c(1/2,1/2))
# get a summary
summary(a)
```
**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'Event'
summary(object, ...)
```

**Arguments**

- **object**: An object
- **...**: Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `Event`.

**Examples**

```r
# create an object
e <- Event()
# get a summary
summary(e)
```
Summary

Summarize the properties of an object.

Usage

```r
## S3 method for class 'F81'
summary(object, ...)
```

Arguments

- `object` An object
- `...` Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F81`.

Examples

```r
# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
```
Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'F84'
summary(object, ...)
```

Arguments

- `object` An object
- `...` Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `F84`.

Examples

```r
# create an object
a <- F84(rate.params=list("Kappa"=3))
# get a summary
summary(a)
```
summarize.FastFieldDeletor

Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'FastFieldDeletor'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `FastFieldDeletor`.

Examples

```r
# create an object
a <- FastFieldDeletor()
# get a summary
summary(a)
```
Summary

GeneralDeletor

Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GeneralDeletor'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralDeletor`.

Examples

```r
# create an object
a <- GeneralDeletor(rate=1, name="Del Bosque")
# get a summary
summary(a)
```
Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GeneralInDel'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInDel`.

Examples

```r
# create an object
a <- GeneralInDel(rate = 1, propose.by = function(process)(sample(c(1, 2, 3), 1)))
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GeneralInsertor'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GeneralInsertor`.

Examples

```r
# create an object
a<-GeneralInsertor(rate=1)
# get a summary
summary(a)
```
summary.GeneraSubstitution

*Summarize the properties of an object*

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'GeneraSubstitution'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `GeneraSubstitution`.

**Examples**

```r
# create an object
a <- GeneralSubstitution(alphabet=BinaryAlphabet(), rate.list=list("0->1"=1,"1->0"=2))
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GTR'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GTR`.

Examples

```r
# create an object
a <- GTR()
# get a summary
summary(a)
```
Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'GY94'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `GY94`.

Examples

```r
# create an object
a <- GY94(kappa=2)
# get a summary
summary(a)
```
**Summary**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'HKY'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `HKY`.

**Examples**

```r
# create an object
a<-NucleotideAlphabet()
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Usage

```r
## S3 method for class 'JC69'
summary(object, ...)  
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `JC69`.

Examples

```r
# create an object
p <- JC69()
# get a summary
summary(p)
```
Arguments

object  An object
...  Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see \texttt{K80}.

Examples

\begin{verbatim}
# create an object
a<-NucleotideAlphabet()
# get a summary
summary(a)
\end{verbatim}

---

\begin{tabular}{ll}
\texttt{summary.K81} & \textit{Summarize the properties of an object} \\
\end{tabular}

Description

Summarize the properties of an object.

Usage

\begin{verbatim}
## S3 method for class 'K81'
summary(object, ...)
\end{verbatim}

Arguments

object  An object
...  Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan
See Also

For more information see `k81`.

Examples

```r
# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
```

summary.PhyloSim  
*Summarize the properties of an object*

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'PhyloSim'
summary(object, ...)
```

Arguments

- **object**: An object
- **...**: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PhyloSim`. 
Examples

# Create a PhyloSim object.
# Provide the phylo object
# and the root sequence.
sim <- PhyloSim(
  name = "TinySim",
  phylo = coal(3),
  root.seq = NucleotideSequence(string = "ATG", processes = list(list(JC69())))
);
  # get a summary
  summary(sim)

summary.Process  Summarize the properties of an object

Description
Summarize the properties of an object.

Usage

## S3 method for class 'Process'
summary(object, ...)

Arguments

object  An object
...
Not used.

Value
Returns a PSRootSummary object.

Author(s)
Botond Sipos, Gregory Jordan

See Also
For more information see Process.

Examples

# create an object
a <- NucleotideAlphabet()
# get a summary
summary(a)
Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'PSRoot'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `PSRoot`.

Examples

```r
# create an object
a<-PSRoot()
# get a summary
summary(a)
```
Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'QMatrix'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `QMatrix`.

Examples

```r
# create an object
a <- QMatrix(alphabet=BinaryAlphabet(), rate.list=list("0->1"=1,"1->0"=3))
# get a summary
summary(a)
```
Summary.Sequence

Summarize the properties of an object.

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'Sequence'
summary(object, ...)
```

Arguments

- **object**: An object
- **...**: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Sequence`.

Examples

```r
# create an object
a <- Sequence()
# get a summary
summary(a)
```
Summary

Summarize the properties of an object.

Description

Summarize the properties of an object.

Usage

```
## S3 method for class 'Site'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `Site`.

Examples

```
# create an object
o<-Site()
# get a summary
summary(o)
```
Summary:

Summarize the properties of an object.

Usage

## S3 method for class 'T92'
summary(object, ...)

Arguments

- **object**
  - An object

- **...**
  - Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see T92.

Examples

```r
# create an object
a <- T92(theta = 0.8)
# get a summary
summary(a)
```
Summarize the properties of an object.

**Description**

Summarize the properties of an object.

**Usage**

```r
## S3 method for class 'TN93'
summary(object, ...)
```

**Arguments**

- `object` An object
- `...` Not used.

**Value**

Returns a PSRootSummary object.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `TN93`.

**Examples**

```r
# create an object
a<-NucleotideAlphabet()
# get a summary
summary(a)
```
summary.ToleranceSubstitution

Summarize the properties of an object

Description

Summarize the properties of an object.

Usage

```r
## S3 method for class 'ToleranceSubstitution'
summary(object, ...)
```

Arguments

- `object`: An object
- `...`: Not used.

Value

Returns a PSRootSummary object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `ToleranceSubstitution`.

Examples

```r
# create an object
a <- ToleranceSubstitution(alphabet=BinaryAlphabet(),rate=list("0->1"=1,"1->0"=2))
# get a summary
summary(a)
```
**summary.UNREST**

Summarize the properties of an object

**Description**
Summarize the properties of an object.

**Usage**

```
## S3 method for class 'UNREST'
summary(object, ...)
```

**Arguments**

- `object`: An object
- `...`: Not used.

**Value**
Returns a PSRootSummary object.

**Author(s)**
Botond Sipos, Gregory Jordan

**See Also**
For more information see `UNREST`.

**Examples**

```r
# create an object
p <- UNREST(rate.list = list(
  "T->C" = 1, "T->A" = 2, "T->G" = 3, "C->T" = 4, "C->A" = 1,
  "C->G" = 2, "A->T" = 3, "A->C" = 4, "A->G" = 1, "G->T" = 2,
  "G->C" = 3, "G->A" = 4
))
# get a summary
summary(p)
```
The T92 class

Description

This class implements the T92 GTR-submodel.

The rate parameters are the following: "Alpha", "Beta","Gamma". The theta virtual field stores the GC content parameter.

Package:
Class T92

Usage

T92(name="Anonymous", rate.params=list(Alpha = 1, Beta = 1), theta=0.5, ...)

Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Object name.</td>
</tr>
<tr>
<td>rate.params</td>
<td>Rate parameters.</td>
</tr>
<tr>
<td>theta</td>
<td>GC content (0.5 by default).</td>
</tr>
<tr>
<td>...</td>
<td>Not used.</td>
</tr>
</tbody>
</table>
Fields and Methods

Methods:

checkConsistency -
getRateParam -
getRateParamList -
getTheta -
setRateParam -
setRateParamList -
setTheta -
summary -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam, setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationException, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan

References

See Also

GTR UNREST GeneralSubstitution HKY

Examples

```r
# create substitution process object
p<-T92(rate.params=list( "Alpha"=10,"Beta"=2),theta=0.8)
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
generateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rc0al(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**TN93**

*The TN93 class*

**Description**

This class implements the Tamura-Nei 93 GTR-submodel.

The rate parameters are the following: "Alpha1", "Alpha2","Beta". Package:

Class TN93

```
Object
~|  
```
Directly known subclasses:

public static class TN93 extends GTR

Usage

TN93(name="Anonymous", rate.params=list(Alpha1 = 1, Alpha2 = 1, Beta = 1),
    base.freqs=c(0.25, 0.25, 0.25, 0.25), ...)

Arguments

   name      Object name.
   rate.params Rate parameters.
   base.freqs Base frequency parameters.
   ...       Not used.

Fields and Methods

Methods:

   checkConsistency -
   getBaseFreqs -
   getRateParam -
   getRateParamList -
   setBaseFreqs -
   setRateParam -
   setRateParamList -
   summary -

Methods inherited from GTR:
checkConsistency, getBaseFreqs, getRateParam, getRateParamList, setBaseFreqs, setRateParam,
setRateParamList, summary

Methods inherited from UNREST:
checkConsistency, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAt-
Site, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$[, $<-, [., [.<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldName, getFieldModifier, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)
Botond Sipos, Gregory Jordan

References

See Also
GTR HKY UNREST GeneralSubstitution

Examples
# create substitution process object
p<-TN93(rate.params=list("Alpha1"=4,"Alpha2"=3,"Beta"=2),
base.freqs=c(2,2,1,3)/9)
 # get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
ToleranceSubstitution

# get rate multipliers
getRateMultipliers(s,p)

# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcol(2))

# run simulation
Simulate(sim)

# print alignment
sim$alignment

The ToleranceSubstitution class

Description

This a class representing a continuous-time Markov process acting on the state space defined by the symbols stored in the Alphabet object passed to the object constructor.

In contrast to GeneralSubstitution, the ToleranceSubstitution class has a site-specific substitution tolerance parameter ("substitution.tolerance") which determines the probability of accepting the proposed events. As a consequence, the branch lengths inferred from the simulated data will no longer correspond to the neutral expectations unless all tolerance values are equal to 1.

Package:

Class ToleranceSubstitution

Directly known subclasses:

public static class ToleranceSubstitution
extends Process

Usage

ToleranceSubstitution(name="Anonymous",alphabet=NA,rate.list=NA,eq.dist=NA,...)
Arguments

name
The name of the object.

alphabet
The alphabet on which the process acts (Alphabet object).

rate.list
A list with the substitution rates. It will be passed to setRateList method.

equ.dists
The equilibrium distribution (see setEquDist.ToleranceSubstitution).

Fields and Methods

Methods:

as.character -
checkConsistency -
clone -
getAlphabet -
getEquDist -
getEventRate -
getEventRateAtSite -
getEventsAtSite -
getQMatrix -
getRate -
getRateList -
hasUndefinedRate -
is -
plot -
rescaleQMatrix -
sampleState -
setAlphabet -
setEquDist -
setQMatrix -
setRate -
setRateList -
summary -

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, Il, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
ToleranceSubstitution

Author(s)
Botond Sipos, Gregory Jordan

See Also
Process QMatrix Event Site GeneralIndel GTR WAG

Examples

# construct a GTR process object, we will use this to fill in the rates
# for the ToleranceSubstitution process.
gtr <- GTR(
  name = "MyGTR",
  rate.params = list(
    "a" = 1, "b" = 2, "c" = 3,
    "d" = 1, "e" = 2, "f" = 3
  ),
  base.freqs = c(2, 2, 1, 1) / 6
)
rate.list.gtr <- gtr$rateList

# Construct the ToleranceSubstitution process.
p <- ToleranceSubstitution(
  name = "MyTolSubst",
  alphabet = NucleotideAlphabet(),
  rate.list = rate.list.gtr,
)
plot(p)

# construct root sequence object
s <- NucleotideSequence(length = 20)

# attach process
attachProcess(s, p)

# sample states from the equilibrium
# distribution of the attached processes
sampleStates(s)

## Set the substitution tolerance parameters for some sites:
setParameterAtSites(s, p, "substitution.tolerance", c(0, 0.05, 0.1), 1:3)

## Plot the substitution tolerance parameters across sites:
plotParametersAtSites(s, p, "substitution.tolerance")

# Construct simulation object:
Translate.CodonSequence

Translate a CodonSequence object

Description

Translate a CodonSequence object.

Usage

## S3 method for class 'CodonSequence'
Translate(this, ...)

Arguments

this A CodonSequence object

... Not used.

Value

The translation as an AminoAcidSequence object.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonSequence.

Examples

# create a CodonSequence object
s<-CodonSequence(string="ATCTTTTCAATGGGCCCCTCCGA")
# get the translation as an AminoAcidSequence object
as<-Translate(s)
as

sim <- PhyloSim(root.seq=s, phylo=tree(3))

# Run simulation:
Simulate(sim)

# Plot alignment:
plot(sim)
Description

Translate a codon.

Usage

## S3 method for class 'CodonAlphabet'
translateCodon(this, codon, ...)

Arguments

- this: A CodonAlphabet object.
- codon: The codon to be translated.
- ...: Not used.

Value

A character vector containing an amino acid IUPAC code.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see CodonAlphabet.

Examples

```r
# create a CodonAlphabet object
a<-CodonAlphabet()
# translate a codon
translateCodon(a,"AGG")
```
Description

Undocumented object (PhyloSim package).
See the corresponding specific methods if applicable.

Usage

```r
## S3 method for class 'PhyloSim'
Undocumented(...)
```

Arguments

... Not used.

Author(s)

Botond Sipos, Gregory Jordan

See Also

For more information see `phylosim`.

---

**UNREST**

The UNREST class

Description

This class implements the UNRESTricted nucleotide substitution model. UNREST objects are basically a GeneralSubstitution process acting on a nucleotide alphabet.

Package:
Class UNREST

```
Object
~~|--
~~++-PSRoot
~~~~~|--
~~~~~~~~---Process
~~~~~~~~~|--
~~~~~~~~~~~~---GeneralSubstitution
~~~~~~~~~~~~~~~~-|--
~~~~~~~~~~~~~~~~~~~++UNREST
```
**UNREST**

**Directly known subclasses:**
F81, F84, GTR, HKY, JC69, K80, K81, T92, TN93

public static class **UNREST**
extends GeneralSubstitution

**Usage**

UNREST(name="Anonymous", rate.list=NA, equ.dist=NA, ...)

**Arguments**

- **name**: Object name.
- **rate.list**: A list of unscaled rates (see setRateList GeneralSubstitution).
- **equ.dist**: Equilibrium distribution.
- ... Additional arguments.

**Fields and Methods**

**Methods:**

checkConsistency -
summary -

**Methods inherited from GeneralSubstitution:**
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

**Methods inherited from Process:**
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

**Methods inherited from PSRoot:**
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

**Methods inherited from Object:**
$, $<-, [[, [][<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save
Author(s)

Botond Sipos, Gregory Jordan

References


See Also

GeneralSubstitution GTR

Examples

```r
p<-UNREST(rate.list=list(
  "T->C"=1, "T->A"=2, "T->G"=3, "C->T"=4, "C->A"=1,
  "C->G"=2, "A->T"=3, "A->C"=4, "A->G"=1, "G->T"=2,
  "G->C"=3, "G->A"=4
))
# get a summary
summary(p)
# display a bubble plot
plot(p)

# The following code demonstrates how to use
# the process in a simulation.

# create a sequence, attach process p
s<-NucleotideSequence(length=20,processes=list(list(p)))
# sample states
sampleStates(s)
# make the first five positions invariable
setRateMultipliers(s,p,0,1:5)
# get rate multipliers
getRateMultipliers(s,p)
# create a simulation object
sim<-PhyloSim(root.seq=s,phylo=rcoral(2))
# run simulation
Simulate(sim)
# print alignment
sim$alignment
```

---

**virtualAssignmentForbidden.PSRoot**

*Throws an error message informing the user about forbidden action on virtual a field*
**Description**

Throws an error message informing the user about forbidden action on virtual a field.

**Usage**

```r
## S3 method for class 'PSRoot'
virtualAssignmentForbidden(this, ...)
```

**Arguments**

- `this`: A PSRoot object.
- `...`: Not used.

**Value**

Throws an error.

**Author(s)**

Botond Sipos, Gregory Jordan

**See Also**

For more information see `PSRoot`.

---

**WAG**

The WAG empirical amino acid substitution model

---

**Description**

Package:

**Class WAG**

```
Object
   `-- PSRoot
   `-- Process
   `-- GeneralSubstitution
   `-- AminoAcidSubst
   `-- WAG
```
Directly known subclasses:

public static class WAG
extends AminoAcidSubst

Usage

WAG(equ.dist=NA, ...)

Arguments

equ.dist Equilibrium distribution.
... Not used.

Fields and Methods

Methods:
No methods defined.

Methods inherited from AminoAcidSubst:
buildFromPAML, checkConsistency, newAAMatrix, setEquDist, summary

Methods inherited from GeneralSubstitution:
as.character, checkConsistency, clone, getAlphabet, getEquDist, getEventRate, getEventRateAtSite, getEventsAtSite, getQMatrix, getRate, getRateList, hasUndefinedRate, is, plot, rescaleQMatrix, sampleState, setAlphabet, setEquDist, setQMatrix, setRate, setRateList, summary

Methods inherited from Process:
!=, ==, as.character, checkConsistency, clone, getAlphabet, getEventsAtSite, getId, getName, getParameterAtSite, getSiteSpecificParamIds, getSiteSpecificParamList, getWriteProtected, hasSiteSpecificParameter, hasUndefinedRate, is, setAlphabet, setId, setName, setParameterAtSite, setSiteSpecificParamIds, setSiteSpecificParamList, setWriteProtected, summary

Methods inherited from PSRoot:
checkConsistency, enableVirtual, getComments, getMethodsList, globalConsistencyCheck, intersect.list, is, is.na, ll, my.all.equal, plot, setComments, setMethodsList, summary, virtualAssignmentForbidden

Methods inherited from Object:
$, $<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clearLookupCache, clone, detach, equals, extend, finalize, getEnvironment, getFieldModifier, getFieldModifiers, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, names, objectSize, print, save

Author(s)

Botond Sipos, Gregory Jordan
References


See Also

AminoAcidSubst GeneralSubstitution UNREST

Examples

```r
# create substitution model object
p <- WAG()
# get object summary
summary(p)
# display a bubble plot
plot(p)

  # The following code demonstrates how to use
  # the process in a simulation.

# create a sequence, attach process p
s <- AminoAcidSequence(length = 10, processes = list(list(p)))
# sample states
sampleStates(s)
  # make the first three positions invariable
setRateMultipliers(s, p, 0, 1:3)
  # get rate multipliers
getRateMultipliers(s, p)
  # create a simulation object
sim <- PhyloSim(root.seq = s, phylo = rcoal(2))
  # run simulation
Simulate(sim)
  # print alignment
sim$alignment
```
Index

! = . Alphabet (== . Alphabet), 12
== . Alphabet - method (== . Alphabet), 12
== . Alphabet, 12
== . Process, 13

all (Undocumented.PhyloSim), 586
Alphabet, 13, 14, 17, 22, 24, 37, 48, 78, 259,
266, 267, 278, 283, 296, 304, 321,
351, 372, 393, 514, 519, 520, 529,
533, 543
Alphabet.== (== . Alphabet), 12
Alphabet.as.character
(as.character.Alphabet), 24
Alphabet.checkConsistency
(checkConsistency.Alphabet), 48
Alphabet.getSize(getSize.Alphabet), 259
Alphabet.getSymbolLength
(getSymbolLength.Alphabet), 266
Alphabet.getSymbols
(getSymbols.Alphabet), 267
Alphabet.getType(getType.Alphabet), 278
Alphabet.getWriteProtected
(getWriteProtected.Alphabet),
283
Alphabet.hasSymbols
(hasSymbols.Alphabet), 295
Alphabet.isEmpty(isEmpty.Alphabet), 321
Alphabet.setSize(setSize.Alphabet), 514
Alphabet.setSymbolLength
(setSymbolLength.Alphabet), 519
Alphabet.setSymbols
(setSymbols.Alphabet), 520
Alphabet.setType(setType.Alphabet), 529
Alphabet.setWriteProtected
(setWriteProtected.Alphabet),
533
Alphabet.summary (summary.Alphabet), 543
AminoAcidAlphabet, 14, 16, 372, 393
AminoAcidSequence, 18, 372, 393, 410
Amino AcidSubst, 20, 21, 46, 49, 92, 132, 326,
328, 336, 337, 340–343, 345, 347,
350, 363, 365, 372, 389, 393, 544,
589, 590
Amino AcidSubst.buildFromPAML
(buildFromPAML.AminoAcidSubst),
46
Amino AcidSubst.checkConsistency
(checkConsistency.AminoAcidSubst),
49
Amino AcidSubst.newAAMatrix
(newAAMatrix.AminoAcidSubst),
350
Amino AcidSubst.summary
(summary.AminoAcidSubst), 544
AnyAlphabet, 14, 22, 372, 393
ape, 368, 398, 400
areSynonymous (Undocumented.PhyloSim), 586
areSynonymous, CodonAlphabet-method
(areSynonymous.CodonAlphabet),
23
areSynonymous, CodonAlphabet, 23
as.character, Alphabet-method
(as.character.Alphabet), 24
as.character, Event-method
(as.character.Event), 25
as.character, GeneralSubstitution-method
(as.character.GeneralSubstitution),
26
as.character, PhyloSim-method
(as.character.PhyloSim), 27
as.character, Process-method
(as.character.Process), 28
as.character, QMatrix-method
(as.character.QMatrix), 29
as.character, Sequence-method
(as.character.Sequence), 30
as.character,Site-method
  (as.character.Site), 31
as.character,ToleranceSubstitution-method
  (as.character.ToleranceSubstitution), 32
as.character.Alphabet, 24
as.character.Event, 25
as.character.GeneralSubstitution, 26
as.character.PhyloSim, 27
as.character.Process, 28
as.character.QMatrix, 29
as.character.Sequence, 30
as.character.Site, 31
as.character.ToleranceSubstitution, 32
attachHookToNode
  (Undocumented.PhyloSim), 586
attachHookToNode,PhyloSim-method
  (attachHookToNode.PhyloSim), 33
attachProcess (Undocumented.PhyloSim), 586
attachProcess,Sequence-method
  (attachProcess.Sequence), 34
attachProcess,Site-method
  (attachProcess.Site), 35
attachProcess.Sequence, 34
attachProcess.Site, 35
attachSeqToNode
  (Undocumented.PhyloSim), 586
attachSeqToNode,PhyloSim-method
  (attachSeqToNode.PhyloSim), 36
attachSeqToNode.PhyloSim, 36
BinaryAlphabet, 14, 37, 372, 393
BinarySequence, 38, 372, 393, 410
BinarySubst, 40, 50, 132, 372, 389, 393, 545
BinarySubst.checkConsistency
  (checkConsistency.BinarySubst), 49
BinarySubst.summary
  (summary.BinarySubst), 545
BrownianInsertor, 42, 45, 50, 103, 126, 129, 250, 279, 372, 389, 393, 508, 530, 546
BrownianInsertor.BrownianPath
  (BrownianPath.BrownianInsertor), 45
BrownianInsertor.checkConsistency
  (checkConsistency.BrownianInsertor), 50
BrownianInsertor.getScale
  (getScale.BrownianInsertor), 250
BrownianInsertor.getType
  (getType.BrownianInsertor), 279
BrownianInsertor.setScale
  (setScale.BrownianInsertor), 508
BrownianInsertor.setType
  (setType.BrownianInsertor), 530
BrownianInsertor.summary
  (summary.BrownianInsertor), 546
BrownianPath (Undocumented.PhyloSim), 586
BrownianPath.BrownianInsertor-method
  (BrownianPath.BrownianInsertor), 45
BrownianPath.BrownianInsertor
  (BrownianPath.BrownianInsertor), 45
buildFromPAML (Undocumented.PhyloSim), 586
buildFromPAML,AminoAcidSubst-method
  (buildFromPAML.AminoAcidSubst), 46
buildFromPAML,CodonSubst-method
  (buildFromPAML.CodonSubst), 47
buildFromPAML,AminoAcidSubst
  (buildFromPAML.AminoAcidSubst), 46
buildFromPAML.CodonSubst
  (buildFromPAML.CodonSubst), 47
checkConsistency (Undocumented.PhyloSim), 586
checkConsistency,Alphabet-method
  (checkConsistency.Alphabet), 48
checkConsistency,AminoAcidSubst-method
  (checkConsistency.AminoAcidSubst), 49
checkConsistency,BinarySubst-method
  (checkConsistency.BinarySubst), 49
checkConsistency,BrownianInsertor-method
  (checkConsistency.BrownianInsertor), 50
checkConsistency,CodonAlphabet-method
  (checkConsistency.CodonAlphabet), 51
checkConsistency,CodonSequence-method
  (checkConsistency.CodonSequence), 51
checkConsistency, CodonSubst-method (checkConsistency.CodonSubst), 52
checkConsistency, CodonUNREST-method (checkConsistency.CodonUNREST), 53
checkConsistency, ContinuousDeletor-method (checkConsistency.ContinuousDeletor), 53
checkConsistency, ContinuousInsertor-method (checkConsistency.ContinuousInsertor), 54
checkConsistency, DiscreteDeletor-method (checkConsistency.DiscreteDeletor), 55
checkConsistency, DiscreteInsertor-method (checkConsistency.DiscreteInsertor), 55
checkConsistency, Event-method (checkConsistency.Event), 56
checkConsistency, F81-method (checkConsistency.F81), 57
checkConsistency, F84-method (checkConsistency.F84), 57
checkConsistency, FastFieldDeletor-method (checkConsistency.FastFieldDeletor), 58
checkConsistency, GeneralDeletor-method (checkConsistency.GeneralDeletor), 59
checkConsistency, GeneralInDel-method (checkConsistency.GeneralInDel), 59
checkConsistency, GeneralInsertor-method (checkConsistency.GeneralInsertor), 60
checkConsistency, GeneralSubstitution-method (checkConsistency.GeneralSubstitution), 61
checkConsistency, GTR-method (checkConsistency.GTR), 61
checkConsistency, GY94-method (checkConsistency.GY94), 62
checkConsistency, HKY-method (checkConsistency.HKY), 63
checkConsistency, JC69-method (checkConsistency.JC69), 63
checkConsistency, K80-method (checkConsistency.K80), 64
checkConsistency, K81-method (checkConsistency.K81), 65
checkConsistency, PhyloSim-method (checkConsistency.PhyloSim), 65
checkConsistency, Process-method (checkConsistency.Process), 66
checkConsistency, PSRoot-method (checkConsistency.PSRoot), 67
checkConsistency, PSRootSummary-method (checkConsistency.PSRootSummary), 67
checkConsistency, QMatrix-method (checkConsistency.QMatrix), 68
checkConsistency, Sequence-method (checkConsistency.Sequence), 69
checkConsistency, Site-method (checkConsistency.Site), 69
checkConsistency, T92-method (checkConsistency.T92), 70
checkConsistency, TN93-method (checkConsistency.TN93), 71
checkConsistency, ToleranceSubstitution-method (checkConsistency.ToleranceSubstitution), 71
checkConsistency, UNREST-method (checkConsistency.UNREST), 72
checkConsistency, Alphabet, 48
checkConsistency, Amino Acid Subst, 49
checkConsistency, BinarySubst, 49
checkConsistency, BrownianInsertor, 50
checkConsistency, Codon Alphabet, 51
checkConsistency, Codon Sequence, 51
checkConsistency, Codon Subst, 52
checkConsistency, Codon UNREST, 53
checkConsistency, Continuous Deletor, 53
checkConsistency, Continuous Insertor, 54
checkConsistency, Discrete Deletor, 55
checkConsistency, Discrete Insertor, 55
checkConsistency, Event, 56
checkConsistency, F81, 57
checkConsistency, F84, 57
checkConsistency, FastField Deletor, 58
checkConsistency, General In Del, 59
checkConsistency, General Insertor, 60
checkConsistency, General Substitution,
checkConsistency.GTR, 61
checkConsistency.GY94, 62
checkConsistency.HKY, 63
checkConsistency.JC69, 63
checkConsistency.K80, 64
checkConsistency.K81, 65
checkConsistency.PhyloSim, 65
checkConsistency.Process, 66
checkConsistency.PSRoot, 67
checkConsistency.PSRootSummary, 67
checkConsistency.QMatrix, 68
checkConsistency.Sequence, 69
checkConsistency.Site, 69
checkConsistency.T92, 70
checkConsistency.TN93, 71
checkConsistency.ToleranceSubstitution, 71
checkConsistency.UNREST, 72
clearStates (Undocumented.PhyloSim), 586
clearStates.Sequence-method
(clearStates.Sequence), 73
clearStates.Sequence, 73
code, GeneralSubstitution-method
(code, GeneralSubstitution), 74
code, Process-method (code, Process), 75
code, Sequence-method (code, Sequence), 76
code, ToleranceSubstitution-method
(code, ToleranceSubstitution), 77
code, GeneralSubstitution, 74
code, Process, 75
code, Sequence, 76
code, ToleranceSubstitution, 77
CodonAlphabet, 14, 24, 51, 78, 79, 268, 277,
322, 323, 372, 393, 521, 528, 547, 585
CodonAlphabet.areSynonymous
.areSynonymous.CodonAlphabet), 23
CodonAlphabet.checkConsistency
(checkConsistency.CodonAlphabet), 51
CodonAlphabet.getTableId
(getTableId.CodonAlphabet), 268
CodonAlphabet.getTransTable
(getTransTable.CodonAlphabet), 276
CodonAlphabet.isStartCodon
.isStartCodon.CodonAlphabet), 322
CodonAlphabet.isStopCodon
.isStopCodon.CodonAlphabet), 323
CodonAlphabet.setTableId
(setTableId.CodonAlphabet), 521
CodonAlphabet.setTransTable
(setTransTable.CodonAlphabet), 527
CodonAlphabet.summary
(summary.CodonAlphabet), 547
CodonAlphabet.translateCodon
(translateCodon.CodonAlphabet), 585
CodonSequence, 52, 80, 81, 209, 355, 356,
358, 359, 361, 362, 372, 393, 410,
466, 584
CodonSequence.checkConsistency
(checkConsistency.CodonSequence), 51
CodonSequence.getOmegas
(getOmegas.CodonSequence), 209
CodonSequence.omegaHist
(omegaHist.CodonSequence), 354
CodonSequence.omegaVarM0
(omegaVarM0.CodonSequence), 355
CodonSequence.omegaVarM1
(omegaVarM1.CodonSequence), 357
CodonSequence.omegaVarM2
(omegaVarM2.CodonSequence), 358
CodonSequence.omegaVarM3
(omegaVarM3.CodonSequence), 360
CodonSequence.omegaVarM4
(omegaVarM4.CodonSequence), 361
CodonSequence.setOmegas
(setOmegas.CodonSequence), 465
CodonSequence.Translate
(Translate.CodonSequence), 584
CodonSubst, 47, 52, 82, 83, 105, 107, 132,
351, 389, 393, 548
CodonSubst.buildFromPAML
(buildFromPAML.CodonSubst), 47
CodonSubst.checkConsistency
(checkConsistency.CodonSubst), 52
CodonSubst.newMatrix
   (newMatrix.CodonSubst), 350
CodonSubst.summary
   (summary.CodonSubst), 548
CodonUNREST, 53, 84, 132, 292, 372, 389, 393
CodonUNREST.checkConsistency
   (checkConsistency.CodonUNREST), 53
ContinuousDeleter, 54, 86, 119, 124, 126, 162, 198, 372, 375, 389, 393, 437, 456, 549
ContinuousDeleter.checkConsistency
   (checkConsistency.ContinuousDeleter), 53
ContinuousDeleter.getDist
   (getDist.ContinuousDeleter), 162
ContinuousDeleter.getMaxLength
   (getMaxLength.ContinuousDeleter), 198
ContinuousDeleter.plot
   (plot.ContinuousDeleter), 374
ContinuousDeleter.setDist
   (setDist.ContinuousDeleter), 436
ContinuousDeleter.setMaxLength
   (setMaxLength.ContinuousDeleter), 456
ContinuousDeleter.summary
   (summary.ContinuousDeleter), 549
ContinuousInserter, 54, 88, 126, 129, 163, 199, 372, 376, 389, 393, 438, 457, 550
ContinuousInserter.checkConsistency
   (checkConsistency.ContinuousInserter), 54
ContinuousInserter.getDist
   (getDist.ContinuousInserter), 163
ContinuousInserter.getMaxLength
   (getMaxLength.ContinuousInserter), 199
ContinuousInserter.plot
   (plot.ContinuousInserter), 375
ContinuousInserter.setDist
   (setDist.ContinuousInserter), 437
ContinuousInsertor.setMaxLength
   (setMaxLength.ContinuousInsertor), 457
ContinuousInsertor.summary
   (summary.ContinuousInsertor), 550
copySubSequence
   (Undocumented.PhyloSim), 586
copySubSequence.Sequence-method
   (copySubSequence.Sequence), 91
copySubSequence.Sequence, 91
cpREV, 20, 92, 132, 372, 389, 393
Debug (Undocumented.PhyloSim), 586
Debug, PhyloSim-method (Debug.PhyloSim), 94
Debug, PhyloSim, 94
default (Undocumented.PhyloSim), 586
deleteSubSequence
   (Undocumented.PhyloSim), 586
deleteSubSequence.Sequence-method
   (deleteSubSequence.Sequence), 95
deleteSubSequence.Sequence, 95
detachHookFromNode
   (Undocumented.PhyloSim), 586
detachHookFromNode.PhyloSim-method
   (detachHookFromNode.PhyloSim), 96
detachHookFromNode.PhyloSim, 96
detachProcess (Undocumented.PhyloSim), 586
detachProcess.Sequence-method
   (detachProcess.Sequence), 97
detachProcess.Sequence-method
   (detachProcess.Site), 98
detachProcess.Sequence, 97
detachProcess.Site, 98
detachSeqFromNode
   (Undocumented.PhyloSim), 586
detachSeqFromNode.PhyloSim-method
   (detachSeqFromNode.PhyloSim), 99
detachSeqFromNode.PhyloSim, 99
DiscreteDeleter, 55, 100, 119, 124, 126, 215, 260, 372, 377, 389, 393, 471, 515, 551
DiscreteDeleter.checkConsistency
   (checkConsistency.DiscreteDeleter),
INDEX

fXTNgetrateparamlist (getrateparamlistNfXT), 241
fXTNsetbasefreqs (setbasefreqsNfXT), 426
fXTNsetkappa (setkappaNfXT), 449
fXTNsetrateparam (setrateparamNfXT), 491
fXTNsetrateparamlist (setrateparamlistNfXT), 499
fXTNsummary (summaryNfXT), 555
FastFieldDeleter, 58, 119, 122, 124, 126, 193, 194, 273, 280, 372, 389, 393, 452, 453, 525, 531, 556
FastFieldDeleter.checkConsistency (checkConsistencyNFastFieldDeleter), 58
FastFieldDeleter.getEventsAtSite (getEventsAtSiteNFastFieldDeleter), 175
FastFieldDeleter.getLengthParam1 (getLengthParam1NFastFieldDeleter), 193
FastFieldDeleter.getLengthParam2 (getLengthParam2NFastFieldDeleter), 194
FastFieldDeleter.getToleranceMargin (getToleranceMarginNFastFieldDeleter), 273
FastFieldDeleter.getType (getTypeNFastFieldDeleter), 280
FastFieldDeleter.setLengthParam1 (setLengthParam1NFastFieldDeleter), 451
FastFieldDeleter.setLengthParam2 (setLengthParam2NFastFieldDeleter), 452
FastFieldDeleter.setToleranceMargin (setToleranceMarginNFastFieldDeleter), 525
FastFieldDeleter.setType (setTypeNFastFieldDeleter), 530
FastFieldDeleter.summary (summaryNFastFieldDeleter), 556
flagTotalRate, Site-method (flagTotalRateSite), 122
flagTotalRate, Site (flagTotalRateSite), 122
GeneralDeleter, 59, 86, 100, 120, 123, 126, 176, 372, 389, 393, 557
GeneralDeleter.checkConsistency (checkConsistencyNGeneralDeleter), 59
GeneralDeleter.getEventsAtSite (getEventsAtSiteNGeneralDeleter), 176
GeneralDeleter.summary (summaryNGeneralDeleter), 557
GeneralInDel, 42, 60, 86, 88, 100, 102, 120, 124, 126, 129, 137, 221, 225, 297, 372, 389, 393, 415, 477, 481, 558
GeneralInDel.checkConsistency (checkConsistencyNGeneralInDel), 59
GeneralInDel.getAcceptBy (getAcceptByNGeneralInDel), 136
GeneralInDel.getProposeBy (getProposeByNGeneralInDel), 221
GeneralInDel.getRate (getRateNGeneralInDel), 225
GeneralInDel.hasUndefinedRate (hasUndefinedRateNGeneralInDel), 296
GeneralInDel.proposeLength (proposeLengthNGeneralInDel), 392
GeneralInDel.setAcceptBy (setAcceptByNGeneralInDel), 414
GeneralInDel.setProposeBy (setProposeByNGeneralInDel), 476
GeneralInDel.setRate (setRateNGeneralInDel), 481
GeneralInDel.summary (summaryNGeneralInDel), 558
GeneralInsertor.checkConsistency (checkConsistencyNGeneralInsertor), 60
GeneralInsertor.generateInsert (generateInsertNGeneralInsertor), 135
GeneralInsertor.getAcceptWin (getAcceptWinNGeneralInsertor), 559
GeneralInsertor.getEventsAtSite
  (getEventsAtSite.GeneralInsertor), 137
GeneralInsertor.getGenerateBy
  (getGenerateBy.GeneralInsertor), 177
GeneralInsertor.getInsertHook
  (getInsertHook.GeneralInsertor), 182
GeneralInsertor.getTemplateSeq
  (getTemplateSeq.GeneralInsertor), 188
GeneralInsertor.setAcceptWin
  (setAcceptWin.GeneralInsertor), 269
GeneralInsertor.setGenerateBy
  (setGenerateBy.GeneralInsertor), 416
GeneralInsertor.setInsertHook
  (setInsertHook.GeneralInsertor), 447
GeneralInsertor.setTemplateSeq
  (setTemplateSeq.GeneralInsertor), 521
GeneralInsertor.summary
  (summary.GeneralInsertor), 559
GeneralSubstitution, 20, 26, 40, 61, 74, 82,
  84, 92, 105, 107, 115, 117, 132, 141,
  166, 168, 171, 179, 222, 226, 229,
  289, 292, 297, 300, 310, 324, 326,
  328, 330, 333, 336, 340, 342, 345,
  347, 363, 365, 372, 378, 389, 393,
  401, 404, 418, 439, 478, 482, 487,
  560, 576, 578, 586, 587, 589
GeneralSubstitution.as.character
  (as.character.GeneralSubstitution), 26
GeneralSubstitution.checkConsistency
  (checkConsistency.GeneralSubstitution), 61
GeneralSubstitution.clone
  (clone.GeneralSubstitution), 74
GeneralSubstitution.getAlphabet
  (getAlphabet.GeneralSubstitution), 140
GeneralSubstitution.getEquDist
  (getEquDist.GeneralSubstitution), 166
GeneralSubstitution.getEventRate
  (getEventRate.GeneralSubstitution), 168
GeneralSubstitution.getEventRateAtSite
  (getEventRateAtSite.GeneralSubstitution), 171
GeneralSubstitution.getEventsAtSite
  (getEventsAtSite.GeneralSubstitution), 178
GeneralSubstitution.getQMatrix
  (getQMatrix.GeneralSubstitution), 222
GeneralSubstitution.getRate
  (getRate.GeneralSubstitution), 226
GeneralSubstitution.getRateList
  (getRateList.GeneralSubstitution), 229
GeneralSubstitution.hasUndefinedRate
  (hasUndefinedRate.GeneralSubstitution), 297
GeneralSubstitution.plot
  (plot.GeneralSubstitution), 378
GeneralSubstitution.rescaleQMatrix
  (rescaleQMatrix.GeneralSubstitution), 401
GeneralSubstitution.sampleState
  (sampleState.GeneralSubstitution), 404
GeneralSubstitution.setAlphabet
  (setAlphabet.GeneralSubstitution), 417
GeneralSubstitution.setEquDist
  (setEquDist.GeneralSubstitution), 439
GeneralSubstitution.setQMatrix
  (setQMatrix.GeneralSubstitution), 477
GeneralSubstitution.setRate
  (setRate.GeneralSubstitution), 482
GeneralSubstitution.setRateList
  (setRateList.GeneralSubstitution), 486
GeneralSubstitution.summary
  (summary.GeneralSubstitution), 560
generateInsert (Undocumented.PhyloSim), 586
generateInsert, GeneralInsertor-method
   (generateInsert.GeneralInsertor),
   135
generateInsert. GeneralInsertor, 135
getAcceptBy (Undocumented.PhyloSim), 586
getAcceptBy, GeneralIndel-method
   (getAcceptBy.GeneralIndel), 136
getAcceptBy. GeneralIndel, 136
getAcceptBy. GeneralInsertor
   (getAcceptBy.GeneralIndel), 136
getAcceptWin (Undocumented.PhyloSim), 586
getAcceptWin, GeneralInsertor-method
   (getAcceptWin.GeneralInsertor),
   137
generateInsert. GeneralInsertor, 137
getAlignment (Undocumented.PhyloSim), 586
getAlignment, PhyloSim-method
   (getAlignment.PhyloSim), 138
getAlignment. PhyloSim, 138
getAlignmentLength
   (Undocumented.PhyloSim), 586
getAlignmentLength, PhyloSim-method
   (getAlignmentLength.PhyloSim),
   139
generateInsert. GeneralInsertor, 139
getAlignmentLength. PhyloSim, 139
getAlphabet (Undocumented.PhyloSim), 586
getAlphabet, GeneralSubstitution-method
   (getAlphabet.GeneralSubstitution),
   140
getAlphabet, Process-method
   (getAlphabet.Process), 141
getAlphabet, QMatrix-method
   (getAlphabet.QMatrix), 142
getAlphabet, Site-method
   (getAlphabet.Site), 143
getAlphabet, ToleranceSubstitution-method
   (getAlphabet.ToleranceSubstitution),
   144
getAlphabet. GeneralSubstitution, 140
getAlphabet. Process, 141
getAlphabet. QMatrix, 142
getAlphabet. Site, 143
getAlphabet. ToleranceSubstitution, 144
getAlphabets (Undocumented.PhyloSim),
getCodonFreqs.GY94, 157
goingComments (Undocumented.PhylSim), 586
getComments, PSRoot-method
  (getComments.PSRoot), 158
getComments.PSRoot, 158
getCumulativeRates
  (Undocumented.PhylSim), 586
getCumulativeRates, Sequence-method
  (getCumulativeRates.Sequence), 159
getCumulativeRates.Sequence, 159
getCumulativeRatesFromRange
  (Undocumented.PhylSim), 586
getCumulativeRatesFromRange, Sequence-method
  (getCumulativeRatesFromRange.Sequence), 160
getCumulativeRatesFromRange.Sequence, 160
defDeletionTolerance
  (Undocumented.PhylSim), 586
defDeletionTolerance, Sequence-method
  (defDeletionTolerance.Sequence), 161
defDeletionTolerance.Sequence, 161
defDist (Undocumented.PhylSim), 586
defDist, ContinuousDeleter-method
  (defDist.ContinuousDeleter), 162
defDist, ContinuousInsertor-method
  (defDist.ContinuousInsertor), 163
defDist.ContinuousDeleter, 162
defDist.ContinuousInsertor, 163
defEdge (Undocumented.PhylSim), 586
defEdge, PhylSim-method
  (defEdge.PhylSim), 164
defEdge.PhylSim, 164
defEdges (Undocumented.PhylSim), 586
defEdges, PhylSim-method
  (defEdges.PhylSim), 165
defEdges.PhylSim, 165
defEquDist (Undocumented.PhylSim), 586
defEquDist, GeneralSubstitution-method
  (defEquDist.GeneralSubstitution), 166
defEquDist, ToleranceSubstitution-method
  (defEquDist.ToleranceSubstitution), 167
defEventRate (Undocumented.PhylSim), 586
defEventRate, GeneralSubstitution-method
  (defEventRate.GeneralSubstitution), 168
defEventRate, QMatrix-method
  (defEventRate.QMatrix), 169
defEventRate, ToleranceSubstitution-method
  (defEventRate.ToleranceSubstitution), 170
defEventRate, GeneralSubstitution, 168
defEventRate.QMatrix, 169
defEventRate.ToleranceSubstitution, 170
defEventRateAtSite
  (Undocumented.PhylSim), 586
defEventRateAtSite, GeneralSubstitution-method
  (defEventRateAtSite.GeneralSubstitution), 171
defEventRateAtSite, ToleranceSubstitution-method
  (defEventRateAtSite.ToleranceSubstitution), 171
defEventRateAtSite, GeneralSubstitution, 171
defEventRateAtSite, ToleranceSubstitution, 172
defEventRateAtSite, GeneralSubstitution, 171
defEventRateAtSite.ToleranceSubstitution, 172
defEvents (Undocumented.PhylSim), 586
defEvents, Sequence-method
  (defEvents.Sequence), 173
defEvents.Sequence, 173
defEvents.Site, 174
defEventsAtSite
  (Undocumented.PhylSim), 586
defEventsAtSite, FastFieldDeleter-method
  (defEventsAtSite.FastFieldDeleter), 175
defEventsAtSite, GeneralDeleter-method
  (defEventsAtSite.GeneralDeleter), 175
defEventsAtSite, GeneralDeleter, 176
defEventsAtSite, GeneralInsertor-method
  (defEventsAtSite GeneralInsertor), 177
defEventsAtSite, GeneralSubstitution-method
  (defEventsAtSite, GeneralSubstitution), 177
<table>
<thead>
<tr>
<th>Line</th>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>178</td>
<td><code>getEventsAtSite, GY94-method</code></td>
<td>179</td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.GY94)</code></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td><code>getEventsAtSite, Process-method</code></td>
<td>180</td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.Process)</code></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td><code>getEventsAtSite, ToleranceSubstitution-method</code></td>
<td>181</td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.ToleranceSubstitution)</code></td>
<td></td>
</tr>
<tr>
<td>181</td>
<td><code>getEventsAtSite, FastFieldDeletor, 175</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.FastFieldDeletor)</code></td>
<td></td>
</tr>
<tr>
<td>175</td>
<td><code>getEventsAtSite, GeneralDeletor, 176</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.Genera</code></td>
<td></td>
</tr>
<tr>
<td>176</td>
<td><code>getEventsAtSite, GeneralInsertor, 177</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.Genera</code></td>
<td></td>
</tr>
<tr>
<td>177</td>
<td><code>getEventsAtSite, GeneralSubstitution, 178</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.Genera</code></td>
<td></td>
</tr>
<tr>
<td>178</td>
<td><code>getEventsAtSite, GY94, 179</code></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td><code>getEventsAtSite.GY94)</code></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td><code>getEventsAtSite, Process, 180</code></td>
<td></td>
</tr>
<tr>
<td>181</td>
<td><code>getEventsAtSite, ToleranceSubstitution, 181</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getEventsAtSite.Genera</code></td>
<td></td>
</tr>
<tr>
<td>182</td>
<td><code>getGenerateBy, (Undocumented.PhyloSim), 586</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getGenerateBy, GeneralInsertor-method</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getGenerateBy, GeneralInsertor)</code></td>
<td></td>
</tr>
<tr>
<td>182</td>
<td><code>getGenerateBy, GeneralInsertor, 182</code></td>
<td></td>
</tr>
<tr>
<td>175</td>
<td><code>getGenerateBy, GeneralDeletor, 175</code></td>
<td></td>
</tr>
<tr>
<td>183</td>
<td><code>getGenerateBy, GeneralDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>183</td>
<td><code>getGenerateBy, GeneralDeletor, Event)</code></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td><code>getId, (Undocumented.PhyloSim), 586</code></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td><code>getId, PhyloSim-method</code></td>
<td></td>
</tr>
<tr>
<td>185</td>
<td><code>getId, Process-method</code></td>
<td></td>
</tr>
<tr>
<td>186</td>
<td><code>getId, QMatrix-method</code></td>
<td></td>
</tr>
<tr>
<td>187</td>
<td><code>getId, Sequence-method</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getId, PhyloSim, 184</code></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td><code>getId, Process, 185</code></td>
<td></td>
</tr>
<tr>
<td>185</td>
<td><code>getId, QMatrix, 186</code></td>
<td></td>
</tr>
<tr>
<td>186</td>
<td><code>getId, Sequence, 187</code></td>
<td></td>
</tr>
<tr>
<td>187</td>
<td><code>getInsertHook, (Undocumented.PhyloSim), 586</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>getInsertHook, GeneralInsertor-method</code></td>
<td></td>
</tr>
<tr>
<td>187</td>
<td><code>getInsertHook, GeneralInsertor, 188</code></td>
<td></td>
</tr>
<tr>
<td>188</td>
<td><code>getInsertHook, GeneralInsertor</code></td>
<td></td>
</tr>
<tr>
<td>188</td>
<td><code>getInsertionTolerance, (Undocumented.PhyloSim), 586</code></td>
<td></td>
</tr>
<tr>
<td>188</td>
<td><code>getInsertionTolerance, Sequence-method</code></td>
<td></td>
</tr>
<tr>
<td>189</td>
<td><code>getInsertionTolerance, Sequence)</code></td>
<td></td>
</tr>
<tr>
<td>190</td>
<td><code>getKappa, F84-method</code>(getKappa.F84)`</td>
<td></td>
</tr>
<tr>
<td>191</td>
<td><code>getKappa, GY94-method</code>(getKappa.GY94)`</td>
<td></td>
</tr>
<tr>
<td>192</td>
<td><code>getKappa, F84, 190</code></td>
<td></td>
</tr>
<tr>
<td>193</td>
<td><code>getKappa, GY94, 191</code></td>
<td></td>
</tr>
<tr>
<td>194</td>
<td><code>getLength, (Undocumented.PhyloSim), 586</code></td>
<td></td>
</tr>
<tr>
<td>195</td>
<td><code>getLength, Sequence-method</code></td>
<td></td>
</tr>
<tr>
<td>196</td>
<td><code>getLength, Sequence, 192</code></td>
<td></td>
</tr>
<tr>
<td>197</td>
<td><code>getLengthParam1, (Undocumented.PhyloSim), 586</code></td>
<td></td>
</tr>
<tr>
<td>198</td>
<td><code>getLengthParam1, FastFieldDeletor-method</code></td>
<td></td>
</tr>
<tr>
<td>199</td>
<td><code>getLengthParam2, FastFieldDeletor-method</code></td>
<td></td>
</tr>
<tr>
<td>194</td>
<td><code>getLengthParam2, FastFieldDeletor, 194</code></td>
<td></td>
</tr>
<tr>
<td>195</td>
<td><code>getLengthParam2, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>195</td>
<td><code>getLengthParam2, FastFieldDeletor, Event)</code></td>
<td></td>
</tr>
<tr>
<td>196</td>
<td><code>getLengthParam1, FastFieldDeletor, 193</code></td>
<td></td>
</tr>
<tr>
<td>197</td>
<td><code>getLengthParam1, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>197</td>
<td><code>getLengthParam2, FastFieldDeletor, Event</code>)</td>
<td></td>
</tr>
<tr>
<td>197</td>
<td><code>getLengthParam2, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>198</td>
<td><code>getLengthParam2, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>199</td>
<td><code>getLengthParam2, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>198</td>
<td><code>getLengthParam2, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>199</td>
<td><code>getLengthParam2, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
<tr>
<td>199</td>
<td><code>getLengthParam2, FastFieldDeletor, Event-method</code></td>
<td></td>
</tr>
</tbody>
</table>
getMethodsList (Undocumented.PhylomSim), 586
getMethodsList (PSRoot-method
  (getMethodsList (PSRoot)), 200
getMethodsList (PSRoot), 200
getName (Event-method (getName (Event)),
  201
getName (PhylomSim-method
  (getName (PhylomSim)), 202
getName (Process-method
  (getName (Process)), 203
getName (QMatrix-method
  (getName (QMatrix)), 204
getName (Sequence-method
  (getName (Sequence)), 205
getName (Event), 201
getName (PhylomSim), 202
getName (Process), 203
generate (QMatrix), 204
generate (Sequence), 205
generate (Undocumented.PhylomSim), 586
generate (PhylomSim-method
  (generate (PhylomSim)), 206
generate (PhylomSim), 206
generate (Undocumented.PhylomSim), 586
generate (PhylomSim-method
  (generate (PhylomSim)), 207
generate (PhylomSim), 207
generate (Undocumented.PhylomSim), 586
generate (PhylomSim-method
  (generate (PhylomSim)), 208
generate (PhylomSim), 208
generate (Undocumented.PhylomSim), 586
generate (PhylomSim-method
  (generate (PhylomSim)), 209
generate (PhylomSim), 209
generate (Undocumented.PhylomSim), 586
generate (PhylomSim-method
  (generate (PhylomSim)), 210
generate (PhylomSim), 210
getProbs (Undocumented.PhylomSim), 586
generate (Probs-method
  (generate (Probs)), 211
generate (Probs), 211
generate (DiscreteDeleter-method
  (generate (Probs)), 215
generate (DiscreteDeleter), 215
generate (DiscreteInsertor-method
  (generate (Probs)), 216
generate (DiscreteInsertor), 216
generate (Process-method
  (generate (Process)), 217
generate (Process), 217
generate (QMatrix-method
  (generate (Probs)), 218
generate (QMatrix), 218
generate (Undocumented.PhylomSim), 586
generate (Processes-method
  (generate (Processes)), 219
generate (Processes), 219
generate (Sequence-method
  (generate (Processes)), 220
generate (Processes), 220
generate (Site-method
  (generate (Processes)), 221
generate (Processes), 221
generate (Site), 221
generate (Undocumented.PhylomSim), 586
generate (Proposes-method
  (generate (Proposes)), 222
generate (Proposes), 222
getQMatrix, GeneralSubstitution-method (getQMatrix.GeneralSubstitution), 222
getQMatrix, ToleranceSubstitution-method (getQMatrix.ToleranceSubstitution), 223
getQMatrix, GeneralSubstitution, 222
getQMatrix, ToleranceSubstitution, 223
generate (Undocumented.PhylSim), 586
generate, Event-method (generate.Event), 224
generate, GeneralIndel-method (generate.GeneralIndel), 225
generate, GeneralSubstitution-method (generate.GeneralSubstitution), 226
generate, QMatrix-method (generate.QMatrix), 227
generate, ToleranceSubstitution-method (generate.ToleranceSubstitution), 228
generate.Event, 224
generate, GeneralIndel, 225
generate, GeneralSubstitution, 226
generate, QMatrix, 227
generate, ToleranceSubstitution, 228
generateList (Undocumented.PhylSim), 586
generateList, GeneralSubstitution-method (generateList.GeneralSubstitution), 229
generateList, QMatrix-method (generateList.QMatrix), 230
generateList, ToleranceSubstitution-method (generateList.ToleranceSubstitution), 231
generateList, GeneralSubstitution, 229
generateList, QMatrix, 230
generateList, ToleranceSubstitution, 231
generateMultipliers (Undocumented.PhylSim), 586
generateMultipliers, Sequence-method (generateMultipliers.Sequence), 232
generateMultipliers, Sequence, 232
generateParam (Undocumented.PhylSim), 586
generateParam, F81-method (generateParam.F81), 233
generateParam, F84-method (generateParam.F84), 233
generateParam, GTR-method (generateParam.GTR), 234
generateParam, HKY-method (generateParam.HKY), 235
generateParam, K80-method (generateParam.K80), 236
generateParam, K81-method (generateParam.K81), 237
generateParam, T92-method (generateParam.T92), 238
generateParam, TN93-method (generateParam.TN93), 239
generateParam, F81, 233
generateParam, F84, 233
generateParam, GTR, 234
generateParam, HKY, 235
generateParam, K80, 236
generateParam, K81, 237
generateParam, T92, 238
generateParam, TN93, 239
generateParamList (Undocumented.PhylSim), 586
generateParamList, F81-method (generateParamList.F81), 240
generateParamList, F84-method (generateParamList.F84), 241
generateParamList, GTR-method (generateParamList.GTR), 242
generateParamList, HKY-method (generateParamList.HKY), 243
generateParamList, K80-method (generateParamList.K80), 244
generateParamList, K81-method (generateParamList.K81), 245
generateParamList, T92-method (generateParamList.T92), 246
generateParamList, TN93-method (generateParamList.TN93), 247
generateParamList, F81, 240
generateParamList, F84, 241
generateParamList, GTR, 242
generateParamList, HKY, 243
generateParamList, K80, 244
generateParamList, K81, 245
generateParamList, T92, 246
generateParamList, TN93, 247
getRootNode (Undocumented.PhylSim), 586
getRootNode,PhylSim-method
 (getRootNode.PhylSim), 248
ggetRootNode.PhylSim, 248
ggetBlock (Undocumented.PhylSim), 586
ggetBlock,PhylSim-method
 (getBlock.PhylSim), 249
ggetBlock.PhylSim, 249
ggetSeq (Undocumented.PhylSim), 586
ggetSeq,PhylSim-method
 (ggetSeq.PhylSim), 249
ggetSeq.PhylSim, 249
gScale (Undocumented.PhylSim), 586
gScale,BrownianInserter-method
 (gScale.BrownianInserter), 250
gScale.BrownianInserter, 250
getScaledMatrix
 (Undocumented.PhylSim), 586
gScaledMatrix,QMatrix-method
 (gScaledMatrix.QMatrix), 251
gScaledMatrix.QMatrix, 251
ggetSeqFromNode (Undocumented.PhylSim), 586
ggetSeqFromNode,PhylSim-method
 (ggetSeqFromNode.PhylSim), 252
ggetSeqFromNode.PhylSim, 252
ggetSequence (Undocumented.PhylSim), 586
ggetSequence,Site-method
 (ggetSequence.Site), 253
ggetSequence.Site, 253
ggetSequences (Undocumented.PhylSim), 586
getSequences,PhylSim-method
 (getSequences.PhylSim), 254
ggetSequences.PhylSim, 254
ggetSite (Undocumented.PhylSim), 586
ggetSite,Event-method (ggetSite.Event), 255
ggetSite.Event, 255
ggetSites (Undocumented.PhylSim), 586
ggetSites,Sequence-method
 (ggetSites.Sequence), 256
ggetSites.Sequence, 256
ggetSiteSpecificParamIds
 (Undocumented.PhylSim), 586
ggetSiteSpecificParamIds,Process-method
 (ggetSiteSpecificParamIds.Process), 257
ggetSiteSpecificParamIds.Process, 257
ggetSiteSpecificParamList
 (Undocumented.PhylSim), 586
ggetSiteSpecificParamList,Process-method
 (ggetSiteSpecificParamList.Process), 258
ggetSiteSpecificParamList.Process, 258
ggetSize (Undocumented.PhylSim), 586
ggetSize,Alphabet-method
 (getSize.Alphabet), 259
ggetSize.Alphabet, 259
ggetSizes (Undocumented.PhylSim), 586
ggetSizes,DiscreteDeleter-method
 (getSizes.DiscreteDeleter), 260
ggetSizes.DiscreteDeleter, 260
ggetSizes.DiscreteInsertor, 261
ggetSite (Undocumented.PhylSim), 586
ggetSite,Site-method (ggetSite.Site), 262
ggetSite.Site, 262
ggetStates (Undocumented.PhylSim), 586
ggetStates,Sequence-method
 (ggetStates.Sequence), 263
ggetStates.Sequence, 263
ggetString (Undocumented.PhylSim), 586
ggetString,Sequence-method
 (getString.Sequence), 264
ggetString.Sequence, 264
ggetString (Undocumented.PhylSim), 586
ggetString,Sequence-method
 (getString.Sequence), 265
ggetString.Sequence, 265
ggetSymbolFreqs (Undocumented.PhylSim), 586
ggetSymbolFreqs,Sequence-method
 (getSymbolFreqs.Sequence), 266
ggetSymbolFreqs.Sequence, 266
ggetSymbolLength
 (Undocumented.PhylSim), 586
ggetSymbolLength,Alphabet-method
 (getSymbolLength.Alphabet), 267
ggetSymbolLength.Alphabet, 267
ggetSymbols (Undocumented.PhylSim), 586
ggetSymbols,Alphabet-method
 (getSymbols.Alphabet), 268
ggetSymbols.Alphabet, 268
gtableId (Undocumented.PhylSim), 586
gtableId,CodonAlphabet-method
 (gtableId.CodonAlphabet), 269
gtableId.CodonAlphabet, 269
ggetTargetState (Undocumented.PhylSim), 586
getTemplateSeq (Undocumented.PhylOm), 586
getTemplateSeq, GeneralInsertor-method
    (getTemplateSeq, GeneralInsertor), 269
getTemplateSeq, GeneralInsertor, 269
getTheta (Undocumented.PhylOm), 586
getTheta, T92-method (getTheta.T92), 270
getTheta.T92, 270
getTipLabels (Undocumented.PhylOm), 586
getTipLabels, PhylOm-method
    (getTipLabels.PhylOm), 271
getTips (Undocumented.PhylOm), 586
getTips, PhylOm-method
    (getTips.PhylOm), 272
getTips, PhylOm, 272
getToleranceMargin (Undocumented.PhylOm), 586
getToleranceMargin, FastFieldDeleter-method
    (getToleranceMargin, FastFieldDeleter), 273
getToleranceMargin, FastFieldDeleter, 273
getTotalRate (Undocumented.PhylOm), 586
getTotalRate, Site-method
    (getTotalRate, Site), 274
getTotalRate, Site, 274
getTotalRates (Undocumented.PhylOm), 586
getTotalRates, Sequence-method
    (getTotalRates, Sequence), 275
getTotalRates, Sequence, 275
getTotalRatesFromRange (Undocumented.PhylOm), 586
getTotalRatesFromRange, Sequence-method
    (getTotalRatesFromRange, Sequence), 275
getTotalRatesFromRange, Sequence, 275
getTransTable (Undocumented.PhylOm), 586
getTransTable, CodonAlphabet-method
    (getTransTable, CodonAlphabet), 276
getTransTable, CodonAlphabet, 276
getTreeLength (Undocumented.PhylOm), 586
getTreeLength, PhylOm-method
    (getTreeLength, PhylOm), 277
globalConsistencyCheck
is.GY94, 310
is.na.PSRoot-method(is.na.PSRoot), 311
is.na.PSRoot, 311
is.phylo, 312
is.Process, 313
is.PSRoot, 314
is.QMatrix, 315
is.Sequence, 316
is.Site, 317
is.tip,PhyloSim-method
(is.tip.PhylSim), 318
is.tip.PhylSim, 318
is.ToleranceSubstitution, 319
isAttached(undocumented.PhylSim), 586
isAttached,Site-method
(isAttached.Site), 320
isAttached.Site, 320
isEmpty(undocumented.PhylSim), 586
isEmpty,Alphabet-method
(isEmpty.Alphabet), 321
isEmpty.Alphabet, 321
isStartCodon(undocumented.PhylSim), 586
isStartCodon,CodonAlphabet-method
(isStartCodon.CodonAlphabet), 322
isStartCodon.CodonAlphabet, 322
isStopCodon(undocumented.PhylSim), 586
isStopCodon,CodonAlphabet-method
(isStopCodon.CodonAlphabet), 323
isStopCodon.CodonAlphabet, 323
JC69, 64, 132, 324, 372, 389, 393, 564, 587
JC69.checkConsistency
(checkConsistency.JC69), 63
JC69.summary(summary.JC69), 564
JTT, 20, 132, 326, 372, 389, 393
JTT.dcmut, 20, 132, 328, 372, 389, 393
K80, 64, 132, 152, 237, 244, 289, 330, 372, 389, 393, 429, 494, 502, 565, 587
K80.checkConsistency
(checkConsistency.K80), 64
K80.getBaseFreqs(getBaseFreqs.K80), 152
K80.getRateParam(getRateParam.K80), 236
K80.getRateParamList
(getRateParamList.K80), 244
K80.setBaseFreqs(setBaseFreqs.K80), 429
K80.setRateParam(setRateParam.K80), 494
K80.setRateParamList
(setRateParamList.K80), 502
K80.summary(summary.K80), 564
K81.checkConsistency
(checkConsistency.K81), 65
K81.getBaseFreqs(getBaseFreqs.K81), 153
K81.getRateParam(getRateParam.K81), 237
K81.getRateParamList
(getRateParamList.K81), 245
K81.setBaseFreqs(setBaseFreqs.K81), 430
K81.setRateParam(setRateParam.K81), 495
K81.setRateParamList
(setRateParamList.K81), 503
K81.summary(summary.K81), 565
LG, 20, 132, 336, 372, 389, 393
list(undocumented.PhylSim), 586
ll,PSRoot-method(ll.PSRoot), 338
ll.PSRoot, 338
Log(undocumented.PhylSim), 586
Log,PhylSim-method(Log,PhylSim), 339
Log,PhylSim, 339
mtArt, 20, 132, 340, 372, 389, 393
mtMam, 20, 132, 342, 372, 389, 393
mtREV24, 20, 132, 344, 372, 389, 393
MtZoa, 20, 132, 347, 372, 389, 393
my(undocumented.PhylSim), 586
my.all.equal,PSRoot-method
(my.all.equal.PSRoot), 349
my.all.equal.PSRoot, 349
newAA Matrix(undocumented.PhylSim), 586
newAA Matrix,AminoAcidSubst-method
(newAA Matrix.AminoAcidSubst), 350
newAA Matrix.AminoAcidSubst, 350
newMatrix(undocumented.PhylSim), 586
newMatrix,CodonSubst-method
(newMatrix.CodonSubst), 350
newMatrix.CodonSubst, 350
NucleotideAlphabet, 14, 351, 372, 393
NucleotideSequence, 352, 372, 393, 403, 410
NucleotideSequence.revComp
(revComp,NucleotideSequence), 403
Object, 14, 17, 18, 20, 22, 37, 38, 40, 42, 78, 80, 82, 84, 86, 88, 92, 100, 102, 105, 107, 110, 115, 117, 120, 124, 126, 129, 132, 289, 292, 300, 324, 326, 328, 330, 333, 336, 340, 342, 344, 347, 351, 352, 363, 365, 369, 389, 393, 396, 410, 539, 576, 578, 581, 586, 589
omegaHist (Undocumented.PhylSim), 586
omegaHist, CodonSequence-method
(omegaHist.CodonSequence), 354
omegaHist.CodonSequence, 354
omegaVarM0 (Undocumented.PhylSim), 586
omegaVarM0, CodonSequence-method
(omegaVarM0.CodonSequence), 355
omegaVarM0, CodonSequence, 355
omegaVarM1 (Undocumented.PhylSim), 586
omegaVarM1, CodonSequence-method
(omegaVarM1.CodonSequence), 357
omegaVarM1, CodonSequence, 357
omegaVarM1@Cont
(Undocumented.PhylSim), 586
omegaVarM2 (Undocumented.PhylSim), 586
omegaVarM2, CodonSequence-method
(omegaVarM2.CodonSequence), 358
omegaVarM2, CodonSequence, 358
omegaVarM3 (Undocumented.PhylSim), 586
omegaVarM3, CodonSequence-method
(omegaVarM3.CodonSequence), 360
omegaVarM3, CodonSequence, 360
omegaVarM4 (Undocumented.PhylSim), 586
omegaVarM4, CodonSequence-method
(omegaVarM4.CodonSequence), 361
omegaVarM4, CodonSequence, 361
PAM, 20, 132, 363, 372, 389, 393
PAM.dcmut, 20, 132, 365, 372, 389, 393
Perform (Undocumented.PhylSim), 586
Perform, Event-method (Perform.Event), 367
Perform.Event, 367
phylosim (PhylSim), 368
PhyloSim.as.character
(as.character.PhylSim), 27
PhyloSim.attachHookToNode
(attachHookToNode.PhylSim), 33
PhyloSim.attachSeqToNode
(attachSeqToNode.PhylSim), 36
PhyloSim.checkConsistency
(checkConsistency.PhylSim), 65
PhyloSim.Debug (Debug.PhylSim), 94
PhyloSim.detachHookFromNode
(detachHookFromNode.PhylSim), 96
PhyloSim.detachSeqFromNode
(detachSeqFromNode.PhylSim), 98
PhyloSim.exportStatTree
(exportStatTree.PhylSim), 113
PhyloSim.getAlignment
(getAlignment.PhylSim), 138
PhyloSim.getAlignmentLength
(getAlignmentLength.PhylSim), 139
PhyloSim.getBranchEvents
(getBranchEvents.PhylSim), 156
PhyloSim/getEdge (getEdge.PhylSim), 164
PhyloSim.getEdges (getEdges.PhylSim), 165
PhyloSim.getId (getId.PhylSim), 184
PhyloSim.getLogFile
(getLogFile.PhylSim), 195
PhyloSim.getLogLevel
(getLogLevel.PhylSim), 196
PhyloSim.getName (getName.PhylSim), 202
PhyloSim.getNedges
(getNedges.PhylSim), 206
PhyloSim.getNodes (getNodes.PhylSim), 207
PhyloSim.getNtips (getNtips.PhylSim), 208
PhyloSim.getPhylo (getPhylo.PhylSim), 213
PhyloSim.getRootNode
(getRootNode.PhylSim), 248
PhyloSim.getRootSeq
(getRootSeq.PhylSim), 249
PhyloSim.getSeqFromNode
(getSeqFromNode.PhylSim), 252
PhyloSim.getSequences
(getSequences.PhyloSim), 254
PhyloSim.getTipLabels
  (getTipLabels.PhyloSim), 271
PhyloSim.getTips (getTips.PhyloSim), 272
PhyloSim.getTreeLength
  (getTreeLength.PhyloSim), 277
PhyloSim.is.tip (is.tip.PhyloSim), 318
PhyloSim.Log (Log.PhyloSim), 339
PhyloSim.plot (plot.PhyloSim), 379
PhyloSim.readAlignment
  (readAlignment.PhyloSim), 398
PhyloSim.readTree (readTree.PhyloSim), 399
PhyloSim.saveAlignment
  (saveAlignment.PhyloSim), 407
PhyloSim.scaleTree
  (scaleTree.PhyloSim), 409
PhyloSim.setAlignment
  (setAlignment.PhyloSim), 417
PhyloSim.setBranchEvents
  (setBranchEvents.PhyloSim), 432
PhyloSim.setEdges (setEdges.PhyloSim), 438
PhyloSim.setId (setId.PhyloSim), 444
PhyloSim.setLogFile
  (setLogFile.PhyloSim), 453
PhyloSim.setLogLevel
  (setLogLevel.PhyloSim), 454
PhyloSim.setName (setName.PhyloSim), 459
PhyloSim.setNedges
  (setNedges.PhyloSim), 463
PhyloSim.setNodes (setNodes.PhyloSim), 464
PhyloSim.setNtips (setNtips.PhyloSim), 464
PhyloSim.setPhylo (setPhylo.PhyloSim), 468
PhyloSim.setRootNode
  (setRootNode.PhyloSim), 506
PhyloSim.setRootSeq
  (setRootSeq.PhyloSim), 507
PhyloSim.setSequences
  (setSequences.PhyloSim), 511
PhyloSim.setTipLabels
  (setTipLabels.PhyloSim), 523
PhyloSim.setTips (setTips.PhyloSim), 524
PhyloSim.setTreeLength
  (setTreeLength.PhyloSim), 528
PhyloSim.Simulate (Simulate.PhyloSim), 537
PhyloSim.summary (summary.PhyloSim), 566
PhyloSim.Undocumented
  (Undocumented.PhyloSim), 586
plot (Undocumented.PhyloSim), 586
plot,ContinuousDeleter-method
  (plot.ContinuousDeleter), 374
plot,ContinuousInsertor-method
  (plot.ContinuousInsertor), 375
plot,DiscreteDeleter-method
  (plot.DiscreteDeleter), 376
plot,DiscreteInsertor-method
  (plot.DiscreteInsertor), 377
plot,GeneralSubstitution-method
  (plot.GeneralSubstitution), 378
plot,PhyloSim-method (plot.PhyloSim), 379
plot,Sequence-method (plot.Sequence), 381
plot,ToleranceSubstitution-method
  (plot.ToleranceSubstitution), 382
plotParametersAtSites
  (Undocumented.PhyloSim), 586
plotParametersAtSites,Sequence-method
  (plotParametersAtSites.Sequence), 383
plotParametersAtSites.Sequence, 383
plusGamma (Undocumented.PhyloSim), 586
plusGamma,Sequence-method
  (plusGamma.Sequence), 384
plusGamma.Sequence, 384
plusInvGamma (Undocumented.PhyloSim), 586
plusInvGamma,Sequence-method
  (plusInvGamma.Sequence), 386
plusInvGamma.Sequence, 386
print,PSRootSummary-method
  (print.PSRootSummary), 387
print, QMatrix-method (print.QMatrix), 388
print.PSRootSummary, 387
print.QMatrix, 388
Process == (==.Process), 13
Process.as.character
(as.character.Process), 28
Process.checkConsistency
(checkConsistency.Process), 66
Process.clone (clone.Process), 75
Process.getAlphabet
(getAlphabet.Process), 141
Process.getEventsAtSite
(getEventsAtSite.Process), 180
Process.getId (getId.Process), 185
Process.getName (getName.Process), 203
Process.getParameterAtSite
(getParameterAtSite.Process), 211
Process.getSiteSpecificParamIds
(getSiteSpecificParamIds.Process), 257
Process.getSiteSpecificParamList
(getSiteSpecificParamList.Process), 258
Process.getWriteProtected
(getWriteProtected.Process), 285
Process.hasSiteSpecificParameter
(hasSiteSpecificParameter.Process), 294
Process.hasUndefinedRate
(hasUndefinedRate.Process), 298
Process.setAlphabet
(setAlphabet.Process), 418
Process.setId (setId.Process), 445
Process.setName (setName.Process), 460
Process.setParameterAtSite
setParameterAtSite.Process), 466
Process.setSiteSpecificParamIds
(setSiteSpecificParamIds.Process), 512
Process.setSiteSpecificParamList
(setSiteSpecificParamList.Process), 513
Process.setWriteProtected
(setWriteProtected.Process), 535
Process.summary (summary.Process), 567
proposeLength (Undocumented.PhyloSim), 586
proposeLength.GeneraInDel-method
(proposeLength.GeneraInDel), 392
proposeLength.GeneraInDel, 392
PSRoot.checkConsistency
(checkConsistency.PSRoot), 67
PSRoot.enableVirtual
(enableVirtual.PSRoot), 109
PSRoot.getComments
(getComments.PSRoot), 158
PSRoot.getMethodsList
(getMethodsList.PSRoot), 200
PSRoot.globalConsistencyCheck
(globalConsistencyCheck.PSRoot), 288
PSRoot.intersect.list
(intersect.list.PSRoot), 303
PSRoot.is.na (is.na.PSRoot), 311
PSRoot.ll (ll.PSRoot), 338
PSRoot.my.all.equal
(my.all.equal.PSRoot), 349
PSRoot.setComments
(setComments.PSRoot), 434
PSRoot.setMethodsList
(setMethodsList.PSRoot), 458
PSRoot.summary (summary.PSRoot), 568
PSRoot.virtualAssignmentForbidden
  (virtualAssignmentForbidden.PSRoot), 588
PSRootSummary, 68, 387, 395, 396
PSRootSummary.checkConsistency
  (checkConsistency.PSRootSummary), 67
PSRootSummary.print
  (print.PSRootSummary), 387
QMatrix, 29, 68, 142, 169, 186, 197, 204, 218,
  227, 230, 251, 315, 372, 388, 393,
  396, 409, 420, 446, 455, 462, 474,
  484, 488, 509, 569
QMatrix.as.character
  (as.character.QMatrix), 29
QMatrix.checkConsistency
  (checkConsistency.QMatrix), 68
QMatrix.getAlphabet
  (getAlphabet.QMatrix), 142
QMatrix.getEventRate
  (getEventRate.QMatrix), 169
QMatrix.getId (getId.QMatrix), 186
QMatrix.getMMatrix (getMMatrix.QMatrix), 197
QMatrix.getName (getName.QMatrix), 204
QMatrix.getProcess
  (getProcess.QMatrix), 218
QMatrix.getRate (getRate.QMatrix), 227
QMatrix.getRateList
  (getRateList.QMatrix), 230
QMatrix.getScaledMatrix
  (getScaledMatrix.QMatrix), 251
QMatrix.getWriteProtected
  (getWriteProtected.QMatrix), 286
QMatrix.print (print.QMatrix), 388
QMatrix.Scale (Scale.QMatrix), 408
QMatrix.setAlphabet
  (setAlphabet.QMatrix), 419
QMatrix.setId (setId.QMatrix), 445
QMatrix.setMMatrix (setMMatrix.QMatrix), 455
QMatrix.setName (setName.QMatrix), 461
QMatrix.setProcess
  (setProcess.QMatrix), 473
QMatrix.setRate (setRate.QMatrix), 484
QMatrix.setRateList
  (setRateList.QMatrix), 487
QMatrix.setScaledMatrix
  (setScaledMatrix.QMatrix), 509
QMatrix.setWriteProtected
  (setWriteProtected.QMatrix), 536
QMatrix.summary (summary.QMatrix), 569
R.oo, 368
readAlignment (Undocumented.PhylSim), 586
readAlignment.PhylSim-method
  (readAlignment.PhylSim), 398
readAlignment.PhylSim, 398
readTree (Undocumented.PhylSim), 586
readTree.PhylSim-method
  (readTree.PhylSim), 399
readTree.PhylSim, 399
rescaleQMatrix (Undocumented.PhylSim), 586
rescaleQMatrix, GeneralSubstitution-method
  (rescaleQMatrix.GeneralSubstitution), 401
rescaleQMatrix, ToleranceSubstitution-method
  (rescaleQMatrix.ToleranceSubstitution), 402
rescaleQMatrix, GeneralSubstitution,
  401
rescaleQMatrix, ToleranceSubstitution,
  402
revComp (Undocumented.PhylSim), 586
revComp, NucleotideSequence-method
  (revComp.NucleotideSequence), 403
revComp.NucleotideSequence, 403
sampleState (Undocumented.PhylSim), 586
sampleState, GeneralSubstitution-method
  (sampleState.GeneralSubstitution), 404
sampleState, ToleranceSubstitution-method
  (sampleState.ToleranceSubstitution), 405
sampleState, GeneralSubstitution, 404
sampleState.ToleranceSubstitution, 405
sampleStates (Undocumented.PhylSim), 586
INDEX

sampleStates.Sequence-method
(sampleStates.Sequence), 406
sampleStates.Sequence, 406
saveAlignment (Undocumented.PhyloSim), 586
saveAlignment.PhyloSim-method
(saveAlignment.PhyloSim), 407
saveLoadReference
(Undocumented.PhyloSim), 586
Scale (Undocumented.PhyloSim), 586
Scale,QMatrix-method (Scale,QMatrix), 408
Scale,QMatrix, 408
scaleTree (Undocumented.PhyloSim), 586
scaleTree.PhyloSim-method
(scaleTree.PhyloSim), 409
scaleTree.PhyloSim, 409
Sequence.as.character
(as.character.Sequence), 30
Sequence.attachProcess
(attachProcess.Sequence), 34
Sequence.checkConsistency
(checkConsistency.Sequence), 69
Sequence.clearStates
(clearStates.Sequence), 73
Sequence.clone (clone.Sequence), 76
Sequence.copySubSequence
(copySubSequence.Sequence), 91
Sequence.deleteSubSequence
(deleteSubSequence.Sequence), 95
Sequence.detachProcess
(detachProcess.Sequence), 97
Sequence.getAlphabets
(getAlphabets.Sequence), 145
Sequence.getAncestral
(getAncestral.Sequence), 146
Sequence.getBigRate
(getBigRate.Sequence), 155
Sequence.getCumulativeRates
(getCumulativeRates.Sequence), 159
Sequence.getCumulativeRatesFromRange
(getCumulativeRatesFromRange.Sequence), 160
Sequence.getDeletionTolerance
(getDeletionTolerance.Sequence), 161
Sequence.getEvents
(getEvents.Sequence), 173
Sequence.getId (getId.Sequence), 187
Sequence.getInsertionTolerance
(getInsertionTolerance.Sequence), 189
Sequence.getLength
(getLength.Sequence), 192
Sequence.getName (getName.Sequence), 205
Sequence.getParameterAtSites
(getParameterAtSites.Sequence), 212
Sequence.getProcesses
(getProcesses.Sequence), 219
Sequence.getRateMultipliers
(getRateMultipliers.Sequence), 232
Sequence.getSites ( getSites.Sequence), 256
Sequence.getStates
(getStates.Sequence), 263
Sequence.getString
(getString.Sequence), 264
Sequence.getSymbolFreqs
(getSymbolFreqs.Sequence), 265
Sequence.getTotalRates
(getTotalRates.Sequence), 275
Sequence.getTotalRatesFromRange
(getTotalRatesFromRange.Sequence), 275
Sequence.getUniqueAlphabets
(getUniqueAlphabets.Sequence), 281
Sequence.getUniqueProcesses
(getUniqueProcesses.Sequence), 282
Sequence.getWriteProtected
(getWriteProtected.Sequence), 287
Sequence.insertSequence
Sequence.plot (plot.Sequence), 381
Sequence.plotParametersAtSites (plotParametersAtSites.Sequence), 383
Sequence.plusGamma (plusGamma.Sequence), 384
Sequence.plusInvGamma (plusInvGamma.Sequence), 386
Sequence.sampleStates (sampleStates.Sequence), 406
Sequence.setAlphabets (setAlphabets.Sequence), 422
Sequence.setAncestral (setAncestral.Sequence), 423
Sequence.setBigRate (setBigRate.Sequence), 431
Sequence.setCumulativeRates (setCumulativeRates.Sequence), 435
Sequence.setDeletionTolerance (setDeletionTolerance.Sequence), 435
Sequence.setId (setId.Sequence), 446
Sequence.setInsertionTolerance (setInsertionTolerance.Sequence), 448
Sequence.setLength (setLength.Sequence), 451
Sequence.setName (setName.Sequence), 462
Sequence.setParameterAtSites (setParameterAtSites.Sequence), 467
Sequence.setProcesses (setProcesses.Sequence), 474
Sequence.setRateMultipliers (setRateMultipliers.Sequence), 489
Sequence.setStates (setStates.Sequence), 517
Sequence.setString (setString.Sequence), 518
Sequence.setTotalRates (setTotalRates.Sequence), 526
Sequence.setUniqueAlphabets (setUniqueAlphabets.Sequence), 531
Sequence.setUniqueProcesses
(setUniqueProcesses.Sequence), 532
Sequence.setWriteProtected (setWriteProtected.Sequence), 536
Sequence.summary (summary.Sequence), 570
setAcceptBy (Undocumented.PhyloSim), 586
setAcceptBy (GeneralInDel-method)
(setAcceptBy.GeneralInDel), 414
setAcceptBy (GeneralInDel, 414
(setAcceptBy.GeneralInDel), 414
setAcceptWin (Undocumented.PhyloSim), 586
setAcceptWin, GeneralInsertor-method
(setAcceptWin.GeneralInsertor), 416
setAlignment (Undocumented.PhyloSim), 586
setAlignment, PhyloSim-method
(setAlignment.PhyloSim), 417
setAlignment.PhyloSim, 417
setAlphabet (Undocumented.PhyloSim), 586
setAlphabet, GeneralSubstitution-method
(setAlphabet.GeneralSubstitution), 417
setAlphabet, Process-method
(setAlphabet.Process), 418
setAlphabet, QMatrix-method
(setAlphabet.QMatrix), 419
setAlphabet, Site-method
(setAlphabet.Site), 420
setAlphabet, ToleranceSubstitution-method
(setAlphabet.ToleranceSubstitution), 421
setAlphabet, GeneralSubstitution, 417
setAlphabet, Process, 418
setAlphabet, QMatrix, 419
setAlphabet, Site, 420
setAlphabet, ToleranceSubstitution, 421
setAlphabets (Undocumented.PhyloSim), 586
setAlphabets, Sequence-method
(setAlphabets.Sequence), 422
setAlphabets.Sequence, 422
setAncestral (Undocumented.PhyloSim), 586
setAncestral, Sequence-method
  (setAncestral.Sequence), 423
setAncestral, Site-method
  (setAncestral.Site), 424
setAncestral, Sequence, 423
setBaseFreqs (Undocumented.PhylSim), 586
setBaseFreqs, F81-method
  (setBaseFreqs.F81), 425
setBaseFreqs, F84-method
  (setBaseFreqs.F84), 426
setBaseFreqs, GTR-method
  (setBaseFreqs.GTR), 427
setBaseFreqs, HKY-method
  (setBaseFreqs.HKY), 428
setBaseFreqs, K80-method
  (setBaseFreqs.K80), 429
setBaseFreqs, K81-method
  (setBaseFreqs.K81), 430
setBaseFreqs, TN93-method
  (setBaseFreqs.TN93), 430
setBigRate (Undocumented.PhylSim), 586
setBigRate, Sequence-method
  (setBigRate.Sequence), 431
setBigRate, Sequence, 431
setBranchEvents
  (Undocumented.PhylSim), 586
setBranchEvents, PhylSim-method
  (setBranchEvents.PhylSim), 432
setCodonFreqs (Undocumented.PhylSim), 586
setCodonFreqs, GY94-method
  (setCodonFreqs.GY94), 433
setCodonFreqs, GY94, 433
setComments (Undocumented.PhylSim), 586
setComments, PSRoot-method
  (setComments.PSRoot), 434
setComments, PSRoot, 434
setCumulativeRates
  (Undocumented.PhylSim), 586
setCumulativeRates, Sequence-method
  (setCumulativeRates.Sequence), 435
setDeletionTolerance
  (Undocumented.PhylSim), 586
setDeletionTolerance, Sequence-method
  (setDeletionTolerance.Sequence), 435
setDist (Undocumented.PhylSim), 586
setDist, ContinuousDeleter-method
  (setDist.ContinuousDeleter), 436
setDist, ContinuousInsertor-method
  (setDist.ContinuousInsertor), 437
setDist, ContinuousDeleter, 436
setDist, ContinuousInsertor, 437
setEvents (Undocumented.PhylSim), 586
setEvents, PhylSim-method
  (setEvents.PhylSim), 441
setEvents, Site-method (setEvents.Site), 441
setGenerateBy (Undocumented.PhylSim), 586
setGenerateBy, GeneralInsertor-method
  (setGenerateBy.GeneralInsertor), 442
setHandler (Undocumented.PhylSim), 586
setHandler, Event-method
setNodes (Undocumented.PhylSim), 586
setNodes,PhyloSim-method
(setNodes.PhylSim), 464
setNodes.PhylSim, 464
setNtips (Undocumented.PhylSim), 586
setNtips,PhyloSim-method
(setNtips.PhylSim), 464
setNtips.PhylSim, 464
setOmegas (Undocumented.PhylSim), 586
setOmegas,CodonSequence-method
(setOmegas.CodonSequence), 465
setOmegas.CodonSequence, 465
setOmegas.Sequence
(setOmegas.CodonSequence), 465
setParameterAtSite
(Undocumented.PhylSim), 586
setParameterAtSite,Process-method
(setParameterAtSite.Process), 466
setParameterAtSite.Process, 466
setParameterAtSites
(Undocumented.PhylSim), 586
setParameterAtSites,Sequence-method
(setParameterAtSites.Sequence), 467
setParameterAtSites.Sequence, 467
setPhylo (Undocumented.PhylSim), 586
setPhylo,PhyloSim-method
(setPhylo.PhylSim), 468
setPhylo.PhylSim, 468
setPosition (Undocumented.PhylSim), 586
setPosition,Event-method
(setPosition.Event), 469
setPosition.Event, 469
setProbS (Undocumented.PhylSim), 586
setProbS,DiscreteDeleter-method
(setProbS.DiscreteDeleter), 470
setProbS,DiscreteInsertor-method
(setProbS.DiscreteInsertor), 471
setProbS.DiscreteDeleter, 470
setProbS.DiscreteInsertor, 471
setProcess (Undocumented.PhylSim), 586
setProcess,Event-method
(setProcess.Event), 472
setProcess,QMatrix-method
(setProcess.QMatrix), 473
setProcess.Event, 472
setProcess.QMatrix, 473
setProcesses (Undocumented.PhylSim), 586
setProcesses,Sequence-method
(setProcesses.Sequence), 474
setProcesses,Site-method
(setProcesses.Site), 475
setProcesses.Sequence, 474
setProcesses.Site, 475
setProposeBy (Undocumented.PhylSim), 586
setProposeBy,GeneralInDel-method
(setProposeBy.GeneralInDel), 476
setPropeBy,GeneralInDel, 476
setProposeBy,GeneralInsertor
(setProposeBy.GeneralInsertor), 476
setQMatrix (Undocumented.PhylSim), 586
setQMatrix,GeneralSubstitution-method
(setQMatrix.GeneralSubstitution), 477
setQMatrix,GeneralSubstitution, 477
setQMatrix,ToleranceSubstitution-method
(setQMatrix.ToleranceSubstitution), 478
setQMatrix.GeneraSubstitution, 477
setQMatrix,GerntalSubstitution, 478
setRate (Undocumented.PhylSim), 586
setRate,Event-method(setRate.Event), 480
setRate,Event, 480
setRate,GeneraInDel-method
(setRate.GeneraInDel), 481
setRate,GeneraSubstitution-method
(setRate.GeneraSubstitution), 482
setRate,GeneralInDel, 481
setRate,GeneralSubstitution-method
(setRate.GeneraSubstitution), 482
setRate,GY94-method(setRate.GY94), 483
setRate,QMatrix-method
(setRate.QMatrix), 484
setRate,GeneralInDel, 481
setRate,GeneralSubstitution, 482, 483
setRate.GY94, 483
setRate.QMatrix, 484
setRate.ToleranceSubstitution, 485
setRateList (Undocumented.PhyloSim), 586
setRateList, GeneralSubstitution-method
(setRateList.GenericSubstitution), 486
setRateList, QMatrix-method
(setRateList.QMatrix), 487
setRateList, ToleranceSubstitution-method
(setRateList.ToleranceSubstitution), 488
setRateList, GeneralSubstitution, 486
setRateList, QMatrix, 487
setRateList, ToleranceSubstitution, 488
setRateMultipliers
(Undocumented.PhyloSim), 586
setRateMultipliers, Sequence-method
(setRateMultipliers.Sequence), 489
setRateMultipliers.Sequence, 489
setRateParam (Undocumented.PhyloSim), 586
setRateParam, F81-method
(setRateParam.F81), 490
setRateParam, F84-method
(setRateParam.F84), 491
setRateParam, GTR-method
(setRateParam.GTR), 492
setRateParam, HKY-method
(setRateParam.HKY), 493
setRateParam, K80-method
(setRateParam.K80), 494
setRateParam, K81-method
(setRateParam.K81), 495
setRateParam, T92-method
(setRateParam.T92), 496
setRateParam, TN93-method
(setRateParam.TN93), 497
setRateParam, F81, 490
setRateParam, F84, 491
setRateParam, GTR, 492
setRateParam, HKY, 493
setRateParam, K80, 494
setRateParam, K81, 495
setRateParam, T92, 496
setRateParam, TN93, 497
setRateParamList
(Undocumented.PhyloSim), 586
setRateParamList, F81-method
(setRateParamList.F81), 498
setRateParamList, F84-method
(setRateParamList.F84), 499
setRateParamList, GTR-method
(setRateParamList.GTR), 500
setRateParamList, HKY-method
(setRateParamList.HKY), 501
setRateParamList, K80-method
(setRateParamList.K80), 502
setRateParamList, K81-method
(setRateParamList.K81), 503
setRateParamList, T92-method
(setRateParamList.T92), 504
setRateParamList, TN93-method
(setRateParamList.TN93), 505
setRateParamList.F81, 498
setRateParamList.F84, 499
setRateParamList.GTR, 500
setRateParamList.HKY, 501
setRateParamList.K80, 502
setRateParamList.K81, 503
setRateParamList.T92, 504
setRateParamList.TN93, 505
setRootNode (Undocumented.PhyloSim), 586
setRootNode, PhyloSim-method
(setRootNode.PhyloSim), 506
setRootNode.PhyloSim, 506
setRootSeq (Undocumented.PhyloSim), 586
setRootSeq.PhyloSim-method
(setRootSeq.PhyloSim), 507
setRootSeq.PhyloSim, 507
setScale (Undocumented.PhyloSim), 586
setScale, BrownianInsertor-method
(setScale.BrownianInsertor), 508
setScale.BrownianInsertor, 508
setScaleMatrix
(Undocumented.PhyloSim), 586
setScaleMatrix, QMatrix-method
(setScaleMatrix.QMatrix), 509
setScaleMatrix.QMatrix, 509
setSequence (Undocumented.PhyloSim), 586
setSequence, Site-method
(setSequence.Site), 510
setSequence.Site, 510
setSequences (Undocumented.PhyloSim), 586
setSequences, PhyloSim-method
  (setSequences.PhylomSim), 511
setSequences, PhyloSim, 511
setSite, (Undocumented, PhyloSim), 586
setSite, Event-method (setSite.Event), 511
setSite, Event, 511
setSiteSpecificParamIds
  (Undocumented, PhyloSim), 586
setSiteSpecificParamIds, Process-method
  (setSiteSpecificParamIds.Process), 512
setSiteSpecificParamIds, Process, 512
setSiteSpecificParamList
  (Undocumented, PhyloSim), 586
setSiteSpecificParamList, Process-method
  (setSiteSpecificParamList.Process), 513
setSiteSpecificParamList, Process, 513
setSize, (Undocumented, PhyloSim), 586
setSize, Alphabet-method
  (setSize.Alphabet), 514
setSize, Alphabet, 514
setSize, (Undocumented, PhyloSim), 586
setSize, DiscreteDeletron-method
  (setSize.DiscreteDeletron), 514
setSize, DiscreteInsertor-method
  (setSize.DiscreteInsertor), 515
setSize, DiscreteDeletron, 514
setSize, DiscreteInsertor, 515
setState, (Undocumented, PhyloSim), 586
setState, Site-method (setState, Site), 516
setState, Site, 516
setState, Site, (Undocumented, PhyloSim), 586
setState, Sequence-method
  (setState.Sequence), 517
setState, Sequence, 517
setString, (Undocumented, PhyloSim), 586
setString, Sequence-method
  (setString.Sequence), 518
setString, Sequence, 518
setSymbolLength
  (Undocumented, PhyloSim), 586
setSymbolLength, Alphabet-method
  (setSymbolLength, Alphabet), 519
setSymbolLength, Alphabet, 519
setSymbols, (Undocumented, PhyloSim), 586
setSymbols, Alphabet-method
  (setSymbols, Alphabet), 520
setSymbols, Alphabet, 520
setTableId, (Undocumented, PhyloSim), 586
setTableId, CodonAlphabet-method
  (setTableId, CodonAlphabet), 521
setTableId, CodonAlphabet, 521
setTargetState, (Undocumented, PhyloSim), 586
setTemplateSeq, (Undocumented, PhyloSim), 586
setTemplateSeq, GeneralInsertor-method
  (setTemplateSeq, GeneralInsertor), 521
setTemplateSeq, GeneralInsertor, 521
setTheta, (Undocumented, PhyloSim), 586
setTheta, T2-method (setTheta, T2), 522
setTheta, T2, 522
setTipLabels, (Undocumented, PhyloSim), 586
setTipLabels, (Undocumented, PhyloSim), 523
setTipLabels, PhyloSim, 523
setTips, (Undocumented, PhyloSim), 586
setTips, PhyloSim-method
  (setTips, PhyloSim), 524
setTips, PhyloSim, 524
setToleranceMargin
  (Undocumented, PhyloSim), 586
setToleranceMargin, FastFieldDeletron-method
  (setToleranceMargin, FastFieldDeletron), 525
setToleranceMargin, FastFieldDeletron, 525
setTotalRate, (Undocumented, PhyloSim), 586
setTotalRate, Site-method
  (setTotalRate, Site), 526
setTotalRate, Site, 526
setTotalRates, (Undocumented, PhyloSim), 586
setTotalRates, Sequence-method
  (setTotalRates, Sequence), 526
setTotalRates, Sequence, 526
setTransTable, (Undocumented, PhyloSim), 586
setTransTable, CodonAlphabet-method
summary, BinarySubst-method
(summary.BinarySubst), 545

summary, BrownianInsertor-method
(summary.BrownianInsertor), 546

summary, CodonAlphabet-method
(summary.CodonAlphabet), 547

summary, CodonSubst-method
(summary.CodonSubst), 548

summary, ContinuousDeleter-method
(summary.ContinuousDeleter), 549

summary, ContinuousInsertor-method
(summary.ContinuousInsertor), 550

summary, DiscreteDeleter-method
(summary.DiscreteDeleter), 551

summary, DiscreteInsertor-method
(summary.DiscreteInsertor), 552

summary, Event-method (summary.Event), 553

summary, F81-method (summary.F81), 554

summary, F84-method (summary.F84), 555

summary, FastFieldDeleter-method
(summary.FastFieldDeleter), 556

summary, GeneralDeleter-method
(summary.GeneralDeleter), 557

summary, GeneralIndel-method
(summary.GeneralIndel), 558

summary, GeneralInsertor-method
(summary.GeneralInsertor), 559

summary, GeneralSubstitution-method
(summary.GeneralSubstitution), 560

summary, GTR-method (summary.GTR), 561

summary, GY94-method (summary.GY94), 562

summary, HKY-method (summary.HKY), 563

summary, JC69-method (summary.JC69), 564

summary, K80-method (summary.K80), 564

summary, K81-method (summary.K81), 565

summary, PhyloSim-method
(summary.PhylodSim), 566

summary, Process-method
(summary.Process), 567

summary, PSRoot-method (summary.PSRoot), 568

summary, QMatrix-method
(summary.QMatrix), 569

summary, Sequence-method
(summary.Sequence), 570

summary, Site-method (summary.Site), 571

summary, T92-method (summary.T92), 572

summary, TN93-method (summary.TN93), 573

summary, ToleranceSubstitution-method
(summary.ToleranceSubstitution), 574

summary, UNREST-method (summary.UNREST), 575

summary, Alphabet, 543

summary, AminoAcidSubst, 544

summary, BinarySubst, 545

summary, BrownianInsertor, 546

summary, CodonAlphabet, 547

summary, CodonSubst, 548

summary, ContinuousDeleter, 549

summary, ContinuousInsertor, 550

summary, DiscreteDeleter, 551

summary, DiscreteInsertor, 552

summary, Event, 553

summary, F81, 554

summary, F84, 555

summary, FastFieldDeleter, 556

summary, GeneralDeleter, 557

summary, GeneralIndel, 558

summary, GeneralInsertor, 559

summary, GeneralSubstitution, 560

summary, GTR, 561

summary, GY94, 562

summary, HKY, 563

summary, JC69, 564

summary, K80, 564

summary, K81, 565

summary, PhyloSim, 566

summary, Process, 567

summary, PSRoot, 568

summary, QMatrix, 569

summary, Sequence, 570

summary, Site, 571

summary, T92, 572

summary, TN93, 573

summary, ToleranceSubstitution, 574

summary, UNREST, 575

T92, 70, 132, 238, 246, 270, 289, 372, 389, 393, 496, 504, 523, 572, 576, 587

T92.checkConsistency
(checkConsistency.T92), 70

T92.getRateParam (getRateParam.T92), 238
T92.getDateParamList
  (getDateParamList.T92), 246
T92.getTheta (getTheta.T92), 270
T92.setRateParam (setRateParam.T92), 496
T92.setRateParamList
  (setRateParamList.T92), 504
T92.setTheta (setTheta.T92), 522
T92.summary (summary.T92), 572
tip (Undocumented.PhyloSim), 586
TN93.checkConsistency (checkConsistency.TN93), 71
TN93.getBaseFreqs (getBaseFreqs.TN93), 154
TN93.getDateParam (getDateParam.TN93), 239
TN93.getDateParamList (getDateParamList.TN93), 247
TN93.setBaseFreqs (setBaseFreqs.TN93), 430
TN93.setRateParam (setRateParam.TN93), 497
TN93.setRateParamList (setRateParamList.TN93), 505
TN93.summary (summary.TN93), 573
ToleranceSubstitution, 32, 72, 77, 144, 167, 170, 173, 182, 223, 228, 231, 299, 319, 383, 389, 393, 402, 405, 422, 441, 479, 486, 489, 574, 581
ToleranceSubstitution.as.character
  (as.character.ToleranceSubstitution), 32
ToleranceSubstitution.checkConsistency (checkConsistency.ToleranceSubstitution), 71
ToleranceSubstitution.clone
  (clone.ToleranceSubstitution), 77
ToleranceSubstitution.getAlphabet (getAlphabet.ToleranceSubstitution), 144
ToleranceSubstitution.getEquiDist (getEquiDist.ToleranceSubstitution), 167
ToleranceSubstitution.getEventRate
  (getEventRate.ToleranceSubstitution), 170
ToleranceSubstitution.getEventRateAtSite
  (getEventRateAtSite.ToleranceSubstitution), 172
ToleranceSubstitution.getEventsAtSite
  (getEventsAtSite.ToleranceSubstitution), 181
ToleranceSubstitution.getQMatrix
  (getQMatrix.ToleranceSubstitution), 223
ToleranceSubstitution.setRate
  (setRate.ToleranceSubstitution), 228
ToleranceSubstitution.setRateList
  (setRateList.ToleranceSubstitution), 231
ToleranceSubstitution.hasUndefinedRate
  (hasUndefinedRate.ToleranceSubstitution), 299
ToleranceSubstitution.plot
  (plot.ToleranceSubstitution), 382
ToleranceSubstitution.rescaleQMatrix
  (rescaleQMatrix.ToleranceSubstitution), 402
ToleranceSubstitution.sampleState
  (sampleState.ToleranceSubstitution), 405
ToleranceSubstitution.setAlphabet
  (setAlphabet.ToleranceSubstitution), 421
ToleranceSubstitution.setEquiDist
  (setEquiDist.ToleranceSubstitution), 440
ToleranceSubstitution.setQMatrix
  (setQMatrix.ToleranceSubstitution), 478
ToleranceSubstitution.setRate
  (setRate.ToleranceSubstitution), 485
ToleranceSubstitution.setRateList
  (setRateList.ToleranceSubstitution), 488
ToleranceSubstitution.summary
  (summary.ToleranceSubstitution), 574
Translate (Undocumented.PhyloSim), 586
Translate, CodonSequence-method
  (Translate.CodonSequence), 584
Translate.CodonSequence, 584
translateCodon (Undocumented.PhyloSim), 586
translateCodon, CodonAlphabet-method
  (translateCodon.CodonAlphabet), 585
translateCodon.CodonAlphabet, 585
Undocumented (Undocumented.PhyloSim), 586
Undocumented.PhyloSim-method
  (Undocumented.PhyloSim), 586
Undocumented.PhyloSim, 586
UNREST, 72, 115, 117, 132, 289, 300, 324, 330,
  333, 372, 389, 393, 575, 576, 579, 586
UNREST.checkConsistency
  (checkConsistency.UNREST), 72
UNREST.summary (summary.UNREST), 575
virtualAssignmentForbidden
  (Undocumented.PhyloSim), 586
virtualAssignmentForbidden, PSRoot-method
  (virtualAssignmentForbidden.PSRoot), 588
virtualAssignmentForbidden.PSRoot, 588
WAG, 20, 132, 372, 389, 393, 589