Package ‘growthPheno’

January 13, 2025

Version 3.1.10

Date 2025-01-13

Title Functional Analysis of Phenotypic Growth Data to Smooth and Extract Traits

Depends R (>= 3.5.0)

Imports dae, GGally, ggplot2, grDevices, Hmisc, JOPS, methods, RColorBrewer, readxl, reshape, stats, stringi, utils

Suggests testthat, nlme, R.rsp, scales

VignetteBuilder R.rsp

Description Assists in the plotting and functional smoothing of traits measured over time and the extraction of features from these traits, implementing the SET (Smoothing and Extraction of Traits) method described in Brien et al. (2020) Plant Methods, 16. Smoothing of growth trends for individual plants using natural cubic smoothing splines or P-splines is available for removing transient effects and segmented smoothing is available to deal with discontinuities in growth trends. There are graphical tools for assessing the adequacy of trait smoothing, both when using this and other packages, such as those that fit nonlinear growth models. A range of per-unit (plant, pot, plot) growth traits or features can be extracted from the data, including single time points, interval growth rates and other growth statistics, such as maximum growth or days to maximum growth. The package also has tools adapted to inputting data from high-throughput phenotyping facilities, such from a Lemna-Tec Scananalyzer 3D (see <`https://www.youtube.com/watch?v=MRAF_mAEa7E/`> for more information). The package 'growthPheno' can also be installed from <`http://chris.brien.name/rpackages/`>.

License GPL (>= 2)

URL `http://chris.brien.name/`

BugReports `https://github.com/briencj/growthPheno/issues`

RoxygenNote 5.0.1

NeedsCompilation no

Author Chris Brien [aut, cre] (<`https://orcid.org/0000-0003-0581-1817`>)

Maintainer Chris Brien <`chris.brien@adelaide.edu.au`>

Repository CRAN

Date/Publication 2025-01-13 03:40:03 UTC

Contents

`growthPheno-package` *Functional Analysis of Phenotypic Growth Data to Smooth and Extract Traits*

Description

Assists in the plotting and functional smoothing of traits measured over time and the extraction of features from these traits, implementing the SET (Smoothing and Extraction of Traits) method described in Brien et al. (2020) Plant Methods, 16. Smoothing of growth trends for individual plants using natural cubic smoothing splines or P-splines is available for removing transient effects and segmented smoothing is available to deal with discontinuities in growth trends. There are graphical tools for assessing the adequacy of trait smoothing, both when using this and other packages, such as those that fit nonlinear growth models. A range of per-unit (plant, pot, plot) growth traits or features can be extracted from the data, including single time points, interval growth rates and other growth statistics, such as maximum growth or days to maximum growth. The package also has tools adapted to inputting data from high-throughput phenotyping facilities, such from a Lemna-Tec Scananalyzer 3D (see <https://www.youtube.com/watch?v=MRAF_mAEa7E/> for more information). The package 'growthPheno' can also be installed from <http://chris.brien.name/rpackages/>.

Version: 3.1.10

Date: 2025-01-13

Index

The following list of functions does not include those that are soft-deprecated, i.e. those that have been available in previous versions of `growthPheno` but will be removed in future versions. For a description of the use of the listed functions and vignettes that are available, see the Overview section below.

(i) Wrapper functions

`traitSmooth`	Obtain smooths for a trait by fitting spline

	functions and, having compared several smooths, allows one of them to be chosen and returned in a `data.frame`.
`traitExtractFeatures`	Extract features, that are single-valued for each individual, from smoothed traits over time.

(ii) Helper functions

`args4chosen_plot`	Creates a `list` of the values for the options of profile plots for the chosen smooth.
`args4chosen_smooth`	Creates a `list` of the values for the smoothing parameters for which a smooth is to be extracted.
`args4meddevn_plot`	Creates a `list` of the values for the options of median deviations plots for smooths.
`args4profile_plot`	Creates a `list` of the values for the options of profile plots for comparing smooths.
`args4smoothing`	Creates a `list` of the values for the smoothing parameters to be passed to a smoothing function.

(iii) Data

`exampleData`	A small data set to use in function examples.
`RicePrepped.dat`	Prepped data from an experiment to investigate a rice germplasm panel.
`RiceRaw.dat`	Data for an experiment to investigate a rice germplasm panel.
`tomato.dat`	Longitudinal data for an experiment to investigate tomato response to mycorrhizal fungi and zinc.

(iv) Plots

`plotAnom`	Identifies anomalous individuals and produces profile plots without them and with just them.
`plotCorrmatrix`	Calculates and plots correlation matrices for a set of responses.
`plotDeviationsBoxes`	Produces boxplots of the deviations of the observed values from the smoothed values over values of x.
`plotImagetimes`	Plots the time within an interval versus the interval. For example, the hour of the day carts are imaged against the days after planting (or some other number of days after an event).
`plotProfiles`	Produces profile plots of longitudinal data for a set of individuals.
`plotSmoothsComparison`	Plots several sets of smoothed values for a response, possibly along with growth rates and optionally including the unsmoothed values, as well as deviations boxplots.
`plotSmoothsMedianDevns`	Calculates and plots the medians of the deviations from the observed values of several sets for smoothed

	values stored in a `data.frame` in long format.
`probeSmooths`	Computes and compares, for a set of smoothing parameters, a response and the smooths of it, possibly along with growth rates calculated from the smooths.

(v) Smoothing and calculation of growth rates and water use traits for each individual (Indv)

`byIndv4Intvl_GRsAvg`	Calculates the growth rates for a specified time interval for individuals in a `data.frame` in long format by taking weighted averages of growth rates for times within the interval.
`byIndv4Intvl_GRsDiff`	Calculates the growth rates for a specified time interval for individuals in a `data.frame` in long format by differencing the values for a response within the interval.
`byIndv4Intvl_ValueCalc`	Calculates a single value that is a `function` of the values of an individual for a response in a `data.frame` in long format over a specified time interval.
`byIndv4Intvl_WaterUse`	Calculates, water use traits (WU, WUR, WUI) over a specified time interval for each individual in a `data.frame` in long format.
`byIndv4Times_GRsDiff`	Adds, to a 'data.frame', the growth rates calculated for consecutive times for individuals in a `data.frame` in long format by differencing response values.
`byIndv4Times_SplinesGRs`	For a response in a `data.frame` in long format, computes, for a single set of smoothing parameters, smooths of the response, possibly along with growth rates calculated from the smooths.
`byIndv_ValueCalc`	Applies a `function` to calculate a single value from an individual's values for a response in a `data.frame` in long format.
`smoothSpline`	Fit a spline to smooth the relationship between a `response` and an `x` in a `data.frame`, optionally computing growth rates using derivatives.
`probeSmooths`	For a response in a `data.frame` in long format, computes and compares, for sets of smoothing parameters, smooths of the response, possibly along with growth rates calculated from the smooths.

(vi) Data frame manipulation

`as.smooths.frame`	Forms a `smooths.frame` from a `data.frame`, ensuring that the correct columns are present.
`designFactors`	Adds the factors and covariates for a blocked,

	split-unit design.
getTimesSubset	Forms a subset of 'responses' in 'data' that contains their values for the nominated times.
importExcel	Imports an Excel imaging file and allows some renaming of variables.
is.smooths.frame	Tests whether an object is of class smooths.frame.
prepImageData	Selects a set variables to be retained in a data frame of longitudinal data.
smooths.frame	Description of a smooths.frame object,
twoLevelOpcreate	Creates a data.frame formed by applying, for each response, a binary operation to the values of two different treatments.
validSmoothsFrame	Checks that an object is a valid smooths.frame.

(vii) General calculations

anom	Tests if any values in a vector are anomalous in being outside specified limits.
calcLagged	Replaces the values in a vector with the result of applying an operation to it and a lagged value.
calcTimes	Calculates for a set of times, the time intervals after an origin time and the position of each within a time interval
cumulate	Calculates the cumulative sum, ignoring the first element if exclude.1st is TRUE.
GrowthRates	Calculates growth rates (AGR, PGR, RGRdiff) between a pair of values in a vector.
WUI	Calculates the Water Use Index (WUI) for a value of the response and of the water use.

(viii) Principal variates analysis (PVA)

intervalPVA.data.frame	Selects a subset of variables using PVA, based on the observed values within a specified time interval
PVA.data.frame	Selects a subset of variables stored in a data.frame using PVA.
PVA.matrix	Selects a subset of variables using PVA based on a correlation matrix.
rcontrib.data.frame	Computes a measure of how correlated each variable in a set is with the other variable, conditional on a nominated subset of them.
rcontrib.matrix	Computes a measure of how correlated each variable in a set is with the other variable, conditional on a nominated subset of them.

Overview

This package can be used to perform a functional analysis of growth data using splines to smooth the trend of individual plant traces over time and then to extract features or tertiarty traits for further analysis. This process is called smoothing and extraction of traits (SET) by Brien et al. (2020), who detail the use of `growthPheno` for carrying out the method. However, `growthPheno` now has the two wrapper, or primary, functions `traitSmooth` and `traitExtractFeatures` that implement the SET approach. These may be the only functions that are used in that the complete SET process can be carried out using only them. The `Tomato` vignette illustrates their use for the example presented in Brien et al. (2020).

The function `traitSmooth` utilizes the secondary functions `probeSmooths`, `plotSmoothsComparison` and `plotSmoothsMedianDevns` and accepts the arguments of the secondary functions. The function `probeSmooths` utilizes the tertiary functions `byIndv4Times_SplinesGRs` and `byIndv4Times_GRsDiff`, which in turn call the function `smoothSpline`. The function `plotSmoothsComparison` calls `plotDeviationsBoxes`. All of these functions play a role in choosing the smoothing method and parameters for a data set.

The primary function `traitExtractFeatures` uses the secondary functions `getTimesSubset` and the set of `byIndv4Intvl_` functions. These functions are concerned with the extraction of traits that yield a single value for each individual in the data.

Recourse to the secondary and terriary functions may be necessary for special cases. Their use is illustrated in the `Rice` vignette.

Use `vignette("Tomato", package = "growthPheno")` or `vignette("Rice", package = "growthPheno")` to access either of the vignettes.

In addition to functions that implement SET approach, `growthPheno` also has functions for importing and organizing the data that are generally applicable, although they do have defaults that make them particularly adapted to data from a high-throughput phenotyping facility based on a Lemna-Tec Scananalyzer 3D system.

Data suitable for use with this package consists of columns of data obtained from a set of individuals (e.g. plants, pots, carts, plots or units) over time. There should be a unique identifier for each individual and a time variable, such as Days after Planting (DAP), that contain no repeats for an individual. The combination of the identifier and a time for an individual should be unique to that individual. For imaging data, the individuals may be arranged in a grid of Lanes × Positions. That is, the minimum set of columns is an individuals, a times and one or more primary trait columns.

Author(s)

Chris Brien [aut, cre] (<https://orcid.org/0000-0003-0581-1817>)

Maintainer: Chris Brien <chris.brien@adelaide.edu.au>

References

Brien, C., Jewell, N., Garnett, T., Watts-Williams, S. J., & Berger, B. (2020). Smoothing and extraction of traits in the growth analysis of noninvasive phenotypic data. *Plant Methods*, **16**, 36. doi:10.1186/s13007020005776.

See Also

`dae`

`anom` *Tests if any values in a vector are anomalous in being outside specified limits*

Description

Test whether any values in `x` are less than the value of `lower`, if it is not `NULL`, or are greater than the value of `upper`, if it is not `NULL`, or both.

Usage

```
anom(x, lower=NULL, upper=NULL, na.rm = TRUE)
```

Arguments

`x`	A `vector` containing the values to be tested.
`lower`	A `numeric` such that values in `x` below it are considered to be anomalous.
`upper`	A `numeric` such that values in `x` above it are considered to be anomalous.
`na.rm`	A `logical` indicating whether `NA` values should be stripped before the testing proceeds.

Value

A `logical` indicating whether any values have been found to be outside the limits specified by `lower` or `upper` or both.

Author(s)

Chris Brien

Examples

```
data(exampleData)
anom.val <- anom(longi.dat$sPSA.AGR, lower=2.5)
```

`args4chosen_plot` *Creates a* `list` *of the values for the options of profile plots for the chosen smooth*

Description

Creates a `list` of the values for the options of profile plots (and boxplots facets) for comparing smooths. Note that `plots.by`, `facet.x`, `facet.y` and `include.raw` jointly define the organization of the plots. The default settings are optimized for `traitSmooth`.

Usage

```
args4chosen_plot(plots.by = NULL,
                 facet.x = ".", facet.y = ".",
                 include.raw = "no",
                 collapse.facets.x = FALSE, collapse.facets.y = FALSE,
                 facet.labeller = NULL, facet.scales = "fixed",
                 breaks.spacing.x = -2, angle.x = 0,
                 colour = "black", colour.column = NULL,
                 colour.values = NULL, alpha = 0.3,
                 addMediansWhiskers = TRUE,
                 ggplotFuncs = NULL,
                 ...)
```

Arguments

`plots.by`	A `character` that gives the names of the set of `factor`s by which the data is to be grouped and a separate plot produced for each group. If `NULL`, no groups are formed. If a set of `factor`s, such as `Type`, `Tuning` and `Method`, that uniquely index the combinations of the smoothing-parameter values is specified, then groups are formed for each combination of the levels of the these `factor`s, and a separate plot is produced for each combination.
`facet.x`	A `character` giving the names of the `factor`s to be used to form subsets to be plotted in separate columns of the profiles plots. The default of `"."` results in no split into columns.
`facet.y`	A `character` giving the `factor`s to be used to form subsets to be plotted in separate rows of the profiles plots. The default of `"."` results in no split into rows.
`include.raw`	A `character` indicating whether plots of the raw (unsmoothed) trait, corresponding to the plots of the smoothed traits, are to be included in profile plots. The options are `no`, `alone`, `facet.x`, or `facet.y`. That is, the plots of the raw traits are plotted separately or as part of either `facet.x` or `facet.y`.
`collapse.facets.x`	A `logical` to indicate whether all variables specified by `facets.x` are to be collapsed to a single variable. Note that the smoothing-parameters factors, if present, are always collapsed.
`collapse.facets.y`	A `logical` to indicate whether all variables specified by `facets.y` are to be collapsed to a single variable. Note that the smoothing-parameters factors, if present, are always collapsed.
`facet.labeller`	A `ggplot` `function` for labelling the facets of a plot produced using the `ggplot` function. For more information see `ggplot`.
`facet.scales`	A `character` specifying whether the scales are shared across all facets of a plot ("fixed"), or do they vary across rows (the default, "free_x"), columns ("free_y"), or both rows and columns ("free")?
`breaks.spacing.x`	A `numeric` whose absolute values specifies the distance between major breaks for the x-axis in a sequence beginning with the minimum x value and continuing

	up to the maximum x value. If it is negative, the breaks that do not have x values in `data` will be omitted. Minor breaks will be at half major break value or, if these do not correspond to x-values in `data` when `breaks.spacing.x` is negative, have a spacing of one. Thus, when `breaks.spacing.x` is negative, grid lines will only be included for x-values that occur in `data`. These settings can be overwritten by supplying, in `ggplotFuncs`, a `scale_x_continuous` function from `ggplot2`.
`angle.x`	A `numeric` between 0 and 360 that gives the angle of the x-axis text to the x-axis. It can also be set by supplying, in `ggplotFuncs`, a `theme` function from `ggplot2`.
`colour`	A `character` specifying a single colour to use in drawing the lines for the profiles. If colouring according to the values of a variable is required then use `colour.column`.
`colour.column`	A `character` giving the name of a column in `data` over whose values the colours of the lines are to be varied. The colours can be specified using `colour.values`.
`colour.values`	A `character` vector specifying the values of the colours to use in drawing the lines for the profiles. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order (usually alphabetical) within the limits of the scale.
`alpha`	A `numeric` specifying the degrees of transparency to be used in plotting the responses. It is a ratio in which the denominator specifies the number of points (or lines) that must be overplotted to give a solid cover.
`addMediansWhiskers`	A `logical` indicating whether plots over time of the medians and outer whiskers are to be added to the plot. The outer whiskers are related to the whiskers on a box-and-whisker and are defined as the median plus (and minus) 1.5 times the interquartile range (IQR). Points lying outside the whiskers are considered to be potential outliers.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object for a profiles plot.
`...`	allows arguments to be passed to other functions; not used at present.

Value

A named `list`.

Author(s)

Chris Brien

See Also

`traitSmooth`, `probeSmooths`, `plotSmoothsComparison` and `args4profile_plot`.

Examples

```
args4chosen_plot(plots.by = "Type",
                 facet.x = "Tuning", facet.y = c("Smarthouse", "Treatment.1"),
                 include.raw = "facet.x",
                 alpha  = 0.4,
                 colour.column = "Method",
                 colour.values = c("orange", "olivedrab"))
```

`args4chosen_smooth` *Creates a `list` of the values for the smoothing parameters for which a smooth is to be extracted*

Description

Creates a `list` of the values for the smoothing parameters for which a smooth is to be extracted. The default settings for these are optimized for `traitSmooth`.

Usage

```
args4chosen_smooth(smoothing.methods = "logarithmic",
                   spline.types = "PS",
                   df = NULL,
                   lambdas = NULL,
                   combinations = "single")
```

Arguments

`smoothing.methods`	A `character` giving the smoothing method for the chosen smooth. The two possibilites are (i) `"direct"`, for directly smoothing the observed `response`, and (ii) `"logarithmic"`, for smoothing the `log`-transformed `response`.
`spline.types`	A `character` giving the type of spline for the chosen smooth. Currently, the possibilites are (i) `"NCSS"`, for natural cubic smoothing splines, and (ii) `"PS"`, for P-splines.
`df`	A `numeric` with single value that specifies, for natural cubic smoothing splines (`NCSS`), the desired equivalent numbers of degrees of freedom of the chosen smooth (trace of the smoother matrix). Lower values result in more smoothing.
`lambdas`	A named `list` or a `numeric` specifying the positive penalties for which the chosen smooth is required.
`combinations`	Generally, this argument should be set to `single` so that ony one value should be supplied to the functions arguments. Also, only one of `df` or `lambdas` should be set.

Value

A named `list`.

Author(s)

Chris Brien

See Also

`traitSmooth` and `probeSmooths`.

Examples

```
args4chosen_smooth(smoothing.methods = "direct",
                   spline.types = "NCSS", df = 4)
args4chosen_smooth(smoothing.methods = "log",
                   spline.types = "PS", lambdas = 0.36)
```

`args4devnboxes_plot` *Creates a* `list` *of the values for the options of profile plots for comparing smooths*

Description

Creates a `list` of the values for the options of deviations boxplots for comparing smooths. Note that `plots.by`, `facet.x` and `facet.y` jointly define the organization of the plots. The default settings are optimized for `traitSmooth` so that, if you want to change any of these from their default settings when using `args4devnboxes_plot` with a function other than `traitSmooth`, then it is recommended that you specify all of them to ensure that the complete set has been correctly specified. Otherwise, the default settings will be those shown here and these may be different to the default settings shown for the function with which you are using `args4devnboxes_plot`.

Usage

```
args4devnboxes_plot(plots.by = "Type",
                      facet.x = c("Method", "Tuning"), facet.y = ".",
                      collapse.facets.x = TRUE, collapse.facets.y = FALSE,
                      facet.labeller = NULL, facet.scales = "fixed",
                      angle.x = 0,
                      which.plots = "none",
                      ggplotFuncs = NULL,
                      ...)
```

Arguments

`plots.by`	A `character` that gives the names of the set of `factor`s by which the data is to be grouped and a separate plot produced for each group. If `NULL`, no groups are formed. If a set of `factor`s, such as `Type`, `Tuning` and `Method`, that uniquely index the combinations of the smoothing-parameter values is specified, then groups are formed for each combination of the levels of the these `factor`s, and a separate plot is produced for each combination.

`facet.x`	A `character` giving the names of the `factor`s to be used to form subsets to be plotted in separate columns of the profiles plots and deviations boxplots. The default of `"."` results in no split into columns.
`facet.y`	A `character` giving the `factor`s to be used to form subsets to be plotted in separate rows of the profiles plots and deviations boxplots. The default of `"."` results in no split into rows.
`collapse.facets.x`	A `logical` to indicate whether all variables specified by `facets.x` are to be collapsed to a single variable. Note that the smoothing-parameters factors, if present, are always collapsed.
`collapse.facets.y`	A `logical` to indicate whether all variables specified by `facets.y` are to be collapsed to a single variable. Note that the smoothing-parameters factors, if present, are always collapsed.
`facet.labeller`	A `ggplot` `function` for labelling the facets of a plot produced using the `ggplot` function. For more information see `ggplot`.
`facet.scales`	A `character` specifying whether the scales are shared across all facets of a plot ("fixed"), or do they vary across rows (the default, "free_x"), columns ("free_y"), or both rows and columns ("free")?
`angle.x`	A `numeric` between 0 and 360 that gives the angle of the x-axis text to the x-axis. It can also be set by supplying, in `ggplotFuncs`, a `theme` function from `ggplot2`.
`which.plots`	A `logical` indicating which plots are to be produced. The options are either `none` or `absolute.deviations` and/or `relative.deviations`. Boxplots of the absolute deviations are specified by `absolute.boxplots`, the absolute deviations being the values of a trait minus their smoothed values (observed - smoothed). Boxplots of the relative deviations are specified by `relative.boxplots`, the relative deviations being the absolute deviations divided by the smoothed values of the trait.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object for a boxplot.
`...`	allows arguments to be passed to other functions; not used at present.

Value

A named `list`.

Author(s)

Chris Brien

See Also

`traitSmooth`, `probeSmooths`, `plotSmoothsComparison` and `args4chosen_plot`.

Examples

```
args4devnboxes_plot(plots.by = "Type",
                    facet.x = "Tuning",
                    facet.y = c("Smarthouse", "Treatment.1"),
                    which.plots = "absolute")
```

`args4meddevn_plot` *Creates a `list` of the values for the options of median deviations plots for smooths*

Description

Creates a `list` of the values for the options of median deviations plots for smooths. Note that the arguments `plots.by`, `plots.group`, `facet.x` and `facet.y` jointly define the organization of the plots. The default settings are optimized for `traitSmooth` so that, if you want to change any of these from their default settings when using `args4meddevn_plot` with a function other than `traitSmooth`, then it is recommended that you specify all of them to ensure that the complete set has been correctly specified. Otherwise, the default settings will be those shown here and these may be different to the default settings shown for the function with which you are using `args4meddevn_plot`.

Usage

```
args4meddevn_plot(plots.by = NULL, plots.group = "Tuning",
                  facet.x = c("Method","Type"), facet.y = ".",
                  facet.labeller = NULL, facet.scales = "free_x",
                  breaks.spacing.x = -4, angle.x = 0,
                  colour.values = NULL, shape.values = NULL,
                  alpha = 0.5,
                  propn.note = TRUE, propn.types = NULL,
                  ggplotFuncs = NULL,
                  ...)
```

Arguments

`plots.by`	A `character` that give the names of the set of `factor`s by which medians deviations data is to be grouped and a separate plot produced for each group. If `NULL`, no groups are formed. If a set of `factor`s, such as `Type`, `Tuning` and `Method`, that uniquely index the combinations of the smoothing-parameter values is specified, then groups are formed for each combination of the levels of the these `factor`s, and a separate plot is produced for each combination.
`plots.group`	A `character` that gives the names of the set of `factor`s by which the subset of medians deviations data within a single facet in a single plot is to be grouped for plotting as separate lines.
`facet.x`	A `character` giving the `factor`s to be used to form subsets to be plotted in separate columns of the medians deviations plots. The default of `"."` results in no split into columns.

`facet.y`	A `character` giving the `factor`s to be used to form subsets to be plotted in separate rows of the medians deviations plots. The default of `"."` results in no split into rows.
`facet.labeller`	A `ggplot` `function` for labelling the facets of a plot produced using the `ggplot` function. For more information see `ggplot`.
`facet.scales`	A `character` specifying whether the scales are shared across all facets of a plot ("fixed"), or do they vary across rows (the default, "free_x"), columns ("free_y"), or both rows and columns ("free")?
`breaks.spacing.x`	A `numeric` whose absolute values specifies the distance between breaks for the x-axis in a sequence beginning with the minimum x value and continuing up to the maximum x value. If it is negative, the breaks that do not have x values in `data` will be omitted. Minor breaks will be at half this value or, if these do not correspond to x-values in `data` when `breaks.spacing.x` is negative, have a spacing of one. Thus, when `breaks.spacing.x` is negative, grid lines will only be included for x-values that occur in `data`. These settings can be overwritten by supplying, in `ggplotFuncs`, a `scale_x_continuous` function from `ggplot2`.
`angle.x`	A `numeric` between 0 and 360 that gives the angle of the x-axis text to the x-axis. It can also be set by supplying, in `ggplotFuncs`, a `theme` function from `ggplot2`.
`colour.values`	A `character` vector specifying the values of the colours to use in drawing the lines for the medians deviations within a facet. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order (usually alphabetical) within the limits of the scale.
`shape.values`	A `numeric` vector specifying the values of the shapes to use in drawing the points for the medians deviations within a facet. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order.
`alpha`	A `numeric` specifying the degrees of transparency to be used in plotting a median deviations plot. It is a ratio in which the denominator specifies the number of points (or lines) that must be overplotted to give a solid cover.
`propn.note`	A `logical` indicating whether a note giving the proportion of the median value of the `response` for each time is to be included in the `medians.deviations` plots.
`propn.types`	A `numeric` giving, for each of the `trait.types`, the proportion of the median value of the `response` for each time to be used to plot envelopes in the median deviations plots. If set to `NULL`, the plots of the proprotion envelopes are omitted.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object for a median-deviations plot.
`...`	allows arguments to be passed to other functions; not used at present.

Value

A named `list`.

Author(s)

Chris Brien

See Also

`traitSmooth`, `probeSmooths` and `plotSmoothsMedianDevns`.

Examples

```
args4meddevn_plot(plots.by = "Type", plots.group = "Tuning",
                  facet.x = "Method", facet.y = ".",
                  propn.types = c(0.02,0.1, 0.2))
```

`args4profile_plot` *Creates a `list` of the values for the options of profile plots for comparing smooths*

Description

Creates a `list` of the values for the options of profile plots for comparing smooths. Note that `plots.by`, `facet.x`, `facet.y` and `include.raw` jointly define the organization of the plots. The default settings are optimized for `traitSmooth` so that, if you want to change any of these from their default settings when using `args4profile_plot` with a function other than `traitSmooth`, then it is recommended that you specify all of them to ensure that the complete set has been correctly specified. Otherwise, the default settings will be those shown here and these may be different to the default settings shown for the function with which you are using `args4profile_plot`.

Usage

```
args4profile_plot(plots.by = "Type",
                  facet.x = c("Method", "Tuning"), facet.y = ".",
                  include.raw = "facet.x",
                  collapse.facets.x = TRUE, collapse.facets.y = FALSE,
                  facet.labeller = NULL, facet.scales = "fixed",
                  breaks.spacing.x = -4, angle.x = 0,
                  colour = "black", colour.column = NULL,
                  colour.values = NULL, alpha = 0.3,
                  addMediansWhiskers = TRUE,
                  ggplotFuncs = NULL,
                  ...)
```

Arguments

`plots.by` A `character` that gives the names of the set of `factor`s by which the data is to be grouped and a separate plot produced for each group. If `NULL`, no groups are formed. If a set of `factor`s, such as `Type`, `Tuning` and `Method`, that uniquely index the combinations of the smoothing-parameter values is specified, then

	groups are formed for each combination of the levels of the these `factor`s, and a separate plot is produced for each combination.
`facet.x`	A `character` giving the names of the `factor`s to be used to form subsets to be plotted in separate columns of the profiles plots and deviations boxplots. The default of `"."` results in no split into columns.
`facet.y`	A `character` giving the `factor`s to be used to form subsets to be plotted in separate rows of the profiles plots and deviations boxplots. The default of `"."` results in no split into rows.
`include.raw`	A `character` indicating whether plots of the raw (unsmoothed) trait, corresponding to the plots of the smoothed traits, are to be included in profile plots. The options are `no`, `alone`, `facet.x`, or `facet.y`. That is, the plots of the raw traits are plotted separately or as part of either `facet.x` or `facet.y`.
`collapse.facets.x`	A `logical` to indicate whether all variables specified by `facets.x` are to be collapsed to a single variable. Note that the smoothing-parameters factors, if present, are always collapsed.
`collapse.facets.y`	A `logical` to indicate whether all variables specified by `facets.y` are to be collapsed to a single variable. Note that the smoothing-parameters factors, if present, are always collapsed.
`facet.labeller`	A `ggplot` `function` for labelling the facets of a plot produced using the `ggplot` function. For more information see `ggplot`.
`facet.scales`	A `character` specifying whether the scales are shared across all facets of a plot ("fixed"), or do they vary across rows (the default, "free_x"), columns ("free_y"), or both rows and columns ("free")?
`breaks.spacing.x`	A `numeric` whose absolute values specifies the distance between breaks for the x-axis in a sequence beginning with the minimum x value and continuing up to the maximum x value. If it is negative, the breaks that do not have x values in `data` will be omitted. Minor breaks will be at half this value or, if these do not correspond to x-values in `data` when `breaks.spacing.x` is negative, have a spacing of one. Thus, when `breaks.spacing.x` is negative, grid lines will only be included for x-values that occur in `data`. These settings can be overwritten by supplying, in `ggplotFuncs`, a `scale_x_continuous` function from `ggplot2`.
`angle.x`	A `numeric` between 0 and 360 that gives the angle of the x-axis text to the x-axis. It can also be set by supplying, in `ggplotFuncs`, a `theme` function from `ggplot2`.
`colour`	A `character` specifying a single colour to use in drawing the lines for the profiles. If colouring according to the values of a variable is required then use `colour.column`.
`colour.column`	A `character` giving the name of a column in `data` over whose values the colours of the lines are to be varied. The colours can be specified using `colour.values`.
`colour.values`	A `character` vector specifying the values of the colours to use in drawing the lines for the profiles. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order (usually alphabetical) within the limits of the scale.

`alpha`	A `numeric` specifying the degrees of transparency to be used in plotting the responses. It is a ratio in which the denominator specifies the number of points (or lines) that must be overplotted to give a solid cover.
`addMediansWhiskers`	A `logical` indicating whether plots over time of the medians and outer whiskers are to be added to the plot. The outer whiskers are related to the whiskers on a box-and-whisker and are defined as the median plus (and minus) 1.5 times the interquartile range (IQR). Points lying outside the whiskers are considered to be potential outliers.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object for a profile plot.
`...`	allows arguments to be passed to other functions; not used at present.

Value

A named `list`.

Author(s)

Chris Brien

See Also

`traitSmooth`, `probeSmooths`, `plotSmoothsComparison` and `args4chosen_plot`.

Examples

```
args4profile_plot(plots.by = "Type",
                  facet.x = "Tuning", facet.y = c("Smarthouse", "Treatment.1"),
                  include.raw = "facet.x",
                  alpha  = 0.4,
                  colour.column = "Method",
                  colour.values = c("orange", "olivedrab"))
```

`args4smoothing` *Creates a* `list` *of the values for the smoothing parameters to be passed to a smoothing function*

Description

Creates a `list` of the values for the smoothing parameters to be passed to a smoothing function. Note that `smoothing.methods`, `spline.types`, `df` and `lambdas` are combined to define the set of smooths. The default settings are optimized for `traitSmooth` so that, if you want to change any of these from their default settings when using `args4smoothing` with a function other than `traitSmooth`, then it is recommended that you specify all of them to ensure that the complete set has been correctly specified. Otherwise, the default settings will be those shown here and these may be different to the default settings shown for the function with which you are using `args4smoothing`.

Usage

```
args4smoothing(smoothing.methods = "logarithmic",
               spline.types = c("NCSS","PS"),
               df = 5:7,
               lambdas = list(PS = round(10^c(-0.5, 0, 0.5, 1),
                                         digits = 3)),
               smoothing.segments = NULL,
               npspline.segments = NULL,
               na.x.action="exclude", na.y.action = "trimx",
               external.smooths = NULL,
               correctBoundaries = FALSE,
               combinations = "allvalid",
               ...)
```

Arguments

`smoothing.methods`	A `character` giving the smoothing method to use. The two possibilites are (i) `"direct"`, for directly smoothing the observed `response`, and (ii) `"logarithmic"`, for smoothing the `log`-transformed `response` and then back-transforming by taking the exponentional of the fitted values.
`spline.types`	A `character` giving the type of spline to use. Currently, the possibilites are (i) `"NCSS"`, for natural cubic smoothing splines, and (ii) `"PS"`, for P-splines.
`df`	A `numeric` with at least one value that specifies, for natural cubic smoothing splines (`NCSS`), the desired equivalent numbers of degrees of freedom of the smooths (trace of the smoother matrix). Lower values result in more smoothing. If `df` = `NULL`, the amount of smoothing can be controlled by including a component named `NCSS` in the `list` for `lambdas`. If `df` is `NULL` and `lambda` does not include a component named `NCSS`, then an error is issued.
`lambdas`	A named `list` or a `numeric` specifying the positive penalties to apply in order to control the amount of smoothing. The amount of smoothing decreases as `lamda` decreases. If `lambdas` is a `list`, then include a components with `lambdas` values and named for each of the specified values of `spline.types` for which `lambdas` are to be used. If `spline.types` includes `PS`, then a component named `PS` with at least one numeric value must be present. If a `numeric`, then it will be converted to a `list` with the single component named `PS`.
`smoothing.segments`	A named `list`, each of whose components is a numeric pair specifying the first and last values of an `times`-interval whose data is to be subjected as an entity to smoothing using splines. The separate smooths will be combined to form a whole smooth for each individual. If `get.rates` includes `smoothed` or is `TRUE`, `rates.method` is `differences` and `ntimes2span` is 2, the smoothed growth rates will be computed over the set of segments; otherwise, they will

	be computed within segments. If `smoothing.segments` is `NULL`, the data is not segmented for smoothing.
`npspline.segments`	A `numeric` specifying, for P-splines (`PS`), the number of equally spaced segments between `min(x)` and `max(x)`, excluding missing values, to use in constructing the B-spline basis for the spline fitting. If `npspline.segments` is NULL, `npspline.segments` is set to the maximum of 10 and `ceiling((nrow(data)-1)/2)` i.e. there will be at least 10 segments and, for more than 22 `times` values, there will be half as many segments as there are `times` values. The amount of smoothing decreases as `npspline.segments` increases. When the data has been segmented for smoothing (`smoothing.segments` is not `NULL`), an `npspline.segments` value can be supplied for each segment.
`na.x.action`	A `character` string that specifies the action to be taken when values of `x` are `NA`. The possible values are `fail`, `exclude` or `omit`. For `exclude` and `omit`, predictions and derivatives will only be obtained for nonmissing values of `x`. The difference between these two codes is that for `exclude` the returned `data.frame` will have as many rows as `data`, the missing values have been incorporated.
`na.y.action`	A `character` string that specifies the action to be taken when values of `y`, or the `response`, are `NA`. The possible values are `fail`, `exclude`, `omit`, `allx`, `trimx`, `ltrimx` or `rtrimx`. For all options, except `fail`, missing values in `y` will be removed before smoothing. For `exclude` and `omit`, predictions and derivatives will be obtained only for nonmissing values of `x` that do not have missing `y` values. Again, the difference between these two is that, only for `exclude` will the missing values be incorporated into the returned `data.frame`. For `allx`, predictions and derivatives will be obtained for all nonmissing `x`. For `trimx`, they will be obtained for all nonmissing `x` between the first and last nonmissing `y` values that have been ordered for `x`; for `ltrimx` and `utrimx` either the lower or upper missing `y` values, respectively, are trimmed.
`external.smooths`	A `data.frame` containing the one or more smooths of a `response` in the column specified by `smoothed.response`. Multiple smoooths should be supplied in long.format with the same columns as the `smooths.frame` `data`, except for the smoothing-parameter columns `Type`, `TunePar`, `TuneVal`, `Tuning` and `Method`. Only those smoothing-parameter columns that are to be used in any of `plots.by`, `plots.group`, `facet.x` and `facet.y` should be included with labels appropriate to the `external.smooths`. Those smoothing-parameter columns not included in `external.smooths` will have columns of `"Other"` added to `external.smooths`. The growth rates will be computed by differencing according to the settings of `get.rates` and `trait.types` in the function that calls `args4smoothing`.
`correctBoundaries`	A `logical` indicating whether the fitted spline values are to have the method of Huang (2001) applied to them to correct for estimation bias at the end-points. Note that `spline.type` must be `NCSS` and `lambda` and `deriv` must be `NULL` for `correctBoundaries` to be set to `TRUE`.
`combinations`	A `character` specifying how the values of the different smoothing parameters are to be combined to specify the smooths that are to be obtained. The option `allvalid` results in a smooth for each of the combinations of the values of

	`smoothing.methods`, `spline.types`, `df` and `lambdas` that are valid; the other smoothing.args will be the same for all smooths. The option `parallel` specifies that, if set, each of four smoothing parameters, `smoothing.methods`, `spline.types`, `df` and `lambdas`, must have the same number of values and that this number is the number of different smooths to be produced. The values of the parameters in the same position within each parameter collectively specify a single smooth. Because the value of only one of `df` and `lambdas` must be specified for a smooth, one of these must be set to `NA` and the other to the desired value for each smooth. If all values for one of them is `NA`, then the argument may be omitted or set to `NULL`. The option `single` is for the specification of a single smooth. This will mean that only one of `df` or `lambdas` should be set.
`...`	allows arguments to be passed to other functions; not used at present.

Value

A named `list`.

Author(s)

Chris Brien

See Also

`traitSmooth` and `probeSmooths`.

Examples

```
args4smoothing(smoothing.methods = "direct",
               spline.types = "NCSS", df = NULL, lambdas = NULL,
               smoothing.segments = NULL, npspline.segments = NULL,
               combinations = "allvalid")
args4smoothing(smoothing.methods = c("log","dir","log"),
               spline.types = c("NCSS","NCSS","PS"),
               df = c(4,5,NA), lambdas = c(NA,NA,0.36),
              combinations = "parallel")
args4smoothing(smoothing.methods = "log",
               spline.types = "PS", df = NULL,
               lambdas = 0.36, combinations = "single")
```

`as.smooths.frame` *Forms a* `smooths.frame` *from a* `data.frame`*, ensuring that the correct columns are present.*

Description

Creates a `smooths.frame` from a `data.frame` by adding the class `smooths.frame` and a set of `attributes` to it.

Usage

```
as.smooths.frame(data, individuals = NULL, times = NULL)
```

Arguments

`data`	A `data.frame` containing the results of smoothing the data on a set of `individuals` over time, the data being arranged in long format both with respect to the times and the smoothing-parameter values. It must contain the columns `Type`, `TunePar`, `TuneVal`, `Tuning` and `Method` that give the smoothing-parameter values that were used to produce each smooth of the data, as well as the columns identifying the `individuals`, the observation times of the responses and the unsmoothed and smoothed responses. Each response occupies a single column.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the `numeric`, or `factor` with numeric levels, that contains the values of the predictor variable to be supplied to `smooth.spline` and to be plotted on the x-axis.

Value

A `smooths.frame`

Author(s)

Chris Brien

See Also

`validSmoothsFrame`, `as.smooths.frame`

Examples

```
dat <- read.table(header = TRUE, text = ”
Type TunePar TuneVal Tuning Method       ID  DAP   PSA      sPSA
NCSS      df       4    df-4 direct 045451-C   28 57.446 51.18456
NCSS      df       4    df-4 direct 045451-C   30 89.306 87.67343
NCSS      df       7    df-7 direct 045451-C   28 57.446 57.01589
NCSS      df       7    df-7 direct 045451-C   30 89.306 87.01316
”)
dat[1:7] <- lapply(dat[1:6], factor)
dat <- as.smooths.frame(dat, individuals = ”ID”, times = ”DAP”)
is.smooths.frame(dat)
validSmoothsFrame(dat)
```

`byIndv4Intvl_GRsAvg` *Calculates the growth rates for a specified time interval for individuals in a* `data.frame` *in long format by taking weighted averages of growth rates for times within the interval.*

Description

Using previously calculated growth rates over time, calculates the Absolute Growth Rates for a specified interval using the weighted averages of AGRs for each time point in the interval (AGR) and the Relative Growth Rates for a specified interval using the weighted geometric means of RGRs for each time point in the interval (RGR).

Usage

```
byIndv4Intvl_GRsAvg(data, responses,
                    individuals = "Snapshot.ID.Tag", times = "DAP",
                    which.rates = c("AGR","RGR"),
                    suffices.rates=c("AGR","RGR"), sep.rates = ".",
                    start.time, end.time,
                    suffix.interval, sep.suffix.interval = ".",
                    sep.levels=".", na.rm=FALSE)
```

Arguments

`data`	A `data.frame` containing the columns from which the growth rates are to be calculated.
`responses`	A `character` giving the names of the `responses` for which there are columns in `data` that contain the growth rates that are to be averaged. The names of the growth rates should have either `AGR` or `RGR` appended to the `responses` names.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used in calculating growth rates and, if a `factor` or `character`, the values should be numerics stored as characters.
`which.rates`	A `character` giving the growth rates that are to be averaged to obtain growth rates for an interval. It should be a combination of one or more of `"AGR"` and `"RGR"`.
`suffices.rates`	A `character` giving the suffices to be appended to `response` to form the names of the columns containing the calculated the growth rates and in which growth rates are to be stored. Their elements will be matched with those of `which.rates`.
`sep.rates`	A `character` giving the character(s) to be used to separate the `suffices.rates` value from a `response` value in constructing the name for a new rate. For no separator, set to `""`.

`start.time`	A `numeric` giving the times, in terms of values in `times`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the start of the interval for which the growth rate is to be calculated.
`end.time`	A `numeric` giving the times, in terms of values `times`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the end of the interval for which the growth rate is to be calculated.
`suffix.interval`	A `character` giving the suffix to be appended to `response.suffices.rates` to form the names of the columns containing the calculated the growth rates.
`sep.suffix.interval`	A `character` giving the separator to use in appending `suffix.inteval` to a growth rate. For no separator, set to `""`.
`sep.levels`	A `character` giving the separator to use when the levels of `individuals` are combined. This is needed to avoid using a `character` that occurs in a `factor` to delimit levels when the levels of `individuals` are combined to identify subsets.
`na.rm`	A `logical` indicating whether `NA` values should be stripped before the calculation of weighted means proceeds.

Details

The `AGR` for an interval is calculated as the weighted mean of the AGRs for times within the interval. The RGR is calculated as the weighted geometric mean of the RGRs for times within the interval; in fact the exponential is taken of the weighted means of the logs of the RGRs. The weights are obtained from the `times`. They are taken as the sum of half the time subintervals before and after each time, except for the end points; the end points are taken to be the subintervals at the start and end of the interval.

Value

A `data.frame` with the growth rates. The name of each column is the concatenation of (i) one of `responses`, (ii) one of `AGR`, `PGR` or `RGR`, or the appropriate element of `suffices.rates`, and (iii) `suffix.interval`, the three components being separated by full stops.

Author(s)

Chris Brien

See Also

`byIndv4Intvl_GRsDiff`, `byIndv4Intvl_WaterUse`, `getTimesSubset`,
`GrowthRates`, `byIndv4Times_SplinesGRs`

Examples

```
data(exampleData)
longi.dat <- byIndv4Times_SplinesGRs(data = longi.dat,
                                     response="PSA", response.smoothed = "sPSA",
                                     individuals = "Snapshot.ID.Tag",
                                     times = "DAP",
```

```
                                      df = 4,
                                      rates.method = "deriv",
                                      which.rates = c("AGR", "RGR"),
                                      suffices.rates = c("AGRdv", "RGRdv"))
sPSA.GR <- byIndv4Intvl_GRsAvg(data = longi.dat,
                               response="sPSA", times = "DAP",
                               which.rates = c("AGR","RGR"),
                               suffices.rates = c("AGRdv","RGRdv"),
                               start.time = 31, end.time = 35,
                               suffix.interval = "31to35")
```

`byIndv4Intvl_GRsDiff` *Calculates the growth rates for a specified time interval for individuals in a* `data.frame` *in long format by differencing the values for a response within the interval.*

Description

Using the values of the responses, calculates the specified combination of the Absolute Growth Rates using differences (AGR), the Proportionate Growth Rates (PGR) and Relative Growth Rates using log differences (RGR) between two nominated time points.

Usage

```
byIndv4Intvl_GRsDiff(data, responses,
                     individuals = "Snapshot.ID.Tag", times = "DAP",
                     which.rates = c("AGR","PGR","RGR"),
                     suffices.rates=NULL, sep.rates = ".",
                     start.time, end.time,
                     suffix.interval, sep.suffix.interval = ".")
```

Arguments

`data`	A `data.frame` containing the column from which the growth rates are to be calculated.
`responses`	A `character` giving the names of the columns in `data` from which the growth rates are to be calculated.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used in calculating growth rates and, if a `factor` or `character`, the values should be numerics stored as characters.
`which.rates`	A `character` giving the growth rates that are to be calculated. It should be a combination of one or more of `"AGR"`, `"PGR"` and `"RGR"`.

`suffices.rates`	A `character` giving the characters to be appended to the names of the responses in constructing the names of the columns containing the calculated growth rates. The order of the suffices in `suffices.rates` should correspond to the order of the elements of `which.rates`.
`sep.rates`	A `character` giving the character(s) to be used to separate the `suffices.rates` value from a `response` value in constructing the name for a new rate. For no separator, set to `""`.
`start.time`	A `numeric` giving the times, in terms of values in `times`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the start of the interval for which the growth rate is to be calculated.
`end.time`	A `numeric` giving the times, in terms of values `times`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the end of the interval for which the growth rate is to be calculated.
`suffix.interval`	A `character` giving the suffix to be appended to `response` to form the names of the columns containing the calculated the growth rates.
`sep.suffix.interval`	A `character` giving the separator to use in appending `suffix.inteval` to a growth rate. For no separator, set to `""`.

Details

The `AGR` is calculated as the difference between the values of `response` at the `end.time` and `start.time` divided by the difference between `end.time` and `start.time`. The PGR is calculated as the ratio of `response` at the `end.time` to that at `start.time` and the ratio raised to the power of the reciprocal of the difference between `end.time` and `start.time`. The `RGR` is calculated as the `log` of the PGR and so is equal to the difference between the logarithms of `response` at the `end.time` and `start.time` divided by the difference between `end.time` and `start.time`.

Value

A `data.frame` with the growth rates. The name of each column is the concatenation of (i) one of `responses`, (ii) one of `AGR`, `PGR` or `RGR`, or the appropriate element of `suffices.rates`, and (iii) `suffix.interval`, the three components being separated by full stops.

Author(s)

Chris Brien

See Also

`byIndv4Intvl_GRsAvg`, `byIndv4Intvl_WaterUse`, `getTimesSubset`, `GrowthRates`, `byIndv4Times_SplinesGRs`

Examples

```
data(exampleData)
sPSA.GR <- byIndv4Intvl_GRsDiff(data = longi.dat,
```

```
                    responses = "sPSA", times = "DAP",
                    which.rates = c("AGR","RGR"),
                    start.time = 31, end.time = 35,
                    suffix.interval = "31to35")
```

`byIndv4Intvl_ValueCalc`

Calculates a single value that is a `function` of the values of an individual for a response in a `data.frame` in long format over a specified time interval.

Description

Splits the values of a response into subsets corresponding individuals and applies a function that calculates a single value from each individual's observations during a specified time interval. It includes the ability to calculate the observation number that is closest to the calculated value of the function and the assocated values of a `factor` or `numeric`.

Usage

```
byIndv4Intvl_ValueCalc(data, response,
                       individuals = "Snapshot.ID.Tag", times = "DAP",
                       FUN = "max", which.obs = FALSE, which.values = NULL,
                       addFUN2name = TRUE, sep.FUNname = ".",
                       start.time=NULL, end.time=NULL,
                       suffix.interval=NULL, sep.suffix.interval = ".",
                       sep.levels=".", weights=NULL, na.rm=TRUE, ...)
```

Arguments

`data`	A `data.frame` containing the column from which the function is to be calculated.
`response`	A `character` giving the name of the column in `data` from which the values of `FUN` are to be calculated.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used in calculating growth rates and, if a `factor` or `character`, the values should be numerics stored as characters.
`FUN`	A `character` giving the name of the `function` that calculates the value for each subset.
`which.obs`	A `logical` indicating whether or not to determine the observation number corresponding to the observed value that is closest to the value of the function, in addition to the value of the function itself. That is, `FUN` need not return an observed value of the `reponse`, e.g. `quantile`.

`which.values`	A `character` giving the name of the `factor` or `numeric` whose values are associated with the `response` values and whose value is to be returned for the observation number whose `response` value corresponds to the observed value closest to the value of the function. That is, `FUN` need not return an observed value of the `reponse`, e.g. `quantile`. In the case of multiple observed response values satisfying this condition, the value of the `which.values` vector for the first of these is returned.
`addFUN2name`	A `logical` that, if `TRUE`, indicates that the `FUN` name is to be added to the names of the columns in the `data.frame` returned by `byIndv4Intvl_ValueCalc`.
`sep.FUNname`	A `character` giving the character(s) to be used to separate the name of `FUN` from the `response` value in constructing the name for a new response. For no separator, set to `""`.
`start.time`	A `numeric` giving the times, in terms of levels of `times.factor`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the start of the interval for which a value is to be calculated. If `start.time` is `NULL`, the interval will start with the first observation. In the case of multiple observed response values satisfying this condition, the first is returned.
`end.time`	A `numeric` giving the times, in terms of levels of `times.factor`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the end of the interval for which a value is to be calculated. If `end.time` is `NULL`, the interval will end with the last observation.
`suffix.interval`	A `character` giving the suffix to be appended to `response` to form the name of the column containing the calculated values. If it is `NULL` then nothing will be appended.
`sep.suffix.interval`	A `character` giving the separator to use in appending `suffix.inteval` to a growth rate. For no separator, set to `""`.
`sep.levels`	A `character` giving the separator to use when the levels of `individuals` are combined. This is needed to avoid using a `character` that occurs in a `factor` to delimit levels when the levels of `individuals` are combined to identify subsets.
`weights`	A `character` giving the name of the column in `data` containing the weights to be supplied as `w` to `FUN`.
`na.rm`	A `logical` indicating whether `NA` values should be stripped before the calculation proceeds.
`...`	allows for arguments to be passed to `FUN`.

Value

A `data.frame`, with the same number of rows as there are `individuals`, containing a column for the `individuals` and a column with the values of the function for the `individuals`. It is also possible to determine observaton numbers or the values of another column in `data` for the `response` values that are closest to the `FUN` results, using either or both of `which.obs` and `which.values`. If `which.obs` is `TRUE`, a column with observation numbers is included in the `data.frame`. If `which.values` is set to the name of a `factor` or a `numeric`, a column containing the levels of that `factor` or the values of that `numeric` is included in the `data.frame`.

The name of the column with the values of the function will be result of concatenating the `response`, `FUN` and, if it is not `NULL`, `suffix.interval`, each separated by a full stop. If `which.obs` is `TRUE`, the column name for the obervations numbers will have `.obs` added after `FUN` into the column name for the function values; if `which.values` is specified, the column name for these values will have a full stop followed by `which.values` added after `FUN` into the column name for the function values.

Author(s)

Chris Brien

See Also

`byIndv4Intvl_GRsAvg`, `byIndv4Intvl_GRsDiff`, `byIndv4Intvl_WaterUse`, `getTimesSubset`

Examples

```
data(exampleData)
sPSA.max <- byIndv4Intvl_ValueCalc(data = longi.dat,
                                   response = "sPSA", times = "DAP",
                                   start.time = 31, end.time = 35,
                                   suffix.interval = "31to35")
AGR.max.dat <- byIndv4Intvl_ValueCalc(data = longi.dat,
                                      response = "sPSA", times = "DAP",
                                      FUN="max",
                                      start.time = 31, end.time = 35,
                                      suffix.interval = "31to35",
                                      which.values = "DAP",
                                      which.obs = TRUE)
```

---

byIndv4Intvl_WaterUse *Calculates, water use traits (WU, WUR, WUI) over a specified time interval for each individual in a* `data.frame` *in long format.*

Description

Calculates one or more of water use (WU), water use rate (WUR), and, for a set of responses, water use indices (WUI)s over a specified time interval for each individual in a `data.frame` in long format.

Usage

```
byIndv4Intvl_WaterUse(data, water.use = "Water.Use", responses = NULL,
                      individuals = "Snapshot.ID.Tag", times = "DAP",
                      trait.types = c("WU", "WUR", "WUI"),
                      suffix.rate = "R", suffix.index = "I",
                      sep.water.traits = "", sep.responses = ".",
                      start.time, end.time,
                      suffix.interval = NULL, sep.suffix.interval = ".",
                      na.rm = FALSE)
```

Arguments

`data`	A `data.frame` containing the column from which the water use traits are to be calculated.
`water.use`	A `character` giving the names of the column in `data` that contains the water use values.
`responses`	A `character` giving the names of the columns in `data` for which `WUI`s are to be calculated.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used identifying the intervals and, if a `factor` or `character`, the values should be numerics stored as characters.
`trait.types`	A `character` listing the trait types to compute and return. It should be some combination of `WU`, `WUR` and `WUI`, or be `all`. See Details for how each is calculated.
`suffix.rate`	A `character` giving the label to be appended to the value of `water.use` to form the name of the WUR.
`suffix.index`	A `character` giving the label to be appended to the value of `water.use` to form the name of the WUI.
`sep.water.traits`	A `character` giving the character(s) to be used to separate the `suffix.rate` and `suffix.index` values from the `responses` values in constructing the name for a new rate/index. The default of `""` results in no separator.
`sep.responses`	A `character` giving the character(s) to be used to separate the `suffix.rate` value from a`responses` value in constructing the name for a new index. For no separator, set to `""`.
`start.time`	A `numeric` giving the times, in terms of values in `times`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the start of the interval for which the WUI is to be calculated.
`end.time`	A `numeric` giving the times, in terms of values `times`, that will give a single value for each `Snapshot.ID.Tag` and that will be taken as the observation at the end of the interval for which the WUI is to be calculated.
`suffix.interval`	A `character` giving the suffix to be appended to the names of the columns for the water use traits to indicate the interval for which the traits have been calculated .
`sep.suffix.interval`	A `character` giving the separator to use in appending `suffix.inteval` to a growth rate. For no separator, set to `""`.
`na.rm`	A `logical` indicating whether `NA` values should be stripped before the calculation proceeds.

Details

- `WU` is the water use and is the sum of the water use after `start.time` until `end.time`. Thus, the water use up to `start.time` is not included. Further, if any water use value in the interval is `NA`, the interval value will be set to `NA`.
- `WUR` is the Water Use Rate and is `WU` divided by the difference between `end.time` and `start.time`.
- `WUI` is the Water Use Index and is calculated as a `response` difference between the start.time and the end.time, which is then divided by the `WU`.

Value

A `data.frame` containing the `individuals` column, `WU` and/or `WUR` and, if requested, a `WUI` for each element of `responses`. The names of `WU` and `WUR` will have `suffix.interval` appended, if it is not `NULL`, separated by a full stop ('.'). The name of each `WUI` will be the concatenation of an element of `responses` with `WUI` and, if not `NULL`, `suffix.interval`, the three components being separated by a full stop ('.').

Author(s)

Chris Brien

See Also

`byIndv4Intvl_GRsAvg`, `byIndv4Intvl_GRsDiff`, `getTimesSubset`, `GrowthRates`

Examples

```
data(exampleData)
WU.WUI_31_35 <- byIndv4Intvl_WaterUse(data = longi.dat,
                                      water.use = "WU", responses = "PSA",
                                      times = "DAP",
                                      trait.types = c("WUR","WUI"),
                                      suffix.rate = ".Rate",
                                      suffix.index = ".Index",
                                      start.time = 31, end.time = 35,
                                      suffix.interval = "31to35")
```

`byIndv4Times_GRsDiff` *Adds, to a* `data.frame`*, the growth rates calculated for consecutive times for individuals in a* `data.frame` *in long format by differencing response values.*

Description

Uses `AGRdiff`, `PGR` and `RGRdiff` to calculate growth rates continuously over time for the `response` by differencing pairs of pairs of `response` values and stores the results in `data`. The subsets are those values with the same levels combinations of the `factor`s listed in `individuals`.

If `avail.time.diffs` is `FALSE`, the differences between consecutive time values are calculated. For this, it is assumed that the same first `times` value is present in `data` for all `individuals`.

Usage

```
byIndv4Times_GRsDiff(data, responses,
                      individuals = "Snapshot.ID.Tag", times = "DAP",
                      which.rates = c("AGR","PGR","RGR"),
                      suffices.rates=NULL, sep.rates = ".",
                      avail.times.diffs = FALSE, ntimes2span = 2)
```

Arguments

`data`	A `data.frame` containing the columns for which growth rates are to be calculated.
`responses`	A `character` giving the names of the columns in `data` for which growth rates are to be calculated.
`individuals`	A `character` giving the name(s) of the `factor`(s) that define the subsets of `response` that correspond to the `response` values for an individual (e.g. plant, pot, cart, plot or unit) for which growth rates are to be calculated continuously. If the columns corresponding to `individuals` are not `factor`(s) then they will be coerced to `factor`(s). The subsets are formed using `split`.
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used in calculating the growth rates. If a `factor` or `character`, the values should be numerics stored as characters.
`which.rates`	A `character` giving the growth rates that are to be calculated. It should be a combination of one or more of `"AGR"`, `"PGR"` and `"RGR"`.
`suffices.rates`	A `character` giving the characters to be appended to the names of the responses to provide the names of the columns containing the calculated growth rates. The order of the suffices in `suffices.rates` should correspond to the order of the elements of `which.rates`. If `NULL`, the values of `which.rates` are used.
`sep.rates`	A `character` giving the character(s) to be used to separate the `suffices.rates` value from a `response` value in constructing the name for a new rate. For no separator, set to `""`.
`avail.times.diffs`	A `logical` indicating whether there is an appropriate column of `times` diffserences that can be used as the denominator in computing the growth rates. If `TRUE`, it will be assumed that the name of the column is the value of `times` with `.diffs` appended. If `FALSE`, a column, whose column name will be the value of `times` with `.diffs` appended, will be formed and saved in the result, overwriting any existing columns with the constructed name in `data`. It will be calculated using the values of times in `data`.
`ntimes2span`	A `numeric` giving the number of values in `times` to span in calculating growth rates by differencing. Each growth rate is calculated as the difference in the values of one of the `responses` for pairs of `times` values that are spanned by `ntimes2span` `times` values divided by the difference between this pair of `times` values. For `ntimes2span` set to 2, a growth rate is the difference between consecutive pairs of values of one of the `responses` divided by the difference between consecutive pairs of `times` values.

Value

A `data.frame` containing `data` to which has been added i) a column for the differences between the `times`, if it is not already in `data`, and (ii) columns with growth rates. The name of the column for `times` differences will be the value of `times` with `".diffs"` appended. The name for each of the growth-rate columns will be either the value of `response` with one of `".AGR"`, `".PGR"` or `"RGR"`, or the corresponding value from `suffices.rates` appended. Each growth rate will be positioned at observation `ceiling(ntimes2span + 1) / 2` relative to the two times from which the growth rate is calculated.

Author(s)

Chris Brien

See Also

`smoothSpline`, `byIndv4Times_SplinesGRs`

Examples

```
data(exampleData)
longi.dat <- byIndv4Times_GRsDiff(data = longi.dat,
                                  response = "sPSA",
                                  individuals = "Snapshot.ID.Tag",
                                  times = "DAP",
                                  which.rates=c("AGR", "RGR"),
                                  avail.times.diffs = TRUE)
```

`byIndv4Times_SplinesGRs`

For a response in a `data.frame` *in long format, computes, for a single set of smoothing parameters, smooths of the response, possibly along with growth rates calculated from the smooths.*

Description

Uses `smoothSpline` to fit a spline to the values of `response` for each `individual` and stores the fitted values in `data`. The degree of smoothing is controlled by the tuning parameters `df` and `lambda`, related to the penalty, and by `npspline.segments`. The `smoothing.method` provides for `direct` and `logarithmic` smoothing.

The Absolute and Relative Growth Rates (AGR and RGR) can be computed either using the first derivatives of the splines or by differencing the smooths. If using the first derivative to obtain growth rates, `correctBoundaries` must be `FALSE`. Derivatives other than the first derivative can also be produced. The function `byIndv4Times_GRsDiff` is used to obtain growth rates by differencing.

The handling of missing values in the observations is controlled via `na.x.action` and `na.y.action`. If there are not at least four distinct, nonmissing x-values, a warning is issued and all smoothed values and derivatives are set to `NA`.

The function `probeSmooths` can be used to investgate the effect the smoothing parameters (`smoothing.method`, `df` or `lambda`) on the smooth that results.

Usage

```
byIndv4Times_SplinesGRs(data, response, response.smoothed = NULL,
                        individuals = "Snapshot.ID.Tag", times,
                        smoothing.method = "direct", smoothing.segments = NULL,
                        spline.type = "NCSS", df=NULL, lambda = NULL,
                        npspline.segments = NULL,
                        correctBoundaries = FALSE,
                        rates.method = "differences",
                        which.rates = c("AGR","RGR"),
                        suffices.rates = NULL, sep.rates = ".",
                        avail.times.diffs = FALSE, ntimes2span = 2,
                        extra.derivs = NULL, suffices.extra.derivs=NULL,
                        sep.levels = ".",
                        na.x.action="exclude", na.y.action = "trimx", ...)
```

Arguments

`data`	A `data.frame` containing the column to be smoothed.
`response`	A `character` giving the name of the column in `data` that is to be smoothed.
`response.smoothed`	A `character` specifying the name of the column containing the values of the smoothed response variable, corresponding to `response`. If `response.smoothed` is `NULL`, then `response.smoothed` is set to the `response` to which is added the prefix `s.`
`individuals`	A `character` giving the name(s) of the `factor`(s) that define the subsets of `response` that correspond to the `response` values for an individual (e.g. plant, pot, cart, plot or unit) that are to be smoothed separately. If the columns corresponding to `individuals` are not `factor`(s) then they will be coerced to `factor`(s). The subsets are formed using `split`.
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used as the values of the predictor variable to be supplied to `smooth.spline` and in calculating growth rates. If a `factor` or `character`, the values should be numerics stored as characters.
`smoothing.method`	A `character` giving the smoothing method to use. The two possibilites are (i) `"direct"`, for directly smoothing the observed `response`, and (ii) `"logarithmic"`, for smoothing the `log`-transformed `response` and then back-transforming by taking the exponentional of the fitted values.
`smoothing.segments`	A named `list`, each of whose components is a numeric pair specifying the first and last values of an `times`-interval whose data is to be subjected as an entity to smoothing using splines. The separate smooths will be combined to form a whole smooth for each individual. If `get.rates` is `TRUE`, `rates.method` is `differences` and `ntimes2span` is 2, the smoothed growth rates will be computed over the set of segments; otherwise, they will be computed within segments. If `smoothing.segments` is `NULL`, the data is not segmented for smoothing.

`spline.type`	A `character` giving the type of spline to use. Currently, the possibilites are (i) `"NCSS"`, for natural cubic smoothing splines, and (ii) `"PS"`, for P-splines.
`df`	A `numeric` specifying, for natural cubic smoothing splines (`NCSS`), the desired equivalent number of degrees of freedom of the smooth (trace of the smoother matrix). Lower values result in more smoothing. If `df = NULL`, the amount of smoothing can be controlled by setting `lambda`. If both `df` and `lambda` are `NULL`, smoothing is controlled by the default arguments for `smooth.spline`, and any that you supply via the ellipsis (`...`) argument.
`lambda`	A `numeric` specifying the positive penalty to apply. The amount of smoothing decreases as `lamda` decreases.
`npspline.segments`	A `numeric` specifying, for P-splines (`PS`), the number of equally spaced segments between `min(times)` and `max(times)`, excluding missing values, to use in constructing the B-spline basis for the spline fitting. If `npspline.segments` is NULL, `npspline.segments` is set to the maximum of 10 and `ceiling((nrow(data)-1)/2)` i.e. there will be at least 10 segments and, for more than 22 `times` values, there will be half as many segments as there are `times` values. The amount of smoothing decreases as `npspline.segments` increases. When the data has been segmented for smoothing (`smoothing.segments` is not `NULL`), an `npspline.segments` value can be supplied for each segment.
`correctBoundaries`	A `logical` indicating whether the fitted spline values are to have the method of Huang (2001) applied to them to correct for estimation bias at the end-points. Note that `spline.type` must be `NCSS` and `lambda` and `deriv` must be `NULL` for `correctBoundaries` to be set to `TRUE`.
`rates.method`	A `character` specifying the method to use in calculating the growth rates. The possibilities are `none`, `differences` and `derivatives`.
`which.rates`	A `character` giving the growth rates that are to be calculated. It should be a combination of one or more of `"AGR"`, `"PGR"` and `"RGR"`.
`suffices.rates`	A `character` giving the characters to be appended to the names of the responses to provide the names of the columns containing the calculated growth rates. The order of the suffices in `suffices.rates` should correspond to the order of the elements of `which.rates`. If `NULL`, the values of `which.rates` are used.
`sep.rates`	A `character` giving the character(s) to be used to separate the `suffices.rates` value from a `response` value in constructing the name for a new rate. For no separator, set to `""`.
`avail.times.diffs`	A `logical` indicating whether there is an appropriate column of `times` diffserences that can be used as the denominator in computing the growth rates. If `TRUE`, it will be assumed that the name of the column is the value of `times` with `.diffs` appended. If `FALSE`, a column, whose column name will be the value of `times` with `.diffs` appended, will be formed and saved in the result, overwriting any existing columns with the constructed name in `data`. It will be calculated using the values of times in `data`.
`ntimes2span`	A `numeric` giving the number of values in `times` to span in calculating growth rates by differencing. Each growth rate is calculated as the difference in the

	values of one of the `responses` for pairs of `times` values that are spanned by `ntimes2span` `times` values divided by the difference between this pair of `times` values. For `ntimes2span` set to 2, a growth rate is the difference between consecutive pairs of values of one of the `responses` divided by the difference between consecutive pairs of `times` values.
`extra.derivs`	A `numeric` specifying one or more orders of derivatives that are required, in addition to any required for calculating the growth rates. When `rates.method` is `derivatives`, these can be derivatives other than the first. Otherwise, any derivatives can be specified.
`suffices.extra.derivs`	A `character` giving the characters to be appended to `response.method` to construct the names of the derivatives. If `NULL` and the derivatives are to be retained, then `.dv` followed by the order of the derivative is appended to `response.method`
.	
`sep.levels`	A `character` giving the separator to use when the levels of `individuals` are combined. This is needed to avoid using a `character` that occurs in a `factor` to delimit levels when the levels of `individuals` are combined to identify subsets.
`na.x.action`	A `character` string that specifies the action to be taken when values of `x`, or the `times`, are `NA`. The possible values are `fail`, `exclude` or `omit`. For `exclude` and `omit`, predictions and derivatives will only be obtained for nonmissing values of `x`. The difference between these two codes is that for `exclude` the returned `data.frame` will have as many rows as `data`, the missing values have been incorporated.
`na.y.action`	A `character` string that specifies the action to be taken when values of `y`, or the `response`, are `NA`. The possible values are `fail`, `exclude`, `omit`, `allx`, `trimx`, `ltrimx` or `rtrimx`. For all options, except `fail`, missing values in `y` will be removed before smoothing. For `exclude` and `omit`, predictions and derivatives will be obtained only for nonmissing values of `x` that do not have missing `y` values. Again, the difference between these two is that, only for `exclude` will the missing values be incorporated into the returned `data.frame`. For `allx`, predictions and derivatives will be obtained for all nonmissing `x`. For `trimx`, they will be obtained for all nonmissing `x` between the first and last nonmissing `y` values that have been ordered for `x`; for `ltrimx` and `utrimx` either the lower or upper missing `y` values, respectively, are trimmed.
`...`	allows for arguments to be passed to `smooth.spline`.

Value

A `data.frame` containing `data` to which has been added a column with the fitted smooth, the name of the column being the value of `response.smoothed`. If `rates.method` is not `none`, columns for the growth rates listed in `which.rates` will be added to `data`; the names each of these columns will be the value of `response.smoothed` with the elements of `which.rates` appended.

When `rates.method` is `derivatives` and `smoothing.method` is `direct`, the `AGR` is obtained from the first derivative of the spline for each value of `times` and the `RGR` is calculated as the `AGR` divided by the value of the `response.smoothed` for the corresponding time. When `rates.method` is `derivatives` and `smoothing.method` is `logarithmic`, the `RGR` is obtained from the first derivative of the spline and the `AGR` is calculated as the `RGR` multiplied by the corresponding value of the `response.smoothed`.

If `extra.derivs` is not `NULL`, the values for the nominated derivatives will also be added to `data`; the names each of these columns will be the value of `response.smoothed` with `.dvf` appended, where `f` is the order of the derivative, or the value of `response.smoothed` with the corresponding element of `suffices.deriv` appended.

Any pre-existing smoothed and growth rate columns in `data` will be replaced. The ordering of the `data.frame` for the `times` values will be preserved as far as is possible; the main difficulty is with the handling of missing values by the function `merge`. Thus, if missing values in `times` are retained, they will occur at the bottom of each subset of `individuals` and the order will be problematic when there are missing values in `y` and `na.y.action` is set to `omit`.

Author(s)

Chris Brien

References

Eilers, P.H.C and Marx, B.D. (2021) *Practical smoothing: the joys of P-splines*. Cambridge University Press, Cambridge.

Huang, C. (2001) Boundary corrected cubic smoothing splines. *Journal of Statistical Computation and Simulation*, **70**, 107-121.

See Also

`smoothSpline`, `probeSmooths`, `byIndv4Times_GRsDiff`, `smooth.spline`, `predict.smooth.spline`, `split`

Examples

```
data(exampleData)
#smoothing with growth rates calculated using derivates
longi.dat <- byIndv4Times_SplinesGRs(data = longi.dat,
                                     response="PSA", response.smoothed = "sPSA",
                                     times="DAP",
                                     df = 4, rates.method = "deriv",
                                     suffices.rates = c("AGRdv", "RGRdv"))
#Use P-splines
longi.dat <- byIndv4Times_SplinesGRs(data = longi.dat,
                                     response="PSA", response.smoothed = "sPSA",
                                     individuals = "Snapshot.ID.Tag", times="DAP",
                                     spline.type = "PS", lambda = 0.1,
                                     npspline.segments = 10,
                                     rates.method = "deriv",
                                     suffices.rates = c("AGRdv", "RGRdv"))
#with segmented smoothing and no growth rates
longi.dat <- byIndv4Times_SplinesGRs(data = longi.dat,
                                     response="PSA", response.smoothed = "sPSA",
                                     individuals = "Snapshot.ID.Tag", times="DAP",
```

```
                    smoothing.segments = list(c(28,34), c(35,42)),
                    df = 5, rates.method = "none")
```

`byIndv_ValueCalc` *Calculates a single value that is a function of an individual's values for a response.*

Description

Applies a `function` to calculate a single value from an individual's values for a response in a `data.frame` in long format. It includes the ability to calculate the observation number that is closest to the calculated value of the function and the assocated values of a `factor` or `numeric`.

Usage

```
byIndv_ValueCalc(data, response, individuals = "Snapshot.ID.Tag",
                 FUN = "max", which.obs = FALSE, which.values = NULL,
                 addFUN2name = TRUE, sep.FUNname = ".",
                 weights = NULL, na.rm=TRUE, sep.levels = ".", ...)
```

Arguments

`data`	A `data.frame` containing the column from which the function is to be calculated.
`response`	A `character` giving the name of the column in `data` from which the values of `FUN` are to be calculated.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`FUN`	A `character` giving the name of the `function` that calculates the value for each subset.
`which.obs`	A `logical` indicating whether or not to determine the observation number corresponding to the observed value that is closest to the value of the function, in addition to the value of the function itself. That is, `FUN` need not return an observed value of the `reponse`, e.g. `quantile`. In the case of multiple observed response values satisfying this condition, the first is returned.
`which.values`	A `character` giving the name of the `factor` or `numeric` whose values are associated with the `response` values and whose value is to be returned for the observation number whose `response` value corresponds to the observed value closest to the value of the function. That is, `FUN` need not return an observed value of the `reponse`, e.g. `quantile`. In the case of multiple observed response values satisfying this condition, the value of the `which.values` vector for the first of these is returned.
`addFUN2name`	A `logical` that, if `TRUE`, indicates that the `FUN` name is to be added to the names of the columns in the `data.frame` returned by `byIndv4Intvl_ValueCalc`.

`sep.FUNname`	A `character` giving the character(s) to be used to separate the name of `FUN` from the `response` value in constructing the name for a new response. For no separator, set to `""`.
`weights`	A `character` giving the name of the column in `data` containing the weights to be supplied as `w` to `FUN`.
`na.rm`	A `logical` indicating whether `NA` values should be stripped before the calculation proceeds.
`sep.levels`	A `character` giving the separator to use when the levels of `individuals` are combined. This is needed to avoid using a `character` that occurs in a `factor` to delimit levels when the levels of `individuals` are combined to identify subsets.
`...`	allows for arguments to be passed to `FUN`.

Value

A `data.frame`, with the same number of rows as there are `individuals`, containing a column for the `individuals` and a column with the values of the function for the `individuals`. It is also possible to determine observaton numbers or the values of another column in `data` for the `response` values that are closest to the `FUN` results, using either or both of `which.obs` and `which.values`. If `which.obs` is `TRUE`, a column with observation numbers is included in the `data.frame`. If `which.values` is set to the name of a `factor` or a `numeric`,a column containing the levels of that `factor` or the values of that `numeric` is included in the `data.frame`.

The name of the column with the values of the function will be formed by concatenating the `response` and `FUN`, separated by a full stop. If `which.obs` is `TRUE`, the column name for the obervations numbers will have `.obs` added after `FUN` into the column name for the function values; if `which.values` is specified, the column name for these values will have a full stop followed by `which.values` added after `FUN` into the column name for the function values.

Author(s)

Chris Brien

See Also

`byIndv4Intvl_ValueCalc`, `byIndv4Times_GRsDiff`, `byIndv4Times_SplinesGRs`

Examples

```
data(exampleData)
sPSA.max.dat <- byIndv_ValueCalc(data=longi.dat, response = "PSA")
AGR.max.dat <- byIndv_ValueCalc(data=longi.dat, response = "sPSA.AGR",
                                FUN="max",
                                which.values = "DAP", which.obs = TRUE)
sPSA.dec1.dat <- byIndv_ValueCalc(data=longi.dat, response = "sPSA",
                                  FUN="quantile",
                                  which.values = "DAP", probs = 0.1)
```

`calcLagged` *Replaces the values in a vector with the result of applying an operation to it and a lagged value*

Description

Replaces the values in `x` with the result of applying an `operation` to it and the value that is `lag` positions either before it or after it in `x`, depending on whether `lag` is positive or negative. For positive `lag` the first `lag` values will be `NA`, while for negative `lag` the last `lag` values will be `NA`. When `operation` is `NULL`, the values are moved `lag` positions down the vector.

Usage

```
calcLagged(x, operation = NULL, lag = 1)
```

Arguments

`x`	A `vector` containing the values on which the calculations are to be made.
`operation`	A `character` giving the operation to be performed on pairs of values in `x`. If `operation` is `NULL` then the values are moved `lag` positions down the vector.
`lag`	A `integer` specifying, for the second value in the pair to be operated on, the number positions it is ahead of or behind the current value.

Value

A `vector` containing the result of applying `operation` to values in `x`. For positive `lag` the first `lag` values will be `NA`, while for negative `lag` the last `lag` values will be `NA`.

Author(s)

Chris Brien

See Also

`Ops`

Examples

```
data(exampleData)
longi.dat$DAP.diffs <- calcLagged(longi.dat$xDAP, operation ="-")
```

calcTimes

Calculates for a set of times, the time intervals after an origin time and the position of each within a time interval

Description

For the column specified in imageTimes, having converted it to `POSIXct` if not already converted, calculates for each value the number of `intervalUnits` between the time and the `startTime`. Then the number of `timePositions` within the `intervals` is calculated for the values in `imageTimes`. The function `difftimes` is used in doing the calculations, but the results are converted to `numeric`. For example `intervals` could correspond to the number of Days after Planting (DAP) and the `timePositions` to the hour within each day.

Usage

```
calcTimes(data, imageTimes = NULL, timeFormat = "%Y-%m-%d %H:%M",
          intervals = "Time.after.Planting..d.", startTime = NULL,
          intervalUnit = "days", timePositions = NULL)
```

Arguments

`data`	A `data.frame` containing any columns specified by `imageTimes`, `intervals` and `timePositions`.
`imageTimes`	A `character` giving the name of the column that contains the time that each cart was imaged. Note that in importing data into R, spaces and nonalphanumeric characters in names are converted to full stops. If `imageTimes` is `NULL` then no calculations are done.
`timeFormat`	A character giving the `POSIXct` format of characters containing times, in particular `imageTimes` and `startTime`. Note that if fractions of seconds are required `options(digits.secs)` must be used to set the number of decimal places and `timeFormat` must use `%OS` for seconds in `timeFormat`.
`intervals`	A `character` giving the name of the column in `data` containing, as a `numeric` or a `factor`, the calculated times after `startTime` to be plotted on the x-axis. It is given as the number of `intervalUnits` between the two times. If `startTime` is `NULL` then `intervals` is not calculated.
`startTime`	A `character` giving the time, in the `POSIXct` format specified by `timeFormat`, to be subtracted from `imageTimes` to caclualte `intervals`. For example, it might be the day of planting or treatment. If `startTime` is `NULL` then `intervals` is not calculated.
`intervalUnit`	A `character` giving the name of the unit in which the values of the intervals should be expressed. It must be one of `"secs"`, `"mins"`, `"hours"` or `"days"`.
`timePositions`	A `character` giving the name of the column in `data` containing, as a `numeric`, the value of the time position within an interval (for example, the time of imaging during the day expressed in hours plus a fraction of an hour). If `timePositions` is `NULL` then it is not calculated.

Value

A `data.frame`, being the unchaged data `data.frame` when `imageTimes` is `NULL` or containing either intervals and/or timePositions depending on which is not `NULL`.

Author(s)

Chris Brien

See Also

`as.POSIXct`, `imagetimesPlot`.

Examples

```
data(exampleData)
raw.dat <- calcTimes(data = raw.dat,
                     imageTimes = "Snapshot.Time.Stamp", timePositions = "Hour")
```

`cumulate` *Calculates the cumulative sum, ignoring the first element if exclude.1st is TRUE*

Description

Uses `cumsum` to calculate the cumulative sum, ignoring the first element if exclude.1st is `TRUE`.

Usage

```
cumulate(x, exclude.1st = FALSE, na.rm = FALSE, ...)
```

Arguments

`x`	A `vector` containing the values to be cumulated.
`exclude.1st`	A `logical` indicating whether or not the first value of the cumulative sum is to be `NA`.
`na.rm`	A `logical` indicating whether `NA` values should be stripped before the computation proceeds
`...`	allows passing of arguments to other functions; not used at present.

Value

A `vector` containing the cumulative sum.

Author(s)

Chris Brien

See Also

`cumsum`

Examples

```
data(exampleData)
PSA.cum <- cumulate(longi.dat$PSA)
```

`designFactors` *Adds the factors and covariates for a blocked, split-unit design*

Description

Add the following `factor`s and covariates to a date frame containing imaging data from the Plant Accelerator: Zone, xZone, SHZone, ZLane, ZMainunit, Subunit and xMainPosn. It checks that the numbers of levels of the `factor`s are consistent with the observed numbers of carts and observations.

Usage

```
designFactors(data, insertName = NULL, designfactorMethod = "LanePosition",
              nzones = 6, nlanesperzone = 4,
              nmainunitsperlane = 11, nsubunitspermain = 2)
```

Arguments

`data`	A `data.frame` to which are to be added the design `factor`s and covariates and which must contain the following columns: Smarthouse, Snapshot.ID.Tag, xDAP, and, if `designfactorMethod = "LanePosition"`, Lane and Position.
`insertName`	A `character` giving the name of the column in the `data.frame` after which the new `factor`s and covariates are to be inserted. If `NULL`, they are added after the last column.
`designfactorMethod`	A `character` giving the method to use to obtain the columns for the design `factor`s Zone, ZLane, Mainunit and Subunit. For `LanePosition`, it is assumed that (i) Lane can be divided into Zone and ZLane, each with nzones and nlanesperzone levels, respectively, and (ii) Position can be divided into Mainunit and Subunit, each with nmainunitsperlane and nmainunitsperlane levels, respectively. The `factor` SHZone is formed by combining Smarthouse and Zone and ZMainunit is formed by combining ZLane and Mainunit. For `StandardOrder`, the `factor`s Zone, ZLane, Mainunit, Subunit are generated in standard order, with the levels of Subunit changing for every observation and the levels of subsequent changing only after all combinations of the levels of the `factor`s to its right have been cycled through.
`nzones`	A `numeric` giving the number of zones in a smarthouse.

`nlanesperzone`	A `numeric` giving the number of lanes in each zone.
`nmainunitsperlane`	A `numeric` giving the number of mainunits in each lane.
`nsubunitspermain`	A `numeric` giving the number of subunits in a main plot.

Details

The `factor`s Zone, ZLane, ZMainunit and Subunit are derived for each Smarthouse based on the values of `nzones`, `nlanesperzone`, `nmainunitsperlane`, `nsubunitspermain`, Zone being the blocks in the split-unit design. Thus, the number of carts in each Smarthouse must be the product of these values and the number of observations must be the product of the numbers of smarthouse, carts and imagings for each cart. If this is not the case, it may be able to be achieved by including in `data` rows for extra observations that have values for the Snapshot.ID.Tag, Smarthouse, Lane, Position and Time.after.Planting..d. and the remaining columns for these rows have missing values (`NA`) Then SHZone is formed by combining Smarthouse and Zone and the covariates cZone, cMainPosn and cPosn calculated. The covariate cZone is calculated from Zone and cMainPosn is formed from the mean of cPosn for each main plot.

Value

A `data.frame` including the columns:

1. Smarthouse: `factor` with levels for the Smarthouse
2. Zone: `factor` dividing the Lanes into groups, usually of 4 lanes
3. cZone: numeric corresponding to Zone, centred by subtracting the mean of the unique positions
4. SHZone: `factor` for the combinations of Smarthouse and Zone
5. ZLane: `factor` for the lanes within a Zone
6. ZMainunit: `factor` for the main units within a Zone
7. Subunit: `factor` for the subunits
8. cMainPosn: numeric for the main-plot positions within a Lane, centred by subtracting the mean of the unique Positions
9. cPosn: numeric for the Positions within a Lane, centred by subtracting the mean of the unique Positions

Author(s)

Chris Brien

Examples

```
data(exampleData)
longi.dat <- prepImageData(data = raw.dat, smarthouse.lev = 1)
longi.dat <- designFactors(data = longi.dat, insertName = "Reps",
                           nzones = 1, nlanesperzone = 1, nmainunitsperlane = 10,
                           designfactorMethod="StandardOrder")
```

`exampleData` *A small data set to use in function examples*

Description

Imaging data for 20 of the plants that were imaged over 14 days from an experiment in a Smarthouse in the Plant Accelerator. Producing these files is illustrated in the `Rice` vignette and the data is used as a small example in the `growthPheno` manual.

Usage

```
data(exampleData)
```

Format

Three `data.frames`:

1. `raw.dat` (280 rows by 33 columns) that contains the imaging data for 20 plants by 14 imaging days as produced by the image processing software;
2. `longi.dat` (280 rows by 37 columns) that contains a modified version of the imaging data for the 20 plants by 14 imaging days in `raw.dat`;
3. `cart.dat` (20 rows by 14 columns) that contains data summarizing the growth features of the 20 plants produced from the data in `longi.dat`.

`getTimesSubset` *Forms a subset of* `responses` *in* `data` *that contains their values for the nominated times*

Description

Forms a subset of each of the `responses` in `data` that contains their values for the nominated times in a single column.

Usage

```
getTimesSubset(data, responses,
               individuals = "Snapshot.ID.Tag", times = "DAP",
               which.times, suffix = NULL, sep.suffix.times = ".",
               include.times = FALSE, include.individuals = FALSE)
```

Arguments

`data`	A `data.frame` containing the column from which the growth rates are to be calculated.
`responses`	A `character` giving the names of the columns in `data` whose values are to be subsetted.
`individuals`	A `character` giving the name of the column in `data` containing an identifier for each individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used to identify the subset and, if a `factor` or `character`, the values should be numerics stored as characters.
`which.times`	A `vector` giving the times that are to be selected.
`suffix`	A `character` giving the suffix to be appended to `responses` to form the names of the columns containing the subset.
`sep.suffix.times`	A `character` giving the separator to use in appending a suffix for times to a trait. For no separator, set to `""`.
`include.times`	A `logical` indicating whether or not to include the `times` in the result, the name in the result having the `suffix` with a separating full appended.
`include.individuals`	A `logical` indicating whether or not to include the `individuals` column in the result.

Value

A `data.frame` containing the subset of `responses` ordered by as many of the initial columns of `data` as are required to uniquely identify each row (see `order` for more information). The names of the columns for each of the `responses` and for `times` in the subset are the concatenation of their names in `data` and `suffix`, separated by a full stop.

Author(s)

Chris Brien

Examples

```
data(exampleData)
sPSALast <- getTimesSubset("sPSA", data = longi.dat, times = "DAP",
                           which.times = c(42), suffix = "last")
```

`growthPheno-deprecated`

Deprecated Functions in the Package growthPheno

Description

These functions have been renamed and deprecated in `growthPheno`:

1. getDates -> `getTimesSubset`
2. anomPlot -> `plotAnom`
3. corrPlot -> `plotCorrmatrix`
4. fitspline -> `smoothSpline`
5. imagetimesPlot -> `plotImagetimes`
6. intervalGRaverage -> `byIndv4Intvl_GRsAvg`
7. intervalGRdiff -> `byIndv4Intvl_GRsDiff`
8. intervalValueCalculate -> `byIndv4Intvl_ValueCalc`
9. intervalWUI -> `byIndv4Intvl_WaterUse`
10. longiPlot -> `plotProfiles`
11. longitudinalPrime -> `prepImageData`
12. plotLongitudinal -> `plotProfiles`
13. plotMedianDeviations -> `plotSmoothsMedianDevns`
14. probeDF -> `probeSmooths`
15. probeSmoothing -> `probeSmooths`
16. splitContGRdiff -> `byIndv4Times_GRsDiff`
17. splitSplines -> `byIndv4Times_SplinesGRs`
18. splitValueCalculate -> `byIndv4Intvl_ValueCalc`

Usage

```
getDates(...)
anomPlot(...)
corrPlot(...)
fitSpline(...)
imagetimesPlot(...)
intervalGRdiff(...)
intervalGRaverage(...)
intervalValueCalculate(...)
intervalWUI(...)
longiPlot(...)
longitudinalPrime(...)
plotLongitudinal(...)
plotMedianDeviations(...)
```

```
probeDF(...)
probeSmoothing(...)
splitContGRdiff(...)
splitSplines(...)
splitValueCalculate(...)
```

Arguments

`...`	absorbs arguments passed from the old functions of the style foo.bar().

Author(s)

Chris Brien

`GrowthRates` *Calculates growth rates (AGR, PGR, RGRdiff) between pairs of values in a vector*

Description

Calculates either the Absolute Growth Rate (AGR), Proportionate Growth Rate (PGR) or Relative Growth Rate (RGR) between pairs of time points, the second of which is `lag` positions before the first. in `x`.

Usage

```
AGRdiff(x, time.diffs, lag=1)
PGR(x, time.diffs, lag=1)
RGRdiff(x, time.diffs, lag=1)
```

Arguments

`x`	A `numeric` from which the growth rates are to be calculated.
`time.diffs`	a `numeric` giving the time differences between successive values in `x`.
`lag`	A `integer` specifying, for the second value in the pair to be operated on, the number positions it is ahead of the current value.

Details

The AGRdiff is calculated as the difference between a pair of values divided by the `time.diffs`. The PGR is calculated as the ratio of a value to a second value which is `lag` values ahead of the first in `x` and the ratio raised to the power of the reciprocal of `time.diffs`. The RGRdiff is calculated as the `log` of the PGR and so is equal to the difference between the logarithms of a pair of values divided by the `time.diffs`. The differences and ratios are obtained using `calcLagged` with `lag = 1`.

Value

A `numeric` containing the growth rates which is the same length as `x` and in which the first `lag` values `NA`.

Author(s)

Chris Brien

See Also

`byIndv4Intvl_GRsAvg`, `byIndv4Intvl_GRsDiff`, `byIndv4Times_GRsDiff`, `byIndv4Times_SplinesGRs`, `calcLagged`

Examples

```
data(exampleData)
longi.dat$PSA.AGR <- with(longi.dat, AGRdiff(PSA, time.diffs = DAP.diffs))
```

`importExcel` *Imports an Excel imaging file and allows some renaming of variables*

Description

Uses `readxl` to import a sheet of imaging data produced by the Lemna Tec Scanalyzer. Basically, the data consists of imaging data obtained from a set of pots or carts over time. There should be a column, which by default is called `Snapshot.ID.Tag`, containing a unique identifier for each cart and a column, which by default is labelled `Snapshot.Time.Stamp`, containing the time of imaging for each observation in a row of the sheet. Also, if `startTime` is not `NULL`, `calcTimes` is called to calculate, or recalculate if already present, `timeAfterStart` from `imageTimes` by subtracting a supplied `startTime`.

Using `cameraType`, `keepCameraType`, `labsCamerasViews` and `prefix2suffix`, some flexibility is provided for renaming the columns with imaging data. For example, if the column names are prefixed with 'RGB_SV1', 'RGB_SV2' or 'RGB_TV', the 'RGB_' can be removed and the 'SV1', 'SV2' or 'TV' become suffices.

Usage

```
importExcel(file, sheet="raw data", sep = ",",
            cartId = "Snapshot.ID.Tag",
            imageTimes = "Snapshot.Time.Stamp",
            timeAfterStart = "Time.after.Planting..d.",
            cameraType = "RGB", keepCameraType = FALSE,
            labsCamerasViews = NULL, prefix2suffix = TRUE,
            startTime = NULL,
            timeFormat = "%Y-%m-%d %H:%M",
            plotImagetimes = TRUE, ...)
```

Arguments

`file`	A `character` giving the path and name of the file containing the data.
`sheet`	A `character` giving the name of the sheet containing the data, that must include columns whose names are as specified by `cartId`, which uniquely indexes the carts in the experiment, and `imageTimes`, which reflects the time of the imaging from which a particular data value was obtained. It is also assumed that a column whose name is specified by `timeAfterStart` is in the sheet or that it will be calculated from `imageTimes` using the value of `startTime` supplied in the function call.
`sep`	A `character` giving the separator used in a `csv` file.
`cartId`	A `character` giving the name of the column that contains the unique Id for each cart. Note that in importing data into R, spaces and nonalphanumeric characters in names are converted to full stops.
`imageTimes`	A `character` giving the name of the column that contains the time that each cart was imaged. Note that in importing data into R, spaces and nonalphanumeric characters in names are converted to full stops.
`timeAfterStart`	A `character` giving the name of the column that contains or is to contain the difference between `imageTimes` and `startTime`. The function `calcTimes` is called to calculate the differences. For example, it might contain the number of days after planting. Note that in importing data into R, spaces and nonalphanumeric characters in names are converted to full stops.
`cameraType`	A `character` string nominating the abbreviation used for the cameraType. A warning will be given if no variable names include this cameraType.
`keepCameraType`	A `logical` specifying whether to retain the `cameraType` in the variables names. It will be the start of the prefix or suffix and separated from the remander of the prefix or suffix by an underscore (_).
`labsCamerasViews`	A named `character` whose elements are new labels for the camera-view combinations and the name of each element is the old label for the camera-view combination in the data being imported. If `labsCamerasViews` is `NULL`, all column names beginning with `cameraType` are classed as imaging variables and the unique prefixes amongst them determined. If no imaging variables are found then no changes are made. Note that if you want to include a recognisable `cameraType` in a camier-view label, it should be at the start of the the label in `labsCamerasViews` and separated from the rest of the label by an underscore (_).
`prefix2suffix`	A `logical` specifying whether the variables names with prefixed camera-view labels are to have those prefixes transferred to become suffices. The prefix is assumed to be all the characters up to the first full stop (.) in the variable name and must contain `cameraType` to be moved. It is generally assumed that the characters up to the first underscore (_) are the camera type and this is removed if `keepCameraType` is `FALSE`. If there is no underscore (_), the whole prefix is moved. If `labsCamerasViews` is `NULL`, all column names beginning with `cameraType` are classed as imaging variables and the unique prefixes amongst them determined. If no imaging variables are found then no changes are made.

`startTime`	A `character` giving the time of planting, in the `POSIXct` format timeFormat, to be subtracted from `imageTimes` in recalculating `timeAfterStart`. If `startTime` is `NULL` then `timeAfterStart` is not recalculated.
`timeFormat`	A `character` giving the `POSIXct` format of characters containing times, in particular `imageTimes` and `startTime`.
`plotImagetimes`	A `logical` indicating whether a plot of the imaging times against the recalculated `Time.After.Planting..d..` It aids in checking `Time.After.Planting..d.` and what occurred in imaging the plants.
`...`	allows for arguments to be passed to `plotImagetimes`. However, if `intervals` is passed an error will occur; use timeAfterStart instead.

Value

A `data.frame` containing the data.

Author(s)

Chris Brien

See Also

`as.POSIXct`, `calcTimes`, `plotImagetimes`

Examples

```
filename <- system.file("extdata/rawdata.xlsx", package = "growthPheno",
                        mustWork = TRUE)
raw.dat <- importExcel(file = filename,
                       startTime  = "2015-02-11 0:00 AM")


camview.labels <- c("SF0", "SL0", "SU0", "TV0")
names(camview.labels) <- c("RGB_Side_Far_0", "RGB_Side_Lower_0",
                           "RGB_Side_Upper_0", "RGB_TV_0")
filename <- system.file("extdata/raw19datarow.csv", package = "growthPheno",
                        mustWork = TRUE)
raw.19.dat <- suppressWarnings(importExcel(file = filename,
                                           cartId = "Snapshot.ID.Tags",
                                           startTime = "06/10/2017 0:00 AM",
                                           timeFormat = "%d/%m/%Y %H:M",
                                           labsCamerasViews = camview.labels,
                                           plotImagetimes = FALSE))
```

`intervalPVA.data.frame`

Selects a subset of variables using Principal Variable Analysis (PVA), based on the observed values within a specified time interval

Description

Principal Variable Analysis (PVA) (Cumming and Wooff, 2007) selects a subset from a set of the variables such that the variables in the subset are as uncorrelated as possible, in an effort to ensure that all aspects of the variation in the data are covered. Here, all observations in a specified time interval are used for calculation the correlations on which the selection is based.

Usage

```
## S3 method for class 'data.frame'
intervalPVA(obj, responses, times = "Days", start.time, end.time,
            nvarselect = NULL, p.variance = 1, include = NULL,
            plot = TRUE, ...)
```

Arguments

`obj`	A `data.frame` containing the columns of variables from which the selection is to be made.
`responses`	A `character` giving the names of the columns in `data` from which the variables are to be selected.
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used to identify the subset and, if a `factor` or `character`, the values should be numerics stored as characters.
`start.time`	A `numeric` giving the time, in terms of values in `times`, at which the time interval begins; observations at this time and up to and including `end.time` will be included.
`end.time`	A `numeric` giving the time, in terms of values in `times`, at the end of the interval; observations after this time will not be included.
`nvarselect`	A `numeric` specifying the number of variables to be selected, which includes those listed in `include`. If `nvarselect = 1`, as many variables are selected as is need to satisfy `p.variance`.
`p.variance`	A `numeric` specifying the minimum proportion of the variance that the selected variables must account for,
`include`	A `character` giving the names of the columns in `data` for the variables whose selection is mandatory.
`plot`	A `logical` indicating whether a plot of the cumulative proportion of the variance explained is to be produced.
`...`	allows passing of arguments to other functions.

Details

The variable that is most correlated with the other variables is selected first for inclusion. The partial correlation for each of the remaining variables, given the first selected variable, is calculated and the most correlated of these variables is selects for inclusion next. Then the partial correlations are adjust for the second included variables. This process is repeated until the specified criteria have been satisfied. The possibilities are to:

1. the default (`nvarselect = NULL` and `p.variance = 1`) select all variables in increasing order of amount of information they provide;
2. select exactly `nvarselect` variables;
3. select just enough variables, up to a maximum of `nvarselect` variables, to explain at least `p.variance`*100 per cent of the total variance.

Value

A `data.frame` giving the results of the variable selection. It will contain the columns `Variable`, `Selected`, `h.partial`, `Added.Propn` and `Cumulative.Propn`.

Author(s)

Chris Brien

References

Cumming, J. A. and D. A. Wooff (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

`PVA`, `rcontrib`

Examples

```
data(exampleData)
longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1)
longi.dat <- within(longi.dat,
                    {
                      Max.Height <- pmax(Max.Dist.Above.Horizon.Line.SV1,
                                         Max.Dist.Above.Horizon.Line.SV2)
                      Density <- PSA/Max.Height
                      PSA.SV = (PSA.SV1 + PSA.SV2) / 2
                      Image.Biomass = PSA.SV * (PSA.TV^0.5)
                      Centre.Mass <- (Center.Of.Mass.Y.SV1 + Center.Of.Mass.Y.SV2) / 2
                      Compactness.SV = (Compactness.SV1 + Compactness.SV2) / 2
                    })
responses <- c("PSA","PSA.SV","PSA.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
results <-  intervalPVA(longi.dat, responses, times = "DAP",
                        start.time = "31", end.time = "31",
                        p.variance=0.9, plot = FALSE)
```

`is.smooths.frame` *Tests whether an object is of class smooths.frame*

Description

A single-line `function` that tests whether an object is of `class` `smooths.frame`.

Usage

```
is.smooths.frame(object)
```

Arguments

`object`	An `object` to be tested.

Value

A `logical`.

Author(s)

Chris Brien

See Also

`validSmoothsFrame`, `as.smooths.frame`

Examples

```
dat <- read.table(header = TRUE, text = "
Type TunePar TuneVal Tuning Method       ID  DAP   PSA      sPSA
NCSS      df       4   df-4 direct 045451-C   28 57.446 51.18456
NCSS      df       4   df-4 direct 045451-C   30 89.306 87.67343
NCSS      df       7   df-7 direct 045451-C   28 57.446 57.01589
NCSS      df       7   df-7 direct 045451-C   30 89.306 87.01316
")
dat[1:7] <- lapply(dat[1:7], factor)
dat <- as.smooths.frame(dat, individuals = "ID", times = "DAP")
is.smooths.frame(dat)
validSmoothsFrame(dat)
```

---

plotAnom *Identifies anomalous individuals and produces profile plots without them and with just them*

Description

Uses `byIndv4Intvl_ValueCalc` and the function `anom` to identify anomalous individuals in longitudinal data. The user can elect to print the anomalous individuals, a profile plot without the anomalous individuals and/or a profile plot with only the anomalous individuals. The plots are produced using `ggplot`. The plot can be facettd so that a grid of plots is produced.

Usage

```
plotAnom(data, response="sPSA",
         individuals="Snapshot.ID.Tag",
         times = "DAP", x = NULL,
         breaks.spacing.x = -2, angle.x = 0,
         vertical.line=NULL,
         groupsFactor=NULL, lower=NULL, upper=NULL,
         start.time=NULL, end.time=NULL,
         suffix.interval=NULL,
         columns.retained=c("Snapshot.ID.Tag", "Smarthouse", "Lane",
                            "Position", "Treatment.1", "Genotype.ID"),
         whichPrint=c("anomalous","innerPlot","outerPlot"), na.rm=TRUE, ...)
```

Arguments

`data`	A `data.frame` containing the data to be tested and plotted.
`response`	A `character` specifying the response variable that is to be tested and plotted on the y-axis.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. If not a `numeric`, it will be converted to a `numeric` and used to provide the values to be plotted on the x-axis. If a `factor` or `character`, the values should be numerics stored as characters.
`x`	A `character` specifying a variable, or a function of variables, to be plotted on the x-axis. If `NULL`, it will be set to the value of `times`, which it can be assumed will be converted to a `numeric`.
`breaks.spacing.x`	A `numeric` whose absolute values specifies the distance between major breaks for the x-axis in a sequence beginning with the minimum x value and continuing up to the maximum x value. If it is negative, the breaks that do not have x values in `data` will be omitted. Minor breaks will be at half major break value or, if

	these do not correspond to x-values in `data` when `breaks.spacing.x` is negative, have a spacing of one. Thus, when `breaks.spacing.x` is negative, grid lines will only be included for x-values that occur in `data`. These settings can be overwritten by supplying, in `ggplotFuncs`, a `scale_x_continuous` function from `ggplot2`.
`angle.x`	A `numeric` between 0 and 360 that gives the angle of the x-axis text to the x-axis. It can also be set by supplying, in `ggplotFuncs`, a `theme` function from `ggplot2`.
`vertical.line`	A `numeric` giving position on the x-axis at which a vertical line is to be drawn. If `NULL`, no line is drawn.
`groupsFactor`	A `factor` giving the name of a `factor` that defines groups of individuals between which the test for anomalous individuals can be varied by setting values for one or more of `lower`, `upper`, `start.time` and `end.time` to be `NULL`, a single value or a set of values whose number equals the number of levels of `groupsFactor`. If `NULL` or only a dingle value is supplied, the test is the same for all individuals.
`lower`	A `numeric` such that values in `response` below it are considered to be anomalous. If `NULL`, there is no testing for values below the lower bound.
`upper`	A `numeric` such that values in `response` above it are considered to be anomalous. If `NULL`, there is no testing for values above the upper bound.
`start.time`	A `numeric` giving the start of the time interval, in terms of a level of `times`, during which testing for anomalous values is to occur. If `NULL`, the interval will start with the first observation.
`end.time`	A `numeric` giving the end of the time interval, in terms of a level of `times`, during which testing for anomalous values is to occur. If `NULL`, the interval will end with the last observation.
`suffix.interval`	A `character` giving the suffix to be appended to `response` to form the name of the column containing the calculated values. If it is `NULL` then nothing will be appended.
`columns.retained`	A `character` giving the names of the columns in `data` that are to be retained in the `data.frame` of anomalous individuals.
`whichPrint`	A `character` indicating what is to be printed. If `anomalous` is included, the `columns.retained` are printed for the anomalous individuals.
`na.rm`	A `logical` indicating whether `NA` values should be stripped before the testing proceeds.
`...`	allows for arguments to be passed to `plotProfiles`.

Value

A `list` with three components:

1. `data`, a data frame resulting from the `merge` of `data` and the `logical` identifying whether or not an individual is anomalous;

2. `innerPlot`, an object of class `ggplot` storing the profile plot of the individuals that are not anomalous;
3. `outerPlot`, an object of class `ggplot` storing the profile plot of only the individuals that are anomalous.

The name of the column indicating anomalous individuals will be result of concatenating the `response`, `anom` and, if it is not `NULL`, `suffix.interval`, each separated by a full stop. The `ggplot` objects can be plotted using `print` and can be modified by adding `ggplot` functions before printing. If there are no observations to plot, `NULL` will be returned for the plot.

Author(s)

Chris Brien

See Also

`anom`, `byIndv4Intvl_ValueCalc`, `ggplot2`.

Examples

```
data(exampleData)
anomalous <- plotAnom(longi.dat, response="sPSA.AGR",
                      times = "xDAP",
                      lower=2.5, start.time=40,
                      vertical.line=29,
                      breaks.spacing.x = 2,
                      whichPrint=c("innerPlot"),
                      y.title="sPSA AGR")
```

`plotCorrmatrix` *Calculates and plots correlation matrices for a set of responses*

Description

Having calculated the correlations a heat map indicating the magnitude of the correlations is produced using `ggplot`. In this heat map, the darker the red in a cell then the closer the correlation is to -1, while the deeper the blue in the cell, then the closer the correlation is to 1. Matrix plots of all pairwise combinations of the variables can be produced that includes the values of the the correlation coefficients. If `pairs.sets` is set, a matrix plot, along with the values of the correlation coefficients, is produced for each of the `pair.sets`. That is, the argument `pairs.sets` can be used to restrict the pairs in a matrix plot to those combinations within each set.

Usage

```
plotCorrmatrix(data, responses, which.plots = c("heatmap","matrixplots"),
               title = NULL, labels = NULL, labelSize = 4, pairs.sets = NULL,
               show.sig = TRUE, axis.text.size = 20, ggplotFuncs = NULL,
               printPlot = TRUE, ...)
```

Arguments

`data`	A `data.frame` containing the columns of variables to be correlated.
`responses`	A `character` giving the names of the columns in `data` containing the variables to be correlated.
`which.plots`	A `character` specifying the plots of the correlations to be produced. The possibilities are one or both of `heatmap` and `matrixplots`.
`title`	Title for the plots.
`labels`	A `character` specifying the labels to be used in the plots. If `labels` is `NULL`, `responses` is used for the `labels`.
`labelSize`	A `numeric` giving the size of the `labels` in the `matrixplots`.
`pairs.sets`	A `list` each of whose components is a `numeric` giving the position of the variable names in `responses` that are to be included in the set. All pairs of variables in this pairs.set will be included in `matrixplots`.
`show.sig`	A `logical` indicating whetherto give asterisks on the `heatmap` and `matrixplots` that indicate that the correlations are significantly different from zero.
`axis.text.size`	A `numeric` giving the size of the `labels` on the axes of the `heatmap`.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object.
`printPlot`	A `logical` indicating whether or not to print the plots.
`...`	allows passing of arguments to other functions; not used at present.

Details

The correlations and their p-values are producced using `rcorr` from the `Hmisc` package. The `heatmap` is produced using `ggplot` and the matrixplots are produced using `GGally`.

Value

A `list` object that has components `heatmap` and `matrixplots`. The component `heatmap` will contain the heatmap plot, if produced, as an object of class `"ggplot"`, which can be plotted using `print`; otherwise `NULL` is returned. Similarly, if not `NULL`, the component `matrixplots` will contain a list with one or more components, depending on the setting of `pair.sets`, each of which is a scatterplot matrix stored as an object of class `"ggmatrix"`.

Author(s)

Chris Brien

See Also

`rcorr`, `GGally`, `ggplot`.

Examples

```
data(exampleData)
longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1)
longi.dat <- within(longi.dat,
                    {
                      Max.Height <- pmax(Max.Dist.Above.Horizon.Line.SV1,
                                         Max.Dist.Above.Horizon.Line.SV2)
                      Density <- PSA/Max.Height
                      PSA.SV = (PSA.SV1 + PSA.SV2) / 2
                      Image.Biomass = PSA.SV * (PSA.TV^0.5)
                      Centre.Mass <- (Center.Of.Mass.Y.SV1 + Center.Of.Mass.Y.SV2) / 2
                      Compactness.SV = (Compactness.SV1 + Compactness.SV2) / 2
                    })
responses <- c("PSA","PSA.SV","PSA.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
plotCorrmatrix(longi.dat, responses, pairs.sets=list(c(1:4),c(5:7)))
```

`plotDeviationsBoxes` *Produces boxplots of the deviations of the observed values from the smoothed values over values of x.*

Description

Produces boxplots of the deviations of the observed values from the smoothed values over values of x.

Usage

```
plotDeviationsBoxes(data, observed, smoothed, x.factor,
                    x.title = NULL, y.titles = NULL,
                    facet.x = ".", facet.y = ".",
                    facet.labeller = NULL,
                    facet.scales = "fixed",
                    angle.x = 0,
                    deviations.plots = "absolute",
                    ggplotFuncs = NULL, printPlot = TRUE, ...)
```

Arguments

`data`	A `data.frame` containing the observed and smoothed values from which the deviations are to be computed.
`observed`	A `character` specifying the response variable for which the observed values are supplied.
`smoothed`	A `character` specifying the smoothed response variable, corresponding to `observed`, for which values are supplied.

`x.factor`	A `character` giving the `factor` to be plotted on the x-axis.
`x.title`	Title for the x-axis. If `NULL` then set to `x`.
`y.titles`	A `character` giving the titles for the y-axis, one for each plot specified `deviations.plots`.
`facet.x`	A `data.frame` giving the variable to be used to form subsets to be plotted in separate columns of plots. Use `"."` if a split into columns is not wanted. For `which.plots` set to `methodcompare` or `dfcompare` `facet.x.pf` is ignored.
`facet.y`	A `data.frame` giving the variable to be used to form subsets to be plotted in separate rows of plots. Use `"."` if a split into columns is not wanted.
`facet.labeller`	A `ggplot` `function` for labelling the facets of a plot produced using the `ggplot` function. For more information see `ggplot`.
`facet.scales`	A `character` specifying whether the scales are shared across all facets of a plot ("fixed"), or do they vary across rows (the default, "free_x"), columns ("free_y"), or both rows and columns ("free")?
`angle.x`	A `numeric` between 0 and 360 that gives the angle of the x-axis text to the x-axis. It can also be set by supplying, in `ggplotFuncs`, a `theme` function from `ggplot2`.
`deviations.plots`	A `character` specifying whether `absolute` and/or `relative` deviations are to be plotted.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object for plotting.
`printPlot`	A `logical` indicating whether or not to print the plots.
`...`	allows passing of arguments to `ggplot`.

Value

A list whose components are named `absolute` and `relative`; a component will contain an object of class `"ggplot"` when the plot has been requested using the `deviations.plots` argument and a `NULL` otherwise. The objects can be plotted using `print`.

Author(s)

Chris Brien

See Also

`plotSmoothsMedianDevns`, `probeSmooths`, `ggplot`.

Examples

```
data(exampleData)

plotDeviationsBoxes(longi.dat, observed = "PSA", smoothed = "sPSA",
                    x.factor="DAP", facet.x.pf = ".", facet.y= ".", df =5)
```

`plotImagetimes` *Plots the position of a time within an interval against the interval for each cart*

Description

Uses `ggplot` to produce a plot of the time position within an interval against the interval. For example, one might plot the hour of the day carts are imaged against the days after planting (or some other number of days after an event). A line is produced for each value of `groupVariable` and the colour is varied according to the value of the `colourVariable`. Each `Smarthouse` is plotted separately. It aids in checking whether delays occurred in imaging the plants.

Usage

```
plotImagetimes(data, intervals = "Time.after.Planting..d.", timePositions = "Hour",
               groupVariable = "Snapshot.ID.Tag", colourVariable = "Lane",
               ggplotFuncs = NULL, printPlot = TRUE)
```

Arguments

`data`	A `data.frame` containing any columns specified by `intervals`, `timePositions`, `groupVariable` and `colourVariable`.
`intervals`	A `character` giving the name of the column in `data` containing, as a `numeric` or a `factor`, the calculated times to be plotted on the x-axis. For example, it could be the days after planting or treatment.
`timePositions`	A `character` giving the name of the column in `data` containing, as a `numeric`, the value of the time position within an interval (for example, the time of imaging during the day expressed in hours plus a fraction of an hour).
`groupVariable`	A `character` giving the name of the column in `data` containing the variable to be used to group the plotting.
`colourVariable`	A `character` giving the name of the column in `data` containing the variable to be used to colour the plotting.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object.
`printPlot`	A `logical` indicating whether or not to print the plot.

Value

An object of class "`ggplot`", which can be plotted using `print`.

Author(s)

Chris Brien

See Also

`ggplot`, `calcTimes`.

Examples

```
data(exampleData)
library(ggplot2)
longi.dat <- calcTimes(longi.dat, imageTimes = "Snapshot.Time.Stamp",
                       timePositions = "Hour")
plotImagetimes(data = longi.dat, intervals = "DAP", timePositions = "Hour",
               ggplotFuncs=list(scale_colour_gradient(low="grey20", high="black"),
                                geom_line(aes(group=Snapshot.ID.Tag, colour=Lane))))
```

`plotProfiles` *Produces profile plots of longitudinal data for a set of individuals*

Description

Produce profile plots of longitudinal data for a response using `ggplot`. A line is drawn for the data for each `individual` and the plot can be faceted so that a grid of plots is produced. For each facet a line for the medians over time can be added, along with the vaue of the outer whiskers (median +/- 1.5 * IQR).

Usage

```
plotProfiles(data, response = "PSA",
             individuals = "Snapshot.ID.Tag", times = "DAP",
             x = NULL, title = NULL,
             x.title = "DAP", y.title = "PSA (kpixels)",
             facet.x = ".", facet.y = ".",
             labeller = NULL, scales = "fixed",
             breaks.spacing.x = -2, angle.x = 0,
             colour = "black",
             colour.column = NULL, colour.values = NULL,
             alpha = 0.1, addMediansWhiskers = FALSE,
             ggplotFuncs = NULL,
             printPlot = TRUE)
```

Arguments

`data`	A `data.frame` containing the data to be plotted.
`response`	A `character` specifying the response variable that is to be plotted on the y-axis.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).

`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. If not a `numeric`, it will be converted to a `numeric` and used to provide the values to be plotted on the x-axis. If a `factor` or `character`, the values should be numerics stored as characters.
`x`	A `character` specifying a variable, or a function of variables, to be plotted on the x-axis. If `NULL`, it will be set to the value of `times`, which it can be assumed will be converted to a `numeric`.
`x.title`	Title for the x-axis.
`y.title`	Title for the y-axis.
`title`	Title for the plot.
`facet.x`	A `data.frame` giving the variable to be used to form subsets to be plotted in separate columns of plots. Use `"."` if a split into columns is not wanted.
`facet.y`	A `data.frame` giving the variable to be used to form subsets to be plotted in separate rows of plots. Use `"."` if a split into rows is not wanted.
`labeller`	A `ggplot` `function` for labelling the facets of a plot produced using the `ggplot` function. For more information see `ggplot`.
`scales`	A `character` specifying whether the scales are shared across all facets of a plot (the default, "fixed"), or do they vary across rows ("free_x"), columns ("free_y"), or both rows and columns ("free")?
`breaks.spacing.x`	A `numeric` whose absolute values specifies the distance between major breaks for the x-axis in a sequence beginning with the minimum x value and continuing up to the maximum x value. If it is negative, the breaks that do not have x values in `data` will be omitted. Minor breaks will be at half major break value or, if these do not correspond to x-values in `data` when `breaks.spacing.x` is negative, have a spacing of one. Thus, when `breaks.spacing.x` is negative, grid lines will only be included for x-values that occur in `data`. These settings can be overwritten by supplying, in `ggplotFuncs`, a `scale_x_continuous` function from `ggplot2`.
`angle.x`	A `numeric` between 0 and 360 that gives the angle of the x-axis text to the x-axis. It can also be set by supplying, in `ggplotFuncs`, a `theme` function from `ggplot2`.
`colour`	A `character` specifying a single colour to use in drawing the lines for the profiles. If colouring according to the values of a variable is required then use `colour.column`.
`colour.column`	A `character` giving the name of a column in `data` over whose values the colours of the lines are to be varied. The colours can be specified using `colour.values`.
`colour.values`	A `character` vector specifying the values of the colours to use in drawing the lines for the profiles. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order (usually alphabetical) with the limits of the scale.
`alpha`	A `numeric` specifying the degrees of transparency to be used in plotting. It is a ratio in which the denominator specifies the number of points (or lines) that must be overplotted to give a solid cover.

`addMediansWhiskers`	A `logical` indicating whether plots over time of the medians and outer whiskers are to be added to the plot. The outer whiskers are related to the whiskers on a box-and-whisker and are defined as the median plus (and minus) 1.5 times the interquartile range (IQR). Points lying outside the whiskers are considered to be potential outliers.
`ggplotFuncs`	A `list`, each element of which contains the results of evaluating a `ggplot` function. It is created by calling the `list` function with a `ggplot` function call for each element. These functions are applied in creating the `ggplot` object.
`printPlot`	A `logical` indicating whether or not to print the plot.

Value

An object of class "`ggplot`", which can be plotted using `print`.

Author(s)

Chris Brien

See Also

`ggplot`, `labeller`.

Examples

```
data(exampleData)
plotProfiles(data = longi.dat, response = "sPSA", times = "DAP")

plt <- plotProfiles(data = longi.dat, response = "sPSA",
                    y.title = "sPSA (kpixels)",
                    facet.x = "Treatment.1", facet.y = "Smarthouse",
                    breaks.spacing.x = 2,
                    printPlot=FALSE)
plt <- plt + ggplot2::geom_vline(xintercept=29, linetype="longdash", linewidth=1) +
             ggplot2::scale_y_continuous(limits=c(0,750))
print(plt)

plotProfiles(data = longi.dat, response = "sPSA", times = "DAP",
             x.title = "DAP", y.title = "sPSA (kpixels)",
             facet.x = "Treatment.1", facet.y = "Smarthouse",
                 ggplotFuncs = list(ggplot2::geom_vline(xintercept=29,
                                                        linetype="longdash",
                                                        size=1),
                                    ggplot2::scale_x_continuous(breaks=seq(28, 42,
                                                                           by=2)),
                                    ggplot2::scale_y_continuous(limits=c(0,750))))
```

`plotSmoothsComparison` *Plots several sets of smoothed values for a response, possibly along with growth rates and optionally including the unsmoothed values, as well as deviations boxplots.*

Description

Plots the smoothed values for an observed `response` and, optionally, the unsmoothed observed `response` using `plotProfiles`. Depending on the setting of `trait.types` (`response`, `AGR` or `RGR`), the computed traits of the Absolute Growth Rates (AGR) and/or the Relative Growth Rates (RGR) are plotted. This function will also calculate and produce, using `plotDeviationsBoxes`, boxplots of the deviations of the supplied smoothed values from the observed response values for the traits and for combinations of the different smoothing parameters and for subsets of non-smoothing-`factor` combinations. The observed and smoothed values are supplied in long format i.e. with the values for each set of smoothing parameters stacked one under the other in the supplied `smooths.frame`. Such data can be generated using `probeSmooths`; to prevent `probeSmooths` producing the plots, which it is does using `plotSmoothsComparison`, `plotDeviationsBoxes` and `plotSmoothsMedianDevns`, set `which.plots` to `none`. The smoothing parameters include `spline.types`, `df`, `lambdas` and `smoothing.methods` (see `probeSmooths`).

Multiple plots, possibly each having multiple facets, are produced using `ggplot2`. The layout of these plots is controlled via the arguments `plots.by`, `facet.x` and `facet.y`. The basic principle is that the number of levels combinations of the smoothing-parameter `factor`s `Type`, `TunePar`, `TuneVal`, `Tuning` (the combination of (`TunePar` and `TuneVal`), and `Method` that are included in `plots.by`, `facet.x` and `facet.y` must be the same as those covered by the combinations of the values supplied to `spline.types`, `df`, `lambdas` and `Method` and incorporated into the `smooths.frame` input to `plotSmoothsComparison` via the `data` argument. This ensures that smooths from different parameter sets are not pooled into the same plot. The `factor`s other than the smoothing-parameter `factor`s can be supplied to the `plots.by` and `facet` arguments.

The following profiles plots can be produced: (i) separate plots of the smoothed traits for each combination of the smoothing parameters (include `Type`, `Tuning` and `Method` in `plots.by`); (ii) as for (i), with the corresponding plot for the unsmoothed trait preceeding the plots for the smoothed trait (also set `include.raw` to `alone`); (iii) profiles plots that compare a smoothed trait for all combinations of the values of the smoothing parameters, arranging the plots side-by-side or one above the other (include `Type`, `Tuning` and `Method` in `facet.x` and/or `facet.y` - to include the unsmoothed trait set `include.raw` to one of `facet.x` or `facet.y`; (iv) as for (iii), except that separate plots are produced for each combination of the levels of the `factor`s in `plot.by` and each plot compares the smoothed traits for the smoothing-parameter `factor`s included in `facet.x` and/or `facet.y` (set both `plots.by` and one or more of `facet.x` and `facet.y`).

Usage

```
plotSmoothsComparison(data, response, response.smoothed = NULL,
                      individuals = "Snapshot.ID.Tag", times = "DAP",
                      trait.types = c("response", "AGR", "RGR"),
                      x.title = NULL, y.titles = NULL,
                      profile.plot.args =
```

```
                          args4profile_plot(plots.by = NULL,
                                            facet.x = ".", facet.y = ".",
                                            include.raw = "no"),
                        printPlot = TRUE, ...)
```

Arguments

`data`	A `smooths.frame`, such as is produced by `probeSmooths` and that contains the data resulting from smoothing a response over time for a set of `individuals`, the data being arranged in long format both with respect to the times and the smoothing-parameter values used in the smoothing. That is, each response occupies a single column. The unsmoothed `response` and the `response.smoothed` are to be plotted for different sets of values for the smoothing parameters. The `smooths.frame` must include the columns `Type`, `TunePar`, `TuneVal`, `Tuning` and `Method`, and the columns nominated using the arguments `individuals`, `times`, `plots.by`, `facet.x`, `facet.y`, `response`, `response.smoothed`, and, if requested, the AGR and the RGR of the `response` and `response.smoothed`. The names of the growth rates should be formed from `response` and `response.smoothed` by adding `.AGR` and `.RGR` to both of them.
`response`	A `character` specifying the response variable for which the observed values are supplied.
`response.smoothed`	A `character` specifying the name of the column containing the values of the smoothed response variable, corresponding to `response` and obtained for the combinations of `smoothing.methods` and `df`, usually using smoothing splines. If `response.smoothed` is `NULL`, then `response.smoothed` is set to the `response` to which is added the prefix `s`.
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used to provide the values to be plotted on the x-axis. If a `factor` or `character`, the values should be numerics stored as characters.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`trait.types`	A `character` giving the `trait.types` that are to be plotted when `which.plots` is `profiles`. Irrespective of the setting of `get.rates`, the nominated traits are plotted. If `all`, each of `response`, `AGR` and `RGR` is plotted.
`x.title`	Title for the x-axis, used for all plots. If `NULL` then set to `times`.
`y.titles`	A `character` giving the titles for the y-axis, one for each trait specified by `trait.types` and used for all plots. If `NULL`, then set to the traits derived for `response` from `trait.types`.
`profile.plot.args`	A named `list` that is most easily generated using `args4profile_plot`, it documenting the options available for varying profile plots and boxplots. *Note that if* `args4profile_plot` *is to be called to change from the default settings given in the default* `plotSmoothsComparison` *call and some of those settings are to be retained, then the arguments whose settings are to be retained must*

	also be included in the call to `args4profile_plot`*; be aware that if you call* `args4profile_plot`*, then the defaults for this call are those for* `args4profile_plot`*,* ***NOT*** *the call to* `args4profile_plot` *shown as the default for* `plotSmoothsComparison`.
`printPlot`	A `logical` indicating whether or not to print any plots.
`...`	allows passing of arguments to `plotProfiles`.

Value

A multilevel `list` that contains the `ggplot` objects for the plots produced. The first-level `list` has a component for each `trait.types` and each of these is a second-level `list` that contains the trait profile plots and for a `trait`. It may contain components labelled `Unsmoothed`, `all` or for one of the levels of the `factors` in `plots.by`; each of these third-level `ists` contains a `ggplot` object that can be plotted using `print`.

Author(s)

Chris Brien

See Also

`traitSmooth`, `probeSmooths`, `args4profile_plot`, `plotDeviationsBoxes`, `plotSmoothsMedianDevns`, `ggplot2`.

Examples

```
data(exampleData)
vline <- list(ggplot2::geom_vline(xintercept=29, linetype="longdash", size=1))
traits <- probeSmooths(data = longi.dat,
                       response = "PSA", response.smoothed = "sPSA",
                       times = "DAP",
                       #only df is changed from the probeSmooth default
                       smoothing.args =
                         args4smoothing(smoothing.methods = "direct",
                                        spline.types = "NCSS",
                                        df = c(4,7), lambdas = NULL),
                       which.plots = "none")
plotSmoothsComparison(data = traits,
                      response = "PSA", response.smoothed = "sPSA",
                      times = "DAP", x.title = "DAP",
                      #only facet.x is changed from the probeSmooth default
                      profile.plot.args =
                        args4profile_plot(plots.by = NULL,
                                          facet.x = "Tuning", facet.y = ".",
                                          include.raw = "no",
                                          ggplotFuncs = vline))
```

---

plotSmoothsDevnBoxplots

Produces boxplots for several sets of deviations of the smoothed values from a response, possibly along with growth rates.

Description

Calculates and produces, using `plotDeviationsBoxes`, boxplots of the deviations of the supplied smoothed values from the observed response values for the traits and for combinations of the different smoothing parameters and for subsets of non-smoothing-`factor` combinations. Which traits are plotted is controlled by `trait.types` and may include the (`response`and the computed traits of the Absolute Growth Rates (AGR) and/or the Relative Growth Rates (RGR). The observed and smoothed values are supplied in long format i.e. with the values for each set of smoothing parameters stacked one under the other in the supplied `smooths.frame`. Such data can be generated using `probeSmooths`.

Multiple plots, possibly each having multiple facets, are produced using `ggplot2`. The layout of these plots is controlled via the arguments `plots.by`, `facet.x` and `facet.y`. The basic principle is that the number of levels combinations of the smoothing-parameter `factor`s `Type`, `TunePar`, `TuneVal`, `Tuning` (the combination of (`TunePar` and `TuneVal`), and `Method` that are included in `plots.by`, `facet.x` and `facet.y` must be the same as those covered by the combinations of the values incorporated into the `smooths.frame` input to `plotSmoothsDevnBoxplots` via the `data` argument. This ensures that smooths from different parameter sets are not pooled into the same plot. The `factor`s other than the smoothing-parameter `factor`s can be supplied to the `plots.by` and `facet` arguments.

Usage

```
plotSmoothsDevnBoxplots(data, response, response.smoothed = NULL,
                        individuals = "Snapshot.ID.Tag", times = "DAP",
                        trait.types = c("response", "AGR", "RGR"),
                        which.plots =  "absolute.boxplots",
                        x.title = NULL, y.titles = NULL,
                        devnboxes.plot.args =
                          args4devnboxes_plot(plots.by = NULL,
                                              facet.x = ".", facet.y = "."),
                        printPlot = TRUE, ...)
```

Arguments

`data`	A `smooths.frame`, such as is produced by `probeSmooths` and that contains the data resulting from smoothing a response over time for a set of `individuals`, the data being arranged in long format both with respect to the times and the smoothing-parameter values used in the smoothing. That is, each response occupies a single column. The unsmoothed `response` and the `response.smoothed` are to be plotted for different sets of values for the smoothing parameters. The `smooths.frame` must include the columns `Type`, `TunePar`, `TuneVal`, `Tuning` and `Method`, and the columns nominated using the arguments `individuals`,

`times`, `plots.by`, `facet.x`, `facet.y`, `response`, `response.smoothed`, and, if requested, the AGR and the RGR of the `response` and `response.smoothed`. The names of the growth rates should be formed from `response` and `response.smoothed` by adding `.AGR` and `.RGR` to both of them.

`response`	A `character` specifying the response variable for which the observed values are supplied.
`response.smoothed`	A `character` specifying the name of the column containing the values of the smoothed response variable, corresponding to `response` and obtained for the combinations of `smoothing.methods` and `df`, usually using smoothing splines. If `response.smoothed` is `NULL`, then `response.smoothed` is set to the `response` to which is added the prefix `s`.
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used to provide the values to be plotted on the x-axis. If a `factor` or `character`, the values should be numerics stored as characters.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`trait.types`	A `character` giving the `trait.types` that are to be plotted. If `all`, each of `response`, `AGR` and `RGR` is plotted.
`which.plots`	A `logical` indicating which plots are to be produced. The options are either `none` or `absolute.deviations` and/or `relative.deviations`. Boxplots of the absolute deviations are specified by `absolute.boxplots`, the absolute deviations being the values of a trait minus their smoothed values (observed - smoothed). Boxplots of the relative deviations are specified by `relative.boxplots`, the relative deviations being the absolute deviations divided by the smoothed values of the trait.
`x.title`	Title for the x-axis, used for all plots. If `NULL` then set to `times`.
`y.titles`	A `character` giving the titles for the y-axis, one for each trait specified by `trait.types` and used for all plots. If `NULL`, then set to the traits derived for `response` from `trait.types`.
`devnboxes.plot.args`	A named `list` that is most easily generated using `args4devnboxes_plot`, it documenting the options available for varying the boxplots. *Note that if* `args4devnboxes_plot` *is to be called to change from the default settings given in the default* `probeSmooths` *call and some of those settings are to be retained, then the arguments whose settings are to be retained must also be included in the call to* `args4devnboxes_plot`*; be aware that if you call* `args4devnboxes_plot`*, then the defaults for this call are those for* `args4devnboxes_plot`*, **NOT** the call to* `args4devnboxes_plot` *shown as the default for* `probeSmooths`.
`printPlot`	A `logical` indicating whether or not to print any plots.
`...`	allows passing of arguments to `plotProfiles`.

Value

A multilevel `list` that contains the `ggplot` objects for the plots produced. The first-level `list` has a component for each `trait.types` and each of these is a second-level `list` with contains the deviations boxplots for a `response`. Each plot is in an object of class `ggplot`, which can be plotted using `print`.

Author(s)

Chris Brien

See Also

`traitSmooth`, `probeSmooths`, `args4profile_plot`, `plotDeviationsBoxes`, `plotSmoothsMedianDevns`, `ggplot`.

Examples

```
data(exampleData)
traits <- probeSmooths(data = longi.dat,
                       response = "PSA", response.smoothed = "sPSA",
                       times = "DAP",
                       #only df is changed from the probeSmooth default
                       smoothing.args =
                         args4smoothing(smoothing.methods = "direct",
                                        spline.types = "NCSS",
                                        df = c(4,7), lambdas = NULL),
                       which.plots = "none")
plotSmoothsDevnBoxplots(data = traits,
                        response = "PSA", response.smoothed = "sPSA",
                        times = "DAP", x.title = "DAP",
                        #only facet.x is changed from the probeSmooth default
                        devnboxes.plot.args =
                          args4devnboxes_plot(plots.by = NULL,
                                              facet.x = "Tuning", facet.y = "."))
```

`plotSmoothsMedianDevns`

Calculates and plots the medians of the deviations from the observed values for several sets of smoothed values stored in a `data.frame` *in long format.*

Description

Calculates and plots the medians of the deviations of the supplied smoothed values from the supplied observed values for traits and combinations of different smoothing parameters, possibly for subsets of non-smoothing-`factor` combinations. The observed and smoothed values are supplied in long format i.e. with the values for each set of smoothing parameters stacked one under the other in the supplied `data.frame`. Such data can be generated using `probeSmooths`;

to prevent probeSmooths producing the plots, which it is does using plotSmoothsComparison, plotDeviationsBoxes and `plotSmoothsMedianDevns`, set `which.plots` to none. The smoothing parameters include `spline.types`, `df`, `lambdas` and `smoothing.methods` (see probeSmooths).

Multiple plots, possibly each having multiple facets, are produced using `ggplot2`. The layout of these plots is controlled via the smoothing-parameter factors `Type`, `Tuning` (the combination of `TunePar` and `TuneVal`) and `Method` that can be supplied to the arguments `plots.by`, `plots.group`, `facet.x` and `facet.y`. These `plots` and `facet` arguments can also include factors other than the smoothing-parameter factors, that are also associated with the data. The basic principle is that the number of levels combinations of the smoothing-parameter factors included in the `plots` and `facet` arguments must be the same as those covered by the combinations of the values supplied to `spline.types`, `df`, `lambdas` and `Method` and incorporated into the smooths.frame input to `plotSmoothsMedianDevns` via the `data` argument. This ensures that smooths from different parameter sets are not pooled in a single plot. Envelopes of the median value of a trait for each factor combination can be added.

Usage

```
plotSmoothsMedianDevns(data, response, response.smoothed = NULL,
                       individuals = "Snapshot.ID.Tag",  times = "DAP",
                       trait.types = c("response", "AGR", "RGR"),
                       x.title = NULL, y.titles = NULL,
                       meddevn.plot.args =
                         args4meddevn_plot(plots.by = NULL, plots.group = NULL,
                                           facet.x = ".", facet.y = ".",
                                           propn.note = TRUE,
                                           propn.types = c(0.1, 0.5, 0.75)),
                       printPlot = TRUE, ...)
```

Arguments

`data`	A smooths.frame, such as is produced by probeSmooths and that contains the data resulting from smoothing a response over time for a set of `individuals`, the data being arranged in long format both with respect to the times and the smoothing-parameter values used in the smoothing. That is, each response occupies a single column. The smooths.frame must include the columns `Type`, `TunePar`, `TuneVal`, `Tuning` and `Method`, and the columns nominated using the arguments `individuals`, `times`, `plots.by`, `facet.x`, `facet.y`, `plots.group`, `response`, `response.smoothed`, and, if requested, the AGR and the RGR of the `response` and `response.smoothed`. The names of the growth rates should be formed from `response` and `response.smoothed` by adding `.AGR` and `.RGR` to both of them.
`response`	A character specifying the response variable for which the observed values are supplied. Depending on the setting of `trait.types`, the observed values of related `trait.types` may also need to be be supplied.
`response.smoothed`	A character specifying the name of the column containing the values of the smoothed response variable, corresponding to `response` and obtained for the combinations of `smoothing.methods` and `df`, usually using smoothing splines.

	If `response.smoothed` is `NULL`, then `response.smoothed` is set to the `response` to which is added the prefix `s`. Depending on the setting of `trait.types`, the smoothed values of related `trait.types` may also need to be be supplied.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used to provide the values to be plotted on the x-axis. If a `factor` or `character`, the values should be numerics stored as characters.
`trait.types`	A `character` giving the traits types that are to be plotted. While `AGR` and `RGR` are commonly used, the names can be arbitrary, except that `response` is a special case that indicates that the original response is to be plotted. If `all`, each of `response`, `AGR` and `RGR` is plotted.
`x.title`	Title for the x-axis. If `NULL` then set to `times`.
`y.titles`	A `character` giving the titles for the y-axis, one for each trait specified by `trait.types`. If `NULL`, then set to the traits derived for `response` from `trait.types`.
`meddevn.plot.args`	A named `list` that is most easily generated using `args4meddevn_plot`, it documenting the options available for varying median deviations plots. *Note that if* `args4meddevn_plot` *is to be called to change from the default settings given in the default* `plotSmoothsMedianDevns` *call and some of those settings are to be retained, then the arguments whose settings are to be retained must also be included in the call to* `args4meddevn_plot`*; be aware that if you call* `args4meddevn_plot`*, then the defaults for this call are those for* `args4meddevn_plot`*, **NOT** the call to* `args4meddevn_plot` *shown as the default for* `plotSmoothsMedianDevns`.
`printPlot`	A `logical` indicating whether or not to print the plot.
`...`	allows passing of arguments to other functions; not used at present.

Value

A `list` that consists of two components: (i) a componenent named `plots` that stores a two-level `list` of the median deviations plots; the first-level `list` has a component for each `trait.types` and each of these `list`(s) is a second-level `list` that contains the set of plots specified by `plots.by` (if `plots.by` is `NULL`, a single plot is stored); (ii) a component named `med.dev.dat` that stores the `data.frame` containing the median deviations that have been plotted. Each plot in the `plots` `list` is in an object of class `ggplot`, which can be plotted using `print`.

Author(s)

Chris Brien

See Also

`traitSmooth`, `probeSmooths`, `args4meddevn_plot`, `plotSmoothsComparison`, `plotDeviationsBoxes`, `ggplot`.

Examples

```
data(exampleData)
vline <- list(ggplot2::geom_vline(xintercept=29, linetype="longdash", size=1))
traits <- probeSmooths(data = longi.dat,
                       response = "PSA", response.smoothed = "sPSA",
                       times = "DAP",
                       get.rates = FALSE, trait.types = "response",
                       smoothing.args =
                         args4smoothing(smoothing.methods = "direct",
                                        spline.types = "NCSS",
                                        df = c(4,7), lambdas = NULL),
                       which.plots = "none")
med <- plotSmoothsMedianDevns(data = traits,
                              response = "PSA", response.smoothed = "sPSA",
                              times = "DAP", trait.types = "response",
                              meddevn.plot.args =
                                args4meddevn_plot(plots.by = NULL,
                                                  plots.group = "Tuning",
                                                  facet.x = ".", facet.y = ".",
                                                  propn.types = 0.02,
                                                  ggplotFuncs = vline))
```

`prepImageData` *Prepares raw imaging data for further processing*

Description

Forms the prime traits by selecting a subset of the traits in a data.frame of imaging data produced by the Lemna Tec Scanalyzer. The imaging traits to be retained are specified using the `traits` and `labsCamerasViews` arguments. Some imaging traits are divided by 1000 to convert them from pixels to kilopixels. Also added are `factor`s and explanatory variates that might be of use in an analysis of the data.

Usage

```
prepImageData(data, cartId = "Snapshot.ID.Tag",
              imageTimes = "Snapshot.Time.Stamp",
              timeAfterStart = "Time.after.Planting..d.",
              PSAcolumn = "Projected.Shoot.Area..pixels.",
              idcolumns = c("Genotype.ID","Treatment.1"),
              traits = list(all = c("Area",
                                    "Boundary.Points.To.Area.Ratio",
                                    "Caliper.Length", "Compactness",
                                    "Convex.Hull.Area"),
                            side = c("Center.Of.Mass.Y",
                                     "Max.Dist.Above.Horizon.Line")),
              labsCamerasViews = list(all = c("SV1", "SV2", "TV"),
                                      side = c("SV1", "SV2")),
```

```
              smarthouse.lev = NULL,
              calcWaterUse = TRUE)
```

Arguments

`data`	A `data.frame` containing the columns specified by `cartId`, `imageTimes`, `timeAfterStart`, `PSAcolumn` `idcolumns`, `traits` and `cameras` along with the following columns: Smarthouse, Lane, Position, Weight.Before, Weight.After, Water.Amount, Projected.Shoot.Area..pixels. The defaults for the arguments to `prepImageData` requires a `data.frame` containing the following columns, although not necessarily in the order given here: Smarthouse, Lane, Position, Weight.Before, Weight.After, Water.Amount, Projected.Shoot.Area..pixels., Area.SV1, Area.SV2, Area.TV, Boundary.Points.To.Area.Ratio.SV1, Boundary.Points.To.Area.Ratio.SV2, Boundary.Points.To.Area.Ratio.TV, Caliper.Length.SV1, Caliper.Length.SV2, Caliper.Length.TV, Compactness.SV1, Compactness.SV2, Compactness.TV, Convex.Hull.Area.SV1, Convex.Hull.Area.SV2, Convex.Hull.Area.TV, Center.Of.Mass.Y.SV1, Center.Of.Mass.Y.SV2, Max.Dist.Above.Horizon.Line.SV1, Max.Dist.Above.Horizon.Line.SV2.
`cartId`	A `character` giving the name of the column that contains the unique Id for each cart.
`imageTimes`	A `character` giving the name of the column that contains the time that each cart was imaged.
`timeAfterStart`	A `character` giving the name of the column that contains the time after some nominated starting time e.g. the number of days after planting.
`PSAcolumn`	A `character` giving the name of the column that contains the projected shoot area.
`idcolumns`	A `character` vector giving the names of the columns that identify differences between the plants or carts e.g. Genotype.ID, Treatment.1, Treatment.2.
`traits`	A `character` or a `list` whose components are `character`s. Each `character` gives the names of the columns for imaging `traits` whose values are required for each of the camera-view combinations given in the corresponding `list` component of `labsCamerasViews`. If `labsCamerasViews` or a component of `labsCamerasViews` is `NULL`, then the contents of `traits` or the coresponding component of `traits` are merely treated as the names of columns to be retained.
`labsCamerasViews`	A `character` or a `list` whose components are `character`s. Each `character` gives the labels of the camera-view combinations for which is required values of each of the imaging `traits` in the corresponding `character` of `traits`. It is assumed that the camera-view labels are appended to the trait names and separated from the trait names by a full stop (.). If `labsCamerasViews` or a component of `labsCamerasViews` is `NULL`, then the contents of the `traits` or the coresponding component of `traits` are merely treated as the names of columns to be retained.

`smarthouse.lev`	A `character` vector giving the levels to use for the Smarthouse `factor`. If `NULL` then the unique values in Smarthouse will be used.
`calcWaterUse`	A `logical` indicating whether to calculate the Water.Loss. If it is `FALSE`, `Water.Before`, `Water.After` and `Water.Amount` will not be in the returned `data.frame`. They can be copied across by listing them in a component of `traits` and set the corresponding component of `cameras` to `NULL`.

Details

The columns are copied from `data`, except for those columns that are calculated from the columns in `data`; those columns that are calculated have '(calculated)' appended in the list under **Value**.

Value

A `data.frame` containing the columns specified by `cartId`, `imageTimes`, `timeAfterStart`, `idcolumns`, `traits` and `cameras`. The defaults will result in the following columns:

1. Smarthouse: `factor` with levels for the Smarthouse
2. Lane: `factor` for lane number in a smarthouse
3. Position: `factor` for east/west position in a lane
4. DAP: `factor` for the number of Days After Planting
5. xDAP: numeric for the DAP (calculated)
6. `cartId`: unique code for each cart
7. `imageTimes`: time at which an image was taken in POSIXct format
8. Hour: hour of the day, to 2 decimal places, at which the image was taken (calculated)
9. Reps: `factor` indexing the replicates for each combination of the `factor`s in `idcolumns` (calculated)
10. `idcolumns`: the columns listed in `idcolumns` that have been converted to `factor`s
11. Weight.Before: weight of the pot before watering (only if `calcWaterUse` is `TRUE`)
12. Weight.After: weight of the pot after watering (only if `calcWaterUse` is `TRUE`)
13. Water.Amount: the weight of the water added (= Water.After - Water.Before) (calculated)
14. WU: the water use calculated as the difference between Weight.Before for the current imaging and the Weight.After for the previous imaging (calculated unless `calcWaterUse` is `FALSE`)
15. PSA: the Projected.Shoot.Area..pixels. divided by 1000 (calculated)
16. PSA.SV1: the Projected.Shoot.Area from Side View 1 divided by 1000 (calculated)
17. PSA.SV2: the Projected.Shoot.Area from Side View 2 divided by 1000 (calculated)
18. PSA.TV: the Projected.Shoot.Area from Top View divided by 1000 (calculated)
19. Boundary.To.PSA.Ratio.SV1
20. Boundary.To.PSA.Ratio.SV2
21. Boundary.To.PSA.Ratio.TV
22. Caliper.Length.SV1
23. Caliper.Length.SV2

24. Caliper.Length.TV
25. Compactness.SV1 from Side View 1
26. Compactness.SV2 from Side View 2
27. Compactness.TV: from Top View
28. Convex.Hull.PSA.SV1: area of Side View 1 Convex Hull divided by 1000 (calculated)
29. Convex.Hull.PSA.SV2: area of Side View 2 Convex Hull divided by 1000 (calculated)
30. Convex.Hull.PSA.TV: Convex.Hull.Area.TV divided by 1000 (calculated)
31. Center.Of.Mass.Y.SV1: Centre of Mass from Side View 1
32. Center.Of.Mass.Y.SV2: Centre of Mass from Side View 2
33. Max.Dist.Above.Horizon.Line.SV1: the Max.Dist.Above.Horizon.Line.SV1 divided by 1000 (calculated)
34. Max.Dist.Above.Horizon.Line.SV2: the Max.Dist.Above.Horizon.Line.SV2 divided by 1000 (calculated)

Author(s)

Chris Brien

Examples

```
data(exampleData)
longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1)

longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1,
                           traits = list(a = "Area", c = "Compactness"),
                           labsCamerasViews = list(all = c("SV1", "SV2", "TV"),
                                                   t = "TV"))

longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1,
                           traits = c("Area.SV1", "Area.SV2", "Area.TV",
                                      "Compactness.TV"),
                           labsCamerasViews = NULL)

longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1,
                           calcWaterUse = FALSE,
                           traits = list(img = c("Area", "Compactness"),
                                         H20 = c("Weight.Before","Weight.After",
                                                 "Water.Amount")),
                          labsCamerasViews = list(all = c("SV1", "SV2", "TV"),
                                                  H2O = NULL))
```

`probeSmooths`	*For a response in a* `data.frame` *in long format, computes and compares, for sets of smoothing parameters, smooths of the response, possibly along with growth rates calculated from the smooths.*

Description

Takes an observed `response` and, for each `individual`, uses `byIndv4Times_SplinesGRs` to smooth its values employing the smoothing parameters specified by (i) `spline.types`, (ii) the tuning parameters, being the degrees of freedom values in `df` or the smoothing penalties in `lambdas`, and (iii) the `smoothing.methods`. The values of these, and other, smoothing arguments are set using the helper function `args4smoothing`.

Provided `get.rates` is `TRUE` or includes `raw` and/or `smoothed` and depending on the setting of `trait.types`, the Absolute Growth Rates (AGR) and/or the Relative Growth Rates (RGR) are calculated for each individual from the unsmoothed, observed `response` and from the smooths of the `response`, using either differences or first derivatives, as specified by `rates.method`.

Generally, profile plots for the traits (a response, an AGR or an RGR) specified in `traits.types` are produced if `which.plots` is `profiles`; if `which.plots` specifies one or more deviations plots, then those deviations plots will also be produced, these being based on the unsmoothed data from which the smoothed data has been subtracted. The layout of the plots is controlled via combinations of one or more of the smoothing-parameter `factor`s `Type`, `TunePar`, `TuneVal`, `Tuning` (the combination of `TunePar` and `TuneVal`) and `Method`, as well as other `factor`s associated with the `data`. The `factor`s that are to be used for the profile plots are supplied via the argument `profile.plot.args` using the helper function `args4profile_plot` and for the and deviations boxplots using the helper function `args4devnboxes_plot`. These helper functions set `plots.by`, `facet.x`, and `facet.y`. For the plots of the medians of the deviations, the `factor`s are supplied via the argument `meddevn.plot.args` using the helper function `args4meddevn_plot` to set `plots.by`, `facet.x`, `facet.y` and `plots.group`. Here, the basic principle is that the number of levels combinations of the smoothing-parameter `factor`s included in the set of `plots` and `facets` arguments to one of these helper functions must be the same as those covered by the combinations of the values supplied to `spline.types`, `df`, `lambdas` and `smoothing.methods` and incorporated into the `smooths.frame`, such as is returned by `probeSmooths`. This ensures that smooths from different parameter sets are not pooled together in a single plot. It is also possible to include `factor`s that are not smoothing-parameter `factor`s in the `plots` amd `facets` arguments.

The following profiles plots can be produced using `args4profile_plot`: (i) separate plots of the smoothed traits for each combination of the smoothing parameters (include `Type`, `Tuning` and `Method` in `plots.by`); (ii) as for (i), with the corresponding plot for the unsmoothed trait preceeding the plots for the smoothed trait (also set `include.raw` to `alone`); (iii) profiles plots that compare a smoothed trait for all combinations of the values of the smoothing parameters, arranging the plots side-by-side or one above the other (include `Type`, `Tuning` and `Method` in `facet.x` and/or `facet.y` - to include the unsmoothed trait set `include.raw` to one of `facet.x` or `facet.y`; (iv) as for (iii), except that separate plots are produced for each combination of the levels of the `factor`s in `plot.by` and each plot compares the smoothed traits for the smoothing-parameter `factor`s included in `facet.x` and/or `facet.y` (set both `plots.by` and one or more of `facet.x` and `facet.y`).

Deviation plots that can be produced are the absolute and relative deviations boxplots and plots of medians deviations (see `which.plots`).

The handling of missing values is controlled via `na.x.action` and `na.y.action` supplied to the helper function `args4smoothing`.

The `probeSmooths` arguments are grouped according to function in the following order:

1. **Data description arguments:** data, response, response.smoothed, individuals, times, keep.columns, trait.types, get.rates, rates.method, ntimes2span.
2. **Smoothing arguments:** smoothing.args (see `args4smoothing`).
3. **General plot control:** x.title, y.titles, facet.labeller, which.plots.
4. **Profile plots (pf) features:** profile.plot.args (see `args4profile_plot`)
5. **Median-deviations (med) plots features:** meddevn.plot.args (see `args4meddevn_plot`)
6. **Deviations boxplots (box) features:** devnboxes.plot.args (see `args4devnboxes_plot`)

Usage

```
probeSmooths(data, response = "PSA", response.smoothed = NULL,
             individuals="Snapshot.ID.Tag", times = "DAP",
             keep.columns = NULL,
             get.rates = TRUE,
             rates.method="differences", ntimes2span = NULL,
             trait.types = c("response", "AGR", "RGR"),
             smoothing.args =
               args4smoothing(smoothing.methods = "direct",
                              spline.types = "NCSS",
                              df = NULL, lambdas = NULL),
             x.title = NULL, y.titles = NULL,
             which.plots = "profiles",
             profile.plot.args =
               args4profile_plot(plots.by = NULL,
                                 facet.x = ".", facet.y = ".",
                                 include.raw = "no"),
             meddevn.plot.args =
               args4meddevn_plot(plots.by = NULL, plots.group = NULL,
                                 facet.x = ".", facet.y = ".",
                                 propn.note = TRUE,
                                 propn.types = c(0.1, 0.5, 0.75)),
             devnboxes.plot.args =
               args4devnboxes_plot(plots.by = NULL,
                                 facet.x = ".", facet.y = ".",
                                 which.plots = "none"),
             ...)
```

Arguments

`data`	A `data.frame` containing the data or a `smooths.frame` as is produced by `probeSmooths`. if data is not a `smooths.frame`, then smoothing will be performed. If `data` is a `smooths.frame`, then the plotting and selection of smooths will be performed as specified by `smoothing.args` and `which.plots`.

`response`	A `character` specifying the response variable to be supplied to `smoothSpline` and that is to be plotted on the y-axis.
`response.smoothed`	A `character` specifying the name of the column containing the values of the smoothed response variable, corresponding to `response`. If `response.smoothed` is `NULL`, then `response.smoothed` is set to the `response` to which is added the prefix `s`.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used as the values of the predictor variable to be supplied to `smooth.spline` and to be plotted on the x-axis. If a `factor` or `character`, the values should be numerics stored as characters.
`keep.columns`	A `character` vector giving the names of columns from `data` that are to be included in the `smooths.frame` that will be returned. Its main use is when no plots are being produced by `probeSmooths`, but there are columns in the supplied `data.frame` that are likely to be needed for the `plots` and `facets` arguments when producing plots subsequently.
`get.rates`	A `logical` or a `character` specifying which of the `response` and the `response.smoothed` are to have growth rates (AGR and/or RGR) computed and stored. If set to `TRUE` or `c("raw", "smoothed")`, growth rates will be obtained for both. Setting to only one of `raw` or `smoothed`, results in the growth rates for either the `response` or the `response.smoothed` being computed. If set to `none` or `FALSE`, no growth rates ar computed. Which growth.rates are computed can be changed using the arguments `traits.types` and the method used for computing them for the `response.smooth` by `rates.method`. The growth rates for the `response` can only be computed by `differencing`.
`rates.method`	A `character` specifying the method to use in calculating the growth rates for `response.smoothed`. The two possibilities are `"differences"` and `"derivatives"`.
`ntimes2span`	A `numeric` giving the number of values in `times` to span in calculating growth rates by differencing. For `ntimes2span` set to `NULL`, if `rates.method` is set to `differences` then `ntimes2span` is set to 2; if `rates.method` is set to `derivatives` then `ntimes2span` is set to 3. Note that when `get.rates` is includes `raw` or is `TRUE`, the growth rates for the unsmoothed response must be calculated by differencing, even if the growth rates for the smoothed response are computed using `derivatives`. When differencing, each growth rate is calculated as the difference in the values of one of the `responses` for pairs of `times` values that are spanned by `ntimes2span` `times` values divided by the difference between this pair of `times` values. For `ntimes2span` set to 2, a growth rate is the difference between consecutive pairs of values of one of the `responses` divided by the difference between consecutive pairs of `times` values.
`trait.types`	A `character` giving the `trait.types` that are to be plotted. If growth rates are included in `trait.types`, then they will be computed for either the `response` and/or the `response.smoothed`, depending on the setting of `get.rates`. Any

	growth rates included in `trait.types` for the `response` that are available in `data`, but have not been specified for computation in `get.rates`, will be retained in the returned `smooths.frame`. If `all`, the `response.smoothed`, its `AGR` and `RGR`, will be plotted. The `response`, and its `AGR` and `RGR`, will be plotted as the plotting options require it.
`smoothing.args`	A `list` that is most easily generated using `args4smoothing`, it documenting the options available for smoothing the data. It gives the settings of `smoothing.methods`, `spline.types`, `df`, `lambdas`, `smoothing.segments`, `npspline.segments`, `na.x.action`, `na.y.action`, `external.smooths`, and `correctBoundaries`, to be used in smoothing the `response` or in selecting a subset of the smooths in `data`, depending on whether `data` is a `data.frame` or a `smooths.frame`, respectively. Set `smoothing.args` to `NULL` if `data` is a `smooths.frame` and only plotting or extraction of a chosen smooth is required.
`x.title`	Title for the x-axis, used for all plots. If `NULL` then set to `times`.
`y.titles`	A `character` giving the titles for the y-axis, one for each trait specified by `trait.types` and used for all plots. If `NULL` then set to the traits derived for `response` from `trait.types`.
`which.plots`	A `logical` indicating which plots are to be produced. The options are either `none` or some combination of `profiles`, `absolute.boxplots`, `relative.boxplots` and `medians.deviations`. The various `profiles` plots that can be poduced are described in the introduction to this function. Boxplots of the absolute deviations are specified by `absolute.boxplots`, the absolute deviations being the values of a trait minus their smoothed values (observed - smoothed). Boxplots of the relative deviations are specified by `relative.boxplots`, the relative deviations being the absolute deviations divided by the smoothed values of the trait. The option `medians.deviations` results in a plot that compares the medians of the absolute deviations over the values of `times` for each combination of the smoothing-parameter values. The arguments to `probeSmooths` that apply to `medians.deviations` plots have the suffix `med`.
`profile.plot.args`	A named `list` that is most easily generated using `args4profile_plot`, it documenting the options available for varying profile plots and boxplots. *Note that if* `args4profile_plot` *is to be called to change from the default settings given in the default* `probeSmooths` *call and some of those settings are to be retained, then the arguments whose settings are to be retained must also be included in the call to* `args4profile_plot`*; be aware that if you call* `args4profile_plot`*, then the defaults for this call are those for* `args4profile_plot`*, **NOT** the call to* `args4profile_plot` *shown as the default for* `probeSmooths`.
`meddevn.plot.args`	A named `list` that is most easily generated using `args4meddevn_plot`, it documenting the options available for varying median deviations plots. *Note that if* `args4meddevn_plot` *is to be called to change from the default settings given in the default* `probeSmooths` *call and some of those settings are to be retained, then the arguments whose settings are to be retained must also be included in the call to* `args4meddevn_plot`*; be aware that if you call* `args4meddevn_plot`*,*

then the defaults for this call are those for `args4meddevn_plot`*, **NOT** the call to* `args4meddevn_plot` *shown as the default for* `probeSmooths`.

`devnboxes.plot.args`
A named `list` that is most easily generated using `args4devnboxes_plot`, it documenting the options available for varying the boxplots. *Note that if* `args4devnboxes_plot` *is to be called to change from the default settings given in the default* `probeSmooths` *call and some of those settings are to be retained, then the arguments whose settings are to be retained must also be included in the call to* `args4devnboxes_plot`*; be aware that if you call* `args4devnboxes_plot`*, then the defaults for this call are those for* `args4devnboxes_plot`*, **NOT** the call to* `args4devnboxes_plot` *shown as the default for* `probeSmooths`.

`...` allows passing of arguments to `plotProfiles`.

Value

A `smooths.frame` that contains the unsmoothed and smoothed data in long format. That is, all the values for either an unsmoothed or a smoothed trait are in a single column. The smooths for a trait for the different combinatons of the smoothing parameters are placed in rows one below the other. The columns that are included in the `smooths.frame` are `Type`, `TunePar`, `TuneVal`, `Tuning` and `Method`, as well as those specified by `individuals`, `times`, `response`, and `response.smoothed`. and any included in the `keep.columns`, `plots` and `facet` arguments. If `trait.types` includes `AGR` or `RGR`, then the included growth rate(s) of the `response` and `response.smoothed` must be present, unless `get.rates` is TRUE or includes `raw` and/or `smoothed`. In this case, the growth rates specified by `trait.types` will be calculated for the responses nominated by `get.rates` and the differences between the `times` used in calculating the rates will be computed and added. Then, the names of the growth rates are formed from `response` and `response.smoothed` by appending `.AGR` and `.RGR` as appropriate; the name of the column with the `times` differences will be formed by appending `.diffs` to the value of `times`. The `external.smooths` will also be included. A `smooths.frame` has the attributes described in `smooths.frame`.

Columns in the supplied `data.frame` that have not been used in `probeSmooths` will not be included in the returned `smooths.frame`. If they might be needed subsequently, such as when extra plots are produced, they can be included in the `smooths.frame` by listing them in a `character` vector for the `keep.columns` argument.

The `smooths.frame` is returned invisibly.

Author(s)

Chris Brien

See Also

`args4smoothing`, , `args4meddevn_plot`, `args4profile_plot`, `traitSmooth`, `smoothSpline`, `byIndv4Times_SplinesGRs`, `byIndv4Times_GRsDiff`, `smooth.spline`, `psNormal`, `plotSmoothsComparison`, `plotSmoothsMedianDevns`, `ggplot`.

Examples

```
data(exampleData)
longi.dat <- longi.dat[1:140,] #reduce to a smaller data set
vline <- list(ggplot2::geom_vline(xintercept=29, linetype="longdash", linewidth=1))
yfacets <- c("Smarthouse", "Treatment.1")
probeSmooths(data = longi.dat,
             response = "PSA", response.smoothed = "sPSA",
             individuals = "Snapshot.ID.Tag",times = "DAP",
             smoothing.args =
               args4smoothing(df = c(4,7),
                              lambda = list(PS = c(0.316,10))),
             profile.plot.args =
               args4profile_plot(plots.by = NULL,
                                 facet.x = "Tuning",
                                 facet.y = c("Smarthouse", "Treatment.1"),
                                 include.raw = "no",
                                 alpha  = 0.4,
                                 colour.column = "Method",
                                 colour.values = c("orange", "olivedrab"),
                                 ggplotFuncs = vline))

#An example that supplies three smoothing schemes to be compared
data(tomato.dat)
probeSmooths(data = tomato.dat,
             response = "PSA", response.smoothed = "sPSA",
             times = "DAP",
             smoothing.args =
               args4smoothing(spline.types      = c(  "N", "NCS",   "P"),
                              df                = c(   4,     6,    NA),
                              lambdas           = c(  NA,    NA,     1),
                              smoothing.methods = c("dir", "log", "log"),
                              combinations      = "parallel"),
             which.plots = "medians.deviations",
             meddevn.plot.args =
               args4meddevn_plot(plots.by = NULL,
                                 plots.group = c("Type", "Tuning", "Method"),
                                 facet.x = ".", facet.y = ".",
                                 propn.note = FALSE, propn.types = NULL))
```

PVA *Selects a subset of variables using Principal Variable Analysis (PVA)*

Description

Principal Variable Analysis (PVA) (Cumming and Wooff, 2007) selects a subset from a set of the variables such that the variables in the subset are as uncorrelated as possible, in an effort to ensure that all aspects of the variation in the data are covered.

Usage

```
PVA(obj, ...)
```

Arguments

`obj`	A `data.frame` containing the columns of variables from which the selection is to be made.
`...`	allows passing of arguments to other functions

Details

`PVA` is the generic function for the `PVA` method. Use methods("PVA") to get all the methods for the PVA generic.

`PVA.data.frame` is a method for a `data.frame`.

`PVA.matrix` is a method for a `matrix`.

Value

A `data.frame` giving the results of the variable selection. It will contain the columns `Variable`, `Selected`, `h.partial`, `Added.Propn` and `Cumulative.Propn`.

Author(s)

Chris Brien

References

Cumming, J. A. and D. A. Wooff (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

`PVA.data.frame`, `PVA.matrix`, `intervalPVA`, `rcontrib`

`PVA.data.frame` *Selects a subset of variables stored in a data.frame using Principal Variable Analysis (PVA)*

Description

Principal Variable Analysis (PVA) (Cumming and Wooff, 2007) selects a subset from a set of the variables such that the variables in the subset are as uncorrelated as possible, in an effort to ensure that all aspects of the variation in the data are covered.

Usage

```
## S3 method for class 'data.frame'
PVA(obj, responses, nvarselect = NULL, p.variance = 1, include = NULL,
    plot = TRUE, ...)
```

Arguments

`obj`	A `data.frame` containing the columns of variables from which the selection is to be made.
`responses`	A `character` giving the names of the columns in `data` from which the variables are to be selected.
`nvarselect`	A `numeric` specifying the number of variables to be selected, which includes those listed in `include`. If `nvarselect = 1`, as many variables are selected as is need to satisfy `p.variance`.
`p.variance`	A `numeric` specifying the minimum proportion of the variance that the selected variables must account for,
`include`	A `character` giving the names of the columns in `data` for the variables whose selection is mandatory.
`plot`	A `logical` indicating whether a plot of the cumulative proportion of the variance explained is to be produced.
`...`	allows passing of arguments to other functions

Details

The variable that is most correlated with the other variables is selected first for inclusion. The partial correlation for each of the remaining variables, given the first selected variable, is calculated and the most correlated of these variables is selects for inclusion next. Then the partial correlations are adjust for the second included variables. This process is repeated until the specified criteria have been satisfied. The possibilities are:

1. the default (`nvarselect = NULL` and `p.variance = 1`), which selects all variables in increasing order of amount of information they provide;
2. to select exactly `nvarselect` variables;
3. to select just enough variables, up to a maximum of `nvarselect` variables, to explain at least `p.variance`*100 per cent of the total variance.

Value

A `data.frame` giving the results of the variable selection. It will contain the columns `Variable`, `Selected`, `h.partial`, `Added.Propn` and `Cumulative.Propn`.

Author(s)

Chris Brien

References

Cumming, J. A. and D. A. Wooff (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

`PVA`, `PVA.matrix`, `intervalPVA.data.frame`, `rcontrib`

Examples

```
data(exampleData)
longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1)
longi.dat <- within(longi.dat,
                    {
                      Max.Height <- pmax(Max.Dist.Above.Horizon.Line.SV1,
                                         Max.Dist.Above.Horizon.Line.SV2)
                      Density <- PSA/Max.Height
                      PSA.SV = (PSA.SV1 + PSA.SV2) / 2
                      Image.Biomass = PSA.SV * (PSA.TV^0.5)
                      Centre.Mass <- (Center.Of.Mass.Y.SV1 + Center.Of.Mass.Y.SV2) / 2
                      Compactness.SV = (Compactness.SV1 + Compactness.SV2) / 2
                    })
responses <- c("PSA","PSA.SV","PSA.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
results <-  PVA(longi.dat, responses, p.variance=0.9, plot = FALSE)
```

`PVA.matrix` *Selects a subset of variables using Principal Variable Analysis (PVA) based on a correlation matrix*

Description

Principal Variable Analysis (PVA) (Cumming and Wooff, 2007) selects a subset from a set of the variables such that the variables in the subset are as uncorrelated as possible, in an effort to ensure that all aspects of the variation in the data are covered.

Usage

```
## S3 method for class 'matrix'
PVA(obj, responses, nvarselect = NULL, p.variance = 1, include = NULL,
    plot = TRUE, ...)
```

Arguments

`obj`	A `matrix` containing the correlation matrix for the variables from which the selection is to be made.
`responses`	A `character` giving the names of the rows and columns in `obj`, being the names of the variables from which the selection is to be made.

`nvarselect`	A `numeric` specifying the number of variables to be selected, which includes those listed in `include`. If `nvarselect = 1`, as many variables are selected as is need to satisfy `p.variance`.
`p.variance`	A `numeric` specifying the minimum proportion of the variance that the selected variables must account for,
`include`	A `character` giving the names of the columns in `data` for the variables whose selection is mandatory.
`plot`	A `logical` indicating whether a plot of the cumulative proportion of the variance explained is to be produced.
`...`	allows passing of arguments to other functions

Details

The variable that is most correlated with the other variables is selected first for inclusion. The partial correlation for each of the remaining variables, given the first selected variable, is calculated and the most correlated of these variables is selects for inclusion next. Then the partial correlations are adjust for the second included variables. This process is repeated until the specified criteria have been satisfied. The possibilities are:

1. the default (`nvarselect = NULL` and `p.variance = 1`), which selects all variables in increasing order of amount of information they provide;
2. to select exactly `nvarselect` variables;
3. to select just enough variables, up to a maximum of `nvarselect` variables, to explain at least `p.variance`*100 per cent of the total variance.

Value

A `data.frame` giving the results of the variable selection. It will contain the columns `Variable`, `Selected`, `h.partial`, `Added.Propn` and `Cumulative.Propn`.

Author(s)

Chris Brien

References

Cumming, J. A. and D. A. Wooff (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

`PVA`, `PVA.data.frame`, `intervalPVA.data.frame`, `rcontrib`

Examples

```
data(exampleData)
longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1)
longi.dat <- within(longi.dat,
                    {
                      Max.Height <- pmax(Max.Dist.Above.Horizon.Line.SV1,
                                         Max.Dist.Above.Horizon.Line.SV2)
                      Density <- PSA/Max.Height
                      PSA.SV = (PSA.SV1 + PSA.SV2) / 2
                      Image.Biomass = PSA.SV * (PSA.TV^0.5)
                      Centre.Mass <- (Center.Of.Mass.Y.SV1 + Center.Of.Mass.Y.SV2) / 2
                      Compactness.SV = (Compactness.SV1 + Compactness.SV2) / 2
                    })
responses <- c("PSA","PSA.SV","PSA.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
R <- Hmisc::rcorr(as.matrix(longi.dat[responses]))$r
results <-  PVA(R, responses, p.variance=0.9, plot = FALSE)
```

`rcontrib` *Computes a measure of how correlated each variable in a set is with the other variable, conditional on a nominated subset of them*

Description

A measure of how correlated a variable is with those in a set is given by the square root of the sum of squares of the correlation coefficients between the variables and the other variables in the set (Cumming and Wooff, 2007). Here, the partial correlation between the subset of the variables listed in `response` that are not listed in `include` is calculated from the partial correlation matrix for the subset, adjusting for those variables in `include`. This is useful for manually deciding which of the variables not in `include` should next be added to it.

Usage

```
rcontrib(obj, ...)
```

Arguments

`obj`	A `data.frame` containing the columns of variables from which the correlation measure is to be calculated.
`...`	allows passing of arguments to other functions

Details

`rcontrib` is the generic function for the `rcontrib` method. Use methods("rcontrib") to get all the methods for the rcontrib generic.

`rcontrib.data.frame` is a method for a `data.frame`.

`rcontrib.matrix` is a method for a `matrix`.

Value

A `numeric` giving the correlation measures.

Author(s)

Chris Brien

References

Cumming, J. A. and D. A. Wooff (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

`PVA`, `intervalPVA`

`rcontrib.data.frame` *Computes a measure of how correlated each variable in a set is with the other variable, conditional on a nominated subset of them*

Description

A measure of how correlated a variable is with those in a set is given by the square root of the sum of squares of the correlation coefficients between the variables and the other variables in the set (Cumming and Wooff, 2007). Here, the partial correlation between the subset of the variables listed in `response` that are not listed in `include` is calculated from the partial correlation matrix for the subset, adjusting for those variables in `include`. This is useful for manually deciding which of the variables not in `include` should next be added to it.

Usage

```
## S3 method for class 'data.frame'
rcontrib(obj, responses, include = NULL, ...)
```

Arguments

`obj`	A `data.frame` containing the columns of variables from which the correlation measure is to be calculated.
`responses`	A `character` giving the names of the columns in `data` from which the correlation measure is to be calculated.
`include`	A `character` giving the names of the columns in `data` for the variables for which other variables are to be adjusted.
`...`	allows passing of arguments to other functions.

Value

A `numeric` giving the correlation measures.

Author(s)

Chris Brien

References

Cumming, J. A. and D. A. Wooff (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

`rcontrib`, `rcontrib.matrix`, `PVA`, `intervalPVA.data.frame`

Examples

```
data(exampleData)
longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1)
longi.dat <- within(longi.dat,
                    {
                      Max.Height <- pmax(Max.Dist.Above.Horizon.Line.SV1,
                                         Max.Dist.Above.Horizon.Line.SV2)
                      Density <- PSA/Max.Height
                      PSA.SV = (PSA.SV1 + PSA.SV2) / 2
                      Image.Biomass = PSA.SV * (PSA.TV^0.5)
                      Centre.Mass <- (Center.Of.Mass.Y.SV1 + Center.Of.Mass.Y.SV2) / 2
                      Compactness.SV = (Compactness.SV1 + Compactness.SV2) / 2
                    })
responses <- c("PSA","PSA.SV","PSA.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
h <-  rcontrib(longi.dat, responses, include = "PSA")
```

`rcontrib.matrix` *Computes a measure of how correlated each variable in a set is with the other variable, conditional on a nominated subset of them*

Description

A measure of how correlated a variable is with those in a set is given by the square root of the sum of squares of the correlation coefficients between the variables and the other variables in the set (Cumming and Wooff, 2007). Here, the partial correlation between the subset of the variables listed in `response` that are not listed in `include` is calculated from the partial correlation matrix for the subset, adjusting for those variables in `include`. This is useful for manually deciding which of the variables not in `include` should next be added to it.

Usage

```
## S3 method for class 'matrix'
rcontrib(obj, responses, include = NULL, ...)
```

Arguments

`obj`	A `matrix` containing the correlations of the variables from which the correlation measure is to be calculated.
`responses`	A `character` giving the names of the columns in `data` from which the correlation measure is to be calculated.
`include`	A `character` giving the names of the columns in `data` for the variables for which other variables are to be adjusted.
`...`	allows passing of arguments to other functions.

Value

A `numeric` giving the correlation measures.

Author(s)

Chris Brien

References

Cumming, J. A. and D. A. Wooff (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

`rcontrib`, `rcontrib.data.frame`, `PVA`, `intervalPVA.data.frame`

Examples

```
data(exampleData)
longi.dat <- prepImageData(data=raw.dat, smarthouse.lev=1)
longi.dat <- within(longi.dat,
                    {
                      Max.Height <- pmax(Max.Dist.Above.Horizon.Line.SV1,
                                         Max.Dist.Above.Horizon.Line.SV2)
                      Density <- PSA/Max.Height
                      PSA.SV = (PSA.SV1 + PSA.SV2) / 2
                      Image.Biomass = PSA.SV * (PSA.TV^0.5)
                      Centre.Mass <- (Center.Of.Mass.Y.SV1 + Center.Of.Mass.Y.SV2) / 2
                      Compactness.SV = (Compactness.SV1 + Compactness.SV2) / 2
                    })
responses <- c("PSA","PSA.SV","PSA.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
R <- Hmisc::rcorr(as.matrix(longi.dat[responses]))$r
h <- rcontrib(R, responses, include = "PSA")
```

`RicePrepped.dat` *Prepped data from an experiment to investigate a rice germplasm panel.*

Description

The data is the full set of Lanes and Positions from an experiment in a Smarthouse at the Plant Accelerator in Adelaide. It is used in the `growthPheno-package` as an executable example to illustrate the use of `growthPheno`. The experiment and data collection are described in Al-Tamimi et al. (2016) and the data is derived from the `data.frame` in the file `00-raw.0254.dat.rda` that is available from Al-Tamimi et al. (2017); halpf od the unprred data is in `RiceRaw.dat`.

Usage

```
data(RicePrepped.dat)
```

Format

A data.frame containing 14784 observations on 32 variables. The names of the columns in the data.frame are:

Column	Name	Class	Description
1	Smarthouse	factor	the Smarthouse in which a cart occurs.
2	Snapshot.ID.Tag	character	a unique identifier for each cart in the experiment.
3	xDAP	numeric	the numbers of days after planting on which the current data was observed.
4	DAST	factor	the numbers of days after the salting treatment on which the current data was observed.
5	xDAST	numeric	the numbers of days after the salting treatment on which the current data was observed.
6	cDAST	numeric	a centered numeric covariate for DAST.
7	DAST.diffs	numeric	the number of days between this and the previous observations (all one for this experiment).
8	Lane	factor	the Lane in the 24 Lane x 24 Positions grid.
9	Position	factor	the Position in the 24 Lane x 24 Positions grid.
10	cPosn	numeric	a centered numeric covaariate for Positions.
11	cMainPosn	numeric	a centered numeric covaariate for Main plots.
12	Zone	factor	the Zone of 4 Lanes to which the current cart belonged.
13	cZone	numeric	a centered numeric covariate for Zone.
14	SHZone	factor	the Zone numbered across the two Smarthouses.
15	ZLane	factor	the number of the Lane within a Zone.
16	ZMainunit	factor	the number of the Main plot within a Zone.
17	Subunit	factor	the number of a Cart within a Main plot.
18	Reps	numeric	the replicate number of each Genotype-Salinity combination.
19	Genotype	factor	the number assigned to the 298 Genotypes in the experiment.
20	Salinity	factor	the Salinity treatment (Control, Salt) allocated to a Cart.

21	PSA	numeric	the Projected shoot area (kpixels).
22	PSA.AGR	numeric	the Absolute Growth Rate for the Projected shoot area (kpixels/day).
23	PSA.RGR	numeric	the Relative Growth Rate for the Projected shoot area (per day).
24	Tr	numeric	the amount of water (g) transpired by a plant.
25	TrR	numeric	the rate of water transpireation (g/day) for a plant.
26	PSA.TUE	numeric	the Transpiration Use Efficiency for PSA (kpixels / day) for a plant.
27	sPSA	numeric	the smoothed Projected shoot area (kpixels).
29	sPSA.AGR	numeric	the smoothed Absolute Growth Rate for the Projected shoot area (kpixels/day).
29	sPSA.RGR	numeric	the smoothed Relative Growth Rate for the Projected shoot area (per day).
30	sTr	numeric	the smoothed amount of water (g) transpired by a plant.
31	sTrR	numeric	the smoothed rate of water transpireation (g/day) for a plant.
32	sPSA.TUE	numeric	the smoothed Transpiration Use Efficiency for PSA (kpixels / day) for a plant.

Source

Al-Tamimi N, Brien C, Oakey H, Berger B, Saade S, Ho YS, Schmockel SM, Tester M, Negrao S. (2017) Data from: Salinity tolerance loci revealed in rice using high-throughput non-invasive phenotyping. Retrieved from: doi:10.5061/dryad.3118j.

References

Al-Tamimi, N, Brien, C.J., Oakey, H., Berger, B., Saade, S., Ho, Y. S., Schmockel, S. M., Tester, M. and Negrao, S. (2016) New salinity tolerance loci revealed in rice using high-throughput non-invasive phenotyping. *Nature Communications*, **7**, 13342. Retrieved from doi:10.1038/ncomms13342.

`RiceRaw.dat` *Data for an experiment to investigate a rice germplasm panel*

Description

The data is half (the first 12 of 24 Lanes) of that from an experiment in a Smarthouse at the Plant Accelerator in Adelaide. It is used in the `growthPheno-package` as an executable example to illustrate the use of `growthPheno`. The experiment and data collection are described in Al-Tamimi et al. (2016) and the data is derived from the `data.frame` in the file `00-raw.0255.dat.rda` that is available from Al-Tamimi et al. (2017).

Usage

```
data(RiceRaw.dat)
```

Format

A data.frame containing 7392 observations on 33 variables.

Source

Al-Tamimi N, Brien C, Oakey H, Berger B, Saade S, Ho YS, Schmockel SM, Tester M, Negrao S: Data from: Salinity tolerance loci revealed in rice using high-throughput non-invasive phenotyping. Retrieved from: doi:10.5061/dryad.3118j.

References

Al-Tamimi, N, Brien, C.J., Oakey, H., Berger, B., Saade, S., Ho, Y. S., Schmockel, S. M., Tester, M. and Negrao, S. (2016) New salinity tolerance loci revealed in rice using high-throughput non-invasive phenotyping. *Nature Communications*, **7**, 13342. Retrieved from doi:10.1038/ncomms13342.

`smooths.frame` *Description of a smooths.frame object*

Description

A `data.frame` of S3-class `smooths.frame` that stores the smooths of one or more responses for several sets of smoothing parameters.

`as.smooths.frame` is function that converts a `data.frame` to an object of this class.

`is.smooths.frame` is the membership function for this class; it tests that an object has class `smooths.frame`.

`validSmoothsFrame` can be used to test the validity of a `smooths.frame`.

Value

A `data.frame` that is also inherits the S3-class `smooths.frame`. It contains the results of smoothing a response over time from a set of `individuals`, the data being arranged in long format both with respect to the times and the smoothing-parameter values used in the smoothing. That is, each response occupies a single column. The `smooths.frame` must include the columns `Type`, `TunePar`, `TuneVal`, `Tuning` (the combination of `TunePar` and `TuneVal`) and `Method`, and the columns that would be nominated using the `probeSmooths` arguments `individuals`, the `plots` and `facet` arguments, `times`, `response`, `response.smoothed`, and, if requested, the AGR and the RGR of the `response` and `response.smoothed`. The names of the growth rates should be formed from `response` and `response.smoothed` by adding `.AGR` and `.RGR` to both of them. The function `probeSmooths` produces a `smooths.frame` for a response.

A `smooths.frame` has the following attributes:

1. `individuals`, the `character` giving the name of the `factor` that define the subsets of the `data` for which each subset corresponds to the `response` values for an individual;
2. `n`, the number of unique `individuals`;
3. `times`, the `character` giving the name of the `numeric`, or `factor` with numeric levels, that contains the values of the predictor variable plotted on the x-axis;
4. `t`, the number of unique values in the `times`;
5. `nschemes`, the number of unique combinations of the smoothing-parameter values in the `smoothsframe`.

Author(s)

Chris Brien

See Also

`probeSmooths`, `is.smooths.frame`, `as.smooths.frame`, `validSmoothsFrame`, `args4smoothing`

Examples

```
dat <- read.table(header = TRUE, text = "
Type TunePar TuneVal Tuning Method       ID  DAP   PSA      sPSA
NCSS      df       4   df-4 direct 045451-C   28 57.446 51.18456
NCSS      df       4   df-4 direct 045451-C   30 89.306 87.67343
NCSS      df       7   df-7 direct 045451-C   28 57.446 57.01589
NCSS      df       7   df-7 direct 045451-C   30 89.306 87.01316
")
dat[1:7] <- lapply(dat[1:6], factor)
dat <- as.smooths.frame(dat, individuals = "ID", times = "DAP")
is.smooths.frame(dat)
validSmoothsFrame(dat)

data(exampleData)
vline <- list(ggplot2::geom_vline(xintercept=29, linetype="longdash", size=1))
smths <- probeSmooths(data = longi.dat,
                      response = "PSA", response.smoothed = "sPSA",
                      times = "DAP",
                      smoothing.args =
                         args4smoothing(smoothing.methods = "direct",
                                        spline.types = "NCSS",
                                        df = c(4,7), lambdas = NULL),
                      profile.plot.args =
                        args4profile_plot(plots.by = NULL,
                                          facet.x = "Tuning",
                                          facet.y = "Treatment.1",
                                          include.raw = "no",
                                          ggplotFuncs = vline))
is.smooths.frame(smths)
validSmoothsFrame(smths)
```

`smoothSpline`

Fit a spline to smooth the relationship between a `response` *and an* `x` *in a* `data.frame`*, optionally computing growth rates using derivatives.*

Description

Uses `smooth.spline` to fit a natural cubic smoothing spline or `JOPS` to fit a P-spline to all the values of `response` stored in `data`.

The amount of smoothing can be controlled by tuning parameters, these being related to the penalty. For a natural cubic smoothing spline, these are `df` or `lambda` and, for a P-spline, it is `lambda`. For a P-spline, `npspline.segments` also influences the smoothness of the fit. The `smoothing.method` provides for `direct` and `logarithmic` smoothing. The method of Huang (2001) for correcting the fitted spline for estimation bias at the end-points will be applied when fitting using a natural cubic smoothing spline if `correctBoundaries` is `TRUE`.

The derivatives of the fitted spline can also be obtained, and the Absolute and Relative Growth Rates (AGR and RGR) computed using them, provided `correctBoundaries` is `FALSE`. Otherwise, growth rates can be obtained by difference using `byIndv4Times_GRsDiff`.

The handling of missing values in the observations is controlled via `na.x.action` and `na.y.action`. If there are not at least four distinct, nonmissing x-values, a warning is issued and all smoothed values and derivatives are set to `NA`.

The function `probeSmooths` can be used to investgate the effect the smoothing parameters (`smoothing.method` and `df` or `lambda`) on the smooth that results.

Usage

```
smoothSpline(data, response, response.smoothed = NULL, x,
             smoothing.method = "direct",
             spline.type = "NCSS", df = NULL, lambda = NULL,
             npspline.segments = NULL, correctBoundaries = FALSE,
             rates = NULL, suffices.rates = NULL, sep.rates = ".",
             extra.derivs = NULL, suffices.extra.derivs=NULL,
             na.x.action = "exclude", na.y.action = "trimx", ...)
```

Arguments

`data`	A `data.frame` containing the column to be smoothed.
`response`	A `character` giving the name of the column in `data` that is to be smoothed.
`response.smoothed`	A `character` specifying the name of the column containing the values of the smoothed response variable, corresponding to `response`. If `response.smoothed` is `NULL`, then `response.smoothed` is set to the `response` to which is added the prefix `s`.
`x`	A `character` giving the name of the column in `data` that contains the values of the predictor variable.
`smoothing.method`	A `character` giving the smoothing method to use. The two possibilites are (i) `"direct"`, for directly smoothing the observed `response`, and (ii) `"logarithmic"`, for smoothing the `log`-transformed `response` and then back-transforming by taking the exponentional of the fitted values.
`spline.type`	A `character` giving the type of spline to use. Currently, the possibilites are (i) `"NCSS"`, for natural cubic smoothing splines, and (ii) `"PS"`, for P-splines.
`df`	A `numeric` specifying, for natural cubic smoothing splines (`NCSS`), the desired equivalent number of degrees of freedom of the smooth (trace of the smoother matrix). Lower values result in more smoothing. If `df` = `NULL`, the amount of smoothing can be controlled by setting `lambda`. If both `df` and `lambda` are `NULL`, smoothing is controlled by the default arguments for `smooth.spline`, and any that you supply via the ellipsis (...) argument.

`lambda`	A `numeric` specifying the positive penalty to apply. The amount of smoothing decreases as `lamda` decreases.
`npspline.segments`	A `numeric` specifying, for P-splines (`PS`), the number of equally spaced segments between `min(x)` and `max(x)`, excluding missing values, to use in constructing the B-spline basis for the spline fitting. If `npspline.segments` is NULL, `npspline.segments` is set to the maximum of 10 and `ceiling((nrow(data)-1)/2)` i.e. there will be at least 10 segments and, for more than 22 `x` values, there will be half as many segments as there are `x` values. The amount of smoothing decreases as `npspline.segments` increases.
`correctBoundaries`	A `logical` indicating whether the fitted spline values are to have the method of Huang (2001) applied to them to correct for estimation bias at the end-points. Note that `spline.type` must be `NCSS` and `lambda` and `deriv` must be `NULL` for `correctBoundaries` to be set to `TRUE`.
`rates`	A `character` giving the growth rates that are to be calculated using derivative. It should be a combination of one or more of `"AGR"`, `"PGR"` and `"RGR"`. If `NULL`, then growth rates are not computed.
`suffices.rates`	A `character` giving the characters to be appended to the names of the responses to provide the names of the columns containing the calculated growth rates. The order of the suffices in `suffices.rates` should correspond to the order of the elements of `which.rates`. If `NULL`, the values of `rates` are used.
`sep.rates`	A `character` giving the character(s) to be used to separate the `suffices.rates` value from a `response` value in constructing the name for a new rate. For no separator, set to `""`.
`extra.derivs`	A `numeric` specifying one or more orders of derivatives that are required, in addition to any required for calculating the growth rates. When `rates.method` is `derivatives`, these can be derivatives other than the first. Otherwise, any derivatives can be specified.
`suffices.extra.derivs`	A `character` giving the characters to be appended to `response.method` to construct the names of the derivatives. If `NULL` and the derivatives are to be retained, then `.dv` followed by the order of the derivative is appended to `response.method`
.	
`na.x.action`	A `character` string that specifies the action to be taken when values of `x` are `NA`. The possible values are `fail`, `exclude` or `omit`. For `exclude` and `omit`, predictions and derivatives will only be obtained for nonmissing values of `x`. The difference between these two codes is that for `exclude` the returned `data.frame` will have as many rows as `data`, the missing values have been incorporated.
`na.y.action`	A `character` string that specifies the action to be taken when values of `y`, or the `response`, are `NA`. The possible values are `fail`, `exclude`, `omit`, `allx`, `trimx`, `ltrimx` or `rtrimx`. For all options, except `fail`, missing values in `y` will be removed before smoothing. For `exclude` and `omit`, predictions and derivatives will be obtained only for nonmissing values of `x` that do not have missing `y` values. Again, the difference between these two is that, only for `exclude` will the missing values be incorporated into the returned `data.frame`. For `allx`,

predictions and derivatives will be obtained for all nonmissing `x`. For `trimx`, they will be obtained for all nonmissing `x` between the first and last nonmissing `y` values that have been ordered for `x`; for `ltrimx` and `utrimx` either the lower or upper missing `y` values, respectively, are trimmed.

`...`	allows for arguments to be passed to `smooth.spline`.

Value

A `list` with two components named `predictions` and `fit.spline`.

The `predictions` component is a `data.frame` containing `x` and the fitted smooth. The names of the columns will be the value of `x` and the value of `response.smoothed`. The number of rows in the `data.frame` will be equal to the number of pairs that have neither a missing `x` or `response` and the order of `x` will be the same as the order in `data`. If `deriv` is not `NULL`, columns containing the values of the derivative(s) will be added to the `data.frame`; the name each of these columns will be the value of `response.smoothed` with `.dvf` appended, where `f` is the order of the derivative, or the value of `response.smoothed` and the corresponding element of `suffices.deriv` appended. If `RGR` is not `NULL`, the RGR is calculated as the ratio of value of the first derivative of the fitted spline and the fitted value for the spline.

The `fit.spline` component is a `list` with components

`x`: the distinct x values in increasing order;

`y`: the fitted values, with boundary values possibly corrected, and corresponding to `x`;

`lev`: leverages, the diagonal values of the smoother matrix (NCSS only);

`lambda`: the value of lambda (corresponding to `spar` for NCSS - see `smooth.spline`);

`df`: the efective degrees of freedom;

`npspline.segments`: the number of equally spaced segments used for smoothing method set to `PS`;

`uncorrected.fit`: the object returned by `smooth.spline` for `smoothing method` set to `NCSS` or by `JOPS::psNormal` for `PS`.

Author(s)

Chris Brien

References

Eilers, P.H.C and Marx, B.D. (2021) *Practical smoothing: the joys of P-splines*. Cambridge University Press, Cambridge.

Huang, C. (2001) Boundary corrected cubic smoothing splines. *Journal of Statistical Computation and Simulation*, **70**, 107-121.

See Also

`byIndv4Times_SplinesGRs`, `probeSmooths`, `byIndv4Times_GRsDiff`, `smooth.spline`, `predict.smooth.spline`, JOPS.

Examples

```
data(exampleData)
fit <- smoothSpline(longi.dat, response="PSA", response.smoothed = "sPSA",
                    x="xDAP", df = 4,
                    rates = c("AGR", "RGR"))
fit <- smoothSpline(longi.dat, response="PSA", response.smoothed = "sPSA",
                    x="xDAP", df = 4,
                    rates = "AGR", suffices.rates = "AGRdv",
                    extra.derivs =  2, suffices.extra.derivs = "Acc")
fit <- smoothSpline(longi.dat, response="PSA", response.smoothed = "sPSA",
                    x="xDAP",
                    spline.type = "PS", lambda = 0.1, npspline.segments = 10,
                    rates = "AGR", suffices.rates = "AGRdv",
                    extra.derivs =  2, suffices.extra.derivs = "Acc")
fit <- smoothSpline(longi.dat, response="PSA", response.smoothed = "sPSA",
                    x="xDAP", df = 4,
                    rates = "AGR", suffices.rates = "AGRdv")
```

`tomato.dat` *Longitudinal data for an experiment to investigate tomato response to mycorrhizal fungi and zinc*

Description

The data is from an experiment in a Smarthouse in the Plant Accelerator and is decribed by Watts-Williams et al. (2019). The experiment involves 32 plants, each placed in a pot in a cart, and the carts were assigned 8 treatments using a randomized complete-block design. The main response is Projected Shoot Area (PSA for short), being the sum of the plant pixels from three images. The eight treatments were the combinations of 4 Zinc (Zn) levels by two Arbuscular Mycorrhiza Fungi (AMF) levls. Each plant was imaged on 35 different days after planting (DAPs). It is used to explore the analysis of growth dynamics.

Usage

```
data(tomato.dat)
```

Format

A data.frame containing 1120 observations on 16 variables. The names of the columns in the data.frame are:

Column	Name	Class	Description
1	Lane	factor	the Lane in the 2 Lane x 16 Positions grid.
2	Position	factor	the Position in the 2 Lane x 16 Positions grid.
3	DAP	factor	the numbers of days after planting on which the current data was observed.
4	Snapshot.ID.Tag	character	a unique identifier for each cart in the experiment.

5	cDAP	numeric	a centered numeric covariate for DAP.
6	DAP.diffs	numeric	the number of days between this and the previous observations (all one for this experiment).
7	cPosn	numeric	a centered numeric covaariate for Positions.
8	Block	factor	the block of the randomized complete-block design to which the current cart belonged.
9	Cart	factor	the number of the cart within a block.
10	AMF	factor	the AMF treatment (- AMF, +AMF) assigned to the cart.
11	Zn	factor	the Zinc level (0, 10, 40, 90) assigned to the cart.
12	Treatments	factor	the combined factor formed from AMF and Zn with levels: (-,0; -,10; -,40; -,90; +,0; +,10; +,40; +,90).
12	Weight.After	numeric	the weight of the cart after watering.
13	Water.Amount	numeric	the weight of the water added to the cart.
14	WU	numeric	the weight of the water used since the previous watering.
15	PSA	numeric	the Projected Shoot Area, being the total number of plant pixels in three plant images.

References

Watts-Williams SJ, Jewell N, Brien C, Berger B, Garnett T, Cavagnaro TR (2019) Using high-throughput phenotyping to explore growth responses to mycorrhizal fungi and zinc in three plant species. *Plant Phenomics*, **2019**, 12.

`traitExtractFeatures` *Extract features, that are single-valued for each individual, from traits observed over time.*

Description

Extract one or more sets of features from traits observed over time, the result being traits that have a single value for each individual. The sets of features are:

1. **single times** – the value for each individual for a single time. (uses `getTimesSubset`)
2. **growth rates for a time interval** – the average growth rate (AGR and/or RGR) over a time interval for each individual. (uses `byIndv4Intvl_GRsDiff` or `byIndv4Intvl_GRsAvg`)
3. **water use traits for a time interval** – the total water use (WU), the water use rate (WUR) and the water use index (WUI) over a time interval for each individual. (uses `byIndv4Intvl_WaterUse so see its documentation for further details`)
4. **growth rates for the imaging period overall** – the average growth rate (AGR and/or RGR) over the whole imaging period for each individual. (uses `byIndv4Intvl_GRsDiff` or `byIndv4Intvl_GRsAvg`)
5. **water use traits for the imaging period overall** – the total water use (WU), the water use rate (WUR) and the water use index (WUI) for the whole imaging period for each individual. (uses `byIndv4Intvl_WaterUse`)
6. **totals for the imaging period overall** – the total over the whole imaging period of a trait for each individual. (uses `byIndv4Intvl_ValueCalc`)

7. **maximum for the imaging period overall** – the maximum value over the whole imaging period, and the time at which it occurred, for each individual. (uses `byIndv4Intvl_ValueCalc`)

The `Tomato` vignette illustrates the use of `traitSmooth` and `traitExtractFeatures` to carry out the SET procedure for the example presented in Brien et al. (2020). Use `vignette("Tomato", package = "growthPheno")` to access it.

Usage

```
traitExtractFeatures(data, individuals = "Snapshot.ID.Tag", times = "DAP",
                     starts.intvl = NULL, stops.intvl = NULL,
                     suffices.intvl = NULL,
                     responses4intvl.rates = NULL,
                     growth.rates = NULL,
                     growth.rates.method = "differences",
                     suffices.growth.rates = NULL,
                     water.use4intvl.traits = NULL,
                     responses4water = NULL,
                     water.trait.types = c("WU", "WUR", "WUI"),
                     suffix.water.rate = "R", suffix.water.index = "I",
                     responses4singletimes = NULL, times.single = NULL,
                     responses4overall.rates = NULL,
                     water.use4overall.water = NULL,
                     responses4overall.water = NULL,
                     responses4overall.totals = NULL,
                     responses4overall.max = NULL,
                     intvl.overall = NULL, suffix.overall = NULL,
                     sep.times.intvl = "to", sep.suffix.times = ".",
                     sep.growth.rates = ".", sep.water.traits = "",
                     mergedata = NULL, ...)
```

Arguments

`data`	A `data.frame` containing the columns specified by `individuals`, `times`, the various `responses` arguments and the `water.use` argument.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the column in `data` containing the times at which the data was collected, either as a `numeric`, `factor`, or `character`. It will be used identifying the intervals and, if a `factor` or `character`, the values should be numerics stored as characters.
`starts.intvl`	A `numeric` giving the times, in terms of values in `times`, that are the initial times for a set of intervals for which `growth.rates` and `water.use` traits are to be obtained. These times may also be used to obtain values for single-time traits (see `responses4singletimes`).
`stops.intvl`	A `numeric` giving the times, in terms of values in `times`, that are the end times for a set of intervals for which `growth.rates` and `water.use` traits are to be

	obtained. These times may also be used to obtain values for single-time traits (see `responses4singletimes`).
`suffices.intvl`	A `character` giving the suffices for intervals specified using `starts.intvl` and `stops.intvl`. If `NULL`, the suffices are automatically generated using `starts.intvl`, `stops.intvl` and `sep.times.intvl`.
`responses4intvl.rates`	A `character` specifying the names of the columns containing responses for which growth rates are to be obtained for the intervals specified by `starts.intvl` and `stops.intvl`. For `growth.rates.method` set to `differences`, the growth rates will be computed from the column of the `response` values whose name is listed in `responses4intvl.rates`. For `growth.rates.method` set to `derivatives`, the growth rates will be computed from a column with the growth rates computed for each time. The name of the column should be a `response` listed in `responses4intvl.rates` to which is appended an element of `suffices.growth.rates`.
`growth.rates.method`	A `character` specifying the method to use in calculating the growth rates over an interval for the responses specified by `responses4intvl.rates`. The two possibilities are `"differences"` and `"ratesaverages"`. For `differences`, the growth rate for an interval is computed by taking differences between the values of a response for pairs of `times`. For `ratesaverage`, the growth rate for an interval is computed by taking weighted averages of growth rates for times within the interval. That is, `differences` operates on the response and `ratesaverage` operates on the growth rates previously calculated from the response, so that the appropriate one of these must be in `data`. The `ratesaverage` option is most appropriate when the growth rates are calculated using the derivatives of a fitted curve. Note that, for responses for which the AGR has been calculated using differences, both methods will give the same result, but the `differences` option will be more efficient than `ratesaverages`.
`growth.rates`	A `character` specifying which growth rates are to be obtained for the intervals specified by `starts.intvl` and `stops.intvl`. It should contain one of both of `"AGR"` and `"RGR"`.
`suffices.growth.rates`	A `character` giving the suffices appended to `responses4intvl.rates` in constructung the column names for the storing the growth rates specified by `growth.rates`. If `suffices.growth.rates` is `NULL`, then `"AGR"` and `"RGR"` will be used.
`water.use4intvl.traits`	A `character` giving the names of the columns in `data` that contain the water use values that are to be used in computing the water use traits (WU, WUR, WUI) for the intervals specified by `starts.intvl` and `stops.intvl`. If there is only one column name, then the `WUI` will be calculated using this name for all column names in `responses4water`. If there are several column names in `water.use4intvl.traits`, then there must be either one or the same number of names in `responses4water`. If both have same number of names, then the two lists of column names will be processed in parallel, so that a single `WUI` will be produced for each pair of `responses4water` and `water.use4intvl.traits` values.
`responses4water`	A `character` giving the names of the columns in `data` that are to provide the nu-

merator in calculating a `WUI` for the intervals specified using `starts.intvl` and `stops.intvl`. The denominator will be the values in the columns in `data` whose names are those given by `water.use4intvl.traits`. If there is only one column name in `responses4water`, then the `WUI` will be calculated using this name for all column names in `responses4water`. If there are several column names in `responses4water`, then there must be either one or the same number of names in `water.use4intvl.traits`. If both have same number of names, then the two lists of column names will be processed in parallel, so that a single `WUI` will be produced for each pair of `responses4water` and `water.use4intvl.traits` values.

See the `Value` section for a description of how `responses4water` is incorporated into the names constructed for the water use traits.

`water.trait.types`

A `character` listing the trait types to compute and return. It should be some combination of `WU`, `WUR` and `WUI`. See Details in `byIndv4Intvl_WaterUse` for how each is calculated.

`suffix.water.rate`

A `character` giving the label to be appended to the value of `water.use4intvl.traits` to form the name of the WUR.

`suffix.water.index`

A `character` giving the label to be appended to the value of `water.use4intvl.traits` to form the name of the WUI.

`responses4singletimes`

A `character` specifying the names of the columns containing `responses` for which a column of the values is to be formed for each response for each of the times values specified in `times.single`. If `times.single` is `NULL`, then the unique values in the combined `starts.intvl` and `stops.intvl` will be used.

`times.single`

A `numeric` giving the times of imaging, for each of which, the values of each `responses4singletimes` will be stored in a column of the resulting `data.frame`. If `NULL`, then the unique values in the combined `starts.intvl` and `stops.intvl` will be used.

`responses4overall.rates`

A `character` specifying the names of the columns containing responses for which growth rates are to be obtained for the whole imaging period i.e. the interval specified by `intvl.overall`. The settings of `growth.rates.method`, `growth.rates`, `suffices.growth.rates`, `sep.growth.rates`, `suffix.overall` and `intvl.overall` will be used in producing the growth rates. See `responses4intvl.rates` for more information about how these arguments are used.

`water.use4overall.water`

A `logical` indicating whether the overall water.traits are to be obtained. The settings of `water.trait.types`, `suffix.water.rate`, `suffix.water.index`, `sep.water.traits`, `suffix.overall` and `intvl.overall` will be used in producing the overall water traits. See `water.use4intvl.traits` for more information about how these arguments are used.

`responses4overall.water`

A `character` giving the names of the columns in `data` that are to provide the numerator in calculating a `WUI` for the interval corresponding to the whole imag-

	ing period. See `response.water` for further details. See `responses4water` for more information about how this argument is processed.
`responses4overall.totals`	A `character` specifying the names of the columns containing responses for which a column of the values is to be formed by summing the response for each individual over the whole of the imaging period.
`responses4overall.max`	A `character` specifying the names of the columns containing responses for which columns of the values are to be formed for the maximum of the response for each individual over the whole of the imaging period and the `times` value at which the maximum occurred.
`intvl.overall`	A `numeric` giving the starts and stop times of imaging. If `NULL`, the start time will be the minimum of `starts.intvl` and the stop time will be the maximum of `stops.intvl`.
`suffix.overall`	A `character` giving the suffix to be appended to the names of traits that apply to the whole imagng period. It applies to `overall.growth.rates`, `water.use4overall.water`, `responses4overall.water` and `responses4overall.totals`. If `NULL`, then nothing will be added.
`sep.times.intvl`	A `character` giving the separator to use in combining a `starts.intvl` with a `stops.intvl` in constructing the suffix to be appended to an interval trait. If set to `NULL` and there is only one value for each of `starts.intvl` and `stops.intvl`, then no suffix will be added; otherwise `sep.times.intvl` set to `NULL` will result in an error.
`sep.suffix.times`	A `character` giving the separator to use in appending a suffix for times to a trait. For no separator, set to `""`.
`sep.growth.rates`	A `character` giving the character(s) to be used to separate the `suffices.growth.rates` value from the `responses4intvl.rates` values in constructing the name for a new rate. It is also used for separating `responses4water` values from the `suffix.water.index`. For no separator, set to `""`.
`sep.water.traits`	A `character` giving the character(s) to be used to separate the `suffix.rate` and `suffix.index` values from the `response` value in constructing the name for a new rate/index. The default of `""` results in no separator.
`mergedata`	A `data.frame` containing a column with the name given in `individuals` and for which there is only one row for each value given in this column. In general, it will be that the number of rows in `mergedata` is equal to the number of unique values in the column in `data` labelled by the value of `individuals`, but this is not mandatory. If `mergedata` is not `NULL`, the values extracted by `traitExtractFeatures` will be `merge`d with it.
`...`	allows passing of arguments to other functions; not used at present.

Value

A `data.frame` that contains an `individuals` column and a column for each extracted trait, in addition to any columns in `mergedata`. The number of rows in the `data.frame` will equal the number of unique element of the `individuals` column in `data`, except when there are extra values in the `individuals` column in `data`. If the latter applies, then the number of rows will equal the number of unique values in the combined `individuals` columns from `mergedata` and `data`.

The names of the columns produced by the function are constructed as follows:

1. **single times** – A name for a single-time trait is formed by appending a full stop to an element of `responses4singletimes`, followed by the value of `times` at which the values were observed.
2. **growth rates for a time interval** – The name for an interval growth rate is constructed by concatenating the relevant element of `responses4intvl.rates`, `growth.rates` and a suffix for the time interval, each separated by a full stop. The interval suffix is formed by joining its `starts.intvl` and `stops.intvl` values, separating them by the value of `sep.times.intvl`.
3. **growth rates for the whole imaging period** – The name for an interval growth rate is constructed by concatenating the relevant element of `responses4intvl.rates`, `growth.rates` and `suffix.overall`, each separated by a full stop.
4. **water use traits for a time interval** – Construction of the names for the three water traits begins with the value of `water.use4intvl.traits`. The rate (WUR) has either `R` or the value of `suffix.water.rate` added to the value of `water.use4intvl.traits`. Similarly the index (WUI) has either `I` or the value of `suffix.water.index` added to it. The WUI also has the element of `responses4water` used in calculating the WUI prefixed to its name. All three water use traits have a suffix for the interval appended to their names. This suffix is contructed by joining its `starts.intvl` and `stops.intvl`, separated by the value of `sep.times.intvl`.
5. **water use traits for the whole imaging period** – Construction of the names for the three water traits begins with the value of `water.use4intvl.traits`. The rate (WUR) has either `R` or the value of `suffix.water.rate` added to the value of `water.use4intvl.traits`. Similarly the index (WUI) has either `I` or the value of `suffix.water.index` added to it. The WUI also has the element of `responses4water` used in calculating the WUI prefixed to its name. All three water use traits have `suffix.overall` appended to their names.
6. **the total for the whole of imaging period** – The name for whole-of-imaging total is formed by combining an element of`responses4overall.totals` with `suffix.overall`, separating them by a full stop.
7. **maximum for the whole of imaging period** – The name of the column with the maximum values will be the result of concatenating the `responses4overall.max`, `"max"` and `suffix.overall`, each separated by a full stop. The name of the column with the value of `times` at which the maximum occurred will be the result of concatenating the `responses4overall.max`, `"max"` and the value of `times`, each separated by a full stop.

The `data.frame` is returned invisibly.

Author(s)

Chris Brien

References

Brien, C., Jewell, N., Garnett, T., Watts-Williams, S. J., & Berger, B. (2020). Smoothing and extraction of traits in the growth analysis of noninvasive phenotypic data. *Plant Methods*, **16**, 36. doi:10.1186/s13007020005776.

See Also

getTimesSubset, byIndv4Intvl_GRsAvg, byIndv4Intvl_GRsDiff, byIndv4Intvl_WaterUse, byIndv_ValueCalc.

Examples

```
#Load dat
data(tomato.dat)

#Define DAP constants
DAP.endpts   <- c(18,22,27,33,39,43,51)
nDAP.endpts <- length(DAP.endpts)
DAP.starts <- DAP.endpts[-nDAP.endpts]
DAP.stops   <- DAP.endpts[-1]
DAP.segs <- list(c(DAP.endpts[1]-1, 39),
                 c(40, DAP.endpts[nDAP.endpts]))
#Add PSA rates and smooth PSA, also producing sPSA rates
tom.dat <- byIndv4Times_SplinesGRs(data = tomato.dat,
                                   response = "PSA", response.smoothed = "sPSA",
                                   times = "DAP", rates.method = "differences",
                                   smoothing.method = "log",
                                   spline.type = "PS", lambda = 1,
                                   smoothing.segments = DAP.segs)

#Smooth WU
tom.dat <- byIndv4Times_SplinesGRs(data = tom.dat,
                                   response = "WU", response.smoothed = "sWU",
                                   rates.method = "none",
                                   times = "DAP",
                                   smoothing.method = "direct",
                                   spline.type = "PS", lambda = 10^(-0.5),
                                   smoothing.segments = DAP.segs)

#Extract single-valued traits for each individual
indv.cols <- c("Snapshot.ID.Tag", "Lane", "Position", "Block", "Cart", "AMF", "Zn")
indv.dat <- subset(tom.dat, subset = DAP == DAP.endpts[1],
                   select = indv.cols)
indv.dat <- traitExtractFeatures(data = tom.dat,
                                 starts.intvl = DAP.starts, stops.intvl = DAP.stops,
                                 responses4singletimes = "sPSA",
                                 responses4intvl.rates = "sPSA",
                                 growth.rates = c("AGR", "RGR"),
                                 water.use4intvl.traits = "sWU",
                                 responses4water = "sPSA",
                                 responses4overall.totals = "sWU",
                                 responses4overall.max = "sPSA.AGR",
                                 mergedata = indv.dat)
```

`traitSmooth` *Obtain smooths for a trait by fitting spline functions and, having compared several smooths, allows one of them to be chosen and returned in a* `data.frame`*.*

Description

Takes a `response` that has been observed for a set of `individuals` over a number `times` and carries out one or more of the following steps:

Smooth: Produces `response.smoothed` using splines for a set of smoothing parameter settings and, optionally, computes growth rates either as differences or derivatives. (see `smoothing.args` below and `args4smoothing`) This step is bypassed if a `data.frame` that is also of class `smooths.frame` is supplied to `data`.

Profile plots: Produces profile plots of `response.smoothed` and its growth rates that compare the smooths; also, boxplots of the deviations of the observed from smoothed data can be obtained. (see `profile.plot.args` below and `args4profile_plot`) Whether these plots are produced is controlled via `which.plots` or whether `profile.plot.args` is set to `NULL`.

Median deviations plots: Produces plots of the medians of the deviations of the observed `response`, and its growth rates, from `response.smoothed`, and its growth rates. These aid in the assessment of the different smooths. (see `meddevn.plot.args` below and `args4meddevn_plot`) Whether these plots are produced is controlled via `which.plots` or whether `meddevn.plot.args` is set to `NULL`.

Deviations boxplots: Produces boxplots of the absolute or relative deviations of the observed `response`, and its growth rates, from `response.smoothed`, and its growth rates. These aid in the assessment of the different smooths. (see `devnboxes.plot.args` below and `args4devnboxes_plot`) Whether these plots are produced is controlled via `which.plots` or whether `devnboxes.plot.args` is set to `NULL`.

Choose a smooth: Extract a single, favoured `response.smoothed`, and its growth rates, for a chosen set of smoothing parameter settings. (see `chosen.smooth.args` below and `args4chosen_smooth`) This step will be omitted if `chosen.smooth.args` is `NULL`.

Chosen smooth plot: Produces profile plots of the chosen smooth and its growth rates. (see `chosen.plot.args` below and `args4chosen_plot`) Whether these plots are produced is controlled by whether `chosen.plot.args` is set to `NULL`.

Each of the '`args4`' functions has a set of defaults that will be used if the corresponding argument, ending in '`.args`', is omitted. The defaults have been optimized for `traitSmooth`.

Input to the function can be either a `data.frame`, that contains data to be smoothed, or a `smooths.frame`, that contains data that has been smoothed. The function can be run (i) without saving any output, (ii) saving the complete set of smooths in a `data.frame` that is also of class `smooths.frame`, (iii) saving a subset of the smooths in a supplied `smooths.frame`, or (iv) saving a single smooth in a `data.frame`, which can be merged with a pre-existing `data.frame` such as the `data.frame` that contains the unsmoothed data.

The `Tomato` vignette illustrates the use of `traitSmooth` and `traitExtractFeatures` to carry out the SET procedure for the example presented in Brien et al. (2020). Use `vignette("Tomato", package = "growthPheno")` to access it.

Usage

```
traitSmooth(data, response, response.smoothed, individuals, times,
            keep.columns = NULL,
            get.rates = TRUE,
            rates.method="differences", ntimes2span = NULL,
            trait.types = c("response", "AGR", "RGR"),
            smoothing.args = args4smoothing(),
            x.title = NULL, y.titles = NULL,
            which.plots = c("profiles", "medians.deviations"),
            profile.plot.args = args4profile_plot(),
            meddevn.plot.args = args4meddevn_plot(),
            devnboxes.plot.args = args4devnboxes_plot(),
            chosen.smooth.args = args4chosen_smooth(),
            chosen.plot.args = args4chosen_plot(),
            mergedata = NULL,
            ...)
```

Arguments

`data`	A `data.frame` containing the data or a `smooths.frame` as is produced by `probeSmooths`. if `data` is not a `smooths.frame`, then smoothing will be performed. If `data` is a `smooths.frame`, then the plotting and selection of smooths will be performed as specified by `smoothing.args` , `which.plots`, `chosen.smooth.args` and `chosen.plot.args`.
`response`	A `character` specifying the response variable to be smoothed.
`response.smoothed`	A `character` specifying the name of the column to contain the values of the smoothed response variable, corresponding to `response`.
`individuals`	A `character` giving the name of the `factor` that defines the subsets of the `data` for which each subset corresponds to the `response` values for an individual (e.g. plant, pot, cart, plot or unit).
`times`	A `character` giving the name of the `numeric`, or `factor` with numeric levels, that contains the values of the predictor variable to be supplied to `smooth.spline` and to be plotted on the x-axis.
`keep.columns`	A `character` vector giving the names of columns from `data` that are to be included in the `smooths.frame` that will be returned.
`get.rates`	A `logical` or a `character` specifying which of the `response` and the `response.smoothed` are to have growth rates (AGR and/or RGR) computed and stored. If set to `TRUE` or `c("raw", "smoothed")`, growth rates will be obtained for both. Setting to only one of `raw` or `smoothed`, results in the growth rates for either the `response` or the `response.smoothed` being computed. If set to `none` or `FALSE`, no growth rates ar computed. Which growth.rates are computed can be changed using the arguments `traits.types` and the method used for computing them for the `response.smooth` by `rates.method`. The growth rates for the `response` can only be computed by `differencing`.
`rates.method`	A `character` specifying the method to use in calculating the growth rates for `response.smoothed`. The two possibilities are `"differences"` and `"derivatives"`.

`ntimes2span`	A `numeric` giving the number of values in `times` to span in calculating growth rates by differencing. For `ntimes2span` set to `NULL`, if `rates.method` is set to `differences` then `ntimes2span` is set to 2; if `rates.method` is set to `derivatives` then `ntimes2span` is set to 3. Note that when `get.rates` is includes `raw` or is `TRUE`, the growth rates for the unsmoothed response must be calculated by differencing, even if the growth rates for the smoothed response are computed using `derivatives`. When differencing, each growth rate is calculated as the difference in the values of one of the `responses` for pairs of `times` values that are spanned by `ntimes2span` `times` values divided by the difference between this pair of `times` values. For `ntimes2span` set to 2, a growth rate is the difference between consecutive pairs of values of one of the `responses` divided by the difference between consecutive pairs of `times` values.
`trait.types`	A `character` giving the `trait.types` that are to be plotted. If growth rates are included in `trait.types`, then they will be computed for either the `response` and/or the `response.smoothed`, depending on the setting of `get.rates`. Any growth rates included in `trait.types` for the `response` that are available in `data`, but have not been specified for computation in `get.rates`, will be retained in the returned `smooths.frame`. If `all`, the `response.smoothed`, its `AGR` and `RGR`, will be plotted. The `response`, and its `AGR` and `RGR`, will be plotted as the plotting options require it.
`smoothing.args`	A `list` that is most easily generated using `args4smoothing`, it documenting the options available for smoothing the data. It gives the settings of `smoothing.methods`, `spline.types`, `df`, `lambdas`, `smoothing.segments`, `npspline.segments`, `na.x.action`, `na.y.action`, `external.smooths`, and `correctBoundaries`, to be used in smoothing the `response` or in selecting a subset of the smooths in `data`, depending on whether `data` is a `data.frame` or a `smooths.frame`, respectively. If `data` is a `data.frame`, then smoothing will be performed. If `data` is a `smooths.frame`, no smoothing will be carried out. If `smoothing.args` is `NULL` then a `smooths.frame` will only be used for plotting. Otherwise, the setting of `smoothing.args` will specifying the smooths that are to be extracted from the `smooths.frame`, in which case `smoothing.args` must specify a subset of the smooths in `data`.
`x.title`	Title for the x-axis, used for all plots. If `NULL` then set to `times`.
`y.titles`	A `character` giving the titles for the y-axis, one for each the response, the AGE and the RGR. They are used for all plots. If `NULL` then they are set to the `response` and the `response` with `.AGR` and `.RGR` appended.
`which.plots`	A `logical` indicating which plots of the smooths specified by `smoothing.args` are to be produced. The options are either none or some combination of `profiles`, `absolute.boxplots`, `relative.boxplots` and `medians.deviations`. The various `profiles` plots that can be poduced are described in the introduction to this function. The plot of a chosen smooth is dealt with separately by the argument `chosen.plot.args`.
`profile.plot.args`	A named `list` that is most easily generated using `args4profile_plot`, it documenting the options available for varying the profile plots. *Note that if* `args4profile_plot` *is being called from* `traitSmooth` *to change some arguments from the default settings, then it is safest to set all of the following arguments in the call:* `plots.by`,

`facet.x` `facet.y` *and* `include.raw`. If this argument is set to `NULL`, these plots will not be produced.

`meddevn.plot.args`
: A named `list` that is most easily generated using `args4meddevn_plot`, it documenting the options available for varying median deviations plots. *Note that if* `args4meddevn_plot` *is being called from* `traitSmooth` *to change some arguments from the default settings, then it is safest to set all of the following arguments in the call:* `plots.by`*,* `plots.group`*,* `facet.x` *and* `facet.y`. If this argument is set to `NULL`, these plots will not be produced.

`devnboxes.plot.args`
: A named `list` that is most easily generated using `args4devnboxes_plot`, it documenting the options available for varying the boxplots. *Note that if* `args4devnboxes_plot` *is being called from* `traitSmooth` *to change some arguments from the default settings, then it is safest to set all of the following arguments in the call:* `plots.by`*,* `facet.x` *and* `facet.y`. If this argument is set to `NULL`, these plots will not be produced.

`chosen.smooth.args`
: A named `list` with just one element or `NULL` for each component. It is most easily generated using `args4chosen_smooth` with `combinations` set to `single`. The call to `args4smoothing` should give the settings of `smoothing.methods`, `spline.types`, `df` and `lambdas` for a single smooth that is to be extracted and that is amongst the smooths that have been produced for the settings specified in `smoothing.methods`. If both `df` and `lambda` in `chosen.smooth.args` are `NULL`, then, depending on the settings for `spline.type` and `smoothng.method`, the value of either `df` or `lambdas` that is the median value or the observed value immediatly below the median value will be added to `chosen.smooth.args`. Otherwise, one of `df` and `lambda` should be `NULL` and the other should be a single `numeric` value. If a value in `chosen.smooth.args` is not amongst those investigated, a value that was investigated will be substituted.

`chosen.plot.args`
: A named `list` that is most easily generated using `args4chosen_plot`, it documenting the options available for varying profile plots. Because this plot includes only a single smooth, the `chosen.smooth.args`, the smoothing-parameter `factor`s are unnecessary and an error will be given if any are included. *Note that if* `args4chosen_plot` *is to be called to change from the default settings given in the default* `traitSmooth` *call, then it is safest to set all of the following arguments in the call:* `plots.by`*,* `facet.x`*,* `facet.y` *and* `include.raw`. If set to `NULL`, then no chosen-smooth plot will be produced.

`mergedata`
: A `data.frame` that is to have the values for the `trait.types` for the smooth specified by `chosen.smooth.args` `merge`d with it. It must contain columns with the names given in `individuals` and `times`, and for which there is only one row for each combination of unique values in these columns. In general, it will be that the number of rows in `mergedata` is equal to the number of unique combinations of the values in the columns of the `chosen.smooth.args` whose names are given by `individuals` and `times`, but this is not mandatory. If only one smooth has been produced, then it will be merged with `data` provided `mergedata` is `NULL` and `data` is not a `smooths.frame`. Othewrwise, a single smooth will be be merged with `mergedata`.

`...`	allows arguments to be passed to `plotProfiles`.

Details

This function is a wrapper function for `probeSmooths`, `plotSmoothsComparison`, `plotSmoothsComparison` and `plotDeviationsBoxes`. It uses the helper functions `args4smoothing`, `args4profile_plot` and `args4meddevn_plot` to se arguments that control the smoothing and plotting.

It takes a `response` that has been observed for a set of `individuals` over a number `times` and produces `response.smoothed`, using `probeSmooths`, for a default set of smoothing parameter settings (see `args4smoothing` for the defaults). The settings can be varied from the defaults by specifying alternate values for the smoothing parameters, the parameters being the type of spline (`spline.types`), the degrees of freedom (`df`) or smoothing penalty (`lambdas`) and `smoothing.methods`. There are also several other smoothing arguments that can be manipulated to affect the smooth (for details see `args4smoothing`). The secondary traits of the absolute growth rate (`AGR`) and relative growth rate (`RGR`) are calculated from the two primary traits, the `response` and `response.smoothed`.

Generally, profile plots for the traits (a response, an AGR or an RGR) specified in `traits.types` are produced if `which.plots` is `profiles`; if `which.plots` specifies one or more deviations plots, then those deviations plots will also be produced, these being based on the unsmoothed data from which the smoothed data has been subtracted. The layout of the plots is controlled via combinations of one or more of the smoothing-parameter `factor`s `Type`, `TunePar`, `TuneVal`, `Tuning` (the combination of `TunePar` and `TuneVal`) and `Method`, as well as other `factor`s associated with the `data`. The `factor`s that are to be used for the profile plots and deviations boxplots are supplied via the argument `profile.plot.args` using the helper function `args4profile_plot` to set `plots.by`, `facet.x`, and `facet.y`; for the plots of the medians of the deviations, the `factor`s are supplied via the argument `meddevn.plot.args` using the helper function `args4meddevn_plot` to set `plots.by`, `facet.x`, `facet.y` and `plots.group`. Here, the basic principle is that the number of levels combinations of the smoothing-parameter `factor`s included in the set of `plots` and `facets` arguments to one of these helper functions must be the same as those covered by the combinations of the values supplied to `spline.types`, `df`, `lambdas` and `smoothing.methods` and incorporated into the `smooths.frame`, such as is returned by `probeSmooths`. This ensures that smooths from different parameter sets are not pooled together in a single plot. It is also possible to include `factor`s that are not smoothing-parameter `factor`s in the `plots` amd `facets` arguments.

The following profiles plots can be produced using `args4profile_plot`: (i) separate plots of the smoothed traits for each combination of the smoothing parameters (include `Type`, `Tuning` and `Method` in `plots.by`); (ii) as for (i), with the corresponding plot for the unsmoothed trait preceeding the plots for the smoothed trait (also set `include.raw` to `alone`); (iii) profiles plots that compare a smoothed trait for all combinations of the values of the smoothing parameters, arranging the plots side-by-side or one above the other (include `Type`, `Tuning` and `Method` in `facet.x` and/or `facet.y` - to include the unsmoothed trait set `include.raw` to one of `facet.x` or `facet.y`; (iv) as for (iii), except that separate plots are produced for each combination of the levels of the `factor`s in `plot.by` and each plot compares the smoothed traits for the smoothing-parameter `factor`s included in `facet.x` and/or `facet.y` (set both `plots.by` and one or more of `facet.x` and `facet.y`).

Deviation plots that can be produced are the absolute and relative deviations boxplots and plots of medians deviations (see `which.plots`).

By default, the single smooth for an arbitrarily chosen combination of the smoothing parameters is returned by the function. The smooth for a single combination other than default combination can be nominated for return using the `chosen.smooth.args` argument. This combination must involve only the supplied values of the smoothing parameters. The values for `response`, the `response.smoothed` and their `AGR`s and `RGR`s are are added to `data`, after any pre-existing columns of these have been removed from `data`. Profile plots of the three smoothed traits are produced using `plotProfiles`. However, if `chosen.smooth.args` is `NULL`, all of the smooths will be returned in a `smooths.frame`, and plots for the single combination of the smoothing parameters will not be produced.

Value

A `smooths.frame` or a `data.frame` that contains the unsmoothed and smoothed data in long format. That is, all the values for either an unsmoothed or a smoothed trait are in a single column.

A `smooths.frame` will be returned when (i) `chosen.smooth.args` is `NULL` and there is more than one smooth specified by the smoothing parameter arguments, or (ii) `chosen.smooth.args` is not `NULL` but `mergedata` is `NULL`. It will contain the smooths for a trait for the different combinatons of the smoothing parameters, the values for the different smooths being placed in rows one below the other. The columns that are included in the `smooths.frame` are `Type`, `TunePar`, `TuneVal`, `Tuning` and `Method`, as well as those specified by `individuals`, `times`, `response`, and `response.smoothed`, and any included in the `keep.columns`, `plots` and `facet` arguments when the smooths were produced. The `AGR` or `RGR` for the `response` and `response.smoothed`, if obtained, will also be included. A `smooths.frame` has the attributes described in `smooths.frame`.

A `data.frame` will be returned when (i) `chosen.smooth.args` and `mergedata` are not `NULL` or (ii) `chosen.smooth.args` is `NULL`, `data` is not a `smooths.frame` and there is only one smooth specified by the smoothing parameter arguments. In either case, if `mergedata` is not `NULL`, the chosen smooth or the single smooth will be merged with the `data.frame` specified by `mergedata`. When there is a single smooth and both `mergedata` and `chosen.smooth.args` are `NULL`, the `data.frame` will include the columns `individuals`, `times`, `response`, and `response.smoothed`, and any included in the `keep.columns`, `plots` and `facet` arguments, as well as any growth rates calculated as a result of `get.rates` and `trait.type`.

The `smooths.frame`/`data.frame` is returned invisibly.

Author(s)

Chris Brien

References

Brien, C., Jewell, N., Garnett, T., Watts-Williams, S. J., & Berger, B. (2020). Smoothing and extraction of traits in the growth analysis of noninvasive phenotypic data. *Plant Methods*, **16**, 36. doi:10.1186/s13007020005776.

See Also

`args4smoothing`, `args4meddevn_plot`, `args4profile_plot`, `args4chosen_smooth`, `args4chosen_plot`, `probeSmooths` `plotSmoothsComparison` and `plotSmoothsMedianDevns`, `ggplot`.

Examples

```
data(exampleData)
longi.dat <- longi.dat[1:140,] #reduce to a smaller data set
vline <- list(ggplot2::geom_vline(xintercept=29, linetype="longdash", linewidth=1))
yfacets <- c("Smarthouse", "Treatment.1")
smth.dat <- traitSmooth(data = longi.dat,
                        response = "PSA", response.smoothed = "sPSA",
                        individuals = "Snapshot.ID.Tag",times = "DAP",
                        keep.columns = yfacets,
                        smoothing.args =
                          args4smoothing(df = c(5,7),
                                         lambda = list(PS = c(0.316,10))),
                        profile.plot.args =
                          args4profile_plot(facet.y = yfacets,
                                            ggplotFuncs = vline),
                        chosen.plot.args =
                          args4chosen_plot(facet.y = yfacets,
                                           ggplotFuncs = vline))
```

`twoLevelOpcreate` *Creates a data.frame formed by applying, for each response, a binary operation to the paired values of two different treatments*

Description

Takes pairs of values for a set of responses indexed by a two-level `treatment.factor` and calculates, for each of pair, the result of applying a binary `operation` to their values for the two levels of the `treatment.factor`. The level of the `treatment.factor` designated the `control` will be on the right of the binary operator and the value for the other level will be on the left.

Usage

```
twoLevelOpcreate(data, responses, treatment.factor = "Treatment.1",
                 suffices.treatment = c("Cont","Salt"), control = 1,
                 columns.suffixed = NULL,
                 operations = "/", suffices.results="OST",
                 columns.retained = c("Snapshot.ID.Tag","Smarthouse","Lane",
                                      "Zone","cZone","SHZone","ZLane",
                                      "ZMainunit","cMainPosn", "Genotype.ID"),
                 by = c("Smarthouse","Zone","ZMainunit"))
```

Arguments

`data`	A `data.frame` containing the columns specified by `treatment.factor`, `columns.retained` and `responses`.
`responses`	A `character` giving the names of the columns in `data` that contain the responses to which the binary operations are to be applied.

`treatment.factor`	A `factor` with two levels corresponding to what is to be designated the `control` and treated observations .
`suffices.treatment`	A `character` giving the characters to be appended to the names of the `responses` and `columns.suffixed` in constructing the names of the columns containing the `responses` and `columns.suffixed` for each level of the `treatment.factor`. The order of the suffices in `suffices.treatment` should correspond to the order of the levels of `treatment.factor`.
`control`	A `numeric`, equal to either 1 or 2, that specifies the level of `treatment.factor` that is the control treatment. The value for the `control` level will be on the right of the binary operator.
`columns.suffixed`	A `character` giving the names of the `columns.retained` in `data` that are to be have the values for each treatment retained and whose names are to be suffixed using `suffices.treatment`. Generally, this is done when `columns.retained` has different values for different levels of the `treatment.factor`.
`operations`	A `character` giving the binary operations to perform on the values for the two different levels of the `treatment.factor`. It should be either of length one, in which case the same operation will be performed for all columns specified in `response.GR`, or equal in length to `response.GR` so its elements correspond to those of `response.GR`.
`suffices.results`	A `character` giving the characters to be appended to the names of the responses in constructing the names of the columns containing the results of applying the operations. The order of the suffices in `suffices.results` should correspond to the order of the operators in `operations`.
`columns.retained`	A `character` giving the names of the columns in `data` that are to be retained in the `data.frame` being created. These are usually `factor`s that index the results of applying the operations and that might be used subsequently.
`by`	A `character` giving the names of the columns in `data` whose combinations will be unique for the observation for each treatment. It is used by `merge` when combining the values of the two treatments in separate columns in the `data.frame` to be returned.

Value

A `data.frame` containing the following columns and the values of the :

1. those from `data` nominated in `columns.retained`;
2. those containing the treated values of the columns whose names are specified in `responses`; the treated values are those having the other level of `treatment.factor` to that specified by `control`;
3. those containing the `control` values of the columns whose names are specified in `responses`; the control values are those having the level of `treatment.factor` specified by `control`;
4. those containing the values calculated using the binary `operations`; the names of these columns will be constructed from `responses` by appending `suffices.results` to them.

Author(s)

Chris Brien

Examples

```
data(exampleData)
responses <- c("sPSA.AGR","sPSA.RGR")
cols.retained <-  c("Snapshot.ID.Tag","Smarthouse","Lane","Position",
                    "DAP","Snapshot.Time.Stamp", "Hour", "xDAP",
                    "Zone","cZone","SHZone","ZLane","ZMainunit",
                    "cMainPosn", "Genotype.ID")
longi.SIIT.dat <-
  twoLevelOpcreate(data = longi.dat, responses = responses,
                   suffices.treatment=c("C","S"),
                   operations = c("-", "/"),
                   suffices.results = c("diff", "SIIT"),
                   columns.retained = cols.retained,
                   by = c("Smarthouse","Zone","ZMainunit","DAP"))
longi.SIIT.dat <- with(longi.SIIT.dat,
                            longi.SIIT.dat[order(Smarthouse,Zone,ZMainunit,DAP),])
```

`validSmoothsFrame` *Checks that an object is a valid* `smooths.frame`.

Description

Checks that an `object` is a `smooths.frame` of S3-class `data.frame` that contains the columns `Type`, `TunePar`, `TuneVal`, `Tuning`, `Method`, as well as the columns specified by the atttributes of the `object`, namely `individuals` and `times`.

Usage

```
validSmoothsFrame(object)
```

Arguments

`object`	a `smooths.frame`.

Value

`TRUE` or a `character` describing why the `object` is not a valid `smooths.frame`.

Author(s)

Chris Brien

See Also

`is.smooths.frame`, `as.smooths.frame`

Examples

```
dat <- read.table(header = TRUE, text = "
Type TunePar TuneVal Tuning Method       ID  DAP   PSA      sPSA
NCSS      df       4   df-4 direct 045451-C   28 57.446 51.18456
NCSS      df       4   df-4 direct 045451-C   30 89.306 87.67343
NCSS      df       7   df-7 direct 045451-C   28 57.446 57.01589
NCSS      df       7   df-7 direct 045451-C   30 89.306 87.01316
")
dat[1:7] <- lapply(dat[1:6], factor)
dat <- as.smooths.frame(dat, individuals = "ID", times = "DAP")
is.smooths.frame(dat)
validSmoothsFrame(dat)
```

`WUI` *Calculates the Water Use Index (WUI)*

Description

Calculates the Water Use Index, returning `NA` if the water use is zero.

Usage

```
WUI(response, water)
```

Arguments

`response`	A `numeric` giving the value of the response achieved.
`water`	A `numeric` giving the amount of water used.

Value

A `numeric` containing the response divided by the water, unless water is zero in which case `NA` is returned.

Author(s)

Chris Brien

Examples

```
data(exampleData)
PSA.WUE <- with(longi.dat, WUI(PSA.AGR, WU))
```

Index