Package ‘RobAStBase’

April 23, 2017

Version 1.0.1
Date 2017-04-23
Title Robust Asymptotic Statistics
Description Base S4-classes and functions for robust asymptotic statistics.
Depends R(>= 2.14.0), methods, rrcov, distr(>= 2.5.2), distrEx(>= 2.5), distrMod(>= 2.5.2), RandVar(>= 0.9.2)
Suggests ROptEst, RUnit (>= 0.4.26)
Imports startupmsg
ByteCompile yes
License LGPL-3
Encoding latin1
URL http://robast.r-forge.r-project.org/
LastChangedDate {$LastChangedDate: 2017-04-23 09:43:24 +0200 (So, 23. Apr 2017) $}
LastChangedRevision {$LastChangedRevision: 935 $}
SVNRevision 930
NeedsCompilation no
Author Matthias Kohl [cre, cph, aut],
Peter Ruckdeschel [aut, cph],
Mykhailo Pupashenko [ctb] (contributed wrapper functions for diagnostic plots),
Gerald Kroisandt [ctb] (contributed testing routines),
R Core Team [ctb, cph] (for source file ‘format.perc’)
Maintainer Matthias Kohl <Matthias.Kohl@stamats.de>
Repository CRAN
Date/Publication 2017-04-23 13:24:21 UTC
### R topics documented:

- RobAStBase-package ........................................ 3
- ALEstimate-class ........................................... 4
- BdStWeight-class ........................................... 6
- biastype-methods ........................................... 7
- BoundedWeight-class ....................................... 7
- checkIC .................................................... 9
- ComparePlot ................................................ 10
- comparePlot-methods ...................................... 11
- ContIC ..................................................... 17
- ContIC-class ............................................... 18
- ContNeighborhood .......................................... 20
- ContNeighborhood-class ................................... 21
- cutoff ....................................................... 22
- cutoff-class ............................................... 23
- ddPlot-methods ............................................. 24
- evalIC ...................................................... 28
- FixRobModel ............................................... 29
- FixRobModel-class ......................................... 30
- generateIC .................................................. 31
- generateIC.fct-methods ................................... 32
- getBiasIC .................................................... 33
- getBoundedIC ............................................... 34
- getFiRisk ..................................................... 35
- getRiskFctBV-methods ..................................... 36
- getRiskIC .................................................... 37
- getweight-methods ......................................... 39
- HampelWeight-class ........................................ 41
- HampelIC-class .............................................. 42
- IC .......................................................... 44
- IC-class ...................................................... 45
- InfluenceCurve ............................................. 46
- InfluenceCurve-class ...................................... 47
- InfoPlot ..................................................... 49
- infoPlot ..................................................... 50
- InfRobModel ................................................ 55
- InfRobModel-class ......................................... 56
- interpolRisk-class ......................................... 57
- kStepEstimate-class ........................................ 58
- kStepEstimator ............................................. 60
- kStepEstimator.start-methods ................................. 63
- locMEstimator ............................................... 64
- makeIC-methods ............................................. 65
- masked-methods ............................................. 67
- MEstimate-class ............................................. 68
- movToRef-methods .......................................... 69
- Neighborhood-class ....................................... 71
**Description**

Base S4-classes and functions for robust asymptotic statistics.

**Details**

Package: RobAStBase

Version: 1.0.1

Date: 2017-04-23

Depends: R(>= 2.14.0), methods, rrcov, distr(>= 2.5.2), distrEx(>= 2.5), distrMod(>= 2.5.2), RandVar(>= 0.9.2)

Suggests: ROptEst, RUnit (>= 0.4.26)

Imports: startupmsg

ByteCompile: yes

License: LGPL-3

URL: http://robast.r-forge.r-project.org/

SVNRevision: 930
**Package versions**

Note: The first two numbers of package versions do not necessarily reflect package-individual development, but rather are chosen for the RobASTXXX family as a whole in order to ease updating "depends" information.

**Author(s)**

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>,
Matthias Kohl <Matthias.Kohl@stamats.de>

Maintainer: Matthias Kohl <matthias.kohl@stamats.de>

**References**


**See Also**

distr-package, distrEx-package, distrMod-package

**Examples**

```r
library(RobAStBase)

## some L2 differentiable parametric family from package distrMod, e.g.
B <- BinomFamily(size = 25, prob = 0.25)

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

---

**Description**

Class of asymptotically linear estimates.

**Objects from the Class**

Objects can be created by calls of the form new("ALEstimate", ...).
**ALEEstimate-class**

**Slots**

- **name**: Object of class "character": name of the estimator.
- **estimate**: Object of class "ANY": estimate.
- **estimate.call**: Object of class "call": call by which estimate was produced.
- **samplesize**: Object of class "numeric" — the samplesize (only complete cases are counted) at which the estimate was evaluated.
- **completecases**: Object of class "logical" — complete cases at which the estimate was evaluated.
- **asvar**: Object of class "OptionalNumericOrMatrix" which may contain the asymptotic (co)variance of the estimator.
- **asbias**: Optional object of class "numeric": asymptotic bias.
- **pIC**: Optional object of class InfluenceCurve: influence curve.
- **nuis.idx**: Object of class "OptionalNumeric": indices of estimate belonging to the nuisance part.
- **fixed**: Object of class "OptionalNumeric": the fixed and known part of the parameter.
- **Infos**: Object of class "matrix" with two columns named method and message: additional informations.
- **trafo**: Object of class "list": a list with components fct and mat (see below).
- **untransformed.estimate**: Object of class "ANY": untransformed estimate.
- **untransformed.asvar**: Object of class "OptionalNumericOrMatrix" which may contain the asymptotic (co)variance of the untransformed estimator.

**Extends**

Class "Estimate", directly.

**Methods**

- **pIC** signature(object = "ALEEstimate"): accessor function for slot pIC.
- **show** signature(object = "ALEEstimate")
- **confint** signature(object = "ALEEstimate", method = "missing"): compute asymptotic (LAN-based) confidence interval neglecting any bias.
- **confint** signature(object = "ALEEstimate", method = "symmetricBias"): compute asymptotic (LAN-based) confidence interval incorporating bias symmetrically.
- **confint** signature(object = "ALEEstimate", method = "onesidedBias"): compute asymptotic (LAN-based) confidence interval incorporating bias one-sided; i.e., positive or negative, respectively.
- **confint** signature(object = "ALEEstimate", method = "asymmetricBias"): compute asymptotic (LAN-based) confidence interval incorporating bias asymmetrically.

**Author(s)**

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

See Also

Estimate-class

Examples

## prototype

new("AWEstimate")

---

## Description

Classes for bounded, robust, standardized weights.

## Objects from the Class

Objects can be created by calls of the form new("BdStWeight", ...); to fill slot weight, you will use the generating functions getweight and minbiasweight.

## Slots

- **name**: Object of class "character"; inherited from class RobWeight.
- **weight**: Object of class "function" — the weight function; inherited from class RobWeight.
- **clip**: Object of class "numeric" — clipping bound(s); inherited from class BoundedWeight.
- **stand**: Object of class "matrix" — standardization.

## Extends

Class "RobWeight", via class "BoundedWeight". Class "BoundedWeight", directly.

## Methods

- **stand**: signature(object = "BdStWeight"): accessor function for slot stand.
- **stand<-**: signature(object = "BdStWeight", value = "matrix"): replacement function for slot stand. This replacement method should be used with great care, as the slot weight is not simultaneously updated and hence, this may lead to inconsistent objects.

## Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>
biastype-methods

References


See Also

BoundedWeight-class, RobWeight-class, IC, InfluenceCurve-class

Examples

## prototype
new("BdStWeight")

biastype-methods

Methods for Function biastype in Package ‘RobAStBase’

Description

biastype-methods

Methods

biastype signature(object = "interpolrisk"): returns the slot biastype of an object of class "interpolrisk".

Examples

myrisk <- MBRRisk(samplesize=100)
biastype(myrisk)

BoundedWeight-class

Robust Weight classes for bounded weights

Description

Classes for bounded, robust weights.

Objects from the Class

Objects can be created by calls of the form new("BoundedWeight", ...).
**Slots**

name Object of class "character"; inherited from class RobWeight.

weight Object of class "function" — the weight function; inherited from class RobWeight.

clip Object of class "numeric" — clipping bound(s).

**Extends**

Class "RobWeight", directly.

**Methods**

clip signature(x1 = "BoundedWeight"): accessor function for slot clip.

clip<- signature(object = "BoundedWeight", value = "numeric"): replacement function for slot clip. This replacement method should be used with great care, as the slot weight is not simultaneously updated and hence, this may lead to inconsistent objects.

**Author(s)**

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

**References**


**See Also**

RobWeight-class, IC, InfluenceCurve-class

**Examples**

```r
## prototype
new("BoundedWeight")
```
checkIC

Generic Function for Checking ICs

Description

Generic function for checking centering and Fisher consistency of ICs.

Usage

checkIC(IC, L2Fam, ...)

Arguments

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

Details

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

Value

The maximum deviation from the IC properties is returned.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily-class, IC-class

Examples

IC1 <- new("IC")
checkIC(IC1)
**Description**

The wrapper `ComparePlot` (capital C!) takes most of arguments to function `comparePlot` (lower case c!) by default and gives a user possibility to run the function with low number of arguments.

**Usage**

```r
ComparePlot(IC1, IC2, y, ..., IC3 = NULL, IC4 = NULL,
            alpha.trsp = 100, with.legend = TRUE, rescale = FALSE,
            withCall = TRUE)
```

**Arguments**

- `IC1`: object of class `IC`
- `IC2`: object of class `IC`
- `IC3`: object of class `IC`
- `IC4`: object of class `IC`
- `y`: optional data argument — for plotting observations into the plot
- `...`: additional parameters (in particular to be passed on to `plot`)
- `alpha.trsp`: the transparency argument (0 to 100) for plotting the data
- `with.legend`: the flag for showing the legend of the plot
- `rescale`: the flag for rescaling the axes for better view of the plot
- `withCall`: the flag for the call output

**Value**

`invisible(NULL)`

**Details**

Calls `comparePlot` with suitably chosen defaults; if `withCall` == `TRUE`, the call to `comparePlot` is returned.

**Examples**

```r
# Gamma
fam <- GammaFamily()
rfam <- InfRobModel(fam, ContNeighborhood(0.5))
IC1 <- optIC(model = fam, risk = asCov())
IC2 <- makeIC(list(function(x)sin(x), function(x)x^2), L2Fam = fam)
Y <- distribution(fam)
y <- r(Y)(100)
ComparePlot(IC1, IC2, y, withCall = TRUE)
```
Description

Plots 2-4 influence curves to the same model.

Usage

```r
comparePlot(obj1, obj2, ...)
## S4 method for signature 'IC,IC'
comparePlot(obj1, obj2, obj3 = NULL, obj4 = NULL, data = NULL, ...
...withSweave = getdistrOption("withSweave"),
forceSameModel = FALSE, main = FALSE, inner = TRUE,
sub = FALSE, col = par("col"), lwd = par("lwd"), lty,
col.inner = par("col.main"), cex.inner = 0.8, bmar =
par("mar")[1], tmar = par("mar")[3],
with.automatic.grid = TRUE, with.legend = FALSE,
legend = NULL, legend.bg = "white", legend.location =
"bottomright", legend.cex = 0.8, withMBR = FALSE, MBRB
= NA, MBR.fac = 2, col.MBR = par("col"), lty.MBR =
"dashed", lwd.MBR = 0.8, x.vec = NULL, scaleX = FALSE,
scaleX.fct, scaleX.inv, scaleY = FALSE, scaleY.fct =
pnorm, scaleY.inv = qnorm, scaleN = 9, x.ticks = NULL,
y.ticks = NULL, mfColRow = TRUE, to.draw.arg = NULL,
cex.pts = 1, cex.pts.fun = NULL, col.pts = par("col"),
pch.pts = 1, jitter.fac = 1, with.lab = FALSE, lab.pts
= NULL, lab.font = NULL, alpha.trsp = NA, whichlbs =
NULL, which.Order = NULL, return.Order = FALSE,
withSubst = TRUE)
```

Arguments

- `obj1` object of class "InfluenceCurve"
- `obj2` object of class "InfluenceCurve" to be compared with `obj1`
- `obj3` optional: object of class "InfluenceCurve" to be compared with `obj1`
- `obj4` optional: object of class "InfluenceCurve" to be compared with `obj1`
- `data` optional data argument — for plotting observations into the plot;
- `withSweave` logical: if TRUE (for working with Sweave) no extra device is opened
- `forceSameModel` logical: shall we check / enforce that the model of the ICs `obj1`, `obj2`, `obj3`, and `obj4` be the same?
- `main` logical: is a main title to be used? or just as argument `main` in `plot.default`.
- `col` color[s] of ICs in arguments `obj1` [....`obj4`].
lwd
linethickness of ICs in arguments obj1[...obj4].

lty
line types of ICs in arguments obj1[...obj4].

inner
logical: do panels have their own titles? or
code vector of / cast to length 'number of plotted dimensions'; if argument
to.draw.arg is used, this refers to a vector of length length(to.draw.arg),
the actually plotted dimensions. For further information, see also description of
argument main in plot.default.

sub
logical: is a sub-title to be used? or
just as argument sub in plot.default.

tmar
top margin – useful for non-standard main title sizes

bmar
bottom margin – useful for non-standard sub title sizes

cex.inner
magnification to be used for inner titles relative to the current setting of cex; as
in par

col.inner
character or integer code; color for the inner title

with.automatic.grid
logical: should a grid be plotted alongside with the ticks of the axes, automatically?
If TRUE a respective call to grid in argument panel.first is ignored.

with.legend
logical: shall a legend be plotted?

legend
either NULL or a list of length (number of plotted panels) of items which can be
used as argument legend in command legend.

legend.location
a valid argument x for legend — the place where to put the legend on the last
issued plot

legend.bg
background color for the legend

legend.cex
magnification factor for the legend

withMBR
logical: shall horizontal lines with min and max of MBRE be plotted for compari-
on?

MBRB
matrix (or NA); coerced by usual recycling rules to a matrix with as many rows
as plotted panels and with first column the lower bounds and the second column
the upper bounds for the respective coordinates (ideally given by the MBR-IC).

MBR.fac
positive factor; scales the bounds given by argument MBRB

col.MBR
color for the MBR lines; as usual col-argument;

lty.MBR
line type for the MBR lines; as usual lty-argument;

lwd.MBR
line width for the MBR lines; as usual lwd-argument;

x.vec
a numeric vector of grid points to evaluate the influence curve; by default, x.vec
is NULL; then the grid is produced automatically according to the distribution of
the IC. x.vec can be useful for usage with a rescaling of the x-axis to avoid that
the evaluation points be selected too unevenly (i.e. on an equally spaced grid
in the original scale, but then, after rescaling non-equally). The grid has to be
specified in original scale; i.e.; when used with rescaling, it should be chosen
non-equally spaced.

scaleX
logical: shall X-axis be rescaled (by default according to the cdf of the underly-
ing distribution)?
scaleY logical; shall Y-axis be rescaled (by default according to a probit scale)?
scaleX.fct an isotone, vectorized function mapping the domain of the IC to [0,1]; if scaleX is TRUE and scaleX.fct is missing, the cdf of the underlying observation distribution.
scaleX.inv the inverse function to scaleX.fct, i.e., an isotone, vectorized function mapping [0,1] to the domain of the IC such that for any x in the domain, scaleX.inv(scaleX.fct(x))==x; if scaleX is TRUE and scaleX.inv is missing, the quantile function of the underlying observation distribution.
scaleY.fct an isotone, vectorized function mapping for each coordinate the range of the respective coordinate of the IC to [0,1]; defaulting to the cdf of $\mathcal{N}(0,1)$; can also be a list of functions with one list element for each of the panels to be plot.
scaleY.inv an isotone, vectorized function mapping for each coordinate the range [0,1] into the range of the respective coordinate of the IC; defaulting to the quantile function of $\mathcal{N}(0,1)$; can also be a list of functions with one list element for each of the panels to be plot.
scaleN integer; defaults to 9; on rescaled axes, number of x and y ticks if drawn automatically;
x.ticks numeric; defaults to NULL; (then ticks are chosen automatically); if non-NULL, user-given x-ticks (on original scale);
y.ticks numeric; defaults to NULL; (then ticks are chosen automatically); if non-NULL, user-given y-ticks (on original scale); can be a list with one (numeric or NULL) item per panel
mfColRow shall default partition in panels be used — defaults to TRUE
to.draw.arg Either NULL (default; everything is plotted) or a vector of either integers (the indices of the subplots to be drawn) or characters — the names of the subplots to be drawn: these names are to be chosen either among the row names of the trafo matrix rownames(trafo(eval(obj@CallL2Fam1@param))) or if the last expression is NULL a vector "dim<dimnr>", dimnr running through the number of rows of the trafo matrix.
withSubst logical; if TRUE (default) pattern substitution for titles and labels is used; otherwise no substitution is used.
col.pts color of the points of the data argument plotted; (may be a vector of length nIC, nIC the number of plotted pICs, i.e., one value for each pIC in arguments obj1, obj2, and, if available, obj3 and obj4, or it can be a matrix n by nIC, n the number of observations prior to any selection, in which case it assigns observation-specific colors to the observations; in this case this overrides settings in the respective col.nonlbl argument.
pch.pts symbol of the points of the data argument plotted (may be a vector of length nIC or a matrix, see col.pts).
cex.pts size of the points of the data argument plotted (may be a vector of length nIC or a matrix, see col.pts).
cex.pts.fun rescaling function for the size of the points to be plotted; either NULL (default), then log(1+abs(x)) is used for each of the rescalings, or a function which is then used for each of the rescalings, or a list of functions; if it is a function or
a list of functions, if necessary it is recycled to length \( nIC \times \text{dim} \) where \( \text{dim} \) is the number of dimensions of the pICs to be plotted; in the index of this list, \( nIC \) is incremented first; then \( \text{dim} \).

with.lab logical; shall labels be plotted to the observations? (May be a vector of length \( nIC \), see \texttt{col.pch} – but not a matrix).

lab.pps character or NULL; labels to be plotted to the observations; can be a vector of length \( n \), \( n \) the number of all observations prior to any selection with \texttt{which.lbs}, \texttt{which.Order}; if \texttt{lab.pps} is NULL, observation indices are used.

lab.font font to be used for labels (may be a vector of length \( nIC \), see \texttt{with.lab}).

alpha.trsp alpha transparency to be added ex post to colors \texttt{col.pch} and \texttt{col.lbl}; if one-dim and NA all colors are left unchanged. Otherwise, with usual recycling rules \texttt{alpha.trsp} gets shorted/prolongated to length the data-symbols to be plotted. Coordinates of this vector \texttt{alpha.trsp} with NA are left unchanged, while for the remaining ones, the alpha channel in rgb space is set to the respective coordinate value of \texttt{alpha.trsp}. The non-NA entries must be integers in \([0, 255]\) (0 invisible, 255 opaque).

jitter.fac jittering factor used in case of a \texttt{DiscreteDistribution} for plotting points of the data argument in a jittered fashion (may be a vector of length 2, see \texttt{with.lab}).

which.lbs either an integer vector with the indices of the observations to be plotted into graph or NULL — then no observation is excluded.

which.Order for each of the given ICs, we order the observations (descending) according to the norm given by the corresponding \texttt{normtype(object)}; then \texttt{which.Order} either is an integer vector with the indices of the \texttt{ordered} observations (remaining after a possible reduction by argument \texttt{which.lbs}) to be plotted into graph or NULL — then no (further) observation is excluded.

return.Order logical; if \texttt{TRUE}, a list of length maximally four with order vectors is returned — one for the ordering w.r.t. each of the given ICs; more specifically, the order of the (remaining) observations given by their original index is returned (remaining means: after a possible reduction by argument \texttt{which.lbs}, and ordering is according to the norm given by \texttt{normtype(object)}); otherwise we return \texttt{invisible()} as usual.

... further arguments to be passed to \texttt{plot}

**Details**

Any parameters of \texttt{plot.default} may be passed on to this particular \texttt{plot} method.

For main-, inner, and subtitles given as arguments \texttt{main}, \texttt{inner}, and \texttt{sub}, top and bottom margins are enlarged to 5 resp. 6 by default but may also be specified by \texttt{tmar} / \texttt{bmar} arguments. If \texttt{main} / \texttt{inner} / \texttt{sub} are logical then if the respective argument is \texttt{FALSE} nothing is done/plotted, but if it is \texttt{TRUE}, we use a default main title taking up the calling arguments in case of \texttt{main}, default inner titles taking up the class and (named) parameter slots of arguments in case of \texttt{inner}, and a "generated on <data>"-tag in case of \texttt{sub}. Of course, if \texttt{main} / \texttt{inner} / \texttt{sub} are character, this is used for the title; in case of \texttt{inner} it is then checked whether it has correct length. If argument \texttt{withSubst} is \texttt{TRUE}, in all title and axis lable arguments, the following patterns are substituted:
comparePlot-methods

"%C1","%C2","%C3","%C4"] class of argument obj<1>, i=1..4
"%A1","%A2","%A3","%A4"] deparsed argument obj<1>, i=1..4
"%O" time/date-string when the plot was generated

If argument ... contains argument ylim, this may either be as in plot.default (i.e. a vector of length 2) or a vector of length 2*(number of plotted dimensions); in the case of longer length, these are the values for ylim for the plotted dimensions of the IC, one pair for each dimension.

In addition, argument ... may contain arguments panel.first, panel.last, i.e., hook expressions to be evaluated at the very beginning and at the very end of each panel (within the then valid coordinates). To be able to use these hooks for each panel individually, they may also be lists of expressions (of the same length as the number of panels and run through in the same order as the panels).

Value

The function returns invisibly a list of elements containing the information needed to compute the respective diagnostic plot. The return value allows to recover all information used to produce the plot for later use in enhanced graphics (e.g. with ggplot).

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


See Also

L2ParamFamily-class, IC-class, plot

Examples

if(require(ROptEst)){

N0 <- NormLocationScaleFamily(mean=0, sd=1)
N0.Rob1 <- InfRobModel(center = N0, neighbor = ContNeighborhood(radius = 0.5))

IC1 <- optIC(model = N0, risk = asCov())
IC2 <- optIC(model = N0.Rob1, risk = asMSE())

comparePlot(IC1,IC2)

set.seed(12); data <- r(N0)(20)
comparePlot(IC1, IC2, data=data, with.lab = TRUE,
which.lbs = c(1:4,15:20),
which.Order = 1:6,
return.Order = TRUE)
## don't run to reduce check time on CRAN
## Not run:
## selection of subpanels for plotting
par(mfrow=c(1,1))
comparePlot(IC1, IC2, mfColRow = FALSE, to.draw.arg=c("mean"),
  panel.first= grid(), ylim=c(-4,4), xlim=c(-6,6))
## matrix-valued ylim
comparePlot(IC1, IC2, panel.first= grid(), ylim=c(-4,4,0,4), xlim=c(-6,6))

x <- c(data,-12,10)
comparePlot(IC1, IC2, data=x, which.Order=10,
  panel.first= grid(), ylim=c(-4,4,0,4), xlim=c(-6,6))

Y <- Chisq(df=1)* DiscreteDistribution(c(-1,1))
comparePlot(IC1, IC2, data=x, which.Order=10,
  scaleX = TRUE, scaleX.fct=pnorm, scaleX.inv=qnorm,
  scaleY = TRUE, scaleY.fct=pY, scaleY.inv=q(Y),
  panel.first= grid(), ylim=c(-4,4,0,4), xlim=c(-6,6))

## with use of trafo-matrix:
G <- GammaFamily(scale = 1, shape = 2)
## explicitely transforming to
## MASS parametrization:
mtrafo <- function(x){
  nms0 <- names(c(main(param(G)), nuisance(param(G))))
  nms <- c("shape", "rate")
  fval0 <- c(x[2], 1/x[1])
  names(fval0) <- nms
  mat0 <- matrix(c(0, -1/x[1]^2, 1, 0), nrow = 2, ncol = 2,
    dimnames = list(nms, nms0))
  list(fval = fval0, mat = mat0)
}
G2 <- G
trafo(G2) <- mtrafo
G2
G2.Rob1 <- InfRobModel(center = G2, neighbor = ContNeighborhood(radius = 0.5))
system.time(IC1 <- optIC(model = G2, risk = asCov()))
system.time(IC2 <- optIC(model = G2.Rob1, risk = asMSE()))
system.time(IC2.i <- optIC(model = G2.Rob1, risk = asMSE(normtype=InfoNorm())))
system.time(IC2.s <- optIC(model = G2.Rob1, risk = asMSE(normtype=SelfNorm())))

comparePlot(IC1, IC2, IC2.i, IC2.s)

## End(Not run)
Description

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

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

with clipping bound $b$, centering constant $a$ and standardizing matrix $A$. $\Lambda$ stands for the L2 derivative of the corresponding L2 differentiable parametric family which can be created via CallL2Fam.

Usage

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

Arguments

- `name` object of class "character".
- `CallL2Fam` object of class "call": creates an object of the underlying L2-differentiable parametric family.
- `Curve` object of class "EuclRandVarList"
- `Risks` object of class "list": list of risks; cf. `RiskType-class`.
- `Infos` matrix of characters with two columns named `method` and `message`: additional informations.
- `clip` positive real: clipping bound.
- `cent` real: centering constant
- `stand` matrix: standardizing matrix
- `w` HampelWeight: weight object
- `lowerCase` optional constant for lower case solution.
- `neighborRadius` radius of the corresponding (unconditional) contamination neighborhood.
- `biastype` BiasType: type of the bias
- `normtype` NormType: type of the norm
- `modifyIC` object of class "OptionalFunction": function of two arguments, which are an L2 parametric family and an optional influence curve. Returns an object of class "IC". This function is mainly used for internal computations!
ContIC-class

Value
Object of class "ContIC"

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

References

See Also
IC-class, ContIC, HampIC-class

Examples
IC1 <- ContIC()
plot(IC1)

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

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

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

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

Slots
CallL2Fam: object of class "call": creates an object of the underlying L2-differentiable parametric family.
name: object of class "character"
Curve: object of class "EuclRandVarList"
modifyIC: Object of class "OptionalFunction": function of two arguments, which are an L2 parametric family and an optional influence curve. Returns an object of class "IC". This slot is mainly used for internal computations!

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

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

clip: object of class "numeric": clipping bound.

cent: object of class "numeric": centering constant.

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

weight: object of class "HampelWeight": weight function

biastype: object of class "BiasType": bias type (symmetric/onsided/asymmetric)

normtype: object of class "NormType": norm type (Euclidean, information/self-standardized)

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

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

Extends

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

Methods

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

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

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

clip signature(x1 = "ContIC"): accessor function for slot clip.

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

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

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

neighbor signature(object = "ContIC"): generates an object of class "ContNeighborhood" with radius given in slot neighborRadius.

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

show signature(object = "ContIC")

Author(s)

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

See Also
IC-class, ContIC HampIC-class

Examples
IC1 <- new("ContIC")
plot(IC1)

ContNeighborhood  Generating function for ContNeighborhood-class

Description
Generates an object of class "ContNeighborhood".

Usage
ContNeighborhood(radius = 0)

Arguments
radius  non-negative real: neighborhood radius.

Value
Object of class "ContNeighborhood"

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

References

See Also
ContNeighborhood-class
ContNeighborhood-class

Examples

ContNeighborhood()

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

Description

Class of (unconditional) contamination neighborhoods.

Objects from the Class

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

Slots

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

Extends

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

Methods

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ContNeighborhood, UncondNeighborhood-class
Examples

new("ContNeighborhood")

cutoff

Generating function(s) for class 'cutoff'

Description

Generating function(s) for class cutoff.

Usage

cutoff(name = "empirical", body.fct0,
       cutoff.quantile = 0.95,
       norm = NormType(), QF, nsim = 100000)
cutoff.sememp(cutoff.quantile = 0.95)
cutoff.chisq(cutoff.quantile = 0.95)
cutoff.quant(qfct)

Arguments

name argument for name slot of cutoff object
body.fct0 a call generated by code wrapped to substitute resp. quote; the body of the
c查封 slot of the cutoff object
cutoff.quantile numeric (in [0,1]); the corresponding slot value for the cutoff object
norm an object of class NormType – the norm/distance by which to produce the cutoff
       value.
nsim integer: the sample size used for determining the quantiles of \((x^TQx)^{1/2}\) for \(x\) a corresponding quadratic form
QF a quadratic (positive semidefinite, symmetric) matrix used as quadratic form
qfct a (nominal) quantile function

Details

cutoff generates a valid object of class "cutoff". As function slot fct may only have a formal argument data, the other arguments to determine the cutoff value, i.e. norm, QF, nsim, cutoff.quantile, nsim have to enter the scope of this function by lexical scoping; now cutoff.quantile, norm, QF are to be taken from the calling environment (not from the defining one), so we have delay evaluation of the function body, which is why we assume it to be given wrapped into substitute resp.
quote. body.fct0 is by default (i.e. if argument body.fct0 is missing) set to
quote(quantile(slot(norm,"fct")(data), cutoff.quantile)), internally, i.e.; to an empirical quantile of the corresponding norms.
cutoff.sememp() is a helper function generating the theoretical (asymptotic) quantile of (the square root of) a corresponding quadratic form, assuming multivariate normality; to determine this quantile nsim simulations are used.

cutoff.chisq() is a helper function generating the theoretical (asymptotic) quantile of (the square root of) a (self-standardized) quadratic form, assuming multivariate normality; i.e.; a corresponding quantile of a Chi-Square distribution.

cutoff.quant() is a helper function generating the theoretical quantile corresponding to the quantile function qfct; if qfct is missing, it searches the caller environment for an object ..Ic1oc, and if this exists it uses the respective model quantile function; the fallback is qnorm. At any rate, if there is an object ..trf in the scope of the function it is used to transfer the quantile (after its evaluation).

Value

Object of class "cutoff".

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also
cutoff-class, ddPlot

Examples
cutoff()
cutoff.sememp()
cutoff.chisq()

cutoff-class  Cutoff class for distance-distance plots

Description

Class of methods to determine cutoff point for distance-distance plots; used to derive other cutoff methods later by method dispatch.

Objects from the Class

Objects could in principle be created by calls of the form new("cutoff", ...). More frequently they are created via the generating function cutoff, respectively via the helper functions cutoff.sememp and cutoff.chisq.
Slots

name: object of class "character"; defaults to "empirical" in prototype;
fct: an object of of class "function"; for this class layer, this function must only have one argument data (which may but need not be used to determine the cutoff point empirically); in derived classes this restriction could be dropped, if corresponding special methods for ddPlot are derived. Defaults to function(data) = quantile(data).
cutoff.quantile: Object of class "numeric": a probability (in [0,1]) to determine the respective quantile (empirical or theoretical) to plot the cutoff line; defaults to 0.95 in prototype;

Methods

cutoff.quantile signature(object = "cutoff"): accessor function for slot cutoff.quantile.
cutoff.quantile< signature(object = "cutoff"): replacement function for slot cutoff.quantile.
fct signature(object = "cutoff"): accessor function for slot fct.
name signature(object = "cutoff"): accessor function for slot name.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

See Also

ddPlot, outlyingPlotIC cutoff

Examples

cutoff()

---

ddPlot-methods  Methods for Function ddPlot in Package 'RobAStBase'

Description

ddPlot-methods

Usage

ddPlot(data, dist.x, dist.y, cutoff.x, cutoff.y, ...)
## S4 method for signature 'matrix'
ddPlot(data, dist.x = NormType(), dist.y = NormType(),
       cutoff.x, cutoff.y, ...
       cutoff.quantile.x = 0.95, cutoff.quantile.y = cutoff.quantile.x,
       transform.x, transform.y = transform.x,
       id.n, cex.pts = 1, lab.pts, jit.pts = 0, alpha.trsp = NA, adj =0, cex.idn,
       col.idn, lty.cutoff, lwd.cutoff, col.cutoff, text.abline = TRUE,
       text.abline.x = NULL, text.abline.y = NULL,
Arguments

data 
data coercable to matrix; the data at which to produce the ddPlot.

... 

further arguments to be passed to plot.default, text, and abline
dist.x 
object of class NormType; the distance for the x axis.
dist.y 
object of class NormType; the distance for the y axis.
cutoff.x 
object of class cutoff; the cutoff information for the x axis (the vertical line discriminating 'good' and 'bad' points).
cutoff.y 
object of class cutoff; the cutoff information for the y axis (the horizontal line discriminating 'good' and 'bad' points).
cutoff.quantile.x

numeric; the cutoff quantile for the x axis.

cutoff.quantile.y

numeric; the cutoff quantile for the y axis.

transform.x

function; a transformation to be performed before determining the distances of the x axis.

transform.y

function; a transformation to be performed before determining the distances of the y axis.

id.n

a set of indices (or a corresponding logical vector); to select a subset of the data in argument data.

cex pts

the corresponding cex argument for plotted points.

lab pts

a vector of labels for the (unsubsetted) data.

jit pts

the corresponding jitter argument for plotted points; may be a vector of length 2 – for separate factors for x- and y-coordinate.

alpha.trsp

alpha transparency to be added ex post to colors col.pch and col.lbl; if one-dim and NA all colors are left unchanged. Otherwise, with usual recycling rules alpha.trsp gets shorted/prolongated to length the data-symbols to be plotted. Coordinates of this vector alpha.trsp with NA are left unchanged, while for the remaining ones, the alpha channel in rgb space is set to the respective coordinate value of alpha.trsp. The non-NA entries must be integers in \([0,255]\) (0 invisible, 255 opaque).

adj

the corresponding argument for text for labelling the outliers.

cex.idn

the corresponding cex argument for text for labelling the outliers.

col.idn

the corresponding col argument for text for labelling the outliers.

lty.cutoff

the corresponding lty argument for abline for drawing the cutoff lines; either one lty-value (one value or vector) or a list of length 2 of lty-values.

lwd.cutoff

(vector cast to length 2): the corresponding lwd argument for abline for drawing the cutoff lines.

col.cutoff

(vector cast to length 2): the corresponding col argument for abline for drawing the cutoff lines.

text.abline

vector of logicals (cast to length 2): shall text be added to cutoff lines.

text.abline.x

text to be added to cutoff lines in x direction; if NULL (default) we use “\([pp] \%\text{-cutoff} = [ff]\)” where [pp] is the percentag up to 2 digits and [ff] is the cutoff value up to 2 digits.

text.abline.y

text to be added to cutoff lines in y direction; if NULL (default) we use “\([pp] \%\text{-cutoff} = [ff]\)” where [pp] is the percentag up to 2 digits and [ff] is the cutoff value up to 2 digits.

cex.abline

vector of numerics (cast to length 2): cex-value for added cutoff text.

col.abline

vector of length 2: color for added cutoff text.

font.abline

vector of length 2: font for added cutoff text.

adj.abline

cast to 2 x 2 matrix (by recycling rules): adjustment values for added cutoff text.
text.abline.x.y
  y-coordinate of text to be added to cutoff lines in x direction; if NULL (default)
  set to mid of mean(par("usr")c(3,4))

text.abline.y.x
  x-coordinate of text to be added to cutoff lines in y direction; if NULL (default)
  set to mid of mean(par("usr")c(1,2))

text.abline.x.x
  x-coordinate of text to be added to cutoff lines in x direction; if NULL (default)
  set to 1.05 times the cutoff value.

text.abline.y.y
  y-coordinate of text to be added to cutoff lines in y direction; if NULL (default)
  set to 1.05 times the cutoff value.

text.abline.x.fmt.cx
  format string (see gettext) to format the cutoff value in label in x direction.

text.abline.x.fmt.cx
  format string to format cutoff probability in label in x direction.

text.abline.y.fmt.cy
  format string to format the cutoff value in label in y direction.

Value
  a list (returned as invisible()) with items

id.x
  the indices of (possibly transformed) data (within subset id.n) beyond the x-
  cutoff

id.y
  the indices of (possibly transformed) data (within subset id.n) beyond the y-
  cutoff
id.xy the indices of (possibly transformed) data (within subset id.n) beyond the x-cutoff and the y-cutoff
qtx the quantiles of the distances of the (possibly transformed) data in x direction
qty the quantiles of the distances of the (possibly transformed) data in y direction
cutoff.x.v the cutoff value in x direction
cutoff.y.v the cutoff value in y direction

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

Examples

MX <- matrix(rnorm(1500), nrow=6)
QM <- matrix(rnorm(36), nrow=6); QM <- QM %*% t(QM)
ddPlot(data=MX, dist.y=QFNorm(QuadF=PosSemDefSymmMatrix(QM)))

Description

Generic function for evaluating ICs.

Usage

evalIC(IC, x)

Arguments

IC object of class "IC"

x numeric vector or matrix

Details

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

Value

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

Author(s)

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

References


See Also

IC-class

FixRobModel  Generating function for FixRobModel-class

Description

Generates an object of class "FixRobModel".

Usage

FixRobModel(center = ParamFamily(modifyParam =
    function(theta) Norm(mean = theta)), neighbor = ContNeighborhood())

Arguments

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

Value

Object of class "FixRobModel"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

FixRobModel-class
Examples

```r
(M1 <- FixRobModel())
```

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

---

**FixRobModel-class**  
Robust model with fixed (unconditional) neighborhood

**Description**

Class of robust models with fixed (unconditional) neighborhoods.

**Objects from the Class**

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

**Slots**

- `center` Object of class "ProbFamily".
- `neighbor` Object of class "UncondNeighborhood".

**Extends**

Class "RobModel", directly.

**Methods**

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

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

*ProbFamily-class, UncondNeighborhood-class, FixRobModel*
generateIC

Examples

new("FixRobModel")

generateIC  Generic function for the generation of influence curves

Description

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

Usage

generateIC(neighbor, L2Fam, ...)

Arguments

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

Value

Object of class "IC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

IC-class, ContIC-class, TotalVarIC-class
**generateIC.fct-methods**

*Generic Function for making ICs consistent at a possibly different model*

---

**Description**

Generic function for providing centering and Fisher consistency of ICs.

**Usage**

```r
generateIC.fct(neighbor, L2Fam, ...)
```

**Arguments**

- `neighbor` object of class "UncondNeighborhood"
- `L2Fam` L2-differentiable family of probability measures; may be missing.
- `...` additional parameters

**Value**

An IC at the model.

**Methods**

```r
generateIC.fct signature(IC = "UncondNeighborhood", L2Fam = "L2ParamFamily": ...)
```

**Author(s)**

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

**References**


**See Also**

- `L2ParamFamily-class`, `IC-class`
getBiasIC

Generic function for the computation of the asymptotic bias for an IC

Description

Generic function for the computation of the asymptotic bias for an IC.

Usage

getBiasIC(IC, neighbor, ...)

## S4 method for signature 'IC,UncondNeighborhood'
getBiasIC(IC, neighbor, L2Fam, biastype = symmetricBias(),
          normtype = NormType(), tol = .Machine$double.eps^0.25, numbeval = 1e5)

Arguments

- IC: object of class "InfluenceCurve"
- neighbor: object of class "Neighborhood".
- ...: additional parameters
- L2Fam: object of class "L2ParamFamily".
- biastype: object of class "BiasType"
- normtype: object of class "NormType"
- tol: the desired accuracy (convergence tolerance).
- numbeval: number of evaluation points.

Value

The bias of the IC is computed.

Methods

IC = "IC", neighbor = "UncondNeighborhood" determines the as. bias by random evaluation of the IC; this random evaluation is done by the internal S4-method .evalBiasIC; this latter dispatches according to the signature IC, neighbor, biastype. For signature IC="IC", neighbor = "ContNeighborhood", biastype = "BiasType", also an argument normtype is used to be able to use self- or information standardizing norms; besides this the signatures IC="IC", neighbor = "TotalVarNeighborhood", biastype = "BiasType", IC="IC", neighbor = "ContNeighborhood", biastype = "onesidedBias", and IC="IC", neighbor = "ContNeighborhood", biastype = "asymmetricBias" are implemented.

Note

This generic function is still under construction.
getBoundedIC

Author(s)
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

See Also
getRiskIC-methods, InfRobModel-class

Description
Generates a bounded influence curve.

Usage
getBoundedIC(L2Fam, D=trafo(L2Fam@param))

Arguments
L2Fam object of class "L2ParamFamily"
D matrix with as many columns as length(L2Fam@param)

Value
(a bounded) pIC (to matrix D) given as object of class "EuclRandVariable"

Author(s)
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References
Description

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

Usage

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

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

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

Arguments

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

Details

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

Value

The finite-sample risk is computed.

Methods

- **risk = "fiUnOvShoot", Distr = "Norm", neighbor = "ContNeighborhood"** computes finite-sample under-/overshoot risk in methods for function `getFixRobIC`.
- **risk = "fiUnOvShoot", Distr = "Norm", neighbor = "TotalVarNeighborhood"** computes finite-sample under-/overshoot risk in methods for function `getFixRobIC`.
**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

fiRisk-class

---

**Description**

getRiskFctBV for a given object of S4 class asGRisk returns a function in bias and variance to compute the asymptotic risk.

**Methods**

- `getRiskFctBV signature(risk = "asGRisk", biastype = "ANY")`: returns an error that the respective method is not yet implemented.
- `getRiskFctBV signature(risk = "asMSE", biastype = "ANY")`: returns a function with arguments bias and variance to compute the asymptotic MSE for a given ALE at a situation where it has bias bias (including the radius!) and variance variance.
- `getRiskFctBV signature(risk = "asSemivar", biastype = "onesidedBias")`: returns a function with arguments bias and variance to compute the asymptotic semivariance error, i.e. \(E[(S_n - \theta)^2_+]\) resp. \(E[(S_n - \theta)^2_-]\), for a given ALE \(S_n\) at a situation where it has one-sided bias bias (including the radius!) and variance variance.
- `getRiskFctBV signature(risk = "asSemivar", biastype = "asymmetricBias")`: returns a function with arguments bias and variance to compute the asymptotic semivariance error, i.e. \(E[\nu_1(S_n - \theta)^2_+ + \nu_2(S_n - \theta)^2_-]\) for a given ALE \(S_n\) at a situation where it has one-sided bias bias (including the radius!) and variance variance.

**Examples**

```r
myrisk <- asMSE()
getRiskFctBV(myrisk)
```
getRiskIC

Generic function for the computation of a risk for an IC

Description

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

Usage

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

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

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

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

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

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

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

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

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

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

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

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

Arguments

IC object of class "InfluenceCurve"
risk object of class "RiskType".
neighbor object of class "Neighborhood".
L2Fam object of class "L2ParamFamily".
... additional parameters
tol the desired accuracy (convergence tolerance).
sampleSize integer: sample size.
Algo "A" or "B".
cont "left" or "right".

Details

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

Value

The risk of an IC is computed.

Methods

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

IC = "IC", risk = "fiUnOvShoot", neighbor = "TotalVarNeighborhood", L2Fam = "missing" finite-sample under-/overshoot risk of IC.

Note
This generic function is still under construction.

Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

See Also
getRiskIC, InfRobModel-class

gewicht-methods Generating weights

Description
Generates weight functions of Hampel / BdSt type for different bias and norm types.

Usage
getweight(Weight, neighbor, biastype, ...)
minbiasweight(Weight, neighbor, biastype, ...)
## S4 method for signature 'HampelWeight,ContNeighborhood,BiasType'
getweight(Weight, neighbor, biastype, normW)
## S4 method for signature 'HampelWeight,ContNeighborhood,BiasType'
minbiasweight(Weight, neighbor, biastype, normW)
## S4 method for signature 'HampelWeight,ContNeighborhood,onesidedBias'
getweight(Weight, neighbor, biastype, ...)
## S4 method for signature 'HampelWeight,ContNeighborhood,onesidedBias'
minbiasweight(Weight, neighbor, biastype, ...)
## S4 method for signature 'HampelWeight,ContNeighborhood,asymmetricBias'
getweight(Weight, neighbor, biastype, ...)
## S4 method for signature 'HampelWeight,ContNeighborhood,asymmetricBias'
minbiasweight(Weight, neighbor, biastype, ...)
## S4 method for signature 'BdStWeight,TotalVarNeighborhood,BiasType'
getweight(Weight, neighbor, biastype, ...)
## S4 method for signature 'BdStWeight,TotalVarNeighborhood,BiasType'
minbiasweight(Weight, neighbor, biastype, ...)

### Arguments

**Weight**  
Object of class "RobWeight".

**neighbor**  
Object of class "Neighborhood".

**biastype**  
Object of class "BiasType".

**normW**  
Object of class "NormType" — only for signature HampelWeight,ContNeighborhood,BiasType.

...  
possibly additional (unused) arguments — like in a call to the less specific methods.

### Details

These functions generate the weight function in slot weight in a corresp. object of class RobWeight and descendants.

### Value

Object of class "HampelWeight" resp. "BdStWeight"

### Methods

**getweight** signature(Weight = "HampelWeight", neighbor = "ContNeighborhood", biastype = "onesidedBias", ...)
with additional argument biastype of class "BiasType": produces weight slot...

**minbiasweight** signature(Weight = "HampelWeight", neighbor = "ContNeighborhood", biastype = "onesidedBias", ...)
with additional argument biastype of class "BiasType": produces weight slot...

**getweight** signature(Weight = "HampelWeight", neighbor = "ContNeighborhood", biastype = "asymmetricBias", ...)
produces weight slot...

**minbiasweight** signature(Weight = "HampelWeight", neighbor = "ContNeighborhood", biastype = "asymmetricBias", ...)
produces weight slot...

**getweight** signature(Weight = "BdStWeight", neighbor = "TotalVarNeighborhood", biastype = "onesidedBias", ...)
produces weight slot...

**minbiasweight** signature(Weight = "BdStWeight", neighbor = "TotalVarNeighborhood", biastype = "onesidedBias", ...)
produces weight slot...
Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


See Also

BdStWeight-class, HampelWeight-class, IC-class

HampelWeight-class  Robust Weight classes for weights of Hampel type

Description

Classes for weights of Hampel type.

Objects from the Class

Objects can be created by calls of the form new("HampelWeight", ...); to fill slot weight, you will use the generating functions getweight and minbiasweight.

Slots

name  Object of class "character"; inherited from class RobWeight.
weight  Object of class "function" — the weight function; inherited from class RobWeight.
clip  Object of class "numeric" — clipping bound(s); inherited from class BoundedWeight.
stand  Object of class "matrix" — standardization; inherited from class BdStWeight.
cent  Object of class "numeric" — centering.

Extends

Class "RobWeight", via class "BoundedWeight". Class "BoundedWeight", via class "BdStWeight". Class "BdStWeight", directly.

Methods

cent signature(object = "HampelWeight"): accessor function for slot cent.
cent<- signature(object = "HampelWeight", value = "matrix"): replacement function for slot cent. This replacement method should be used with great care, as the slot weight is not simultaneously updated and hence, this may lead to inconsistent objects.
Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


See Also

`BdStWeight-class`, `BoundedWeight-class`, `RobWeight-class`, `IC`, `InfluenceCurve-class`

Examples

```r
## prototype
new("HampelWeight")
```

HampIC-class  

Influence curve of Hampel type

Description

Class of (partial) influence curves of Hampel (= total variation or contamination) type; used as common mother class for classes `ContIC` and `TotalVarIC`.

Objects from the Class

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

Slots

- `callL2Fam` object of class "call": creates an object of the underlying L2-differentiable parametric family.
- `name` object of class "character"
- `Curve` object of class "EuclRandVarList"
- `modifyIC` Object of class "OptionalFunction": function of two arguments, which are an L2 parametric family and an optional influence curve. Returns an object of class "IC". This slot is mainly used for internal computations!
- `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.
- `stand` object of class "matrix": standardizing matrix.
**HampIC-class**

- **weight** object of class "RobWeight": weight function
- **biastype** object of class "BiasType": bias type (symmetric/onsided/asymmetric)
- **normtype** object of class "NormType": norm type (Euclidean, information/self-standardized)
- **lowerCase** object of class "OptionalNumeric": optional constant for lower case solution.
- **neighborRadius** object of class "numeric": radius of the corresponding (unconditional) contamination neighborhood.

**Extends**

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

**Methods**

- **stand** signature(object = "HampIC"): accessor function for slot stand.
- **weight** signature(object = "HampIC"): accessor function for slot weight.
- **biastype** signature(object = "HampIC"): accessor function for slot biastype.
- **normtype** signature(object = "HampIC"): accessor function for slot normtype.
- **lowerCase** signature(object = "HampIC"): accessor function for slot lowerCase.
- **neighborRadius** signature(object = "HampIC"): accessor function for slot neighborRadius.
- **neighborRadius<-** signature(object = "HampIC"): replacement function for slot neighborRadius.

**Author(s)**

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

**References**


**See Also**

- IC-class

**Examples**

```r
IC1 <- new("HampIC")
plot(IC1)
```
Generating function for IC-class

Description
Generates an object of class "IC".

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

Arguments
- name: Object of class "character": the name of the IC.
- CallL2Fam: object of class "call": creates an object of the underlying L2-differentiable
  parametric family.
- Curve: object of class "EuclRandVarList".
- Risks: object of class "list": list of risks; cf. RiskType-class.
- Infos: matrix of characters with two columns named method and message: additional
  informations.
- modifyIC: Object of class "OptionalFunction": function of two arguments, which are an
  L2 parametric family and an optional influence curve. Returns an object of class
  "IC" at the parameter value of the L2 parametric family. This function is mainly
  used for internal computations!

Value
Object of class "IC"

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

References
Hampel et al. (1986) Robust Statistics. The Approach Based on Influence Functions. New York:
Wiley.
Dissertation.

See Also
IC-class
IC-class

Examples

IC1 <- IC()
plot(IC1)

IC-class  Influence curve

Description

Class of (partial) influence curves.

Objects from the Class

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

Slots

CallL2Fam Object of class "call": creates an object of the underlying L2-differentiable parametric family.
modifyIC Object of class "OptionalFunction": function of two arguments, which are an L2 parametric family and an optional influence curve. Returns an object of class "IC". This slot is mainly used for internal computations!
name Object of class "character".
Curve Object of class "EuclRandVarList".
Risks Object of class "list": list of risks; cf. RiskType-class.
Infos Object of class "matrix" with two columns named method and message: additional informations.

Extends

Class "InfluenceCurve", directly.

Methods

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

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

References

See Also
InfluenceCurve-class, IC

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

________________________________________________________________
InfluenceCurve	Generating function for InfluenceCurve-class

Description
Generates an object of class "InfluenceCurve".

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

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

Object of class "InfluenceCurve"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`InfluenceCurve-class`

Examples

```r
class("influencecurve")
```

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

---

**Description**

Class of influence curves (functions).

**Objects from the Class**

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

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

**Methods**

- `name` signature(object = "InfluenceCurve"): accessor function for slot `name`.
- `name<-` signature(object = "InfluenceCurve"): replacement function for slot `name`.
- `Curve` signature(object = "InfluenceCurve"): accessor function for slot `Curve`.
- `Map` signature(object = "InfluenceCurve"): accessor function for slot `Map` of slot `Curve`.
- `Domain` signature(object = "InfluenceCurve"): accessor function for slot `Domain` of slot `Curve`.
- `Range` signature(object = "InfluenceCurve"): accessor function for slot `Range` of slot `Curve`.
- `Infos` signature(object = "InfluenceCurve"): accessor function for slot `Infos`.
- `Infos<-` signature(object = "InfluenceCurve"): replacement function for slot `Infos`.
- `addInfo<-` signature(object = "InfluenceCurve"): function to add an information to slot `Infos`.
- `Risks` signature(object = "InfluenceCurve"): accessor function for slot `Risks`. By means of internal function `.evalListRec` recursively evaluates all non evaluated calls and writes back the evaluated calls to the calling environment.
- `Risks<-` signature(object = "InfluenceCurve"): replacement function for slot `Risks`.
- `addRisk<-` signature(object = "InfluenceCurve"): function to add a risk to slot `Risks`.
- `show` signature(object = "InfluenceCurve")

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**


**See Also**

- `InfluenceCurve`, `RiskType-class`
Examples

new("InfluenceCurve")

Description

The wrapper InfoPlot (capital I!) takes most of arguments to the plot method infoPlot (lower case i!) by default and gives a user possibility to run the function with low number of arguments.

Usage

InfoPlot(IC, data, ..., alpha.trsp = 100,
          with.legend = TRUE, rescale = FALSE, withCall = TRUE)

Arguments

IC object of class IC
data optional data argument — for plotting observations into the plot
... additional parameters (in particular to be passed on to plot)
alpha.trsp the transparency argument (0 to 100) for ploting the data
with.legend the flag for showing the legend of the plot
rescale the flag for rescaling the axes for better view of the plot
withCall the flag for the call output

Value

invisible(NULL)

Details

Calls infoPlot with suitably chosen defaults. If withCall == TRUE, the call to infoPlot is returned

Examples

# Gamma
fam <- GammaFamily()
IC <- optIC(model = fam, risk = asCov())
Y <- distribution(fam)
data <- r(Y)(500)
InfoPlot(IC, data, withCall = FALSE)
infoPlot

Plot absolute and relative information

Description

Plot absolute and relative information of influence curves.

Usage

infoPlot(object, ...)  ## S4 method for signature 'IC'
infoPlot(object, data = NULL, ...,
withSweave = getdistrOption("withSweave"), col = par("col"), lwd = par("lwd"),
ltY, colII = grey(0.5), lwdII = 0.7 * par("lwd"), ltyII = "dotted", main = FALSE,
inner = TRUE, sub = FALSE, col.inner = par("col.main"), cex.inner = 0.8,
bfmar = par("mar")[1], tmar = par("mar")[3], with.automtic.grid = TRUE,
with.legend = TRUE, legend = NULL, legend.bg = "white",
legend.location = "bottomright", legend.cex = 0.8, x.vec = NULL,
scaleX = FALSE, scaleX.fct, scaleX.inv, scaleY = FALSE, scaleY.fct = pnorm,
scaleY.inv = qnorm, scaleN = 9, x.ticks = NULL, y.ticks = NULL,
mfColRow = TRUE, to.draw.arg = NULL, cex.pts = 1, cex.pts.fun = NULL,
col.pts = par("col"), pch.pts = 1, jitter.fac = 1, with.lab = FALSE,
lab.pts = NULL, lab.font = NULL, alpha.trsp = NA, which.lbs = NULL,
which.Order = NULL, return.Order = FALSE, ylab.abs = "absolute information",
ylab.rel = "relative information", withSubst = TRUE)

Arguments

object

data

withSweave

main

inner

sub

tmar

bmar

col

object of class "InfluenceCurve"
optional data argument — for plotting observations into the plot;
logical: if TRUE (for working with Sweave) no extra device is opened
logical: is a main title to be used? or
just as argument main in plot.default.
logical: do panels have their own titles? or
character vector of / cast to length 'number of compared dimensions'; if argument
to.draw.arg is used, this refers to a vector of length 1 (absolute information) +
length(to.draw.arg), the actually plotted relative informations. For
further information, see also main in plot.default.
logical: is a sub-title to be used? or
just as argument sub in plot.default.
top margin – useful for non-standard main title sizes; may be a vector with
individual values for each of the panels to be plotted.
bottom margin – useful for non-standard sub title sizes; may be a vector with
individual values for each of the panels to be plotted.
color of IC in argument object.
lwd  linewidth of IC in argument object.
ltty  line-type of IC in argument object.
colI  color of the classically optimal IC.
lwdI  linewidth of the classically optimal IC.
ltyI  line-type of the classically optimal IC.
cex.inner  magnification to be used for inner titles relative to the current setting of cex; as in \texttt{par}.
col.inner  character or integer code; color for the inner title
with.automatic.grid  logical; should a grid be plotted alongside with the ticks of the axes, automatically? If \texttt{TRUE} a respective call to \texttt{grid} in argument \texttt{panel.first} is ignored.
with.legend  logical; shall a legend be plotted?
legend  either \texttt{NULL} or a list of length (number of plotted panels) of items which can be used as argument \texttt{legend} in command \texttt{legend}.
legend.location  a valid argument \texttt{x} for \texttt{legend} — the place where to put the legend on the last issued plot — or a list of length (number of plotted panels) of such arguments, one for each plotted panel.
legend.bg  background color for the legend
legend.cex  magnification factor for the legend
x.vec  a numeric vector of grid points to evaluate the influence curve; by default, \texttt{x.vec} is \texttt{NULL}; then the grid is produced automatically according to the distribution of the IC. \texttt{x.vec} can be useful for usage with a rescaling of the x-axis to avoid that the evaluation points be selected too unevenly (i.e. on an equally spaced grid in the original scale, but then, after rescaling non-equally). The grid has to be specified in original scale; i.e.; when used with rescaling, it should be chosen non-equally spaced.
scaleX  logical; shall X-axis be rescaled (by default according to the cdf of the underlying distribution)?
scaleY  logical; shall Y-axis be rescaled for abs.info-plot (by default according to a probit scale)?
scaleX.fct  an isotone, vectorized function mapping the domain of the IC to \([0,1]\); if \texttt{scaleX} is \texttt{TRUE} and \texttt{scaleX.fct} is missing, the cdf of the underlying observation distribution.
scaleX.inv  the inverse function to \texttt{scale.fct}, i.e., an isotone, vectorized function mapping \([0,1]\) to the domain of the IC such that for any \texttt{x} in the domain, \texttt{scaleX.inv(scaleX.fct(x))==x}; if \texttt{scaleX} is \texttt{TRUE} and \texttt{scaleX.inv} is missing, the quantile function of the underlying observation distribution.
scaleY.fct  an isotone, vectorized function mapping the range of the norm of the IC to \([0,1]\); defaulting to the cdf of \(\mathcal{N}(0, 1)\); can also be a list of functions with one list element for each of the panels to be plot.
scaleY.inv  an isotone, vectorized function mapping \([0,1]\) into the range of the norm of the IC; defaulting to the quantile function of \(\mathcal{N}(0, 1)\); can also be a list of functions with one list element for each of the panels to be plot.
scaleN  integer; defaults to 9; on rescaled axes, number of x and y ticks if drawn automatically;

x.ticks  numeric; defaults to NULL; (then ticks are chosen automatically); if non-NULL, user-given x-ticks (on original scale);

y.ticks  numeric; defaults to NULL; (then ticks are chosen automatically); if non-NULL, user-given y-ticks (on original scale); can be a list with one (numeric or NULL) item per panel

mfColRow  shall default partition in panels be used — defaults to TRUE

to.draw.arg  Either NULL (default; everything is plotted) or a vector making a selection among the relative information plots; the absolute information being plotted in any case. This vector is either a vector of integers (the indices of the subplots to be drawn) or characters — the names of the subplots to be drawn: these names are to be chosen either among the row names of the trafo matrix rownames(traf(o(eval(object@CallL2Fam)@para)) or if the last expression is NULL a vector "dim<dimnr>", dimnr running through the number of rows of the trafo matrix.

withSubst  logical; if TRUE (default) pattern substitution for titles and labels is used; otherwise no substitution is used.

col.pts  color of the points of the data argument plotted; (may be a vector of length 2, one value for the classical IC, one for the IC in argument object, or it can be a matrix n by 2, n the number of observations prior to any selection, in which case it assigns observation-specific colors to the observations; in this case this overrides settings in the respective col.nonlbl argument.

pch.pts  symbol of the points of the data argument plotted (may be a vector of length 2 or a matrix, see col.pts).

cex.pts  size of the points of the data argument plotted (may be a vector of length 2 or a matrix, see col.pts).

cex.pts.fun  rescaling function for the size of the points to be plotted; either NULL (default), then \( \log(1+|x|) \) is used for each of the rescalings, or a function which is then used for each of the rescalings, or a list of functions; if it is a function or a list of functions, if necessary it is recycled to length 2 \(*\ dim\) (where 2 is for the two pICs plotted, i.e., the classically optimal one and argument IC, and dim is the number of dimensions of the pICs to be plotted; in the index of this list, the index for classical vs. IC is incremented first; then dim.

with.lab  logical; shall labels be plotted to the observations? (may be a vector of length 2, see col.pts – but not a matrix)

lab.pts  character or NULL; labels to be plotted to the observations; can be a vector of length n, n the number of all observations prior to any selection with which.lbs, which.order; if lab.pts is NULL, observation indices are used.

lab.font  font to be used for labels; (may be a vector of length 2, see with.lab).

alpha.trsp  alpha transparency to be added ex post to colors col.pch and col.nonlbl; if one-dim and NA all colors are left unchanged. Otherwise, with usual recycling rules alpha.trsp gets shortened/prolongated to length the number of panel data-symbols to be plotted. Coordinates of this vector alpha.trsp with NA are left unchanged, while for the remaining ones, the alpha channel in rgb space is set to the respective coordinate value of alpha.trsp. The non-NA entries must be integers in \([0,255]\) (0 invisible, 255 opaque).
jitter.fac jittering factor used in case of a `DiscreteDistribution` for plotting points of the data argument in a jittered fashion (may be a vector of length 2, see `with.lab`).

which.lbs either an integer vector with the indices of the observations to be plotted into graph or NULL — then no observation is excluded

which.Order we order the observations (descending) according to the norm given by `normtype(object)`; then which.Order either is an integer vector with the indices of the `ordered` observations (remaining after a possible reduction by argument `which.lbs`) to be plotted into graph or NULL — then no (further) observation is excluded.

return.Order logical; if TRUE, a list of length two with order vectors is returned — one for ordering w.r.t. the given IC, one for ordering w.r.t. the classically optimal IC; more specifically, the order of the (remaining) observations given by their original index is returned (remaining means: after a possible reduction by argument `which.lbs`, and ordering is according to the norm given by `normtype(object)`); otherwise we return `invisible()` as usual.

ylab.abs character; label to be used for y-axis in absolute information panel

ylab.rel character; label to be used for y-axis in relative information panel

... further parameters for `plot`

Details

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

Any parameters of `plot.default` may be passed on to this particular `plot` method.

For main-, inner, and subtitles given as arguments main, inner, and sub, top and bottom margins are enlarged to 5 resp. 6 by default but may also be specified by tmar / bmar arguments. If main / inner / sub are logical then if the respective argument is FALSE nothing is done/plotted, but if it is TRUE, we use a default main title taking up the calling arguments in case of main, default inner titles taking up the class and (named) parameter slots of arguments in case of inner, and a "generated on <data>"-tag in case of sub. Of course, if main / inner / sub are character, this is used for the title; in case of inner it is then checked whether it has correct length. If argument withSubst is TRUE, in all title and axis lable arguments, the following patterns are substituted:

"%C" class of argument object

"%A" deparsed argument object

"%D" time/date-string when the plot was generated

If argument ... contains argument ylim, this may either be as in `plot.default` (i.e. a vector of length 2) or a vector of length 2*(number of plotted dimensions + e), where e is 1 or 0 depending on whether absolute information is plotted or not; in the case of longer length, if e is 1, the first two elements are the values for ylim in panel "Abs", while the last 2*(number of plotted dimensions) are the values for ylim for the plotted dimensions of the IC, one pair for each dimension.

Similarly, if argument ... contains arguments xaxt or yaxt, these may be vectorized, with one value for each of the panels to be plotted. This is useful for stacking panels over each other, using a common x-axis (see example below).
The ... argument may also contain an argument with box which if TRUE warrants that even if xaxt and yaxt both are FALSE, a box is drawn around the respective panel.

In addition, argument ... may contain arguments panel.first, panel.last, i.e., hook expressions to be evaluated at the very beginning and at the very end of each panel (within the then valid coordinates). To be able to use these hooks for each panel individually, they may also be lists of expressions (of the same length as the number of panels and run through in the same order as the panels).

Value

The function returns invisibly a list of elements containing the information needed to compute the respective diagnostic plot. The return value allows to recover all information used to produce the plot for later use in enhanced graphics (e.g. with ggplot).

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

L2ParamFamily-class, IC-class

Examples

```r
N <- NormLocationScaleFamily(mean=0, sd=1)
IC1 <- optIC(model = N, risk = asCov())
infoPlot(IC1)

## don't run to reduce check time on CRAN
## Not run:
## selection of subpanels for plotting
par(mfrow=c(1,2))
infoPlot(IC1, mfColRow = FALSE, to.draw.arg=c("Abs","sd"))
infoPlot(IC1, mfColRow = FALSE, to.draw.arg=c("Abs","sd"), log="y")

infoPlot(IC1, mfColRow = FALSE, to.draw.arg=c("Abs","mean"),
    panel.first= grid(), ylim = c(0,4), xlim = c(-6,6))
infoPlot(IC1, mfColRow = FALSE, to.draw.arg=c("Abs","mean"),
    panel.first= grid(), ylim = c(0,4), xlim = c(-3,0), xlab = "x")

par(mfrow=c(1,3))
infoPlot(IC1, mfColRow = FALSE, panel.first= grid(),
    ylim = c(0,4,0..3,0..8), xlim=c(-6,6))
par(mfrow=c(1,1))

data <- r(N)(20)
```
InfRobModel

Generating function for InfRobModel-class

Description
Generates an object of class "InfRobModel".

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

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

Value
Object of class "FixRobModel"

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


See Also

RobModel-class, FixRobModel-class

Examples

(M1 <- InfRobModel())

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

infrobmodelMclass  Robust model with infinitesimal (unconditional) neighborhood

Description

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

Objects from the Class

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

Slots

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

Extends

Class "RobModel", directly.

Methods

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

Author(s)

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

References


See Also

ProbFamily-class, UncondNeighborhood-class, InfRobModel

Examples

new("InfRobModel")

interpolRisk-class Interpolated Risks

Description

Class of risks for which algorithms dispatch to speed-up algorithms

Usage

MBRRisk(samplesize=100)
OMSRRisk(samplesize=100)
RMXRRisk(samplesize=100)

Arguments

samplesize sample size at which to look at the risk.

Details

The main purpose of classes OMSRRisk, MBRRisk, and RMXRRisk is to help to dispatch into speed-up algorithms later in function roptest. In all these risks, we assume convex contamination neighborhoods. OMSRRisk stands for optimal MSE-robust estimation (where we assume a radius r of 0.5), RMXRRisk stands for optimal optimally RMX-robust estimation and MBRRisk stands for optimal Bias-robust estimation. All these risks have an additional slot samplesize, defaulting to 100, and for which there is a replacement and an accessor method.

Objects from the Class

interpolRisk is a virtual class: No objects may be created from it. the other classes are generated via generating functions.

Slots

type Object of class "character": type of risk. (Inherited from RiskType).
Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

Examples

\begin{verbatim}
new("OMSRisk")
OMSRisk()
RMXRRisk()
MBRRisk()
myrisk <- MBRisk(samplesize=100)
samplesize(myrisk)
samplesize(myrisk) <- 20
\end{verbatim}

kStepEstimate-class.  

Description

Class of asymptotically linear estimates.

Objects from the Class

Objects can be created by calls of the form \texttt{new("kStepEstimate", ...)}. More frequently they are created via the generating function \texttt{kStepEstimator}.

Slots

- \texttt{name}: Object of class "character": name of the estimator.
- \texttt{estimate}: Object of class "ANY": estimate.
- \texttt{estimate.call}: Object of class "call": call by which estimate was produced.
- \texttt{samplesize}: object of class "numeric" — the samplesize (only complete cases are counted) at which the estimate was evaluated.
- \texttt{completecases}: object of class "logical" — complete cases at which the estimate was evaluated.
- \texttt{asvar}: object of class "OptionalNumericOrMatrix" which may contain the asymptotic (co)variance of the estimator.
- \texttt{asbias}: Optional object of class "numeric": asymptotic bias.
- \texttt{pIC}: Optional object of class InfluenceCurve: influence curve.
- \texttt{nuis.idx}: object of class "OptionalNumeric": indices of estimate belonging to the nuisance part.
- \texttt{fixed}: object of class "OptionalNumeric": the fixed and known part of the parameter.
- \texttt{steps}: Object of class "integer": number of steps.
- \texttt{Infos}: object of class "matrix" with two columns named method and message: additional informations.
- \texttt{trafo}: object of class "list": a list with components fct and mat (see below).
untransformed.estimate: Object of class "ANY": untransformed estimate.
untransformed.asvar: object of class "OptionalNumericOrMatrix" which may contain the asymptotic (co)variance of the untransformed estimator.

pICList: Optional object of class "OptionalList": the list of (intermediate) (partial) influence curves used; only filled when called from kStepEstimator with argument withPICList==TRUE.

ICList: Optional object of class "OptionalList": the list of (intermediate) (total) influence curves used; only filled when called from kStepEstimator with argument withICList==TRUE.

start: The argument start — of class "StartClass" used in call to kStepEstimator.

startval: Object of class matrix: the starting value with which the k-step Estimator was initialized (in p-space / transformed).

ustartval: Object of class matrix: the starting value with which the k-step Estimator was initialized (in k-space / untransformed).

ksteps: Object of class "OptionalMatrix": the intermediate estimates (in p-space) for the parameter; only filled when called from kStepEstimator.

uksteps: Object of class "OptionalMatrix": the intermediate estimates (in k-space) for the parameter; only filled when called from kStepEstimator.

Extends
Class "ALEstimate", directly.
Class "Estimate", by class "ALEstimate"

Methods

steps signature(object = "kStepEstimate"): accessor function for slot steps.
ksteps signature(object = "kStepEstimate"): accessor function for slot ksteps; has additional argument diff, defaulting to FALSE; if the latter is TRUE, the starting value from slot startval is prepended as first column; otherwise we return the corresponding increments in each step.

uksteps signature(object = "kStepEstimate"): accessor function for slot uksteps; has additional argument diff, defaulting to FALSE; if the latter is TRUE, the starting value from slot ustartval is prepended as first column; otherwise we return the corresponding increments in each step.

start signature(object = "kStepEstimate"): accessor function for slot start.

startval signature(object = "kStepEstimate"): accessor function for slot startval.

ustartval signature(object = "kStepEstimate"): accessor function for slot startval.

ICList signature(object = "kStepEstimate"): accessor function for slot ICList.

pICList signature(object = "kStepEstimate"): accessor function for slot pICList.

show signature(object = "kStepEstimate"): a show method;

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

Function for the computation of k-step estimates

Description

Function for the computation of k-step estimates.

Usage

kStepEstimator(x, IC, start = NULL, steps = 1L, 
useLast = getRobAStBaseOption("kStepUseLast"),
withUpdateInKer = getRobAStBaseOption("withUpdateInKer"),
IC.UpdateInKer = getRobAStBaseOption("IC.UpdateInKer"),
withICList = getRobAStBaseOption("withICList"),
withPICList = getRobAStBaseOption("withPICList"),
na.rm = TRUE, startArgList = NULL, ..., 
withLogScale = TRUE, withEvalAsVar = TRUE)

Arguments

x sample

IC object of class "IC"

start initial estimate (for full parameter, i.e. in dimension k respective joint length of main and nuisance part of the parameter): either a numerical value, or an object of class "Estimate" or a function producing either a numerical value, or an object of class "Estimate" when evaluated at x,...; if missing or NULL, we use slot startPar of the L2family L2Fam from within IC

steps integer: number of steps

useLast which parameter estimate (initial estimate or k-step estimate) shall be used to fill the slots pIC, asvar and asbias of the return value.

withUpdateInKer if there is a non-trivial trafo in the model with matrix D, shall the parameter be updated on ker(D)?

IC.UpdateInKer if there is a non-trivial trafo in the model with matrix D, the IC to be used for this; if NULL the result of getboundedIC(L2Fam, D) is taken; this IC will then be projected onto ker(D).

na.rm logical: if TRUE, the estimator is evaluated at complete.cases(x).

startArgList a list of arguments to be given to argument start if the latter is a function; this list by default already starts with two unnamed items, the sample x, and the model eval(CallL2Fam(IC)).

withPICList logical: shall slot pICList of return value be filled?
withICList logical: shall slot ICLlist of return value be filled?
... additional parameters
withLogScale logical: if TRUE, a scale component (if existing and found with name scalename) is computed on log-scale and backtransformed afterwards (default). This avoids crossing 0.
withEvalAsVar logical: if TRUE (default), tells R to evaluate the asymptotic variance or just to produces a call to do so.

Details

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

The default value of argument useLast is set by the global option kStepUseLast which by default is set to FALSE. In case of general models useLast remains unchanged during the computations. However, if slot CallL2Fam of IC generates an object of class "L2GroupParamFamily" the value of useLast is changed to TRUE. Explicitly setting useLast to TRUE should be done with care as in this situation the influence curve is re-computed using the value of the one-step estimate which may take quite a long time depending on the model.

If useLast is set to TRUE and slot modifyIC of IC is filled with some function (which can be used to re-compute the IC for a different parameter), the computation of asvar, asbias and IC is based on the k-step estimate.

Value

Object of class "kStepEstimate".

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>,
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


See Also

IC-class, kStepEstimate-class

Examples

if(require(ROptEst)){
  # 1. generate a contaminated sample
  ind <- rbinom(100, size=1, prob=0.05)
  x <- rnorm(100, mean=0, sd=(1-ind) + ind*9)
### 2. Kolmogorov-Smirnov minimum distance estimator

(est0 <- MDEstimator(x=x, NormLocationScaleFamily()))

### 3. k-step estimation: radius known

N1 <- NormLocationScaleFamily(mean=estimate(est0)['mean'], sd=estimate(est0)['sd'])
N1.Rob <- InfRobModel(center = N1, neighbor = ContNeighborhood(radius = 0.5))
IC1 <- optIC(model = N1.Rob, risk = asMSE())
(est1 <- kStepEstimator(x, IC1, est0, steps = 3, withPIC = TRUE))
estimate(est1)
ksteps(est1)
pIList(est1)
start(est1)

## don't run to reduce check time on CRAN
## Not run:
## a transformed model

tfct <- function(x){
  nms0 <- c("mean","sd")
  nms <- "comb"
  fval0 <- x[1]+2*x[2]
  names(fval0) <- nms
  mat0 <- matrix(c(1,2), nrow = 1, dimnames = list(nms, nms0))
  return(list(fval = fval0, mat = mat0))
}
N1.traf <- N1; trafo(N1.traf) <- tfct
N1R.traf <- N1.Rob; trafo(N1R.traf) <- tfct
IC1.traf <- optIC(model = N1R.traf, risk = asMSE())
(est0.traf <- MDEstimator(x, N1.traf))
(est1.traf <- kStepEstimator(x, IC1.traf, est0, steps = 3,
  withIC = TRUE, withPIC = TRUE, withUpdateInKer = FALSE))
(est1a.traf <- kStepEstimator(x, IC1.traf, est0, steps = 3,
  withIC = TRUE, withPIC = TRUE, withUpdateInKer = TRUE))
estimate(est1.traf)
ksteps(est1.traf)
pIList(est1.traf)
startval(est1.traf)
untransformed.estimate(est1.traf)
uksteps(est1.traf)
ICList(est1.traf)
ustartval(est1.traf)
## Description

`kStepEstimator.start-methods`; these are called from within `kStepEstimator` to produce a numeric value of for the starting estimator in the end.

### Usage

```r
kStepEstimator.start(start, ...)  # S4 method for signature 'numeric'
kStepEstimator.start(start, nrvalues, ...)  # S4 method for signature 'Estimate'
kStepEstimator.start(start, nrvalues, ...)  # S4 method for signature 'function'
kStepEstimator.start(start, x, nrvalues, na.rm, L2Fam, startList)
```

### Arguments

- **start**: the start slot of an object of class `kStepEstimator`
- **nrvalues**: numeric; dimension $k$ of the original model, i.e.; length of the untransformed parameter, or joint length of main and nuisance part of the parameter.
- **x**: the data at which the starting estimator is to be evaluated.
- **na.rm**: logical: if `TRUE`, the estimator is evaluated at `complete.cases(x)`.
- **startList**: a list of arguments to be given to the call to `start` if this is a function;
- **L2Fam**: the parametric family;
- **...**: further arguments for `kStepEstimator.start`.

### Value

A numeric vector with the corresponding value of the start estimator (in $k$ space)

### Methods

- **`kStepEstimator.start`** signature(start = "numeric"): returns the unchanged argument `start` if it has the correct length; otherwise throws an error.

- **`kStepEstimator.start`** signature(start = "Estimate"): returns slot `untransformed.estimate` of `start` if it is not `NULL`, and else slot `estimate` if the latter has dimension `nrvalues`.

- **`kStepEstimator.start`** signature(start = "function"): returns `kStepEstimator.start(do.call(start, args=c(list(x), ...)))` where, if `na.rm == TRUE`, beforehand `x` has been modified to `x <- complete.cases(x)`. 
Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


See Also

`kStepEstimator`, `ALEstimate-class`

---

**Description**

Generic function for the computation of location M estimates.

**Usage**

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

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

**Arguments**

- `x` sample
- `IC` object of class "InfluenceCurve"
- `...` additional parameters
- `eps` the desired accuracy (convergence tolerance).
- `na.rm` logical: if TRUE, the estimator is evaluated at complete.cases(x).

**Details**

Given some sample `x` and some influence curve `IC` an M estimate is computed by solving the corresponding M equation.

**Value**

Object of class "MEstimate"

**Methods**

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

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

References

See Also
InfluenceCurve-class, MEstimate-class

makeIC-methods

Generic Function for making ICs consistent at a possibly different model

Description
Generic function for providing centering and Fisher consistency of ICs.

Usage
makeIC(IC, L2Fam, ...)  
## S4 method for signature 'IC,missing'
makeIC(IC)
## S4 method for signature 'IC,L2ParamFamily'
makeIC(IC, L2Fam)
## S4 method for signature 'list,L2ParamFamily'
makeIC(IC, L2Fam, forceIC = TRUE, name, Risks, Infos, modifyIC = NULL)
## S4 method for signature 'function,L2ParamFamily'
makeIC(IC, L2Fam, forceIC = TRUE, name, Risks, Infos, modifyIC = NULL)

Arguments
IC  object of class "IC" for signature IC="IC", respectively a list of functions in one argument for signature IC="list", respectively a function in one argument for signature IC="function".
L2Fam  L2-differentiable family of probability measures; may be missing.
forceIC  logical: shall centeredness and Fisher consistency be enforced applying an affine linear transformation?
name  Object of class "character"; the name of the IC
Risks: object of class "list": list of risks; cf. RiskType-class.
Infos: matrix of characters with two columns named method and message: additional informations.
modifyIC: Object of class "OptionalFunction": function of two arguments, which are an L2 parametric family and an optional influence curve. Returns an object of class "IC" at the parameter value of the L2 parametric family. This function is mainly used for internal computations!
... additional parameters

Value
An IC of class "IC" at the model.

Methods

makeIC signature(IC = "IC", L2Fam = "missing": creates an object of class "IC" at the parametric model of its own slot CallL2Fam; enforces IC conditions centeredness and Fisher consistency, applying an affine linear transformation.
makeIC signature(IC = "IC", L2Fam = "L2ParamFamily": creates an object of class "IC" at the parametric model L2Fam; enforces IC conditions centeredness and Fisher consistency, applying an affine linear transformation.
makeIC signature(IC = "list", L2Fam = "L2ParamFamily": creates an object of class "IC" out of a list of functions given by argument IC at the parametric model L2Fam; enforces IC conditions centeredness and Fisher consistency, applying an affine linear transformation.
makeIC signature(IC = "function", L2Fam = "L2ParamFamily": creates an object of class "IC" out of a function given by argument IC at the parametric model L2Fam; enforces IC conditions centeredness and Fisher consistency, applying an affine linear transformation.

Author(s)
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References

See Also
L2ParamFamily-class, IC-class

Examples

```r
## default IC
IC1 <- new("IC")

## L2-differentiable parametric family
masked-methods

Masked Methods from Packages 'stats' and 'graphics' in Package 'RobAStBase'

Description

masked methods from packages stats and graphics

Usage

clip(x1,...)
  ## S4 method for signature 'ANY'
clip(x1,x2,y1,y2)
MEstimate-class

```
start(x,...)
## S4 method for signature 'ANY'
start(x,...)
```

**Arguments**

- `x,...` see `start`.
- `x1,x2,y1,y2` see `clip`.

**Details**

In order to make accessible the otherwise masked functions `start`, `clip`, we generate corresponding S4-methods.

**Value**

see `start`, `clip`.

**Author(s)**

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

---

**Description**

Class of asymptotically linear estimates.

**Objects from the Class**

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

**Slots**

- `name` Object of class "character": name of the estimator.
- `estimate` Object of class "ANY": estimate.
- `samplesize` Object of class "numeric": sample size.
- `asvar` Optional object of class "matrix": asymptotic variance.
- `asbias` Optional object of class "numeric": asymptotic bias.
- `pic` Optional object of class InfluenceCurve: influence curve.
- `nuis.idx` object of class "OptionalNumeric": indices of estimate belonging to the nuisance part.
- `mroot` Object of class "numeric": value of the M equation at the estimate.
- `Infos` object of class "matrix" with two columns named method and message: additional informations.
Extends

Class "AListimate", directly.
Class "Estimate", by class "AListimate".

Methods

Mroot signature(object = "MEstimate"): accessor function for slot Mroot.
show signature(object = "MEstimate")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

AListimate-class

Examples

## prototype
new("MEstimate")

movToRef-methods

Methods for Functions moving from and to reference parameter in
Package ‘ROptEst’

Description

In optIC a gain in accuracy can be obtained when computing the optimally-robust ICs at a reference
parameter of the model (instead of an arbitrary one). To this end, moveL2Fam2RefParam moved the
model to the reference parameter and moveICBackFromRefParam moves the obtained optimal IC
back to the original parameter.

Usage

moveL2Fam2RefParam(L2Fam, ...)
moveICBackFromRefParam(IC, L2Fam, ...)

Arguments

L2Fam object of class L2ParamFamily
IC IC of class HampIC
... further arguments to be passed on.
Details

moveL2Fam2RefParam and moveICBackFromRefParam are used internally in functions robtest and robest to compute the optimally robust influence function according to the arguments given to them.

Value

moveL2Fam2RefParam

the L2 Family transformed to reference parameter.

moveICBackFromRefParam

the backtransformed IC.

Methods

moveL2Fam2RefParam signature(L2Fam = "L2ParamFamily"): returns L2Fam unchanged.

moveL2Fam2RefParam signature(L2Fam = "L2LocationFamily"): moves L2Fam to location 0.

moveL2Fam2RefParam signature(L2Fam = "L2ScaleFamily"): moves L2Fam to location 0 and scale 1.

moveL2Fam2RefParam signature(L2Fam = "L2LocationScaleFamily"): moves L2Fam to location 0 and scale 1.

moveL2Fam2RefParam signature(L2Fam = "L2LocationUnknownScaleFamily"): moves L2Fam to location 0.

moveL2Fam2RefParam signature(L2Fam = "L2ScaleUnknownLocationFamily"): moves L2Fam to scale 1.

moveICBackFromRefParam signature(IC = "IC", L2Fam = "L2ParamFamily"): returns IC unchanged.

moveICBackFromRefParam signature(IC = "IC", L2Fam = "L2LocationFamily"): moves IC in IC back to original location in L2Fam.

moveICBackFromRefParam signature(IC = "IC", L2Fam = "L2ScaleFamily"): moves IC in IC back to original location and scale in L2Fam, rescaling risk where necessary.

moveICBackFromRefParam signature(IC = "IC", L2Fam = "L2LocationScaleFamily"): moves IC in IC back to original location and scale in L2Fam, rescaling risk where necessary.

moveICBackFromRefParam signature(IC = "IC", L2Fam = "L2LocationUnknownScaleFamily"): moves IC in IC back to original location and scale in L2Fam, rescaling risk where necessary.

moveICBackFromRefParam signature(IC = "L2ScaleUnknownLocationFamily"): moves IC in IC back to original location and scale in L2Fam, rescaling risk where necessary.

moveICBackFromRefParam signature(IC = "HampIC", L2Fam = "L2ParamFamily"): moves IC in IC back to original location and scale in L2Fam (and in addition changes Lagrange multipliers accordingly), rescaling risk where necessary.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>
Neighborhood-class

Description

Class of neighborhoods of families of probability measures.

Objects from the Class

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

Slots

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

Methods

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

ProbFamily-class
Methods for Function normtype in Package 'RobAStBase'

Description

normtype-methods

Methods

normtype signature(object = "interpolrisk"): returns the slot normtype of an object of class "interpolrisk".

Examples

myrisk <- MBRisk(samplesize=100)
normtype(myrisk)

oneStepEstimator Function for the computation of one-step estimates

Description

Function for the computation of one-step estimates.

Usage

oneStepEstimator(x, IC, start = NULL,
                 useLast = getRobAStBaseOption("kStepUseLast"),
                 withUpdateInKer = getRobAStBaseOption("withUpdateInKer"),
                 IC.UpdateInKer = getRobAStBaseOption("IC.UpdateInKer"),
                 na.rm = TRUE, startArgList = NULL, ...)

Arguments

x sample
IC object of class "InfluenceCurve"
start initial estimate (for full parameter, i.e. in dimension \(k\) respective joint length of
main and nuisance part of the parameter): either a numerical value, or an object
of class "Estimate" or a function producing either a numerical value, or an
object of class "Estimate" when evaluated at \(x, \ldots\); if missing or NULL, we
use slot startPar of the L2family L2Fam from within IC.
useLast which parameter estimate (initial estimate or one-step estimate) shall be used to
fill the slots pIC, asvar and asbias of the return value.
withUpdateInKer if there is a non-trivial transform in the model with matrix $D$, shall the parameter be updated on $\text{ker}(D)$?

IC.UpdateInKer if there is a non-trivial transform in the model with matrix $D$, the IC to be used for this; if NULL the result of getboundedIC(L2Fam, D) is taken; this IC will then be projected onto $\text{ker}(D)$.

na.rm logical: if TRUE, the estimator is evaluated at complete.cases(x).

startArgList a list of arguments to be given to argument start if the latter is a function; this list by default already starts with two unnamed items, the sample x, and the model eval(lRfam(ic)); in case IC is not of class IC, the model argument L2Fam will be set to NULL.

... additional arguments

Details

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

In case IC is an object of class "IC" the slots asvar and asbias of the return value are filled (based on the initial estimate).

The default value of argument useLast is set by the global option kStepUseLast which by default is set to FALSE. In case of general models useLast remains unchanged during the computations. However, if slot Call1L2Fam of IC generates an object of class "L2GroupParamFamily" the value of useLast is changed to TRUE. Explicitly setting useLast to TRUE should be done with care as in this situation the influence curve is re-computed using the value of the one-step estimate which may take quite a long time depending on the model.

If useLast is set to TRUE and slot modifyIC of IC is filled with some function (which can be used to re-compute the IC for a different parameter), the computation of asvar, asbias and IC is based on the one-step estimate.

Value

Object of class "kStepEstimate"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>,
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


See Also

InfluenceCurve-class, kStepEstimate-class
Generic function for the computation of optimally robust ICs

Usage

optic(model, risk, ...)

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

Arguments

- **model**: probability model.
- **risk**: object of class "RiskType".
- **...**: additional parameters.

Details

The classical optimal IC which is optimal in sense of the Cramer-Rao bound is computed.

Value

Some optimally robust IC is computed.

Methods

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

*InfluenceCurve-class, RiskType-class*
**OptionalInfluenceCurve-class**

### Examples

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

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

---

**OptionalInfluenceCurve-class**

*Some helper Classes in package ‘RobAStBase’*

### Description

Some helper Classes in package 'RobAStBase': Classes OptionalInfluenceCurve, OptionalpICList, StartClass, PIClist

### Class Unions

OptionalInfluenceCurve is a class union of classes InfluenceCurve and NULL — it is the slot class of slot pIC in ALEstimate; OptionalpICList is a class union of classes pICList and NULL — it is the slot class of slot pICList in kStepEstimate; StartClass is a class union of classes function, numeric and Estimate — it is the slot class of slot start in kStepEstimate.

### List Classes

pICList is a descendant of class list which requires its members —if any— to be of class pIC.

### Methods

- `show` signature(object = "OptionalpICList"): particular show-method.
- `show` signature(object = "pICList"): particular show-method.

### Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

### References


### See Also

*InfluenceCurve, RiskType-class*
Function `outlyingPlotIC` in Package ‘RobAStBase’

Description

`outlyingPlotIC` produces an outlyingness plot based on distances applied to ICs

Usage

```r
outlyingPlotIC(data, IC.x, IC.y = IC.x, dist.x = NormType(), dist.y,
    cutoff.x = cutoff.sememp(0.95), cutoff.y = cutoff.chisq(0.95), ...,
    cutoff.quantile.x = 0.95, cutoff.quantile.y = cutoff.quantile.x,
    id.n, cex.pts = 1, lab.pts, jitt.pts = 0, alpha.trsp = NA, adj, cex.idn,
    col.idn, lty.cutoff, lwd.cutoff, col.cutoff, robCov.x = TRUE,
    robCov.y = TRUE, tf.x = data, tf.y = data, jitt.fac=10, doplot = TRUE,
    main = gettext("Outlyingness \n by means of a distance-distance plot")
)
```

Arguments

data: data coercable to `matrix`; the data at which to produce the `ddPlot`.

IC.x: object of class `IC` the influence curve to produce the distances for the x axis.

IC.y: object of class `IC` the influence curve to produce the distances for the y axis.

...: further arguments to be passed to `plot.default`, `text`, and `abline`

dist.x: object of class `NormType`; the distance for the x axis.

dist.y: object of class `NormType`; the distance for the y axis.

cutoff.x: object of class `cutoff`; the cutoff information for the x axis (the vertical line discriminating 'good' and 'bad' points).

cutoff.y: object of class `cutoff`; the cutoff information for the y axis (the horizontal line discriminating 'good' and 'bad' points).

cutoff.quantile.x: numeric; the cutoff quantile for the x axis.

cutoff.quantile.y: numeric; the cutoff quantile for the y axis.

id.n: a set of indices (or a corresponding logical vector); to select a subset of the data in argument `data`.

cex.pts: the corresponding `cex` argument for plotted points.

lab.pts: a vector of labels for the (unsubsetted) data.

jitt.pts: the corresponding `jitter` argument for plotted points; may be a vector of length 2 – for separate factors for x- and y-coordinate.
alpha.trsp alpha transparency to be added ex post to colors col.pch and col.lbl; if one-
dim and NA all colors are left unchanged. Otherwise, with usual recycling rules
alpha.trsp gets shorted/prolongated to length the data-symbols to be plotted.
Coordinates of this vector alpha.trsp with NA are left unchanged, while for
the remaining ones, the alpha channel in rgb space is set to the respective coor-
dinate value of alpha.trsp. The non-NA entries must be integers in [0,255] (0
invisible, 255 opaque).

adj the corresponding argument for text for labelling the outliers.
cex.idn the corresponding cex argument for text for labelling the outliers.
col.idn the corresponding col argument for text for labelling the outliers.
lty.cutoff the corresponding lty argument for abline for drawing the cutoff lines.
lwd.cutoff the corresponding lwd argument for abline for drawing the cutoff lines.
col.cutoff the corresponding col argument for abline for drawing the cutoff lines.
robCov.x shall x-distances be based on MCD, i.e., robust covariances (TRUE) or on clas-
sical covariance be used?
robCov.y shall y-distances be based on MCD, i.e., robust covariances (TRUE) or on clas-
sical covariance be used?
tf.x transformation for x axis: a function returning the transformed x-coordinates
when applied to the data; by default identity.
tf.y transformation for y axis: a function returning the transformed y-coordinates
when applied to the data; by default identity.
jitt.fac factor for jittering, see jitter;
doplot logical; shall a plot be produced? if FALSE only the return values are produced.
main the main title.

Details
calls a corresponding ddPlot method to produce the plot.

Value
a list (returned as invisible()) with items

id.x the indices of (possibly transformed) data (within subset id.n) beyond the x-
cutoff
id.y the indices of (possibly transformed) data (within subset id.n) beyond the y-
cutoff
id.xy the indices of (possibly transformed) data (within subset id.n) beyond the x-
cutoff and the y-cutoff
qtx the quantiles of the distances of the (possibly transformed) data in x direction
qty the quantiles of the distances of the (possibly transformed) data in y direction
cutoff.x.v the cutoff value in x direction
cutoff.y.v the cutoff value in y direction
Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

Examples

```r
if(require(ROptEst)){
  ## generates normal location and scale family with mean = -2 and sd = 3
  N0 <- NormLocationScaleFamily()
  N0.IC0 <- optIC(model = N0, risk = asCov())
  N0.Rob1 <- InfRobModel(center = N0, neighbor = ContNeighborhood(radius = 0.5))
  N0.IC1 <- optIC(model = N0.Rob1, risk = asMSE())
  xn <- c(rnorm(100), rcauchy(20)+20)
  outlyingPlotIC(xn, IC.x=N0.IC0)
  outlyingPlotIC(xn, IC.x=N0.IC1)
}
```

plot-methods

Methods for Function plot in Package ‘RobAStBase’

Description

plot-methods

Usage

plot(x, y, ...)

## S4 method for signature 'IC, missing'
plot(x, ..., withSweave = getdistrOption("withSweave"),
       main = FALSE, inner = TRUE, sub = FALSE,
       col.inner = par("col.main"), cex.inner = 0.8,
       bmar = par("mar")[1], tmar = par("mar")[3],
       with.automatic.grid = TRUE,
       with.legend = FALSE, legend = NULL, legend.bg = "white",
       legend.location = "bottomright", legend.cex = 0.8,
       withMBR = FALSE, MBRR = NA, MBR.fac = 2, col.MBR = par("col"),
       lty.MBR = "dashed", lwd.MBR = 0.8,
       x.vec = NULL, scaleX = FALSE, scaleX.fct, scaleX.inv,
       scaleY = FALSE, scaleY.fct = pnorm, scaleY.inv=qnorm,
       scaleN = 9, x.ticks = NULL, y.ticks = NULL,
       mfColRow = TRUE, to.draw.arg = NULL,
       withSubst = TRUE)

## S4 method for signature 'IC, numeric'
plot(x, y, ..., cex.pts = 1, col.pts = par("col"), pch.pts = 1, jitter.fac = 1, with.lab = FALSE, lab.pts = NULL,
       lab.font = NULL, alpha.trsp = NA, which.lbs = NULL,
       which.Order = NULL, return.Order = FALSE)
Arguments

- **x**: object of class "IC": IC to be plotted
- **y**: missing or numeric (a dataset, e.g.)
- **withSweave**: logical: if TRUE (for working with Sweave) no extra device is opened
- **main**: logical: is a main title to be used? or just as argument main in `plot.default`.
- **inner**: logical: do panels have their own titles? or character vector of / cast to length 'number of plotted dimensions'; if argument to `draw.arg` is used, this refers to a vector of length `length(to.draw.arg)`, the actually plotted dimensions. For further information, see also description of argument `main` in `plot.default`.
- **sub**: logical: is a sub-title to be used? or just as argument `sub` in `plot.default`.
- **tmar**: top margin – useful for non-standard main title sizes
- **bmar**: bottom margin – useful for non-standard sub title sizes
- **cex.inner**: magnification to be used for inner titles relative to the current setting of `cex`; as in `par`
- **col.inner**: character or integer code; color for the inner title
- **with-automatic.grid**: logical: should a grid be plotted alongside with the ticks of the axes, automatically? If TRUE a respective call to `grid` in argument `panel.first` is ignored.
- **with.legend**: logical: shall a legend be plotted?
- **legend**: either `NULL` or a list of length (number of plotted panels) of items which can be used as argument `legend` in command `legend`.
- **legend.location**: a valid argument `x` for `legend` — the place where to put the legend on the last issued plot — or a list of length (number of plotted panels) of such arguments, one for each plotted panel.
- **legend.bg**: background color for the legend
- **legend.cex**: magnification factor for the legend
- **withMBR**: logical; shall horizontal lines with min and max of MBRE be plotted for comparison?
- **MBRB**: matrix (or NA); coerced by usual recycling rules to a matrix with as many rows as plotted panels and with first column the lower bounds and the second column the upper bounds for the respective coordinates (ideally given by the MBR-IC).
- **MBR.fac**: positive factor; scales the bounds given by argument `MBRB`
- **col.MBR**: color for the MBR lines; as usual `col`-argument;
- **lty.MBR**: line type for the MBR lines; as usual `lty`-argument;
- **lwd.MBR**: line width for the MBR lines; as usual `lwd`-argument;
x.vec a numeric vector of grid points to evaluate the influence curve; by default, x.vec is NULL; then the grid is produced automatically according to the distribution of the IC. x.vec can be useful for usage with a rescaling of the x-axis to avoid that the evaluation points be selected too unevenly (i.e. on an equally spaced grid in the original scale, but then, after rescaling non-equally). The grid has to be specified in original scale; i.e.; when used with rescaling, it should be chosen non-equally spaced.

scaleX logical; shall X-axis be rescaled (by default according to the cdf of the underlying distribution)?

scaleY logical; shall Y-axis be rescaled (by default according to a probit scale)?

scaleX.fct an isotone, vectorized function mapping the domain of the IC to [0,1]; if scaleX is TRUE and scaleX.fct is missing, the cdf of the underlying observation distribution; can also be a list of functions with one list element for each of the panels to be plot.

scaleX.inv the inverse function to scale.fct, i.e., an isotone, vectorized function mapping [0,1] to the domain of the IC such that for any x in the domain, scaleX.inv(scaleX.fct(x))==x; if scaleX is TRUE and scaleX.inv is missing, the quantile function of the underlying observation distribution; can also be a list of functions with one list element for each of the panels to be plot.

scaleY.fct an isotone, vectorized function mapping for each coordinate the range of the respective coordinate of the IC to [0,1]; defaulting to the cdf of $\mathcal{N}(0,1)$.

scaleY.inv an isotone, vectorized function mapping for each coordinate the range [0,1] into the range of the respective coordinate of the IC; defaulting to the quantile function of $\mathcal{N}(0,1)$.

scaleN integer; defaults to 9; on rescaled axes, number of x and y ticks if drawn automatically;

x.ticks numeric; defaults to NULL; (then ticks are chosen automatically); if non-NULL, user-given x-ticks (on original scale);

y.ticks numeric; defaults to NULL; (then ticks are chosen automatically); if non-NULL, user-given y-ticks (on original scale); can be a list with one (numeric or NULL) item per panel

mfColRow shall default partition in panels be used — defaults to TRUE

to.draw.arg Either NULL (default; everything is plotted) or a vector of either integers (the indices of the subplots to be drawn) or characters — the names of the subplots to be drawn: these names are to be chosen either among the row names of the trafo matrix rownames(trafo(eval(x@CallL2Fam@param))) or if the last expression is NULL a vector "dim<dimnr>", dimnr running through the number of rows of the trafo matrix.

withSubst logical; if TRUE (default) pattern substitution for titles and labels is used; otherwise no substitution is used.

cex.pts size of the points of the second argument plotted, can be a vector; if it has the same length as the number of observations prior to any selection then it overrides argument cex.nonlbl and assigns individual sizes to each of the points; otherwise it is recycled to the length of selected observations.
col.pch  color of the points of the second argument plotted, can be a vector as in cex.pch.

pchNpts  symbol of the points of the second argument plotted, can be a vector as in cex.pch.

with.lab  logical; shall labels be plotted to the observations?

lab.pch  character or NULL; labels to be plotted to the observations; if NULL observation indices;

lab.font  font to be used for labels (of the observations).

alpha.trsp  alpha transparency to be added ex post to colors col.pch and col.lab; if one-
dim and NA all colors are left unchanged. Otherwise, with usual recycling rules
alpha.trsp gets shorted/prolongated to length the data-symbols to be plotted.
Coordinates of this vector alpha.trsp with NA are left unchanged, while for
the remaining ones, the alpha channel in rgb space is set to the respective coor-
dinate value of alpha.trsp. The non-NA entries must be integers in [0,255] (0
invisible, 255 opaque).

jitter.fac  jittering factor used in case of a DiscreteDistribution for plotting points of
the second argument in a jittered fashion.

which.lbs  either an integer vector with the indices of the observations to be plotted into
graph or NULL — then no observation is excluded

which.Order  we order the observations (descending) according to the norm given by normtype(object);
then which.Order either is an integer vector with the indices of the ordered ob-
servations (remaining after a possible reduction by argument which.lbs) to be
plotted into graph or NULL — then no (further) observation is excluded.

return.Order  logical; if TRUE, an order vector is returned; more specifically, the order of
the (remaining) observations given by their original index is returned (remain-
ing means: after a possible reduction by argument which.lbs, and ordering
is according to the norm given by normtype(object)); otherwise we return
invisible() as usual.

...  further parameters for plot

Details

Any parameters of plot.default may be passed on to this particular plot method.

We start describing the IC,missing-method: For main-, inner, and subtitles given as arguments
main, inner, and sub, top and bottom margins are enlarged to 5 resp. 6 by default but may also
be specified by tmar / bmar arguments. If main / inner / sub are logical then if the respective
argument is FALSE nothing is done/plotted, but if it is TRUE, we use a default main title taking up
the calling arguments in case of main, default inner titles taking up the class and (named) parameter
slots of arguments in case of inner, and a "generated on <data>"-tag in case of sub. Of course,
if main / inner / sub are character, this is used for the title; in case of inner it is then checked
whether it has correct length. If argument withSubst is TRUE, in all title and axis lable arguments,
the following patterns are substituted:

"%C"  class of argument object

"%A"  deparsed argument object

"%D"  time/date-string when the plot was generated
If argument ... contains argument ylim, this may either be as in plot.default (i.e. a vector of length 2) or a vector of length 2*(number of plotted dimensions + 2), where the first two elements are the values for ylim in panel "d", the first two are for ylim resp. xlim for panels "p" and "q", and the last 2*(number of plotted dimensions) are the values for ylim for the plotted dimensions of the L2derivative, one pair for each dimension.

The IC_numeric-method calls the IC_missing-method but in addition plots the values of a dataset into the IC.

In addition, argument ... may contain arguments panel.first, panel.last, i.e., hook expressions to be evaluated at the very beginning and at the very end of each panel (within the then valid coordinates). To be able to use these hooks for each panel individually, they may also be lists of expressions (of the same length as the number of panels and run through in the same order as the panels).

Value

The function returns invisibly a list of elements containing the information needed to compute the respective diagnostic plot. The return value allows to recover all information used to produce the plot for later use in enhanced graphics (e.g. with ggplot).

Examples

IC1 <- new("IC")
plot(IC1)
plot(IC1, main = TRUE, panel.first= grid(),
     col = "blue", cex.main = 2, cex.inner = 1)

### selection of subpanels for plotting
N <- NormLocationScaleFamily(mean=0, sd=1)
IC2 <- optIC(model = N, risk = asCov())
par(mfrow=c(1,1))
plot(IC2, main = TRUE, panel.first= grid(),
     col = "blue", cex.main = 2, cex.inner = 0.6,
     mfColRow = FALSE, to.draw.arg=c("sd"))

## xlim and ylim arguments
plot(IC2, main = TRUE, panel.first= grid(),
     ylim=c(-3,3), xlim=c(-2,3))
plot(IC2, main = TRUE, panel.first= grid(),
     ylim=c(-3,3,-1,3), xlim=c(-2,3),
     with.legend = TRUE)

data <- r(N)(30)
plot(IC2, data, panel.first= grid(),
     ylim = c(-3,3,-1,3), xlim=c(-2,3),
     cex.pts = 3, pch.pts = 1:2, col.pts="green",
     with.lab = TRUE, which.lbs = c(1:4,15:20),
     which.Order = 1:6, return.Order = TRUE)
**PlotIC**

*Wrapper function for plot method for IC*

**Description**

The wrapper PlotIC takes most of arguments to the plot method by default and gives a user possibility to run the function with low number of arguments.

**Usage**

```r
PlotIC(IC, y, ..., alpha.trsp = 100, with.legend = TRUE,
       rescale = FALSE, withCall = TRUE)
```

**Arguments**

- **IC** object of class IC
- **y** optional data argument — for plotting observations into the plot
- **...** additional parameters (in particular to be passed on to plot)
- **alpha.trsp** the transparency argument (0 to 100) for plotting the data
- **with.legend** the flag for showing the legend of the plot
- **rescale** the flag for rescaling the axes for better view of the plot
- **withCall** the flag for the call output

**Value**

invisible(NULL)

**Details**

Calls `plot` with suitably chosen defaults; if `withCall` == TRUE, the call to `plot` is returned.

**Examples**

```r
# Gamma
fam <- GammaFamily()
rfam <- InfRobModel(fam, ContNeighborhood(0.5))
IC <- optIC(model = fam, risk = asCov())
Y <- distribution(fam)
y <- r(Y)(1000)
PlotIC(IC, y, withCall = FALSE)
```
Description

We generalize function `qqplot` from package `stats` to be applicable to distribution and probability model objects. In this context, `qqplot` produces a QQ plot of data (argument `x`) against a (model) distribution. For arguments `y` of class RobModel, points at a high “distance” to the model are plotted smaller. For arguments `y` of class kStepEstimate, points at with low weight in the [p]IC are plotted bigger and their color gets faded out slowly. Graphical parameters may be given as arguments to `qqplot`.

Usage

```r
qqplot(x, y, ...)  
## S4 method for signature 'ANY,RobModel'
qqplot(x, y,
  n = length(x), withIdLine = TRUE, withConf = TRUE,
  withConf.pw = withConf, withConf.sim = withConf,
  plot.it = TRUE, xlab = deparse(substitute(x)),
  ylab = deparse(substitute(y)), ..., distance = NormType(),
  n.adj = TRUE)
## S4 method for signature 'ANY,InfRobModel'
qqplot(x, y, n = length(x), withIdLine = TRUE,
  withConf = TRUE, withConf.pw = withConf, withConf.sim = withConf,
  plot.it = TRUE, xlab = deparse(substitute(x)), ylab =
  deparse(substitute(y)), ..., cex.pts.fun = NULL, n.adj = TRUE)
## S4 method for signature 'ANY,kStepEstimate'
qqplot(x, y,
  n = length(x), withIdLine = TRUE, withConf = TRUE,
  withConf.pw = withConf, withConf.sim = withConf,
  plot.it = TRUE, xlab = deparse(substitute(x)),
  ylab = deparse(substitute(y)), ...,
  exp.cex2.lbl = -.15,
  exp.cex2.pch = -.35,
  exp.fadcol.lbl = 1.85,
  exp.fadcol.pch = 1.85,
  bg = "white")
```

Arguments

- `x` data to be checked for compatibility with distribution/model `y`.
- `y` object of class "RobModel", of class "InfRobModel" or of class "kStepEstimate".
- `n` numeric; number of quantiles at which to do the comparison.
- `withIdLine` logical; shall line `y = x` be plotted in?
withConf   logical; shall confidence lines be plotted?
withConf.pw logical; shall pointwise confidence lines be plotted?
withConf.sim logical; shall simultaneous confidence lines be plotted?
plot.it    logical; shall be plotted at all (inherited from \texttt{qqplot})?
xlab       x-label
ylab       y-label
... further parameters for method \texttt{qqplot} with signature \texttt{ANY,ProbFamily} (see \texttt{qqplot}) or with function \texttt{plot}
cex pts.fun rescaling function for the size of the points to be plotted; either NULL (default), then $\log(1+\text{abs}(x))$ is used, or a function which is then used.
n.adj      logical; shall sample size be adjusted for possible outliers according to radius of the corresponding neighborhood?
distance  a function mapping observations $x$ to the positive reals; used to determine the size of the plotted points (the larger $\text{distance}(x)$, the smaller the points are plotted.
exp.cex2.lbl for objects \texttt{kStepEstimate} based on a \texttt{[p]IC} of class \texttt{HampIC}: exponent for the weights of this \texttt{[p]IC} used to magnify the labels.
exp.cex2.pch for objects \texttt{kStepEstimate} based on a \texttt{[p]IC} of class \texttt{HampIC}: exponent for the weights of this \texttt{[p]IC} used to magnify the symbols.
exp.fadcol.lbl for objects \texttt{kStepEstimate} based on a \texttt{[p]IC} of class \texttt{HampIC}: exponent for the weights of this \texttt{[p]IC} used to find out-fading colors.
exp.fadcol.pch for objects \texttt{kStepEstimate} based on a \texttt{[p]IC} of class \texttt{HampIC}: exponent for the weights of this \texttt{[p]IC} used to find out-fading colors.
b= background color to fade against

\textbf{Details}

\texttt{qqplot signature(x = "ANY", y = "RobModel"):} produces a QQ plot of a dataset $x$ against the theoretical quantiles of distribution of robust model $y$.

\texttt{qqplot signature(x = "ANY", y = "InfRobModel"):} produces a QQ plot of a dataset $x$ against the theoretical quantiles of distribution of infinitesimally robust model $y$.

\texttt{qqplot signature(x = "ANY", y = "kStepEstimate"):} produces a QQ plot of a dataset $x$ against the theoretical quantiles of the model distribution of model at which the corresponding \texttt{kStepEstimate} $y$ had been calibrated at. By default, if the \texttt{[p]IC} of the \texttt{kStepEstimate} is of class \texttt{HampIC}, i.e.; has a corresponding weight function, points (and, if \texttt{withLab==TRUE}, labels) are scaled and faded according to this weight function. Corresponding arguments \texttt{exp.cex2.pch} and \texttt{exp.fadcol.pch} control this scaling and fading, respectively (and analogously \texttt{exp.cex2.lbl} and \texttt{exp.fadcol.lbl} for the labels). The choice of these arguments has to be done on a case-by-case basis. Positive exponents induce fading, magnification with increasing weight, for negative exponents the same is true for decreasing weight; higher (absolute) values increase the speed of fading / magnification.
Value

As for function `qqplot` from package `stats`: a list with components

- `x` The x coordinates of the points that were/would be plotted
- `y` The corresponding quantiles of the second distribution, *including NAs.*

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


See Also

`qqplot` from package `stats` – the standard QQ plot function, `qqplot` from package `distr` for comparisons of distributions, and `qqplot` from package `distrMod` (which is called immediately by this method), as well as `qqbounds`, used by `qqplot` to produce confidence intervals.

Examples

```r
qqplot(rnorm(40, mean = 15, sd = sqrt(30)), Chisq(df=15))
RobM <- InfRobModel(center = NormLocationFamily(mean=13,sd=sqrt(28)),
  neighbor = ContNeighborhood(radius = 0.4))
x <- rnorm(20, mean = 15, sd = sqrt(30))
qqplot(x, RobM)

## \donttest to reduce check time
qqplot(x, RobM, alpha.CI=0.9, add.points.CI=FALSE)

## further examples for ANY,kStepEstimator-method
## in example to roptest() in package ROptEst
```

Description

Provides information on the (intended) masking of and (non-intended) masking by other other functions in package `RobAStBase`

Usage

```r
RobAStBaseMASK(library = NULL)
```
Arguments

library a character vector with path names of R libraries, or NULL. The default value of NULL corresponds to all libraries currently known. If the default is used, the loaded packages are searched before the libraries

Value

no value is returned

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

Examples

RobAStBaseOptions()
Global Options

**kStepUseLast:** The default value of argument kStepUseLast is FALSE. Explicitly setting kStepUseLast to TRUE should be done with care as in this situation the influence curve in case of oneStepEstimator and kStepEstimator is re-computed using the value of the one- resp. k-step estimate which may take quite a long time depending on the model.

**withUpdateInKer:** if there is a non-trivial trafo in the model with matrix $D$, shall the parameter be updated on $\ker(D)$? defaults to FALSE.

**IC.UpdateInKer:** if there is a non-trivial trafo in the model with matrix $D$, the IC to be used for this; if NULL the result of getboundedicHlRfamLdI is taken; this IC will then be projected onto $\ker(D)$; defaults to NULL.

**all.verbose:** argument verbose passed on by default to many calls of optIC, radiusminimaxIC, getinfRobIC etc.; well suited for testing purposes.

**withPICList:** logical: shall slot picList of return value of kStepEstimator be filled?

**withICList:** logical: shall slot ICList of return value of kStepEstimator be filled?

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

See Also

options, getOption

Examples

```r
RobAStBaseOptions()
RobAStBaseOptions("kStepUseLast")
RobAStBaseOptions("kStepUseLast" = TRUE)
# or
RobAStBaseOptions(kStepUseLast = 1e-6)
getRobAStBaseOption("kStepUseLast")
```

Description

Control classes in package RobAStBase.

Objects from the Class

This class is virtual; that is no objects may be created.

Slots

```
  name  Object of class "character": name of the control object.
```

---

RobAStControl-class   Control classes in package RobAStBase
RobModel-class

Methods

name signature(object = "RobAStControl"): accessor function for slot name.

name<- signature(object = "RobAStControl", value = "character"): replacement function for slot name.

Author(s)

Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References


RobModel-class Robust model

Description

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

Objects from the Class

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

Slots

center Object of class "ProbFamily"
neighbor Object of class "Neighborhood"

Methods

center signature(object = "RobModel"): accessor function for slot center.
center<- signature(object = "RobModel"): replacement function for slot center.
neighbor signature(object = "RobModel"): accessor function for slot neighbor.
neighbor<- signature(object = "RobModel"): replacement function for slot neighbor.
trafo signature(object = "RobModel", param = "missing"): accessor function for slot trafo of slot center.
trafo<- signature(object = "RobModel"): replacement function for slot trafo of slot center.
Author(s)
Matthias Kohl <Matthias.Kohl@stamats.de>

References

See Also
ProbFamily-class, Neighborhood-class

RobWeight-class  Robust Weight classes

Description
Classes for robust weights.

Objects from the Class
Objects can be created by calls of the form new("RobWeight", ...).

Slots
name Object of class "character".
weight Object of class "function" — the weight function.

Methods

name signature(object = "RobWeight"): accessor function for slot name.
name<- signature(object = "RobWeight"): replacement function for slot name.
weight signature(object = "RobWeight"): accessor function for slot weight.
weight<- signature(object = "RobWeight"): replacement function for slot weight.

Author(s)
Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de>

References
samplesize-methods

See Also

InfluenceCurve-class, IC

Examples

```r
## prototype
new("RobWeight")
```

```r
myrisk <- MBRRisk(samplesize=100)
samplesize(myrisk)
samplesize(myrisk) <- 20
```

TotalVarIC

**Generating function for TotalVarIC-class**

**Description**

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

\[
\eta = c \vee A \Lambda \wedge d
\]

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

Usage

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

Arguments

name object of class "character".
CallL2Fam object of class "call": creates an object of the underlying L2-differentiable
parametric family.
Curve object of class "EuclRandVarList".
Risks object of class "list": list of risks; cf. RiskType-class.
Infos matrix of characters with two columns named method and message: additional
informations.
clipLo negative real: lower clipping bound.
clipUp positive real: lower clipping bound.
stand matrix: standardizing matrix
w BdStWeight: weight object
lowerCase optional constant for lower case solution.
neighborRadius radius of the corresponding (unconditional) contamination neighborhood.
biastype BiasType: type of the bias
normtype NormType: type of the norm
modifyIC object of class "OptionalFunction": function of two arguments, which are an
L2 parametric family and an optional influence curve. Returns an object of class
"IC". This function is mainly used for internal computations!

Value

Object of class "TotalVarIC"

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

sertation.
**TotalVarIC-class**

**See Also**

IC-class, ContIC

**Examples**

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

---

**TotalVarIC-class**

Influence curve of total variation type

**Description**

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

\[
\eta = c \lor AA \land d
\]

with lower clipping bound \( c \), upper clipping bound \( d \) and standardizing matrix \( A \). \( \Lambda \) stands for the L2 derivative of the corresponding L2 differentiable parametric family which can be created via \texttt{callL2Fam}.

**Objects from the Class**

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

**Slots**

- \texttt{callL2Fam} object of class "call": creates an object of the underlying L2-differentiable parametric family.
- \texttt{name} object of class "character".
- \texttt{Curve} object of class "EuclRandVarList".
- \texttt{modifyIC} Object of class "OptionalFunction": function of two arguments, which are an L2 parametric family and an optional influence curve. Returns an object of class "IC". This slot is mainly used for internal computations!
- \texttt{Risks} object of class "list": list of risks; cf. \texttt{RiskType-class}.
- \texttt{Infos} object of class "matrix" with two columns named method and message: additional informations.
- \texttt{clipLo} object of class "numeric": lower clipping bound.
- \texttt{clipUp} object of class "numeric": upper clipping bound.
- \texttt{stand} object of class "matrix": standardizing matrix.
- \texttt{weight} object of class "BdStWeight": weight function
- \texttt{biastype} object of class "BiasType": bias type (symmetric/onsided/asymmetric)
- \texttt{normtype} object of class "NormType": norm type (Euclidean, information/self-standardized)
- \texttt{neighborRadius} object of class "numeric": radius of the corresponding (unconditional) contamination neighborhood.
Extends

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

Methods

CallL2Fam<- signature(object = "TotalVarIC"): replacement function for slot CallL2Fam.
clipLo signature(object = "TotalVarIC"): accessor function for slot clipLo.
clipLo<- signature(object = "TotalVarIC"): replacement function for slot clipLo.
clipUp signature(object = "TotalVarIC"): accessor function for slot clipUp.
clipUp<- signature(object = "TotalVarIC"): replacement function for slot clipUp.
clip signature(x1 = "TotalVarIC"): returns clipUp-clipLo.
stand<- signature(object = "TotalVarIC"): replacement function for slot stand.
lowerCase<- signature(object = "TotalVarIC"): replacement function for slot lowerCase.
neighbor signature(object = "TotalVarIC"): generates an object of class "TotalVarNeighborhood" with radius given in slot neighborRadius.
generateIC signature(neighbor = "TotalVarNeighborhood", L2Fam = "L2ParamFamily"): generate an object of class "TotalVarIC". Rarely called directly.
show signature(object = "TotalVarIC")

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

IC-class, ContIC, HampIC-class

Examples

ICI <- new("TotalVarIC")
plot(ICI)
TotalVarNeighborhood Class

Generating function for TotalVarNeighborhood-class

Description
Generates an object of class "TotalVarNeighborhood".

Usage
TotalVarNeighborhood(radius = 0)

Arguments
radius non-negative real: neighborhood radius.

Value
Object of class "ContNeighborhood"

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

References

See Also
TotalVarNeighborhood-class

Examples
TotalVarNeighborhood()

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

Total variation neighborhood

Description

Class of (unconditional) total variation neighborhoods.

Objects from the Class

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

Slots

- `type`: Object of class "character": "(uncond.) total variation neighborhood".
- `radius`: Object of class "numeric": neighborhood radius.

Extends

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

Methods

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

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

`TotalVarNeighborhood, UncondNeighborhood-class`

Examples

`new("TotalVarNeighborhood")`
Description

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

Objects from the Class

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

Slots

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

Extends

Class "Neighborhood", directly.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References


See Also

Neighborhood-class
Index

*Topic classes
  ALEstimate-class, 4
  BdStWeight-class, 6
  biastype-methods, 7
  BoundedWeight-class, 7
  ContIC-class, 18
  ContNeighborhood-class, 21
  cutoff-class, 23
  FixRobModel-class, 30
  getRiskFctBV-methods, 36
  HampelWeight-class, 41
  HampIC-class, 42
  IC-class, 45
  InfluenceCurve-class, 47
  InfRobModel-class, 56
  interpolRisk-class, 57
  kStepEstimate-class, 58
  kStepEstimator.start-methods, 63
  MEstimate-class, 68
  movToRef-methods, 69
  Neighborhood-class, 71
  normtype-methods, 72
  OptionalInfluenceCurve-class, 75
  RobAStControl-class, 88
  RobModel-class, 89
  RobWeight-class, 90
  samplesize-methods, 91
  TotalVarIC-class, 93
  TotalVarNeighborhood-class, 96
  UncondNeighborhood-class, 97

*Topic distribution
  plot-methods, 78
  qqplot, 84
  RobAStBaseMASK, 86

*Topic documentation
  RobAStBaseMASK, 86

*Topic hplot
  cutoff, 22
  ddPlot-methods, 24
  outlyingPlotIC, 76
  qqplot, 84

*Topic methods
  ddPlot-methods, 24
  masked-methods, 67
  plot-methods, 78

*Topic misc
  RobAStBaseOptions, 87

*Topic models
  ContNeighborhood, 20
  ContNeighborhood-class, 21
  FixRobModel, 29
  FixRobModel-class, 30
  getBoundedIC, 34
  InfRobModel, 55
  InfRobModel-class, 56
  Neighborhood-class, 71
  RobModel-class, 89
  TotalVarNeighborhood, 95
  TotalVarNeighborhood-class, 96
  UncondNeighborhood-class, 97

*Topic package
  RobAStBase-package, 3

*Topic programming
  RobAStBaseMASK, 86

*Topic robust
  checkIC, 9
  comparePlot-methods, 11
  ContIC, 17
  evalIC, 28
  generateIC, 31
  generateIC.fct-methods, 32
  getBiasIC, 33
  getFin, 35
  getRiskIC, 37
  getweight-methods, 39
  IC, 44
  IC-class, 45
  InfluenceCurve, 46
INDEX

99

influenceCurve-class, 47
infoPlot, 50
kStepEstimator, 60
locMEstimator, 64
makeIC-methods, 65
oneStepEstimator, 72
optIC, 74
OptionalInfluenceCurve-class, 75
RobASTBaseOptions, 87
TotalVarIC, 91
TotalVarIC-class, 93

*Topic
univar
kStepEstimator, 60
locMEstimator, 64
oneStepEstimator, 72

abline, 26, 77
addInfo<-(InfluenceCurve-class), 47
addInfo<-, InfluenceCurve-method
(InfluenceCurve-class), 47
addRisk<-(InfluenceCurve-class), 47
addRisk<-, InfluenceCurve-method
(InfluenceCurve-class), 47
ALEstimate-class, 4
asbias (ALEstimate-class), 4
asbias, ALEstimate-method
(ALEstimate-class), 4

BdStWeight-class, 6
biastype (biastype-methods), 7
biastype, HampelWeight-method (HampelWeight-class), 42
biastype, interpolRisk-method
(biastype-methods), 7
biastype-methods, 7
BoundedWeight-class, 7

CallL2Fam (IC-class), 45
CallL2Fam, IC-method (IC-class), 45
CallL2Fam<-(IC-class), 45
CallL2Fam<-, ContIC-method
(ContIC-class), 18
CallL2Fam<-, IC-method (IC-class), 45
CallL2Fam<-, TotalVarIC-method
(TotalVarIC-class), 93

cent (ContIC-class), 18
cent, ContIC-method (ContIC-class), 18
cent, HampelWeight-method
(HampelWeight-class), 41
cent<-(ContIC-class), 18
cent<-, ContIC-method (ContIC-class), 18
cent<-, HampelWeight-method
(HampelWeight-class), 41
center (RobModel-class), 89
center, RobModel-method
(RobModel-class), 89
center<-(RobModel-class), 89
center<-, RobModel-method
(RobModel-class), 89

checkIC, 9
checkIC, IC, L2ParamFamily-method
(IC-class), 45
checkIC, IC, missing-method (IC-class), 45
clip, 68
clip (masked-methods), 67
clip, ANY-method (masked-methods), 67
clip, BoundedWeight-method
(BoundedWeight-class), 7
clip, ContIC-method (ContIC-class), 18
clip, TotalVarIC-method
(TotalVarIC-class), 93
clip-methods (masked-methods), 67
clip<-(ContIC-class), 18
clip<-, BoundedWeight-method
(BoundedWeight-class), 7
clip<-, ContIC-method (ContIC-class), 18
clipLo (TotalVarIC-class), 93
clipLo, TotalVarIC-method
(TotalVarIC-class), 93
clipLo<-(TotalVarIC-class), 93
clipLo<-, TotalVarIC-method
(TotalVarIC-class), 93
clipUp (TotalVarIC-class), 93
clipUp, TotalVarIC-method
(TotalVarIC-class), 93
clipUp<-(TotalVarIC-class), 93
clipUp<-, TotalVarIC-method
(TotalVarIC-class), 93

ComparePlot, 10
comparePlot (comparePlot-methods), 11
comparePlot, IC, IC-method
(comparePlot-methods), 11
comparePlot-methods, 11

confint, ALEstimate, asymmetricBias-method
(ALEstimate-class), 4
confint, ALEstimate, missing-method
(ALEstimate-class), 4
INDEX

confint, ALEstimate, onesidedBias-method (ALEstimate-class), 4
cconfint, ALEstimate, symmetricBias-method (ALEstimate-class), 4
ContIC, 17, 18, 20, 93, 94
ContIC-class, 18
ContNeighborhood, 20, 21
ContNeighborhood-class, 21
Curve, InfluenceCurve-method (InfluenceCurve-class), 47
Curve, InfluenceCurve-method (InfluenceCurve-class), 47
cutoff, 22, 23, 24
cutoff-class, 23
cutoff, quantile (cutoff-class), 23
cutoff, quantile, method (cutoff-class), 23
cutoff, quantile< (cutoff-class), 23
cutoff, quantile<, method (cutoff-class), 23
ddPlot, 23, 24, 77
ddPlot (ddPlot-methods), 24
ddPlot, data.frame-method (ddPlot-methods), 24
ddPlot, matrix-method (ddPlot-methods), 24
ddPlot, numeric-method (ddPlot-methods), 24
ddPlot-methods, 24
Domain, InfluenceCurve-method (InfluenceCurve-class), 47
evalIC, 28
evalIC, IC, method (IC-class), 45
evalIC, IC, numeric-method (IC-class), 45
fct, cutoff-method (cutoff-class), 23
FixRobModel, 29, 30
FixRobModel-class, 30
generateIC.fct-methods, 32
generateIC, 31
generateIC, ContNeighborhood, L2ParamFamily-method (ContIC-class), 18
generateIC, TotalVarNeighborhood, L2ParamFamily-method (TotalVarIC-class), 93
generateIC.fct (generateIC.fct-methods), 32
generateIC.fct, UncondNeighborhood, L2ParamFamily-method (generateIC.fct-methods), 32
getBiasIC, 33, 38
getBiasIC, IC, UnconfNeighborhood-method (getBiasIC), 33
getBiasIC-methods (getBiasIC), 33
getBoundedIC, 34
getFiRisk, 35
getFiRisk, fiUnOvShoot, Norm, ContNeighborhood-method (getFiRisk), 35
getFiRisk, fiUnOvShoot, Norm, TotalVarNeighborhood-method (getFiRisk), 35
getFiRisk-methods (getFiRisk), 35
getOption, 88
getRiskFctBV (getRiskFctBV-methods), 36
getRiskFctBV, asRisk, ANY-method (getRiskFctBV-methods), 36
getRiskFctBV, asMSE, ANY-method (getRiskFctBV-methods), 36
getRiskFctBV, asSemivar, asymmetricBias-method (getRiskFctBV-methods), 36
getRiskFctBV, asSemivar, onesidedBias-method (getRiskFctBV-methods), 36
getRiskFctBV, interpolRisk, ANY-method (getRiskFctBV-methods), 36
getRiskFctBV-methods, 36
getRiskIC, 37, 39
getRiskIC, IC, asBias, UnconfNeighborhood, L2ParamFamily-method (getRiskIC), 37
getRiskIC, IC, asBias, UnconfNeighborhood, missing-method (getRiskIC), 37
getRiskIC, IC, asCov, missing, L2ParamFamily-method (getRiskIC), 37
getRiskIC, IC, asCov, missing, missing-method (getRiskIC), 37
getRiskIC, IC, asMSE, UnconfNeighborhood, L2ParamFamily-method (getRiskIC), 37
getRiskIC, IC, asMSE, UnconfNeighborhood, missing-method (getRiskIC), 37
getRiskIC, IC, fiUnOvShoot, ContNeighborhood, missing-method (getRiskIC), 37
getRiskIC, IC, fiUnOvShoot, TotalVarNeighborhood, missing-method (getRiskIC), 37
getRiskIC, IC, trAsCov, missing, L2ParamFamily-method (getRiskIC), 37
getRiskIC, IC, trAsCov, missing, missing-method (getRiskIC), 37
getRiskIC-methods, 37
getRiskIC, TotalVarIC, asUnOvShoot, UnconfNeighborhood, missing-method (getRiskIC), 37
INDEX

getRiskIC-methods (getRiskIC), 37
getRobastBaseOption (RobastBaseOptions), 87
gettexf, 27
getweight, 6, 41
getweight (getweight-methods), 39
getweight, BdStWeight, TotalVarNeighborhood, BiasType-
   (getweight-methods), 39
getweight, HampelWeight, ContNeighborhood, BiasType-
   (getweight-methods), 39
getweight, HampelWeight, ContNeighborhood, BiasType-
   (getweight-methods), 39
getweight-methods, 39
HampelWeight-class, 41
HampIC-class, 42
IC, 7, 8, 42, 44, 46, 91
IC-class, 45
ICList (kStepEstimate-class), 58
ICList, kStepEstimate-method
   (kStepEstimate-class), 58
InfluenceCurve, 46, 48, 75
InfluenceCurve-class, 47
InfoPlot, 49
infoPlot, 50
infoPlot, IC-method (infoPlot), 50
infoPlot-methods (infoPlot), 50
Infos (InfluenceCurve-class), 47
Infos, InfluenceCurve-method
   (InfluenceCurve-class), 47
Infos<-, InfluenceCurve-method
   (InfluenceCurve-class), 47
InfRobModel, 55, 57
InfRobModel-class, 56
interpolRisk-class, 57
jitter, 77
kStepEstimate-class, 58
kStepEstimator, 60, 64
kStepEstimator.start
   (kStepEstimator.start-methods), 63
kStepEstimator.start, Estimate-method
   (kStepEstimator.start-methods), 63
kStepEstimator.start, function-method
   (kStepEstimator.start-methods), 63
kStepEstimator.start, numeric-method
   (kStepEstimator.start-methods), 63
kStepEstimator.start-methods, 63
ksteps (kStepEstimate-class), 58
kstepBiasEstimate-method (kStepEstimate-class), 58
kstepEstimate
   (RobastBaseOptions), 87
kstepEstimator
   (RobastBaseOptions), 87
kstepBiasEstimate-method
   (RobastBaseOptions), 87
legend, 12, 54, 79
locMEstimator, 64
locMEstimator, numeric, InfluenceCurve-method
   (locMEstimator), 64
locMEstimator-methods (locMEstimator), 64
lowerCase (HampIC-class), 42
lowerCase, HampIC-method (HampIC-class), 42
lowerCase<-, ContIC-class), 18
lowerCase<-, ContIC-method
   (ContIC-class), 18
lowerCase<-, TotalVarIC-method
   (TotalVarIC-class), 93
makeIC (makeIC-methods), 65
makeIC, function, L2ParamFamily-method
   (makeIC-methods), 65
makeIC, IC, L2ParamFamily-method
   (makeIC-methods), 65
makeIC, IC, missing-method
   (makeIC-methods), 65
makeIC, list, L2ParamFamily-method
   (makeIC-methods), 65
makeIC-methods, 65
Map, InfluenceCurve-method
   (InfluenceCurve-class), 47
masked-methods, 67
maskedMethods (masked-methods), 67
MASKING (RobastBaseMask), 86
MBRRisk (interpolRisk-class), 57
MBRRisk-class (interpolRisk-class), 57
MEEstimate-class, 68
minbiasweight, 6, 41
minbiasweight (getweight-methods), 39
minbiasweight, BdStWeight, TotalVarNeighborhood, BiasType-
   (getweight-methods), 39

101
minbiasweight, HampelWeight, ContNeighborhood, asymmetricBiasMethod
  (getWeight-methods), 39
minbiasweight, HampelWeight, ContNeighborhood, BiastypeMethod
  (getWeight-methods), 39
minbiasweight, HampelWeight, ContNeighborhood, one-sidedRobAStControl-method
  (getWeight-methods), 39
minbiasweight, HampelWeight, ContNeighborhood, symmetricBiasMethod
  (getWeight-methods), 39
modifyIC(IC-class), 45
modifyIC, IC-method (IC-class), 45
moveICBackFromRefParam
  (moveToRef-methods), 69
moveICBackFromRefParam, HampIC, L2ParamFamily-method
  (moveToRef-methods), 69
moveICBackFromRefParam, IC, L2LocationFamily-method
  (moveToRef-methods), 69
moveICBackFromRefParam, IC, L2LocationScaleFamily-method
  (moveToRef-methods), 69
moveICBackFromRefParam, IC, L2ScaleFamily-method
  (moveToRef-methods), 69
moveICBackFromRefParam-methods
  (moveToRef-methods), 69
moveL2Fam2RefParam
  (moveToRef-methods), 69
moveL2Fam2RefParam, L2LocationFamily-method
  (moveToRef-methods), 69
moveL2Fam2RefParam, L2LocationScaleFamily-method
  (moveToRef-methods), 69
moveL2Fam2RefParam, L2ParamFamily-method
  (moveToRef-methods), 69
moveL2Fam2RefParam, L2ScaleFamily-method
  (moveToRef-methods), 69
moveL2Fam2RefParam-methods
  (moveToRef-methods), 69
moveToRef-methods, 69
Mroot
  (MEstimate-class), 68
Mroot, MEstimate-method
  (MEstimate-class), 68
name
  (cutoff-class), 23
name, InfluenceCurve-method
  (InfluenceCurve-class), 47
name, RobAStControl-method
  (RobAStControl-class), 88
name, RobModel-method
  (RobModel-class), 89
NAME, 86
name, cutoff-method
  (cutoff-class), 23
name, InfluenceCurve-method
  (InfluenceCurve-class), 47
name, RobAStControl-method
  (RobAStControl-class), 88
name, RobModel-method
  (RobModel-class), 89
name, RobWeight-method
  (RobWeight-class), 90
neighbor (RobModel-class), 89
neighbor, ContIC-method (ContIC-class), 18
neighbor, RobModel-method
  (RobModel-class), 89
neighbor, TotalVarIC-method
  (TotalVarIC-class), 93
neighbor<- (RobModel-class), 89
neighbor<-, InfRobModel-method
  (InfRobModel-class), 56
neighbor<-, RobModel-method
  (RobModel-class), 89
neighborRadius (HampIC-class), 42
neighborRadius<- (HampIC-class), 42
neighborRadius-, HampIC-method
  (HampIC-class), 42
normtype (normtype-methods), 72
normtype, HampIC-method (HampIC-class), 42
normtype, interpolRisk-method
  (normtype-methods), 72
normtype-methods, 72
OMSRisk
  (interpolRisk-class), 57
OMSRisk-class (interpolRisk-class), 57
oneStepEstimator, 72
optIC, 74
optIC, L2ParamFamily, asCov-method
  (optIC), 74
optIC-methods (optIC), 74
OptionalInfluenceCurve-class, 75
OptionalICList-class
  (OptionalInfluenceCurve-class), 75
options, 88
outlyingPlotIC, 24, 76
INDEX

par, 12, 51, 79
pIC(ALEstimate-class), 4
pIC, ALEstimate-method
(ALEstimate-class), 4
pICList (kStepEstimate-class), 58
pICList, kStepEstimate-method
(kStepEstimate-class), 58
pICList-class
(OptionalInfluenceCurve-class), 75
plot, 15
plot(plot-methods), 78
plot, IC, missing-method (plot-methods), 78
plot, IC, numeric-method (plot-methods), 78
plot-methods, 78
plot.default, 11, 12, 50, 79
PlotIC, 83
qqbounds, 86
qqplot, 84, 84, 85, 86
qqplot, ANY, InfRobModel-method (qqplot), 84
qqplot, ANY, kStepEstimate-method
(qqplot), 84
qqplot, ANY, RobModel-method (qqplot), 84
qqplot-methods (qqplot), 84
radius (Neighborhood-class), 71
radius, Neighborhood-method
(Neighborhood-class), 71
radius<-(Neighborhood-class), 71
radius<-, Neighborhood-method
(Neighborhood-class), 71
Range, InfluenceCurve-method
(InfluenceCurve-class), 47
Risks (InfluenceCurve-class), 47
Risks, InfluenceCurve-method
(InfluenceCurve-class), 47
Risks<-(InfluenceCurve-class), 47
Risks<-, InfluenceCurve-method
(InfluenceCurve-class), 47
RMXRRisk (interpolRisk-class), 57
RMXRRisk-class (interpolRisk-class), 57
RobAStBase (RobAStBase-package), 3
RobAStBase-package, 3
RobAStBaseMASK, 86
RobAStBaseOptions, 87
RobAStControl-class, 88
RobModel-class, 89
RobWeight-class, 90
samplesize (samplesize-methods), 91
samplesize, interpolRisk-method
(samplesize-methods), 91
samplesize<-(samplesize-methods), 91
samplesize<-, interpolRisk-method
(samplesize-methods), 91
show, ALEstimate-method
(ALEstimate-class), 4
show, ContIC-method (ContIC-class), 18
show, FixRobModel-method
(FixRobModel-class), 30
show, IC-method (IC-class), 45
show, InfluenceCurve-method
(InfluenceCurve-class), 47
show, InfRobModel-method
(InfRobModel-class), 56
show, kStepEstimate-method
(kStepEstimate-class), 58
show, MEstimate-method
(MEstimate-class), 68
show, Neighborhood-method
(Neighborhood-class), 71
show, OptionalpICList-method
(OptionalInfluenceCurve-class), 75
show, pICList-method
(OptionalInfluenceCurve-class), 75
show, TotalVarIC-method
(TotalVarIC-class), 93
stand (HampIC-class), 42
stand, BdStWeight-method
(BdStWeight-class), 6
stand, HampIC-method (HampIC-class), 42
stand<-(ContIC-class), 18
stand<-, BdStWeight-method
(BdStWeight-class), 6
stand<-, ContIC-method (ContIC-class), 18
stand<-, TotalVarIC-method
(TotalVarIC-class), 93
start, 68
start (masked-methods), 67
start, ANY-method (masked-methods), 67
INDEX

start,kStepEstimate-method
  (kStepEstimate-class), 58
start-methods (masked-methods), 67
StartClass-class
  (OptionalInfluenceCurve-class), 75
startval (kStepEstimate-class), 58
startval,kStepEstimate-method
  (kStepEstimate-class), 58
steps (kStepEstimate-class), 58
steps,kStepEstimate-method
  (kStepEstimate-class), 58

text, 26, 77
TotalVarIC, 91
TotalVarIC-class, 93
TotalVarNeighborhood, 95, 96
TotalVarNeighborhood-class, 96
trafo,RobModel,missing-method
  (RobModel-class), 89
trafo<-,RobModel-method
  (RobModel-class), 89
type,Neighborhood-method
  (Neighborhood-class), 71

uksteps (kStepEstimate-class), 58
uksteps,kStepEstimate-method
  (kStepEstimate-class), 58
UncondNeighborhood-class, 97
ustartval (kStepEstimate-class), 58
ustartval,kStepEstimate-method
  (kStepEstimate-class), 58

weight (RobWeight-class), 90
weight,HampIC-method (HampIC-class), 42
weight,RobWeight-method
  (RobWeight-class), 90
weight<-(RobWeight-class), 90
weight<-,RobWeight-method
  (RobWeight-class), 90
weight<--methods (RobWeight-class), 90