Package 'imageData'

May 8, 2026

Version 0.1.64

Date 2025-07-31

Title Aids in Processing and Plotting Data from a Lemna-Tec Scananalyzer

Depends R (>= 3.1.0)

Imports dae, ggplot2, stats, readxl, Hmisc, GGally, RColorBrewer, reshape, grid

Suggests testthat, R.rsp

VignetteBuilder R.rsp

Description Note that 'imageData' has been superseded by 'growthPheno'. The package 'growthPheno' incorporates all the functionality of 'imageData' and has functionality not available in 'imageData', but some 'imageData' functions have been renamed. The 'imageData' package is no longer maintained, but is retained for legacy purposes.

License GPL (>= 2)

URL http://chris.brien.name

RoxygenNote 5.0.1

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2025-07-31 08:00:02 UTC

Contents

`imageData-package` *Aids in Processing and Plotting Data from a Lemna-Tec Scananalyzer*

Description

Note that 'imageData' has been superseded by 'growthPheno'. The package 'growthPheno' incorporates all the functionality of 'imageData' and has functionality not available in 'imageData', but some 'imageData' functions have been renamed. The 'imageData' package is no longer maintained, but is retained for legacy purposes.

Version: 0.1.64

Date: 2025-07-31

Index

For an overview of the use of these functions and an example see below.

(i) Data

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

(ii) Data frame manipulation

designFactors	Adds the factors and covariates for a blocked, split-plot design.
getDates	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.
longitudinalPrime	Selects a set variables to be retained in a data frame of longitudinal data.
twoLevelOpcreate	Creates a data.frame formed by applying, for each response, abinary operation to the values of two different treatments.

(iii) Plots

anomPlot	Identifies anomalous individuals and produces longitudinal plots without them and with just them.
corrPlot	Calculates and plots correlation matrices for a set of responses.
imagetimesPlot	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).
longiPlot	Plots longitudinal data from a Lemna Tec Scananalyzer.
probeDF	Compares, for a set of specified values of df, a response and the smooths of it, possibly along with growth rates calculated from the smooths.

(iv) Calculations value-by-value

GrowthRates	Calculates growth rates (AGR, PGR, RGRdiff) between pairs of values in a vector.
WUI	Calculates the Water Use Index (WUI).
anom	Tests if any values in a vector are anomalous in being outside specified limits.
calcTimes	Calculates for a set of times, the time intervals after an origin time and the position of each with in that time.

calcLagged	Replaces the values in a vector with the result of applying an operation to it and a lagged value.
cumulate	Calculates the cumulative sum, ignoring the first element if exclude.1st is TRUE.

(v) Calculations over multiple values

fitSpline	Produce the fits from a natural cubic smoothing spline applied to a response in a 'data.frame'.
intervalGRaverage	Calculates the growth rates for a specified time interval by taking weighted averages of growth rates for times within the interval.
intervalGRdiff	Calculates the growth rates for a specified time interval.
intervalValueCalculate	Calculates a single value that is a function of an individual's values for a response over a specified time interval.
intervalWUI	Calculates water use indices (WUI) over a specified time interval to a data.frame.

(vi) Caclulations in each split of a 'data.frame'

splitContGRdiff	Adds the growth rates calculated continuously over time for subsets of a response to a 'data.frame'.
splitSplines	Adds the fits after fitting a natural cubic smoothing spline to subsets of a response to a 'data.frame'.
splitValueCalculate	Calculates a single value that is a function of an individual's values for a response.

(vii) Principal variates analysis (PV A)

intervalPVA	Selects a subset of variables observed within a specified time interval using PVA.
PVA	Selects a subset of variables using PVA.
rcontrib	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 carry out a full seven-step process to produce phenotypic traits from measurements made in a high-throughput phenotyping facility, such as one based on a Lemna-Tec Scananalyzer 3D system and described by Al-Tamimi et al. (2016). Otherwise, individual functions can be used to carry out parts of the process.

The basic data consists of imaging data obtained from a set of pots or carts over time. The carts are

arranged in a grid of Lanes × Positions. There should be a unique identifier for each cart, which by default is Snapshot.ID.Tag, and variable giving the Days after Planting for each measurement, by default Time.after.Planting..d.. In some cases, it is expected that there will be a column labelled Snapshot.Time.Stamp, which reflects the time of the imaging from which a particular data value was obtained.

The full seven-step process is as follows:

1. Use importExcel to import the raw data from the Excel file. This step should also involve any editing of the data needed to take account of mishaps during the data collection and the need to remove faulty data (produces raw.dat). Generally, data can be removed by replacing only values for responses with missing values (NA) for carts whose data is to be removed, leaving the identifying information intact.
2. Use longitudinalPrime to select a subset of the imaging variables produced by the Lemna Tec Scanalyzer and, if the design is a blocked, split-plot design, use designFactors to add covariates and factors that might be used in the analysis (produces the data frame longi.prime.dat).
3. Add derived traits that result in a value for each observation: use splitContGRdiff to obtain continuous growth rates i.e. a growth rate for each time of observation, except the first; WUI to produce continuous Water Use Efficiency Indices (WUE) and cumulate to produce cumulative responses. (Produces the data frame longi.dat.)
4. Use splitSplines to fit splines to smooth the longitudinal trends in the primary traits and calculate continuous growth rates from the smoothed response (added to the data frame longi.dat). There are two options for calculating continuous smoothed growth rates: (i) by differencing — use splitContGRdiff; (ii) from the first derivatives of the splines — in splitSplines include 1 in the deriv argument, include "AGR" in suffices.deriv and set the RGR to say "RGR". Optionally, use probeDF to compare the smooths for a number of values of df and, if necessary, re-run splitSplines with a revised value of df.
5. Perform an exploratory examination of the unsmoothed data by using longiPlot to produce longitudinal plots of unsmoothed imaging traits and continuous growth rates. Also, use longiPlot to plot the smoothed imaging traits and continuous growth rates and anomPlot to check for anomalies in the data.
6. Produce cart data: traits for which there is a single value for each Snapshot.ID.Tag or cart. (produces the data frame cart.dat)
   (a) Set up a cart data.frame with the factors and covariates for a single observation from all carts. This can be done by subsetting longi.dat so that there is one entry for each cart.
   (b) Use getDates to add traits at specific times to the cart data.frame, often the first and last day of imaging for each Snapshot.ID.Tag. The times need to be selected so that there is one and only one observation for each cart. Also form traits, such as growth rates over the whole imaging period, based on these values
   (c) Based on the longitudinal plots, decide on the intervals for which growth rates and WUEs are to be calculated. The growth rates for intervals are calculated from the continuous growth rates, using intervalGRdiff, if the continuous growth rates were calculated by differencing, or intervalGRaverage, if they were calculated from first derivatives. To calculate WUEs for intervals, use intervalWUI, The interval growth rates and WUEs are added to the cart data.frame.
7. (Optional) There is also the possibility that, for experiments investigating salinity, the Shoot Ion Independent Tolerance (SIIT) index can be calculated using twoLevelOpcreate.

Author(s)

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

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

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.

See Also

dae

Examples

```
## Not run:
### This example can be run because the data.frame RiceRaw.dat is available with the package
#'# Step 1: Import the raw data
data(RiceRaw.dat)

#'# Step 2: Select imaging variables and add covariates and factors (produces longi.dat)
longi.dat <- longitudinalPrime(data=RiceRaw.dat, smarthouse.lev=c("NE","NW"))

longi.dat <- designFactors(longi.dat, insertName = "xDays",
                           designfactorMethod="StandardOrder")

#'## Particular edits to longi.dat
longi.dat <- within(longi.dat,
                    {
                      Days.after.Salting <- as.numfac(Days) - 29
                    })
longi.dat <- with(longi.dat, longi.dat[order(Snapshot.ID.Tag,Days), ])

#'# Step 3: Form derived traits that result in a value for each observation
#'### Set responses
responses.image <- c("Area")
responses.smooth <- paste(responses.image, "smooth", sep=".")

#'## Form growth rates for each observation of a subset of responses by differencing
longi.dat <- splitContGRdiff(longi.dat, responses.image,
                             INDICES="Snapshot.ID.Tag",
                             which.rates = c("AGR","RGR"))

#'## Form Area.WUE
longi.dat <- within(longi.dat,
                    {
                      Area.WUE <- WUI(Area.AGR*Days.diffs, Water.Loss)
                    })

#'## Add cumulative responses
```

```
longi.dat <- within(longi.dat,
                    {
                      Water.Loss.Cum <- unlist(by(Water.Loss, Snapshot.ID.Tag,
                                                  cumulate, exclude.1st=TRUE))
                      WUE.cum <- Area / Water.Loss.Cum
                    })

#'# Step 4: Fit splines to smooth the longitudinal trends in the primary traits and
#'# calculate their growth rates
#'
#'## Smooth responses
#+
for (response in c(responses.image, "Water.Loss"))
  longi.dat <- splitSplines(longi.dat, response, x="xDays", INDICES = "Snapshot.ID.Tag",
                            df = 4, na.rm=TRUE)
longi.dat <- with(longi.dat, longi.dat[order(Snapshot.ID.Tag, xDays), ])

#'## Loop over smoothed responses, forming growth rates by differences
#+
responses.GR <- paste(responses.smooth, "AGR", sep=".")
longi.dat <- splitContGRdiff(longi.dat, responses.smooth,
                             INDICES="Snapshot.ID.Tag",
                             which.rates = c("AGR","RGR"))

#'## Finalize longi.dat
longi.dat <- with(longi.dat, longi.dat[order(Snapshot.ID.Tag, xDays), ])

#'# Step 5: Do exploratory plots on unsmoothed and smoothed longitudinal data
responses.longi <- c("Area","Area.AGR","Area.RGR", "Area.WUE")
responses.smooth.plot <- c("Area.smooth","Area.smooth.AGR","Area.smooth.RGR")
titles <- c("Total area (1000 pixels)",
            "Total area AGR (1000 pixels per day)", "Total area RGR (per day)",
            "Total area WUE (1000 pixels per mL)")
titles.smooth<-titles
nresp <- length(responses.longi)
limits <- list(c(0,1000), c(-50,125), c(-0.05,0.40), c(0,30))

#' ### Plot unsmoothed profiles for all longitudinal  responses
#+ "01-ProfilesAll"
klimit <- 0
for (k in 1:nresp)
{
  klimit <- klimit + 1
  longiPlot(data = longi.dat, response = responses.longi[k],
            y.title = titles[k], x="xDays+35.42857143",
            ggplotFuncs = list(geom_vline(xintercept=29, linetype="longdash", size=1),
                               scale_x_continuous(breaks=seq(28, 42, by=2)),
                               scale_y_continuous(limits=limits[[klimit]])))
}


#' ### Plot smoothed profiles for all longitudinal responses - GRs by difference
#+ "01-SmoothedProfilesAll"
```

```
nresp.smooth <- length(responses.smooth.plot)
limits <- list(c(0,1000), c(0,100), c(0.0,0.40))
for (k in 1:nresp.smooth)
{
  longiPlot(data = longi.dat, response = responses.smooth.plot[k],
            y.title = titles.smooth[k], x="xDays+35.42857143",
            ggplotFuncs = list(geom_vline(xintercept=29, linetype="longdash", size=1),
                               scale_x_continuous(breaks=seq(28, 42, by=2)),
                               scale_y_continuous(limits=limits[[klimit]])))
  print(plt)
}


#'### AGR anomalies - plot without anomalous plants followed by plot of anomalous plants
#+ "01-0254-AGRanomalies"
anom.ID <- vector(mode = "character", length = 0L)
response <- "Area.smooth.AGR"
cols.output <- c("Snapshot.ID.Tag", "Smarthouse", "Lane", "Position",
                 "Treatment.1", "Genotype.ID", "Days")
anomalous <- anomPlot(longi.dat, response=response, lower=2.5, start.time=40,
                    x = "xDays+35.42857143", vertical.line=29, breaks=seq(28, 42, by=2),
                      whichPrint=c("innerPlot"), y.title=response)
subs <- subset(anomalous$data, Area.smooth.AGR.anom & Days==42)
if (nrow(subs) == 0)
{ cat("\n#### No anomalous data here\n\n")
} else
{
  subs <- subs[order(subs["Smarthouse"],subs["Treatment.1"], subs[response]),]
  print(subs[c(cols.output, response)])
  anom.ID <- unique(c(anom.ID, subs$Snapshot.ID.Tag))
  outerPlot <- anomalous$outerPlot  + geom_text(data=subs,
                                                aes_string(x = "xDays+35.42857143",
                                                           y = response,
                                                           label="Snapshot.ID.Tag"),
                                                size=3, hjust=0.7, vjust=0.5)
  print(outerPlot)
}


#'# Step 6: Form single-value plant responses in Snapshot.ID.Tag order.
#'
#'## 6a) Set up a data frame with factors only
#+
cart.dat <- longi.dat[longi.dat$Days == 31,
                      c("Smarthouse","Lane","Position","Snapshot.ID.Tag",
                        "xPosn","xMainPosn",
                        "Zones","xZones","SHZones","ZLane","ZMainplots", "Subplots",
                        "Genotype.ID","Treatment.1")]
cart.dat <- cart.dat[do.call(order, cart.dat), ]

#'## 6b) Get responses based on first and last date.
#'
#'### Observation for first and last date
```

```
cart.dat <- cbind(cart.dat, getDates(responses.image, data = longi.dat,
                                     which.times = c(31), suffix = "first"))
cart.dat <- cbind(cart.dat, getDates(responses.image, data = longi.dat,
                                     which.times = c(42), suffix = "last"))
cart.dat <- cbind(cart.dat, getDates(c("WUE.cum"),
                                     data = longi.dat,
                                     which.times = c(42), suffix = "last"))
responses.smooth <- paste(responses.image, "smooth", sep=".")
cart.dat <- cbind(cart.dat, getDates(responses.smooth, data = longi.dat,
                                     which.times = c(31), suffix = "first"))
cart.dat <- cbind(cart.dat, getDates(responses.smooth, data = longi.dat,
                                     which.times = c(42), suffix = "last"))

#'### Growth rates over whole period.
#+
tottime <- 42 - 31
cart.dat <- within(cart.dat,
                   {
                     Area.AGR <- (Area.last - Area.first)/tottime
                     Area.RGR <- log(Area.last / Area.first)/tottime
                   })

#'### Calculate water index over whole period
cart.dat <- merge(cart.dat,
                  intervalWUI("Area", water.use = "Water.Loss",
                              start.times = c(31),
                              end.times = c(42),
                              suffix = NULL,
                              data = longi.dat, include.total.water = TRUE),
                  by = c("Snapshot.ID.Tag"))
names(cart.dat)[match(c("Area.WUI","Water.Loss.Total"),names(cart.dat))] <-
        c("Area.Overall.WUE", "Water.Loss.Overall")
cart.dat$Water.Loss.rate.Overall <- cart.dat$Water.Loss.Overall / (42 - 31)

#'## 6c) Add growth rates and water indices for intervals
#'### Set up intervals
#+
start.days <- list(31,35,31,38)
end.days <- list(35,38,38,42)
suffices <- list("31to35","35to38","31to38","38to42")

#'### Rates for specific intervals from the smoothed data by differencing
#+
for (r in responses.smooth)
{ for (k in 1:length(suffices))
  {
    cart.dat <- merge(cart.dat,
                      intervalGRdiff(r,
                                     which.rates = c("AGR","RGR"),
                                     start.times = start.days[k][[1]],
                                     end.times = end.days[k][[1]],
                                     suffix.interval = suffices[k][[1]],
                                     data = longi.dat),
```

```
                      by = "Snapshot.ID.Tag")
  }
}

#'### Water indices for specific intervals from the unsmoothed and smoothed data
#+
for (k in 1:length(suffices))
{
  cart.dat <- merge(cart.dat,
                    intervalWUI("Area", water.use = "Water.Loss",
                                start.times = start.days[k][[1]],
                                end.times = end.days[k][[1]],
                                suffix = suffices[k][[1]],
                                data = longi.dat, include.total.water = TRUE),
                    by = "Snapshot.ID.Tag")
  names(cart.dat)[match(paste("Area.WUI", suffices[k][[1]], sep="."),
                        names(cart.dat))] <- paste("Area.WUE", suffices[k][[1]], sep=".")
  cart.dat[paste("Water.Loss.rate", suffices[k][[1]], sep=".")] <-
           cart.dat[[paste("Water.Loss.Total", suffices[k][[1]], sep=".")]] /
                                          ( end.days[k][[1]] - start.days[k][[1]])
}

cart.dat <- with(cart.dat, cart.dat[order(Snapshot.ID.Tag), ])

#'# Step 7: Form continuous and interval SIITs
#'
#'## 7a) Calculate continuous
#+
cols.retained <-  c("Snapshot.ID.Tag","Smarthouse","Lane","Position",
                    "Days","Snapshot.Time.Stamp", "Hour", "xDays",
                    "Zones","xZones","SHZones","ZLane","ZMainplots",
                    "xMainPosn", "Genotype.ID")
responses.GR <- c("Area.smooth.AGR","Area.smooth.AGR","Area.smooth.RGR")
suffices.results <- c("diff", "SIIT", "SIIT")
responses.SIIT <- unlist(Map(paste, responses.GR, suffices.results,sep="."))

longi.SIIT.dat <-
  twoLevelOpcreate(responses.GR, longi.dat, suffices.treatment=c("C","S"),
                   operations = c("-", "/", "/"), suffices.results = suffices.results,
                   columns.retained = cols.retained,
                   by = c("Smarthouse","Zones","ZMainplots","Days"))
longi.SIIT.dat <- with(longi.SIIT.dat,
                           longi.SIIT.dat[order(Smarthouse,Zones,ZMainplots,Days),])

#' ### Plot SIIT profiles
#'
#+ "03-SIITProfiles"
k <- 2
nresp <- length(responses.SIIT)
limits <- with(longi.SIIT.dat, list(c(min(Area.smooth.AGR.diff, na.rm=TRUE),
                                      max(Area.smooth.AGR.diff, na.rm=TRUE)),
                                    c(0,3),
                                    c(0,1.5)))
```

```
#Plots
for (k in 1:nresp)
{
  longiPlot(data = longi.SIIT.dat, x="xDays+35.42857143",
            response = responses.SIIT[k],
            y.title=responses.SIIT[k],
            facet.x="Smarthouse", facet.y=".",
            ggplotFuncs = list(geom_vline(xintercept=29, linetype="longdash", size=1),
                               scale_x_continuous(breaks=seq(28, 42, by=2)),
                               scale_y_continuous(limits=limits[[klimit]])))
}

#'## 7b) Calculate interval SIITs and check for large values for SIIT for Days 31to35
#+ "01-SIITIntClean"
suffices <- list("31to35","35to38","31to38","38to42")
response <- "Area.smooth.RGR.31to35"
SIIT <- paste(response, "SIIT", sep=".")
responses.SIITinterval <- as.vector(outer("Area.smooth.RGR", suffices, paste, sep="."))

cart.SIIT.dat <- twoLevelOpcreate(responses.SIITinterval, cart.dat,
                                  suffices.treatment=c("C","S"),
                                  suffices.results="SIIT",
                                  columns.suffixed="Snapshot.ID.Tag")
tmp<-na.omit(cart.SIIT.dat)
print(summary(tmp[SIIT]))
big.SIIT <- with(tmp, tmp[tmp[SIIT] > 1.15, c("Snapshot.ID.Tag.C","Genotype.ID",
                                              paste(response,"C",sep="."),
                                              paste(response,"S",sep="."), SIIT)])
big.SIIT <- big.SIIT[order(big.SIIT[SIIT]),]
print(big.SIIT)
plt <- ggplot(tmp, aes_string(SIIT)) +
          geom_histogram(aes(y = ..density..), binwidth=0.05) +
          geom_vline(xintercept=1.15, linetype="longdash", size=1) +
          theme_bw() + facet_grid(Smarthouse ~.)
print(plt)
plt <- ggplot(tmp, aes_string(x="Smarthouse", y=SIIT)) +
          geom_boxplot() + theme_bw()
print(plt)
remove(tmp)

## End(Not run)
```

---

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$Area.smooth.AGR, lower=2.5)
```

---

anomPlot *Identifies anomalous individuals and produces longitudinal plots without them and with just them*

Description

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

Usage

```
anomPlot(data, x="xDays+24.16666667", response="Area.smooth.RGR",
         individuals="Snapshot.ID.Tag",
         breaks=seq(12, 36, by=2), vertical.line=NULL,
         groupsFactor=NULL, lower=NULL, upper=NULL,
         start.time=NULL, end.time=NULL, times.factor = "Days",
         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.
x	A character giving the variable to be plotted on the x-axis.
response	A character specifying the response variable that is to be tested and plotted on the y-axis.
individuals	A character giving the name(s) of the factor(s) that define the subsets of the data for which each subset corresponds to the response value for an individual.
breaks	A numeric vector giving the breaks to be plotted on the x-axis scale.
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.factor, 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.factor, during which testing for anomalous values is to occur. If NULL, the interval will end with the last observation.
times.factor	A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels should be numeric values stored as characters.
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 longiPlot.

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 longitudinal plot of the individuals that are not anomalous;
3. outerPlot, an object of class ggplot storing the longitudinal 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, intervalValueCalculate, ggplot2.

Examples

```
data(exampleData)
anomalous <- anomPlot(longi.dat, response="Area.smooth.AGR",
                      lower=2.5, start.time=40,
                      x = "xDays+35.42857143", vertical.line=29,
                      breaks=seq(28, 42, by=2),
                      whichPrint=c("innerPlot"),
                      y.title="Area.smooth.AGR")
```

---

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$Days.diffs <- calcLagged(longi.dat$xDays, operation ="-")
```

---

calcTimes *Calculates for a set of times, the time intervals after an origin time and the position of each with in that time*

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 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")
```

corrPlot *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. Also produced is a matrix plot of all pairwise combinations of the variables. The matrix plot contains a scatter diagram for each pair, as well as the value of the correlation coefficient. The argument pairs.sets can be used to restrict the pairs in the matrix plot to those combinations within each set.

Usage

```
corrPlot(responses, data, which.plots = c("heatmap","matrixplot"),
                        title = NULL, labels = NULL, labelSize = 4,
                        show.sig = FALSE, pairs.sets = NULL, ...)
```

Arguments

responses	A character giving the names of the columns in data containing the variables to be correlated.
data	A data.frame containing the columns of variables to be correlated.
which.plots	A character specifying the plots of the correlations to be produced.
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 matrixplot.
show.sig	A logical indicating whether or not to give asterisks indicating significance on the plot.
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 a matrix plot.
...	allows passing of arguments to other functions

Value

NULL.

Author(s)

Chris Brien

See Also

ggplot2.

Examples

```
## Not run:
data(exampleData)
responses <- c("Area","Area.SV","Area.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
corrPlot(responses, longi.dat, pairs.sets=list(c(1:4),c(5:7)))


## End(Not run)
```

`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)
```

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`.

Value

A `vector` containing the cumulative sum.

Author(s)

Chris Brien

See Also

`cumsum`

Examples

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

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

Description

Add the following factors and covariates to a date frame containing imaging data from the Plant Accelerator: Zones, xZones, SHZones, ZLane, ZMainplots, Subplots and xMainPosn. It checks that the numbers of levels of the factors are consistent with the observed numbers of carts and observations.

Usage

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

Arguments

data	A data.frame to which are to be added the design factors and covariates and which must contain the following columns: Smarthouse, Snapshot.ID.Tag, XDays, xPosn and, if designfactorMethod = "LanePosition", Lane and Position.
insertName	A character giving the name of the column in the data.frame after which the new factors 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 factors Zones, ZLane, Mainplots and Subplots. For LanePosition, it is assumed that (i) Lane can be divided into Zones and ZLane, each with nzones and nlanesperzone levels, respectively, and (ii) Position can be divided into Mainplots and Subplots, each with nmainplotsperlane and nmainplotsperlane levels, respectively. The factor SHZones is formed by combining Smarthouse and Zones and ZMainplots is formed by combining ZLane and Mainplots. For StandardOrder, the factors Zones, ZLane, Mainplots, Subplots are generated in standard order, with the levels of Subplots changing for every observation and the levels of subsequent changing only after all combinations of the levels of the factors 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.
nmainplotsperlane	A numeric giving the number of mainplots in each lane.
nsubplotspermain	A numeric giving the number of subplots in a main plot.

Details

The factors Zones, ZLane, ZMainplots and Subplots are derived for each Smarthouse based on the values of `nzones`, `nlanesperzone`, `nmainplotsperlane`, `nsubplotspermain`, Zones being the blocks in the split-plot 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 SHZones is formed by combining Smarthouse and Zones and the covariates xZones and xMainPosn calculated. The covariate xZones is calculated from Zones and xMainPosn is formed from the mean of xPosn for each main plot.

Value

A `data.frame` including the columns:

1. Smarthouse: factor with levels for the Smarthouse
2. Zones: factor dividing the Lanes into groups, usually of 4 lanes
3. xZones: numeric corresponding to Zones, centred by subtracting the mean of the unique positions
4. SHZones: factor for the combinations of Smarthouse and Zones
5. ZLane: factor for the lanes within a Zone
6. ZMainplots: factor for the main plots within a Zone
7. Subplots: factor for the subplots
8. xMainPosn: numeric for the main-plot positions within a Lane, centred by subtracting the mean of the unique positions

Author(s)

Chris Brien

Examples

```
data(exampleData)
longi.dat <- designFactors(longi.prime.dat, insertName = "xDays",
                           nzones = 1, nlanesperzone = 1, nmainplotsperlane = 10,
                           designfactorMethod="StandardOrder")
```

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

Description

Imaging data for 20 of the plants from an experiment in a Smarthouse in the Plant Accelerator. It is used as a small example in the documentation for `imageData`.

Usage

```
data(exampleData)
```

Format

Four `data.frames`: raw.dat (280 rows by 33 columns), longi.prime.dat (280 rows by 45 columns), longi.dat (280 rows by 63 columns), cart.dat (20 rows by 14 columns).

fitSpline — *Produce the fits from a natural cubic smoothing spline applied to a response in a* `data.frame`

Description

Uses `smooth.spline` to fit a spline to all the values of `response` stored in `data`.

The amount of smoothing can be controlled by `df`. If `df = NULL`, the amount of smoothing is controlled by the default arguments and those you supply for `smooth.spline`. The method of Huang (2001) for correcting the fitted spline for estimation bias at the end-points will be applied if `correctBoundaries` is `TRUE`.

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

By default, `smooth.spline` will issue an error if there are not at least four distinct x-values. On the other hand, `fitSplines` issues a warning and sets all smoothed values and derivatives to `NA`. The handling of missing values in the observations is controlled via `na.x.action` and `na.y.action`.

Usage

```
fitSpline(data, response, x, df=NULL, smoothing.scale = "identity",
          correctBoundaries = FALSE,
          deriv=NULL, suffices.deriv=NULL, RGR=NULL, AGR=NULL,
          na.x.action="exclude", na.y.action = "exclude", ...)
```

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.
x	A character giving the name of the column in data that contains the values of the predictor variable.
df	A numeric specifying 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 is controlled by the default arguments for and those that you supply to smooth.spline.
smoothing.scale	A character giving the scale on which smoothing is to be performed. The two possibilites are "identity", for directly smoothing the observed response, and "logarithmic", for scaling the log-transformed response.
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 deriv must be NULL for correctBoundaries to be set to TRUE.
deriv	A numeric specifying one or more orders of derivatives that are required.
suffices.deriv	A character giving the characters to be appended to the names of the derivatives.
RGR	A character giving the character to be appended to the smoothed response to create the RGR name, but only when smoothing.scale is identity. When smoothing.scale is identity: (i) if RGR is not NULL deriv must include 1 so that the first derivative is available for calculating the RGR; (ii) if RGR is NULL, the RGR is not calculated from the AGR. When smoothing.scale is logarithmic, the RGR is the backtransformed first derivative and so, to obtain it, merely include 1 in deriv and any suffix for it in suffices.deriv.
AGR	A character giving the character to be appended to the smoothed response to create the AGR name, but only when smoothing.scale is logarithmic. When smoothing.scale is logarithmic: (i) if AGR is not NULL, deriv must include 1 so that the first derivative is available for calculating the AGR; (ii) If AGR is NULL, the AGR is not calculated from the RGR. When smoothing.scale is identity, the AGR is the first derivative and so, to obtain it, merely include 1 in deriv and any suffix for it in suffices.deriv.
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 data.frame containing x and the fitted smooth. The names of the columns will be the value of x and the value of response with .smooth appended. 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 it will have the same order of x as 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 with .smooth.dvf appended, where f is the order of the derivative, or the value of response with .smooth. 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.

Author(s)

Chris Brien

References

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

See Also

splitSplines, smooth.spline, predict.smooth.spline, splitContGRdiff

Examples

```
data(exampleData)
fit <- fitSpline(longi.dat, response="Area", , x="xDays", df = 4,
                 deriv=c(1,2), suffices.deriv=c("AGRdv","Acc"))
```

---

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

Description

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

Usage

```
getDates(responses, times.factor = "Days", data, which.times,
         suffix = NULL, include.times.factor = FALSE,
         include.individuals = FALSE, individuals = "Snapshot.ID.Tag")
```

Arguments

responses	A character giving the names of the columns in data whose values are to be subsetted.
times.factor	A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels will be used to identify the subset and should be numeric values stored as characters.
data	A data.frame containing the column from which the growth rates are to be calculated.
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.
include.times.factor	A logical indicating whether or not to include the times.factor 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.
individuals	A character giving the name of the column in data containing an identifier for each individual.

Value

A data.frame containing the subset of responses ordered by as many of the initial columns as are required to uniquely identify each row (see order for more information). The names of the columns for responses and times.factor 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)
AreaLast <- getDates("Area.smooth", data = longi.dat,
                     which.times = c(42), suffix = "last")
```

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

`intervalGRaverage`, `intervalGRdiff`, `splitContGRdiff`, `splitSplines`, `calcLagged`

Examples

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

`imagetimesPlot` *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

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

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 `ggplot2` function. It is created by calling the `list` function with a `ggplot2` function call for each element.

Value

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

Author(s)

Chris Brien

See Also

ggplot2, calcTimes.

Examples

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

---

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",
            imagetimesPlot = 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.
imagetimesPlot	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 imagetimesPlot. 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, imagetimesPlot

Examples

```
## Not run:
raw.0169.dat <- importExcel(file = "0169 analysis_20140603.xlsx",
                            startTime = "2013-05-23 8:00 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")
raw.19.dat <- suppressWarnings(importExcel(file = "./data/raw19datarow.csv",
                                           cartId = "Snapshot.ID.Tags",
                                           startTime = "06/10/2017 0:00 AM",
                                           timeFormat = "%d/%m/%Y %H:M",
                                           labsCamerasViews = camview.labels,
                                           imagetimesPlot = FALSE))

## End(Not run)
```

---

intervalGRaverage *Calculates the growth rates for a specified time interval 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

```
intervalGRaverage(responses, individuals = "Snapshot.ID.Tag",
                  which.rates = c("AGR","RGR"), suffices.rates=c("AGR","RGR"),
                  start.time, end.time, times.factor = "Days", suffix.interval,
                  data, sep=".", na.rm=TRUE)
```

Arguments

`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(s) of the `factor`(s) that define the subsets of the `data` for which each subset corresponds to the `responses` for an individual.
`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 `"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`.
`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 the growth rate is to be calculated.
`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 the growth rate is to be calculated.
`times.factor`	A `character` giving the name of the column in `data` containing the factor for times at which the data was collected. Its levels will be used in calculating growth rates and should be numeric values stored as characters.
`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.
`data`	A `data.frame` containing the columns from which the growth rates are to be calculated.
`sep`	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.factor`. 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

`intervalGRdiff`, `intervalWUI`, `splitValueCalculate`, `getDates`, `GrowthRates`, `splitSplines`, `splitContGRdiff`

Examples

```
data(exampleData)
longi.dat <- splitSplines(longi.dat, response="Area", x="xDays",
                          INDICES = "Snapshot.ID.Tag",
                          df = 4, deriv=1, suffices.deriv="AGRdv", RGR="RGRdv")
Area.smooth.GR <- intervalGRaverage("Area.smooth", which.rates = c("AGR","RGR"),
                                    suffices.rates = c("AGRdv","RGRdv"),
                                    start.time = 31, end.time = 35,
                                    suffix.interval = "31to35",
                                    data = longi.dat)
```

---

intervalGRdiff *Calculates the growth rates for a specified time 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

```
intervalGRdiff(responses, individuals = "Snapshot.ID.Tag",
               which.rates = c("AGR","PGR","RGR"), suffices.rates=NULL,
               times.factor = "Days", start.times, end.times, suffix.interval,
               data)
```

Arguments

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(s) of the factor(s) that define the subsets of the data for which each subset corresponds to the responses for an individual.
which.rates	A character giving the growth rates that are to be calculated. It should be a combination "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.
times.factor	A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels will be used in calculating growth rates and should be numeric values stored as characters.
start.times	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 the growth rate is to be calculated.
end.times	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 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.
data	A data.frame containing the column from which the growth rates are to be calculated.

Details

The AGR is calculated as the difference between the values of response at the end.times and start.times divided by the difference between end.times and start.times. The PGR is calculated as the ratio of response at the end.times to that at start.times and the ratio raised to the power of the reciprocal of the difference between end.times and start.times. 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.times and start.times divided by the difference between end.times and start.times.

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

intervalGRaverage, intervalWUI, getDates, GrowthRates, splitSplines, splitContGRdiff

Examples

```
data(exampleData)
Area.smooth.GR <- intervalGRdiff("Area.smooth", which.rates = c("AGR","RGR"),
                                 start.times = 31, end.times = 35,
                                 suffix.interval = "31to35",
                                 data = longi.dat)
```

---

intervalPVA *Selects a subset of variables observed within a specified time interval using Principal Variable Analysis (PVA)*

Description

Principal Variable Analysis (PVA) (Cummings, 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

```
intervalPVA(responses, data, times.factor = "Days", start.time, end.time,
            nvarselect = NULL, p.variance = 1, include = NULL,
            plot = TRUE, ...)
```

Arguments

responses	A character giving the names of the columns in data from which the variables are to be selected.
data	A data.frame containing the columns of variables from which the selection is to be made.
times.factor	A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels will be used to identify the subset and should be numeric values stored as characters.
start.time	A numeric giving the time, in terms of a level of times.factor, 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 levels of times.factor, 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. Wood (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

PVA, rcontrib

Examples

```
data(exampleData)
responses <- c("Area","Area.SV","Area.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
results <-  intervalPVA(responses, longi.dat,
                        start.time = "31", end.time = "31",
                        p.variance=0.9, plot = FALSE)
```

intervalValueCalculate

Calculates a single value that is a function of an individual's values for a response 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 that corresponds to the calculated value of the function.

Usage

```
intervalValueCalculate(response, weights=NULL, individuals = "Snapshot.ID.Tag",
                       FUN = "max", which.obs = FALSE, which.levels = NULL,
                       start.time=NULL, end.time=NULL, times.factor = "Days",
                       suffix.interval=NULL, data, sep=".", na.rm=TRUE, ...)
```

Arguments

response	A character giving the name of the column in data from which the values of FUN are to be calculated.
weights	A character giving the name of the column in data containing the weights to be supplied as w to FUN.
individuals	A character giving the name(s) of the factor(s) that define the subsets of the data for which each subset corresponds to the response value for an individual.
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 corresponding to the value of the function, instead of the value of the function itself.
which.levels	A character giving the name of the factor whose levels are to be identified as the level of the observation whose value matches the value of the function.
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 the growth rate is to be calculated. If start.time is NULL, the interval will start with the first observation.
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 the growth rate is to be calculated. If end.time is NULL, the interval will end with the last observation.
times.factor	A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels should be numeric values stored as characters.

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.

data
: A data.frame containing the column from which the function is to be calculated.

na.rm
: A logical indicating whether NA values should be stripped before the calculation proceeds.

sep
: 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, a column with the values of the function for the individuals, and a column with the values of the times.factor. 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.

Author(s)

Chris Brien

See Also

intervalGRaverage, intervalGRdiff, intervalWUI, splitValueCalculate, getDates

Examples

```
data(exampleData)
Area.smooth.max <- intervalValueCalculate("Area.smooth",
                                          start.time = 31, end.time = 35,
                                          suffix.interval = "31to35",
                                          data = longi.dat)
```

---

intervalWUI *Calculates water use indices (WUI) over a specified time interval to a data.frame*

Description

Calculates the Water Use Index (WUI) between two time points for a set of responses.

Usage

```
intervalWUI(responses, water.use = "Water.Use",
            individuals = "Snapshot.ID.Tag", times.factor = "Days",
            start.times, end.times, suffix.interval = NULL,
            data, include.total.water = FALSE, na.rm = FALSE)
```

Arguments

responses	A character giving the names of the columns in data from which the growth rates are to be calculated.
water.use	A character giving the names of the column in data which contains the water use values.
individuals	A character giving the name(s) of the factor(s) that define the subsets of the data for which each subset corresponds to the responses for an individual.
times.factor	A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels will be used in identifying the intervals and should be numeric values stored as characters.
start.times	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 the growth rate is to be calculated.
end.times	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 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.
data	A data.frame containing the column from which the growth rates are to be calculated.
include.total.water	A logical indicating whether or not to include a column in the results for the total of water.use for the interval for each individual.
na.rm	A logical indicating whether NA values should be stripped before the calculation proceeds.

Details

The WUI is calculated as the difference between the values of a response at the end.times and start.times divided by the sum of the water use after start.times until end.times. Thus, the water use up to start.times is not included.

Value

A data.frame containing the WUIs, the name of each column being the concatenation of one of responses, WUI and, if not NULL, suffix.interval, the three components being separated by a full stop. If the total water is to be included, the name of the column will be the concatenation of water.use, Total and the suffix, each separated by a full stop('.').

Author(s)

Chris Brien

See Also

intervalGRaverage, intervalGRdiff, splitValueCalculate, getDates, GrowthRates

Examples

```
data(exampleData)
Area.smooth.WUI <- intervalWUI("Area", water.use = "Water.Loss",
                               start.times = 31, end.times = 35,
                               suffix = "31to35",
                               data = longi.dat, include.total.water = TRUE)
```

---

longiPlot *Plots longitudinal data from a Lemna Tec Scananalyzer*

Description

Produce profile or longitudinal plots of the data from a Lemna Tec Scananalyzer using ggplot. A line is drawn for the data for each Snapshot.ID.Tag and the plot can be facetted so that a grid of plots is produced.

Usage

```
longiPlot(data, x = "xDays+44.5", response = "Area", individuals="Snapshot.ID.Tag",
          x.title = "Days", y.title = "Area (1000 pixels)", title = NULL,
          facet.x = "Treatment.1", facet.y =  "Smarthouse", labeller = NULL,
          colour = "black", colour.column=NULL, colour.values=NULL,
          alpha = 0.1, ggplotFuncs = NULL, printPlot = TRUE)
```

Arguments

data	A data.frame containing the data to be plotted.
x	A character giving the variable to be plotted on the x-axis.
response	A character specifying the response variable that is to be plotted on the y-axis.
individuals	A character giving the name(s) of the factor(s) that define the subsets of the data for which each subset corresponds to the response values for an individual.
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 columns is not wanted.
labeller	A ggplot2 function for labelling the facets of a plot produced using the ggplot2 function. For more information see 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 line) that must be overplotted to give a solid cover.
ggplotFuncs	A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element.
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

ggplot2, ggplot2::labeller.

Examples

```
data(exampleData)
longiPlot(data = longi.dat, response = "Area.smooth")

plt <- longiPlot(data = longi.dat, response = "Area.smooth", x.title = "DAP",
                 y.title = "Area.smooth", x="xDays+35.42857143", printPlot=FALSE)
plt <- plt + 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))
print(plt)

longiPlot(data = longi.dat, response = "Area.smooth", x.title = "DAP",
          y.title = "Area.smooth", x="xDays+35.42857143",
```

```
        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))))
```

`longitudinalPrime` *Selects a set variables to be retained in a data frame of longitudinal data*

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 10000 to convert them from pixels to kilopixels. Also added are factors and explanatory variates that might be of use in an analysis.

Usage

```
longitudinalPrime(data, cartId = "Snapshot.ID.Tag",
                  imageTimes = "Snapshot.Time.Stamp",
                  timeAfterStart = "Time.after.Planting..d.",
                  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,
                  calcWaterLoss = TRUE, pixelsPERcm)
```

Arguments

`data` A `data.frame` containing the columns specified by `cartId`, `imageTimes`, `timeAfterStart`, `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 `longitudinalPrime` 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.

`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 corresponding 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.

`calcWaterLoss` 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`.

`pixelsPERcm` A `numeric` giving the number of pixels per cm for the images. *No longer used.*

Details

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

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. Days: factor for the number of Days After Planting (DAP)
5. `cartId`: unique code for each cart
6. `imageTimes`: time at which an image was taken in POSIXct format
7. Reps: factor indexing the replicates for each combination of the factors in `idcolumns` (calculated)
8. xPosn: numeric for the Positions within a Lane (calculated)
9. Hour: hour of the day, to 2 decimal places, at which the image was taken (calculated)
10. xDays: numeric for the DAP that is centred by subtracting the mean of the unique days (calculated)
11. `idcolumns`: the columns listed in `idcolumns` that have been converted to factors
12. Weight.Before: weight of the pot before watering (only if `calcWaterLoss` is `TRUE`)
13. Weight.After: weight of the pot after watering (only if `calcWaterLoss` is `TRUE`)
14. Water.Amount: the weight of the water added (= Water.After - Water.Before) (calculated)
15. Water.Loss: the difference between Weight.Before for the current imaging and the Weight.After for the previous imaging (calculated unless `calcWaterLoss` is `FALSE`)
16. Area: the Projected.Shoot.Area..pixels. divided by 1000 (calculated)
17. Area.SV1: the Projected.Shoot.Area from Side View 1 divided by 1000 (calculated)
18. Area.SV2: the Projected.Shoot.Area from Side View 2 divided by 1000 (calculated)
19. Area.TV: the Projected.Shoot.Area from Top View divided by 1000 (calculated)
20. Boundary.To.Area.Ratio.SV1
21. Boundary.To.Area.Ratio.SV2
22. Boundary.To.Area.Ratio.TV
23. Caliper.Length.SV1
24. Caliper.Length.SV2
25. Caliper.Length.TV
26. Compactness.SV1 from Side View 1
27. Compactness.SV2 from Side View 2
28. Compactness.TV: from Top View
29. Convex.Hull.Area.SV1: area of Side View 1 Convex Hull divided by 1000 (calculated)
30. Convex.Hull.Area.SV2: area of Side View 2 Convex Hull divided by 1000 (calculated)
31. Convex.Hull.TV: Convex.Hull.Area.TV divided by 1000 (calculated)
32. Center.Of.Mass.Y.SV1: Centre of Mass from Side View 1
33. Center.Of.Mass.Y.SV2: Centre of Mass from Side View 2
34. Max.Dist.Above.Horizon.Line.SV1: the Max.Dist.Above.Horizon.Line.SV1 converted to cm using `pixelsPERcm` (calculated)
35. Max.Dist.Above.Horizon.Line.SV2: the Max.Dist.Above.Horizon.Line.SV2 converted to cm using `pixelsPERcm` (calculated)

Author(s)

Chris Brien

Examples

```
data(exampleData)
longiPrime.dat <- longitudinalPrime(data=raw.dat, smarthouse.lev=1)

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

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

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

---

probeDF — *Compares, for a set of specified values of df, a response and the smooths of it, possibly along with growth rates calculated from the smooths*

Description

Takes a `response` and, for each `individual`, uses `splitSplines` to smooth its values for each individual using the degrees of freedom values in `df`. Provided `get.rates` is `TRUE`, both the Absolute Growth Rates (AGR) and the Relative Growth Rates (RGR) are calculated for each smooth, either using differences or first derivatives. A combination of the unsmoothed and smoothed values, as well as the AGR and RGR, can be plotted for each value in `df`. Note that the arguments that modify the plots apply to all plots that are produced. The handling of missing values is controlled via `na.x.action` and `na.y.action`

Usage

```
probeDF(data, response = "Area", xname="xDays", individuals="Snapshot.ID.Tag",
        na.x.action="exclude", na.y.action = "exclude",
        df, smoothing.scale = "identity", correctBoundaries = FALSE,
        get.rates = TRUE, rates.method="differences",
        times.factor = "Days", x = NULL, x.title = NULL,
```

```
facet.x = "Treatment.1", facet.y =   "Smarthouse", labeller = NULL,
colour = "black", colour.column=NULL, colour.values=NULL, alpha = 0.1,
which.traits = c("response", "AGR", "RGR"),
which.plots = "smoothedonly",
deviations.boxplots = "none",
ggplotFuncs = NULL,
...)
```

Arguments

data	A data.frame containing the data.
response	A character specifying the response variable to be supplied to smooth.spline and that is to be plotted on the y-axis.
xname	A character giving the name of the numeric that contains the values of the predictor variable to be supplied to smooth.spline.
individuals	A character giving the name(s) of the factor(s) that define the subsets of the data for which each subset corresponds to the response values for an individual.
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.
df	A numeric specifying the set of degrees of freedom to be probed.
smoothing.scale	A character giving the scale on which smoothing is to be performed. The two possibilites are "identity", for directly smoothing the observed response, and "logarithmic", for smoothing the log-transformed response.
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 if rates.method is set to "derivatives" then it is not possible to have correctBoundaries set to TRUE.
get.rates	A logical specifying whether or not the growth rates (AGR and RGR) are to be computed and stored.

rates.method — A character specifying the method to use in calculating the growth rates. The two possibilities are "differences" and "derivates".

times.factor — A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels will be used in calculating growth rates and should be numeric values stored as characters.

x — A character giving the variable to be plotted on the x-axis. If x is NULL then xname is used.

x.title — Title for the x-axis. If NULL then set to times.factor.

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 columns is not wanted.

labeller — A ggplot2 function for labelling the facets of a plot produced using the ggplot2 function. For more information see 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 line) that must be overplotted to give a solid cover.

which.traits — A character giving the traits that are to be produced. One of more of response, AGR and RGR. If all, all three traits are produced. The unsmoothed growth rates are not calculated if only smoothed plots are requested.

which.plots — A character giving the plots that are to be produced. If none, no plots are produced. If smoothedonly, plots of the smoothed traits are plotted. If bothseparately, plots of the unsmoothed trait followed by the smoothed trait are produced for each trait. If compare, a combined plot of the unsmoothed trait and the smoothed trait is produced for each value of df.

deviations.boxplots — A character specifying whether boxplots of the absolute and/or relative deviations of the values of a trait from their smoothed values are to be produced (observed - smoothed). If none, no plots are produced. The argument which.traits controls the traits for which boxplots are produced.

ggplotFuncs — A list, each element of which contains the results of evaluating a ggplot2 function. It is created by calling the list function with a ggplot2 function call for each element. Note that these functions are applied to all three plots produced.

... — allows passing of arguments to longiPlot.

Value

A data.frame containing individuals, times.factor, facet.x, facet.y, xname, response, and, for each df, the smoothed response, the AGR and the RGR. It is returned invisibly. The names of the new data are constructed by joining elements separated by full stops (.). In all cases, the last element is the value of df. For the smoothed response, the other elements are response and "smooth"; for AGR and RGR, the other elements are the name of the smoothed response and either "AGR" or "RGR".

Author(s)

Chris Brien

See Also

splitSplines, splitContGRdiff, smooth.spline, ggplot2.

Examples

```
data(exampleData)
vline <- list(ggplot2::geom_vline(xintercept=20, linetype="longdash", size=1),
              ggplot2::scale_x_continuous(breaks=seq(12, 36, by=2)))
probeDF(data = longi.dat, response = "Area", df = c(4,7), x="xDays+24.16666667",
        ggplotFuncs=vline)
```

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

Description

Principal Variable Analysis (PVA) (Cummings, 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(responses, data, nvarselect = NULL, p.variance = 1, include = NULL,
    plot = TRUE, ...)
```

Arguments

responses	A character giving the names of the columns in data from which the variables are to be selected.
data	A data.frame containing the columns of 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. Wood (2007) Dimension reduction via principal variables. *Computational Statistics and Data Analysis*, **52**, 550–565.

See Also

intervalPVA, rcontrib

Examples

```
data(exampleData)
responses <- c("Area","Area.SV","Area.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
results <-  PVA(responses, longi.dat, 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 (Cummings, 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(responses, data, include = NULL)
```

Arguments

`responses`	A `character` giving the names of the columns in `data` from which the correlation measure is to be calculated.
`data`	A `data.frame` containing the columns of variables 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.

Value

A `numeric` giving the correlation measures.

Author(s)

Chris Brien

References

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

See Also

`PVA`, `intervalPVA`

Examples

```
data(exampleData)
responses <- c("Area","Area.SV","Area.TV", "Image.Biomass", "Max.Height","Centre.Mass",
               "Density", "Compactness.TV", "Compactness.SV")
h <-  rcontrib(responses, longi.dat, include = "Area")
```

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

Description

The data is from an experiment in a Smarthouse in the Plant Accelerator. It is described in Al-Tamimi et al. (2016). It is used in imageData-package as an executable example to illustrate the use of imageData.

Usage

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

Format

A data.frame containing 14784 observations on 33 variables.

Source

It will be made available on Dryad

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.

---

splitContGRdiff *Adds the growth rates calculated continuously over time for subsets of a response to a* data.frame

---

Description

Uses AGRdiff, PGR and RGRdiff to calculate growth rates continuously over time for a subset of the values of response and stores the results in data. The subsets are those values with the same levels combinations of the factors listed in INDICES.

Usage

```
splitContGRdiff(data, responses, INDICES,
                which.rates = c("AGR","PGR","RGR"), suffices.rates=NULL,
                times.factor = "Days")
```

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.
INDICES	A character giving the name(s) of the factor(s) that define the subsets of response for which growth rates are to be calculated continuously. If the columns corresponding to INDICES are not factor(s) then they will be coerced to factor(s). The subsets are formed using by.
which.rates	A character giving the growth rates that are to be calculated. It should be a combination "AGR", "PGR" and "RGR".
times.factor	A character giving the name of the column in data containing the factor for times at which the data was collected. Its levels will be used in calculating growth rates and should be numeric values stored as characters.
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.

Value

A data.frame containing data to which has been added a column for the differences between the times.factor, if it is not already in data, and columns with growth rates. The name of the column for times.factor differences will be the times.factor with ".diff" appended and, for each of the growth-rate columns will be the value of response with one of ".AGR", ".PGR" or "RGR" or the corresponding value from suffices.GR appended.

Author(s)

Chris Brien

See Also

fitSpline, splitSplines

Examples

```
data(exampleData)
longi.dat <- splitContGRdiff(longi.dat, response="Area.smooth",
                             INDICES = "Snapshot.ID.Tag", which.rates=c("AGR", "RGR"))
```

splitSplines *Adds the fits after fitting a natural cubic smoothing spline to subsets of a response to a* data.frame

Description

Uses fitSpline to fit a spline to a subset of the values of response and stores the fitted values in data. The subsets are those values with the same levels combinations of the factors listed in INDICES and the degrees of smoothing is controlled by df. The derivatives of the fitted spline can also be obtained, as can the Relative Growth Rates (RGR).

By default, smooth.spline will issue an error if there are not at least four distinct x-values. On the other hand, fitSpline issues a warning and sets all smoothed values and derivatives to NA. The handling of missing values in the observations is controlled via na.x.action and na.y.action.

Usage

```
splitSplines(data, response, x, INDICES, df = NULL, smoothing.scale = "identity",
             correctBoundaries = FALSE,
             deriv = NULL, suffices.deriv=NULL, RGR=NULL, AGR=NULL, sep=".",
             na.x.action="exclude", na.y.action = "exclude", ...)
```

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.
x	A character giving the name of the column in data that contains the values of the predictor variable.
INDICES	A character giving the name(s) of the factor(s) that define the subsets of response that are to be smoothed separately. If the columns corresponding to INDICES are not factor(s) then they will be coerced to factor(s). The subsets are formed using split.
df	A numeric specifying 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, ordinary leave-one-out cross-validation is used to determine the amount of smooth.
smoothing.scale	A character giving the scale on which smoothing is to be performed. The two possibilites are "identity", for directly smoothing the observed response, and "logarithmic", for scaling the log-transformed response.
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 deriv must be NULL for correctBoundaries to be set to TRUE.
deriv	A numeric specifying one or more orders of derivatives that are required.

suffices.deriv A character giving the characters to be appended to the names of the derivatives.

RGR A character giving the character to be appended to the smoothed response to create the RGR name, but only when smoothing.scale is identity. When smoothing.scale is identity: (i) if RGR is not NULL deriv must include 1 so that the first derivative is available for calculating the RGR; (ii) if RGR is NULL, the RGR is not calculated from the AGR. When smoothing.scale is logarithmic, the RGR is the backtransformed first derivative and so, to obtain it, merely include 1 in deriv and any suffix for it in suffices.deriv.

AGR A character giving the character to be appended to the smoothed response to create the AGR name, but only when smoothing.scale is logarithmic. When smoothing.scale is logarithmic: (i) if AGR is not NULL, deriv must include 1 so that the first derivative is available for calculating the AGR; (ii) If AGR is NULL, the AGR is not calculated from the RGR. When smoothing.scale is identity, the AGR is the first derivative and so, to obtain it, merely include 1 in deriv and any suffix for it in suffices.deriv.

sep A character giving the separator to use when the levels of INDICES are combined. This is needed to avoid using a character that occurs in a factor to delimit levels when the levels of INDICES are combined to identify subsets.

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 data.frame containing data to which has been added a column with the fitted smooth, the name of the column being response with .smooth appended to it. If deriv is not NULL, columns containing the values of the derivative(s) will be added to data; the name each of these columns will be the value of response with .smooth.dvf appended, where f is the order of the derivative, or the value of response with .smooth. 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. Any pre-existing smoothed and derivative columns in data will be replaced. The ordering of the data.frame for the x 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 x are retained, they will occur at the bottom of each subset of INDICES 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

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

See Also

fitSpline, smooth.spline, predict.smooth.spline, splitContGRdiff, split

Examples

```
data(exampleData)
longi.dat <- splitSplines(longi.dat, response="Area", x="xDays",
                          INDICES = "Snapshot.ID.Tag",
                          df = 4, deriv=1, suffices.deriv="AGRdv", RGR="RGRdv")
```

---

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

Description

Splits the values of a response into subsets corresponding individuals and applies a function that calculates a single value to each individual's observations. It includes the ability to calculate the observation that corresponds to the calculated value of the function.

Usage

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

Arguments

response	A character giving the name of the column in data from which the values of FUN are to be calculated.
weights	A character giving the name of the column in data containing the weights to be supplied as w to FUN.
individuals	A character giving the name(s) of the factor(s) that define the subsets of the data for which each subset corresponds to the response value for an individual.

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 corresponding to the value of the function, instead of the value of the function itself.
which.levels	A character giving the name of the factor whose levels are to be identified as the level of the observation whose value matches the value of the function.
data	A data.frame containing the column from which the function is to be calculated.
na.rm	A logical indicating whether NA values should be stripped before the calculation proceeds.
sep	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 the values of the function for the individuals.

Author(s)

Chris Brien

See Also

splitContGRdiff, splitSplines

Examples

```
data(exampleData)
Area.smooth.max <- splitValueCalculate("Area.smooth", data = longi.dat)
```

---

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(responses, data, 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",
                                      "Zones","xZones","SHZones","ZLane",
                                      "ZMainplots","xMainPosn", "Genotype.ID"),
                 by = c("Smarthouse","Zones","ZMainplots"))
```

Arguments

responses	A character giving the names of the columns in data that contain the responses to which the binary operations are to be applied.
data	A data.frame containing the columns specified by treatment.factor, columns.retained and responses.
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 factors 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("Area.smooth.AGR","Area.smooth.RGR")
cols.retained <-  c("Snapshot.ID.Tag","Smarthouse","Lane","Position",
                    "Days","Snapshot.Time.Stamp", "Hour", "xDays",
                    "Zones","xZones","SHZones","ZLane","ZMainplots",
                    "xMainPosn", "Genotype.ID")
longi.SIIT.dat <-
  twoLevelOpcreate(responses, longi.dat, suffices.treatment=c("C","S"),
                   operations = c("-", "/"),
                   suffices.results = c("diff", "SIIT"),
                   columns.retained = cols.retained,
                   by = c("Smarthouse","Zones","ZMainplots","Days"))
longi.SIIT.dat <- with(longi.SIIT.dat,
                           longi.SIIT.dat[order(Smarthouse,Zones,ZMainplots,Days),])
```

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 water divided by the response, unless water is zero in which caseNA is returned.

Author(s)

Chris Brien

Examples

```
data(exampleData)
Area.WUE <- with(longi.dat, WUI(Area.AGR, Water.Loss))
```

Index