Package 'DRomics'

September 23, 2020

Title Dose Response for Omics

Version 2.1-3

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) 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. In order to link responses across biological levels based on a common method, 'DRomics' also handles apical data as long as they are continuous and follow a Gaussian distribution for each dose or concentration, with a common standard error. For further details see Larras et al (2018) <DOI:10.1021/acs.est.8b04752> at <https://hal.archives-ouvertes.fr/hal-02309919>.

Depends R (>= 3.5.0), limma, utils, grDevices, DESeq2, SummarizedExperiment

Imports stats, graphics, ggplot2

Suggests parallel, shiny, shinyBS, shinyjs, testthat

License GPL (>= 2)

Encoding UTF-8

URL https://lbbe.univ-lyon1.fr/-DRomics-.html, 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 2020-09-23 07:40:03 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 deals with transcriptomics and metabolomics data, and with apical data (as long as they are continuous and follow a Gaussian distribution for each dose or concentration, with a common standard error) in order to offer the possibility to link the responses across biological levels (from molecular to apical observations) using a common method.

DRomics provides eight functions for the main workflow:

- `microarraydata`, `RNAseqdata`, `metabolomicdata` and `continuousanchoringdata` to check the conformity of the dataset, normalize and transform data depending of the type of data (see next paragraph for details),
- `itemselect` to select monotonic and 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.
- and bmdboot to estimate the uncertainty on benchmark doses using bootstrap.

and other functions to help the biological interpretation of the workflow results and especially the analysis of results by groups, with groups defined from functional annotation:

- ecdfplotwithCI to plot BMD values with confidence intervals as an ECDF plot partitionned by groups,
- bmdplotwithgradient to plot BMD values as an ECDF plot, with a horizontal color gradient coding for the signal, partitioned by groups,
- ecdfquantileplot to plot a defined quantile of the BMD values (e.g. the median) per group,
- curvesplot to plot fitted curves of significant items splitted and/or colored by group and
- targetplot to plot dose-response raw data of target items (whether or not their response is considered significant) with fitted curves if available.

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

- Single-channel microarray data (previously transformed in log2) that must be imported using the function microarraydata,
- RNAseq (in raw counts, so integer values) that must be imported using the function RNAseqdata,
- metabolomic data that must be imported using the function metabolomicdata and
- anchoring apical data that must be imported using the function continuousanchoringdata.

  For metabolomic data, all the pretreatment steps must be done before their importation and they should be imported in log scale so that they can be directly fitted by least-square regression (assuming a Gaussian error model valid) without any transformation.

  Anchoring apical data must be continuous and imported in a scale that enables the use of least-square regression (assuming a Gaussian error model valid).

Below is proposed an example including each step or the workflow on microarray data (sample data from Larras et al. 2018) and another example on metabolomic data (results from Larras et al. 2020) to show how some functions provided in the package can be used to help to link the workflow results with the biological meaning of selected items.

Author(s)

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

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

Larras F, Billoir E, Scholz S, Tarkka M, Wubet T, Tarkka M, Delignette-Muller ML and Schmitt-Jansen M and (2020). A multi-omics concentration-response framework uncovers novel understanding of triclosan effects in the chlorophyte Scenedesmus vacuolatus. Journal of Hazardous Materials.https://doi.org/10.1016/j.jhazmat.2020.122727

See Also

See microarraydata, RNAseqdata, metabolomicdata, continuousanchoringdata, itemselect, drcfit, bmdcalc, bmdboot, bmdplotwithgradient, ecdfplotwithCI, ecdfquantileplot, curvesplot, targetplot for details about each function.

Examples

```
#### Different steps of the workflow on an example with microarray data ####

# Step 1: importation, check and normalization of data if needed
#
   # Import and check
datafilename <- system.file("extdata", "transcripto_sample.txt", package="DRomics")

   # Normalization
   # here cyclicloess normalization of a small microarray data set
   # (sample of a real data set)
(o <- microarraydata(datafilename, check = TRUE, norm.method = "cycliccloess"))
plot(o)

# Step 2: selection of significantly responding items
#
   # 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))

   # Results of the fit per item are stored in the output fitres
f$fitres

   # Default plot provides fitted curves with observed points
plot(f)

   # Plot of residuals is also recommended to validate the Gaussian error model
plot(f, plot.type = "fitted_residuals")

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

(r <- bmdcalc(f, z = 1, x = 10))

   # Distribution of benchmark doses as an ECDF plot
   # (ECDF = Empirical Cumulative Density Function)
```

```
plot(r, BMDtype = "zSD", plottype = "ecdf")

plot(r, BMDtype = "xfold", plottype = "ecdf")

   # Histogram of benchmark doses
plot(r, BMDtype = "zSD", plottype = "hist", hist.bins = 10)

   # Density plot of benchmark doses
plot(r, BMDtype = "zSD", plottype = "density")

   # Plot of the distribution of benchmark doses grouped by reponse trend
plot(r, BMDtype = "zSD", plottype = "ecdf", by = "trend")

   # Same plot with the add of a color gradient for each item coding for
   # the intensity of the signal (after shift of the control signal at 0)
   # as a function of the dose

bmdplotwithgradient(r$res, BMDtype = "zSD",
                       facetby = "trend", shapeby = "model")

# 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

   # Results of the fit per item and confidence intervals are stored in the output res
b$res

      # Distribution of benchmark doses as an ECDF plot
      # with confidence intervals on each BMD values
plot(b, BMDtype = "zSD")

   # Same plot with BMD grouped by rsponse trend
plot(b, BMDtype = "zSD", by = "trend")



# About using the DRomics-shiny app for this workflow
#

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


#### Help for biological interpretation of DRomics outputs ####
#### based on an example with metabolomic data             ####


# 1. Import the dataframe with metabolomic results to use:
# the output $res of bmdcalc() or bmdboot() functions)
# from step 4 or 5 of the main DRomics workflow
```

```
# This step will not be necessary if previous steps are done directly
# in R using the DRomics package as described previously
# but in this example we did it to take a real example
# that took a long time to run but from which results are stored in the package

   # code to import the file for this example in our package
metabresfilename <- system.file("extdata", "triclosanSVmetabres.txt", package="DRomics")
metabres <- read.table(metabresfilename, header = TRUE, stringsAsFactors = TRUE)

   # to see the structure of this file
str(metabres)

# 2. Import the dataframe with functional annotation (or any other
# descriptor/category you want to use, here KEGG metabolic pathway)
# of each item present in the 'res' file.
# Be cautious, this file must be produced by the user.
# Each item may have more than one annotation (-> more than one line)

   # code to import the file for this example in our package
metabannotfilename <- system.file("extdata", "triclosanSVmetabannot.txt",
                            package="DRomics")
metabannot <- read.table(metabannotfilename, header = TRUE, stringsAsFactors = TRUE)

   # to see the structure of this file
str(metabannot)

# 3. Merging of both previous dataframes
# in order to obtain a so-called 'extenderes' dataframe
# gathering for each item metrics derived from the DRomics workflow and
# functional annotation

metabextendedres <- merge(x = metabres, y = metabannot, by.x = "id", by.y = "metab.code")
str(metabextendedres)


# 4. Graphical representations provided in the package

   # BMDplot with gradient splitted here by metabolic pathway
bmdplotwithgradient(metabextendedres, BMDtype = "zSD",
                     facetby = "path_class", shapeby = "trend")

   # the same representation with labels of items (so without shapeby)
bmdplotwithgradient(metabextendedres, BMDtype = "zSD", facetby = "path_class",
                      add.label = TRUE)

   # an ECDFplot of BMD_zSD with confidence intervals splitted
   # here by metabolic pathway
   # with color coding for dose-response trend
ecdfplotwithCI(variable = metabextendedres$BMD.zSD,
               CI.lower = metabextendedres$BMD.zSD.lower,
               CI.upper = metabextendedres$BMD.zSD.upper,
               by = metabextendedres$path_class,
```

```
               CI.col = metabextendedres$trend)


   # an ECDFplot of quantiles of BMD-zSD calculated
   # here by metabolic pathway
ecdfquantileplot(variable = metabextendedres$BMD.zSD,
               by = metabextendedres$path_class,
               quantile.prob = 0.25)

   # Plot of the dose-response curves for a specific metabolic pathway
   # in this example the "lipid metabolism" pathclass
LMmetabextendedres <- metabextendedres[metabextendedres$path_class == "Lipid metabolism", ]
curvesplot(LMmetabextendedres, facetby = "id", npoints = 100, line.size = 1,
           colorby = "trend",
           xmin = 0, xmax = 8)

#### Help for multiomics approach                                  ####
#### an example with metabolomics and transcriptomics              ####
#### data for Scenedesmus and triclosan (cf. Larras et al. 2020 ####

# 5. following the same steps described before for metabolomics
# import and merge the results for microarray data

contigresfilename <- system.file("extdata", "triclosanSVcontigres.txt", package="DRomics")
contigres <- read.table(contigresfilename, header = TRUE, stringsAsFactors = TRUE)
str(contigres)

contigannotfilename <- system.file("extdata", "triclosanSVcontigannot.txt",
                              package="DRomics")
contigannot <- read.table(contigannotfilename, header = TRUE, stringsAsFactors = TRUE)
str(contigannot)

contigextendedres <- merge(x = contigres, y = contigannot, by.x = "id", by.y = "contig")
str(contigextendedres)


# 6. Comparison of results obtained at both molecular levels (metabolites and contigs)

   # Binding of dataframes at both levels adding a variable coding
   # for the level
extendedres <- rbind(metabextendedres, contigextendedres)
extendedres$level <- factor(c(rep("metabolites", nrow(metabextendedres)),
                              rep("contigs", nrow(contigextendedres))))
str(extendedres)

   # Frequencies of pathways by molecular levels
(t.pathways <- table(extendedres$path_class, extendedres$level))
original.par <- par()
par(las = 2, mar = c(4,13,1,1))
barplot(t(t.pathways), beside = TRUE, horiz = TRUE,
      cex.names = 0.7, legend.text = TRUE,
      main = "Frequencies of pathways")
```

```
   # Proportions of pathways by molecular levels
(t.prop.pathways <- prop.table(t.pathways, margin = 2))
barplot(t(t.prop.pathways), beside = TRUE, horiz = TRUE,
      cex.names = 0.7, legend.text = TRUE,
      main = "Proportion of pathways")
par(original.par)

if (require(ggplot2))
{
      # an ECDF plot of BMD_zSD by pathways
      # using color
   ggplot(extendedres, aes(x = BMD.zSD, color = level)) +
      stat_ecdf(geom = "step") + ylab("ECDF")

      # or using facets
   ggplot(extendedres, aes(x = BMD.zSD)) + facet_wrap(~ level) +
      stat_ecdf(geom = "step") + ylab("ECDF")

}

   # an ECDF plot of BMD_zSD with confidence intervals splitted
   # here by metabolic pathway
   # with color coding for molecular level
ecdfplotwithCI(variable = extendedres$BMD.zSD,
               CI.lower = extendedres$BMD.zSD.lower,
               CI.upper = extendedres$BMD.zSD.upper,
               by = extendedres$path_class,
               CI.col = extendedres$level) + labs(col = "Molecular level")

   # an ECDFplot of BMD_zSD with confidence intervals splitted
   # here by molecular level
ecdfplotwithCI(variable = extendedres$BMD.zSD,
               CI.lower = extendedres$BMD.zSD.lower,
               CI.upper = extendedres$BMD.zSD.upper,
               by = extendedres$level)


   # Plot of the dose-response curves for a specific metabolic pathway
   # in this example the "lipid metabolism" pathclass
LMres <- extendedres[extendedres$path_class == "Lipid metabolism", ]
curvesplot(LMres, facetby = "level", free.y.scales = TRUE, npoints = 100, line.size = 1,
           colorby = "trend",
           xmin = 0, xmax = 8) + labs(col = "DR_trend")
```

---

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.upperthe 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)
if (require(ggplot2)) plot(b) + scale_x_log10() # in BMD log10 scale
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 = "cycliccloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.01))
(f <- drcfit(s_quad, progressbar = TRUE))
(r <- bmdcalc(f))
plot(r)
if (require(ggplot2))
  plot(r, plottype = "ecdf") + scale_x_log10() # with log10 dose scale

# 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)
```

`bmdplotwithgradient` *BMD plot with color gradient*

Description

Provides an ECDF plot of BMD values with a horizontal color gradient coding, for each item, for the theoretical signal as a function of the dose (concentration). The idea is to display the amplitude and the intensity of the response of each item on the BMD ECDF plot, in addition to the BMD ordered values. This plot is of interest especially when not too much items are presented. To maximize the lisibility of the plot, one can manually pre-select items based on its own criteria (e.g. functional group of interest).

Usage

```
bmdplotwithgradient(extendedres, BMDtype = c("zSD", "xfold"),
                                  xmin, xmax, y0shift = TRUE,
                                  facetby, shapeby, npoints = 50,
                                  line.size, point.size = 1,
                                  ncol4faceting, limits4colgradient,
                                 lowercol = "darkgreen", uppercol = "darkred",
                                  add.label, label.size = 2,
                                  BMD_log_transfo = FALSE)
```

Arguments

extendedres	the dataframe of results provided by bmdcalc (res) or a subset of this data frame (selected lines). This dataframe can be extended with additional columns coming for example from the functional annotation of items, and some lines can be replicated if their corresponding item has more than one annotation. This extended dataframe must at least contain the column giving the BMD values (BMD.zSD or BMD.xfold depending of chosen BMDtype), identification of each curve (id), the column model naming the fitted model and the values of the parameters (columns b, c, d, e, f).
BMDtype	The type of BMD to plot, "zSD" (default choice) or "xfold".
xmin	Optional minimal dose/concentration for definition of the x range.
xmax	Optional maximal dose/concentration for definition of the x range (can be defined as max(f$omicdata$dose) with f the output of drcfit() for example).
y0shift	If TRUE (default choice) for each item the signal is 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).
shapeby	optional argument naming the column of extendedres chosen to shape the BMD points (no difference if shapeby if omitted).
npoints	Number of points computed on each curve in order to define the signal color gradient (= number of doses or concentrations for which the theoretical signal is computed from the fitted model for each item).
line.size	Size of the horizontal lines for plotting each signal color gradient.
point.size	Size of the BMD points.
ncol4faceting	Number of columns for facetting.
limits4colgradient	Optional vector giving minimal and maximal value of the signal for the color gradient.
lowercol	Chosen color for the lower values of the signal.
uppercol	Chosen color for the upper values of the signal.
add.label	Points are replaced by labels of items if TRUE.
label.size	Size of labels if add.label is TRUE.
BMD_log_transfo	If TRUE a log transformation of the BMD is used in the plot. This option cannot be used with a null value of xmin in input.

Details

BMD values are plotted as an ECDF plot, as with plot.bmdcalc using "ecdf" as plottype. In addition is plotted an horizontal color gradient for each item coding for the signal level at each dose (or concentration). The optional use of columns to code for shape and/or facets for each item is particularly intended to give a view of all the dose-response per group (e.g. metabolic pathways). Those groups must be coded in a column of extendedres. In case where one item is allocated to more

than one group during the annotation process, the line of this item must be replicated in extendedres as many times as the number of annotation groups in which it was allocated.

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, and so the corresponding signal color gradient, in the range `[xmin ; xmax]`.

Value

a ggplot object.

Author(s)

Marie-Laure Delignette-Muller

See Also

See `plot.bmdcalc` and `plot.bmdboot`.

Examples

```
# (1)
# 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)
r <- bmdcalc(f)

# Plot of all the BMD values with color dose-response gradient
#
bmdplotwithgradient(r$res, BMDtype = "zSD")



# Plot of all the BMD values with color dose-response gradient
# with definition of xmax from the maximal tested dose
#
bmdplotwithgradient(r$res, BMDtype = "zSD",
                    xmax = max(f$omicdata$dose))

# Add of item labels
bmdplotwithgradient(r$res, BMDtype = "zSD",
                    xmax = max(f$omicdata$dose), add.label = TRUE)

# The same plot in log scale (we have to define xmin in this case)
#
bmdplotwithgradient(r$res, BMDtype = "zSD",
                    BMD_log_transfo = TRUE)
```

```
# The same plot in log scale with defining xmin and xmax at a chosen values
#
bmdplotwithgradient(r$res, BMDtype = "zSD",
                    xmin =  min(f$omicdata$dose[f$omicdata$dose != 0] / 2),
                    xmax = max(f$omicdata$dose),
                    BMD_log_transfo = TRUE)

# Plot of all the BMD values with color dose-response gradient
# faceted by response trend and shaped by model
#
bmdplotwithgradient(r$res, BMDtype = "zSD",
                      facetby = "trend", shapeby = "model")

# (2)
# Plot of BMD values with color dose-response gradient
# faceted by metabolic pathway (from annotation of the selected items)
# and shaped by dose-response trend

# An example from the paper published by Larras et al. 2020
# in Journal of Hazardous Materials
# https://doi.org/10.1016/j.jhazmat.2020.122727
# A example of plot obtained with this function is in Figure 5 in Larras et al. 2020


# the dataframe with metabolomic results (output $res of bmdcalc() or bmdboot() functions)
resfilename <- system.file("extdata", "triclosanSVmetabres.txt", package="DRomics")
res <- read.table(resfilename, header = TRUE, stringsAsFactors = TRUE)
str(res)

# the dataframe with annotation of each item identified in the previous file
# each item may have more than one annotation (-> more than one line)
annotfilename <- system.file("extdata", "triclosanSVmetabannot.txt", package="DRomics")
annot <- read.table(annotfilename, header = TRUE, stringsAsFactors = TRUE)
str(annot)

# Merging of both previous dataframes
# in order to obtain an extenderes dataframe
extendedres <- merge(x = res, y = annot, by.x = "id", by.y = "metab.code")
head(extendedres)


### (2.a) BMDplot with gradient by pathway
bmdplotwithgradient(extendedres, BMDtype = "zSD",
                    facetby = "path_class",
                      shapeby = "trend")

# (2.b) The same example forcing the limits of the colour gradient at other
# values than observed minimal and maximal values of the signal
bmdplotwithgradient(extendedres, BMDtype = "zSD",
                      facetby = "path_class",
                      shapeby = "trend",
                      limits4colgradient = c(-1, 1))
```

```
# (2.c) The same example changing the gradient colors and the line size
bmdplotwithgradient(extendedres, BMDtype = "zSD",
                      facetby = "path_class",
                       shapeby = "trend",
                       line.size = 3,
                       lowercol = "darkblue", uppercol = "orange")

# (2.d) The same example with only lipid metabolism pathclass
# and identification of the metabolites
LMres <- extendedres[extendedres$path_class == "Lipid metabolism", ]
bmdplotwithgradient(LMres, BMDtype = "zSD",
                       line.size = 3,
                       add.label = TRUE, label.size = 3)


# (3)
# 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 = "cycliclоess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.001))
(f <- drcfit(s_quad, progressbar = TRUE))
(r <- bmdcalc(f))

bmdplotwithgradient(r$res, BMDtype = "zSD",
                      facetby = "trend",
                      shapeby = "model")
bmdplotwithgradient(r$res, BMDtype = "zSD",
                      xmax = max(f$omicdata$dose), facetby = "trend",
                      shapeby = "model")
```

`continuousanchoringdata` *Import and check of continuous anchoring apical data*

Description

Continuous anchoring apical data are imported from a .txt file (internally imported using the function `read.table`) and checked or from a R object of class `data.frame` (see the description of argument `file` for the required format of data). No transformation is provided in this function. If needed the pretreatment of data must be done before importation of data, so that they can be directly modelled using a Gaussian error model. This strong hypothesis is required both for selection of responsive endpoints and for dose-reponse modelling.

Usage

```
continuousanchoringdata(file, check = TRUE)

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

Arguments

file	The name of the .txt file (e.g. "mydata.txt") containing one row per endpoint, with the first column corresponding to the identifier of each endpoint, and the other columns giving the measured values of the endpoint for each replicate at each dose or concentration. In the first line, after a name for the endpoint column, we must have the tested doses or concentrations in a numeric format for the corresponding replicate (for example, if there are triplicates for each treatment, the first line could be "endpoint", 0, 0, 0, 0.1, 0.1, 0.1, etc.). This file is imported within the function using the function read.table with its default field separator (sep argument). Alternatively an R object of class data.frame can be directly given in input, corresponding to the output of read.table(file,header = FALSE) on a file described as above. The two alternatives are illustrated below in examples.
check	If TRUE the format of the input file is checked.
x	An object of class "continuousanchoringdata".
...	further arguments passed to print or plot functions.

Details

This function imports the data, checks their format (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 endpoints.

Value

continuousanchoringdata returns an object of class "continuousanchoringdata", 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 endpoints, 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 `continuousanchoringdata` 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 `continuousanchoringdata` 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 `microarraydata`, `RNAseqdata` and `microarraydata` for other types of data.

Examples

```
# (1) import and check of continuous anchoring data
# (an example with two apical endpoints of an example given in the package (see ?Scenedesmus))
#
datafilename <- system.file("extdata", "apical_anchoring.txt", package = "DRomics")

o <- continuousanchoringdata(datafilename, check = TRUE)
print(o)

plot(o)

# If you want to use your own data set just replace datafilename,
# the first argument of continuousanchoringdata(),
# 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.

# Use of an R object of class data.frame
# on the same example (see ?Scenedesmus for details)
data(Scenedesmus_apical)
o <- continuousanchoringdata(Scenedesmus_apical)
print(o)

plot(o)
```

curvesplot *Plot of fitted curves*

Description

Provides a plot of all the fitted curves from a dataframe of the main workflow results, possibly extended with additional information (e.g. groups from functional annotation) used to color and/or split the curves.

Usage

```
curvesplot(extendedres, xmin = 0, xmax, y0shift = TRUE,
                  facetby, free.y.scales = FALSE, colorby, removelegend = FALSE,
                        npoints = 500, line.size = 0.2,
                        line.alpha = 1, dose_log_transfo = FALSE)
```

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 replicated 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).
free.y.scales	if TRUE the y scales are free in the different facets.
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.
dose_log_transfo	If TRUE a log transformation of the dose is used in the plot. This option needs a definition of a strictly positive value of xmin in input.

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 same plot with dose in log scale (need x != 0 in input)
curvesplot(f$fitres, xmin = 0.1, xmax = max(f$omicdata$dose),
  dose_log_transfo = TRUE)

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

# plot of curves colored by models
curvesplot(r$res, xmax = max(f$omicdata$dose), colorby = "model")

# plot of curves facetted by trends
curvesplot(r$res, xmax = max(f$omicdata$dose), facetby = "trend")

# the same plot with free y scales
curvesplot(r$res, xmax = max(f$omicdata$dose), facetby = "trend",
```

```
  free.y.scales =  TRUE)

# (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")


# (5) An example from data published by Larras et al. 2020
# in Journal of Hazardous Materials
# https://doi.org/10.1016/j.jhazmat.2020.122727

# a dataframe with metabolomic results (output $res of bmdcalc() or bmdboot() functions)
resfilename <- system.file("extdata", "triclosanSVmetabres.txt", package="DRomics")
res <- read.table(resfilename, header = TRUE, stringsAsFactors = TRUE)
str(res)

# a dataframe with annotation of each item identified in the previous file
# each item may have more than one annotation (-> more than one line)
annotfilename <- system.file("extdata", "triclosanSVmetabannot.txt", package="DRomics")
annot <- read.table(annotfilename, header = TRUE, stringsAsFactors = TRUE)
str(annot)

# Merging of both previous dataframes
# in order to obtain an extenderes dataframe
# bootstrap results and annotation
extendedres <- merge(x = res, y = annot, by.x = "id", by.y = "metab.code")
head(extendedres)

# Plot of the dose-response curves for a specific pathway
# in this example the "lipid metabolism" pathclass
```

```
LMres <- extendedres[extendedres$path_class == "Lipid metabolism", ]
curvesplot(LMres, facetby = "id", npoints = 100, line.size = 1,
           colorby = "trend",
           xmin = 0, xmax = 8)
```

`drcfit` *Dose response modelling for responsive items*

Description

Fits dose reponse models to responsive items.

Usage

```
drcfit(itemselect, sigmoid.model = c("Hill", "log-probit"),
                  information.criterion = c("AIC", "BIC"),
                   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"),
   dose_log_transfo = FALSE, ...)
```

Arguments

`itemselect`	An object of class `"itemselect"` returned by the function `itemselect`.
`sigmoid.model`	The chosen sigmoid model, `"Hill"` (default choice) or `"log-probit"`.
`information.criterion`	The information criterion used to select the best fit model, `"AIC"` (default choice) or `"BIC"`.
`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.
dose_log_transfo	to put at TRUE to use a log transformation for the dose axis (only available if the dose is in x-axis, so not available for plot.type "fitted_residuals").
...	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 (or BIC) value. Items with the best AIC (or BIC) value not lower than the AIC (or BIC) 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 for which a successful fit is reached (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).

information.criterion — The information criterion used to select the best fit model as given in input.

information.criterion.val — a data frame reporting AIC (or BIC) values for each selected item (one line per item) and each fitted model (one colum per model with the AIC (or BIC) value fixed at Inf when the fit failed).

n.failure — The number of previously selected items on which the workflow failed to fit an acceptable model.

unfitres — A data frame reporting the results on each selected item for which no successful fit is reached (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), and code for the reason of the fitting failure (cause, equal to "constant.model" if the best fit model is a constant model or "trend.in.residuals" if the best fit model is rejected due to quadratic trend on residuals.)

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 and targetplot to plot target items (even if non responsive or unfitted).

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 = "cycliccloess")
s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.05)
f <- drcfit(s_quad, progressbar = TRUE)

# Default plot
plot(f)


# The same plot with log transformation of the doses
plot(f, dose_log_transfo = TRUE)

# The same plot in x log scale choosing x limits for plot
if (require(ggplot2))
plot(f, dose_log_transfo = TRUE) +
        scale_x_log10(limits = c(0.1, 10))

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

# Same plot of residuals with log transformation of the doses
plot(f, plot.type = "dose_residuals", dose_log_transfo = TRUE)

# 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 = "cycliccloess"))
(s_quad <- itemselect(o, select.method = "quadratic", FDR = 0.05))
(f <- drcfit(s_quad, progressbar = TRUE))

# Default plot
plot(f)
```

```
# Plot of the fit 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"))

# Look at the table of results for successful fits
head(f$fitres)

# Look at the table of results for unsuccessful fits
head(f$unfitres)


# (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

Provides an ECDF plot of a variable, with x-error bars for given confidence intervals on this variable, possibly partitioned by groups. In the context of this package this function is intended to be used with the BMD as the variable and with groups defined by the user from functional annotation.

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. In the context of the package this variable may be a BMD.
`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). In the context of this package this factor may code for groups defined by the user from functional annotation.
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.

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(1) # 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

# 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 from data published by Larras et al. 2020
# in Journal of Hazardous Materials
# https://doi.org/10.1016/j.jhazmat.2020.122727

# This function can also be used to go deeper in the exploration of the biological
# meaning of the responses. Here is an example linking the DRomics outputs
# with the functional annotation of the responding metabolites of the microalgae
# Scenedesmus vacuolatus to the biocide triclosan.
# This extra step uses a dataframe previously built by the user which links the items
# to the biological information of interest (e.g. KEGG pathways).

# importation of a dataframe with metabolomic results
# (output $res of bmdcalc() or bmdboot() functions)
resfilename <- system.file("extdata", "triclosanSVmetabres.txt", package="DRomics")
res <- read.table(resfilename, header = TRUE, stringsAsFactors = TRUE)
str(res)

# importation of a dataframe with annotation of each item
# identified in the previous file (this dataframe must be previously built by the user)
# each item may have more than one annotation (-> more than one line)
annotfilename <- system.file("extdata", "triclosanSVmetabannot.txt", package="DRomics")
annot <- read.table(annotfilename, header = TRUE, stringsAsFactors = TRUE)
str(annot)

# Merging of both previous dataframes
# in order to obtain an extenderes dataframe
# bootstrap results and annotation
annotres <- merge(x = res, y = annot, by.x = "id", by.y = "metab.code")
head(annotres)

### an ECDFplot with confidence intervals by pathway
# with color coding for dose-response trend
ecdfplotwithCI(variable = annotres$BMD.zSD,
               CI.lower = annotres$BMD.zSD.lower,
               CI.upper = annotres$BMD.zSD.upper,
               by = annotres$path_class,
               CI.col = annotres$trend)


# (3) 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 = "cycliccloess"))
(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

# (3.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")

# (3.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)


# (3.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)
```

`ecdfquantileplot` *ECDF plot of a given quantile of a variable calculated by group*

Description

Plots a given quantile of a variable calculated by group as an ECDF plot with points sized by the numbers of items per group. In the context of this package this function is intended to be used with the BMD as the variable and with groups defined by the user from functional annotation.

Usage

```
ecdfquantileplot(variable, by, quantile.prob = 0.5, title)
```

Arguments

variable	A numeric vector corresponding to the variable on which we want to calculate the given quantile by group. In the context of the package this variable may be a BMD.
by	A factor of the same length defining the groups. In the context of this package this factor may code for groups defined by the user from functional annotation.
quantile.prob	The probability (in]0, 1[) defining the quantile to calculate on each group.
title	An optional title for the plot.

Details

The given quantile is calculated for each group (e.g.from all items of a metabolic pathway) using function quantile and plotted as an ECDF plot. In this ECDF plot of quantiles each point is sized according to the number of items in the corresponding group (e.g. metabolic pathway).

Value

a ggplot object.

Author(s)

Marie-Laure Delignette-Muller

See Also

See quantile.

Examples

```
# (1) An example from data published by Larras et al. 2020
# in Journal of Hazardous Materials
# https://doi.org/10.1016/j.jhazmat.2020.122727

# a dataframe with metabolomic results (output $res of bmdcalc() or bmdboot() functions)
resfilename <- system.file("extdata", "triclosanSVmetabres.txt", package="DRomics")
res <- read.table(resfilename, header = TRUE, stringsAsFactors = TRUE)
str(res)

# a dataframe with annotation of each item identified in the previous file
# each item may have more than one annotation (-> more than one line)
annotfilename <- system.file("extdata", "triclosanSVmetabannot.txt", package="DRomics")
annot <- read.table(annotfilename, header = TRUE, stringsAsFactors = TRUE)
str(annot)

# Merging of both previous dataframes
# in order to obtain an extenderes dataframe
# bootstrap results and annotation
annotres <- merge(x = res, y = annot, by.x = "id", by.y = "metab.code")
head(annotres)
```

```
### an ECDFplot of quantiles of BMD-zSD calculated by pathway
ecdfquantileplot(variable = annotres$BMD.zSD,
                 by = annotres$path_class,
                 quantile.prob = 0.25)
```

`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 `"microarraydata"`, `"RNAseqdata"`, `"metabolomicdata"` or `"continuousanchoringdata"` respectively returned by functions `microarraydata`, `RNAseqdata`, `metabolomicdata` or `continuousanchoringdata`.
`select.method`	`"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, the `DESeq2` package for RNAseq data and the `lm` function for continuous anchoring 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). For continuous anchoring data, the lm and anova functions are used to fit the model and compare it to the null model, and the pvalues are then corrected using the function p.adjust with the Benjamini-Hochberg method.

- 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", "metabolomicdata" or "continuousanchoringdata" 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 = "cycliccloess"))

# 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 or from a R object of class `data.frame` (see the description of argument `file` for the

required format of data). No normalization nor transformation is provided in this function. The pre-treatment 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, and the other columns giving the responses of the item for each replicate at each dose or concentration. In the first line, after a name for the identifier column, we must have the tested doses or concentrations in a numeric format for the corresponding replicate (for example, if there are triplicates for each treatment, the first line could be "item", 0, 0, 0, 0.1, 0.1, 0.1, etc.). This file is imported within the function using the function read.table with its default field separator (sep argument). Alternatively an R object of class data.frame can be directly given in input, corresponding to the output of read.table(file,header = FALSE) on a file described as above. The two alternatives are illustrated below in examples.
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, checks their format (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 and 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 given in the package (see ?Scenedesmus))
#
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.

# Use of an R object of class data.frame
# An example using the complete data set
# Scenedesmus_metab (see ?Scenedesmus for details)
data(Scenedesmus_metab)
```

```
(o <- metabolomicdata(Scenedesmus_metab))
plot(o)
```

`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 or from a R object of class `data.frame` (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, and the other columns giving the responses of the item for each replicate at each dose or concentration. In the first line, after a name for the identifier column, we must have the tested doses or concentrations in a numeric format for the corresponding replicate (for example, if there are triplicates for each treatment, the first line could be "item", 0, 0, 0, 0.1, 0.1, 0.1, etc.). This file is imported within the function using the function `read.table` with its default field separator (sep argument). Alternatively an R object of class `data.frame` can be directly given in input, corresponding to the output of `read.table(file,header = FALSE)` on a file described as above.
`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, checks their format (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 published in Larras et al. 2018
# Transcriptomic effect of triclosan in the chlorophyte Scenedesmus vacuolatus)
#
datafilename <- system.file("extdata", "transcripto_very_small_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 (integer values) are imported from a .txt file (internally imported using the function `read.table`), checked or from a R object of class `data.frame` (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"),
          transfo.blind = TRUE, round.counts = FALSE)

## 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, and the other columns giving the responses of the item for each replicate at each dose or concentration. In the first line, after a name for the identifier column, we must have the tested doses or concentrations in a numeric format for the corresponding replicate (for example, if there are triplicates for each treatment, the first line could be "item", 0, 0, 0, 0.1, 0.1, 0.1, etc.). This file is imported within the function using the function read.table with its default field separator (sep argument). Alternatively an R object of class data.frame can be directly given in input, corresponding to the output of read.table(file,header = FALSE) on a file described as above. The two alternatives are illustrated below in examples.
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.
transfo.blind	Argument given to function rlog or vst, see rlog and vst for an explaination, by default at TRUE as in the DESeq2 package .
round.counts	Put it to TRUE if your counts come from Kallisto or Salmon in order to round them before treatment with DESeq2.
x	An object of class "RNAseqdata".
...	further arguments passed to print or plot functions.

Details

This function imports the data, checks their format (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
# Effect on mouse kidney transcriptomes of tetrachloroethylene
# (see ? Zhou for details)
#
datafilename <- system.file("extdata", "RNAseq_sample.txt", package="DRomics")
(o <- RNAseqdata(datafilename, check = TRUE, transfo.method = "vst"))
plot(o, range = 1e6)

# 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.

# Use of an R object of class data.frame
# below the same example taking a subsample of the data set
# Zhou_kidney_pce (see ?Zhou for details)
data(Zhou_kidney_pce)
subsample <- Zhou_kidney_pce[1:1000, ]
```

```
(o <- RNAseqdata(subsample, check = TRUE, transfo.method = "vst"))
plot(o, range = 1e6)


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

data(Zhou_kidney_pce)

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

# regularized logarithm
(o2 <- RNAseqdata(Zhou_kidney_pce, check = TRUE, transfo.method = "rlog"))
plot(o2)

# variance stabilizing tranformation (blind to the experimental design)
(o3 <- RNAseqdata(Zhou_kidney_pce, check = TRUE, transfo.method = "vst",
      transfo.blind = TRUE))
plot(o3)

# regularized logarithm
(o4 <- RNAseqdata(Zhou_kidney_pce, check = TRUE, transfo.method = "rlog",
      transfo.blind = TRUE))
plot(o4)
```

`Scenedesmus` *Concentration-response effect of triclosan in Scenedesmus vacuolatus*

Description

Metabolomic and apical data sets for the effect of triclosan in the chlorophyte Scenedesmus vacuolatus.

Usage

```
data(Scenedesmus_metab)
data(Scenedesmus_apical)
```

Format

`Scenedesmus_metab` contains one row per metabolite, with the first column corresponding to the identifier of each metabolite, and the other columns giving the log10 tranformed area under the curve for each replicate at each concentration. In the first line, after the name for the identifier column, we have the tested concentrations for each corresponding replicate.

Scenedesmus_apical contains one row per apical endpoint, with the first column corresponding to the identifier of each endpoint, and the other columns giving the measured value of this each endpoint for each replicate at each concentration. In the first line, after the name for the identifier column, we have the tested concentrations for each corresponding replicate.

Source

Larras, F., Billoir, E., Scholz, S., Tarkka, M., Wubet, T., Delignette-Muller, M. L., & Schmitt-Jansen, M. (2020). A multi-omics concentration-response framework uncovers novel understanding of triclosan effects in the chlorophyte Scenedesmus vacuolatus. *Journal of Hazardous Materials*, 122727.

Examples

```
# (1.1) load of metabolomics data
#
data(Scenedesmus_metab)
head(Scenedesmus_metab)
str(Scenedesmus_metab)



# (1.2) import and check of metabolomics data
#
(o_metab <- metabolomicdata(Scenedesmus_metab))
plot(o_metab)

# (2.1) load of apical data
#
data(Scenedesmus_apical)
head(Scenedesmus_apical)
str(Scenedesmus_apical)

# (2.2) import and check of apical data
#
(o_apical <- continuousanchoringdata(Scenedesmus_apical))
plot(o_apical)

# (2.3) selection of responsive endpoints on apical data
#
(s_apical <- itemselect(o_apical, select.method = "quadratic", FDR = 0.05))

# (2.4) fit of dose-response models on apical data
#
(f_apical <- drcfit(s_apical, progressbar = TRUE))
f_apical$fitres
plot(f_apical)
plot(f_apical, dose_log_trans = TRUE)
plot(f_apical, plot.type = "dose_residuals")

# (2.5) Benchmark dose calculation on apical data
#
```

```
r_apical <- bmdcalc(f_apical, z = 1)
r_apical$res
```

`targetplot` *Dose-reponse plot for target items*

Description

Plots dose-response raw data of target items (whether or not their response is considered significant) with fitted curves if available.

Usage

```
targetplot(items, f, add.fit = TRUE, dose_log_transfo = FALSE)
```

Arguments

`items`	A character vector specifying the identifiers of the items to plot.
`f`	An object of class `"drcfit"`.
`add.fit`	If `TRUE` the fitted curve is added for items which were selected as responsive items and for which a best fit model was obtained.
`dose_log_transfo`	to put at `TRUE` to use a log transformation for the dose axis.

Value

a ggplot object.

Author(s)

Marie-Laure Delignette-Muller

See Also

See `plot.drcfit`.

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)

# Plot of chosen items with fitted curves when available
#
targetitems <- c("88.1", "1", "3", "15")
targetplot(targetitems, f = f)



# The same plot in x log scale
#
targetplot(targetitems, f = f, dose_log_transfo = TRUE)

# The same plot in x log scale choosing x limits for plot
if (require(ggplot2))
targetplot(targetitems, f = f, dose_log_transfo = TRUE) +
        scale_x_log10(limits = c(0.1, 10))

# The same plot without fitted curves
#
targetplot(targetitems, f = f, add.fit = FALSE)
```

Zhou *Dose-response liver and kidney transcriptomic effect of Trichloroethylene end Tetrachloroethylene in mouse*

Description

RNAseq data sets for the effect of Trichloroethylene (TCE) and Tetrachloroethylene (PCE) on mouse liver and kidney. Each of those environmental contaminants was administered by gavage in aqueous vehicle to male B6C3F1/J mice, within a dose-reponse design including five doses plus the control.

Usage

```
data(Zhou_kidney_pce)
data(Zhou_kidney_tce)
data(Zhou_liver_pce)
data(Zhou_liver_tce)
```

Format

`Zhou_kidney_pce`, `Zhou_kidney_tce`, `Zhou_liver_pce` nd `Zhou_liver_tce` each contains one row per transcript, with the first column corresponding to the identifier of each transcript, and the other columns giving the count of reads for each replicate at each dose. In the first line, after the name for the identifier column, we have the tested doses for each corresponding replicate.

Source

Zhou, Y. H., Cichocki, J. A., Soldatow, V. Y., Scholl, E. H., Gallins, P. J., Jima, D., ... & Rusyn, I. 2017. Comparative dose-response analysis of liver and kidney transcriptomic effects of trichloroethylene and tetrachloroethylene in B6C3F1 mouse. *Toxicological sciences*, **160(1)**, 95-110.

Examples

```
# (1) load of data
#
data(Zhou_kidney_pce)
head(Zhou_kidney_pce)
str(Zhou_kidney_pce)

data(Zhou_kidney_tce)
head(Zhou_kidney_tce)

data(Zhou_liver_pce)
head(Zhou_liver_pce)

data(Zhou_liver_tce)
head(Zhou_liver_tce)



# (2) import, check, normalization and transformation of a sample
# of one of those datasets
#
d <- Zhou_kidney_pce[1:501, ]
(o <- RNAseqdata(d))
plot(o)


# (3) analysis of the whole dataset (for kidney and PCE)
# (may be long to run)

d <- Zhou_kidney_pce
(o <- RNAseqdata(d))
plot(o)
(s <- itemselect(o, select.method = "quadratic", FDR = 0.01))
(f <- drcfit(s, progressbar = TRUE))
head(f$fitres)

plot(f)
plot(f, dose_log_trans = TRUE)
plot(f, plot.type = "dose_residuals")

r <- bmdcalc(f, z = 1)
plot(r)
if (require(ggplot2))
  plot(r) + scale_x_log10() # same plot in log scale of BMD
plot(r, by = "trend")
head(r$res)
```

Index