Package ‘ROptEstOld’

April 11, 2019

Version 1.2.0
Date 2019-04-02
Title Optimally Robust Estimation - Old Version
Description Optimally robust estimation using S4 classes and methods. Old version still
needed for current versions of ROptRegTS and RobRex.
Depends R(>= 3.4), methods, distr(>= 2.8.0), distrEx(>= 2.8.0),
        RandVar(>= 1.2.0), evd
ByteCompile yes
License LGPL-3
URL http://robast.r-forge.r-project.org/
Encoding latin1
LastChangedDate {$LastChangedDate: 2019-04-02 21:10:23 +0200 (Di, 02.
        Apr 2019) $}
LastChangedRevision {$LastChangedRevision: 1215 $}
VCS/SVNRevision 1214
NeedsCompilation no
Author Matthias Kohl [aut, cre, cph]
Maintainer Matthias Kohl <Matthias.Kohl@stamats.de>
Repository CRAN
Date/Publication 2019-04-11 06:35:44 UTC

R topics documented:

asBias ................................................................. 4
asBias-class ............................................................. 5
asCov ................................................................. 6
asCov-class ............................................................. 6
asGRisk-class .......................................................... 7
asHampel .......................................................... 8
asHampel-class .......................................................... 9
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GumbelParameter-class</td>
<td>65</td>
</tr>
<tr>
<td>IC</td>
<td>66</td>
</tr>
<tr>
<td>IC-class</td>
<td>67</td>
</tr>
<tr>
<td>InfluenceCurve</td>
<td>68</td>
</tr>
<tr>
<td>InfluenceCurve-class</td>
<td>70</td>
</tr>
<tr>
<td>infoPlot</td>
<td>71</td>
</tr>
<tr>
<td>InfRobModel</td>
<td>72</td>
</tr>
<tr>
<td>InfRobModel-class</td>
<td>73</td>
</tr>
<tr>
<td>ksEstimator</td>
<td>74</td>
</tr>
<tr>
<td>L2ParamFamily</td>
<td>76</td>
</tr>
<tr>
<td>L2ParamFamily-class</td>
<td>77</td>
</tr>
<tr>
<td>leastFavorableRadius</td>
<td>79</td>
</tr>
<tr>
<td>LnormScaleFamily</td>
<td>80</td>
</tr>
<tr>
<td>locMEstimator</td>
<td>81</td>
</tr>
<tr>
<td>lowerCaseRadius</td>
<td>82</td>
</tr>
<tr>
<td>Neighborhood-class</td>
<td>83</td>
</tr>
<tr>
<td>NonSymmetric</td>
<td>84</td>
</tr>
<tr>
<td>NonSymmetric-class</td>
<td>84</td>
</tr>
<tr>
<td>NormLocationFamily</td>
<td>85</td>
</tr>
<tr>
<td>NormLocationScaleFamily</td>
<td>86</td>
</tr>
<tr>
<td>NormScaleFamily</td>
<td>87</td>
</tr>
<tr>
<td>OddSymmetric</td>
<td>88</td>
</tr>
<tr>
<td>OddSymmetric-class</td>
<td>89</td>
</tr>
<tr>
<td>oneStepEstimator</td>
<td>89</td>
</tr>
<tr>
<td>optIC</td>
<td>90</td>
</tr>
<tr>
<td>optRisk</td>
<td>92</td>
</tr>
<tr>
<td>ParamFamily</td>
<td>94</td>
</tr>
<tr>
<td>ParamFamily-class</td>
<td>95</td>
</tr>
<tr>
<td>ParamFamParameter</td>
<td>97</td>
</tr>
<tr>
<td>ParamFamParameter-class</td>
<td>98</td>
</tr>
<tr>
<td>PoisFamily</td>
<td>99</td>
</tr>
<tr>
<td>ProbFamily-class</td>
<td>100</td>
</tr>
<tr>
<td>radiusMinimaxIC</td>
<td>101</td>
</tr>
<tr>
<td>RiskType-class</td>
<td>102</td>
</tr>
<tr>
<td>RobModel-class</td>
<td>103</td>
</tr>
<tr>
<td>ROptEstOldConstants</td>
<td>104</td>
</tr>
<tr>
<td>TotalVarIC</td>
<td>104</td>
</tr>
<tr>
<td>TotalVarIC-class</td>
<td>106</td>
</tr>
<tr>
<td>TotalVarNeighborhood</td>
<td>107</td>
</tr>
<tr>
<td>TotalVarNeighborhood-class</td>
<td>108</td>
</tr>
<tr>
<td>trAsCov</td>
<td>109</td>
</tr>
<tr>
<td>trAsCov-class</td>
<td>110</td>
</tr>
<tr>
<td>trFiCov</td>
<td>111</td>
</tr>
<tr>
<td>trFiCov-class</td>
<td>112</td>
</tr>
<tr>
<td>UncondNeighborhood-class</td>
<td>113</td>
</tr>
<tr>
<td>Index</td>
<td>114</td>
</tr>
</tbody>
</table>
asBias

Generating function for asBias-class

Description

Generates an object of class "asBias".

Usage

asBias()

Value

Object of class "asBias"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

asBias-class

Examples

asBias()

## The function is currently defined as
function(){ new("asBias") }
asBias-class

| asBias-class | Standardized Asymptotic Bias |

**Description**

Class of standardized asymptotic bias; i.e., the neighborhood radius is omitted respectively, set to 1.

**Objects from the Class**

Objects can be created by calls of the form `new("asBias", ...)`. More frequently they are created via the generating function `asBias`.

**Slots**

- **type**: Object of class "character": “asymptotic bias”.

**Extends**

Class "asRisk", directly.
Class "RiskType", by class "asRisk".

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

[asRisk-class, asBias]

**Examples**

`new("asBias")`
asCov-class

Description
Generates an object of class "asCov".

Usage
asCov()

Value
Object of class "asCov"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
asCov-class

Examples
asCov()

## The function is currently defined as
function(){ new("asCov") }
Slots

type: Object of class "character": “asymptotic covariance”.

Extends

Class "asRisk", directly.
Class "RiskType", by class "asRisk".

Methods

No methods defined with class "asCov" in the signature.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

asRisk-class, asCov

Examples

new("asCov")

---

Description

Class of special convex asymptotic risks.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

type: Object of class "character".
Extends

Class "asRisk", directly.
Class "RiskType", by class "asRisk".

Methods

No methods defined with class "asGRisk" in the signature.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

asRisk-class

Description

Generates an object of class "asHampel".

Usage

asHampel(bound = Inf)

Arguments

bound positive real: bias bound

Value

Object of class asHampel

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
References


See Also

ashampel-class

Examples

ashampel()

## The function is currently defined as
function(bound = Inf){ new("ashampel", bound = bound) }

ashampel-class  Asymptotic Hampel risk

Description

Class of asymptotic Hampel risk which is the trace of the asymptotic covariance subject to a given bias bound (bound on gross error sensitivity).

Objects from the Class

Objects can be created by calls of the form new("ashampel", ...). More frequently they are created via the generating function asHampel.

Slots

type: Object of class "character": “trace of asymptotic covariance for given bias bound”.

bound: Object of class "numeric": given positive bias bound.

Extends

Class "asRisk", directly.
Class "RiskType", by class "asRisk".

Methods

bound signature(object = "ashampel"): accessor function for slot bound.

show signature(object = "ashampel")
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
asRisk-class, ashampel

Examples
new("ashampel")

---

asMSE

Generating function for asMSE-class

Description
Generates an object of class "asMSE".

Usage
asMSE()

Value
Object of class "asMSE"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
asMSE-class
Examples

```r
asMSE()
```

```r
## The function is currently defined as function(){ new("asMSE") }```

---

**asMSE-class**  
*Asymptotic mean square error*

**Description**

Class of asymptotic mean square error.

**Objects from the Class**

Objects can be created by calls of the form `new("asMSE", ...)`. More frequently they are created via the generating function `asMSE`.

**Slots**

- **type**: Object of class "character": “asymptotic mean square error”.

**Extends**

Class "asGRisk", directly.
Class "asRisk", by class "asGRisk".
Class "RiskType", by class "asGRisk".

**Methods**

No methods defined with class "asMSE" in the signature.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

`asGRisk-class`, *asMSE*

**Examples**

```r
new("asMSE")
```
Description

Class of asymptotic risks.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

type: Object of class "character".

Extends

Class "RiskType", directly.

Methods

No methods defined with class "asRisk" in the signature.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

RiskType-class
asUnOvShoot

Generating function for asUnOvShoot-class

Description

Generates an object of class "asUnOvShoot".

Usage

asUnOvShoot(width = 1.960)

Arguments

width positive real: half the width of given confidence interval.

Value

Object of class "asUnOvShoot"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

asUnOvShoot-class

Examples

asUnOvShoot()

## The function is currently defined as
function(width = 1.960) { new("asUnOvShoot", width = width) }
asUnOvShoot-class

Asymptotic under-/overshoot probability

Description

Class of asymptotic under-/overshoot probability.

Objects from the Class

Objects can be created by calls of the form `new("asUnOvShoot", ...)`. More frequently they are created via the generating function `asUnOvShoot`.

Slots

type: Object of class "character": “asymptotic under-/overshoot probability”.
width: Object of class "numeric": half the width of given confidence interval.

Extends

Class "asGRisk", directly.
Class "asRisk", by class "asGRisk".
Class "RiskType", by class "asGRisk".

Methods

width signature(object = "asUnOvShoot"): accessor function for slot width.
show signature(object = "asUnOvShoot")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

asGRisk-class

Examples

`new("asUnOvShoot")`
**BinomFamily**

Generating function for Binomial families

**Description**
Generates an object of class "L2ParamFamily" which represents a Binomial family where the probability of success is the parameter of interest.

**Usage**
```r
BinomFamily(size = 1, prob = 0.5, trafo)
```

**Arguments**
- `size` number of trials
- `prob` probability of success
- `trafo` matrix: transformation of the parameter

**Details**
The slots of the corresponding L2 differentiable parameteric family are filled.

**Value**
Object of class "L2ParamFamily"

**Author(s)**
Matthias Kohl <Matthias.Kohl@stamats.de>

**References**

**See Also**
- `L2ParamFamily-class`, `Binom-class`

**Examples**
```r
(B1 <- BinomFamily(size = 25, prob = 0.25))
plot(B1)
FisherInfo(B1)
checkL2deriv(B1)
```
checkIC

Generic Function for Checking ICs

Description

Generic function for checking centering and Fisher consistency of ICs.

Usage

checkIC(IC, L2Fam, ...)

Arguments

IC          object of class "IC"
L2Fam       L2-differentiable family of probability measures.
...         additional parameters

Details

The precisions of the centering and the Fisher consistency are computed.

Value

The maximum deviation from the IC properties is returned.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily-class, IC-class

Examples

IC1 <- new("IC")
checkIC(IC1)
checkL2deriv

Generic function for checking L2-derivatives

Description

Generic function for checking the L2-derivative of an L2-differentiable family of probability measures.

Usage

checkL2deriv(L2Fam, ...)

Arguments

L2Fam       L2-differentiable family of probability measures
...         additional parameters

Details

The precisions of the centering and the Fisher information are computed.

Value

The maximum deviation is returned.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily-class

Examples

F1 <- new("L2ParamFamily")
checkL2deriv(F1)
Generating function for ContIC-class

Description

Generates an object of class "ContIC"; i.e., an influence curves $\eta$ of the form

$$\eta = (A\Lambda - a) \min(1, b/|A\Lambda - a|)$$

with clipping bound $b$, centering constant $a$ and standardizing matrix $A$. $\Lambda$ stands for the L2 derivative of the corresponding L2 differentiable parametric family which can be created via callL2Fam.

Usage

```r
ContIC(name, CallL2Fam = call("L2ParamFamily"),
       Curve = EuclRandVarList(RealRandVariable(Map = c(function(x){x}),
                                  Domain = Reals())),
       Risks, Infos, clip = Inf, cent = 0, stand = as.matrix(1),
       lowerCase = NULL, neighborRadius = 0)
```

Arguments

- **name** object of class "character".
- **CallL2Fam** object of class "call": creates an object of the underlying L2-differentiable parametric family.
- **Curve** object of class "EuclRandVarList"
- **Risks** object of class "list": list of risks; cf. RiskType-class.
- **Infos** matrix of characters with two columns named method and message: additional informations.
- **clip** positive real: clipping bound.
- **cent** real: centering constant
- **stand** matrix: standardizing matrix
- **lowerCase** optional constant for lower case solution.
- **neighborRadius** radius of the corresponding (unconditional) contamination neighborhood.

Value

Object of class "ContIC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
References


See Also

IC-class, ContIC

Examples

IC1 <- ContIC()
plot(IC1)

ContIC-class: Influence curve of contamination type

Description

Class of (partial) influence curves of contamination type; i.e., influence curves \( \eta \) of the form

\[
\eta = (A \Lambda - a) \min(1, b/|A \Lambda - a|)
\]

with clipping bound \( b \), centering constant \( a \) and standardizing matrix \( A \). \( \Lambda \) stands for the L2 derivative of the corresponding L2 differentiable parametric family created via the call in the slot CallL2Fam.

Objects from the Class

Objects can be created by calls of the form new("ContIC", ...). More frequently they are created via the generating function ContIC, respectively via the method generateIC.

Slots

CallL2Fam: object of class "call": creates an object of the underlying L2-differentiable parametric family.
name: object of class "character"
Curve: object of class "EuclRandVarList"
Risks: object of class "list": list of risks; cf. RiskType-class.
Infos: object of class "matrix" with two columns named method and message: additional informations.
clip: object of class "numeric": clipping bound.
cent: object of class "numeric": centering constant.
stand: object of class "matrix": standardizing matrix.
lowerCase: object of class "OptionalNumeric": optional constant for lower case solution.
neighborRadius: object of class "numeric": radius of the corresponding (unconditional) contamination neighborhood.
**ContIC-class**

**Extends**

Class "IC", directly.
Class "InfluenceCurve", by class "IC".

**Methods**

**CallL2Fam** <- signature(object = "ContIC"): replacement function for slot CallL2Fam.
**cent** signature(object = "ContIC"): accessor function for slot cent.
**cent**<- signature(object = "ContIC"): replacement function for slot cent.
**clip** signature(object = "ContIC"): accessor function for slot clip.
**clip**<- signature(object = "ContIC"): replacement function for slot clip.
**stand** signature(object = "ContIC"): accessor function for slot stand.
**stand**<- signature(object = "ContIC"): replacement function for slot stand.
**lowerCase** signature(object = "ContIC"): accessor function for slot lowerCase.
**lowerCase**<- signature(object = "ContIC"): replacement function for slot lowerCase.
**neighborRadius** signature(object = "ContIC"): accessor function for slot neighborRadius.
**neighborRadius**<- signature(object = "ContIC"): replacement function for slot neighborRadius.
**generateIC** signature(neighbor = "ContNeighborhood", L2Fam = "L2ParamFamily"): generate an object of class "ContIC". Rarely called directly.
**show** signature(object = "ContIC")

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

*IC-class, ContIC*

**Examples**

ICI <- new("ContIC")
plot(ICI)
Description

Generates an object of class "ContNeighborhood".

Usage

ContNeighborhood(radius = 0)

Arguments

radius non-negative real: neighborhood radius.

Value

Object of class "ContNeighborhood"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ContNeighborhood-class

Examples

ContNeighborhood()

## The function is currently defined as
function(radius = 0){
    new("ContNeighborhood", radius = radius)
}
ContNeighborhood-class

Contamination Neighborhood

Description

Class of (unconditional) contamination neighborhoods.

Objects from the Class

Objects can be created by calls of the form `new("ContNeighborhood", ...)`. More frequently they are created via the generating function `ContNeighborhood`.

Slots

type: Object of class "character": "(uncond.) convex contamination neighborhood".
radius: Object of class "numeric": neighborhood radius.

Extends

Class "UncondNeighborhood", directly.
Class "Neighborhood", by class "UncondNeighborhood".

Methods

No methods defined with class "ContNeighborhood" in the signature.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`ContNeighborhood, UncondNeighborhood-class`

Examples

`new("ContNeighborhood")`
**evalIC**

Generic function for evaluating ICs

**Description**

Generic function for evaluating ICs.

**Usage**

```r
evalIC(IC, x)
```

**Arguments**

- `IC` object of class "IC"
- `x` numeric vector or matrix

**Details**

The list of random variables contained in the slot `curve` is evaluated at `x`.

**Value**

In case `x` is numeric a vector and in case `x` is matrix a matrix is returned.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

- IC-class
EvenSymmetric-class

Generating function for EvenSymmetric-class

---

### Description
Generates an object of class "EvenSymmetric".

### Usage
```
EvenSymmetric(SymmCenter = 0)
```

### Arguments
- `SymmCenter` numeric: center of symmetry

### Value
Object of class "EvenSymmetric"

### Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

### See Also
- `evenSymmetric-class`, `functionsymmetry-class`

### Examples
```
EvenSymmetric()

## The function is currently defined as
function(SymmCenter = 0){
  new("EvenSymmetric", SymmCenter = SymmCenter)
}
```

---

EvenSymmetric-class

Class for Even Functions

---

### Description
Class for even functions.

### Objects from the Class
Objects can be created by calls of the form `new("EvenSymmetric")`. More frequently they are created via the generating function `EvenSymmetric`. 
slots

- type: Object of class "character": contains "even function"
- SymmCenter: Object of class "numeric": center of symmetry

extends

- Class "FunctionSymmetry", directly.
- Class "Symmetry", by class "FunctionSymmetry".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

EvenSymmetric, FunctionSymmetry-class

Examples

```r
new("EvenSymmetric")
```

---

**ExpScaleFamily**

*Generating function for exponential scale families*

Description

Generates an object of class "L2ParamFamily" which represents an exponential scale family.

Usage

```r
ExpScaleFamily(rate = 1, trafo)
```

Arguments

- `rate` rate
- `trafo` matrix: optional transformation of the parameter

Details

The slots of the corresponding L2 differentiable parameteric family are filled. The scale parameter corresponds to 1/rate.

Value

Object of class "L2ParamFamily"
Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily-class, Exp-class

Examples

```r
(E1 <- ExpScaleFamily())
plot(E1)
Map(L2deriv(E1)[[1]])
checkL2deriv(E1)
```

---

**fiBias**

*Generating function for fiBias-class*

Description

Generates an object of class "fiBias".

Usage

`fiBias()`

Value

Object of class "fiBias"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`fiBias-class`
fiBias-class

Examples

    fiBias()
    
    ## The function is currently defined as
    function(){ new("fiBias") }

fiBias-class  Finite-sample Bias

Description

Class of finite-sample bias.

Objects from the Class

Objects can be created by calls of the form new("fiBias", ...). More frequently they are created via the generating function fiBias.

Slots

    type: Object of class "character": “finite-sample bias”.

Extends

    Class "fiRisk", directly.
    Class "RiskType", by class "fiRisk".

Methods

    No methods defined with class "fiBias" in the signature.

Author(s)

    Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

    fiRisk-class, fiBias

Examples

    new("fiBias")
fiCov-class  Generating function for fiCov-class

Description
Generates an object of class "fiCov".

Usage
ascov()

Value
Object of class "fiCov"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
fiCov-class

Examples
fiCov()

## The function is currently defined as
function(){ new("FiCov") }

fiCov-class  Finite-sample covariance

Description
Class of finite-sample covariance.

Objects from the Class
Objects can be created by calls of the form new("fiCov", ...). More frequently they are created via the generating function fiCov.
Slots

  type: Object of class "character": “finite-sample covariance”.

Extends

  Class "fiRisk", directly.
  Class "RiskType", by class "fiRisk".

Methods

  No methods defined with class "fiCov" in the signature.

Author(s)

  Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

  fiRisk-class, fiCov

Examples

  new("fiCov")

---

fiHampel  Generating function for fiHampel-class

Description

  Generates an object of class "fiHampel".

Usage

  fiHampel(bound = Inf)

Arguments

  bound  positive real: bias bound

Value

  Object of class fiHampel
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
fiHampel-class

Examples
fiHampel()

## The function is currently defined as
function(bound = Inf) { new("fiHampel", bound = bound) }

---

fiHampel-class Finite-sample Hampel risk

Description
Class of finite-sample Hampel risk which is the trace of the finite-sample covariance subject to a given bias bound (bound on gross error sensitivity).

Objects from the Class
Objects can be created by calls of the form new("fiHampel", ...). More frequently they are created via the generating function fiHampel.

Slots
- type: Object of class "character": “trace of finite-sample covariance for given bias bound”.
- bound: Object of class "numeric": given positive bias bound.

Extends
Class "fiRisk", directly.
Class "RiskType", by class "fiRisk".

Methods
bound signature(object = "fiHampel"): accessor function for slot bound.
show signature(object = "fiHampel")
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
fiRisk-class, fiHampel

Examples
new("fiHampel")

fiMSE Generating function for fiMSE-class

Description
Generates an object of class "fiMSE".

Usage
asMSE()

Value
Object of class "fiMSE"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
fiMSE-class

Examples
fiMSE()

## The function is currently defined as
function(){ new("fiMSE") }
fiMSE-class

Finite-sample mean square error

Description

Class of asymptotic mean square error.

Objects from the Class

Objects can be created by calls of the form new("fiMSE", ...). More frequently they are created via the generating function fimse.

Slots

type: Object of class "character": “finite-sample mean square error”.

Extends

Class "fiRisk", directly.
Class "RiskType", by class "fiRisk".

Methods

No methods defined with class "fiMSE" in the signature.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

fiRisk-class, fiMSE

Examples

new("fiMSE")
Description

Class of finite-sample risks.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

type: Object of class "character".

Extends

Class "RiskType", directly.

Methods

No methods defined with class "fiRisk" in the signature.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

RiskType-class

Generating function for fiUnOvShoot-class

Description

Generates an object of class "fiUnOvShoot".

Usage

fiUnOvShoot(width = 1.960)
Arguments

width: positive real: half the width of given confidence interval.

Value

Object of class "fiUn0vShoot"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

fiUn0vShoot-class

Examples

fiUn0vShoot()

## The function is currently defined as
function(width = 1.960){ new("fiUn0vShoot", width = width) }

---

**fiUn0vShoot-class**

* Finite-sample under-/overshoot probability

Description

Class of finite-sample under-/overshoot probability.

Objects from the Class

Objects can be created by calls of the form `new("fiUn0vShoot", ...`). More frequently they are created via the generating function `fiUn0vShoot`.

Slots

type: Object of class "character": “finite-sample under-/overshoot probability”.
width: Object of class "numeric": half the width of given confidence interval.
**FixRobModel**

**Extends**
Class "fiRisk", directly.
Class "RiskType", by class "fiRisk".

**Methods**

width signature(object = "fiUnOvShoot"): accessor function for slot width.
show signature(object = "fiUnOvShoot")

**Author(s)**
Matthias Kohl <Matthias.Kohl@stamats.de>

**References**

**See Also**
fiRisk-class

**Examples**

new("fiUnOvShoot")

---

### FixRobModel

**Generating function for FixRobModel-class**

**Description**
Generates an object of class "FixRobModel".

**Usage**

FixRobModel(center = ParamFamily(), neighbor = ContNeighborhood())

**Arguments**

center object of class "ProbFamily"
neighbor object of class "UncondNeighborhood"
Value
Object of class "FixRobModel"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
FixRobModel-class

Examples
(M1 <- FixRobModel())

## The function is currently defined as
function(center = ParamFamily(), neighbor = ContNeighborhood()){
    new("FixRobModel", center = center, neighbor = neighbor)
}

FixRobModel-class  Robust model with fixed (unconditional) neighborhood

Description
Class of robust models with fixed (unconditional) neighborhoods.

Objects from the Class
Objects can be created by calls of the form new("FixRobModel", ...). More frequently they are created via the generating function FixRobModel.

Slots
center: Object of class "ProbFamily".
neighbor: Object of class "UncondNeighborhood".

Extends
Class "RobModel", directly.
Methods

neighbor<- signature(object = "FixRobModel"): replacement function for slot neighbor<-
show signature(object = "FixRobModel")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ProbFamily-class, UncondNeighborhood-class, FixRobModel

Examples

new("FixRobModel")

Class of Symmetries for Functions

Description

Class of symmetries for functions.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

type: Object of class "character": describes type of symmetry.
SymmCenter: Object of class "OptionalNumeric": center of symmetry.

Extends

Class "Symmetry", directly.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
FunSymmList

Generating function for FunSymmList-class

Description

Generates an object of class "FunSymmList".

Usage

FunSymmList(...)

Arguments

... Objects of class "FunctionSymmetry" which shall form the list of symmetry types.

Value

Object of class "FunSymmList"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

FunSymmList-class

Examples

FunSymmList(NonSymmetric(), EvenSymmetric(SymmCenter = 1),
          OddSymmetric(SymmCenter = 2))

## The function is currently defined as
function (...){
  new("FunSymmList", list(...))
}

See Also

Symmetry-class, OptionalNumeric-class
FunSymmList-class

List of Symmetries for a List of Functions

Description
Create a list of symmetries for a list of functions

Objects from the Class
Objects can be created by calls of the form new("FunSymmList", ...). More frequently they are created via the generating function FunSymmList.

Slots
.Data: Object of class "list". A list of objects of class "FunctionSymmetry".

Extends
Class "list", from data part.
Class "vector", by class "list".

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

See Also
FunctionSymmetry-class

Examples
new("FunSymmList", list(NonSymmetric(), EvenSymmetric(SymmCenter = 1),
                               OddSymmetric(SymmCenter = 2)))

GammaFamily

Generating function for Gamma families

Description
Generates an object of class "L2ParamFamily" which represents a Gamma family.

Usage
GammaFamily(scale = 1, shape = 1, trafo)
Arguments

- **scale**: positive real: scale parameter
- **shape**: positive real: shape parameter
- **trafo**: matrix: transformation of the parameter

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "L2ParamFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

- `L2ParamFamily-class`, `Gammad-class`

Examples

distrExOptions("EupperTruncQuantile" = 1e-15) # problem with q(Gamma)(1) = NaN
(G1 <- GammaFamily())
FisherInfo(G1)
checkL2deriv(G1)
distrExOptions("EupperTruncQuantile" = 0) # default

---

**generateIC**

Generic function for the generation of influence curves

Description

This function is rarely called directly. It is used by other functions to create objects of class "IC".

Usage

generateIC(neighbor, L2Fam, ...)

getAsRisk

Arguments

  neighbor   Object of class "Neighborhood".
  L2Fam     L2-differentiable family of probability measures.
  ...       additional parameters

Value

  Object of class "IC"

Author(s)

  Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

  IC-class, ContIC-class, TotalVarIC-class

getAsRisk

  *Generic Function for Computation of Asymptotic Risks*

Description

  Generic function for the computation of asymptotic risks. This function is rarely called directly. It is used by other functions.

Usage

  getAsRisk(risk, L2deriv, neighbor, ...)

  ## S4 method for signature 'asMSE,UnivariateDistribution,Neighborhood'
  getAsRisk(risk, L2deriv, neighbor, clip, cent, stand, trafo)

  ## S4 method for signature 'asMSE,EuclRandVariable,Neighborhood'
  getAsRisk(risk, L2deriv, neighbor, clip, cent, stand, trafo)

  ## S4 method for signature 'asBias,UnivariateDistribution,ContNeighborhood'
  getAsRisk(risk, L2deriv, neighbor, trafo)

  ## S4 method for signature 'asBias,UnivariateDistribution,TotalVarNeighborhood'
  getAsRisk(risk, L2deriv, neighbor, trafo)
getAsRisk

## S4 method for signature 'asBias,RealRandVariable,ContNeighborhood'
getAsRisk(risk, L2deriv, neighbor, Distr, L2derivDistrSymm, trafo, 
z.start, A.start, maxiter, tol)

## S4 method for signature 'asCov,UnivariateDistribution,ContNeighborhood'
getAsRisk(risk, L2deriv, neighbor, clip, cent, stand)

## S4 method for signature 'asCov,UnivariateDistribution,TotalVarNeighborhood'
getAsRisk(risk, L2deriv, neighbor, clip, cent, stand)

## S4 method for signature 'asCov,RealRandVariable,ContNeighborhood'
getAsRisk(risk, L2deriv, neighbor, Distr, clip, cent, stand)

## S4 method for signature 'trAsCov,UnivariateDistribution,UncondNeighborhood'
getAsRisk(risk, L2deriv, neighbor, clip, cent, stand)

## S4 method for signature 'trAsCov,RealRandVariable,ContNeighborhood'
getAsRisk(risk, L2deriv, neighbor, Distr, clip, cent, stand)

## S4 method for signature
## 'asUnOvShoot,UnivariateDistribution,UncondNeighborhood'
getAsRisk(risk, L2deriv, neighbor, clip, cent, stand, trafo)

### Arguments

- **risk**: object of class "asRisk".
- **L2deriv**: \( L_2 \)-derivative of some \( L_2 \)-differentiable family of probability distributions.
- **neighbor**: object of class "Neighborhood".
- **clip**: optimal clipping bound.
- **cent**: optimal centering constant.
- **stand**: standardizing matrix.
- **trafo**: matrix: transformation of the parameter.
- **Distr**: object of class "Distribution".
- **L2derivDistrSymm**: object of class "DistrSymmList".
- **z.start**: initial value for the centering constant.
- **A.start**: initial value for the standardizing matrix.
- **maxiter**: the maximum number of iterations.
- **tol**: the desired accuracy (convergence tolerance).

### Value

The asymptotic risk is computed.
Methods

risk = "asMSE", L2deriv = "UnivariateDistribution", neighbor = "Neighborhood": computes asymptotic mean square error in methods for function getInfRobIC.

risk = "asMSE", L2deriv = "EuclRandVariable", neighbor = "Neighborhood": computes asymptotic mean square error in methods for function getInfRobIC.

risk = "asBias", L2deriv = "UnivariateDistribution", neighbor = "ContNeighborhood": computes standardized asymptotic bias in methods for function getInfRobIC.

risk = "asBias", L2deriv = "UnivariateDistribution", neighbor = "TotalVarNeighborhood": computes standardized asymptotic bias in methods for function getInfRobIC.

risk = "asBias", L2deriv = "RealRandVariable", neighbor = "ContNeighborhood": computes standardized asymptotic bias in methods for function getInfRobIC.

risk = "asCov", L2deriv = "UnivariateDistribution", neighbor = "ContNeighborhood": computes asymptotic covariance in methods for function getInfRobIC.

risk = "asCov", L2deriv = "UnivariateDistribution", neighbor = "TotalVarNeighborhood": computes asymptotic covariance in methods for function getInfRobIC.

risk = "asCov", L2deriv = "RealRandVariable", neighbor = "ContNeighborhood": computes asymptotic covariance in methods for function getInfRobIC.

risk = "trAsCov", L2deriv = "UnivariateDistribution", neighbor = "UncondNeighborhood": computes trace of asymptotic covariance in methods for function getInfRobIC.

risk = "trAsCov", L2deriv = "RealRandVariable", neighbor = "ContNeighborhood": computes trace of asymptotic covariance in methods for function getInfRobIC.

risk = "asUnOvShoot", L2deriv = "UnivariateDistribution", neighbor = "UncondNeighborhood": computes asymptotic under-/overshoot risk in methods for function getInfRobIC.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

asRisk-class
getFiRisk  

Generic Function for Computation of Finite-Sample Risks

Description

Generic function for the computation of finite-sample risks. This function is rarely called directly. It is used by other functions.

Usage

getFiRisk(risk, Distr, neighbor, ...)

## S4 method for signature 'fiUnOvShoot,Norm,ContNeighborhood'
getFiRisk(risk, Distr, neighbor,
          clip, stand, sampleSize, Algo, cont)

## S4 method for signature 'fiUnOvShoot,Norm,TotalVarNeighborhood'
getFiRisk(risk, Distr, neighbor,
          clip, stand, sampleSize, Algo, cont)

Arguments

risk  object of class "RiskType".
Distr object of class "Distribution".
neighbor object of class "Neighborhood".
... additional parameters.
clip positive real: clipping bound
stand standardizing constant/matrix.
sampleSize integer: sample size.
Algo "A" or "B".
cont "left" or "right".

Details

The computation of the finite-sample under-/overshoot risk is based on FFT. For more details we refer to Section 11.3 of Kohl (2005).

Value

The finite-sample risk is computed.

Methods

risk = "fiUnOvShoot", Distr = "Norm", neighbor = "ContNeighborhood" computes finite-sample under-/overshoot risk in methods for function getFixRobIC.

risk = "fiUnOvShoot", Distr = "Norm", neighbor = "TotalVarNeighborhood" computes finite-sample under-/overshoot risk in methods for function getFixRobIC.
**getFixClip**

**Author(s)**

Matthias Kohl &lt;Matthias.Kohl@stamats.de&gt;

**References**


**See Also**

fiRisk-class

---

### Description

Generic function for the computation of the optimal clipping bound in case of robust models with fixed neighborhoods. This function is rarely called directly. It is used to compute optimally robust ICs.

### Usage

getFixClip(clip, Distr, risk, neighbor, ...)

### Arguments

- **clip**: positive real: clipping bound
- **Distr**: object of class "Distribution".
- **risk**: object of class "RiskType".
- **neighbor**: object of class "Neighborhood".
- ... additional parameters.

### Value

The optimal clipping bound is computed.
Methods

clip = "numeric", Distr = "Norm", risk = "fiUnOvShoot", neighbor = "ContNeighborhood"
optimal clipping bound for finite-sample under-/overshoot risk.

clip = "numeric", Distr = "Norm", risk = "fiUnOvShoot", neighbor = "TotalVarNeighborhood"
optimal clipping bound for finite-sample under-/overshoot risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ContIC-class, TotalVarIC-class

getFixRobIC

Generic Function for the Computation of Optimally Robust ICs

Description

Generic function for the computation of optimally robust ICs in case of robust models with fixed neighborhoods. This function is rarely called directly.

Usage

getFixRobIC(Distr, risk, neighbor, ...)

### S4 method for signature 'Norm,fiUnOvShoot,UncondNeighborhood'
getFixRobIC(Distr, risk, neighbor,
            sampleSize, upper, maxiter, tol, warn, Algo, cont)

Arguments

Distr object of class "Distribution".
risk object of class "RiskType".
neighbor object of class "Neighborhood".
... additional parameters.
sampleSize integer: sample size.
upper upper bound for the optimal clipping bound.
getIneffDiff

maxiter  the maximum number of iterations.
tol      the desired accuracy (convergence tolerance).
warn     logical: print warnings.
Algo     "A" or "B".
cont     "left" or "right".

Value

The optimally robust IC is computed.

Methods

Distr = "Norm", risk = "fiUnOvShoot", neighbor = "UncondNeighborhood" computes the optimally robust influence curve for one-dimensional normal location and finite-sample under-/overshoot risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

FixRobModel-class

getIneffDiff

Generic Function for the Computation of Inefficiency Differences

Description

Generic function for the computation of inefficiency differences. This function is rarely called directly. It is used to compute the radius minimax IC and the least favorable radius.

Usage

getIneffDiff(radius, L2Fam, neighbor, risk, ...)

## S4 method for signature 'numeric,L2ParamFamily,UncondNeighborhood,asMSE'
getIneffDiff(radius, L2Fam, neighbor, risk, loRad, upRad, loRisk, upRisk, z.start = NULL, A.start = NULL, upper.b, MaxIter, eps, warn)
Arguments

radius neighborhood radius.
L2Fam L2-differentiable family of probability measures.
neighbor object of class "Neighborhood".
risk object of class "RiskType".
... additional parameters
loRad the lower end point of the interval to be searched.
upRad the upper end point of the interval to be searched.
loRisk the risk at the lower end point of the interval.
upRisk the risk at the upper end point of the interval.
z.start initial value for the centering constant.
A.start initial value for the standardizing matrix.
upper.b upper bound for the optimal clipping bound.
MaxIter the maximum number of iterations
eps the desired accuracy (convergence tolerance).
warn logical: print warnings.

Value

The inefficiency difference between the left and the right margin of a given radius interval is computed.

Methods

radius = "numeric", L2Fam = "L2ParamFamily", neighbor = "UncondNeighborhood", risk = "asMSE": computes difference of asymptotic MSE–inefficiency for the boundaries of a given radius interval.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

radiusMinimaxIC, leastFavorableRadius
getInfCent

Generic Function for the Computation of the Optimal Centering Constant/Lower Clipping Bound

Description

Generic function for the computation of the optimal centering constant (contamination neighborhoods) respectively, of the optimal lower clipping bound (total variation neighborhood). This function is rarely called directly. It is used to compute optimally robust ICs.

Usage

getInfCent(L2deriv, neighbor, ...)

## S4 method for signature 'UnivariateDistribution,ContNeighborhood'
getInfCent(L2deriv, neighbor, clip, cent, tol.z, symm, trafo)

## S4 method for signature 'UnivariateDistribution,TotalVarNeighborhood'
getInfCent(L2deriv, neighbor, clip, cent, tol.z, symm, trafo)

## S4 method for signature 'RealRandVariable,ContNeighborhood'
getInfCent(L2deriv, neighbor, Distr, z.comp, stand, cent, clip)

Arguments

L2deriv       L2-derivative of some L2-differentiable family of probability measures.
neighbor     object of class "Neighborhood".
...           additional parameters.
Distr         distribution of L2-differentiable family.
clip          optimal clipping bound.
cent          optimal centering constant.
stand         standardizing matrix.
tol.z        the desired accuracy (convergence tolerance).
symm        logical: indicating symmetry of L2deriv.
trafo         matrix: transformation of the parameter.
z.comp       logical vector: indication which components of the centering constant have to be computed.

Value

The optimal centering constant is computed.
Methods

L2deriv = "UnivariateDistribution", neighbor = "ContNeighborhood" computation of optimal centering constant.

L2deriv = "UnivariateDistribution", neighbor = "TotalVarNeighborhood" computation of optimal lower clipping bound.

L2deriv = "RealRandVariable", neighbor = "ContNeighborhood" computation of optimal centering constant.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ContIC-class, TotalVarIC-class

getInfClip

Generic Function for the Computation of the Optimal Clipping Bound

Description

Generic function for the computation of the optimal clipping bound in case of infinitesimal robust models. This function is rarely called directly. It is used to compute optimally robust ICs.

Usage

getInfClip(clip, L2deriv, risk, neighbor, ...)

## S4 method for signature
## 'numeric,UnivariateDistribution,asMSE,ContNeighborhood'
getInfClip(clip, L2deriv, risk, neighbor, cent, symm, trafo)

## S4 method for signature
## 'numeric,UnivariateDistribution,asMSE,TotalVarNeighborhood'
getInfClip(clip, L2deriv, risk, neighbor, cent, symm, trafo)

## S4 method for signature
## 'numeric,UnivariateDistribution,asUnOvShoot,UncondNeighborhood'
getInfClip(clip, L2deriv, risk, neighbor, Distr, stand, cent, trafo)

## S4 method for signature
## 'numeric,UnivariateDistribution,asUnOvShoot,UncondNeighborhood'
getInfClip(clip, L2deriv, risk, neighbor, cent, symm, trafo)
Arguments

- **clip**: positive real: clipping bound
- **L2deriv**: L2-derivative of some L2-differentiable family of probability measures.
- **risk**: object of class "RiskType".
- **neighbor**: object of class "Neighborhood".
- **...**: additional parameters.
- **cent**: optimal centering constant.
- **stand**: standardizing matrix.
- **Distr**: object of class "Distribution".
- **symm**: logical: indicating symmetry of L2deriv.
- **trafo**: matrix: transformation of the parameter.

Value

The optimal clipping bound is computed.

Methods

- **clip = "numeric", L2deriv = "UnivariateDistribution", risk = "asMSE", neighbor = "ContNeighborhood"**
  - optimal clipping bound for asymptotic mean square error.
- **clip = "numeric", L2deriv = "UnivariateDistribution", risk = "asMSE", neighbor = "TotalVarNeighborhood"**
  - optimal clipping bound for asymptotic mean square error.
- **clip = "numeric", L2deriv = "EuclRandVariable", risk = "asMSE", neighbor = "ContNeighborhood"**
  - optimal clipping bound for asymptotic mean square error.
- **clip = "numeric", L2deriv = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "UncondNeighborhood"**
  - optimal clipping bound for asymptotic under-/overshoot risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

- `ContIC-class`
- `TotalVarIC-class`
getInfGamma  

Generic Function for the Computation of the Optimal Clipping Bound

Description

Generic function for the computation of the optimal clipping bound. This function is rarely called directly. It is called by getInfClip to compute optimally robust ICs.

Usage

getInfGamma(L2deriv, risk, neighbor, ...)  
## S4 method for signature 'UnivariateDistribution,asMSE,ContNeighborhood'
getInfGamma(L2deriv, risk, neighbor, cent, clip)  
## S4 method for signature  
## 'UnivariateDistribution,asGRisk,TotalVarNeighborhood'
getInfGamma(L2deriv, risk, neighbor, cent, clip)  
## S4 method for signature 'RealRandVariable,asMSE,ContNeighborhood'
getInfGamma(L2deriv, risk, neighbor, Distr, stand, cent, clip)  
## S4 method for signature  
## 'UnivariateDistribution,asUnOVShoot,ContNeighborhood'
getInfGamma(L2deriv, risk, neighbor, cent, clip)

Arguments

L2deriv  
L2-derivative of some L2-differentiable family of probability measures.

risk  
object of class "RiskType".

neighbor  
object of class "Neighborhood".

...  
additional parameters

cent  
optimal centering constant.

clip  
optimal clipping bound.

stand  
standardizing matrix.

Distr  
object of class "Distribution".

Details

The function is used in case of asymptotic G-risks; confer Ruckdeschel and Rieder (2004).
Methods

L2deriv = "UnivariateDistribution", risk = "asMSE", neighbor = "ContNeighborhood" used by getInfClip.

L2deriv = "UnivariateDistribution", risk = "asGRisk", neighbor = "TotalVarNeighborhood" used by getInfClip.

L2deriv = "RealRandVariable", risk = "asMSE", neighbor = "ContNeighborhood" used by getInfClip.

L2deriv = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "ContNeighborhood" used by getInfClip.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

asGRisk-class, asMSE-class, asUnOvShoot-class, ContIC-class, TotalVarIC-class

Description

Generic function for the computation of optimally robust ICs in case of infinitesimal robust models. This function is rarely called directly.

Usage

getInfRobIC(L2deriv, risk, neighbor, ...)

## S4 method for signature 'UnivariateDistribution,asCov,ContNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, Finfo, trafo)

## S4 method for signature 'UnivariateDistribution,asCov,TotalVarNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, Finfo, trafo)

## S4 method for signature 'RealRandVariable,asCov,ContNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, Distr, Finfo, trafo)

## S4 method for signature 'UnivariateDistribution,asBias,ContNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, symm, Finfo, trafo, 
upper, maxiter, tol, warn)

## S4 method for signature 'UnivariateDistribution,asBias,TotalVarNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, symm, Finfo, trafo, 
upper, maxiter, tol, warn)

## S4 method for signature 'RealRandVariable,asBias,ContNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, Distr, DistrSymm, L2derivSymm, 
L2derivDistrSymm, Finfo, z.start, A.start, trafo, upper, maxiter, tol, warn)

## S4 method for signature 'UnivariateDistribution,asHampel,UncondNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, symm, Finfo, trafo, 
upper, maxiter, tol, warn)

## S4 method for signature 'RealRandVariable,asHampel,ContNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, Distr, DistrSymm, L2derivSymm, 
L2derivDistrSymm, Finfo, trafo, z.start, A.start, upper, maxiter, tol, warn)

## S4 method for signature 'UnivariateDistribution,asRisk,UncondNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, symm, Finfo, trafo, 
upper, maxiter, tol, warn)

## S4 method for signature 'RealRandVariable,asRisk,ContNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, Distr, DistrSymm, L2derivSymm, 
L2derivDistrSymm, Finfo, trafo, z.start, A.start, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,asUnOvShoot,UncondNeighborhood'
getInfRobIC(L2deriv, risk, neighbor, symm, Finfo, trafo, 
upper, maxiter, tol, warn)

### Arguments

- **L2deriv**: L2-derivative of some L2-differentiable family of probability measures.
- **risk**: object of class "RiskType".
- **neighbor**: object of class "Neighborhood".
- **Distr**: object of class "Distribution".
- **symm**: logical: indicating symmetry of L2deriv.
- **DistrSymm**: object of class "DistributionSymmetry".
- **L2derivSymm**: object of class "FunSymmList".
- **L2derivDistrSymm**: object of class "DistrSymmList".
Finfo Fisher information matrix.
z.start initial value for the centering constant.
A.start initial value for the standardizing matrix.
trafo matrix: transformation of the parameter.
upper upper bound for the optimal clipping bound.
maxiter the maximum number of iterations.
tol the desired accuracy (convergence tolerance).
warn logical: print warnings.

Value

The optimally robust IC is computed.

Methods

L2deriv = "UnivariateDistribution", risk = "asCov", neighbor = "ContNeighborhood" computes the classical optimal influence curve for L2 differentiable parametric families with unknown one-dimensional parameter.

L2deriv = "UnivariateDistribution", risk = "asBias", neighbor = "ContNeighborhood" computes the bias optimal influence curve for L2 differentiable parametric families with unknown one-dimensional parameter.

L2deriv = "RealRandVariable", risk = "asBias", neighbor = "ContNeighborhood" computes the bias optimal influence curve for L2 differentiable parametric families with unknown k-dimensional parameter (k > 1) where the underlying distribution is univariate.

L2deriv = "UnivariateDistribution", risk = "asHampel", neighbor = "UncondNeighborhood" computes the optimally robust influence curve for L2 differentiable parametric families with unknown one-dimensional parameter.

L2deriv = "RealRandVariable", risk = "asHampel", neighbor = "ContNeighborhood" computes the optimally robust influence curve for L2 differentiable parametric families with unknown k-dimensional parameter (k > 1) where the underlying distribution is univariate.

L2deriv = "UnivariateDistribution", risk = "asGRisk", neighbor = "UncondNeighborhood" computes the optimally robust influence curve for L2 differentiable parametric families with unknown one-dimensional parameter.

L2deriv = "RealRandVariable", risk = "asGRisk", neighbor = "ContNeighborhood" computes the optimally robust influence curve for L2 differentiable parametric families with unknown k-dimensional parameter (k > 1) where the underlying distribution is univariate.
getInfStand

L2deriv = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "UncondNeighborhood"
computes the optimally robust influence curve for one-dimensional L2 differentiable paramet-
ric families and asymptotic under-/overshoot risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

sertation.

See Also

InfRobModel-class

getInfStand                      Generic Function for the Computation of the Standardizing Matrix

Description

Generic function for the computation of the standardizing matrix which takes care of the Fisher
consistency of the corresponding IC. This function is rarely called directly. It is used to compute
optimally robust ICs.

Usage

getInfStand(L2deriv, neighbor, ...)

# S4 method for signature 'UnivariateDistribution,ContNeighborhood'
getInfStand(L2deriv, neighbor, clip, cent, trafo)

# S4 method for signature 'UnivariateDistribution,TotalVarNeighborhood'
getInfStand(L2deriv, neighbor, clip, cent, trafo)

# S4 method for signature 'RealRandVariable,ContNeighborhood'
getInfStand(L2deriv, neighbor, Distr, A.comp, stand, clip, cent, trafo)

Arguments

L2deriv       L2-derivative of some L2-differentiable family of probability measures.
neighbor      object of class "Neighborhood"
...           additional parameters
getRiskIC

clip  optimal clipping bound.
cent  optimal centering constant.
stand standardizing matrix.
Distr object of class "Distribution".
trafo matrix: transformation of the parameter.
A.comp matrix: indication which components of the standardizing matrix have to be computed.

Value

The standardizing matrix is computed.

Methods

L2deriv = "UnivariateDistribution", neighbor = "ContNeighborhood" computes standardizing matrix.

L2deriv = "UnivariateDistribution", neighbor = "TotalVarNeighborhood" computes standardizing matrix.

L2deriv = "RealRandVariable", neighbor = "ContNeighborhood" computes standardizing matrix.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ContIC-class, TotalVarIC-class

getRiskIC  Generic function for the computation of a risk for an IC

Description

Generic function for the computation of a risk for an IC.
Usage

getRiskIC(IC, risk, neighbor, L2Fam, ...)

## S4 method for signature 'IC,asCov,missing,missing'
getRiskIC(IC, risk, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,asCov,missing,L2ParamFamily'
getRiskIC(IC, risk, L2Fam, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,transCov,missing,missing'
getRiskIC(IC, risk, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,transCov,missing,L2ParamFamily'
getRiskIC(IC, risk, L2Fam, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,asBias,ContNeighborhood,missing'
getRiskIC(IC, risk, neighbor, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,asBias,ContNeighborhood,L2ParamFamily'
getRiskIC(IC, risk, neighbor, L2Fam, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,asBias,TotalVarNeighborhood,missing'
getRiskIC(IC, risk, neighbor, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,asBias,TotalVarNeighborhood,L2ParamFamily'
getRiskIC(IC, risk, neighbor, L2Fam, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,asMSE,UncondNeighborhood,missing'
getRiskIC(IC, risk, neighbor, tol = .Machine$double.eps^0.25)

## S4 method for signature 'IC,asMSE,UncondNeighborhood,L2ParamFamily'
getRiskIC(IC, risk, neighbor, L2Fam, tol = .Machine$double.eps^0.25)

## S4 method for signature 'TotalVarIC,asUnOvShoot,UncondNeighborhood,missing'
getRiskIC(IC, risk, neighbor)

## S4 method for signature 'IC,fiUnOvShoot,ContNeighborhood,missing'
getRiskIC(IC, risk, neighbor, sampleSize, Algo = "A", cont = "left")

## S4 method for signature 'IC,fiUnOvShoot,TotalVarNeighborhood,missing'
getRiskIC(IC, risk, neighbor, sampleSize, Algo = "A", cont = "left")

Arguments

IC object of class "InfluenceCurve"

risk object of class "RiskType".

neighbor object of class "Neighborhood".
getRiskIC

L2Fam object of class "L2ParamFamily".

... additional parameters
tol the desired accuracy (convergence tolerance).
sampleSize integer: sample size.
Algo "A" or "B".
cont "left" or "right".

Details

To make sure that the results are valid, it is recommended to include an additional check of the IC properties of IC using checkIC.

Value

The risk of an IC is computed.

Methods

IC = "IC", risk = "asCov", neighbor = "missing", L2Fam = "missing" asymptotic covariance of IC.
IC = "IC", risk = "asCov", neighbor = "missing", L2Fam = "L2ParamFamily" asymptotic covariance of IC under L2Fam.
IC = "IC", risk = "trAsCov", neighbor = "missing", L2Fam = "missing" asymptotic covariance of IC.
IC = "IC", risk = "trAsCov", neighbor = "missing", L2Fam = "L2ParamFamily" asymptotic covariance of IC under L2Fam.
IC = "IC", risk = "asBias", neighbor = "ContNeighborhood", L2Fam = "missing" asymptotic bias of IC under convex contaminations.
IC = "IC", risk = "asBias", neighbor = "ContNeighborhood", L2Fam = "L2ParamFamily" asymptotic bias of IC under convex contaminations and L2Fam.
IC = "IC", risk = "asBias", neighbor = "TotalVarNeighborhood", L2Fam = "missing" asymptotic bias of IC in case of total variation neighborhoods.
IC = "IC", risk = "asBias", neighbor = "TotalVarNeighborhood", L2Fam = "L2ParamFamily" asymptotic bias of IC under L2Fam in case of total variation neighborhoods.
IC = "IC", risk = "asMSE", neighbor = "UncondNeighborhood", L2Fam = "missing" asymptotic mean square error of IC.
IC = "IC", risk = "asMSE", neighbor = "UncondNeighborhood", L2Fam = "L2ParamFamily" asymptotic mean square error of IC under L2Fam.
IC = "TotalVarIC", risk = "asUnOvShoot", neighbor = "UncondNeighborhood", L2Fam = "missing" asymptotic under-/overshoot risk of IC.
IC = "IC", risk = "fiUnOvShoot", neighbor = "ContNeighborhood", L2Fam = "missing" finite-sample under-/overshoot risk of IC.
IC = "IC", risk = "fiUnOvShoot", neighbor = "TotalVarNeighborhood", L2Fam = "missing" finite-sample under-/overshoot risk of IC.
Note

This generic function is still under construction.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

generic.riskicMmethods, InfRobModel-class

Gumbel

Generating function for Gumbel-class

Description

Generates an object of class "Gumbel".

Usage

Gumbel(loc = 0, scale = 1)

Arguments

loc real number: location parameter of the Gumbel distribution.
scale positive real number: scale parameter of the Gumbel distribution

Value

Object of class "Gumbel"

Note

The class "Gumbel" is based on the code provided by the package evd.
Gumbel-class

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

See Also
Gumbel-class, rgumbel

Examples

```r
(G1 <- Gumbel(loc = 1, scale = 2))
plot(G1)
loc(G1)
scale(G1)
loc(G1) <- -1
scale(G1) <- 2
plot(G1)

E(Gumbel()) # Euler's constant
E(G1, function(x)(x^2))

## The function is currently defined as
function(loc = 0, scale = 1){
  new("Gumbel", loc = loc, scale = scale)
}
```

---

Gumbel-class  Gumbel distribution

Description

The Gumbel cumulative distribution function with location parameter \( \text{loc} = \mu \) and scale parameter \( \text{scale} = \sigma \) is

\[
F(x) = \exp \left( - \exp \left( - \frac{(x - \mu)}{\sigma} \right) \right)
\]

for all real \( x \), where \( \sigma > 0 \); c.f. rgumbel. This distribution is also known as extreme value distribution of type I; confer Chapter~22 of Johnson et al. (1995).

Usage

```r
E(object, fun, cond, ...)
## S4 method for signature 'Gumbel,missing,missing'
E(object, low = NULL, upp = NULL, ...)
var(x, ...)
## S4 method for signature 'Gumbel'
var(x, ...)
skewness(x, ...)
## S4 method for signature 'Gumbel'
skewness(x, ...)
kurtosis(x, ...)
```
## Gumbel-class

```r
## S4 method for signature 'Gumbel'
kurtosis(x, ...)
```

### Arguments

- **object**: object of class "Distribution"
- **fun**: if missing the (conditional) expectation is computed else the (conditional) expectation of `fun` is computed.
- **cond**: if not missing the conditional expectation given `cond` is computed.
- **low**: lower bound of integration range.
- **upp**: upper bound of integration range.
- **x**: object of class "UnivariateDistribution"
- **...**: additional arguments to `fun`

### Objects from the Class

Objects can be created by calls of the form `new("Gumbel", loc, scale)`. More frequently they are created via the generating function `Gumbel`.

### Slots

- **img**: Object of class "Reals".
- **param**: Object of class "GumbelParameter".
- **r**: `rgumbel`
- **d**: `dgumbel`
- **p**: `pgumbel`
- **q**: `qgumbel`
- **gaps**: (numeric) matrix or NULL
- **.withArith**: logical: used internally to issue warnings as to interpretation of arithmetics
- **.withSim**: logical: used internally to issue warnings as to accuracy
- **.logExact**: logical: used internally to flag the case where there are explicit formulae for the log version of density, cdf, and quantile function
- **.lowerExact**: logical: used internally to flag the case where there are explicit formulae for the lower tail version of cdf and quantile function
- **Symmetry**: object of class "DistributionSymmetry"; used internally to avoid unnecessary calculations.

### Extends

- Class "AbscontDistribution", directly.
- Class "UnivariateDistribution", by class "AbscontDistribution".
- Class "Distribution", by class "AbscontDistribution".
Methods

initialize signature(.Object = "Gumbel"): initialize method.
loc signature(object = "Gumbel"): wrapped access method for slot loc of slot param.
scale signature(x = "Gumbel"): wrapped access method for slot scale of slot param.
loc<- signature(object = "Gumbel"): wrapped replace method for slot loc of slot param.
scale<- signature(x = "Gumbel"): wrapped replace method for slot scale of slot param.
+ signature(e1 = "Gumbel", e2 = "numeric"): result again of class "Gumbel"; exact.
* signature(e1 = "Gumbel", e2 = "numeric"): result again of class "Gumbel"; exact.
E signature(object = "Gumbel", fun = "missing", cond = "missing"): exact evaluation of expectation using explicit expressions.
var signature(x = "Gumbel"): exact evaluation of expectation using explicit expressions.
skewness signature(x = "Gumbel"): exact evaluation of expectation using explicit expressions.
kurtosis signature(x = "Gumbel"): exact evaluation of expectation using explicit expressions.
median signature(x = "Gumbel"): exact evaluation of expectation using explicit expressions.
IQR signature(x = "Gumbel"): exact evaluation of expectation using explicit expressions.

Note

This class is based on the code provided by the package evd.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

rgumbel, AbscontDistribution-class

Examples

(G1 <- new("Gumbel", loc = 1, scale = 2))
plot(G1)
loc(G1)
scale(G1)
loc(G1) <- -1
scale(G1) <- 2
plot(G1)
GumbelLocationFamily  Generating function for Gumbel location families

Description
Generates an object of class "L2ParamFamily" which represents a Gumbel location family.

Usage
GumbelLocationFamily(loc = 0, scale = 1, trafo)

Arguments
- loc: location parameter
- scale: scale parameter
- trafo: matrix: transformation of the parameter

Details
The slots of the corresponding L2 differentiable parameteric family are filled.

Value
Object of class "L2ParamFamily"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
L2ParamFamily-class, Gumbel-class

Examples
distrExOptions("ElowerTruncQuantile" = 1e-15) # problem with non-finite function value
(GI <- GumbelLocationFamily())
plot(GI)
Map(L2deriv(GI)$[[1]])
checkL2deriv(GI)
distrExOptions("ElowerTruncQuantile" = 0) # default
**GumbelParameter-class**  
_PARAMETER OF GUMBEL DISTRIBUTIONS_

---

**Description**

The class of the parameter of Gumbel distributions.

**Objects from the Class**

Objects can be created by calls of the form `new("GumbelParameter", ...)`.

**Slots**

- **loc** real number: location parameter of a Gumbel distribution.
- **scale** positive real number: scale parameter of a Gumbel distribution.
- **name** default name is “parameter of a Gumbel distribution”.

**Extends**

Class "Parameter", directly.  
Class "OptionalParameter", by class "Parameter".

**Methods**

- **loc** signature(object = "GumbelParameter"): access method for slot loc.
- **scale** signature(x = "GumbelParameter"): access method for slot scale.
- **loc<-** signature(object = "GumbelParameter"): replace method for slot loc.
- **scale<-** signature(x = "GumbelParameter"): replace method for slot scale.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**See Also**

_Gumbel-class, Parameter-class_

**Examples**

`new("GumbelParameter")`
IC

Generating function for IC-class

Description

Generates an object of class "IC".

Usage

IC(name, Curve = EuclRandVarList(RealRandVariable(Map = list(function(x){x}),
                          Domain = Reals())),
    Risks, Infos, CallL2Fam = call("L2ParamFamily"))

Arguments

name
Object of class "character".

CallL2Fam
object of class "call": creates an object of the underlying L2-differentiable
parametric family.

Curve
object of class "EuclRandVarList".

Risks
object of class "list": list of risks; cf. RiskType-class.

Infos
matrix of characters with two columns named method and message: additional
informations.

Value

Object of class "IC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


tion.

See Also

IC-class
Examples

IC1 <- IC()
plot(IC1)

## The function is currently defined as
IC <- function(name, Curve = EuclRandVarList(RealRandVariable(Map = list(function(x)(x))),
   Domain = Reals()), Risks, Infos, CallL2Fam = call("L2ParamFamily")){
  if(missing(name))
    name <- "square integrable (partial) influence curve"
  if(missing(Risks))
    Risks <- list()
  if(missing(Infos))
    Infos <- matrix(c(character(0), character(0)), ncol=2,
      dimnames=list(character(0), c("method", "message")))
  return(new("IC", name = name, Curve = Curve, Risks = Risks,
     Infos = Infos, CallL2Fam = CallL2Fam))
}

IC-class

Influence curve

Description

Class of (partial) influence curves.

Objects from the Class

Objects can be created by calls of the form new("IC", ...). More frequently they are created via the generating function IC.

Slots

CallL2Fam: Object of class "call": creates an object of the underlying L2-differentiable parametric family.

name: Object of class "character".

Curve: Object of class "EuclRandVarList".

Risks: Object of class "list": list of risks; cf. RiskType-class.

Infos: Object of class "matrix" with two columns named method and message: additional informations.

Extends

Class "InfluenceCurve", directly.
Methods

CallL2Fam  signature(object = "IC"): accessor function for slot CallL2Fam.
CallL2Fam<- signature(object = "IC"): replacement function for slot CallL2Fam.
checkIC signature(IC = "IC", L2Fam = "missing"): check centering and Fisher consistency of IC assuming the L2-differentiable parametric family which can be generated via the slot CallL2Fam of IC.
checkIC signature(IC = "IC", L2Fam = "L2ParamFamily"): check centering and Fisher consistency of IC assuming the L2-differentiable parametric family L2Fam.
evalIC signature(IC = "IC", x = "numeric"): evaluate IC at x.
evalIC signature(IC = "IC", x = "matrix"): evaluate IC at the rows of x.
infoPlot signature(object = "IC"): Plot absolute and relative information of IC.
plot signature(x = "IC")
show signature(object = "IC")

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

InfluenceCurve-class, IC

Examples

IC1 <- new("IC")
plot(IC1)

---

InfluenceCurve Generating function for InfluenceCurve-class

Description

Generates an object of class "InfluenceCurve".
InfluenceCurve

Usage

InfluenceCurve(name, Curve = EuclRandVarList(EuclRandVariable(Domain = Reals())), Risks, Infos)

Arguments

name character string: name of the influence curve
Curve object of class "EuclRandVarList"
Risks list of risks
Infos matrix of characters with two columns named method and message: additional informations

Value

Object of class "InfluenceCurve"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

InfluenceCurve-class

Examples

InfluenceCurve()

## The function is currently defined as
InfluenceCurve <- function(name, Curve = EuclRandVarList(EuclRandVariable(Domain = Reals())), Risks, Infos){
  if(missing(name))
    name <- "influence curve"
  if(missing(Risks))
    Risks <- list()
  if(missing(Infos))
    Infos <- matrix(c(character(0),character(0)), ncol=2,
                     dimnames=List(character(0), c("method", "message")))
  return(new("InfluenceCurve", name = name, Curve = Curve,
              Risks = Risks, Infos = Infos))
}
Description

Class of influence curves (functions).

Objects from the Class

Objects can be created by calls of the form `new("InfluenceCurve", ...)`. More frequently they are created via the generating function `influencecurve`.

Slots

- name: object of class "character"
- Curve: object of class "EuclRandVarList"
- Risks: object of class "list": list of risks; cf. `RiskType-class`
- Infos: object of class "matrix" with two columns named method and message: additional informations.

Methods

- `name` signature(object = "InfluenceCurve"): accessor function for slot name.
- `name<-` signature(object = "InfluenceCurve"): replacement function for slot name.
- Curve signature(object = "InfluenceCurve"): accessor function for slot Curve.
- Map signature(object = "InfluenceCurve"): accessor function for slot Map of slot Curve.
- Domain signature(object = "InfluenceCurve"): accessor function for slot Domain of slot Curve.
- Range signature(object = "InfluenceCurve"): accessor function for slot Range of slot Curve.
- Infos signature(object = "InfluenceCurve"): accessor function for slot Infos.
- Infos<- signature(object = "InfluenceCurve"): replacement function for slot Infos.
- addInfo<- signature(object = "InfluenceCurve"): function to add an information to slot Infos.
- Risks signature(object = "InfluenceCurve"): accessor function for slot Risks.
- Risks<- signature(object = "InfluenceCurve"): replacement function for slot Risks.
- addRisk<- signature(object = "InfluenceCurve"): function to add a risk to slot Risks.
- show signature(object = "InfluenceCurve")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
References


See Also

`influencecurve`, `risktypeMclass`

Examples

```r
new("InfluenceCurve")
```

infoPlot

Plot absolute and relative information

Description

Plot absolute and relative information of influence curves.

Usage

```r
infoPlot(object)
```

Arguments

object object of class "InfluenceCurve"

Details

Absolute information is defined as the square of the length of an IC. The relative information is defined as the absolute information of one component with respect to the absolute information of the whole IC; confer Section 8.1 of Kohl (2005).

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`L2ParamFamily-class, IC-class`
InfRobModel

Examples

N <- NormLocationScaleFamily(mean = 0, sd = 1)
ICI <- optIC(model = N, risk = asCov())
infoPlot(ICI)

InfRobModel Generating function for InfRobModel-class

Description

Generates an object of class "InfRobModel".

Usage

InfRobModel(center = L2ParamFamily(), neighbor = ContNeighborhood())

Arguments

center object of class "ProbFamily"
neighbor object of class "UncondNeighborhood"

Value

Object of class "FixRobModel"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

RobModel-class, FixRobModel-class

Examples

(M1 <- InfRobModel())

## The function is currently defined as
function(center = L2ParamFamily(), neighbor = ContNeighborhood()){
  new("InfRobModel", center = center, neighbor = neighbor)
}
InfRobModel-class

Robust model with infinitesimal (unconditional) neighborhood

Description

Class of robust models with infinitesimal (unconditional) neighborhoods; i.e., the neighborhood is shrinking at a rate of $\sqrt{n}$.

Objects from the Class

Objects can be created by calls of the form `new("InfRobModel", ...`). More frequently they are created via the generating function `InfRobModel`.

Slots

center: Object of class "ProbFamily".
neighbor: Object of class "UncondNeighborhood".

Extends

Class "RobModel", directly.

Methods

`neighbor<-` signature(object = "InfRobModel"): replacement function for slot `neighbor<-`

`show` signature(object = "InfRobModel")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`ProbFamily-class`, `UncondNeighborhood-class`, `InfRobModel`

Examples

`new("InfRobModel")`
ksEstimator

Generic Function for the Computation of the Kolmogorov Minimum Distance Estimator

Description

Generic function for the computation of the Kolmogorov(-Smirnov) minimum distance estimator.

Usage

ksEstimator(x, distribution, ...)  

## S4 method for signature 'numeric,Binom'
ksEstimator(x, distribution, param, eps = .Machine$double.eps^0.5)

## S4 method for signature 'numeric,Pois'
ksEstimator(x, distribution, param, eps = .Machine$double.eps^0.5)

## S4 method for signature 'numeric,Norm'
ksEstimator(x, distribution, param, eps = .Machine$double.eps^0.5)

## S4 method for signature 'numeric,Lnorm'
ksEstimator(x, distribution, param, eps = .Machine$double.eps^0.5)

## S4 method for signature 'numeric,Gumbel'
ksEstimator(x, distribution, param, eps = .Machine$double.eps^0.5)

## S4 method for signature 'numeric,Exp'
ksEstimator(x, distribution, param, eps = .Machine$double.eps^0.5)

## S4 method for signature 'numeric,Gammad'
ksEstimator(x, distribution, param, eps = .Machine$double.eps^0.5)

Arguments

- x: sample
- distribution: object of class "Distribution"
- ...: additional parameters
- param: name of the unknown parameter. If missing all parameters of the corresponding distribution are estimated.
- eps: the desired accuracy (convergence tolerance).

Details

In case of discrete distributions the Kolmogorov distance is computed and the parameters which lead to the minimum distance are returned. In case of absolutely continuous distributions ks.test is called and the parameters which minimize the corresponding test statistic are returned.
ksEstimator

Value

The Kolmogorov minimum distance estimator is computed. Returns a list with components named like the parameters of distribution.

Methods

- \texttt{x = "numeric", distribution = "Binom"} Binomial distributions.
- \texttt{x = "numeric", distribution = "Pois"} Poisson distributions.
- \texttt{x = "numeric", distribution = "Norm"} Normal distributions.
- \texttt{x = "numeric", distribution = "Lnorm"} Lognormal distributions.
- \texttt{x = "numeric", distribution = "Gumbel"} Gumbel distributions.
- \texttt{x = "numeric", distribution = "Exp"} Exponential distributions.
- \texttt{x = "numeric", distribution = "Gamma"} Gamma distributions.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

- \texttt{Distribution-class}

Examples

\begin{verbatim}
x <- rnorm(100, mean = 1, sd = 2)
ksEstimator(x=x, distribution = Norm()) # estimate mean and sd
ksEstimator(x=x, distribution = Norm(mean = 1), param = "sd") # estimate sd
ksEstimator(x=x, distribution = Norm(sd = 2), param = "mean") # estimate mean
mean(x)
median(x)
sd(x)
mad(x)
\end{verbatim}
L2ParamFamily  Generating function for L2ParamFamily-class

Description

Generates an object of class "L2ParamFamily".

Usage

L2ParamFamily(name, distribution = Norm(), distrSymm, 
main = 0, nuisance, trafo, param, props = character(0), 
L2deriv = EuclRandVarList(RealRandVariable(list(function(x) {x}),
  Domain = Reals())), 
L2derivSymm, L2derivDistr, L2derivDistrSymm, FisherInfo)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>character string: name of the family</td>
</tr>
<tr>
<td>distribution</td>
<td>object of class &quot;Distribution&quot;: member of the family</td>
</tr>
<tr>
<td>distrSymm</td>
<td>object of class &quot;DistributionSymmetry&quot;: symmetry of distribution.</td>
</tr>
<tr>
<td>main</td>
<td>numeric vector: main parameter</td>
</tr>
<tr>
<td>nuisance</td>
<td>numeric vector: nuisance parameter</td>
</tr>
<tr>
<td>trafo</td>
<td>matrix: transformation of the parameter</td>
</tr>
<tr>
<td>param</td>
<td>object of class &quot;ParamFamParameter&quot;: parameter of the family</td>
</tr>
<tr>
<td>props</td>
<td>character vector: properties of the family</td>
</tr>
<tr>
<td>L2deriv</td>
<td>object of class &quot;EuclRandVariable&quot;: L2 derivative of the family</td>
</tr>
<tr>
<td>L2derivSymm</td>
<td>object of class &quot;FunSymmList&quot;: symmetry of the maps contained in L2deriv</td>
</tr>
<tr>
<td>L2derivDistr</td>
<td>object of class &quot;UnivarDistrList&quot;: distribution of L2deriv</td>
</tr>
<tr>
<td>L2derivDistrSymm</td>
<td>object of class &quot;DistrSymmList&quot;: symmetry of the distributions contained in L2derivDistr</td>
</tr>
<tr>
<td>FisherInfo</td>
<td>object of class &quot;PosDefSymmMatrix&quot;: Fisher information of the family</td>
</tr>
</tbody>
</table>

Details

If name is missing, the default “L2 differentiable parametric family of probability measures” is used. In case distrSymm is missing it is set to NoSymmetry(). If param is missing, the parameter is created via main, nuisance and trafo as described in ParamFamParameter. In case L2derivSymm is missing, it is filled with an object of class FunSymmList with entries NonSymmetric(). In case L2derivDistr is missing, it is computed via imageDistr. If L2derivDistrSymm is missing, it is set to an object of class DistrSymmList with entries NoSymmetry(). In case FisherInfo is missing, it is computed from L2deriv using E.
Value
Object of class "L2ParamFamily"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
L2ParamFamily-class

Examples
```r
F1 <- L2ParamFamily()
plot(F1)
```

Description
Class of L2 differentiable parametric families.

Objects from the Class
Objects can be created by calls of the form `new("L2ParamFamily", ...). More frequently they are created via the generating function L2ParamFamily.

Slots
name: object of class "character": name of the family.
distribution: object of class "Distribution": member of the family.
distrSymm: object of class "DistributionSymmetry": symmetry of distribution.
param: object of class "ParamFamParameter": parameter of the family.
props: object of class "character": properties of the family.
L2deriv: object of class "EuclRandVariable": L2 derivative of the family.
L2derivSymm: object of class "FunSymmList": symmetry of the maps included in L2deriv.
L2derivDistr: object of class "UnivarDistrList": list which includes the distribution of L2deriv.
L2derivDistrSymm: object of class "DistrSymmList": symmetry of the distributions included in L2derivDistr.
FisherInfo: object of class "PosDefSymmMatrix": Fisher information of the family.
L2ParamFamily-class

Extends

Class "ParamFamily", directly.
Class "ProbFamily", by class "ParamFamily".

Methods

L2deriv signature(object = "L2ParamFamily"): accessor function for L2deriv.
L2derivSymm signature(object = "L2ParamFamily"): accessor function for L2derivSymm.
L2derivDistr signature(object = "L2ParamFamily"): accessor function for L2derivDistr.
L2derivDistrSymm signature(object = "L2ParamFamily"): accessor function for L2derivDistrSymm.
FisherInfo signature(object = "L2ParamFamily"): accessor function for FisherInfo.
checkL2deriv signature(object = "L2ParamFamily"): check centering of L2deriv and compute precision of Fisher information.
E signature(object = "L2ParamFamily", fun = "EuclRandVariable", cond = "missing"): expectation of fun under the distribution of object.
E signature(object = "L2ParamFamily", fun = "EuclRandMatrix", cond = "missing"): expectation of fun under the distribution of object.
E signature(object = "L2ParamFamily", fun = "EuclRandVarList", cond = "missing"): expectation of fun under the distribution of object.
plot signature(x = "L2ParamFamily"): plot of distribution and L2deriv.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily, ParamFamily-class

Examples

F1 <- new("L2ParamFamily")
plot(F1)
Description
Generic function for the computation of least favorable radii.

Usage

\[
\text{leastfavorableradius}(\text{lRfam}, \text{neighbor}, \text{risk}, \ldots)
\]

## S4 method for signature 'L2ParamFamily,UncondNeighborhood,asGRisk'

最少favorableradius(L2Fam, neighbor, risk, rho, upRad = 1, 
z.start = NULL, A.start = NULL, upper = 100, maxiter = 100, 
tol = .Machine$double.eps^0.4, warn = FALSE)

Arguments
- \text{L2Fam} \quad L2-differentiable family of probability measures.
- \text{neighbor} \quad object of class "Neighborhood".
- \text{risk} \quad object of class "RiskType".
- \ldots \quad additional parameters
- \text{upRad} \quad the upper end point of the radius interval to be searched.
- \text{rho} \quad The considered radius interval is: \([r\rho, r/\rho]\) with \(\rho \in (0, 1)\).
- \text{z.start} \quad initial value for the centering constant.
- \text{A.start} \quad initial value for the standardizing matrix.
- \text{upper} \quad upper bound for the optimal clipping bound.
- \text{maxiter} \quad the maximum number of iterations
- \text{tol} \quad the desired accuracy (convergence tolerance).
- \text{warn} \quad logical: print warnings.

Value
The least favorable radius and the corresponding inefficiency are computed.

Methods
\[
\text{L2Fam} = "L2ParamFamily", \text{neighbor} = "UncondNeighborhood", \text{risk} = "asGRisk" \quad \text{computation of the least favorable radius.}
\]

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>
References


See Also

radiusMinimaxIC

Examples

N <- NormLocationFamily(mean=0, sd=1)
leastFavorableRadius(L2Fam=N, neighbor=ContNeighborhood(),
risk=asMSE(), rho=0.5)


Generating function for lognormal scale families

Description

Generates an object of class "L2ParamFamily" which represents a lognormal scale family.

Usage

LnormScaleFamily(meanlog = 0, sdlog = 1, trafo)

Arguments

meanlog mean of the distribution on the log scale
sdlog standard deviation of the distribution on the log scale
trafo matrix: transformation of the parameter

Details

The slots of the corresponding L2 differentiable parametric family are filled.

Value

Object of class "L2ParamFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
locMEstimator

References


See Also

l2ParamFamily-class, lnorm-class

Examples

```r
(L1 <- LnormScaleFamily())
plot(L1)
Map(L2deriv(L1)[[1]])
checkL2deriv(L1)
```

---

### Description

Generic function for the computation of location M estimators.

### Usage

```r
locMEstimator(x, IC, ...)
```

```r
## S4 method for signature 'numeric,InfluenceCurve'
locMEstimator(x, IC, eps = .Machine$double.eps^0.5)
```

### Arguments

- `x`  
  sample

- `IC`  
  object of class "InfluenceCurve"

- `...`  
  additional parameters

- `eps`  
  the desired accuracy (convergence tolerance).

### Value

Returns a list with component

- `loc`  
  M estimator of location

### Methods

- `x = "numeric", IC = "InfluenceCurve"`  
  univariate location.
lowerCaseRadius

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
InfluenceCurve-class

Description
The lower case radius is computed; confer Subsection 2.1.2 in Kohl (2005).

Usage
lowerCaseRadius(L2Fam, neighbor, risk, ...)

Arguments
L2Fam
L2 differentiable parametric family
neighbor
object of class "Neighborhood"
risk
object of class "RiskType"
... additional parameters

Value
lower case radius

Methods
L2Fam = "L2ParamFamily", neighbor = "ContNeighborhood", risk = "asMSE" lower case radius for risk "asMSE" in case of "ContNeighborhood".
L2Fam = "L2ParamFamily", neighbor = "TotalVarNeighborhood", risk = "asMSE" lower case radius for risk "asMSE" in case of "TotalVarNeighborhood".

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>
References

See Also
L2ParamFamily-class, Neighborhood-class

Examples
lowerCaseRadius(BinomFamily(size = 10), ContNeighborhood(), asMSE())
lowerCaseRadius(BinomFamily(size = 10), TotalVarNeighborhood(), asMSE())

Description
Class of neighborhoods of families of probability measures.

Objects from the Class
A virtual Class: No objects may be created from it.

Slots
- type: Object of class "character": type of the neighborhood.
- radius: Object of class "numeric": neighborhood radius.

Methods
- type signature(object = "Neighborhood"): accessor function for slot type.
- radius signature(object = "Neighborhood"): accessor function for slot radius.
- show signature(object = "Neighborhood")

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
ProbFamily-class
**NonSymmetric-class**

### Generating function for NonSymmetric-class

**Description**

Generates an object of class "NonSymmetric".

**Usage**

NonSymmetric()

**Value**

Object of class "NonSymmetric"

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**See Also**

NonSymmetric-class, FunctionSymmetry-class

**Examples**

NonSymmetric()

```r
## The function is currently defined as
function(){ new("NonSymmetric") }
```

---

**NonSymmetric-class**

**Class for Non-symmetric Functions**

### Description

Class for non-symmetric functions.

### Objects from the Class

Objects can be created by calls of the form `new("NonSymmetric")`. More frequently they are created via the generating function `NonSymmetric`.

### Slots

- `type`: Object of class "character": contains "non-symmetric function"
- `SymmCenter`: Object of class "NULL"
Extends

Class "FunctionSymmetry", directly.
Class "Symmetry", by class "FunctionSymmetry".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

NonSymmetric

Examples

new("NonSymmetric")

NormLocationFamily  Generating function for normal location families

Description

Generates an object of class "L2ParamFamily" which represents a normal location family.

Usage

NormLocationFamily(mean = 0, sd = 1, trafo)

Arguments

mean  mean
sd    standard deviation
trafo matrix: transformation of the parameter

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "L2ParamFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
References


See Also

\texttt{L2ParamFamily-class, Norm-class}

Examples

\begin{verbatim}
(N1 <- NormLocationFamily())
plot(N1)
L2derivDistr(N1)
\end{verbatim}

\begin{verbatim}
NormLocationScaleFamily
  Generating function for normal location and scale families
\end{verbatim}

Description

Generates an object of class "L2ParamFamily" which represents a normal location and scale family.

Usage

\begin{verbatim}
NormLocationScaleFamily(mean = 0, sd = 1, trafo)
\end{verbatim}

Arguments

\begin{verbatim}
mean
sd
trafo
\end{verbatim}

standard deviation

matrix: transformation of the parameter

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "L2ParamFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also

L2ParamFamily-class, Norm-class

Examples

(N1 <- NormLocationScaleFamily())
plot(N1)
FisherInfo(N1)
checkL2deriv(N1)

NormScaleFamily

Generating function for normal scale families

Description

Generates an object of class "L2ParamFamily" which represents a normal scale family.

Usage

NormScaleFamily(sd = 1, mean = 0, trafo)

Arguments

sd standard deviation
mean mean
trafo matrix: transformation of the parameter

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "L2ParamFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily-class, Norm-class
OddSymmetric

Generating function for OddSymmetric-class

Examples

```r
(N1 <- NormScaleFamily())
plot(N1)
FisherInfo(N1)
checkL2deriv(N1)
```

Description

Generates an object of class "OddSymmetric".

Usage

```
OddSymmetric(SymmCenter = 0)
```

Arguments

- **SymmCenter**: numeric; center of symmetry

Value

Object of class "OddSymmetric"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

- OddSymmetric-class, FunctionSymmetry-class

Examples

```r
OddSymmetric()
```

## The function is currently defined as

```r
function(SymmCenter = 0){
    new("OddSymmetric", SymmCenter = SymmCenter)
}
```
OddSymmetric-class

Class for Odd Functions

Description
Class for odd functions.

Objects from the Class
Objects can be created by calls of the form new("OddSymmetric"). More frequently they are created via the generating function OddSymmetric.

Slots
- type: Object of class "character": contains “odd function"
- SymmCenter: Object of class "numeric": center of symmetry

Extends
Class "FunctionSymmetry", directly.
Class "Symmetry", by class "FunctionSymmetry".

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

See Also
OddSymmetric, FunctionSymmetry-class

Examples
new("OddSymmetric")

donestepestimator

Generic function for the computation of one-step estimators

Description
Generic function for the computation of one-step estimators.

Usage
oneStepEstimator(x, IC, start)
Arguments

- **x** sample
- **IC** object of class "InfluenceCurve"
- **start** initial estimate

Details

Given an initial estimation **start**, a sample **x** and an influence curve **IC** the corresponding one-step estimator is computed.

Value

The one-step estimation is computed.

Methods

- **x = "numeric", IC = "InfluenceCurve", start = "numeric"** univariate samples.
- **x = "numeric", IC = "InfluenceCurve", start = "list"** univariate samples.
- **x = "matrix", IC = "InfluenceCurve", start = "numeric"** multivariate samples.
- **x = "matrix", IC = "InfluenceCurve", start = "list"** multivariate samples.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

- **InfluenceCurve-class**

Description

Generic function for the computation of optimally robust ICs.

---

**optIC**

*Generic function for the computation of optimally robust ICs*
optIC

Usage

optIC(model, risk, ...)

## S4 method for signature 'L2ParamFamily,asCov'
optIC(model, risk)

## S4 method for signature 'InfRobModel,asRisk'
optIC(model, risk, z.start = NULL, A.start = NULL, upper = 1e4,
       maxiter = 50, tol = .Machine$double.eps^0.4, warn = TRUE)

## S4 method for signature 'InfRobModel,asUnOvShoot'
optIC(model, risk, upper = 1e4, maxiter = 50,
       tol = .Machine$double.eps^0.4, warn = TRUE)

## S4 method for signature 'FixRobModel,fiUnOvShoot'
optIC(model, risk, sampleSize, upper = 1e4, maxiter = 50,
       tol = .Machine$double.eps^0.4, warn = TRUE, Algo = "A", cont = "left")

Arguments

- **model**: probability model.
- **risk**: object of class "RiskType".
- **...**: additional parameters.
- **z.start**: initial value for the centering constant.
- **A.start**: initial value for the standardizing matrix.
- **upper**: upper bound for the optimal clipping bound.
- **maxiter**: the maximum number of iterations.
- **tol**: the desired accuracy (convergence tolerance).
- **warn**: logical: print warnings.
- **sampleSize**: integer: sample size.
- **Algo**: "A" or "B".
- **cont**: "left" or "right".

Details

In case of the finite-sample risk "fiUnOvShoot" one can choose between two algorithms for the computation of this risk where the least favorable contamination is assumed to be left or right of some bound. For more details we refer to Section 11.3 of Kohl (2005).

Value

Some optimally robust IC is computed.
Methods

model = "L2ParamFamily", risk = "asCov" computes classical optimal influence curve for L2 differentiable parametric families.

model = "InfRobModel", risk = "asRisk" computes optimally robust influence curve for robust models with infinitesimal neighborhoods and various asymptotic risks.

model = "InfRobModel", risk = "asUnOvShoot" computes optimally robust influence curve for robust models with infinitesimal neighborhoods and asymptotic under-/overshoot risk.

model = "FixRobModel", risk = "fiUnOvShoot" computes optimally robust influence curve for robust models with fixed neighborhoods and finite-sample under-/overshoot risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

InfluenceCurve-class, RiskType-class

Examples

B <- BinomFamily(size = 25, prob = 0.25)

## classical optimal IC
IC0 <- optIC(model = B, risk = asCov())
plot(IC0) # plot IC
checkIC(IC0, B)

---

OptRisk

Generic function for the computation of the minimal risk

Description

Generic function for the computation of the optimal (i.e., minimal) risk for a probability model.
Usage

optRisk(model, risk, ...)

## S4 method for signature 'InfRobModel,asRisk'
optRisk(model, risk, z.start = NULL, A.start = NULL, upper = 1e4,
         maxiter = 50, tol = .Machine$double.eps^0.4, warn = TRUE)

## S4 method for signature 'FixRobModel,fiUnOvShoot'
optRisk(model, risk, samplesize, upper = 1e4, maxiter = 50,
         tol = .Machine$double.eps^0.4, warn = TRUE, Algo = "A", cont = "left")

Arguments

model       probability model
risk        object of class RiskType
...         additional parameters
z.start     initial value for the centering constant.
A.start     initial value for the standardizing matrix.
upper       upper bound for the optimal clipping bound.
maxiter     the maximum number of iterations
tol         the desired accuracy (convergence tolerance).
warn        logical: print warnings.
samplesize  integer: sample size.
Algo        "A" or "B".
cont        "left" or "right".

Details

In case of the finite-sample risk "fiUnOvShoot" one can choose between two algorithms for the computation of this risk where the least favorable contamination is assumed to be left or right of some bound. For more details we refer to Section 11.3 of Kohl (2005).

Value

The minimal risk is computed.

Methods

model = "L2ParamFamily", risk = "asCov"  asymptotic covariance of L2 differentiable parameteric family.
model = "InfRobModel", risk = "asRisk"   asymptotic risk of a infinitesimal robust model.
model = "FixRobModel", risk = "fiUnOvShoot" finite-sample under-/overshoot risk of a robust model with fixed neighborhood.
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
RiskType-class

Examples
optRisk(model = NormLocationScaleFamily(), risk = asCov())

Generating function for ParamFamily-class

Description
Generates an object of class "ParamFamily".

Usage
ParamFamily(name, distribution = Norm(), distrSymm, main = 0, nuisance, trafo, param, props = character(0))

Arguments
name character string: name of family
distribution object of class "Distribution": member of the family
distrSymm object of class "DistributionSymmetry": symmetry of distribution.
main numeric vector: main parameter
nuisance numeric vector: nuisance parameter
trafo matrix: transformation of the parameters
param object of class "ParamFamParameter": parameter of the family
props character vector: properties of the family
Details

If `name` is missing, the default ""parametric family of probability measures"" is used. In case `distrSymm` is missing it is set to `NoSymmetry()`. If `param` is missing, the parameter is created via `main`, `nuisance` and `trafo` as described in `ParamFamParameter`.

Value

Object of class "ParamFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

ParamFamily-class

Examples

```r
F1 <- ParamFamily()
plot(F1)

## The function is currently defined as
function(name, distribution = Norm(), main = 0, nuisance, trafo, param, props = character(0)){
  if(missing(name))
    name <- "parametric family of probability measures"
  if(missing(distrSymm)) distrSymm <- NoSymmetry()
  if(missing(param))
    param <- ParamFamParameter(name = paste("parameter of", name),
                              main = main, nuisance = nuisance, trafo = trafo)
  return(new("ParamFamily", name = name, distribution = distribution,
             distrSymm = distrSymm, param = param, props = props))
}
```

ParamFamily-class  Parametric family of probability measures.

Description

Class of parametric families of probability measures.

Objects from the Class

Objects can be created by calls of the form `new("ParamFamily", ...)`. More frequently they are created via the generating function `ParamFamily`.
Slots

param: Object of class "ParamFamParameter": parameter of the family.
name: Object of class "character": name of the family.
distribution: Object of class "Distribution": member of the family.
distrSymm: Object of class "DistributionSymmetry": symmetry of distribution.
props: Object of class "character": properties of the family.

Extends

Class "ProbFamily", directly.

Methods

main signature(object = "ParamFamily"): wrapped accessor function for slot main of slot param.
nuisance signature(object = "ParamFamily"): wrapped accessor function for slot nuisance of slot param.
trafo signature(object = "ParamFamily"): wrapped accessor function for slot trafo of slot param.
param signature(object = "ParamFamily"): accessor function for slot param.
plot signature(x = "ParamFamily"): plot of slot distribution.
show signature(object = "ParamFamily")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

Distribution-class

Examples

F1 <- new("ParamFamily") # prototype
plot(F1)
Generating function for `ParamFamParameter-class`

**Description**

Generates an object of class "ParamFamParameter".

**Usage**

```r
ParamFamParameter(name, main = numeric(0), nuisance, trafo)
```

**Arguments**

- **name**: character string: name of parameter
- **main**: numeric vector: main parameter
- **nuisance**: numeric vector: nuisance parameter
- **trafo**: matrix: transformation of the parameter

**Details**

If `name` is missing, the default ""parameter of a parametric family of probability measures"" is used. If `nuisance` is missing, the nuisance parameter is set to `NULL`. The number of columns of `trafo` have to be equal and the number of rows have to be not larger than the sum of the lengths of `main` and `nuisance`. If `trafo` is missing, no transformation to the parameter is applied; i.e., `trafo` is set to an identity matrix.

**Value**

Object of class "ParamFamParameter"

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**See Also**

`ParamFamParameter-class`

**Examples**

```r
ParamFamParameter(main = 0, nuisance = 1, trafo = diag(c(1,2)))
```

```r
# The function is currently defined as
function(name, main = numeric(0), nuisance, trafo){
  if(missing(name))
    name <- "parameter of a parametric family of probability measures"
  if(missing(nuisance))
    nuisance <- NULL
```
if(missing(trafo))
trafo <- diag(length(main)+length(nuisance))

return(new("ParamFamParameter", name = name, main = main,
nuisance = nuisance, trafo = trafo))

ParamFamParameter-class

Parameter of a parametric family of probability measures

Description

Class of the parameter of parametric families of probability measures.

Objects from the Class

Objects can be created by calls of the form new("ParamFamParameter", ...). More frequently they are created via the generating function ParamFamParameter.

Slots

main: Object of class "numeric": main parameter.
nuisance: Object of class "OptionalNumeric": optional nuisance parameter.
trafo: Object of class "matrix": transformation of the parameter.
name: Object of class "character": name of the parameter.

Extends

Class "Parameter", directly.
Class "OptionalParameter", by class "Parameter".

Methods

main signature(object = "ParamFamParameter"): accessor function for slot main.
main<- signature(object = "ParamFamParameter"): replacement function for slot main.
nuisance signature(object = "ParamFamParameter"): accessor function for slot nuisance.
nuisance<- signature(object = "ParamFamParameter"): replacement function for slot nuisance.
trafo signature(object = "ParamFamParameter"): accessor function for slot trafo.
trafo<- signature(object = "ParamFamParameter"): replacement function for slot trafo.
length signature(x = "ParamFamParameter"): sum of the lengths of main and nuisance.
show signature(object = "ParamFamParameter")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
PoisFamily

See Also

Parameter-class

Examples

new("ParamFamParameter")

-----

PoisFamily Generating function for Poisson families

Description

Generates an object of class "L2ParamFamily" which represents a Poisson family.

Usage

PoisFamily(lambda = 1, trafo)

Arguments

lambda positive mean
trafo matrix: transformation of the parameter

Details

The slots of the corresponding L2 differentiable parameteric family are filled.

Value

Object of class "L2ParamFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily-class, Pois-class

Examples

(P1 <- PoisFamily(lambda = 4.5))
plot(P1)
FisherInfo(P1)
checkL2deriv(P1)
ProbFamily-class  

Family of probability measures

Description

Class of families of probability measures.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

- name: Object of class "character": name of the family.
- distribution: Object of class "Distribution": member of the family.
- distrSymm: Object of class "DistributionSymmetry": symmetry of distribution.
- props: Object of class "character": properties of the family.

Methods

- name signature(object = "ProbFamily"): accessor function for slot name.
- name<- signature(object = "ProbFamily"): replacement function for slot name.
- distribution signature(object = "ProbFamily"): accessor function for slot distribution.
- distrSymm signature(object = "ProbFamily"): accessor function for slot distrSymm.
- props signature(object = "ProbFamily"): accessor function for slot props.
- props<- signature(object = "ProbFamily"): replacement function for slot props.
- addProp<- signature(object = "ProbFamily"): add a property to slot props.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

Distribution-class
Generic function for the computation of the radius minimax IC

Description

Generic function for the computation of the radius minimax IC.

Usage

radiusMinimaxIC(L2Fam, neighbor, risk, ...)

## S4 method for signature 'L2ParamFamily,UncondNeighborhood,asGRisk'
radiusMinimaxIC(L2Fam, neighbor, risk,
    loRad, upRad, z.start = NULL, A.start = NULL, upper = 1e5,
    maxiter = 100, tol = .Machine$double.eps^0.4, warn = FALSE)

Arguments

L2Fam L2-differentiable family of probability measures.
neighbor object of class "Neighborhood".
risk object of class "RiskType".
... additional parameters.
loRad the lower end point of the interval to be searched.
upRad the upper end point of the interval to be searched.
z.start initial value for the centering constant.
A.start initial value for the standardizing matrix.
upper upper bound for the optimal clipping bound.
maxiter the maximum number of iterations
tol the desired accuracy (convergence tolerance).
warn logical: print warnings.

Value

The radius minimax IC is computed.

Methods

L2Fam = "L2ParamFamily", neighbor = "UncondNeighborhood", risk = "asGRisk": computation of the radius minimax IC for an L2 differentiable parametric family.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
RiskType-class

References


See Also

radiusMinimaxIC

Examples

N <- NormLocationFamily(mean=0, sd=1)
radiusMinimaxIC(L2Fam=N, neighbor=ContNeighborhood(),
                   risk=asMSE(), loRad=0.1, upRad=0.5)

RiskType-class  Risk

Description

Class of risks; e.g., estimator risks.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

type: Object of class "character": type of risk.

Methods

type  signature(object = "RiskType"): accessor function for slot type.

show  signature(object = "RiskType")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>
**RobModel-class**

**Description**

Class of robust models. A robust model consists of family of probability measures center and a neighborhood neighbor about this family.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

- **center**: Object of class "ProbFamily"
- **neighbor**: Object of class "Neighborhood"

**Methods**

- **center** signature(object = "RobModel"): accessor function for slot center.
- **center<-** signature(object = "RobModel"): replacement function for slot center.
- **neighbor** signature(object = "RobModel"): accessor function for slot neighbor.
- **neighbor<-** signature(object = "RobModel"): replacement function for slot neighbor.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

*ProbFamily-class, Neighborhood-class*
**Description**

Constants built into `ROptEstOld`.

**Usage**

```r
EULERMASCHERONICONSTANT
APERYCONSTANT
```

**Details**

`ROptEstOld` has a small number of built-in constants.

The following constants are available:

- **EULERMASCHERONICONSTANT**: the Euler Mascheroni constant
  \[ \gamma = -\Gamma'(1) \]
  given in [http://mathworld.wolfram.com/Euler-MascheroniConstant.html](http://mathworld.wolfram.com/Euler-MascheroniConstant.html) (48);

- **APERYCONSTANT**: the Apéry constant
  \[ \zeta(3) = \frac{5}{2} \sum_{k \geq 1} \frac{(-1)^{k-1}}{k^3 \binom{2k}{k}} \]
  as given in [http://mathworld.wolfram.com/AperysConstant.html](http://mathworld.wolfram.com/AperysConstant.html), equation (8);

These are implemented as variables in the `ROptEstOld` name space taking appropriate values.

**Examples**

```r
EULERMASCHERONICONSTANT
APERYCONSTANT
```

---

**TotalVarIC**

*Generating function for TotalVarIC-class*

**Description**

Generates an object of class "TotalVarIC"; i.e., an influence curves \( \eta \) of the form

\[ \eta = c \lor A\Lambda \land d \]

with lower clipping bound \( c \), upper clipping bound \( d \) and standardizing matrix \( A \). \( \Lambda \) stands for the L2 derivative of the corresponding L2 differentiable parametric family which can be created via `callRL2Fam`.
Usage

TotalVarIC(name, CallL2Fam = call("L2ParamFamily"),
  Curve = EuclRandVarList(RealRandVariable(Map = c(function(x) {x}),
    Domain = Reals())),
  Risks, Infos, clipLo = -Inf, clipUp = Inf, stand = as.matrix(1),
  lowerCase = NULL, neighborRadius = 0)

Arguments

name object of class "character".
CallL2Fam object of class "call": creates an object of the underlying L2-differentiable
  parametric family.
Curve object of class "EuclRandVarList".
Risks object of class "list": list of risks; cf. RiskType-class.
Infos matrix of characters with two columns named method and message: additional
  informations.
clipLo negative real: lower clipping bound.
clipUp positive real: lower clipping bound.
stand matrix: standardizing matrix
lowerCase optional constant for lower case solution.
neighborRadius radius of the corresponding (unconditional) contamination neighborhood.

Value

Object of class "TotalVarIC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

 sertation.

See Also

IC-class, ContIC

Examples

IC1 <- TotalVarIC()
plot(IC1)
TotalVarIC-class

Influence curve of total variation type

Description

Class of (partial) influence curves of total variation type. i.e., an influence curves $\eta$ of the form

$$\eta = c \lor AA \land d$$

with lower clipping bound $c$, upper clipping bound $d$ and standardizing matrix $A$. $\Lambda$ stands for the L2 derivative of the corresponding L2 differentiable parametric family which can be created via CallL2Fam.

Objects from the Class

Objects can be created by calls of the form `new("TotalVarIC", ...)`. More frequently they are created via the generating function `TotalVarIC`, respectively via the method `generateIC`.

Slots

CallL2Fam: object of class "call": creates an object of the underlying L2-differentiable parametric family.

name: object of class "character".

Curve: object of class "EuclRandVarList".

Risks: object of class "list": list of risks; cf. RiskType-class.

Infos: object of class "matrix" with two columns named method and message: additional informations.

clipLo: object of class "numeric": lower clipping bound.

clipUp: object of class "numeric": upper clipping bound.

stand: object of class "matrix": standardizing matrix.

lowerCase: object of class "OptionalNumeric": optional constant for lower case solution.

neighborRadius: object of class "numeric": radius of the corresponding (unconditional) contamination neighborhood.

Extends

Class "IC", directly.
Class "InfluenceCurve", by class "IC".
Methods

**CallL2Fam**<- signature(object = "TotalVarIC"): replacement function for slot CallL2Fam.

**clipLo** signature(object = "TotalVarIC"): accessor function for slot clipLo.

**clipLo**<- signature(object = "TotalVarIC"): replacement function for slot clipLo.

**clipUp** signature(object = "TotalVarIC"): accessor function for slot clipUp.

**clipUp**<- signature(object = "TotalVarIC"): replacement function for slot clipUp.

**stand** signature(object = "TotalVarIC"): accessor function for slot stand.

**stand**<- signature(object = "TotalVarIC"): replacement function for slot stand.

**neighborRadius** signature(object = "TotalVarIC"): accessor function for slot neighborRadius.

**neighborRadius**<- signature(object = "TotalVarIC"): replacement function for slot neighborRadius.

**generateIC** signature(neighbor = "TotalVarNeighborhood", L2Fam = "L2ParamFamily"): generate an object of class "TotalVarIC". Rarely called directly.

**show** signature(object = "TotalVarIC")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

**IC-class, ContIC**

Examples

```r
IC1 <- new("TotalVarIC")
plot(IC1)
```

TotalVarNeighborhood  Generating function for TotalVarNeighborhood-class

Description

Generates an object of class "TotalVarNeighborhood".

Usage

TotalVarNeighborhood(radius = 0)
Arguments

radius non-negative real: neighborhood radius.

Value

Object of class "ContNeighborhood"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

TotalVarNeighborhood-class

Examples

TotalVarNeighborhood()

## The function is currently defined as
function(radius = 0){
    new("TotalVarNeighborhood", radius = radius)
}

Description

Class of (unconditional) total variation neighborhoods.

Objects from the Class

Objects can be created by calls of the form new("TotalVarNeighborhood", ...). More frequently they are created via the generating function TotalVarNeighborhood.

Slots

type: Object of class "character": “(uncond.) total variation neighborhood”.
radius: Object of class "numeric": neighborhood radius.
**Extends**

Class "UncondNeighborhood", directly.  
Class "Neighborhood", by class "UncondNeighborhood".

**Methods**

No methods defined with class "TotalVarNeighborhood" in the signature.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

`TotalVarNeighborhood, UncondNeighborhood-class`

**Examples**

```r
new("TotalVarNeighborhood")
```

---

**Description**

Generates an object of class "trAsCov".

**Usage**

`trAsCov()`

**Value**

Object of class "trAsCov"

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>
References

See Also
   - `trAsCov-class`

Examples
   ```r
   trAsCov()
   ```
   ```r
   ## The function is currently defined as
   function(){ new("trAsCov") }
   ```

---

`trAsCov-class`  
Trace of asymptotic covariance

Description
Class of trace of asymptotic covariance.

Objects from the Class
Objects can be created by calls of the form `new("trAsCov", ...)`. More frequently they are created via the generating function `trAsCov`.

Slots
   - `type`: Object of class "character": “trace of asymptotic covariance”.

Extends
   - Class "asRisk", directly.
   - Class "RiskType", by class "asRisk".

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References
See Also

asRisk-class, trAscov

Examples

new(“trAscov”)

trFicov

Generating function for trFicov-class

Description

Generates an object of class "trFicov".

Usage

trFicov()

Value

Object of class "trFicov"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

trFicov-class

Examples

trFicov()

## The function is currently defined as function(){ new("trFicov") }
trFiCov-class  

Trace of finite-sample covariance

Description

Class of trace of finite-sample covariance.

Objects from the Class

Objects can be created by calls of the form `new("trFiCov", ...`). More frequently they are created via the generating function `trFiCov`.

Slots

- `type`: Object of class "character": “trace of finite-sample covariance”.

Extends

Class "fiRisk", directly.
Class "RiskType", by class "fiRisk".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`fiRisk-class, trFiCov`

Examples

`new("trFiCov")`
UncondNeighborhood-class

Unconditional neighborhood

Description
Class of unconditional (errors-in-variables) neighborhoods.

Objects from the Class
A virtual Class: No objects may be created from it.

Slots
- type: Object of class "character": type of the neighborhood.
- radius: Object of class "numeric": neighborhood radius.

Extends
Class "Neighborhood", directly.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
Neighborhood-class
Index

*Topic classes
asBias-class, 5
asCov-class, 6
asRisk-class, 7
asHampel-class, 9
asMSE-class, 11
asRISK-class, 12
asUnOvShoot-class, 14
ContIC-class, 19
ContNeighborhood-class, 22
EvenSymmetric-class, 24
fiBias-class, 27
fiCov-class, 28
fiHampel-class, 30
fiMSE-class, 32
fiRisk-class, 33
fiUnOvShoot-class, 34
FixRobModel-class, 36
FunctionSymmetry-class, 37
FunSymmList-class, 39
IC-class, 67
InfluenceCurve-class, 70
InfRobModel-class, 73
L2ParamFamily-class, 77
Neighborhood-class, 83
NonSymmetric-class, 84
OddSymmetric-class, 89
ParamFamily-class, 95
ParamFamParameter-class, 98
ProbFamily-class, 100
RiskType-class, 102
RobModel-class, 103
TotalVarIC-class, 106
TotalVarNeighborhood-class, 108
trAsCov-class, 110
trFiCov-class, 112
UncondNeighborhood-class, 113

*Topic distribution
Gumbel, 60
Gumbel-class, 61
GumbelParameter-class, 65
ParamFamily, 94

*Topic models
BinomFamily, 15
ContNeighborhood, 21
ContNeighborhood-class, 22
ExpScaleFamily, 25
FixRobModel, 35
FixRobModel-class, 36
GammaFamily, 39
GumbelLocationFamily, 64
GumbelParameter-class, 65
InfRobModel, 72
InfRobModel-class, 73
L2ParamFamily, 76
L2ParamFamily-class, 77
LnormScaleFamily, 80
Neighborhood-class, 83
NormLocationFamily, 85
NormLocationScaleFamily, 86
NormScaleFamily, 87
ParamFamily-class, 95
PoisFamily, 99
ProbFamily-class, 100
RobModel-class, 103
TotalVarNeighborhood, 107
TotalVarNeighborhood-class, 108
UncondNeighborhood-class, 113

*Topic robust
asBias, 4
asCov, 6
asHampel, 8
asMSE, 10
asUnOvShoot, 13
checkIC, 16
checkL2deriv, 17
ContIC, 18
evalIC, 23
### INDEX

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvenSymmetric</td>
<td>24</td>
</tr>
<tr>
<td>fiBias</td>
<td>26</td>
</tr>
<tr>
<td>fiCov</td>
<td>28</td>
</tr>
<tr>
<td>fiHampel</td>
<td>29</td>
</tr>
<tr>
<td>fiMSE</td>
<td>31</td>
</tr>
<tr>
<td>fiUnOvShoot</td>
<td>33</td>
</tr>
<tr>
<td>FunSymmList</td>
<td>38</td>
</tr>
<tr>
<td>generateIC</td>
<td>40</td>
</tr>
<tr>
<td>getAsRisk</td>
<td>41</td>
</tr>
<tr>
<td>getFiRisk</td>
<td>44</td>
</tr>
<tr>
<td>getFixClip</td>
<td>45</td>
</tr>
<tr>
<td>getFixRobIC</td>
<td>46</td>
</tr>
<tr>
<td>getIneffDiff</td>
<td>47</td>
</tr>
<tr>
<td>getInfCent</td>
<td>49</td>
</tr>
<tr>
<td>getInfClip</td>
<td>50</td>
</tr>
<tr>
<td>getInfGamma</td>
<td>52</td>
</tr>
<tr>
<td>getInfRobIC</td>
<td>53</td>
</tr>
<tr>
<td>getInfStand</td>
<td>56</td>
</tr>
<tr>
<td>getRiskIC</td>
<td>57</td>
</tr>
<tr>
<td>IC</td>
<td>66</td>
</tr>
<tr>
<td>InfluenceCurve</td>
<td>68</td>
</tr>
<tr>
<td>infoPlot</td>
<td>71</td>
</tr>
<tr>
<td>kSEstimator</td>
<td>74</td>
</tr>
<tr>
<td>leastFavorableRadius</td>
<td>79</td>
</tr>
<tr>
<td>locMEstimator</td>
<td>81</td>
</tr>
<tr>
<td>lowerCaseRadius</td>
<td>82</td>
</tr>
<tr>
<td>NonSymmetric</td>
<td>84</td>
</tr>
<tr>
<td>OddSymmetric</td>
<td>88</td>
</tr>
<tr>
<td>oneStepEstimator</td>
<td>89</td>
</tr>
<tr>
<td>optIC</td>
<td>90</td>
</tr>
<tr>
<td>optRisk</td>
<td>92</td>
</tr>
<tr>
<td>ParamFamParameter</td>
<td>97</td>
</tr>
<tr>
<td>radiusMinimaxIC</td>
<td>101</td>
</tr>
<tr>
<td>RobModel-class</td>
<td>103</td>
</tr>
<tr>
<td>TotalVarIC</td>
<td>104</td>
</tr>
<tr>
<td>trAsCov</td>
<td>109</td>
</tr>
<tr>
<td>trFiCov</td>
<td>111</td>
</tr>
</tbody>
</table>

* Topic sysdata

**ROptEstOldConstants** | 104

*,Gumbel,numeric-method (Gumbel-class) | 61

+,Gumbel,numeric-method (Gumbel-class) | 61

addInfo<-(InfluenceCurve-class), 70
addInfo<-(InfluenceCurve-method (InfluenceCurve-class)), 70
addProp<-(ProbFamily-class), 100

addRisk<-(InfluenceCurve-class), 70
addRisk<-(InfluenceCurve-method (InfluenceCurve-class)), 70

APERYCON (ROptEstOldConstants), 104

asBias | 4, 5

asBias-class | 5

asCov | 6, 7

asCov-class | 6

asGRisk-class | 7

asHampel | 8, 10

asHampel-class | 9

asMSE | 10, 11

asMSE-class | 11

asRisk-class | 12

asUnOvShoot | 13

asUnOvShoot-class | 14

BinomFamily | 15

bound (asHampel-class), 9

bound, asHampel-method (asHampel-class), 9

bound, fiHampel-method (fiHampel-class), 30

CallL2Fam (IC-class), 67

CallL2Fam, IC-method (IC-class), 67

CallL2Fam<-(IC-class), 67

CallL2Fam<-,ContIC-method (ContIC-class), 19

CallL2Fam<-,IC-method (IC-class), 67

CallL2Fam<-,TotalVarIC-method (TotalVarIC-class), 106

cent (ContIC-class), 19

cent,ContIC-method (ContIC-class), 19

cent<-(ContIC-class), 19

cent<-,ContIC-method (ContIC-class), 19

center (RobModel-class), 103

center,RobModel-method (RobModel-class), 103

center<-(RobModel-class), 103

center<-,RobModel-method (RobModel-class), 103

checkIC | 16

checkIC,IC,L2ParamFamily-method (IC-class), 67

checkIC,IC,missing-method (IC-class), 67

checkL2deriv | 17
checkL2deriv, L2ParamFamily-method
   (L2ParamFamily-class), 77
clip(ContIC-class), 19
clip, ContIC-method (ContIC-class), 19
clip<-(ContIC-class), 19
clip<-, ContIC-method (ContIC-class), 19
clipLo (TotalVarIC-class), 106
clipLo<-(TotalVarIC-class), 106
clipLo<-, TotalVarIC-method
   (TotalVarIC-class), 106
clipUp (TotalVarIC-class), 106
clipUp, TotalVarIC-method
   (TotalVarIC-class), 106
clipUp<-(TotalVarIC-class), 106
clipUp<-, TotalVarIC-method
   (TotalVarIC-class), 106
ContIC, 18, 19, 20, 105, 107
ContIC-class, 19
ContNeighborhood, 21, 22
ContNeighborhood-class, 22
Curve (InfluenceCurve-class), 70
curve, InfluenceCurve-method
   (InfluenceCurve-class), 70
distribution (ProbFamily-class), 100
distribution, ProbFamily-method
   (ProbFamily-class), 100
distrSymm (ProbFamily-class), 100
distrSymm, ProbFamily-method
   (ProbFamily-class), 100
Domain, InfluenceCurve-method
   (InfluenceCurve-class), 70
E (Gumbel-class), 61
E, Gumbel, missing, missing-method
   (Gumbel-class), 61
E, L2ParamFamily, EuclRandMatrix, missing-method
   (L2ParamFamily-class), 77
E, L2ParamFamily, EuclRandVariable, missing-method
   (L2ParamFamily-class), 77
E, L2ParamFamily, EuclRandVarList, missing-method
   (L2ParamFamily-class), 77
E-methods (Gumbel-class), 61
EULERMASCHERONICONSTANT
   (ROptEstOldConstants), 104
evalIC, 23
evalIC, IC, matrix-method (IC-class), 67
evalIC, IC, numeric-method (IC-class), 67
EvenSymmetric, 24, 25
EvenSymmetric-class, 24
ExpScaleFamily, 25
fiBias, 26, 27
fiBias-class, 27
fiCov, 28, 29
fiCov-class, 28
fiHampel, 29, 31
fiHampel-class, 30
fiMSE, 31, 32
fiMSE-class, 32
fiRisk-class, 33
FisherInfo (L2ParamFamily-class), 77
FisherInfo, L2ParamFamily-method
   (L2ParamFamily-class), 77
fiUnOvShoot, 33
fiUnOvShoot-class, 34
FixRobModel, 35, 37
FixRobModel-class, 36
FunctionSymmetry-class, 37
FunSymmList, 38
FunSymmList-class, 39
GammaFamily, 39
generateIC, 40
generateIC, ContNeighborhood, L2ParamFamily-method
   (ContIC-class), 19
generateIC, TotalVarNeighborhood, L2ParamFamily-method
   (TotalVarIC-class), 106
getAsRisk, 41
getAsRisk, asBias, RealRandVariable, ContNeighborhood-method
   (getAsRisk), 41
getAsRisk, asBias, RealRandVariable, ContNeighborhood-method
   (getAsRisk), 41
getAsRisk, asBias, UnivariateDistribution, ContNeighborhood-method
   (getAsRisk), 41
getAsRisk, asBias, UnivariateDistribution, TotalVarNeighborhood-method
   (getAsRisk), 41
getAsRisk, asBias, UnivariateDistribution, TotalVarNeighborhood-method
   (getAsRisk), 41
getAsRisk, asCov, RealRandVariable, ContNeighborhood-method
   (getAsRisk), 41
getAsRisk, asCov, RealRandVariable, ContNeighborhood-method
   (getAsRisk), 41
getAsRisk, asCov, UnivariateDistribution, ContNeighborhood-method
   (getAsRisk), 41
getAsRisk, asCov, UnivariateDistribution, ContNeighborhood-method
   (getAsRisk), 41
getAsRisk, asCov, UnivariateDistribution, TotalVarNeighborhood-method
   (getAsRisk), 41
getAsRisk, asCov, UnivariateDistribution, TotalVarNeighborhood-method
   (getAsRisk), 41
getAsRisk, asSSE, EuclRandVariable, Neighborhood-method
   (getAsRisk), 41
getAsRisk, asSSE, UnivariateDistribution, Neighborhood-method
   (getAsRisk), 41
getRiskIC,IC,asCov,missing,L2ParamFamily-methods

Gumbel-method

(ksEstimator), 74
getRiskIC,IC,asCov,missing,missing-method

(ksEstimator), 74
getRiskIC,IC,asMSE,UncondNeighborhood,L2ParamFamily-methods

numeric,Norm-method

(ksEstimator), 74
getRiskIC,IC,asMSE,UncondNeighborhood,missing-method

(ksEstimator), 74
getRiskIC,IC,fiUnovShoot,ContNeighborhood,missing-method

(ksEstimator), 74
getRiskIC,IC,fiUnovShoot,TotalVarNeighborhood,missing-method

(ksEstimator), 74
getRiskIC,IC,trAsCov,missing,L2ParamFamily-methods

totalvaricLmethods (Gumbel-class), 61
getRiskIC,IC,trAsCov,missing,missing-method

(ksEstimator), 74
getRiskIC,IC,TotalVarIC,asUnOvShoot,UncondNeighborhood

method

L2deriv (L2ParamFamily-class), 77
getRiskIC-methods (getRiskIC), 57
Gumbel, 60
Gumbel-class, 61
GumbelLocationFamily, 64
GumbelParameter-class, 65

IC, 66, 68
IC-class, 67
InfluenceCurve, 68, 71
InfluenceCurve-class, 70
infoPlot, 71
infoPlot,IC-method (IC-class), 67
Infos (InfluenceCurve-class), 70
Infos,InfluenceCurve-method

(InfRobModel), 72, 73
InfRobModel-class, 73
initialize,Gumbel-method

(Gumbel-class), 61

ksEstimator, 74
ksEstimator,numeric,Binom-method

(ksEstimator), 74
ksEstimator,numeric,Exp-method

(ksEstimator), 74
ksEstimator,numeric,Gammad-method

(ksEstimator), 74
kurtosis (Gumbel-class), 61
kurtosis (GumbelMclass), 61
kurtosis (GumbelMmethod (Gumbel-class), 61
L2deriv (L2ParamFamily-class), 77
L2deriv,L2ParamFamily-method

(ksEstimator), 74
L2derivDistr (L2ParamFamily-class), 77
L2derivDistr,L2ParamFamily-method

(ksEstimator), 74
L2derivDistrSymm (L2ParamFamily-class), 77
L2derivDistrSymm,L2ParamFamily-method

(ksEstimator), 74
L2ParamFamily, 76, 78
L2ParamFamily-class, 77
L2derivSymm (L2ParamFamily-class), 77
L2derivSymm,L2ParamFamily-method

(ksEstimator), 74
length,ParamFamily-method

(ParamFamily-class), 98
LnormScaleFamily, 80
loc (GumbelParameter-class), 65
loc,Gumbel-method (Gumbel-class), 61
loc,GumbelParameter-method

(GumbelParameter-class), 65
loc< (GumbelParameter-class), 65
loc<,Gumbel-method (Gumbel-class), 61
loc<,GumbelParameter-method

(GumbelParameter-class), 65
locMEstimator, 81
locMEstimator,numeric,InfluenceCurve-method

(locMEstimator), 81
locMLEstimator-methods (locMLEstimator), 81
lowerCase (ContIC-class), 19
lowerCase,ContIC-method (ContIC-class), 19
lowerCase,TotalVarIC-method (TotalVarIC-class), 106
lowerCase<- (ContIC-class), 19
lowerCase<-,ContIC-method (ContIC-class), 19
lowerCase<-,TotalVarIC-method (TotalVarIC-class), 106
lowerCaseRadius, 82
lowerCaseRadius, L2ParamFamily,ContNeighborhood method (lowerCaseRadius), 82
lowerCaseRadius, L2ParamFamily,TotalVarNeighborhood method (lowerCaseRadius), 82
lowerCaseRadius-methods (lowerCaseRadius), 82
main (ParamFamParameter-class), 98
main,ParamFamily-method (ParamFamily-class), 95
main,ParamFamParameter-method (ParamFamParameter-class), 98
main<-,ParamFamParameter-method (ParamFamParameter-class), 98
Map, InfluenceCurve-method (InfluenceCurve-class), 70
name, InfluenceCurve-method (InfluenceCurve-class), 70
name, ProbFamily-method (ProbFamily-class), 100
name, RobModel-method (RobModel-class), 103
name<-, InfluenceCurve-method (InfluenceCurve-class), 70
name<-, ProbFamily-method (ProbFamily-class), 100
neighbor (RobModel-class), 103
neighbor, RobModel-method (RobModel-class), 103
neighbor<-,RobModel-method (RobModel-class), 103
Neighborhood-class, 83
neighborRadius (ContIC-class), 19
neighborRadius, ContIC-method (ContIC-class), 19
neighborRadius, TotalVarIC-method (TotalVarIC-class), 106
neighborRadius<-,ContIC-class), 19
neighborRadius<-,ContIC-method (ContIC-class), 19
neighborRadius<-,TotalVarIC-method (TotalVarIC-class), 106
neighborRadius<-,TotalVarIC-method (TotalVarIC-class), 106
NonSymmetric-method, 85
NonSymmetric-class, 84
NormLocationScaleFamily, 85
NormScaleFamily, 87
nuisance (ParamFamParameter-class), 98
nuisance, ParamFamily-method (ParamFamily-class), 95
nuisance, ParamFamParameter-method (ParamFamParameter-class), 98
nuisance<-, ParamFamParameter-method (ParamFamParameter-class), 98
OddSymmetric, 88, 89
OddSymmetric-class, 89
oneStepEstimator, 89
oneStepEstimator, matrix, InfluenceCurve, list-method (oneStepEstimator), 89
oneStepEstimator, matrix, InfluenceCurve, numeric-method (oneStepEstimator), 89
oneStepEstimator, numeric, InfluenceCurve, list-method (oneStepEstimator), 89
oneStepEstimator, numeric, InfluenceCurve, numeric-method (oneStepEstimator), 89
oneStepEstimator-methods (oneStepEstimator), 89
optIC, 90
optIC, FixRobModel, fillUnOvShoot-method (optIC), 90
optIC, InfRobModel, asRisk-method (optIC), 90
optIC, InfRobModel, asUnOvShoot-method (optIC), 90
optIC, L2ParamFamily, asCov-method (optIC), 90
optIC-methods (optIC), 90
optRisk, 92
optRisk, FixRobModel, fiUnOvShoot-method (optRisk), 92
optRisk, InfRobModel, asRisk-method (optRisk), 92
optRisk, L2ParamFamily, asCov-method (optRisk), 92
optRisk-methods (optRisk), 92
param, ParamFamily-method (ParamFamily-class), 95
ParamFamily, 94
ParamFamily-class, 95
ParamFamParameter, 76, 95, 97
ParamFamParameter-class, 98
plot, IC, ANV-method (IC-class), 67
plot, L2ParamFamily, ANV-method (L2ParamFamily-class), 77
plot, ParamFamily, ANV-method (ParamFamily-class), 95
PoisFamily, 99
ProbFamily-class, 100
props (ProbFamily-class), 100
props, ProbFamily-method (ProbFamily-class), 100
props<- (ProbFamily-class), 100
props<-, ProbFamily-method (ProbFamily-class), 100
radius (Neighborhood-class), 83
radius, Neighborhood-method (Neighborhood-class), 83
radiusMinimaxIC, 48, 80, 101, 102
radiusMinimaxIC, L2ParamFamily, UncondNeighborhood-method (radiusMinimaxIC), 101
radiusMinimaxIC-methods (radiusMinimaxIC), 101
Range, InfluenceCurve-method (InfluenceCurve-class), 70
rgumbel, 61, 63
Risks (InfluenceCurve-class), 70
Risks, InfluenceCurve-method (InfluenceCurve-class), 70
Risks<- (InfluenceCurve-class), 70
Risks<-, InfluenceCurve-method (InfluenceCurve-class), 70
RiskType-class, 102
RobModel-class, 103
ROptEstOldConstants, 104
scale, Gumbel-method (Gumbel-class), 61
scale, GumbelParameter-method (GumbelParameter-class), 65
scale<-, Gumbel-method (Gumbel-class), 61
scale<-, GumbelParameter-method (GumbelParameter-class), 65
show, asHampel-method (asHampel-class), 9
show, asUnOvShoot-method (asUnOvShoot-class), 14
show, ContIC-method (ContIC-class), 19
show, fiHampel-method (fiHampel-class), 30
show, fiUnOvShoot-method (fiUnOvShoot-class), 34
show, FixRobModel-method (FixRobModel-class), 36
show, IC-method (IC-class), 67
show, InfluenceCurve-method (InfluenceCurve-class), 70
show, InfRobModel-method (InfRobModel-class), 73
show, Neighborhood-method (Neighborhood-class), 83
show, ParamFamily-method (ParamFamily-class), 95
show, ParamFamParameter-method (ParamFamParameter-class), 98
show, RiskType-method (RiskType-class), 102
show, TotalVarIC-method (TotalVarIC-class), 106
skewness (Gumbel-class), 61
skewness-methods (Gumbel-class), 61
stand (ContIC-class), 19
stand, ContIC-method (ContIC-class), 19
stand, TotalVarIC-method (TotalVarIC-class), 106
stand<-, (ContIC-class), 19
stand<-, ContIC-method (ContIC-class), 19
stand<-, TotalVarIC-method (TotalVarIC-class), 106
TotalVarIC, 104
TotalVarIC-class, 106
TotalVarNeighborhood, 107, 109
INDEX

TotalVarNeighborhood-class, 108
trafo(ParamFamParameter-class), 98
trafo,ParamFamily-method
   (ParamFamily-class), 95
trafo,ParamFamParameter-method
   (ParamFamParameter-class), 98
trafo<-(ParamFamParameter-class), 98
trafo<-,ParamFamParameter-method
   (ParamFamParameter-class), 98
traAsCov, 109, 111
traAsCov-class, 110
traFiCov, 111, 112
traFiCov-class, 112
type,Neighborhood-method
   (Neighborhood-class), 83
type,RiskType-method (RiskType-class),
   102

UncondNeighborhood-class, 113

var (Gumbel-class), 61
var,Gumbel-method (Gumbel-class), 61
var-methods (Gumbel-class), 61

width (asUnOvShoot-class), 14
width,asUnOvShoot-method
   (asUnOvShoot-class), 14
width,fiUnOvShoot-method
   (fiUnOvShoot-class), 34