

# Package ‘RobLoxBioC’

January 17, 2012

**Version** 0.8.2

**Date** 2012-01-16

**Title** Infinitesimally robust estimators for preprocessing omics data

**Description** Functions for the determination of optimally robust influence curves and estimators for preprocessing omics data, in particular gene expression data.

**Depends** R(>= 2.14.0), methods, Biobase, affy, beadarray, distr, RobLox, lattice, RColorBrewer

**Author** Matthias Kohl <Matthias.Kohl@stamats.de>

**Maintainer** Matthias Kohl <Matthias.Kohl@stamats.de>

**LazyLoad** yes

**ByteCompile** yes

**License** LGPL-3

**URL** <http://robast.r-forge.r-project.org/>

**Encoding** latin1

**LastChangedDate** {`$LastChangedDate`: 2012-01-16 22:36:38 +0100 (Mo, 16. Jan 2012) `$`}

**LastChangedRevision** {`$LastChangedRevision`: 457 `$`}

**SVNRevision** 454

**Repository** CRAN

**Date/Publication** 2012-01-17 21:19:28

## R topics documented:

RobLoxBioC-package	2
KolmogorovMinDist	3
robloxbioc	5
SimStudies	9

<b>Index</b>	<b>11</b>
--------------	-----------

---

RobLoxBioC-package      *Infinitesimally robust estimators for preprocessing omics data*

---

## Description

Functions for the determination of optimally robust influence curves and estimators for preprocessing omics data, in particular gene expression data.

## Details

Package: RobLoxBioC  
Version: 0.8.1  
Date: 2012-01-16  
Depends: R(>= 2.14.0), methods, Biobase, affy, beadarray, distr, RobLox, lattice, RColorBrewer  
LazyLoad: yes  
ByteCompile: yes  
License: LGPL-3  
URL: <http://robast.r-forge.r-project.org/>  
SVNRevision: 457  
Encoding: latin1

## 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)

Matthias Kohl <Matthias.Kohl@stamats.de>

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

## References

- Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation.
- Kohl M. and Deigner H.P. (2009). Using infinitesimally robust estimators for preprocessing gene expression data. In preparation.
- Rieder, H. (1994) *Robust Asymptotic Statistics*. New York: Springer.
- Rieder, H., Kohl, M. and Ruckdeschel, P. (2008) The Costs of not Knowing the Radius. *Statistical Methods and Applications* 17(1) 13-40.

**See Also**

[roblox](#), [rowRoblox](#)

**Examples**

```
library(RobLoxBioC)
```

---

KolmogorovMinDist	<i>Generic Function for Computing Minimum Kolmogorov Distance for Biological Data</i>
-------------------	---

---

**Description**

Generic function for computing minimum Kolmogorov distance for biological data.

**Usage**

```
KolmogorovMinDist(x, D, ...)

## S4 method for signature 'matrix, Norm'
KolmogorovMinDist(x, D, mad0 = 1e-4)

## S4 method for signature 'AffyBatch, AbscontDistribution'
KolmogorovMinDist(x, D, bg.correct = TRUE, pmcorrect = TRUE,
                  verbose = TRUE)

## S4 method for signature 'BeadLevelList, AbscontDistribution'
KolmogorovMinDist(x, D, log = FALSE, imagesPerArray = 1, what = "G",
                  probes = NULL, arrays = NULL)
```

**Arguments**

x	biological data.
D	object of class "UnivariateDistribution".
...	additional parameters.
mad0	scale estimate used if computed MAD is equal to zero. Median and MAD are used as start parameter for optimization.
bg.correct	if TRUE MAS 5.0 background correction is performed; confer <a href="#">bg.correct.mas</a> .
pmcorrect	if TRUE log <sub>2</sub> (PM/MM) is used. If FALSE only log <sub>2</sub> (PM) is used.
verbose	logical: if TRUE, some messages are printed.
log	if TRUE, then the log <sub>2</sub> intensities for each bead-type are summarized.
imagesPerArray	Specifies how many images (strips) there are per array. Normally 1 for a SAM and 1 or 2 for a BeadChip. The images (strips) from the same array will be combined so that each column in the output represents a sample.

what	character string specifying which intensities/values to summarize. See <a href="#">getArrayData</a> for a list of possibilities.
probes	Specify particular probes to summarize. If left NULL then all the probes on the first array are used.
arrays	integer (scalar or vector) specifying the strips/arrays to summarize. If NULL, then all strips/arrays are summarized.

### Details

The minimum Kolmogorov distance is computed for each row of a matrix, each Affymetrix probe, or each Illumina bead, respectively.

So far, only the minimum distance to the set of normal distributions can be computed.

### Value

List with components `dist` containing a numeric vector or matrix with minimum Kolmogorov distances and `n` a numeric vector or matrix with the corresponding sample sizes.

### Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

### References

- Huber, P.J. (1981) *Robust Statistics*. New York: Wiley.  
 Rieder, H. (1994) *Robust Asymptotic Statistics*. New York: Springer.

### See Also

[KolmogorovDist](#), [MDEstimator](#)

### Examples

```
set.seed(123) # to have reproducible results for package checking

## matrix method for KolmogorovMinDist
ind <- rbinom(200, size=1, prob=0.05)
X <- matrix(rnorm(200, mean=ind*3, sd=(1-ind) + ind*9), nrow = 2)
KolmogorovMinDist(X, D = Norm())

## using Affymetrix-Data
data(SpikeIn)
probes <- log2(pm(SpikeIn))
(res <- KolmogorovMinDist(probes, Norm()))
boxplot(res$dist)

## Not run:
## "Not run" just because of computation time
require(affydata)
data(Dilution)
```

```

res <- KolmogorovMinDist(Dilution[,1], Norm())
summary(res$dist)
boxplot(res$dist)
plot(res$n, res$dist, pch = 20, main = "Kolmogorov distance vs. sample size",
      xlab = "sample size", ylab = "Kolmogorov distance",
      ylim = c(0, max(res$dist)))
uni.n <- min(res$n):max(res$n)
lines(uni.n, 1/(2*uni.n), col = "orange", lwd = 2)
legend("topright", legend = "minimal possible distance", fill = "orange")

## End(Not run)

## using Illumina-Data
## Not run:
## "Not run" just because of computation time
data(BLData)
res <- KolmogorovMinDist(BLData, Norm(), arrays = 1)
res1 <- KolmogorovMinDist(BLData, log = TRUE, Norm(), arrays = 1)
summary(cbind(res$dist, res1$dist))
boxplot(list(res$dist, res1$dist), names = c("raw", "log-raw"))
sort(unique(res1$n))
plot(res1$n, res1$dist, pch = 20, main = "Kolmogorov distance vs. sample size",
      xlab = "sample size", ylab = "Kolmogorov distance",
      ylim = c(0, max(res1$dist)), xlim = c(min(res1$n), 56))
uni.n <- min(res1$n):56
lines(uni.n, 1/(2*uni.n), col = "orange", lwd = 2)
legend("topright", legend = "minimal possible distance", fill = "orange")

## End(Not run)

```

---

robloxbioc

*Generic Function for Preprocessing Biological Data*


---

## Description

Generic function for preprocessing biological data using optimally robust (rmx) estimators; confer Rieder (1994), Kohl (2005), Rieder et al (2008).

## Usage

```

robloxbioc(x, ...)

## S4 method for signature 'matrix'
robloxbioc(x, eps = NULL, eps.lower = 0, eps.upper = 0.05, steps = 3L,
           fsCor = TRUE, mad0 = 1e-4)

## S4 method for signature 'AffyBatch'
robloxbioc(x, bg.correct = TRUE, pm.correct = TRUE, normalize = FALSE,
           add.constant = 32, verbose = TRUE, eps = NULL,

```

```
eps.lower = 0, eps.upper = 0.05, steps = 3L, fsCor = TRUE,
mad0 = 1e-4, contrast.tau = 0.03, scale.tau = 10,
delta = 2^(-20), sc = 500)
```

```
## S4 method for signature 'BeadLevelList'
robloxbioc(x, log = TRUE, imagesPerArray = 1, what = "G", probes = NULL,
arrays = NULL, eps = NULL, eps.lower = 0, eps.upper = 0.05,
steps = 3L, fsCor = TRUE, mad0 = 1e-4)
```

## Arguments

<code>x</code>	biological data.
<code>...</code>	additional parameters.
<code>eps</code>	positive real ( $0 < \text{eps} \leq 0.5$ ): amount of gross errors. See details below.
<code>eps.lower</code>	positive real ( $0 \leq \text{eps.lower} \leq \text{eps.upper}$ ): lower bound for the amount of gross errors. See details below.
<code>eps.upper</code>	positive real ( $\text{eps.lower} \leq \text{eps.upper} \leq 0.5$ ): upper bound for the amount of gross errors. See details below.
<code>steps</code>	positive integer. k-step is used to compute the optimally robust estimator.
<code>fsCor</code>	logical: perform finite-sample correction. See function <a href="#">finiteSampleCorrection</a> .
<code>mad0</code>	scale estimate used if computed MAD is equal to zero
<code>bg.correct</code>	if TRUE MAS 5.0 background correction is performed; confer <a href="#">bg.correct.mas</a> .
<code>pmcorrect</code>	method used for PM correction; TRUE calls an algorithm which is comparable to the algorithm of MAS 5.0; confer <a href="#">pmcorrect.mas</a> . If FALSE only the PM intensities are used.
<code>normalize</code>	logical: if TRUE, Affymetrix scale normalization is performed.
<code>add.constant</code>	constant added to the MAS 5.0 expression values before the normalization step. Improves the variance of the measure one no longer divides by numbers close to 0 when computing fold-changes.
<code>verbose</code>	logical: if TRUE, some messages are printed.
<code>contrast.tau</code>	a number denoting the contrast tau parameter; confer the MAS 5.0 PM correction algorithm.
<code>scale.tau</code>	a number denoting the scale tau parameter; confer the MAS 5.0 PM correction algorithm.
<code>delta</code>	a number denoting the delta parameter; confer the MAS 5.0 PM correction algorithm.
<code>sc</code>	value at which all arrays will be scaled to.
<code>log</code>	if TRUE, then the log <sub>2</sub> intensities for each bead-type are summarized.
<code>imagesPerArray</code>	Specifies how many images (strips) there are per array. Normally 1 for a SAM and 1 or 2 for a BeadChip. The images (strips) from the same array will be combined so that each column in the output represents a sample.
<code>what</code>	character string specifying which intensities/values to summarize. See <a href="#">getArrayData</a> for a list of possibilities.

probes	Specify particular probes to summarize. If left NULL then all the probes on the first array are used.
arrays	integer (scalar or vector) specifying the strips/arrays to summarize. If NULL, then all strips/arrays are summarized.

## Details

The optimally-robust resp. the radius-minimax (rmx) estimator for normal location and scale is used to preprocess biological data. The computation uses a k-step construction with median and MAD as starting estimators; cf. Rieder (1994) and Kohl (2005).

If the amount of gross errors (contamination) is known, it can be specified by `eps`. The radius of the corresponding infinitesimal contamination neighborhood (infinitesimal version of Tukey's gross error model) is obtained by multiplying `eps` by the square root of the sample size.

If the amount of gross errors (contamination) is unknown, which is typically the case, try to find a rough estimate for the amount of gross errors, such that it lies between `eps.lower` and `eps.upper`.

If `eps` is NULL, the radius-minimax (rmx) estimator in sense of Rieder et al. (2001, 2008), respectively Section 2.2 of Kohl (2005) is used.

The algorithm used for Affymetrix data is similar to MAS 5.0 (cf. Affymetrix (2002)). The main difference is the substitution of the Tukey one-step estimator by our rmx k-step ( $k \geq 1$ ) estimator in the PM/MM correction step. The optional scale normalization is performed as given in Affymetrix (2002).

In case of Illumina data, the rmx estimator is used to summarize the bead types. The implementation for the most part was taken from function `createBeadSummaryData`.

For sample size  $\leq 2$ , median and MAD are used for estimation.

If `eps = 0`, mean and sd are computed.

## Value

Return value depends on the class of `x`. In case of "matrix" a matrix with columns "mean" and "sd" is returned. In case of "AffyBatch" an object of class "ExpressionSet" is returned.

## Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

## References

- Affymetrix, Inc. (2002). *Statistical Algorithms Description Document*. Affymetrix, Santa Clara.
- Kohl, M. (2005) *Numerical Contributions to the Asymptotic Theory of Robustness*. Bayreuth: Dissertation.
- Rieder, H. (1994) *Robust Asymptotic Statistics*. New York: Springer.
- Rieder, H., Kohl, M. and Ruckdeschel, P. (2008) The Costs of not Knowing the Radius. *Statistical Methods and Applications* 17(1) 13-40.
- Rieder, H., Kohl, M. and Ruckdeschel, P. (2001) The Costs of not Knowing the Radius. Submitted. Appeared as discussion paper Nr. 81. SFB 373 (Quantification and Simulation of Economic Processes), Humboldt University, Berlin; also available under [www.uni-bayreuth.de/departments/math/org/mathe7/RIEDER/pubs/RR.pdf](http://www.uni-bayreuth.de/departments/math/org/mathe7/RIEDER/pubs/RR.pdf)

**See Also**

[roblox](#), [rowRoblox](#), [AffyBatch-class](#), [generateExprVal.method.mas](#), [ExpressionSet-class](#), [createBeadSummaryData](#)

**Examples**

```

set.seed(123) # to have reproducible results for package checking

## similar to rowRoblox of package RobLox
ind <- rbinom(200, size=1, prob=0.05)
X <- matrix(rnorm(200, mean=ind*3, sd=(1-ind) + ind*9), nrow = 2)
robloxbioc(X)
robloxbioc(X, steps = 5)
robloxbioc(X, eps = 0.05)
robloxbioc(X, eps = 0.05, steps = 5)

## the function is designed for large scale problems
X <- matrix(rnorm(50000*20, mean = 1), nrow = 50000)
system.time(robloxbioc(X))

## using Affymetrix-Data
## confer example to generateExprVal.method.mas
## A more worked out example can be found in the scripts folder
## of the package.
data(SpikeIn)
probes <- pm(SpikeIn)
mas <- generateExprVal.method.mas(probes)
r1 <- 2^robloxbioc(log2(t(probes)))
concentrations <- as.numeric(colnames(SpikeIn))
plot(concentrations, mas$exprs, log="xy", ylim=c(50,10000), type="b",
      ylab = "expression measures")
points(concentrations, r1[,1], pch = 20, col="orange", type="b")
legend("topleft", c("MAS", "roblox"), pch = c(1, 20))

## Not run:
## "Not run" just because of computation time
require(affydata)
data(Dilution)
eset <- robloxbioc(Dilution)
## Affymetrix scale normalization
eset1 <- robloxbioc(Dilution, normalize = TRUE)

## End(Not run)

## using Illumina-Data
## Not run:
## "Not run" just because of computation time
data(BLData)
BSData <- robloxbioc(BLData, eps.upper = 0.5)

## End(Not run)

```

---

 SimStudies

 Perform Monte-Carlo Study.
 

---

## Description

The function `AffySimStudy` can be used to perform Monte-Carlo studies comparing Tukey's bi-weight and `rmx` estimators for normal location and scale. The function `IlluminaSimStudy` can be used to perform Monte-Carlo studies comparing Illumina's default method - a Huber-type skipped mean and sd (cf. Hampel (1985)) - and `rmx` estimators for normal location and scale. In addition, maximum likelihood (ML) estimators (mean and sd) and median and MAD are computed. The comparison is based on the empirical MSE.

## Usage

```
AffySimStudy(n, M, eps, seed = 123, eps.lower = 0, eps.upper = 0.05,
             steps = 3L, fsCor = TRUE, contD, plot1 = FALSE,
             plot2 = FALSE, plot3 = FALSE)
IlluminaSimStudy(n, M, eps, seed = 123, eps.lower = 0, eps.upper = 0.05,
                 steps = 3L, fsCor = TRUE, contD, plot1 = FALSE,
                 plot2 = FALSE, plot3 = FALSE)
```

## Arguments

<code>n</code>	integer; sample size, should be at least 3.
<code>M</code>	integer; Monte-Carlo replications.
<code>eps</code>	amount of contamination in $[0, 0.5]$ .
<code>seed</code>	random seed.
<code>eps.lower</code>	used by <code>rmx</code> estimator.
<code>eps.upper</code>	used by <code>rmx</code> estimator.
<code>steps</code>	integer; steps used for estimator construction.
<code>fsCor</code>	logical; use finite-sample correction.
<code>contD</code>	object of class "UnivariateDistribution"; contaminating distribution.
<code>plot1</code>	logical; plot cdf of ideal and real distribution.
<code>plot2</code>	logical; plot 20 (or <code>M</code> if <code>M &lt; 20</code> ) randomly selected samples.
<code>plot3</code>	logical; generate boxplots of the results.

## Details

Normal location and scale with mean = 0 and sd = 1 is used as ideal model (without restriction due to equivariance).

Since there is no estimator which yields reliable results if 50 percent or more of the observations are contaminated, we use a modification where we re-simulate all samples including at least 50 percent contaminated data.

We use function `rowRoblox` for the computation of the `rmx` estimator.

**Value**

Data.frame including empirical MSE (standardized by sample size n) and relMSE with respect to the rmx estimator.

**Author(s)**

Matthias Kohl <Matthias.Kohl@stamats.de>

**References**

Affymetrix, Inc. (2002). *Statistical Algorithms Description Document*. Affymetrix, Santa Clara.

Hampel F.R. (1985). The breakdown points of the mean combined with some rejection rules. *Technometrics*, 27(2):95-107.

**See Also**

[rowRoblox](#)

**Examples**

```
set.seed(123) # to have reproducible results for package checking
```

```
AffySimStudy(n = 11, M = 100, eps = 0.02, contD = Norm(mean = 0, sd = 3),  
             plot1 = TRUE, plot2 = TRUE, plot3 = TRUE)
```

```
IlluminaSimStudy(n = 30, M = 100, eps = 0.02, contD = Norm(mean = 0, sd = 3),  
                 plot1 = TRUE, plot2 = TRUE, plot3 = TRUE)
```

# Index

- \*Topic **package**
  - RobLoxBioC-package, 2
- \*Topic **robust**
  - KolmogorovMinDist, 3
  - robloxbioc, 5
  - SimStudies, 9
- AffyBatch-class, 8
- AffySimStudy (SimStudies), 9
- bg.correct.mas, 3, 6
- createBeadSummaryData, 7, 8
- ExpressionSet-class, 8
- finiteSampleCorrection, 6
- generateExprVal.method.mas, 8
- getArrayData, 4, 6
- IlluminaSimStudy (SimStudies), 9
- KolmogorovDist, 4
- KolmogorovMinDist, 3
- KolmogorovMinDist, AffyBatch, AbscontDistribution-method (KolmogorovMinDist), 3
- KolmogorovMinDist, BeadLevelList, AbscontDistribution-method (KolmogorovMinDist), 3
- KolmogorovMinDist, matrix, Norm-method (KolmogorovMinDist), 3
- KolmogorovMinDist-methods (KolmogorovMinDist), 3
- MDEstimator, 4
- pmcorrect.mas, 6
- roblox, 3, 8
- RobLoxBioC (RobLoxBioC-package), 2
- robloxbioc, 5
- robloxbioc, AffyBatch-method (robloxbioc), 5
- robloxbioc, BeadLevelList-method (robloxbioc), 5
- robloxbioc, matrix-method (robloxbioc), 5
- robloxbioc-methods (robloxbioc), 5
- RobLoxBioC-package, 2
- rowRoblox, 3, 8–10
- SimStudies, 9