Package 'DRomics'

September 16, 2019

Title Dose Response for Omics

Version 2.0-1

Description Several functions are provided for dose-response (or concentration-response) characterization from omics data. 'DRomics' is especially dedicated to omics data obtained using a typical dose-response design, favoring a great number of tested doses (or concentrations, at least 5, and the more the better) rather than a great number of replicates (no need of three replicates). 'DRomics' provides functions 1) to check, normalize and or transform data, 2) to select monotonic or biphasic significantly responding items (e.g. probes, metabolites), 3) to choose the best-fit model among a predefined family of monotonic and biphasic models to describe each selected item 4) to derive a benchmark dose or concentration and a typology of response from each fitted curve. In the available version data are supposed to be single-channel microarray data in log2, RNAseq data in raw counts or already pretreated metabolomic data in log scale. For further details see Larras et al (2018) <DOI:10.1021/acs.est.8b04752>.

Depends R (>= 3.4.0), limma, utils, grDevices, DESeq2

Imports stats, graphics, ggplot2

Suggests parallel, shiny, shinyBS, shinyjs, testthat

License GPL (>= 2)

Encoding UTF-8

URL `https://github.com/aursiber/DRomics`

Contact Marie-Laure Delignette-Muller

`<marielaure.delignettemuller@vetagro-sup.fr>`

NeedsCompilation no

Author Marie-Laure Delignette-Muller [aut],
Elise Billoir [aut],
Floriane Larras [ctb],
Aurelie Siberchicot [aut, cre]

Maintainer Aurelie Siberchicot `<aurelie.siberchicot@univ-lyon1.fr>`

Repository CRAN

Date/Publication 2019-09-16 10:40:08 UTC

R topics documented:

`DRomics-package` *Overview of the* **DRomics** *package*

Description

DRomics provides several functions for dose-response (or concentration-response) characterization from omics data. It is especially dedicated to omics data obtained using a typical dose-response design, favoring a great number of tested doses (or concentrations, at least 5, and the more the better) rather than a great number of replicates (no need of three replicates). DRomics provides seven main functions described as follows:

- `microarraydata`, `RNAseqdata` and `metabolomicdata` to check, normalize and transform data dependending of the type of omics data (see next paragraph for details),
- `itemselect` to select monotonic or biphasic significant responses,
- `drcfit` to choose the best-fit model among a predefined family of monotonic and biphasic models to describe each significant response and classify it in a typology of response,
- and `bmdcalc` to derive a benchmark dose or concentration from each fitted curve.

The available version supports three types of data and should not be used with other types of data:

- Single-channel microarray data (previously transformed in log2) that must imported using the function `microarraydata`,
- RNAseq (in raw counts) that must imported using the function `RNAseqdata` and
- metabolomic data that must imported using the function `metabolomicdata`. For metabolomic data, all the pretreatment steps must be done before importation of data and data should be imported in log scale so that data can be directly fitted by least-square regression (assuming a Gaussian error model valid) without any transformation.

Below is proposed an example including each step or the workflow on microarray data.

Author(s)

Marie-Laure Delignette-Muller, Elise Billoir, Floriane Larras and Aurelie Siberchicot.

See Also

See microarraydata, RNAseqdata, metabolomicdata, itemselect, drcfit, bmdcalc, bmdboot for details about each function.

Examples

```
# Step 1: importation, check and normalization of data if need
#
## here cyclicloess normalization of a small microarray data set
## (sample of a real data set)

datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")
(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
plot(o)

# Step 2: item selection using the quadratic method
#
## the quadratic method is the one we preconize to select both
## monotonic and biphasic curves from
## a typical dose-response design (with few replicates per dose)

(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.001))


# Step 3: fit of dose-response models, choice of the best fit for each curve
# and definition of the typology of response
#

(f <- drcfit(s_quad, progressbar = TRUE))
f$fitres
plot(f)

# Step 4: calculation of x-fold and z-SD benchmark doses
#

(r <- bmdcalc(f, z = 1, x = 10))
plot(r, BMDtype = "zSD", plottype = "ecdf")
plot(r, BMDtype = "xfold", plottype = "hist", by = "trend", hist.bins = 10)
plot(r, BMDtype = "xfold", plottype = "hist", by = "model", hist.bins = 10)
plot(r, BMDtype = "xfold", plottype = "hist", by = "typology", hist.bins = 10)

# Step 5: calculation of confidence intervals on the BMDs by bootstrap
#

(b <- bmdboot(r, niter = 100)) # niter to put at 1000 for a reasonable precision
plot(b, BMDtype = "zSD")
plot(b, BMDtype = "zSD", by = "trend")
b$res
```

```
# About using the DRomics-shiny app
#

if(interactive()) {
  appDir <- system.file("DRomics-shiny", package = "DRomics")
  shiny::runApp(appDir, display.mode = "normal")
}
```

---

bmdboot — *Computation of confidence interval on benchmark doses by bootstrap*

Description

Computes 95 percent confidence intervals on x-fold and z-SD benchmark doses by bootstrap.

Usage

```
bmdboot(r, items = r$res$id, niter = 1000,
                    conf.level = 0.95,
                    tol = 0.5, progressbar = TRUE,
                    parallel = c("no", "snow", "multicore"), ncpus)

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

## S3 method for class 'bmdboot'
plot(x, BMDtype = c("zSD", "xfold"), remove.infinite = TRUE,
              by = c("none", "trend", "model", "typology"), CI.col = "blue",  ...)
```

Arguments

`r`	An object of class `"bmdcalc"` returned by the function `bmdcalc`.
`items`	A character vector specifying the identifiers of the items for which you want the computation of confidence intervals. If omitted the computation is done for all the items.
`niter`	The number of samples drawn by bootstrap.
`conf.level`	Confidence level of the intervals.
`tol`	The tolerance in term of proportion of bootstrap samples on which the fit of the model is successful (if this proportion is below the tolerance, NA values are given for the limits of the confidence interval.
`progressbar`	If `TRUE` a progress bar is used to follow the bootstrap process.
`parallel`	The type of parallel operation to be used, `"snow"` or `"multicore"` (the second one not being available on Windows), or `"no"` if no parallel operation.

`ncpus`	Number of processes to be used in parallel operation : typically one would fix it to the number of available CPUs.
`x`	An object of class `"bmdboot"`.
`BMDtype`	The type of BMD to plot, `"zSD"` (default choice) or `"xfold"`.
`remove.infinite`	If TRUE the confidence intervals with non finite upper bound are not plotted.
`by`	If not at `"none"` the plot is split by the indicated factor (`"trend"`, `"model"` or `"typology"`).
`CI.col`	The color to draw the confidence intervals.
`...`	Further arguments passed to graphical or print functions.

Details

Non-parametric bootstrapping is used, where mean centered residuals are bootstrapped. For each item, bootstrapped parameter estimates are obtained by fitting the model on each of the resampled data sets. If the fitting procedure fails to converge in more than tol*100% of the cases, NA values are given for the confidence interval. Otherwise, bootstraped BMD are computed from bootstrapped parameter estimates using the same method as in `bmdcalc`. Confidence intervals on BMD are then computed using percentiles of the bootstrapped BMDs. For example 95 percent confidence intervals are computed using 2.5 and 97.5 percentiles of the bootstrapped BMDs. In cases where the bootstrapped BMD cannot be estimated as not reached at the highest tested dose or not reachable due to model asymptotes, it was given an infinite value `Inf`, so as to enable the computation of the lower limit of the BMD confidence interval if a sufficient number of bootstrapped BMD values were estimated to finite values.

Value

`bmdboot` returns an object of class `"bmdboot"`, a list with 3 components:

`res`	a data frame reporting the results of the fit, BMD computation and bootstrap on each specified item sorted in the ascending order of the adjusted p-values. The different columns correspond to the identifier of each item (`id`), the row number of this item in the initial data set (`irow`), the adjusted p-value of the selection step (`adjpvalue`), the name of the best fit model (`model`), the number of fitted parameters (`nbpar`), the values of the parameters `b`, `c`, `d`, `e` and `f`, (`NA` for non used parameters), the residual standard deviation (`SDres`), the typology of the curve (`typology`, (twelve class typology described in the help of the `drcfit` function)), the rough trend of the curve (`trend`) defined with four classes (U, bell, increasing or decreasing shape), the theoretical value at the control (`y0`), the theoretical `y` range for `x` within the range of tested doses (`yrange`) and for biphasic curves the `x` value at which their extremum is reached (`xextrem`) and the corresponding `y` value (`yextrem`), the BMD-zSD value (`BMD.zSD`) and the BMD-xfold value (`BMD.xfold`), `BMD.zSD.lower` and `BMD.zSD.upper`the lower and upper bounds of the confidence intervals of the BMD-zSD value, `BMD.xfold.lower` and `BMD.xfold.upper` the lower and upper bounds of the confidence intervals of the BMD-xfold value and `nboot.successful` the number of successful fits on bootstrapped samples for each item.

z	Value of z given in input to define the BMD-zSD.
x	Value of x given in input as a percentage to define the BMD-xfold.
tol	The tolerance given in input in term of tolerated proportion of failures of fit on bootstrapped samples.
niter	The number of samples drawn by bootstrap (given in input).

Author(s)

Marie-Laure Delignette-Muller

References

Huet S, Bouvier A, Poursat M-A, Jolivet E (2003) Statistical tools for nonlinear regression: a practical guide with S-PLUS and R examples. Springer, Berlin, Heidelberg, New York.

See Also

See bmdcalc for details about the computation of benchmark doses.

Examples

```
# (1) a toy example (a very small subsample of a microarray data set)
#
datafilename <- system.file("extdata", "transcripto_very_small_sample.txt",
package="DRomics")

# to test the package on a small but not very small data set
# use the following commented line
# datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess")
s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.001)
f <- drcfit(s_quad, progressbar = TRUE)
r <- bmdcalc(f)
set.seed(1234) # to get reproducible results with a so small number of iterations
(b <- bmdboot(r, niter = 5)) # with a non reasonable value for niter
# !!!! TO GET CORRECT RESULTS
# !!!! niter SHOULD BE FIXED FAR LARGER , e.g. to 1000
# !!!! but the run will be longer
b$res
plot(b) # plot of BMD.zSD after removing of BMDs with infinite upper bounds


plot(b, remove.infinite = FALSE) # plot of BMD.zSD without removing of BMDs
                                 # with infinite upper bounds



# bootstrap on only a subsample of items, here those best fitted by the linear model
# with a greater number of iterations
```

```
(b.lin.95 <- bmdboot(r, items = r$res$id[r$res$model == "Gauss-probit"],
                       niter = 1000, progressbar = TRUE))
b.lin.95$res

# same bootstrap but changing the default confidence level (0.95) to 0.90
(b.lin.90 <- bmdboot(r, items = r$res$id[r$res$model == "Gauss-probit"],
                       niter = 1000, conf.level = 0.9, progressbar = TRUE))
b.lin.90$res


# (2) an example on a microarray data set (a subsample of a greater data set)
#

datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.001))
(f <- drcfit(s_quad, progressbar = TRUE))
(r <- bmdcalc(f))
(b <- bmdboot(r, niter = 100)) # niter to put at 1000 for a better precision

# different plots of BMD-zSD
plot(b, by = "trend")
plot(b, by = "model")
plot(b, by = "typology")

# a plot of BMD-xfold (by default BMD-zSD is plotted)
plot(b, BMDtype = "xfold")


# (3) Comparison of parallel and non parallel implementations
#

  # to be tested with a greater number of iterations
   system.time(b1 <- bmdboot(r, niter = 100, progressbar = TRUE))
  system.time(b2 <- bmdboot(r, niter = 100, progressbar = FALSE, parallel = "snow", ncpus = 2))
```

bmdcalc — *Computation of benchmark doses for responsive items*

Description

Computes x-fold and z-SD benchmark doses for each responsive item using the best fit dose-reponse model.

Usage

```
bmdcalc(f, z = 1, x = 10)

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

## S3 method for class 'bmdcalc'
plot(x, BMDtype = c("zSD", "xfold"),
            plottype = c("ecdf", "hist", "density"),
            by = c("none", "trend", "model", "typology"),
            hist.bins = 30, ...)
```

Arguments

`f`	An object of class "drcfit" returned by the function `drcfit`.
`z`	Value of z defining the BMD-zSD as the dose at which the response is reaching y0 +/- z * SD, with y0 the level at the control given by the dose-response fitted model and SD the residual standard deviation of the dose-response fitted model.
`x`	Value of x given as a percentage and defining the BMD-xfold as the dose at which the response is reaching y0 +/- (x/100) * y0, with y0 the level at the control given by the dose-response fitted model. For `print` and `plot` functions, an object of class "bmdcalc".
`BMDtype`	The type of BMD to plot, "zSD" (default choice) or "xfold".
`plottype`	The type plot, "ecdf" for an empirical cumulative distribution plot (default choice), "hist" for a histogram or "density" for a density plot.
`by`	If different from "none" the plot is split by trend (if "trend"), by model (if "model") or by typology (if "typology").
`hist.bins`	The number of bins, only used for histogram(s).
`...`	further arguments passed to graphical or print functions.

Details

Two types of benchmark doses (BMD) were computed for each responsive item using the best fit dose-reponse model previously obtained using the `drcfit` function :

- the BMD-zSD defined as the dose at which the response is reaching y0 +/- z * SD, with y0 the level at the control given by the dose-response model, SD the residual standard deviation of the dose response model fit and z given as an input (z fixed to 1 by default),
- the BMD-xfold defined as the dose at which the response is reaching y0 +/- (x/100) * y0, with y0 the level at the control given by the dose-response fitted model and x the percentage given as an input (x fixed at 10 by default.)

When there is no analytical solution for the BMD, it is numerically searched along the fitted curve using the `uniroot` function.

In cases where the BMD cannot be reached due to the asymptote at high doses, `NaN` is returned. In cases where the BMD is not reached at the highest tested dose, `NA` is returned.

Value

bmdcalc returns an object of class "bmdcalc", a list with 4 components:

res	a data frame reporting the results of the fit and BMD computation on each selected item sorted in the ascending order of the adjusted p-values returned by function itemselect. The different columns correspond to the identifier of each item (id), the row number of this item in the initial data set (irow), the adjusted p-value of the selection step (adjpvalue), the name of the best fit model (model), the number of fitted parameters (nbpar), the values of the parameters b, c, d, e and f, (NA for non used parameters), the residual standard deviation (SDres), the typology of the curve (typology, (twelve class typology described in the help of the drcfit function)), the rough trend of the curve (trend) defined with four classes (U, bell, increasing or decreasing shape), the theoretical value at the control (y0), the theoretical y range for x within the range of tested doses (yrange) and for biphasic curves the x value at which their extremum is reached (xextrem) and the corresponding y value (yextrem), the BMD-zSD value (BMD.zSD) and the BMD-xfold value (BMD.xfold).
z	Value of z given in input to define the BMD-zSD.
x	Value of x given in input as a percentage to define the BMD-xfold.
microarraydata	The corresponding object of class "microarraydata" given in input (component of itemselect).

Author(s)

Marie-Laure Delignette-Muller and Elise Billoir

References

Larras F, Billoir E, Baillard V, Siberchicot A, Scholz S, Wubet T, Tarkka M, Schmitt-Jansen M and Delignette-Muller ML (2018). DRomics: a turnkey tool to support the use of the dose-response framework for omics data in ecological risk assessment. Environmental science & technology.https://doi.org/10.1021/acs.est.8b04752

See Also

See uniroot for details about the function used for the numerical search of the benchmark dose for cases where there is no analytical solution.

Examples

```
# (1) a toy example (a very small subsample of a microarray data set)
#
datafilename <- system.file("extdata", "transcripto_very_small_sample.txt", package="DRomics")

# to test the package on a small (for a quick calculation) but not very small data set
# use the following commented line
# datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")
```

```
(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.01))
(f <- drcfit(s_quad, progressbar = TRUE))
(r <- bmdcalc(f))
plot(r)

# changing the values of z and x for BMD calculation
(rb <- bmdcalc(f, z = 2, x = 50))
plot(rb)

# (2) an example on a microarray data set (a subsample of a greater data set)
#

datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

# to test the package on a small (for a quick calculation) but not very small data set
# use the following commented line
# datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.01))
(f <- drcfit(s_quad, progressbar = TRUE))
(r <- bmdcalc(f))
plot(r)

# different plots of BMD-zSD

plot(r, plottype = "hist")
plot(r, plottype = "density")
plot(r, plottype = "density", by = "trend")
plot(r, plottype = "ecdf", by = "trend")
plot(r, plottype = "ecdf", by = "model")
plot(r, plottype = "ecdf", by = "typology")

# a plot of BMD-xfold (by default BMD-zSD is plotted)
plot(r, BMDtype = "xfold", plottype = "hist", by = "typology", hist.bins = 10)
```

curvesplot *Plot of fitted curves*

Description

Plots fitted curves from an extended dataframe of the fitted results

Usage

```
curvesplot(extendedres, xmin = 0, xmax, y0shift = TRUE,
                          facetby, colorby, removelegend = FALSE,
                              npoints = 50, line.size = 0.2,
                              line.alpha = 1)
```

Arguments

`extendedres`	the dataframe of results provided by bmdcalc (res) or drcfit (fitres) or a subset of this data frame (selected lines). This dataframe can be extended with additional columns coming for example from the annotation of items, and some lines can be duplicated if their corresponding item has more than one annotation. This extended dataframe must at least contain the column giving the identification of each curve (`id`), the column `model` naming the fitted model and the values of the parameters (columns `b`, `c`, `d`, `e`, `f`).
`xmin`	Minimal dose/concentration for definition of the x range (by default 0).
`xmax`	Maximal dose/concentration for definition of the x range (can be defined as `max(f$omicdata$dose)` with `f` the output of `drcfit()`).
`y0shift`	If `TRUE` (default choice) curves are all shifted to have the theoretical signal at the control at 0.
`facetby`	optional argument naming the column of `extendedres` chosen to split the plot in facets (no split if omitted).
`colorby`	optional argument naming the column of `extendedres` chosen to color the curves (no color if omitted).
`removelegend`	If `TRUE` the color legend is removed (useful if the number of colors is great.
`npoints`	Number of points computed on each curve to plot it.
`line.size`	Size of the lines for plotting curves.
`line.alpha`	Transparency of the lines for plotting curves.

Details

For each item of the extended dataframe, the name of the model (column `model`) and the values of the parameters (columns `b`, `c`, `d`, `e`, `f`) are used to compute theoretical dose-response curves in the range `[xmin ; xmax]`.

Value

a ggplot object.

Author(s)

Marie-Laure Delignette-Muller

See Also

See `plot.bmdboot`.

Examples

```
# A toy example on a very small subsample of a microarray data set)
#
datafilename <- system.file("extdata", "transcripto_very_small_sample.txt",
package="DRomics")

o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess")
s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.01)
f <- drcfit(s_quad, progressbar = TRUE)

# (1)
# Default plot of all the curves
#
curvesplot(f$fitres, xmax = max(f$omicdata$dose))

# the equivalent using the output of bmdcalc
(r <- bmdcalc(f))
curvesplot(r$res, xmax = max(f$omicdata$dose))

# plot of only U-shape curves colored by model
resU <- r$res[r$res$trend == "U", ]
curvesplot(resU, xmax = max(f$omicdata$dose), colorby = "model")



# (2)
# Plot of all the curves without shifting y0 values to 0
#
curvesplot(f$fitres, xmax = max(f$omicdata$dose), y0shift = FALSE)

# (3)
# Plot of all the curves colored by model, with one facet per trend
#
curvesplot(f$fitres, xmax = max(f$omicdata$dose),
  facetby = "trend", colorby = "model")

# playing with size and transparency of lines
curvesplot(f$fitres, xmax = max(f$omicdata$dose),
  facetby = "trend", colorby = "model",
  line.size = 1, line.alpha = 0.5)

# (4) an example on a microarray data set (a subsample of a greater data set)
#
datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.001))
(f <- drcfit(s_quad, progressbar = TRUE))
(r <- bmdcalc(f))

curvesplot(f$fitres, xmax = max(f$omicdata$dose), facetby = "typology")
```

drcfit *Dose response modelling for responsive items*

Description

Fits dose reponse models to responsive items.

Usage

```
drcfit(itemselect, sigmoid.model = c("Hill", "log-probit"),
                   progressbar = TRUE, saveplot2pdf = TRUE,
                   parallel = c("no", "snow", "multicore"), ncpus)

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

## S3 method for class 'drcfit'
plot(x, items,
  plot.type = c("dose_fitted", "dose_residuals","fitted_residuals"), ...)
```

Arguments

`itemselect`	An object of class `"itemselect"` returned by the function `itemselect`.
`sigmoid.model`	The chosen sigmoid model, `"Hill"` (default choice) or `"log-probit"`.
`progressbar`	If `TRUE` a progress bar is used to follow the fitting process.
`saveplot2pdf`	If `TRUE` a pdf file named drcfitplot.pdf is saved containing all the fitted dose-response curves sorted by adjusted p-values of the selection step.
`parallel`	The type of parallel operation to be used, `"snow"` or `"multicore"` (the second one not being available on Windows), or `"no"` if no parallel operation.
`ncpus`	Number of processes to be used in parallel operation : typically one would fix it to the number of available CPUs.
`x`	An object of class `"drcfit"`.
`items`	Argument of the `plot.drcfit` function : the number of the first fits to plot (20 items max) or the character vector specifying the identifiers of the items to plot (20 items max).
`plot.type`	the type of plot, by default `"dose_fitted"` for the plot of fitted curves with the observed points added to the plot and the observed means at each dose added as black plain circles, `"dose_residuals"` for the plot of the residuals as function of the dose, and `fitted_residuals` for the plot of the residuals as function of the fitted value.
`...`	further arguments passed to graphical or print functions.

Details

For each selected item, five dose-response models (linear, Hill, exponential, Gauss-probit and log-Gauss-probit, see Larras et al. 2018 for their definition) were fitted by non linear regression, using the `nls` function. The best one was chosen as the one giving the lowest AIC value. Items with the best AIC value not lower than the AIC value of the null model (constant model) minus 2 were eliminated. Items with the best fit showing a global significant quadratic trend of the residuals as a function of the dose (in rank-scale) were also eliminated (the best fit is considered as not reliable in such cases). Each retained item is classified in a twelve class typology depending of the chosen model and of its parameter values :

- H.inc for increasing Hill curves (or lP.inc if `sigmoid.model = "log-probit"`),
- H.dec for decreasing Hill curves (or lP.dec if `sigmoid.model = "log-probit"`),
- L.inc for increasing linear curves,
- L.dec for decreasing linear curves,
- E.inc.convex for increasing convex exponential curves,
- E.dec.concave for decreasing concave exponential curves,
- E.inc.concave for increasing concave exponential curves,
- E.dec.convex for decreasing convex exponential curves,
- GP.U for U-shape Gauss-probit curves,
- GP.bell for bell-shape Gauss-probit curves,
- lGP.U for U-shape log-Gauss-probit curves,
- lGP.bell for bell-shape log-Gauss-probit curves.

Each retained item is also classified in four classes by its global trend :

- inc for increasing curves,
- dec for decreasing curves ,
- U for U-shape curves,
- bell for bell-shape curves.

Some curves fitted by a Gauss-probit model can be classified as increasing or decreasing when the dose value at which their extremum is reached is at zero.

Value

`drcfit` returns an object of class `"drcfit"`, a list with 4 components:

`fitres`	a data frame reporting the results of the fit on each selected item (one line per item) sorted in the ascending order of the adjusted p-values returned by function `itemselect`. The different columns correspond to the identifier of each item (`id`), the row number of this item in the initial data set (`irow`), the adjusted p-value of the selection step (`adjpvalue`), the name of the best fit model (`model`), the number of fitted parameters (`nbpar`), the values of the parameters `b`, `c`, `d`, `e` and `f`, (`NA` for non used parameters), the residual standard deviation (`SDres`), the typology of the curve (`typology`), the rough trend of the curve (`trend`) defined

with four classes (U, bell, increasing or decreasing shape), the theoretical value at the control `y0`), the theoretical y range for x within the range of tested doses (`yrange`), for biphasic curves the x value at which their extremum is reached (`xextrem`) and the corresponding y value (`yextrem`).

`omicdata`	The corresponding object of class `"microarraydata"` given in input (component of itemselect).
`n.failure`	The number of previously selected items on which the workflow failed to fit an acceptable model.
`AIC.val`	a data frame reporting AIC values for each selected item (one line per item) and each fitted model (one colum per model with the AIC value fixed at `Inf` when the fit failed).

Author(s)

Marie-Laure Delignette-Muller

References

Larras F, Billoir E, Baillard V, Siberchicot A, Scholz S, Wubet T, Tarkka M, Schmitt-Jansen M and Delignette-Muller ML (2018). DRomics: a turnkey tool to support the use of the dose-response framework for omics data in ecological risk assessment. Environmental science & technology.https://doi.org/10.1021/acs.est.8b04752

See Also

See `nls` for details about the non linear regression function.

Examples

```
# (1) a toy example (a very small subsample of a microarray data set)
#
datafilename <- system.file("extdata", "transcripto_very_small_sample.txt", package="DRomics")

# to test the package on a small (for a quick calculation) but not very small data set
# use the following commented line
# datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.05))
(f <- drcfit(s_quad, progressbar = TRUE))

# Default plot
plot(f)

# Plot of residuals as function of the dose
plot(f, plot.type = "dose_residuals")


# plot of residuals as function of the fitted value
```

```
plot(f, plot.type = "fitted_residuals")


# (2) an example on a microarray data set (a subsample of a greater data set)
#

datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.05))
(f <- drcfit(s_quad, progressbar = TRUE))

# Default plot
plot(f)

# Plot of the first 12 most responsive items
plot(f, items = 12)

# Plot of the chosen items in the chosen order
plot(f, items = c("301.2", "363.1", "383.1"))


# (3) Comparison of parallel and non paralell implementations on a
#     larger selection of items
#

   s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.05)
    system.time(f1 <- drcfit(s_quad, progressbar = TRUE))
   system.time(f2 <- drcfit(s_quad, progressbar = FALSE, parallel = "snow", ncpus = 2))
```

`ecdfplotwithCI` *ECDF plot of a variable with given confidence intervals on this variable*

Description

Plots the distribution of a variable as an ecdf, with x-error bars for given confidence intervals on this variable.

Usage

```
ecdfplotwithCI(variable, CI.lower, CI.upper, by, CI.col = "blue",
CI.alpha = 1, add.point = TRUE, point.size = 1, point.type = 16)
```

Arguments

`variable`	A numeric vector of the variable to plot.
`CI.lower`	A corresponding numeric vector (same length) with the lower bounds of the confidence intervals.

`CI.upper`	A corresponding numeric vector (same length) with the upper bounds of the confidence intervals.
`by`	A factor of the same length for split of the plot by this factor (no split if omitted).
`CI.col`	The color to draw the confidence intervals (unique color) of a factor coding for the color.
`CI.alpha`	Optional transparency of the lines used to draw the confidence intervals.
`add.point`	If `TRUE` points are added to confidence intervals.
`point.size`	Size of the added points in case `add.point` is `TRUE`.
`point.type`	Shape of the added points in case `add.point` is `TRUE` defined as an integer coding for a unique common shape or as a factor coding for the shape.

Details

A function to plot a variable as an ECDF plot, optionally splitted by the groups coded in the argument `by`, with the confidence intervals associated to each value of the variable, given in `CI.lower` and `CI.upper`, and optionally differentially colored according to the factor given in the argument `CI.col`.

Value

a ggplot object.

Author(s)

Marie-Laure Delignette-Muller

See Also

See `plot.bmdboot`.

Examples

```
# (1) a toy example (a very small subsample of a microarray data set)
#
datafilename <- system.file("extdata", "transcripto_very_small_sample.txt",
package="DRomics")
o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess")
s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.001)
f <- drcfit(s_quad, progressbar = TRUE)
r <- bmdcalc(f)
set.seed(1234) # to get reproducible results with a so small number of iterations
(b <- bmdboot(r, niter = 5)) # with a non reasonable value for niter
# !!!! TO GET CORRECT RESULTS
# !!!! niter SHOULD BE FIXED FAR LARGER , e.g. to 1000
# !!!! but the run will be longer
b$res
# manual ecdf plot of the bootstrap results as an ecdf distribution
# on BMD, plot that could also be obtained with plot(b)
# in this simple case
#
a <- b$res[is.finite(b$res$BMD.zSD.upper), ]
ecdfplotwithCI(variable = a$BMD.zSD, CI.lower = a$BMD.zSD.lower,
              CI.upper = a$BMD.zSD.upper, CI.col = "red")


# (2) an example on a microarray data set (a subsample of a greater data set)
#

datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.001))
(f <- drcfit(s_quad, progressbar = TRUE))
(r <- bmdcalc(f))
(b <- bmdboot(r, niter = 100)) # niter to put at 1000 for a better precision

# (2.a)
# manual ecdf plot of the bootstrap results as an ecdf distribution
# on BMD for each trend
# plot that could also be obtained with plot(b, by = "trend")
# in this simple case
#
a <- b$res[is.finite(b$res$BMD.zSD.upper), ]
ecdfplotwithCI(variable = a$BMD.zSD, CI.lower = a$BMD.zSD.lower,
              CI.upper = a$BMD.zSD.upper, by = a$trend, CI.col = "red")

# (2.b)
# ecdf plot of the bootstrap results as an ecdf distribution
# on BMD for each model
# with the color of the confidence intervals coding for the trend
#
ecdfplotwithCI(variable = a$BMD.zSD, CI.lower = a$BMD.zSD.lower,
              CI.upper = a$BMD.zSD.upper, by = a$model, CI.col = a$trend)

# changing the size of the points and the transparency of CI lines
ecdfplotwithCI(variable = a$BMD.zSD, CI.lower = a$BMD.zSD.lower,
              CI.upper = a$BMD.zSD.upper, by = a$model, CI.col = a$trend,
              CI.alpha = 0.5, point.size = 0.5)

# with the model coding for the type of points
ecdfplotwithCI(variable = a$BMD.zSD, CI.lower = a$BMD.zSD.lower,
              CI.upper = a$BMD.zSD.upper, CI.col = a$trend,
              CI.alpha = 0.5, point.size = 0.5, point.type = a$model)


# (2.c)
# ecdf plot of the bootstrap results as an ecdf distribution on
# on BMD_L (lower value of the confidence interval) for each trend
#
ecdfplotwithCI(variable = a$BMD.zSD.lower, CI.lower = a$BMD.zSD.lower,
              CI.upper = a$BMD.zSD.upper, by = a$model, CI.col = a$trend,
                add.point = FALSE)
```

`itemselect` *Selection of significantly responsive items*

Description

Significantly responsive items are selected using one of the three proposed methods: a quadratic trend test, a linear trend test or an ANOVA-based test.

Usage

```
itemselect(omicdata, select.method = c("quadratic", "linear", "ANOVA"),
  FDR = 0.05, max.ties.prop = 0.2)

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

Arguments

`omicdata`	An object of class `"microarray"`, `"RNAseq"` or `"metabolomic"` respectively returned by functions `microarraydata`, `RNAseqdata` or `metabolomicdata`.
`select.method`	The chosen method for selecting items using the `limma` package for microarray and metabolomic data and the `DESeq2` package for RNAseq data : `"quadratic"` for a quadratic trend test on dose ranks, `"linear"` for a linear trend test on dose ranks and `"ANOVA"` for an ANOVA-type test (see details for further explaination).
`FDR`	The threshold in term of FDR (False Discovery Rate) for selecting responsive items.
`max.ties.prop`	The maximal tolerated proportion of tied values for each item, above which the item cannot be selected (must be in]0, 0.5], and by default fixed at 0.2 - see details for a description of this filtering step).
`x`	An object of class `"itemselect"`.
`...`	further arguments passed to print function.

Details

The selection of responsive items is performed using the `limma` package for microarray and metabolomic data and the `DESeq2` package for RNAseq data. Three methods are proposed (as described below). Within `limma` those methods are implemented using functions `lmFit`, `eBayes` and `topTable` with p-values ajusted for multiple testing using the Benjamini-Hochberg method, with the false discovery rate given in input (argument `FDR`). Within `DESeq2` those methods are implemented using functions `DESeqDataSetFromMatrix`, `DESeq` and `results` with p-values ajusted for multiple testing using the Benjamini-Hochberg method, with the false discovery rate given in input (argument `FDR`).

- The ANOVA_based test (`"ANOVA"`) is classically used for selection of omics data in the general case but it requires many replicates per dose to be efficient, and is thus not really suited for a dose-response design.
- The linear trend test (`"linear"`) aims at detecting monotonic trends from dose-response designs, whatever the number of replicates per dose. As proposed by Tukey (1985), it tests the global significance of a linear model describing the response as a function of the dose in rank-scale.
- The quadratic trend test (`"quadratic"`) tests the global significance of a quadratic model describing the response as a function of the dose in rank-scale. It is a variant of the linear trend method that aims at detecting monotonic and non monotonic trends from a dose-response designs, whatever the number of replicates per dose (default chosen method).

A filter on the proportion of tied values is also performed whatever the type of data, assuming tied values correspond to a minimal common value at which non detections were imputed. All items having a proportion of such tied minimal values above the input argument `max.ties.prop` are not selected.

Value

`itemselect` returns an object of class `"itemselect"`, a list with 5 components:

`adjpvalue`	the vector of the p-values adjusted by the Benjamini-Hochberg method for selected items (adjpvalue inferior to FDR) sorted in ascending order
`selectindex`	the corresponding vector of row indices of selected items in the object omicdata
`omicdata`	The corresponding object of class `"microarraydata"`, `"RNAseqdata"` or `"metabolomicdata"` given in input.
`select.method`	The selection method given in input.
`FDR`	The threshold in term of FDR given in input.

The print of a `"itemselect"` object gives the number of selected items and the identifiers of the 20 most responsive items.

Author(s)

Marie-Laure Delignette-Muller

References

Tukey JW, Ciminera JL and Heyse JF (1985), *Testing the statistical certainty of a response to increasing doses of a drug*. Biometrics, 295-301.

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, and Smyth, GK (2015), *limma powers differential expression analyses for RNA-sequencing and microarray studies*. Nucleic Acids Research 43, e47.

Love MI, Huber W, and Anders S (2014), *Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2*. Genome biology, 15(12), 550.

See Also

See `lmFit`, `eBayes` and `topTable` for details about the used functions of the `limma` package and `DESeqDataSetFromMatrix`, `DESeq` and `results` for details about the used functions of the `DESeq2` package.

Examples

```
# (1) an example on a microarray data set (a subsample of a greater data set)
#
datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

(o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess"))

# 1.a using the quadratic trend test
#
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.05))

# 1.b using the linear trend test
#
(s_lin <- itemselect(o, select.method = "linear", FDR = 0.05))

# 1.c using the ANOVA-based test
#
(s_ANOVA <- itemselect(o, select.method = "ANOVA", FDR = 0.05))

# 1.d using the quadratic trend test with a smaller false discovery rate
#
(s_quad.2 <- itemselect(o, select.method = "quadratic", FDR = 0.001))
```

`metabolomicdata` *Import and check of metabolomic data*

Description

Metabolomic data are imported from a .txt file (internally imported using the function `read.table`) and checked (see the description of argument `file` for the required format of data). No normalization nor transformation is provided in this function. The pretreatment of metabolomic data must be done before importation of data, and data must be imported in log scale, so that they can be directly modelled using a Gaussian error model. This strong hypothesis is required both for selection of items and for dose-reponse modelling. A basic procedure for this pre-treatment of metabolomic data could follow the three steps described thereafter: i) removing of metabolites for which the proportion of missing data (non detections) across all the samples is too high (more than 20 to 50 percents according to your tolerance level); ii) retrieving of missing values data using half minimum method (i.e. half of the minimum value found for a metabolite across all samples); iii) log-transformation of values. If a scaling to the total intensity (normalization by sum of signals in each sample) or another normalization is necessary and pertinent, we recommend to do it before those three previously decribed steps.

Usage

```
metabolomicdata(file, check = TRUE)

## S3 method for class 'metabolomicdata'
print(x, ...)
## S3 method for class 'metabolomicdata'
plot(x, ...)
```

Arguments

`file`	The name of the .txt file (e.g. `"mydata.txt"`) containing one row per item, with the first column corresponding to the identifier of each item, in a column named `"item"`, and the other columns giving the responses of the item for each replicate at each dose or concentration. The names of the corresponding columns must correspond to the tested doses or concentrations in a numeric format for the corresponding replicate (for example, if there are triplicates for each treatment, column names can be "item", 0, 0, 0, 0.1, 0.1, 0.1, etc.
`check`	If TRUE the format of the input file is checked.
`x`	An object of class `"metabolomic"`.
`...`	further arguments passed to print or plot functions.

Details

This function imports the data from a .txt file, using the function `read.table` with its default field separator (sep argument), then checks the format of data (see the description of argument `file` for the required format of data) and gives in the `print` information that should help the user to check that the coding of data is correct : the tested doses (or concentrations) the number of replicates for each dose, the number of items, the identifiers of the first 20 items.

Value

`metabolomicdata` returns an object of class "metabolomicdata", a list with 5 components:

`data`	the numeric matrix of responses of each item in each replicate (one line per item, one column per replicate)
`dose`	the numeric vector of the tested doses or concentrations corresponding to each column of data
`item`	the character vector of the identifiers of the items, corresponding to each line of data
`design`	a table with the experimental design (tested doses and number of replicates for each dose) for control by the user
`data.mean`	the numeric matrix of mean responses of each item per dose (mean of the corresponding replicates) (one line per item, one column per unique value of the dose

The print of a `metabolomic` object gives the tested doses (or concentrations) and number of replicates for each dose, the number of items, the identifiers of the first 20 items (for check of good coding of data) and the normalization method. The plot of a `metabolomic` object shows the data distribution for each dose or concentration and replicate.

Author(s)

Marie-Laure Delignette-Muller

See Also

See `read.table` the function used to import data, and `RNAseqdata` and `microarraydata` for other types of data.

Examples

```
# (1) import and check of metabolomic data
# (an example on a subsample of a greater data set)
#
datafilename <- system.file("extdata", "metabolo_sample.txt", package="DRomics")

o <- metabolomicdata(datafilename, check = TRUE)
print(o)
plot(o)

# If you want to use your own data set just replace datafilename,
# the first argument of metabolomicdata(),
# by the name of your data file (e.g. "mydata.txt")
#
# You should take care that the field separator of this data file is one
# of the default field separators recognised by the read.table() function
# when it is used with its default field separator (sep argument)
# Tabs are recommended.
```

`microarraydata` *Import, check and normalization of single-channel microarray data*

Description

Single-channel microarray data in log2 are imported from a .txt file (internally imported using the function `read.table`), checked (see the description of argument `file` for the required format of data) and normalized (between arrays normalization). `omicdata` is a deprecated version of `microarraydata`.

Usage

```
microarraydata(file, check = TRUE,
  norm.method = c("cyclicloess", "quantile", "scale", "none"))

omicdata(file, check = TRUE,
  norm.method = c("cyclicloess", "quantile", "scale", "none"))

## S3 method for class 'microarraydata'
print(x, ...)
## S3 method for class 'microarraydata'
plot(x, ...)
```

Arguments

`file`	The name of the .txt file (e.g. `"mydata.txt"`) containing one row per item, with the first column corresponding to the identifier of each item, in a column named `"item"`, and the other columns giving the responses of the item for each replicate at each dose or concentration. The names of the corresponding columns must correspond to the tested doses or concentrations in a numeric format for the corresponding replicate (for example, if there are triplicates for each treatment, column names can be "item", 0, 0, 0, 0.1, 0.1, 0.1, etc.
`check`	If TRUE the format of the input file is checked.
`norm.method`	If `"none"` no normalization is performed, else a normalization is performed using the function normalizeBetweenArrays of the `limma` package using the specified method.
`x`	An object of class `"microarraydata"`.
`...`	further arguments passed to print or plot functions.

Details

This function imports the data from a .txt file, using the function `read.table` with its default field separator (sep argument), then checks the format of data (see the description of argument `file` for the required format of data) and gives in the `print` information that should help the user to check that the coding of data is correct : the tested doses (or concentrations) the number of replicates for each dose, the number of items, the identifiers of the first 20 items. If the argument `norm.method` is not `"none"`, data are normalized using the function `normalizeBetweenArrays` of the `limma` package using the specified method : `"cyclicloess"` (default choice), `"quantile"` or `"scale"`.

Value

`microarraydata` returns an object of class "microarraydata", a list with 7 components:

`data`	the numeric matrix of normalized responses of each item in each replicate (one line per item, one column per replicate)
`dose`	the numeric vector of the tested doses or concentrations corresponding to each column of data
`item`	the character vector of the identifiers of the items, corresponding to each line of data
`design`	a table with the experimental design (tested doses and number of replicates for each dose) for control by the user
`data.mean`	the numeric matrix of mean responses of each item per dose (mean of the corresponding replicates) (one line per item, one column per unique value of the dose

`norm.method`	The normalization method specified in input
`data.beforenorm`	the numeric matrix of responses of each item in each replicate (one line per item, one column per replicate) before normalization

The print of a `microarraydata` object gives the tested doses (or concentrations) and number of replicates for each dose, the number of items, the identifiers of the first 20 items (for check of good coding of data) and the normalization method. The plot of a `microarraydata` object shows the data distribution for each dose or concentration and replicate before and after normalization.

Author(s)

Marie-Laure Delignette-Muller

References

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, and Smyth, GK (2015), *limma powers differential expression analyses for RNA-sequencing and microarray studies*. Nucleic Acids Research 43, e47.

See Also

See `read.table` the function used to import data, `normalizeBetweenArrays` for details about the normalization and `RNAseqdata` and `metabolomicdata` for other types of data.

Examples

```
# (1) import, check and normalization of microarray data
# (an example on a subsample of a greater data set)
#
datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")
o <- microarraydata(datafilename, check = TRUE, norm.method = "cyclicloess")
print(o)
plot(o)

# If you want to use your own data set just replace datafilename,
# the first argument of microarraydata(),
# by the name of your data file (e.g. "mydata.txt")
#
# You should take care that the field separator of this data file is one
# of the default field separators recognised by the read.table() function
# when it is used with its default field separator (sep argument)
# Tabs are recommended.


# (2) normalization with other methods
(o.2 <- microarraydata(datafilename, check = TRUE, norm.method = "quantile"))
plot(o.2)
(o.3 <- microarraydata(datafilename, check = TRUE, norm.method = "scale"))
plot(o.3)
```

RNAseqdata *Import, check and normalization and transformation of RNAseq data*

Description

RNAseq data in raw counts are imported from a .txt file (internally imported using the function `read.table`), checked (see the description of argument `file` for the required format of data), normalized with respect to library size and tranformed in a log2 scale using variance stabilizing transformation or regularized logarithm.

Usage

```
RNAseqdata(file, check = TRUE, transfo.method = c("rlog", "vst"))

## S3 method for class 'RNAseqdata'
print(x, ...)
## S3 method for class 'RNAseqdata'
plot(x, ...)
```

Arguments

`file`	The name of the .txt file (e.g. `"mydata.txt"`) containing one row per item, with the first column corresponding to the identifier of each item, in a column named `"item"`, and the other columns giving the responses of the item for each replicate at each dose or concentration. The names of the corresponding columns must correspond to the tested doses or concentrations in a numeric format for the corresponding replicate (for example, if there are triplicates for each treatment, column names can be "item", 0, 0, 0, 0.1, 0.1, 0.1, etc.
`check`	If TRUE the format of the input file is checked.
`transfo.method`	The method chosen to transform raw counts in a log2 scale using the `DESeq2`: `"rlog"` for regularized logarithm or `"vst"` for variance stabilizing transformation.
`x`	An object of class `"RNAseqdata"`.
`...`	further arguments passed to print or plot functions.

Details

This function imports the data from a .txt file, using the function `read.table` with its default field separator (sep argument), then checks the format of data (see the description of argument `file` for the required format of data) and gives in the `print` information that should help the user to check that the coding of data is correct : the tested doses (or concentrations) the number of replicates for each dose, the number of items, the identifiers of the first 20 items. Data are normalized with respect to library size and tranformed using functions `rlog` or `vst` of the `DESeq2` package depending on the specified method : `"rlog"` (recommended default choice) or `"vst"`.

Value

`RNAseqdata` returns an object of class "RNAseqdata", a list with 7 components:

`data`	the numeric matrix of normalized and transformed responses of each item in each replicate (one line per item, one column per replicate)
`dose`	the numeric vector of the tested doses or concentrations corresponding to each column of data
`item`	the character vector of the identifiers of the items, corresponding to each line of data
`design`	a table with the experimental design (tested doses and number of replicates for each dose) for control by the user
`data.mean`	the numeric matrix of mean responses of each item per dose (mean of the corresponding replicates) (one line per item, one column per unique value of the dose
`transfo.method`	The transformation method specified in input
`raw.counts`	the numeric matrix of non transformed responses (raw counts) of each item in each replicate (one line per item, one column per replicate) before normalization

The print of a `RNAseqdata` object gives the tested doses (or concentrations) and number of replicates for each dose, the number of items, the identifiers of the first 20 items (for check of good coding of data) and the tranformation method. The plot of a `RNAseqdata` object shows the data distribution for each dose or concentration and replicate before and after normalization and tranformation.

Author(s)

Marie-Laure Delignette-Muller

References

Love MI, Huber W, and Anders S (2014), *Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2*. Genome biology, 15(12), 550.

See Also

See `read.table` the function used to import data, `rlog` and `vst` for details about the transformation methods and `microarraydata` and `metabolomicdata` for other types of data.

Examples

```
# (1) import, check, normalization and transformation of RNAseq data
# An example on a subsample of a data set published by Zhou et al. 2017
# (in Toxicological sciences, 160, 95-110)
# Effect on kidney transcriptomes of tetrachloroethylene
#
datafilename <- system.file("extdata", "RNAseq_sample.txt", package="DRomics")
(o <- RNAseqdata(datafilename, check = TRUE, transfo.method = "rlog"))
plot(o)
```

```
# If you want to use your own data set just replace datafilename,
# the first argument of RNAseqdata(),
# by the name of your data file (e.g. "mydata.txt")
#
# You should take care that the field separator of this data file is one
# of the default field separators recognised by the read.table() function
# when it is used with its default field separator (sep argument)
# Tabs are recommended.


# (2) transformation with two methods on the whole data set

datafilename <- system.file("extdata", "Zhou_kidney_pce.txt", package="DRomics")

# variance stabilizing tranformation
(o1 <- RNAseqdata(datafilename, check = TRUE, transfo.method = "vst"))
plot(o1)

# regularized logarithm
(o2 <- RNAseqdata(datafilename, check = TRUE, transfo.method = "vst"))
plot(o2)
```

Index