Package ‘ROptRegTS’

February 19, 2015

Version 0.9.1
Date 2013-09-12
Title Optimally robust estimation for regression-type models
Description Optimally robust estimation for regression-type models using S4 classes and methods
Depends R (>= 2.14.0), ROptEstOld(>= 0.9.2)
Imports methods, RandVar(>= 0.9.2), distr(>= 2.5.2), distrEx(>= 2.4)
Author Matthias Kohl <Matthias.Kohl@stamats.de>, Peter Ruckdeschel
Maintainer Matthias Kohl <Matthias.Kohl@stamats.de>
LazyLoad yes
ByteCompile yes
License LGPL-3
Encoding latin1
URL http://robast.r-forge.r-project.org/
LastChangedDate {$LastChangedDate: 2013-09-14 14:15:52 +0200 (Sa, 14. Sep 2013) $}
LastChangedRevision {$LastChangedRevision: 713 $}
SVNRevision 696
NeedsCompilation no
Repository CRAN
Date/Publication 2013-09-14 15:40:34

R topics documented:

Av1CondContIC .................................................. 3
Av1CondContIC-class ......................................... 4
Av1CondContNeighborhood .................................. 6
Av1CondContNeighborhood-class .......................... 7
Av1CondNeighborhood ....................................... 8
The table below lists the R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Av1CondTotalVarIC</td>
<td>9</td>
</tr>
<tr>
<td>Av1CondTotalVarIC-class</td>
<td>10</td>
</tr>
<tr>
<td>Av1CondTotalVarNeighborhood</td>
<td>12</td>
</tr>
<tr>
<td>Av1CondTotalVarNeighborhood-class</td>
<td>13</td>
</tr>
<tr>
<td>Av2CondContIC</td>
<td>14</td>
</tr>
<tr>
<td>Av2CondContIC-class</td>
<td>15</td>
</tr>
<tr>
<td>Av2CondContNeighborhood</td>
<td>17</td>
</tr>
<tr>
<td>Av2CondContNeighborhood-class</td>
<td>18</td>
</tr>
<tr>
<td>Av2CondNeighborhood-class</td>
<td>19</td>
</tr>
<tr>
<td>AvCondNeighborhood-class</td>
<td>20</td>
</tr>
<tr>
<td>CondContIC</td>
<td>21</td>
</tr>
<tr>
<td>CondContIC-class</td>
<td>22</td>
</tr>
<tr>
<td>CondContNeighborhood</td>
<td>24</td>
</tr>
<tr>
<td>CondContNeighborhood-class</td>
<td>25</td>
</tr>
<tr>
<td>CondIC</td>
<td>26</td>
</tr>
<tr>
<td>CondIC-class</td>
<td>27</td>
</tr>
<tr>
<td>CondNeighborhood-class</td>
<td>28</td>
</tr>
<tr>
<td>CondTotalVarIC</td>
<td>29</td>
</tr>
<tr>
<td>CondTotalVarIC-class</td>
<td>31</td>
</tr>
<tr>
<td>CondTotalVarNeighborhood</td>
<td>33</td>
</tr>
<tr>
<td>CondTotalVarNeighborhood-class</td>
<td>34</td>
</tr>
<tr>
<td>FixRobRegTypeModel</td>
<td>35</td>
</tr>
<tr>
<td>FixRobRegTypeModel-class</td>
<td>36</td>
</tr>
<tr>
<td>generateIC-methods</td>
<td>37</td>
</tr>
<tr>
<td>getAsRiskRegTS</td>
<td>37</td>
</tr>
<tr>
<td>getFiRiskRegTS</td>
<td>41</td>
</tr>
<tr>
<td>getFixClipRegTS</td>
<td>43</td>
</tr>
<tr>
<td>getFixRobRegTypeIC</td>
<td>44</td>
</tr>
<tr>
<td>getIneffDiff-methods</td>
<td>46</td>
</tr>
<tr>
<td>getInfCentRegTS</td>
<td>46</td>
</tr>
<tr>
<td>getInfClipRegTS</td>
<td>49</td>
</tr>
<tr>
<td>getInfGammaRegTS</td>
<td>51</td>
</tr>
<tr>
<td>getInfRobRegTypeIC</td>
<td>55</td>
</tr>
<tr>
<td>getInfStandRegTS</td>
<td>61</td>
</tr>
<tr>
<td>InfRobRegTypeModel</td>
<td>64</td>
</tr>
<tr>
<td>InfRobRegTypeModel-class</td>
<td>65</td>
</tr>
<tr>
<td>L2RegTypeFamily</td>
<td>66</td>
</tr>
<tr>
<td>L2RegTypeFamily-class</td>
<td>68</td>
</tr>
<tr>
<td>leastFavorableRadius-methods</td>
<td>70</td>
</tr>
<tr>
<td>NormLinRegFamily</td>
<td>70</td>
</tr>
<tr>
<td>NormLinRegInterceptFamily</td>
<td>71</td>
</tr>
<tr>
<td>NormLinRegScaleFamily</td>
<td>72</td>
</tr>
<tr>
<td>optIC-methods</td>
<td>74</td>
</tr>
<tr>
<td>radiusMinimaxIC-method</td>
<td>75</td>
</tr>
<tr>
<td>RegTypeFamily</td>
<td>76</td>
</tr>
<tr>
<td>RegTypeFamily-class</td>
<td>77</td>
</tr>
</tbody>
</table>

Index 79
Generating function for `Av1CondContIC`-class

**Description**

Generates an object of class "Av1CondContIC"; i.e., an influence curves $\eta$ of the form

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

with clipping bound $b$, centering function $a$ 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
Av1CondContIC(name, CallL2Fam = call("L2RegTypeFamily"),
    Curve = EuclRandVarList(RealRandVariable(
        Map = list(function(x)(x[1]*x[2])),
        Domain = EuclideanSpace(dimension = 2))),
    Risks, Infos, clip = Inf, stand = as.matrix(),
    cent = EuclRandVarList(RealRandVariable(
        Map = list(function(x)(numeric(length(x)))),
        Domain = EuclideanSpace(dimension = 2))),
    lowercase = NULL, neighborRadius = 0)
```

**Arguments**

- `name` object of class "character".
- `CallL2Fam` object of class "call": creates an object of the underlying L2-differentiable regression type 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` object of class "EuclRandVarList": centering function.
- `stand` matrix: standardizing matrix.
- `lowerCase` optional constant for lower case solution.
- `neighborRadius` radius of the corresponding (unconditional) contamination neighborhood.

**Value**

Object of class "Av1CondContIC"

**Author(s)**

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

See Also
CondIC-class, Av1CondContIC-class

Examples
IC1 <- Av1CondContIC()
IC1

Av1CondContIC-class Conditionally centered influence curve of contamination type

Description
Class of conditionally centered (partial) influence curves of contamination type for average conditional contamination neighborhoods; i.e., influence curves $\eta$ of the form
\[
\eta = (A\Lambda - a) \min(1, b/|A\Lambda - a|)
\]
with clipping bound $b$, centering function $a$ and standardizing matrix $A$. $\Lambda$ stands for the L2 derivative of the corresponding L2 differentiable regression type family created via the call in the slot CalllRfam.

Objects from the Class
Objects can be created by calls of the form new("Av1CondContIC", ...). More frequently they are created via the generating function Av1CondContIC, respectively via the method generateIC.

Slots
CalllRfam: object of class "call": creates an object of the underlying L2-differentiable regression type 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 "EuclRandVarList": centering function.
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 average conditional contamination neighborhood.
**Av1CondContIC-class**

**Extends**

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

**Methods**

- CallL2Fam<- signature(object = "Av1CondContIC"): replacement function for slot CallL2Fam.
- cent signature(object = "Av1CondContIC"): accessor function for slot cent.
- cent<- signature(object = "Av1CondContIC"): replacement function for slot cent.
- clip signature(object = "Av1CondContIC"): accessor function for slot clip.
- clip<- signature(object = "Av1CondContIC"): replacement function for slot clip.
- stand signature(object = "Av1CondContIC"): accessor function for slot stand.
- stand<- signature(object = "Av1CondContIC"): replacement function for slot stand.
- lowerCase signature(object = "Av1CondContIC"): accessor function for slot lowerCase.
- lowerCase<- signature(object = "Av1CondContIC"): replacement function for slot lowerCase.
- neighborRadius signature(object = "Av1CondContIC"): accessor function for slot neighborRadius.
- neighborRadius<- signature(object = "Av1CondContIC"): replacement function for slot neighborRadius.
- generateIC signature(neighbor = "Av1CondContNeighborhood", L2Fam = "L2RegTypeFamily").
- generateIC signature(neighbor = "Av1CondContNeighborhood", L2Fam = "L2RegTypeFamily"): generate an object of class "Av1CondContIC". Rarely called directly.
- show signature(object = "Av1CondContIC")

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

CondIC-class, Av1CondContIC

**Examples**

```r
IC1 <- new("Av1CondContIC")
IC1
```
Av1CondContNeighborhood

Generating function for Av1CondContNeighborhood-class

Description

Generates an object of class "Av1CondContNeighborhood".

Usage

Av1CondContNeighborhood(radius = 0, radiusCurve = function(x){1})

Arguments

radius non-negative real: neighborhood radius.
radiusCurve real-valued, non-negative function with L1 norm <= 1.

Value

Object of class "Av1CondContNeighborhood"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

Av1CondContNeighborhood-class

Examples

Av1CondContNeighborhood()

# The function is currently defined as
function(radius = 0, radiusCurve = function(x){1}){
    new("Av1CondContNeighborhood", radius = radius, radiusCurve = radiusCurve)
}
Av1CondContNeighborhood-class

Average conditional contamination neighborhood

Description

Class of average conditional contamination neighborhoods (exponent = 1); i.e. only radius curves $\varepsilon$ with $\|\varepsilon\|_1 \leq 1$.

Objects from the Class

Objects can be created by calls of the form `new("Av1CondContNeighborhood", ...)`. More frequently they are created via the generating function `Av1CondContNeighborhood`.

Slots

type: Object of class "character": “average conditional convex contamination neighborhood”.
radius: Object of class "numeric": neighborhood radius.
radiusCurve: Object of class "function": radius curve with L1 norm $\leq 1$.
exponent: equal to 1.

Extends

Class "Av1CondNeighborhood", directly.
Class "AvCondNeighborhood", by class "Av1CondNeighborhood".
Class "CondNeighborhood", by class "Av1CondNeighborhood".
Class "Neighborhood", by class "Av1CondNeighborhood".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

Av1CondNeighborhood-class

Examples

`new("Av1CondContNeighborhood")`
Av1CondNeighborhood-class

Average conditional neighborhood

Description

Class of average conditional neighborhoods (exponent $\leq 1$); i.e. only radius curves $\varepsilon$ with $\|\varepsilon\|_1 \leq 1$.

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.
- **radiusCurve**: Object of class "function": radius curve with L1 norm $\leq 1$.
- **exponent**: equal to 1.

Extends

Class "AvCondNeighborhood", directly.
Class "CondNeighborhood", by class "AvCondNeighborhood".
Class "Neighborhood", by class "AvCondNeighborhood".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

AvCondNeighborhood-class
Generating function for Av1CondTotalVarIC-class

Description
Generates an object of class "Av1CondContIC"; i.e., an influence curves \( \eta \) of the form

\[
\eta = A x \Lambda_f \min(1, \max(c(x)/(|A x|\Lambda_f), (c(x) + b)/(|A x|\Lambda_f)))
\]

with lower clipping function \( c \), standardized bias \( b \) and standardizing matrix \( A \). \( \Lambda_f \) stands for the L2 derivative of the corresponding error distribution.

Usage
\[
\text{Av1CondTotalVarIC(nameL calllRfam \ J call(BlRregtypefamilyBIL curve \] euclrandvarlist(realrandvariable(map \] list(function(xI {x{1} * x{R}}IL domain \] euclideanspace(dimension \] RIIIL risksL infosL clipup \] infL stand \] asNmatrix(1IL cliplo \] realrandvariable(map \] list(function(xI {Minf}IL domain \] euclideanspace(dimension \] 1IIL lowercase \] nullL neighborradius \] PI
\]

Arguments
- **name** object of class "character".
- **CallL2Fam** object of class "call": creates an object of the underlying L2-differentiable regression type 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.
- **clipUp** positive real: standardized bias.
- **clipLo** object of class "RealRandVariable": lower clipping function.
- **stand** matrix: standardizing matrix.
- **lowerCase** optional constant for lower case solution.
- **neighborRadius** radius of the corresponding (unconditional) contamination neighborhood.

Value
Object of class "Av1CondTotalVarIC"

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>
Av1CondTotalVarIC-class

References


See Also

CondIC-class, Av1CondTotalVarIC-class

Examples

ICI <- Av1CondTotalVarIC()
ICI

---

Av1CondTotalVarIC-class

*Conditionally centered influence curve of total variation type*

Description

Class of conditionally centered (partial) influence curves of contamination type for average conditional total variation neighborhoods; i.e., influence curves $\eta$ of the form

$$\eta = Ax\Lambda_f \min(1, \max(c(x)/(|Ax|\Lambda_f), (c(x) + b)/(|Ax|\Lambda_f)))$$

with lower clipping function $c$, standardized bias $b$ and standardizing matrix $A$. $\Lambda_f$ stands for the L2 derivative of the corresponding error distribution.

Objects from the Class

Objects can be created by calls of the form `new("Av1CondTotalVarIC", ...)`. More frequently they are created via the generating function `Av1CondTotalVarIC`, respectively via the method `generateIC`.

Slots

- `callL2Fam`: object of class "call"; creates an object of the underlying L2-differentiable regression type 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.
- `clipUp`: object of class "numeric": standardized bias.
- `clipLo`: object of class "RealRandVariable": lower clipping function.
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 average conditional contamination neighborhood.

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

Methods

CallL2Fam<- signature(object = "Av1CondTotalVarIC"): replacement function for slot CallL2Fam.
clipLo signature(object = "Av1CondTotalVarIC"): accessor function for slot clipLo.
clipLo<- signature(object = "Av1CondTotalVarIC"): replacement function for slot clipLo.
clipUp signature(object = "Av1CondTotalVarIC"): accessor function for slot clipUp.
clipUp<- signature(object = "Av1CondTotalVarIC"): replacement function for slot clipUp.
stand signature(object = "Av1CondTotalVarIC"): accessor function for slot stand.
stand<- signature(object = "Av1CondTotalVarIC"): replacement function for slot stand.
lowerCase signature(object = "Av1CondTotalVarIC"): accessor function for slot lowerCase.
lowerCase<- signature(object = "Av1CondTotalVarIC"): replacement function for slot lowerCase.
neighborRadius signature(object = "Av1CondTotalVarIC"): accessor function for slot neighborRadius.
neighborRadius<- signature(object = "Av1CondTotalVarIC"): replacement function for slot neighborRadius.
generateIC signature(neighbor = "Av1CondTotalVarNeighborhood", L2Fam = "L2RegTypeFamily"): generate an object of class "Av1CondTotalVarIC". Rarely called directly.
show signature(object = "Av1CondTotalVarIC")

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

References

See Also
CondIC-class, Av1CondTotalVarIC

Examples
IC1 <- new("Av1CondTotalVarIC")
IC1
Av1CondTotalVarNeighborhood

Generating function for Av1CondTotalVarNeighborhood-class

Description

Generates an object of class "Av1CondTotalVarNeighborhood".

Usage

Av1CondTotalVarNeighborhood(radius = 0, radiusCurve = function(x){1})

Arguments

  radius    non-negative real: neighborhood radius.
  radiusCurve real-valued, non-negative function with L1 norm <= 1.

Value

Object of class "Av1CondTotalVarNeighborhood"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

Av1CondTotalVarNeighborhood-class

Examples

Av1CondTotalVarNeighborhood()

## The function is currently defined as
function(radius = 0, radiusCurve = function(x){1}){
  new("Av1CondTotalVarNeighborhood", radius = radius, radiusCurve = radiusCurve)
}
Description

Class of average conditional total variation neighborhoods (exponent == 1); i.e. only radius curves ε with ∥ε∥₁ ≤ 1.

Objects from the Class

Objects can be created by calls of the form `new("Av1CondTotalVarNeighborhood", ...)`. More frequently they are created via the generating function `Av1CondTotalVarNeighborhood`.

Slots

type: Object of class "character": “average conditional total variation neighborhood”.
radius: Object of class "numeric": neighborhood radius.
radiusCurve: Object of class "function": radius curve with L₁ norm <= 1.
exponent: equal to 1.

Extends

Class "Av1CondNeighborhood", directly.
Class "Av1CondNeighborhood", by class "Av1CondNeighborhood".
Class "CondNeighborhood", by class "Av1CondNeighborhood".
Class "Neighborhood", by class "Av1CondNeighborhood".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`Av1CondNeighborhood-class`

Examples

`new("Av1CondTotalVarNeighborhood")`
Description

Generates an object of class "Av2CondContIC"; i.e., an influence curves $\eta$ of the form

$$\eta = AK^{-1}x(\Lambda f - z) \min(1, c/|\Lambda f - z|)$$

with $K = Exx^T$, clipping bound $c$, centering constant $z$ and standardizing constant $A$. $\Lambda f$ stands for the L2 derivative of the corresponding error distribution.

Usage

Av2CondContIC(nameL calllRfam = call("L2RegTypeFamily"),
  Curve = EuclRandVarList(RealRandVariable(    
    Map = list(function(x) {x[1] * x[2]}),
    Domain = EuclideanSpace(dimension = 2)),
  Risks, Infos, clip = Inf, stand = 1, cent = 0, lowerCase = NULL,
  neighborRadius = 0)

Arguments

- name object of class "character".
- CallL2Fam object of class "call": creates an object of the underlying L2-differentiable regression type 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 real: standardizing constant
- lowerCase optional constant for lower case solution.
- neighborRadius radius of the corresponding (unconditional) contamination neighborhood.

Value

Object of class "Av2CondContIC"

Author(s)

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


### See Also

CondIC-class, Av2CondContIC-class

### Examples

```r
IC1 <- Av2CondContIC()
IC1
```

### Description

Class of conditionally centered (partial) influence curves of contamination type for average square conditional contamination neighborhoods; i.e., influence curves $\eta$ of the form

$$\eta = AK^{-1}x(\Lambda_f - z) \min(1, c/|\Lambda_f - z|)$$

with $K = Exx^T$, clipping bound $c$, centering constant $z$ and standardizing constant $A$. $\Lambda_f$ stands for the L2 derivative of the corresponding error distribution.

### Objects from the Class

Objects can be created by calls of the form `new("Av2CondContIC", ...`). More frequently they are created via the generating function `Av2CondContIC`, respectively via the method `generateIC`.

### Slots

- **CallL2Fam**: object of class "call": creates an object of the underlying L2-differentiable regression type 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 "numeric": standardizing constant.
- **lowerCase**: object of class "OptionalNumeric": optional constant for lower case solution.
- **neighborRadius**: object of class "numeric": radius of the corresponding average conditional contamination neighborhood.
Av2CondContIC-class

Extends

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

Methods

CallL2Fam<- signature(object = "Av2CondContIC"): replacement function for slot CallL2Fam.

cent signature(object = "Av2CondContIC"): accessor function for slot cent.

cent<- signature(object = "Av2CondContIC"): replacement function for slot cent.

clip signature(object = "Av2CondContIC"): accessor function for slot clip.

clip<- signature(object = "Av2CondContIC"): replacement function for slot clip.

stand signature(object = "Av2CondContIC"): accessor function for slot stand.

stand<- signature(object = "Av2CondContIC"): replacement function for slot stand.

lowerCase signature(object = "Av2CondContIC"): accessor function for slot lowerCase.

lowerCase<- signature(object = "Av2CondContIC"): replacement function for slot lowerCase.

neighborRadius signature(object = "Av2CondContIC"): accessor function for slot neighborRadius.

neighborRadius<- signature(object = "Av2CondContIC"): replacement function for slot neighborRadius.

generateIC signature(neighbor = "Av2CondContNeighborhood", L2Fam = "L2RegTypeFamily"): generate an object of class "Av2CondContIC". Rarely called directly.

show signature(object = "Av2CondContIC")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

CondIC-class, Av2CondContIC

Examples

IC1 <- new("Av2CondContIC")
IC1
**Av2CondContNeighborhood**

*Generating function for Av2CondContNeighborhood-class*

---

**Description**

Generates an object of class "Av2CondContNeighborhood".

**Usage**

```r
Av2CondContNeighborhood(radius = 0, radiusCurve = function(x){1})
```

**Arguments**

- `radius`: non-negative real: neighborhood radius.
- `radiusCurve`: real-valued, non-negative function with L2 norm <= 1.

**Value**

Object of class "Av1CondContNeighborhood"

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

`Av2CondContNeighborhood-class`

**Examples**

```r
Av2CondContNeighborhood()

## The function is currently defined as
function(radius = 0, radiusCurve = function(x){1})){
    new("Av2CondContNeighborhood", radius = radius, radiusCurve = radiusCurve)
}
```
Av2CondContNeighborhood-class

Average square conditional contamination neighborhood

Description

Class of average square conditional contamination neighborhoods (exponent == 2); i.e. only radius curves $\varepsilon$ with $\|\varepsilon\|_2 \leq 1$.

Objects from the Class

Objects can be created by calls of the form new("Av2CondContNeighborhood", ...). More frequently they are created via the generating function Av2CondContNeighborhood.

Slots

type: Object of class "character": “average square conditional convex contamination neighborhood”.

radius: Object of class "numeric": neighborhood radius.

radiusCurve: Object of class "function": radius curve with L2 norm <= 1.

exponent: equal to 2.

Extends

Class "Av2CondNeighborhood", directly.
Class "AvCondNeighborhood", by class "Av2CondNeighborhood".
Class "CondNeighborhood", by class "Av2CondNeighborhood".
Class "Neighborhood", by class "Av2CondNeighborhood".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

Av2CondNeighborhood-class

Examples

new("Av2CondContNeighborhood")
Av2CondNeighborhood-class

Average square conditional neighborhood

Description

Class of average square conditional neighborhoods (exponent == 2); i.e. only radius curves \( \varepsilon \) with \( \| \varepsilon \|_2 \leq 1 \).

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.
- radiusCurve: Object of class "function": radius curve with L2 norm <= 1.
- exponent: equal to 2.

Extends

Class "AvCondNeighborhood", directly.
Class "CondNeighborhood", by class "AvCondNeighborhood".
Class "Neighborhood", by class "AvCondNeighborhood".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

AvCondNeighborhood-class
AvCondNeighborhood-class

*Average conditional neighborhood*

### Description

Class of average conditional neighborhoods; i.e. only radius curves $\varepsilon$ with $\|\varepsilon\|_\alpha \leq 1$ for given exponent $\alpha$.

### 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.
- **radiusCurve**: Object of class "function": radius curve.
- **exponent**: Object of class "numeric": positive integer or Inf.

### Extends

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

### Methods

- **show** signature(object = "AvCondNeighborhood")

### Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

### References


### See Also

*CondNeighborhood-class*
Generating function for CondContIC-class

Description

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

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

with clipping function $b$, centering function $a$ 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
CondContIC(name, CallL2Fam = call("L2RegTypeFamily"),
Curve = EuclRandVarList(RealRandVariable(
    Map = list(function(x)(x[1]*x[2])),
    Domain = EuclideanSpace(dimension = 2))),
Risks, Infos,
clip = RealRandVariable(Map = list(function(x){Inf }), Domain = Reals()),
stand = as.matrix(1),
cent = EuclRandVarList(RealRandVariable(
    Map = list(function(x){numeric(length(x))}),
    Domain = EuclideanSpace(dimension = 2)),
lowerCase = NULL, neighborRadius = 0, neighborRadiusCurve = function(x){1})
```

Arguments

- **name**: object of class "character".
- **CallL2Fam**: object of class "call": creates an object of the underlying L2-differentiable regression type 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**: object of class "RealRandVariable": clipping function.
- **cent**: object of class "EuclRandVarList": centering function.
- **stand**: matrix: standardizing matrix.
- **lowerCase**: optional constant for lower case solution.
- **neighborRadius**: radius of the corresponding conditional contamination neighborhood.
- **neighborRadiusCurve**: radius curve of the corresponding conditional contamination neighborhood.
Value

Object of class "CondContIC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

CondIC-class, CondContIC-class

Examples

IC1 <- CondContIC()
IC1

| CondContIC-class | Conditionally centered influence curve of contamination type |

Description

Class of conditionally centered (partial) influence curves of contamination type for conditional contamination neighborhoods; i.e., influence curves $\eta$ of the form

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

with clipping function $b$, centering function $a$ and standardizing matrix $A$. $\Lambda$ stands for the L2 derivative of the corresponding L2 differentiable regression type family created via the call in the slot CallL2Fam.

Objects from the Class

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

**CallL2Fam**: object of class "call": creates an object of the underlying L2-differentiable regression type 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 "RealRandVariable": clipping function.

**cent**: object of class "EuclRandVarList": centering function.

**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 conditional contamination neighborhood.

**neighborRadiusCurve**: object of class "function": radius curve of the corresponding conditional contamination neighborhood.

### Extends

Class "CondIC", directly.

Class "IC", by class "CondIC".

Class "InfluenceCurve", by class "CondIC".

### Methods

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

**cent** signature(object = "CondContIC"): accessor function for slot cent.

**cent<-** signature(object = "CondContIC"): replacement function for slot cent.

**clip** signature(object = "CondContIC"): accessor function for slot clip.

**clip<-** signature(object = "CondContIC"): replacement function for slot clip.

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

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

**lowerCase** signature(object = "CondContIC"): accessor function for slot lowerCase.

**lowerCase<-** signature(object = "CondContIC"): replacement function for slot lowerCase.

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

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

**neighborRadiusCurve** signature(object = "CondContIC"): accessor function for slot neighborRadiusCurve.

**neighborRadiusCurve<-** signature(object = "CondContIC"): replacement function for slot neighborRadiusCurve.

**generateIC** signature(neighbor = "CondContNeighborhood", L2Fam = "L2RegTypeFamily"): generate an object of class "CondContIC". Rarely called directly.

**show** signature(object = "CondContIC")
CondContNeighborhood

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

References

See Also
CondIC-class, CondContIC

Examples
IC1 <- new("CondContIC")
IC1

CondContNeighborhood Generating function for CondContNeighborhood-class

Description
Generates an object of class "CondContNeighborhood".

Usage
CondContNeighborhood(radius = 0, radiusCurve = function(x){1})

Arguments
radius non-negative real: neighborhood radius.
radiusCurve real-valued, non-negative function.

Value
Object of class "CondContNeighborhood"

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

References
See Also

CondContNeighborhood-class

Examples

CondContNeighborhood()

## The function is currently defined as
function(radius = 0, radiusCurve = function(x){1}){
    new("CondContNeighborhood", radius = radius, radiusCurve = radiusCurve)
}

CondContNeighborhood-class

Conditional contamination neighborhood

Description

Class of conditional (error-free-variables) convex contamination neighborhoods.

Objects from the Class

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

Slots

type: Object of class "character": “conditional convex contamination neighborhood”.
radius: Object of class "numeric": neighborhood radius.
radiusCurve: Object of class "function": radius curve

Extends

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also

CondContNeighborhood, CondNeighborhood-class

Examples

new("CondContNeighborhood")

Generating function for CondIC-class

Description

Generates an object of class "CondIC".

Usage

CondIC(name, Curve = EuclRandVarList(EuclRandVariable(
    Map = list(function(x){x[1] * x[2]}),
    Domain = EuclideanSpace(dimension = 2),
    Range = Reals())),
    Risks, Infos, CallL2Fam = call("L2RegTypeFamily"))

Arguments

name character string: name.
CallL2Fam object of class "call": creates an object of "L2RegTypeFamily".
Curve object of class "EuclRandVariable": curve
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 "CondIC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
CondIC-class

Examples
CondIC()

## The function is currently defined as
function(name, Curve = EuclRandVariable(Map = list(function(x){x[1]*x[2]}),
   Domain = EuclideanSpace(dimension = 2)),
   Risks, Infos, CallL2Fam = call("L2RegTypeFamily")){
   if(missing(name))
      name <- "Influence curve for a L_2 differentiable regression type family"
   if(missing(Risks))
      Risks <- list()
   if(missing(Icons))
      Infos <- matrix(c(character(0),character(0), ncol=2,
         dimnames=list(character(0), c("method", "message")))
   return(new("CondIC", name = name, Curve = Curve, Risks = Risks,
      Infos = Infos, CallL2Fam = CallL2Fam))
}

Description
Class of conditionally centered partial influence curves.

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

Slots
CallL2Fam: Object of class "call": creates an object of class "L2RegTypeFamily".
name: Object of class "character": name
Curve: Object of class "EuclRandVariable": curve.
Risks: Object of class "list": list of risks; cf. RiskType-class.
Infos: Object of class "matrix" with two columns named method and message: additional infor-
mations.

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

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

**checkIC** signature(IC = "CondIC", L2Fam = "missing"): check conditional centering and Fisher consistency of IC assuming the L2-differentiable regression-type family which can be created via the slot CallL2Fam of IC.

**checkIC** signature(IC = "CondIC", L2Fam = "L2RegTypeFamily"): check conditional centering and Fisher consistency of IC assuming the L2-differentiable regression-type family L2Fam.

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

*InfluenceCurve-class, IC-class*

Examples

new("CondIC")

---

**CondNeighborhood-class**

Conditional neighborhood

Description

Class of conditional (error-free-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.

radiusCurve: Object of class "function": radius curve.
CondTotalVarIC

Extends
Class "Neighborhood", directly.

Methods
radiusCurve signature(object = "CondNeighborhood"): accessor function for slot radiusCurve.
show signature(object = "CondNeighborhood")

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

References

See Also
Neighborhood-class

CondTotalVarIC Generating function for CondTotalVarIC-class

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

\[
\eta = \max(c(x), \min(Ax \Lambda_f, b(x))
\]

with lower clipping function \( c \), upper clipping function \( b \) and standardizing matrix \( A \). \( \Lambda_f \) stands for the L2 derivative of the corresponding error distribution.

Usage
CondTotalVarIC(name, CallL2Fam = call("L2RegTypeFamily"),
    Curve = EuclRandVarList(RealRandVariable(
        Map = list(function(x) {x[1] * x[2]}),
        Domain = EuclideanSpace(dimension = 2))),
    Risks, Infos,
clipUp = RealRandVariable(Map = list(function(x) {Inf}), Domain = Reals()),
    stand = as.matrix(1),
clipLo = RealRandVariable(Map = list(function(x) {-Inf}), Domain = Reals()),
    lowerCase = NULL, neighborRadius = 0, neighborRadiusCurve = function(x){1}})
Arguments

name object of class "character".
CallL2Fam object of class "call": creates an object of the underlying L2-differentiable regression type 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.
clipUp object of class "RealRandVariable": upper clipping function.
clipLo object of class "RealRandVariable": lower clipping function.
stand matrix: standardizing matrix.
lowerCase optional constant for lower case solution.
neighborRadius radius of the corresponding conditional total variation neighborhood.
neighborRadiusCurve radius curve of the corresponding conditional total variation neighborhood.

Value

Object of class "CondTotalVarIC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

CondIC-class, CondTotalVarIC-class

Examples

IC1 <- CondTotalVarIC()
IC1
CondTotalVarIC-class

Conditionally centered influence curve of total variation type

Description

Class of conditionally centered (partial) influence curves of contamination type for average conditional total variation

\[ \eta = \max(c(x), \min(Ax\Lambda_f, b(x))) \]

with lower clipping function \( c \), upper clipping function \( b \) and standardizing matrix \( A \). \( \Lambda_f \) stands for the L2 derivative of the corresponding error distribution.

Objects from the Class

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

Slots

- `callL2Fam`: object of class "call": creates an object of the underlying L2-differentiable regression type 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.
- `clipUp`: object of class "RealRandVariable": upper clipping function.
- `clipLo`: object of class "RealRandVariable": lower clipping function.
- `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 conditional contamination neighborhood.
- `neighborRadiusCurve`: object of class "numeric": radius curve of the corresponding conditional contamination neighborhood.

Extends

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

CallL2Fam <- signature(object = "CondTotalVarIC"): replacement function for slot CallL2Fam.
clipLo signature(object = "CondTotalVarIC"): accessor function for slot clipLo.
clipLo <- signature(object = "CondTotalVarIC"): replacement function for slot clipLo.
clipUp signature(object = "CondTotalVarIC"): accessor function for slot clipUp.
clipUp <- signature(object = "CondTotalVarIC"): replacement function for slot clipUp.
stand signature(object = "CondTotalVarIC"): accessor function for slot stand.
stand <- signature(object = "CondTotalVarIC"): replacement function for slot stand.
lowerCase signature(object = "CondTotalVarIC"): accessor function for slot lowerCase.
lowerCase <- signature(object = "CondTotalVarIC"): replacement function for slot lowerCase.
neighborRadius signature(object = "CondTotalVarIC"): accessor function for slot neighborRadius.
neighborRadius <- signature(object = "CondTotalVarIC"): replacement function for slot neighborRadius.
neighborRadiusCurve signature(object = "CondTotalVarIC"): accessor function for slot neighborRadiusCurve.
neighborRadiusCurve <- signature(object = "CondTotalVarIC"): replacement function for slot neighborRadiusCurve.
generateIC signature(neighbor = "CondTotalVarNeighborhood", L2Fam = "L2RegTypeFamily"): generate an object of class "CondTotalVarIC". Rarely called directly.
show signature(object = "CondTotalVarIC")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

CondIC-class, CondTotalVarIC

Examples

IC1 <- new("CondTotalVarIC")
IC1
Description

Generates an object of class "CondTotalVarNeighborhood".

Usage

CondTotalVarNeighborhood(radius = 0, radiusCurve = function(x){1})

Arguments

radius non-negative real: neighborhood radius.
radiusCurve real-valued, non-negative function.

Value

Object of class "ContNeighborhood"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

CondTotalVarNeighborhood-class

Examples

CondTotalVarNeighborhood()

## The function is currently defined as
function(radius = 0, radiusCurve = function(x){1}){
    new("CondTotalVarNeighborhood", radius = radius, radiusCurve = radiusCurve)
}
CondTotalVarNeighborhood-class

Conditional total variation neighborhood

Description

Class of conditional (error-free-variables) total variation neighborhoods.

Objects from the Class

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

Slots

type: Object of class "character": “conditional total variation neighborhood”.
radius: Object of class "numeric": neighborhood radius.
radiusCurve: Object of class "function": radius curve

Extends

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

CondTotalVarNeighborhood, CondNeighborhood-class

Examples

new("CondTotalVarNeighborhood")
FixRobRegTypeModel

Generating function for FixRobRegTypeModel-class

Description

Generates an object of class "FixRobRegTypeModel".

Usage

FixRobRegTypeModel(center = RegTypeFamily(), neighbor = ContNeighborhood())

Arguments

center: object of class "RegTypeFamily"
neighbor: object of class "Neighborhood"

Value

Object of class "FixRobRegTypeModel"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

FixRobRegTypeModel-class

Examples

FixRobRegTypeModel()

### The function is currently defined as

def function(center = RegTypeFamily(), neighbor = ContNeighborhood()){
    new("FixRobRegTypeModel", center = center, neighbor = neighbor)
}
FixRobRegTypeModel-class

Robust regression-type model with fixed neighborhood

Description

Class of robust regression-type models with fixed (conditional or unconditional) neighborhoods.

Objects from the Class

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

Slots

- `center`: Object of class "RegTypeFamily".
- `neighbor`: Object of class "Neighborhood".

Extends

Class "RobModel", directly.

Methods

- `neighbor<-` signature(object = "FixRobRegTypeModel") replacement function for slot `neighbor`.
- `show` signature(object = "FixRobRegTypeModel")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`RegTypeFamily-class`, `Neighborhood-class`, `FixRobRegTypeModel`

Examples

`new("FixRobRegTypeModel")`
Description
Methods for function generateIC in package ROptRegTS.

Methods

- `neighbor = "ContNeighborhood", L2Fam = "L2RegTypeFamily"` generate an object of class "ContIC". Rarely called directly.
- `neighbor = "TotalVarNeighborhood", L2Fam = "L2RegTypeFamily"` generate an object of class "TotalVarIC". Rarely called directly.

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

See Also
generateIC

generateIC-methods

getAsRiskRegTS

Generic Function for Computation of Asymptotic Risks in case of Regression-Type Models

Description
Generic function for the computation of asymptotic risks in case of regression-type models. This function is rarely called directly. It is used by other functions.

Usage

```r
getAsRiskRegTS(risk, ErrorL2deriv, Regressor, neighbor, ...)  
```  
## S4 method for signature
## 'asMSE,UnivariateDistribution,Distribution,Neighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, clip, cent, stand, trafo)

## S4 method for signature
## 'asMSE,UnivariateDistribution,Distribution,Av2CondContNeighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, clip, cent, stand, trafo)
```
getAsRiskRegTS

# S4 method for signature 'asMSE,EuclRandVariable,Distribution,Neighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, clip, cent, stand, trafo)

# S4 method for signature
# 'asBias,
#  UnivariateDistribution,
#  UnivariateDistribution,
#  ContNeighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, ErrorL2derivDistrSymm, trafo, maxiter, tol)

# S4 method for signature
# 'asBias,
#  UnivariateDistribution,
#  UnivariateDistribution,
#  Av1CondContNeighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, ErrorL2derivDistrSymm, trafo, maxiter, tol)

# S4 method for signature
# 'asBias,
#  UnivariateDistribution,
#  UnivariateDistribution,
#  Av1CondTotalVarNeighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, ErrorL2derivDistrSymm, trafo, maxiter, tol)

# S4 method for signature
# 'asBias,
#  UnivariateDistribution,
#  MultivariateDistribution,
#  ContNeighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, ErrorL2derivDistrSymm, trafo, maxiter, tol)

# S4 method for signature
# 'asBias,
#  UnivariateDistribution,
#  MultivariateDistribution,
#  Av1CondContNeighborhood'
getAsRiskRegTS(
  risk, ErrorL2deriv, Regressor, neighbor, ErrorL2derivDistrSymm, trafo, maxiter, tol)
getAsRiskRegTS(
  risk, ErrorL2deriv, Regessor, neighbor, ErrorL2derivDistrSymm, 
  trafo, maxiter, tol)

getAsRiskRegTS(
  risk, ErrorL2deriv, Regessor, neighbor, ErrorL2derivDistrSymm, 
  trafo, maxiter, tol)

getAsRiskRegTS(
  risk, ErrorL2deriv, Regessor, neighbor, ErrorDistr, trafo, z.start, 
  A.start, maxiter, tol)

getAsRiskRegTS(
  risk, ErrorL2deriv, Regessor, neighbor, ErrorDistr, trafo, z.start, 
  A.start, maxiter, tol)

getAsRiskRegTS(
  risk, ErrorL2deriv, Regessor, neighbor, clip, cent, stand)

getAsRiskRegTS(
  risk, ErrorL2deriv, Regessor, neighbor, clip, cent, stand)

Arguments

risk object of class "asRisk".
ErrorL2deriv L2-derivative of ErrorDistr.
Regressor regressor.
neighbor object of class "Neighborhood".
... additional parameters.
clip optimal clipping bound.
cent optimal centering constant/function.
stand standardizing matrix.
trafo matrix: transformation of the parameter.
ErrorDistr error distribution.
ErrorL2derivDistrSymm symmetry of ErrorL2derivDistr.
maxiter the maximum number of iterations
tol the desired accuracy (convergence tolerance).
z.start initial value for the centering constant/function.
A.start initial value for the standardizing matrix.

Value
The asymptotic risk is computed.

Methods
- **risk = "asMSE", ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", neighbor = "Neighborhood"** computes asymptotic mean square error in methods for function getInfRobRegTypeIC.
- **risk = "asMSE", ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", neighbor = "Av2CondContNeighborhood"** computes asymptotic mean square error in methods for function getInfRobRegTypeIC.
- **risk = "asMSE", ErrorL2deriv = "EuclRandVariable", Regressor = "Distribution", neighbor = "Neighborhood"** computes asymptotic mean square error in methods for function getInfRobRegTypeIC.
- **risk = "asBias", ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "ContNeighborhood"** computes standardized asymptotic bias in methods for function getInfRobRegTypeIC.
- **risk = "asBias", ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "Av1CondContNeighborhood"** computes standardized asymptotic bias in methods for function getInfRobRegTypeIC.
- **risk = "asBias", ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "Av1CondTotalVarNeighborhood"** computes standardized asymptotic bias in methods for function getInfRobRegTypeIC.
- **risk = "asBias", ErrorL2deriv = "UnivariateDistribution", Regressor = "MultivariateDistribution", neighbor = "ContNeighborhood"** computes standardized asymptotic bias in methods for function getInfRobRegTypeIC.
- **risk = "asBias", ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "Av1CondContNeighborhood"** computes standardized asymptotic bias in methods for function getInfRobRegTypeIC.
- **risk = "asBias", ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", neighbor = "ContNeighborhood"** computes standardized asymptotic bias in methods for function getInfRobRegTypeIC.
getFiRiskRegTS

risk = "asBias", ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", neighbor = "Av1CondContNeighborhood"
computes standardized asymptotic bias in methods for function getInfRobRegTypeIC.

risk = "asUnOvShoot", ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "UncondNeighborhood"
computes asymptotic under-/overshoot risk in methods for function getInfRobRegTypeIC.

risk = "asUnOvShoot", ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "CondNeighborhood"
computes asymptotic under-/overshoot risk in methods for function getInfRobRegTypeIC.

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

References

See Also

asRisk-class

generic function for the computation of finite-sample risks in regression-type models. This function is rarely called directly. It is used by other functions.

Description

getFiRiskRegTS(risk, ErrorDistr, Regressor, neighbor, ...)  

Usage

getFiRiskRegTS(risk, ErrorDistr, Regressor, neighbor, ...)  

## S4 method for signature
## 'fiUnOvShoot,Norm,UnivariateDistribution,ContNeighborhood'
getFiRiskRegTS(  
    risk, ErrorDistr, Regressor, neighbor, clip, stand, sampleSize,  
    Algo, cont)

## S4 method for signature
## 'fiUnOvShoot,Norm,UnivariateDistribution,TotalVarNeighborhood'
getFiRiskRegTS(  
    risk, ErrorDistr, Regressor, neighbor, clip, stand, sampleSize,
getFiRiskRegTS

Algo, cont)

## S4 method for signature
## 'fiUnOvShoot, Norm, UnivariateDistribution, CondContNeighborhood'
getiRiskRegTS(
  risk, ErrorDistr, Regressor, neighbor, clip, stand, sampleSize, cont)

## S4 method for signature
## 'fiUnOvShoot, Norm, UnivariateDistribution, CondTotalVarNeighborhood'
getiRiskRegTS(
  risk, ErrorDistr, Regressor, neighbor, clip, stand, sampleSize, cont)

Arguments

- **risk**: object of class "RiskType".
- **ErrorDistr**: error distribution
- **Regressor**: regressor
- **neighbor**: object of class "Neighborhood".
- **clip**: optimal clipping bound/function.
- **stand**: standardizing 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 Subsections 12.1.3 and 12.2.3 of Kohl (2005).

Value

The finite-sample risk is computed.

Methods

- **risk = "fiUnOvShoot", ErrorDistr = "Norm", Regressor = "UnivariateDistribution", neighbor = "ContNeighborhood"**
  computes finite-sample under-/overshoot risk in methods for function 'getFixRobRegTypeIC'.
- **risk = "fiUnOvShoot", ErrorDistr = "Norm", Regressor = "UnivariateDistribution", neighbor = "TotalVarNeighborhood"**
  computes finite-sample under-/overshoot risk in methods for function 'getFixRobRegTypeIC'.
- **risk = "fiUnOvShoot", ErrorDistr = "Norm", Regressor = "UnivariateDistribution", neighbor = "CondContNeighborhood"**
  computes finite-sample under-/overshoot risk in methods for function 'getFixRobRegTypeIC'.
- **risk = "fiUnOvShoot", ErrorDistr = "Norm", Regressor = "UnivariateDistribution", neighbor = "CondTotalVarNeighborhood"**
  computes finite-sample under-/overshoot risk in methods for function 'getFixRobRegTypeIC'.
getFixClipRegTS

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

References

See Also
fiRisk-class

---

getFixClipRegTS

Generic Function for the Computation of the Optimal Clipping Bound

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

Usage
getFixClipRegTS(clip, ErrorDistr, Regressor, risk, neighbor, ...)

Arguments
clip optimal clipping bound.
ErrorDistr error distribution.
Regressor regressor.
risk object of class "RiskType".
neighbor object of class "Neighborhood".
... additional parameters.

Value
The optimal clipping bound/function is computed.
Methods

clip = "numeric", ErrorDistr = "Norm", Regressor = "UnivariateDistribution", risk = "fiUnOvShoot", neighbor = "ContNeighborhood"
Optimal clipping bound for finite-sample under-/overshoot risk.

clip = "numeric", ErrorDistr = "Norm", Regressor = "UnivariateDistribution", risk = "fiUnOvShoot", neighbor = "TotalVarNeighborhood"
Optimal clipping bound for finite-sample under-/overshoot risk.

clip = "numeric", ErrorDistr = "Norm", Regressor = "numeric", risk = "fiUnOvShoot", neighbor = "CondContNeighborhood"
Optimal clipping function for finite-sample under-/overshoot risk.

clip = "numeric", ErrorDistr = "Norm", Regressor = "numeric", risk = "fiUnOvShoot", neighbor = "CondTotalVarNeighborhood"
Optimal clipping function for finite-sample under-/overshoot risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ContIC-class, TotalVarIC-class, Av1CondContIC-class, Av2CondContIC-class, Av1CondTotalVarIC-class, CondContIC-class, CondTotalVarIC-class

getFixRobRegTypeIC  Generic Function for the Computation of Optimally Robust Regression-Type ICs

Description

Generic function for the computation of optimally robust regression-type ICs in case of fixed robust models. This function is rarely called directly.

Usage

getFixRobRegTypeIC(ErrorDistr, Regressor, risk, neighbor, ...)

## S4 method for signature
## 'Norm,UnivariateDistribution,fiUnOvShoot,UncondNeighborhood'
ggetFixRobRegTypeIC(ErrorDistr, Regressor, risk, neighbor, sampleSize, upper, maxiter, tol, warn, Algo, cont)
getFixRobRegTypeIC

## S4 method for signature
## 'Norm,UnivariateDistribution,fiUnOvShoot,CondNeighborhood'
getFixRobRegTypeIC(ErrorDistr, Regressor, risk, neighbor, sampleSize, upper, maxiter, tol, warn, Algo, cont)

**Arguments**

- **ErrorDistr**: error distribution
- **Regressors**: regressor
- **risk**: object of class "RiskType".
- **neighbor**: object of class "Neighborhood".
- **samplesize**: integer: sample size.
- **upper**: upper bound for the optimal clipping bound.
- **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**

- **ErrorDistr = "Norm", Regressor = "UnivariateDistribution", risk = "fiUnOvShoot", neighbor = "UncondNeighborhood"** computes the optimally robust influence curve for one-dimensional normal regression and finite-sample under-/overshoot risk.

- **ErrorDistr = "Norm", Regressor = "UnivariateDistribution", risk = "fiUnOvShoot", neighbor = "CondNeighborhood"** computes the optimally robust influence curve for one-dimensional normal regression and finite-sample under-/overshoot risk.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


Methods for Function `getIneffDiff` in Package ‘ROptRegTS’

## Description

Methods for function `getIneffDiff` in package `ROptRegTS`. These methods are rarely called directly. They are used to compute the radius minimax IC and the least favorable radius.

## Methods

- **radius = "numeric", L2Fam = "L2RegTypeFamily", neighbor = "Neighborhood", risk = "asMSE"**
  - Computes difference of asymptotic MSE–inefficiency for the boundaries of a given radius interval.

- **radius = "numeric", L2Fam = "L2RegTypeFamily", neighbor = "Av2CondContNeighborhood", risk = "asMSE"**
  - Computes difference of asymptotic MSE–inefficiency for the boundaries of a given radius interval.

## Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

## See Also

- `getIneffDiff`

---

**getInfCentRegTS**

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

## Description

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

getInfCentRegTS(ErrorL2deriv, Regressor, neighbor, ...)

## S4 method for signature
## 'UnivariateDistribution, UnivariateDistribution, ContNeighborhood'
getInfCentRegTS(
    ErrorL2deriv, Regressor, neighbor, clip, cent, stand, z.comp)

## S4 method for signature
## 'UnivariateDistribution, UnivariateDistribution, TotalVarNeighborhood'
getInfCentRegTS(
    ErrorL2deriv, Regressor, neighbor, clip, cent, z.comp)

## S4 method for signature
## 'UnivariateDistribution, numeric, CondTotalVarNeighborhood'
getInfCentRegTS(
    ErrorL2deriv, Regressor, neighbor, clip, cent, z.comp)

## S4 method for signature
## 'UnivariateDistribution, UnivariateDistribution,
##    Av1CondContNeighborhood'
getInfCentRegTS(
    ErrorL2deriv, Regressor, neighbor, clip, cent, stand, z.comp, x.vec)

## S4 method for signature
## 'UnivariateDistribution, UnivariateDistribution,
##    Av1CondTotalVarNeighborhood'
getInfCentRegTS(
    ErrorL2deriv, Regressor, neighbor, clip, cent, stand, z.comp, x.vec,
    tol.z)

## S4 method for signature
## 'UnivariateDistribution, MultivariateDistribution, ContNeighborhood'
getInfCentRegTS(
    ErrorL2deriv, Regressor, neighbor, clip, cent, stand, z.comp)

## S4 method for signature
## 'UnivariateDistribution, MultivariateDistribution,
##    Av1CondContNeighborhood'
getInfCentRegTS(
    ErrorL2deriv, Regressor, neighbor, clip, cent, stand, z.comp, x.vec)

## S4 method for signature
## 'UnivariateDistribution, Distribution, Av2CondContNeighborhood'
getInfCentRegTS(

...
getInfCentRegTS

Arguments

ErrorL2deriv  L2-derivative of ErrorDistr.
Regressor    regressor.
neighbor     object of class "Neighborhood".
             additional parameters.
clip         optimal clipping bound.
cent         optimal centering constant/function.
stand        standardizing matrix.
z.comp       which components of the centering constant/function have to be computed.
x.vec        (approximated) support of Regressor.
tol.z        the desired accuracy (convergence tolerance).
ErrorDistr   error distribution.

Value

The optimal centering constant/function is computed.

Methods


ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "Av1CondTotalVarNeighborhood" computation of optimal lower clipping function.

getInfClipRegTS

**ErrorL2deriv** = "UnivariateDistribution", **Regressor** = "MultivariateDistribution", **neighbor** = "Av1CondContNeighborhood"
computation of optimal centering function.

**ErrorL2deriv** = "UnivariateDistribution", **Regressor** = "MultivariateDistribution", **neighbor** = "Av1CondTotalVarNeighborhood"
computation of optimal lower clipping function.

**ErrorL2deriv** = "UnivariateDistribution", **Regressor** = "Distribution", **neighbor** = "Av2CondContNeighborhood"
computation of optimal centering constant.

**ErrorL2deriv** = "RealRandVariable", **Regressor** = "Distribution", **neighbor** = "ContNeighborhood"
computation of optimal centering constant.

**ErrorL2deriv** = "RealRandVariable", **Regressor** = "Distribution", **neighbor** = "Av1CondContNeighborhood"
computation of optimal centering function.

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

**References**

**See Also**
ContIC-class, Av1CondContIC-class, Av2CondContIC-class, Av1CondTotalVarIC-class, CondContIC-class,
CondTotalVarIC-class

---

getInfClipRegTS  
**Generic Function for the Computation of the Optimal Clipping Bound**

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

**Usage**
getInfClipRegTS(clip, ErrorL2deriv, Regressor, risk, neighbor, ...)

## S4 method for signature
## 'numeric,UnivariateDistribution,Distribution,asMSE,Neighborhood'
getInfClipRegTS(
  clip, ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent)

## S4 method for signature
## 'numeric,'
getInfClipRegTS(
  clip, ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent)

Arguments

clip         optimal clipping bound.
ErrorL2deriv L2-derivative of ErrorDistr.
Regressors   regressor.
risk         object of class "RiskType".
neighbor     object of class "Neighborhood".
...          additional parameters.
cent         optimal centering constant/function.
stand        standardizing matrix.
z.comp       which components of the centering constant/function have to be computed.
ErrorDistr   error distribution.
trafo        matrix: transformation of the parameter.

Value

The optimal clipping bound/function is computed.
### Methods

- **clip = "numeric", ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asMSE", neighbor**
  - optimal clipping bound for asymptotic mean square error.

- **clip = "numeric", ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asMSE", neighbor = "Av1CondTotalVarNeighborhood"**
  - optimal clipping bound for asymptotic mean square error.

- **clip = "numeric", ErrorL2deriv = "EuclRandVariable", Regressor = "Distribution", risk = "asMSE", neighbor = "Neighborhood"**
  - optimal clipping bound for asymptotic mean square error.

- **clip = "numeric", ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "UncondNeighborhood"**
  - optimal clipping bound for asymptotic under-/overshoot risk.

- **clip = "numeric", ErrorL2deriv = "UnivariateDistribution", Regressor = "numeric", risk = "asUnOvShoot", neighbor = "CondNeighborhood"**
  - optimal clipping function for asymptotic under-/overshoot risk.

### Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

### References


### See Also

- ContIC-class, TotalVarIC-class, Av1CondContIC-class, Av2CondContIC-class, Av1CondTotalVarIC-class, CondContIC-class, CondTotalVarIC-class

---

### Description

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

### Usage

```r
getInfGammaRegTS(ErrorL2deriv, Regressor, risk, neighbor, ...)
```

```
## S4 method for signature
## 'UnivariateDistribution,UnivariateDistribution,asMSE,ContNeighborhood'
getInfGammaRegTS(
    ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent, clip)
```
getInfGammaRegTS

## S4 method for signature
## 'UnivariateDistribution,'
## UnivariateDistribution,
## asMSE,
## Av1CondContNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,'
## UnivariateDistribution,
## asMSE,
## Av1CondTotalVarNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,'
## MultivariateDistribution,
## asMSE,
## ContNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,'
## MultivariateDistribution,
## asMSE,
## Av1CondContNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,'
## MultivariateDistribution,
## asMSE,
## Av1CondTotalVarNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,'
## Distribution, asMSE, Av2CondContNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, z.comp, stand, cent, clip)

## S4 method for signature
## 'RealRandVariable,'
## Distribution, asMSE, ContNeighborhood'
getInfGammaRegTS(
getInfGammaRegTS

getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, ErrorDistr, stand, cent, clip)

## S4 method for signature
## 'RealRandVariable,Distribution,asMSE,Av1CondContNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, ErrorDistr, stand, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,
## UnivariateDistribution,
## asUnOvShoot,
## ContNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,
## UnivariateDistribution,
## asUnOvShoot,
## TotalVarNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, cent, clip)

## S4 method for signature
## 'UnivariateDistribution,numeric,asUnOvShoot,CondContNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, clip)

## S4 method for signature
## 'UnivariateDistribution,numeric,asUnOvShoot,CondTotalVarNeighborhood'
getInfGammaRegTS(
  ErrorL2deriv, Regressor, risk, neighbor, clip)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorL2deriv</td>
<td>L2-derivative of ErrorDistr.</td>
</tr>
<tr>
<td>Regressor</td>
<td>regressor.</td>
</tr>
<tr>
<td>risk</td>
<td>object of class &quot;RiskType&quot;.</td>
</tr>
<tr>
<td>neighbor</td>
<td>object of class &quot;Neighborhood&quot;.</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters.</td>
</tr>
<tr>
<td>clip</td>
<td>optimal clipping bound.</td>
</tr>
<tr>
<td>cent</td>
<td>optimal centering constant/function.</td>
</tr>
<tr>
<td>stand</td>
<td>standardizing matrix.</td>
</tr>
<tr>
<td>z.comp</td>
<td>which components of the centering constant/function have to be computed.</td>
</tr>
<tr>
<td>ErrorDistr</td>
<td>error distribution.</td>
</tr>
</tbody>
</table>
Details
The function is used in case of asymptotic G-risks; confer Ruckdeschel and Rieder (2004).

Methods

ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asMSE", neighbor = "ContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asMSE", neighbor = "Av1CondContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asMSE", neighbor = "Av1CondTotalVarNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "MultivariateDistribution", risk = "asMSE", neighbor = "ContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "MultivariateDistribution", risk = "asMSE", neighbor = "Av1CondContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "MultivariateDistribution", risk = "asMSE", neighbor = "Av1CondTotalVarNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asMSE", neighbor = "Av2CondContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", risk = "asMSE", neighbor = "ContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", risk = "asMSE", neighbor = "Av1CondContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "ContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "Av1CondContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "numeric", risk = "asUnOvShoot", neighbor = "CondContNeighborhood" used by getInfClipRegTS.

ErrorL2deriv = "UnivariateDistribution", Regressor = "numeric", risk = "asUnOvShoot", neighbor = "CondTotalVarNeighborhood" used by getInfClipRegTS.

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

References
**getInfRobRegTypeIC**  
*Generic Function for the Computation of Optimally Robust Regression-Type ICs*

**Description**

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

**Usage**

```r
getInfRobRegTypeIC

## S4 method for signature
## 'UnivariateDistribution,
##   UnivariateDistribution,
##   asBias,
##   ContNeighborhood'
getInfRobRegTypeIC(
  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
  RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,
##   UnivariateDistribution,
##   asBias,
##   Av1CondContNeighborhood'
getInfRobRegTypeIC(
  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
  RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,
##   UnivariateDistribution,
##   asBias,
##   Av1CondTotalVarNeighborhood'
getInfRobRegTypeIC(
  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
  RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asBias,Av2CondContNeighborhood'
```

**See Also**

asMSE-class, asUn0vShoot-class, ContIC-class, Av1CondContIC-class, Av2CondContIC-class, Av1CondTotalVarIC-class, CondContIC-class, CondTotalVarIC-class
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asCov,ContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asCov,TotalVarNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asCov,CondContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asCov,CondTotalVarNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asCov,A1CondContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asCov,A2CondContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asCov,A1CondTotalVarNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo)
getInfRobRegTypeIC

getInfRobRegTypeIC(
    ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
    RegSymm, Finfo, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asGRisk,ContNeighborhood'
getInfRobRegTypeIC(
    ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
    RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asGRisk,Av1CondContNeighborhood'
getInfRobRegTypeIC(
    ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
    RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,Distribution,asGRisk,Av2CondContNeighborhood'
getInfRobRegTypeIC(
    ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
    RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,
## Distribution,
## asGRisk,
## Av1CondTotalVarNeighborhood'
getInfRobRegTypeIC(
    ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
    RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,
## MultivariateDistribution,
## asBias,
## ContNeighborhood'
getInfRobRegTypeIC(
    ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
    RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,
## MultivariateDistribution,
## asBias,
## Av1CondContNeighborhood'
getInfRobRegTypeIC(
    ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,
    RegSymm, Finfo, trafo, upper, maxiter, tol, warn)
getInfRobRegTypeIC

## S4 method for signature
## 'UnivariateDistribution,
## MultivariateDistribution,
## asBias,
## Av1CondTotalVarNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'RealRandVariable,Distribution,asBias,ContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorSymm, RegSymm,  ErrorDistr, ErrorL2derivSymm, ErrorL2derivDistrSymm, Finfo, trafo,  upper, z.start, A.start, maxiter, tol, warn)

## S4 method for signature
## 'RealRandVariable,Distribution,asBias,Av1CondContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorSymm, RegSymm,  ErrorDistr, ErrorL2derivSymm, ErrorL2derivDistrSymm, Finfo, trafo,  upper, z.start, A.start, maxiter, tol, warn)

## S4 method for signature
## 'RealRandVariable,Distribution,asCov,ContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorDistr, Finfo, trafo)

## S4 method for signature
## 'RealRandVariable,Distribution,asCov,Av1CondContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorDistr, Finfo, trafo)

## S4 method for signature
## 'RealRandVariable,Distribution,asGRisk,ContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorSymm, RegSymm,  ErrorDistr, ErrorL2derivSymm, ErrorL2derivDistrSymm, Finfo, trafo,  upper, z.start, A.start, maxiter, tol, warn)

## S4 method for signature
## 'RealRandVariable,Distribution,asGRisk,Av1CondContNeighborhood'
getInfRobRegTypeIC(  ErrorL2deriv, Regressor, risk, neighbor, ErrorSymm, RegSymm,  ErrorDistr, ErrorL2derivSymm, ErrorL2derivDistrSymm, Finfo, trafo,  upper, z.start, A.start, maxiter, tol, warn)
getInfRobRegTypeIC

## S4 method for signature
## 'UnivariateDistribution,'  
##   UnivariateDistribution,
##   asUnOvShoot,
##   'UncondNeighborhood'
getInfRobRegTypeIC(  
  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  
  RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

## S4 method for signature
## 'UnivariateDistribution,'  
##   UnivariateDistribution,
##   asUnOvShoot,
##   'CondNeighborhood'
getInfRobRegTypeIC(  
  ErrorL2deriv, Regressor, risk, neighbor, ErrorL2derivDistrSymm,  
  RegSymm, Finfo, trafo, upper, maxiter, tol, warn)

### Arguments

- **ErrorL2deriv**: L2-derivative of `ErrorDistr`.
- **Regressor**: regressor.
- **risk**: object of class "RiskType".
- **neighbor**: object of class "Neighborhood".
- **...**: additional parameters.
- **ErrorSymm**: symmetry of `ErrorDistr`.
- **ErrorL2derivDistrSymm**: symmetry of `ErrorL2derivDistr`.
- **RegSymm**: symmetry of `RegDistr`.
- **ErrorDistr**: error distribution.
- **ErrorL2derivSymm**: symmetry of `ErrorL2deriv`.
- **Finfo**: Fisher information 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.
- **z.start**: initial value for the centering constant/function.
- **A.start**: initial value for the standardizing matrix.

### Value

The optimally robust IC is computed.
Methods

- `ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asBias", neighbor = "ContNeighborhood"` computes the bias optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asBias", neighbor = "Av1CondContNeighborhood"` computes the bias optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asBias", neighbor = "Av1CondTotalVarNeighborhood"` computes the bias optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asBias", neighbor = "Av2CondContNeighborhood"` computes the bias optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asCov", neighbor = "ContNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asCov", neighbor = "TotalVarNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asCov", neighbor = "CondContNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asCov", neighbor = "CondTotalVarNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asCov", neighbor = "Av1CondContNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asCov", neighbor = "Av2CondContNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", risk = "asCov", neighbor = "Av1CondTotalVarNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "UnivariateDistribution", Regressor = "MultivariateDistribution", risk = "asBias", neighbor = "TotalVarNeighborhood"` computes the optimally robust influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", risk = "asBias", neighbor = "ContNeighborhood"` computes the bias optimal influence curve for L2 differentiable regression-type families.
- `ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", risk = "asCov", neighbor = "ContNeighborhood"` computes the classical optimal influence curve for L2 differentiable regression-type families.
getInfStandRegTS

ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", risk = "asCov", neighbor = "Av1CondContNeighborhood"
computes the classical optimal influence curve for L2 differentiable regression-type families.

ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", risk = "asGRisk", neighbor = "ContNeighborhood"
computes the optimally robust influence curve for L2 differentiable regression-type families.

ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", risk = "asGRisk", neighbor = "Av1CondContNeighborhood"
computes the optimally robust influence curve for L2 differentiable regression-type families.

ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "UncondNeighborhood"
computes the optimally robust influence curve for L2 differentiable regression-type families.

ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", risk = "asUnOvShoot", neighbor = "CondNeighborhood"
computes the optimally robust influence curve for L2 differentiable regression-type families.

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

References

See Also
InfRobRegTypeModel-class

getInfStandRegTS

**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
getInfStandRegTS(ErrorL2deriv, Regressor, neighbor, ...)

## S4 method for signature
## 'UnivariateDistribution,UnivariateDistribution,ContNeighborhood'
getInfStandRegTS(
    ErrorL2deriv, Regressor, neighbor, z.comp, clip, cent, stand, trafo)

## S4 method for signature
## 'UnivariateDistribution,UnivariateDistribution,TotalVarNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, clip, cent)

## S4 method for signature
## 'UnivariateDistribution,
##  UnivariateDistribution,
##  CondTotalVarNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, clip, cent)

## S4 method for signature
## 'UnivariateDistribution,
##  UnivariateDistribution,
##  Av1CondContNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, z.comp, clip, cent, stand, trafo)

## S4 method for signature
## 'UnivariateDistribution,
##  UnivariateDistribution,
##  Av1CondTotalVarNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, z.comp, clip, cent, stand, trafo)

## S4 method for signature
## 'UnivariateDistribution,
##  MultivariateDistribution,ContNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, z.comp, clip, cent, stand, trafo)

## S4 method for signature
## 'UnivariateDistribution,
##  MultivariateDistribution,
##  Av1CondContNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, z.comp, clip, cent, stand, trafo)

## S4 method for signature
## 'UnivariateDistribution,
##  MultivariateDistribution,
##  Av1CondTotalVarNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, z.comp, clip, cent, stand, trafo)

## S4 method for signature
## 'UnivariateDistribution,Distribution,Av2CondContNeighborhood'
getInfStandRegTS(  
  ErrorL2deriv, Regressor, neighbor, z.comp, clip, cent, stand, trafo)
## S4 method for signature 'RealRandVariable,Distribution,ContNeighborhood'
getInfStandRegTS(
  ErrorL2deriv, Regressor, neighbor, ErrorDistr, A.comp, stand, clip, 
  cent, trafo)

## S4 method for signature
## 'RealRandVariable,Distribution,Av1CondContNeighborhood'
getInfStandRegTS(
  ErrorL2deriv, Regressor, neighbor, ErrorDistr, A.comp, stand, clip, 
  cent, trafo)

### Arguments

**ErrorL2deriv**  
L2-derivative of ErrorDistr.

**Regressor**  
regressor.

**neighbor**  
object of class "Neighborhood".

**...**  
additional parameters.

**ErrorDistr**  
error distribution.

**clip**  
optimal clipping bound/function.

**cent**  
optimal centering constant/function.

**stand**  
standardizing clipping bound.

**z.comp**  
which components of the centering constant/function have to be computed.

**A.comp**  
which components of the standardizing matrix have to be computed.

**trafo**  
matrix: transformation of the parameter.

### Value

The standardizing matrix is computed.

### Methods

**ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "ContNeighborhood"**
computes standardizing matrix.

**ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "TotalVarNeighborhood"**
computes standardizing constant.

**ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "CondTotalVarNeighborhood"**
computes standardizing constant.

**ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "Av1CondContNeighborhood"**
computes standardizing matrix.

**ErrorL2deriv = "UnivariateDistribution", Regressor = "UnivariateDistribution", neighbor = "Av1CondTotalVarNeighborhood"**
computes standardizing matrix.

**ErrorL2deriv = "UnivariateDistribution", Regressor = "MultivariateDistribution", neighbor = "ContNeighborhood"**
computes standardizing matrix.
ErrorL2deriv = "UnivariateDistribution", Regressor = "MultivariateDistribution", neighbor = "Av1CondTotalVarNeighborhood"
computes standardizing matrix.

ErrorL2deriv = "UnivariateDistribution", Regressor = "Distribution", neighbor = "Av2CondContNeighborhood"
computes standardizing matrix.

ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", neighbor = "ContNeighborhood"
computes standardizing matrix.

ErrorL2deriv = "RealRandVariable", Regressor = "Distribution", neighbor = "Av1CondContNeighborhood"
computes standardizing matrix.

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

References

See Also
ContIC-class, TotalVarIC-class, Av1CondContIC-class, Av2CondContIC-class, Av1CondTotalVarIC-class, CondContIC, CondTotalVarIC

InfRobRegTypeModel Generating function for InfRobRegTypeModel-class

Description
Generates an object of class "InfRobRegTypeModel".

Usage
InfRobRegTypeModel(center = L2RegTypeFamily(), neighbor = ContNeighborhood())

Arguments
center object of class "L2RegTypeFamily"
neighbor object of class "Neighborhood"

Value
Object of class "InfRobRegTypeModel"
**InfRobRegTypeModel-class**

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

**References**

**See Also**
InfRobRegTypeModel-class

**Examples**

```r
InfRobRegTypeModel()
```

```r
## The function is currently defined as
function(center = L2RegTypeFamily(), neighbor = ContNeighborhood()) {
  new("InfRobRegTypeModel", center = center, neighbor = neighbor)
}
```

**Description**

Class of robust regression-type models with infinitesimal (conditional or 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("InfRobRegTypeModel", ...). More frequently they are created via the generating function InfRobRegTypeModel.

**Slots**

- `center`: Object of class "L2RegTypeFamily".
- `neighbor`: Object of class "Neighborhood".

**Extends**

Class "RobModel", directly.
Methods

```
neighbor<- signature(object = "InfRobRegTypeModel"): replacement function for slot neighbor.
show signature(object = "InfRobRegTypeModel")
```

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`L2RegTypeFamily-class, Neighborhood-class, InfRobRegTypeModel`

Examples

```
new("InfRobRegTypeModel")
```

Description

Generates an object of class "RegTypeFamily".

Usage

```
L2RegTypeFamily(name, distribution = LMCondDistribution(), distrSymm,
    main = 0, nuisance, trafo, param, props = character(0),
    L2deriv = EuclRandVarList(EuclRandVariable(
        Map = list(function(x) {x[1] * x[2]}),
        Domain = EuclideanSpace(dimension = 2),
        dimension = 1)),
    ErrorDistr = Norm(), ErrorSymm, RegDistr = Norm(), RegSymm,
    Regressor = RealRandVariable(Map = list(function(x) {x}), Domain = Reals()),
    ErrorL2deriv = EuclRandVarList(RealRandVariable(Map = list(function(x) {x}),
        Domain = Reals())),
    ErrorL2derivSymm, ErrorL2derivDistr, ErrorL2derivDistrSymm, FisherInfo)
```
L2RegTypeFamily

Arguments

- `name` name of the family
- `distribution` conditional distribution (given the regressor)
- `distrSymm` symmetry of distribution
- `ErrorDistr` error distribution
- `ErrorSymm` object of class "DistributionSymmetry": symmetry of ErrorDistr
- `main` main parameter
- `nuisance` optional nuisance parameter
- `trafo` matrix: optional transformation of the parameter
- `param` parameter of the family
- `props` properties of the family
- `RegDistr` regressor distribution
- `RegSymm` object of class "DistributionSymmetry": symmetry of RegDistr
- `Regressor` regressor
- `L2deriv` object of class "EuclRandVariable": L2 derivative
- `ErrorL2deriv` object of class "EuclRandVariable": L2 derivative of ErrorDistr
- `ErrorL2derivDistr` distribution of ErrorL2deriv
- `ErrorL2derivSymm` object of class "FunSymmList": symmetry of ErrorL2deriv
- `ErrorL2derivDistrSymm` object of class "DistrSymmList": symmetry of ErrorL2derivDistr
- `FisherInfo` Fisher information matrix

Details

If `name` is missing, the default “L2 differentiable regression type family” is used. If `param` is missing, the parameter is created via `main`, `nuisance` and `trafo` as described in `ParamFamParameter`. In case `distrSymm`, `ErrorSymm`, `RegSymm` is missing, they are set to `NoSymmetry()`. If `FisherInfo` is missing, it is computed via numerical integration.

Value

Object of class "L2RegTypeFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also

L2RegTypeFamily-class

Examples

L2RegTypeFamily()

L2RegTypeFamily-class  L2 differentiable parametric regression-type family

Description

Class for L2 differentiable parametric regression-type families.

Objects from the Class

Objects can be created by calls of the form new("L2RegTypeFamily", ...). More frequently they are created via the generating function L2RegTypeFamily.

Slots

L2deriv: Object of class "EuclRandVarList": L2 derivative.
ErrorL2deriv: Object of class "EuclRandVarList": L2 derivative of ErrorDistr.
ErrorL2derivSymm: Object of class "FunSymmList": symmetry of ErrorL2deriv.
ErrorL2derivDistr: Object of class "DistrList": distribution of ErrorL2deriv.
ErrorL2derivDistrSymm: Object of class "DistrSymmList": symmetry of ErrorL2derivDistr.
FisherInfo: Object of class "PosDefSymmMatrix": Fisher information.
ErrorDistr: Object of class "Distribution": error distribution.
ErrorSymm: Object of class "DistributionSymmetry": symmetry of ErrorDistr.
RegDistr: Object of class "Distribution": regressor distribution.
RegSymm: Object of class "DistributionSymmetry": symmetry of RegDistr.
Regressor: Object of class "EuclRandVariable": regressor.
param: Object of class "ParamFamParameter": parameter of the family.
props: Object of class "character": properties of the family.
name: Object of class "character": name of the family.
distribution: Object of class "CondDistribution": conditional distribution given the regressor.
distrSymm: Object of class "DistributionSymmetry": symmetry of distribution.

Extends

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

**L2deriv** signature(object = "L2RegTypeFamily"): accessor function for slot L2deriv.

**FisherInfo** signature(object = "L2RegTypeFamily"): accessor function for slot FisherInfo.

**ErrorL2deriv** signature(object = "L2RegTypeFamily"): accessor function for slot ErrorL2deriv.

**ErrorL2derivDistr** signature(object = "L2RegTypeFamily"): accessor function for slot ErrorL2derivDistr.

**ErrorL2derivSymm** signature(object = "L2RegTypeFamily"): accessor function for slot ErrorL2derivSymm.

**ErrorL2derivDistrSymm** signature(object = "L2RegTypeFamily"): accessor function for slot ErrorL2derivDistrSymm.

**checkL2deriv** signature(object = "L2RegTypeFamily"): check centering of L2deriv and compute precision of Fisher information.

**checkIC** signature(IC = "IC", L2Fam = "missing"): check centering and Fisher consistency of IC assuming the L2-differentiable regression-type family which can be created via the slot CallL2Fam of IC.

**checkIC** signature(IC = "IC", L2Fam = "L2RegTypeFamily"): check centering and Fisher consistency of IC assuming the L2-differentiable regression-type family L2Fam.

**E** signature(object = "L2RegTypeFamily", fun = "EuclRandVariable", cond = "missing"): expectation of fun under object.

**E** signature(object = "L2RegTypeFamily", fun = "EuclRandMatrix", cond = "missing"): expectation of fun under object.

**E** signature(object = "L2RegTypeFamily", fun = "EuclRandVarList", cond = "missing"): expectation of fun under object.

**show** signature(object = "L2RegTypeFamily")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

RegTypeFamily-class

Examples

new("L2RegTypeFamily")
Methods for function leastFavorableRadius in Package 'ROptRegTS'

Description

Methods for function leastFavorableRadius in package ROptRegTS.

Methods

L2Fam = "L2RegTypeFamily", neighbor = "Neighborhood", risk = "asGRisk" The least favorable radius and the corresponding inefficiency are computed.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

leastFavorableRadius

NormLinRegFamily Generating function for linear regression family

Description

Generates an object of class "L2RegTypeFamily" which represents a linear regression family with standard normal distributed errors and random regressor.

Usage

NormLinRegFamily(theta, trafo, RegDistr = Norm(), RegSymm, Reg2Mom)

Arguments

theta linear regression parameter
trafo matrix: transformation of the parameter
RegDistr regressor distribution
RegSymm symmetry of the regressor distribution
Reg2Mom second moment matrix of regressor

Details

In case theta is missing, it is set to 0. If Reg2Mom is missing, it is computed via E.
**Value**

Object of class "L2RegTypeFamily"

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

`L2RegTypeFamily-class`

**Examples**

```r
(LM1 <- NormLinRegFamily(Reg2Mom = matrix(1)))
Map(L2deriv(LM1)[[1]])
FisherInfo(LM1)
checkL2deriv(LM1)
```

---

**Description**

Generates an object of class "L2RegTypeFamily" which represents a linear regression family with standard normal distributed errors and random regressor where the intercept is unknown.

**Usage**

```
NormLinRegInterceptFamily(theta, intercept = 0, trafo, RegDistr = Norm(),
RegSymm, Reg2Mom, nuisance = FALSE)
```

**Arguments**

- `theta`: linear regression parameter
- `intercept`: intercept parameter
- `trafo`: matrix: transformation of the parameter
- `RegDistr`: regressor distribution
- `RegSymm`: symmetry of the regressor distribution
- `Reg2Mom`: second moment matrix of regressor
- `nuisance`: logical: is intercept nuisance parameter
Details

In case \( \theta \) is missing, it is set to 0. If \( \text{Reg}2\text{Mom} \) is missing, it is computed via \( E \).

Value

Object of class "L2RegTypeFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2RegTypeFamily-class

Examples

```r
(LM1 <- NormLinRegInterceptFamily(Reg2Mom = matrix(1)))
Map(L2deriv(LM1)[[1]])
FisherInfo(LM1)
checkL2deriv(LM1)
```

---

**NormLinRegScaleFamily**  
*Generating Function for Linear Regression Family with Unknown Scale*

Description

Generates an object of class "L2RegTypeFamily" which represents a linear regression family with standard normal distributed errors and random regressor where the scale of the error distribution is unknown.

Usage

```r
NormLinRegScaleFamily(theta, scale = 1, trafo, RegDistr = Norm(),
                       RegSymm, Reg2Mom, nuisance = FALSE)
```
Arguments

theta  linear regression parameter
scale  scale parameter for error distribution
trafo  matrix: transformation of the parameter
RegDistr regressor distribution
RegSymm symmetry of the regressor distribution
Reg2Mom second moment matrix of regressor
nuisance logical: is scale nuisance parameter

Details

In case theta is missing, it is set to 0. If Reg2Mom is missing, it is computed via E.

Value

Object of class "L2RegTypeFamily"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2RegTypeFamily-class

Examples

```r
(LM1 <- NormLinRegScaleFamily(Reg2Mom = matrix(1)))
Map(L2deriv(LM1)[[1]])
FisherInfo(LM1)
checkL2deriv(LM1)
```
Description

Methods for function `optIC` in package `ROptRegTS`.

Usage

```r
## S4 method for signature 'L2RegTypeFamily,asCov'
optIC(model, risk)

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

## S4 method for signature 'FixRobRegTypeModel,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".
- `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 boundary curve. For more details we refer to Subsections 12.1.3 and 12.2.3 of Kohl (2005).
Some optimally robust IC is computed.

Methods

- `model = "L2RegTypeFamily", risk = "asCov"` computes classical optimal influence curve for L2 differentiable regression-type families.
- `model = "InfRobRegTypeModel", risk = "asRisk"` computes optimally robust influence curve for robust regression-type models with infinitesimal neighborhoods and various asymptotic risks.
- `model = "InfRobRegTypeModel", risk = "asUnOvShoot"` computes optimally robust influence curve for robust regression-type models with infinitesimal neighborhoods and asymptotic under-/overshoot risk.
- `model = "FixRobRegTypeModel", risk = "fiUnOvShoot"` computes optimally robust influence curve for robust regression-type models with fixed neighborhoods and finite-sample under-/overshoot risk.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

- `optIC`

Methods for function `radiusMinimaxIC` in package `ROptRegTS`.

Methods

- `L2Fam = "L2RegTypeFamily", neighbor = "Neighborhood", risk = "asGRisk"` computation of the radius minimax IC for an L2 differentiable regression-type family.
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

See Also
radiusMinimaxIC

RegTypeFamily

Generating function for RegTypeFamily-class

Description
Generates an object of class "RegTypeFamily".

Usage
RegTypeFamily(name, distribution = LMCondDistribution(), distrSymm,
    ErrorDistr = Norm(), ErrorSymm, main = 0, nuisance, trafo,
    param, props = character(0), RegDistr = Norm(), RegSymm,
    Regressor = RealRandVariable(c(function(x) {x}, Domain = Reals())))

Arguments

name name of the family
distribution conditional distribution (given the regressor)
distrSymm symmetry of distribution
ErrorDistr error distribution
ErrorSymm symmetry of ErrorDistr
main main parameter
nuisance optional nuisance parameter
trafo matrix: optional transformation of the parameter
param parameter of the family
props properties of the family
RegDistr regressor distribution
RegSymm symmetry of RegDistr
Regressor regressor

Details
If name is missing, the default “regression type family” is used. If param is missing, the parameter is created via main, nuisance and trafo as described in ParamFamParameter. In case distrSymm, ErrorSymm or RegSymm is missing, they are set to NoSymmetry().
**RegTypeFamily-class**

**Value**
Object of class "RegTypeFamily"

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

**See Also**
ParamFamily-class

**Examples**
RegTypeFamily()

---

**Description**
Class for parametric regression-type families.

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

**Slots**
- ErrorDistr: object of class "Distribution": error distribution.
- ErrorSymm: object of class "DistributionSymmetry": symmetry of the error distribution.
- RegDistr: object of class "Distribution": regressor distribution.
- RegSymm: object of class "DistributionSymmetry": symmetry of the regressor distribution.
- Regressor: object of class "EuclRandVariable": regressor.
- param: object of class "ParamFamParameter": parameter of the family.
- props: object of class "character": properties of the family.
- name: object of class "character": name of the family.
- distribution: object of class "CondDistribution": distribution given the regressor.

**Extends**
Class "ParamFamily", directly.
Class "ProbFamily", by class "ParamFamily".
Methods

ErrorDistr signature(object = "RegTypeFamily"): accessor function for slot ErrorDistr.
ErrorSymm signature(object = "RegTypeFamily"): accessor function for slot ErrorSymm.
RegDistr signature(object = "RegTypeFamily"): accessor function for slot RegDistr.
Regressor signature(object = "RegTypeFamily"): accessor function for slot Regressor.
RegSymm signature(object = "RegTypeFamily"): accessor function for slot RegSymm.
show signature(object = "RegTypeFamily")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

ParamFamily-class

Examples

new("RegTypeFamily")
Index

*Topic classes
Av1CondContIC-class, 4
Av1CondContNeighborhood-class, 7
Av1CondNeighborhood-class, 8
Av1CondTotalVarIC-class, 10
Av1CondTotalVarNeighborhood-class, 13
Av2CondContIC-class, 15
Av2CondContNeighborhood-class, 18
Av2CondNeighborhood-class, 19
AvCondNeighborhood-class, 20
CondContIC-class, 22
CondContNeighborhood-class, 25
CondIC-class, 27
CondNeighborhood-class, 28
CondTotalVarIC-class, 31
CondTotalVarNeighborhood-class, 34
FixRobRegTypeModel-class, 36
InfRobRegTypeModel-class, 65
L2RegTypeFamily-class, 66
L2RegTypeFamily-class, 68
NormLinRegFamily, 70
NormLinRegInterceptFamily, 71
NormLinRegScaleFamily, 72
RegTypeFamily, 76
RegTypeFamily-class, 77

*Topic robust
Av1CondContIC, 3
Av1CondContIC-class, 4
Av1CondContNeighborhood, 6
Av1CondContNeighborhood-class, 7
Av1CondNeighborhood-class, 8
Av1CondTotalVarIC-class, 12
Av1CondTotalVarNeighborhood-class, 13
Av2CondContNeighborhood, 17
Av2CondContNeighborhood-class, 18
Av2CondNeighborhood-class, 19
AvCondNeighborhood-class, 20
CondContIC, 21
CondContIC-class, 22
CondContNeighborhood, 24
CondContNeighborhood-class, 25
CondIC, 26
CondIC-class, 27
CondNeighborhood-class, 28
CondTotalVarIC, 29
CondTotalVarIC-class, 31
CondTotalVarNeighborhood, 33
CondTotalVarNeighborhood-class, 34
FixRobRegTypeModel, 35
FixRobRegTypeModel-class, 36
generateIC-methods, 37
getAsRiskRegTS, 37
getFiRiskRegTS, 41
getFixClipRegTS, 43
getFixRobRegTypeIC, 44
getIneffDiff-methods, 46
getInfCentRegTS, 46
getInfClipRegTS, 49
getInfGammaRegTS, 51
getInfRobRegTypeIC, 55
getInfStandRegTS, 61
InfRobRegTypeModel, 64
InfRobRegTypeModel-class, 65
L2RegTypeFamily, 66
L2RegTypeFamily-class, 68
leastFavorableRadius-methods, 70
NormLinRegFamily, 70
NormLinRegInterceptFamily, 71
NormLinRegScaleFamily, 72
optIC-methods, 74
radiusMinimaxIC-methods, 75
RegTypeFamily, 76
RegTypeFamily-class, 77

Av1CondContIC, 3, 5
Av1CondContIC-class, 4
Av1CondContNeighborhood, 6
Av1CondContNeighborhood-class, 7
Av1CondContNeighborhood-class, 8
Av1CondTotalVarIC, 9, 11
Av1CondTotalVarIC-class, 10
Av1CondTotalVarNeighborhood, 12
Av1CondTotalVarNeighborhood-class, 13
Av2CondContIC, 14, 16
Av2CondContIC-class, 15
Av2CondContNeighborhood, 17
Av2CondContNeighborhood-class, 18
Av2CondNeighborhood-class, 19
AvCondNeighborhood-class, 20

CallL2Fam<-, Av1CondContIC-method
(Av1CondContIC-class), 4

CallL2Fam<-, Av2CondContIC-method
(Av2CondContIC-class), 15

CallL2Fam<-, CondContIC-method
(CondContIC-class), 22

CallL2Fam<-, CondContIC-method
(CondContIC-class), 27

CallL2Fam<-, CondTotalVarIC-method
(CondTotalVarIC-class), 31
clipLo<-, Av1CondTotalVarIC-method
(Av1CondTotalVarIC-class), 10
clipLo<-, CondTotalVarIC-method
(CondTotalVarIC-class), 31
clipUp, Av1CondTotalVarIC-method
(Av1CondTotalVarIC-class), 10
clipUp, CondTotalVarIC-method
(CondTotalVarIC-class), 31
clipUp<-, Av1CondTotalVarIC-method
(Av1CondTotalVarIC-class), 10
clipUp<-, CondTotalVarIC-method
(CondTotalVarIC-class), 31
CondContIC, 21, 24, 64
CondContIC-class, 22
CondContNeighborhood, 24, 26
CondContNeighborhood-class, 25
CondIC, 26
CondIC-class, 27
CondNeighborhood-class, 28
CondTotalVarIC, 29, 32, 64
CondTotalVarIC-class, 31
CondTotalVarNeighborhood, 33, 34
CondTotalVarNeighborhood-class, 34
E, L2RegTypeFamily, EuclRandMatrix, missing-method
(L2RegTypeFamily-class), 68
E, L2RegTypeFamily, EuclRandVariable, missing-method
(L2RegTypeFamily-class), 68
E, L2RegTypeFamily, EuclRandVarList, missing-method
(L2RegTypeFamily-class), 68
ErrorDistr (RegTypeFamily-class), 77
ErrorDistr, RegTypeFamily-method
(RegTypeFamily-class), 77
ErrorL2deriv (L2RegTypeFamily-class), 68
ErrorL2deriv, L2RegTypeFamily-method
(L2RegTypeFamily-class), 68
ErrorL2derivDistr
(L2RegTypeFamily-class), 68
ErrorL2derivDistr, L2RegTypeFamily-method
(L2RegTypeFamily-class), 68
ErrorL2derivDistrSymm
(L2RegTypeFamily-class), 68
ErrorL2derivDistrSymm, L2RegTypeFamily-method
(L2RegTypeFamily-class), 68
ErrorL2derivSymm
(L2RegTypeFamily-class), 68
ErrorL2derivSymm, L2RegTypeFamily-method
(L2RegTypeFamily-class), 68
ErrorSymm (RegTypeFamily-class), 77
ErrorSymm, RegTypeFamily-method
(RegTypeFamily-class), 77
FisherInfo, L2RegTypeFamily-method
(L2RegTypeFamily-class), 68
FixRobRegTypeModel, 35, 36
FixRobRegTypeModel-class, 36
generateIC, 37
generateIC, Av1CondContNeighborhood, L2RegTypeFamily-method
(Av1CondContIC-class), 4
generateIC, Av1CondTotalVarNeighborhood, L2RegTypeFamily-method
(Av1CondTotalVarIC-class), 10
generateIC, Av2CondContNeighborhood, L2RegTypeFamily-method
(Av2CondIC-class), 15
generateIC, CondContNeighborhood, L2RegTypeFamily-method
(CondContIC-class), 22
generateIC, CondTotalVarNeighborhood, L2RegTypeFamily-method
(CondTotalVarIC-class), 31
generateIC, CondNeighborhood, L2RegTypeFamily-method
(generateIC-methods), 37
generateIC, TotalVarNeighborhood, L2RegTypeFamily-method
(generateIC-methods), 37
generateIC-methods, 37
getAsRiskRegTS, 37
getAsRiskRegTS, asBias, RealRandVariable, Distribution, Av1CondTotalVarIC-method
(getAsRiskRegTS), 37
getAsRiskRegTS, asBias, RealRandVariable, ContNeighborhood-method
(getAsRiskRegTS), 37
getAsRiskRegTS, asBias, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asBias, UnivariateDistribution, Multivariate
(getAsRiskRegTS), 37
getAsRiskRegTS, asBias, UnivariateDistribution, Multivariate
(getAsRiskRegTS), 37
getAsRiskRegTS, asBias, UnivariateDistribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asBias, UnivariateDistribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, EuclRandVariable, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
getAsRiskRegTS, asMSE, UnivariateDistribution, Distribution, Neighbo-
(getAsRiskRegTS), 37
show, $Av1CondTotalVarIC$-method
   (Av1CondTotalVarIC-class), 10
show, $Av2CondContIC$-method
   (Av2CondContIC-class), 15
show, $AvCondNeighborhood$-method
   (AvCondNeighborhood-class), 20
show, $CondContIC$-method
   (CondContIC-class), 22
show, $CondIC$-method (CondIC-class), 27
show, $CondNeighborhood$-method
   (CondNeighborhood-class), 28
show, $CondTotalVarIC$-method
   (CondTotalVarIC-class), 31
show, $FixRobRegTypeModel$-method
   (FixRobRegTypeModel-class), 36
show, $InfRobRegTypeModel$-method
   (InfRobRegTypeModel-class), 65
show, $L2RegTypeFamily$-method
   (L2RegTypeFamily-class), 68
show, $RegTypeFamily$-method
   (RegTypeFamily-class), 77
stand, $Av1CondContIC$-method
   (Av1CondContIC-class), 4
stand, $Av1CondTotalVarIC$-method
   (Av1CondTotalVarIC-class), 10
stand, $Av2CondContIC$-method
   (Av2CondContIC-class), 15
stand, $CondContIC$-method
   (CondContIC-class), 22
stand, $CondTotalVarIC$-method
   (CondTotalVarIC-class), 31
stand<-, $Av1CondContIC$-method
   (Av1CondContIC-class), 4
stand<-, $Av1CondTotalVarIC$-method
   (Av1CondTotalVarIC-class), 10
stand<-, $Av2CondContIC$-method
   (Av2CondContIC-class), 15
stand<-, $CondContIC$-method
   (CondContIC-class), 22
stand<-, $CondTotalVarIC$-method
   (CondTotalVarIC-class), 31