Package ‘spuRs’

February 20, 2015

Type Package

Title Functions and Datasets for ‘Introduction to Scientific Programming and Simulation Using R’.

Version 2.0.0

Date 2014-07-21

Author Owen Jones, Robert Maillardet, Andrew Robinson, Olga Borovkova, and Steven Carnie

Maintainer Andrew Robinson <A.Robinson@ms.unimelb.edu.au>

Depends R (>= 2.10), MASS, lattice

Description This package provides functions and datasets from the book ‘Introduction to Scientific Programming and Simulation Using R’.

License GPL-3

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-24 08:22:47

R topics documented:

spuRs-package .......................................................... 2
bisection ................................................................. 3
booking_clerkMC ...................................................... 4
CMCSimulation ......................................................... 5
fitDistances ............................................................ 6
fixedpoint .............................................................. 7
fixedpoint_show ....................................................... 8
kew ................................................................. 9
MCEstimation .......................................................... 9
MCSimulation ......................................................... 10
mean.transectHolder .................................................. 11
mean.trapTransect ..................................................... 12
newtonraphson ....................................................... 13
newtonraphson_show ............................................... 14
Description

This package provides scripts, functions and datasets for the book "Introduction to Scientific Programming and Simulation Using R."

Details

Package: spuRs
Type: Package
Version: 1.0
Date: 2008-08-12
License: TBA
LazyLoad: yes

The datasets are: kew, treeg, trees, ufc.plots, and ufc.

The main functions are: bisection, fixedpoint, fixedpoint\_show, newtonraphson, newtonraphson\_show, prime, primesieve, and vol.m3.

We also provide functions relevant to the seedtrap case study: mean and print functions, an sd function, simulate, fitDistances, and constructors. See Chapters 8 and 21 for more details.

Type ?<object> to learn more about these objects, e.g. ?ufc

Author(s)

Owen Jones, Robert Maillardet, Andrew Robinson, Olga Borovkova, and Steve Carnie. Maintainer:
Andrew Robinson <A.Robinson@ms.unimelb.edu.au>
References
Jones, O.D., R.J. Maillardet, and A.P. Robinson. 2009. Introduction to Scientific Programming and Simulation Using R. Chapman And Hall/CRC.

bisection

A function of the bisection algorithm.

Description
 Applies the bisection algorithm to find x such that \( ftn(x) = x \).

Usage

```
 bisection(ftn, x.l, x.r, tol = 1e-09)
```

Arguments

- `ftn`: the function.
- `x.l`: is the lower starting point.
- `x.r`: is the upper starting point.
- `tol`: distance of successive iterations at which algorithm terminates.

Details
We assume that `ftn` is a function of a single variable.

Value
Returns the value of x at which \( ftn(x) = x \). If the function fails to converge within `max.iter` iterations, returns `NULL`.

References

See Also
newtonraphson, fixedpoint

Examples
```r
ftn5 <- function(x) return(log(x)-exp(-x))
bisection(ftn5, 1, 2, tol = 1e-6)
```
booking_clerkMC

A function to simulate the harassed booking clerk Markov chain.

Description

Simulates the harassed booking clerk Markov chain with given arrival and service rates up to t.end. The state space is (C(t),X(t),Y(t)), where C(t) represents the status of the clerk, X(t) the number of people waiting, and Y(t) the number of calls waiting. C(t) is 0 if clerk is idle, 1 if clerk is serving a person and 2 if clerk is serving a call.

Usage

booking_clerkMC(personArrRate, callArrRate, personServRate, callServRate, t.end)

Arguments

personArrRate the person arrival rate.
callArrRate the call arrival rate.
personServRate the person service rate.
callServRate the call service rate.
t.end the time of the time period to be simulated i.e. (0,t.end).

Details

We assume that all given rates are finite and positive.

Value

Returns the matrix (t.hist, state.hist) containing the realisation of the chain.

References


Examples

booking_clerkMC(3,6,5,8,1)
DESCRIPTION

This function simulates a continuous time finite state space Markov chain with known rate matrix Q, state space 0,1,...,n and initial state i for the time period (0,T). If plotflag is TRUE it also produces a plot.

USAGE

CMCSimulation(Q,i,Tend,plotflag = FALSE)

ARGUMENTS

Q the rate matrix.
i the initial state.
Tend the end of the simulation period (0,T).
plotflag flag indicating if plot needed

DETAILS

We assume that Q is well defined rate matrix.

VALUE

Returns the matrix (statehist,timehist) containing the realisation of the chain for the specified period. The function also produces a plot of the realisation.

REFERENCES


SEE ALSO

MCSimulation

EXAMPLES

Q <- matrix(c(-24,24,0, 2,-14,12, 0,4,-4),
           nrow = 3, ncol = 3, byrow = TRUE)
CMCSimulation(Q,0,1)
fitDistances

Function to fit a model to seed transect distance/count data.

Description
This function uses maximum likelihood to fit a nominated probability density function to the data of a seedtrap transect holder.

Usage
fitDistances(x, family)

Arguments

x an object of class transectHolder
family the nominated distribution, which must be one of those distributions that can be fit by fitdistr of the MASS package.

Value
The function returns the parameter estimates for the nominated family.

References

See Also
fitdistr, trapTransect

Examples
library(MASS)
s1 <- trapTransect(distances = 1:4, seed.counts = c(4, 3, 2, 0))
allTraps <- transectHolder(s1, family="Weibull")
fitDistances(allTraps, "exponential")
fixedpoint

A function of the fixed point algorithm.

Description

Applies the fixed point algorithm to find \( x \) such that \( ftn(x) = x \).

Usage

\[
\text{fixedpoint}(ftn, x0, \text{tol} = 1e-09, \text{max.iter} = 100)
\]

Arguments

- \( ftn \): the function.
- \( x0 \): is the initial guess at the fixed point.
- \( \text{tol} \): distance of successive iterations at which algorithm terminates.
- \( \text{max.iter} \): maximum number of iterations.

Details

We assume that \( ftn \) is a function of a single variable.

Value

Returns the value of \( x \) at which \( ftn(x) = x \). If the function fails to converge within \( \text{max.iter} \) iterations, returns \text{NULL}.

References


See Also

newtonraphson, bisection

Examples

\[
\begin{align*}
ftn1 & \leftarrow \text{function}(x) \text{ return}(\exp(\exp(-x))) \\
\text{fixedpoint}(ftn1, 2, \text{tol} = 1e-6)
\end{align*}
\]
fixedpoint_show

A function of the fixed point algorithm.

Description

Applies the fixed point algorithm to find x such that ftn(x) == x, and plots the process.

Usage

```r
fixedpoint_show(ftn, x0, xmin = x0 - 1, xmax = x0 + 1)
```

Arguments

- `ftn` the function.
- `x0` is the initial guess at the fixed point.
- `xmin` ~~Describe xmin here~~
- `xmax` ~~Describe xmax here~~

Details

We assume that `ftn` is a function of a single variable.

Value

Returns the value of x at which ftn(x) == x. If the function fails to converge within `max.iter` iterations, returns `NULL`.

References


See Also

- `fixedpoint`
303 years of monthly rainfall data from Kew Gardens, London, U.K.

Description

Usage
data(kew)

Format
A wide-format data frame with 303 observations. Each month has its own column.

Source
Data obtained from the U.S. National Climatic Data Centre, Global Historical Climatology Network data base (GHCN-Monthly Version 2) http://www.ncdc.noaa.gov/oa/climate/ghcn-monthly/.

References

Examples
data(kew)

MCEstimation A function to estimate the transition matrix for a discrete time Markov chain.

Description
This function estimates the transition matrix for a discrete time Markov chain with state space 0,1,...,n given a realisation. The chain has n+1 states.

Usage
MCEstimation(statehist,n)

Arguments
statehist the realisation of the chain.
n the highest numbered state.
Details

We assume that the state space is 0,1,...,n. n is assumed known as it cannot be reliably inferred from the realisation.

Value

Returns the empirical transition matrix obtained by calculating the observed frequencies of actual transitions in the realisation.

References


See Also

mcsimulation

Examples

```r
P <- matrix(c(0.5,0.5,0,0,0.7,0.1,0.2,0,0.1,0.1,0.8,0,0.0.7,0.3),
            nrow = 4, ncol = 4, byrow = TRUE)
statehist<-MCSimulation(P, 0, 3000)
MCEstimation(statehist, 3)
```

---

**MCSimulation**  
*A function to simulate a discrete time Markov chain.*

Description

This function simulates a discrete time Markov chain with transition matrix P, state space 0,1,...,n and initial state i for nsteps transitions.

Usage

MCSimulation(P,i,nsteps)

Arguments

- **P**  
  the transition matrix.
- **i**  
  the initial state.
- **nsteps**  
  the number of transitions to be simulated.

Details

We assume that P is well defined transition matrix with rows summing to 1.
mean.transectHolder

Value

Returns the vector statehist containing the realisation of the chain for nsteps transitions.

References


See Also

MCEstimation, CMCSimulation

Examples

```r
P <- matrix(c(0.5,0.5,0,0.7,0.1,0.2,0,0.1,0.1,0.8,0,0.7,0.3),
           nrow = 4, ncol = 4, byrow = TRUE)
MCSimulation(P, 0, 250)
```

mean.transectHolder  

Function to compute the mean dispersal distance along a transect of seed traps.

Description

This function computes the mean dispersal distance along a transect of seed traps.

Usage

```r
## S3 method for class transectHolder
## S3 method for class 'transectHolder'
mean(x, ...)
```

Arguments

- `x`: an object representing a transect of seed traps.
- `...`: further arguments passed to or from other methods.

Value

The mean seed dispersal distance is returned.

References

See Also

`transectHolder`

Examples

```r
transect.1 <- trapTransct(distances = 1:4,
seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransct(distances = 1:3,
seed.counts = c(3, 2, 1))
transect.3 <- trapTransct(distances=(1:5)/2,
seed.counts = c(3, 4, 2, 3, 1))
al1Traps <- transectHolder(transect.1, transect.2, transect.3,
family="Weibull")

mean(al1Traps)
```

mean.trapTransct  
*Function to compute the mean dispersal distance along a transect of seed traps.*

Description

This function computes the mean dispersal distance along a transect of seed traps.

Usage

```r
## S3 method for class trapTransct
## S3 method for class 'trapTransct'
mean(x, ...)
```

Arguments

- `x`       an object representing a transect of seed traps.
- `...`     further arguments passed to or from other methods.

Value

The mean seed dispersal distance is returned.

References


See Also

`trapTransct`
newtonraphson

Examples

```r
s1 <- trap Transect(distances = 1:4, seed.count = c(4, 3, 2, 0))
mean(s1)
```

---

newtonraphson

A function of the Newton-Raphson algorithm.

Description

Applies the Newton-Raphson algorithm to find x such that \( ftn(x)[1] = 0 \).

Usage

```r
newtonraphson(ftn, x0, tol = 1e-09, max.iter = 100)
```

Arguments

- **ftn**: the function.
- **x0**: is the initial guess at the fixed point.
- **tol**: distance of successive iterations at which algorithm terminates.
- **max.iter**: maximum number of iterations.

Value

Returns the value of x at which \( ftn(x)[1] = 0 \). If the function fails to converge within \( \text{max.iter} \) iterations, returns NULL.

References


See Also

- `fixedpoint`
- `bisection`

Examples

```r
ftn4 <- function(x) {
  # returns function value and its derivative at x
  fx <- log(x) - exp(-x)
dfx <- 1/x + exp(-x)
  return(c(fx, dfx))
}
newtonraphson(ftn4, 2, 1e-6)
```
newtonraphson_show  A function of the Newton-Raphson algorithm, plotting the path.

Description

Applies the Newton-Raphson algorithm to find x such that ftn(x)[1] == 0, and plots the trace of the estimate.

Usage

newtonraphson_show(ftn, x0, xmin = x0 - 1, xmax = x0 + 1)

Arguments

- **ftn**: the function.
- **x0**: the initial guess of the fixed point.
- **xmin**: lower limit for plotting.
- **xmax**: upper limit for plotting.

Value

Returns the value of x at which ftn(x)[1] == 0. If the function fails to converge within `max.iter` iterations, returns `NULL`.

References


See Also

- `newtonraphson`

---

prime  Function to assess whether or not an integer is prime.

Description

An inefficient, brute-force algorithm to assess whether or not an integer is prime.

Usage

prime(n)
primesieve

Arguments

  n                 The integer.

Details

  The function assumes that \( n \) is a positive integer.

Value

  The function returns a logical object that is TRUE if the integer is prime.

References


See Also

  primesieve

Examples

  prime(10)
  prime(7)

---

primesieve  Function to identify all the primes in a vector of positive integers.

Description

  This function uses the Sieve of Eratosthenes to find all the primes less than or equal to a given integer.

Usage

  primesieve(sieved, unsieved)

Arguments

  sieved               Identified primes (empty vector for initialization)
  unsieved             Candidate integers

Details

  The function assumes that \( \text{unsieved} \) is a vector of positive integers.
print.transectHolder

Value

Returns a vector of primes sieved (selected) from the input vector.

References


See Also

prime

Examples

primesieve(c(), 2:200)

print.transectHolder  Function to print a transectHolder object usefully.

Description

This function prints the details of a transectHolder object.

Usage

## S3 method for class transectHolder
## S3 method for class 'transectHolder'
print(x, ...)

Arguments

x  An object representing a transect of seed traps.

...  further arguments passed to or from other methods.

Details

The print function simply uses str on the transectHolder object.

Value

This function is called for its side-effect, which is to print the object informatively.

References

print.trapTransect

See Also

transectHolder

Examples

transect.1 <- trapTransact(distances = 1:4,
                      seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransact(distances = 1:3,
                      seed.counts = c(3, 2, 1))
transect.3 <- trapTransact(distances=(1:5)/2,
                      seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                           family="Weibull")
allTraps

print.trapTransact  Function to print a trapTransact object usefully.

Description

This function prints the details of a trapTransact object.

Usage

## S3 method for class trapTransact
## S3 method for class 'trapTransact'
print(x, ...)

Arguments

x  An object representing a transect of seed traps.
...
 further arguments passed to or from other methods.

Details

The print function simply uses str on the trapTransact object.

Value

This function is called for its side-effect, which is to print the object informatively.

References

A function which uses the Fourth order Runge-Kutta method with adaptive step size to solve a system of ODE’s.

Description

This function simulates a discrete time Markov chain with transition matrix \( P \), state space \( 0,1,...,n \) and initial state \( i \) for \( n \) steps transitions.

Usage

\[
\text{RK4adapt}(\text{dydt, } \mathbf{t0}, \mathbf{y0}, \mathbf{t1}, \mathbf{h0} = 1, \text{tol} = 1e-10, \ldots)
\]

Arguments

- \text{dydt} \quad \text{a function giving the gradient of y(t).}
- \mathbf{t0} \quad \text{initial value of t.}
- \mathbf{y0} \quad \text{initial value of y(t).}
- \mathbf{t1} \quad \text{system solved up to time t1.}
- \mathbf{h0} \quad \text{initial step size}
- \text{tol} \quad \text{tolerance for adapting step size.}
- \ldots \quad \text{pass arguments to function dydt.}

Details

We assume that \( P \) is well defined transition matrix with rows summing to 1.

Value

Returns a list with elements \( t \), a vector giving times, and \( y \), a matrix whose rows give the solution at successive times.

References

sd.transectHolder

Function to compute the sd dispersal distance along a transect of seed traps.

Description

This function computes the standard deviation of the dispersal distances along a transect of seed traps.

Usage

sd.transectHolder(transectHolder)

Arguments

transectHolder an object representing a transect of seed traps.

Value

The standard deviation of the seed dispersal distances is returned.

References


See Also

transectHolder
**Examples**

```r
transect.1 <- trapTransect(distances = 1:4,
                           seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3,
                           seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2,
                           seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                           family="Weibull")

mean(allTraps)

sd.transectHolder(allTraps)
```

**simulate.transectHolder**

*Function to simulate a modelled seed rain from a transectHolder*

**Description**

This function simulates a two-dimensional seed rain according to the model stored in a transectHolder object. The angle of the seed location from the parent plant is uniformly distributed on [0, 2 \pi).

**Usage**

```r
## S3 method for class transectHolder
## S3 method for class 'transectHolder'
simulate(object, nsim=1, seed=NULL, ...)
```

**Arguments**

- `object` the transectHolder object for simulation
- `nsim` the number of seeds to simulate.
- `seed` if not NULL, set the seed to this value before simulation.
- `...` additional optional arguments (ignored here).

**Value**

A dataframe with n rows with the following components:

- `distances` seed distances to parent plant
- `angles` seed angles to parent plant, in radians
- `x` x-location of seed
- `y` y-location of seed
References

See Also
transectHolder

Examples
transect.1 <- trapTransect(distances = 1:4, 
                        seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3, 
                        seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2, 
                        seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3, 
                        family="Weibull")

Details
This function constructs a transectHolder object given a collection of trapTransect objects and a nominated probability density function to fit to the seed count profile.

Usage
transectHolder(..., family = "exponential")

Arguments
... one or more trapTransect objects
family the probability density function to fit to the distance count profiles.

Details
This function is a constructor.
The nominated distribution, which must be one of those distributions that can be fit by fitdistr of the MASS package.
Value

A transectHolder object, which is a list comprising

- `transects` a list one or more trapTransect objects,
- `family` the name of the distribution to which the transect data has been fit,
- `parameters` the estimated parameters for that distribution,
- `rng` the corresponding random number generator for simulations.

References


See Also

- `trapTransect`

Examples

```r
transect.1 <- trapTransect(distances = 1:4,
                           seed.counts = c(4, 3, 2, 0))
transect.2 <- trapTransect(distances = 1:3,
                           seed.counts = c(3, 2, 1))
transect.3 <- trapTransect(distances=(1:5)/2,
                           seed.counts = c(3, 4, 2, 3, 1))
allTraps <- transectHolder(transect.1, transect.2, transect.3,
                           family="Weibull")
allTraps
```

---

**Description**

This function constructs a trapTransect object given a vector of trap distances from the parent plant, a vector of trap seed counts corresponding to the trap distances, and a single trap area.

**Usage**

```r
trapTransect(distances, seed.counts, trap.area = 0.0001)
```

**Arguments**

- `distances` A vector of trap distances from the parent plant.
- `seed.counts` A vector of seed counts in each trap.
- `trap.area` The surface area of each trap.
Details

This function is a constructor.

Value

A trapTransect object, which is a list comprising three objects:

- distances: A vector of trap distances from the parent plant.
- seed.counts: A vector of seed counts in each trap.
- trap.area: The surface area of each trap.

References


See Also

mean.trapTransect, print.trapTransect

Examples

```r
s1 <- trapTransect(distances = 1:4, seed.counts = c(4, 3, 2, 0))
s1
mean(s1)
```

data(treeg)

Grand fir tree growth data from northern and central Idaho, USA.

Description

A sample of 66 grand fir (Abies grandis) trees was selected from national forests around northern and central Idaho. The trees were selected to be dominant in their environment, with no visible evidence of crown damage, forks, broken tops, etc. For each tree the habitat type and the national forest from which it came were recorded. We have data from nine national forests and six different habitat types.

Usage

data(treeg)
Format

A data frame with 542 observations on the following 6 variables.

- **tree.ID** Tree number.
- **forest** National forest number.
- **habitat** Habitat code (see Details).
- **dbh.in** Bole diameter at 1.37 m, in inches
- **height.ft** Tree height, in feet.
- **age** Age at which measurement was taken.

Details

For each tree the height, diameter and age were measured (age is measured using tree rings), then the tree was split lengthways, which allows you to determine the height and diameter of the tree at any age. In this instance height and diameter were recorded for the age the tree was felled and then at ten year periods going back in time. The diameter of the tree was measured at a height of 1.37 m (4’6”), which is called *breast height* in forestry. The height refers to the height of the main trunk only.

The habitats corresponding to codes 1 through 5 are: Ts/Pach; Ts/Op; Th/Pach; AG/Pach and PA/Pach. These codes refer to the climax tree species, which is the most shade-tolerant species that can grow on the site, and the dominant understorey plant, respectively. Ts refers to *Thuja plicata* and *Tsuga heterophylla*, Th refers to *Thuja plicata*, AG is *Abies grandis*, PA is *Picea engelmannii* and *Abies lasiocarpa*, Pach is *Pachistima myrsinites*, and Op is the nasty *Oplopanaz horridurn*. Grand fir is considered a major climax species for AG/Pach, a major seral species for Th/Pach and PA/Pach, and a minor seral species for Ts/Pach and Ts/Op. Loosely speaking, a community is *seral* if there is evidence that at least some of the species are temporary, and *climax* if the community is self-regenerating (Daubenmire, 1952).

Source

These data were kindly supplied by Dr Al Stage, Principal Mensurationist (retired), USDA Forest Service Forest Sciences Laboratory, Moscow, ID, USA.

References


Examples

data(treeeg)
trees  

von Guttenberg Norway spruce tree measurement data

Description

These are a subset of the von Guttenberg data, a set of measurements on Norway spruce (*Picea abies* [L.] Karst) in several different locations and site categories.

Usage

data(trees)

Format

A data frame with 1200 observations on the following 3 variables.

ID A factor identifying the tree by location, site, and tree number.
Age The age at which the tree was measured.
Vol The bole volume of the tree, in cubic dm.

Source

These data were kindly provided by Professor Boris Zeide, University of Arkansas, Monticello, AK, USA, and are further documented in Zeide (1993).

References


Examples

data(trees)

ufc  

Upper Flat Creek forest cruise tree data

Description

These are a subset of the tree measurement data from the Upper Flat Creek unit of the University of Idaho Experimental Forest, which was measured in 1991.

Usage

data(ufc)
Format

A data frame with 336 observations on the following 5 variables.

- `plot` plot label
- `tree` tree label
- `species` species kbd with levels DF, GF, WC, WL
- `dbh.cm` tree diameter at 1.37 m. from the ground, measured in centimetres.
- `height.m` tree height measured in metres

Details

The inventory was based on variable radius plots with 6.43 sq. m. per ha. BAF (Basal Area Factor). The forest stand was 121.5 ha. This version of the data omits errors, trees with missing heights, and uncommon species. The four species are Douglas-fir, grand fir, western red cedar, and western larch.

Source

The data are provided courtesy of Harold Osborne and Ross Appelgren of the University of Idaho Experimental Forest.

References


See Also

- `ufcplots`

Examples

```r
data(ufc)
```

Description

These are a subset of the plot measurement data from the Upper Flat Creek unit of the University of Idaho Experimental Forest, which was measured in 1991.

Usage

```r
data(ufc.plots)
```
**vol.m3**

**Format**

A data frame with 144 observations on the following 6 variables.

- **plot** plot label
- **north.n** northerly plot count
- **east.n** easterly plot count
- **north** northerly coordinate
- **east** easterly coordinate
- **vol.m3.ha** total above-ground merchantable volume, in cubic metres per hectare.

**Source**

The data are provided courtesy of Harold Osborne and Ross Appelgren of the University of Idaho Experimental Forest.

**References**


**See Also**

ufc

**Examples**

```r
data(ufc.plots)
```

**vol.m3**

Function to compute the volume of a tree bole assuming a particular shape.

**Description**

This function computes the volume of a tree bole given its basal diameter and length, assuming that the bole is a frustum of a geometric solid.

**Usage**

```r
vol.m3(dbh.cm, height.m, multiplier = 0.5)
```

**Arguments**

- **dbh.cm** basal diameter in cm.
- **height.m** height in m.
- **multiplier** shape, expressed as a multiplier.
Details

Commonly-used shapes are:

- 1/3 conoid
- 1/2 second-degree paraboloid
- 1 cylinder

Value

The volume is returned, in units of cubic metres.

Examples

vol.m3(30, 30)
vol.m3(30, 30, 1)
Index

*Topic **Markov chain estimation**
  MCEstimation, 9
*Topic **Markov chain simulation**
  booking_clerkMC, 4
  CMCSimulation, 5
  MCSimulation, 10
*Topic **Numerical solution of system of ODE’s**
  RK4adapt, 18
*Topic **datasets**
  kew, 9
  treeg, 23
  trees, 25
  ufc, 25
  ufc.plots, 26
*Topic **data**
  transectHolder, 21
  trapTransect, 22
*Topic **distribution**
  simulate.transectHolder, 20
*Topic **manip**
  fitDistances, 6
  prime, 14
  primesieve, 15
  vol.m3, 27
*Topic **optimize**
  bisection, 3
  fixedpoint, 7
  fixedpoint_show, 8
  newtonraphson, 13
  newtonraphson_show, 14
*Topic **package**
  spuRs-package, 2
*Topic **print**
  print.transectHolder, 16
  print.trapTransect, 17
*Topic **univar**
  mean.transectHolder, 11
  mean.trapTransect, 12

sd.transectHolder, 19
bisection, 3, 7, 13
booking_clerkMC, 4
CMCSimulation, 5, 11
fitDistances, 6
fitdistr, 6
fixedpoint, 3, 7, 8, 13
fixedpoint_show, 8
kew, 9
MCEstimation, 9, 11
MCSimulation, 5, 10, 10
mean.transectHolder, 11
mean.trapTransect, 12, 23
newtonraphson, 3, 7, 13, 14
newtonraphson_show, 14
prime, 14, 16
primesieve, 15, 15
print.transectHolder, 16
print.trapTransect, 17, 23
RK4adapt, 18
sd.transectHolder, 19
simulate.transectHolder, 20
spuRs (spuRs-package), 2
spuRs-package, 2
transectHolder, 12, 17, 19, 21, 21
trapTransect, 6, 12, 18, 22, 22
treeg, 23
trees, 25
ufc, 25, 27
ufc.plots, 26, 26
vol.m3, 27