Package ‘pvldcurve’

October 5, 2020

Type Package

Title Simplifies the Analysis of Pressure Volume and Leaf Drying Curves

Version 1.2.6

Depends ggplot2, R (>= 2.10)

Suggests knitr, rmarkdown

VignetteBuilder knitr

Description Simplifies the manufacturing, analysis and display of pressure volume and leaf drying curves. From the progression of the curves turgor loss point, osmotic potential, apoplastic fraction as well as minimum conductance and stomatal closure can be derived. Methods adapted from Bartlett, Scoffoni, Sack (2012) <doi:10.1111/j.1461-0248.2012.01751.x> and Sack, Scoffoni, PrometheusWikiContributors (2011) <http://prometheuswiki.org/tiki-index.php?page=Minimum+epidermal+conductance+%28gmin%2C+a.k.a.+cuticular+conductance%29>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

NeedsCompilation no

Author Anna Raesch [aut, cre, cph]

Maintainer Anna Raesch <annaraesch@gmail.com>

Repository CRAN

Date/Publication 2020-10-05 11:30:02 UTC

R topics documented:

`ApplyCombMod` *Apply a combined exponential and linear model*

Description

a non linear model combining an exponential and a linear fit is applied to the data using the Gauss-Newton algorithm of nls. starting values are calculated based on the data. Weights are applied to the model based on the estimated insecurity of the data quality.

Usage

```
ApplyCombMod(data, y = "y", x = "x")
```

Arguments

`data`	data frame containg x and y data to which the model is ought to be applied to
`y`	name of column in data containing y data
`x`	name of column in data containing x data

Value

model parameters

`ApplyCombMod2` *Apply a combined exponential and linear model*

Description

a non linear model combining an exponential and a linear fit is applied to the data using nls. starting values are calculated based on the data. Constraints are applied to the model based on the known constraints of the aimed model.

Usage

```
ApplyCombMod2(data, y = "y", x = "x")
```

Arguments

`data`	data frame containg x and y data to which combined exponentail and linear model is ought to be applied to
`y`	name of column in data containing y data
`x`	name of column in data containing x data

Value

model parameters

`Conductance` *Leaf Conductance*

Description

Calculates mole-based or concentraction-based conductance (stomatal or minimal conductance) (mmol s^-1 m^-2 or mm s^-1) of the double-sided leaf area by experimental weight loss data and weather data

Usage

```
Conductance(data, sample = "sample", fresh.weight = "fitted.fw",
  date.and.time = "date.and.time", leaf.area = "leaf.area",
  humidity = "humidity", temperature = "temperature",
  atmospheric.pressure = 101.35, driving.force = "mole")
```

Arguments

`data`	data frame, with columns of equal length containg at least columns with time (and date) of the fresh weight measurements,the measured fresh weights (g) and the single-sided leaf area (cm^2) of the sample as well as the average relative humidity (%) and temperature (degree Celsius) during the measurement intervals. The data is to be ordered chronologically by sample. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample ID; default: "sample"
`fresh.weight`	optional name of the column in data containing the numeric fresh weight values (g); default: "fitted.fw" (fresh weight corrected by noises as outputted for leaf drying curves by the function FittedFW)
`date.and.time`	optional name of the column in data containing the time of the fresh weight measurements as class POSIXct; default: "date.and.time"
`leaf.area`	optional name of the column in data containing the numeric single-sided leaf area values (cm^2); default: "leaf.area"
`humidity`	optional name of the column in data containing the numeric humidity values (%); default: "humidity"
`temperature`	optional name of the column in data containing the numeric temperature values (degree Celsius); default: "temperature"
`atmospheric.pressure`	optional, giving the numeric atmospheric pressure in kPA, default = 101.325 (atmospheric pressure at sea level)
`driving.force`	optional; possible values: mole or conc; defines whether conductance is expressed on the basis of a mole fraction-based (default) or a concentration-based driving force

Details

Calculates mole-based conductance (mmol s^-1 m^-2) as:

$$g = T/VPD$$

whereas T = transpiration (mmol s^-1 m^-2) is calculated as:

$$T = \Delta FM * 1000 * (\Delta t * 60 * LA * 2/10000 * 18.01528)^-1$$

whereas ΔFM = fresh matter reduction (g), Δt = time interval (min), LA = single-sided leaf area (cm^2) and VPD = vapor pressure deficit (mol * mol^-1) is calculated as:

$$VPD = (1 - RH/100) * (VPsat/AP)$$

whereas RH = relative humidity (%), VPsat = saturation vapor pressure (kPA), AP = atmospheric pressure (kPA), whereas:

$$VPsat = 0.61121 * exp((18.678 - T/234.5)(T * 257.14 + T))$$

where T = air temperature (degree Celsius)

Concentration based conductance (mm s^-1) is derived from mole-based conductance g(mol) as:

$$g(conc) = g(mol) * R * (T + 273.15)/AP/1000$$

whereas: R = gas constant (8.3144598 J (mol * K)^-1) and T = absolute temperature (degree Celsius)

Value

The original data frame extended by a numeric column with the mole-based or the concentration-based conductance (mmol s^-1 m^-2, mm s^-1) of the double-sided leaf area (conductance). The first value of each sample is NA since conductance values are computed from row i and i-1

Examples

```
# get example data
weight_loss_data <- leaf_drying_data
weather_data <- weather_data
df <- WeatherAllocation(weight_loss_data, weather_data)  # allocate weather to weight loss data

# extend the data frame by mole-based conductance values
df_with_conductance <- Conductance(df, fresh.weight = "fresh.weight")

# extend the data frame by concentration-based conductance values
df_with_conductance <- Conductance(df, fresh.weight = "fresh.weight", driving.force = "conc")

# calculate with atmospheric pressure of 99.8 kPA
df_with_conductance <- Conductance(df, fresh.weight = "fresh.weight", atmospheric.pressure = 99.8)
```

`ExtractFitParam` *Extracts the fitting parameters from results list*

Description

Extracts the coefficients and confidence intervals from the fitting results of the functions analysing the pressure volume curve (TurgorLossPoint, OsmoticPot and ModElasticity) or the functions analysing the leaf drying curve (StomatalClosure and Gmin)

Usage

```
ExtractFitParam(result_list)
```

Arguments

`result_list`	output list from the functions TurgorLossPoint, OsmoticPot, ModElasticity, StomatalClosure or Gmin

Value

data frame containing the coefficients and the 0.95 confidence interval of the coefficients from the fit

---

`ExtractParam` *Extracts parameters from result list*

Description

Extracts the curve parameters from the result lists of the functions analysing the pressure volume curve (TurgorLossPoint, OsmoticPot and ModElasticity) or the functions analysing the leaf drying curve (StomatalClosure and Gmin)

Usage

```
ExtractParam(result_list)
```

Arguments

`result_list`	output list from the functions TurgorLossPoint, OsmoticPot, ModElasticity, StomatalClosure or Gmin

Value

data frame containing the results from the curve analysis only, depending on the function used, relative water deficit at turgor loss point (rwd.tlp), water potential at turgor loss point (water.pot.tlp), apoplastic fraction (apo.fract), osmotic potential at full saturation (osmotic.pot.full.sat), modulus of elasticity (modulus.elasticity), time since start of desiccation of stomatal closure (stom.clos.time), relative water deficit at stomatal closure (stom.clos.rwd), minimum conductance determined by the intercept of the leaf drying curve fit with the point of stomatal closure (stom.clos.gmin), by taking the mean from all minimum conductance values (mean.gmin) or minimum conductance determined by extrapolation of minimum conductance curve linearly to full saturation (gmin.full.sat)

Examples

```
# use pressure volume data provided by package
pv_data <- pressure_volume_data

# do pressure volume curve analysis
pv_data <- RelativeWaterDeficit(pv_data)
results <- OsmoticPot(pv_data, graph = FALSE)

# extract curve values
ExtractParam(results)
```

FitLeafArea *Leaf area fitting*

Description

Fits randomly measured leaf area values linearly to fresh weight values. Useful if the leaf area changes during a measurement series but is only randomly measured.

Usage

```
FitLeafArea(data, sample = "sample", fresh.weight = "fresh.weight",
  leaf.area = "leaf.area")
```

Arguments

`data`	data frame, with columns of equal length, containing at least columns with the the fresh.weight (g) and the leaf.area (cm^2) values, ordered by sample by descending fresh weight. A column containing the sample IDs is optionally required if several samples were measured.At least 3 leaf area values are required.
`sample`	string, optional name of the column in data containing the sample ID, default: "sample"
`fresh.weight`	optional name of the column in data containing the numeric fresh weight values (g); default: "fresh.weight"
`leaf.area`	optional name of the column in data containing the numeric single-sided leaf area values (cm^2); default: "leaf.area"

Details

fits given leaf area values linearly to the respective fresh weight values and calculates leaf area values for the fresh weight values based on the fit

Value

the original data frame extended by a numeric column containing the fitted leaf area values (leaf.area.fitted)

Examples

```
# get example data
df <- data.frame(
  sample = c(as.integer(rep(1, times = 6))),
  fresh.weight = c(1.23, 1.19, 1.15, 1.12, 1.09, 1.0),
  leaf.area = c(10.5, NA, NA, 9.8, NA, 8.4))
# fit leaf area
df_new <- FitLeafArea(df)
```

`FittedFW` *Correct continuous fresh weight measurements*

Description

Corrects fresh weight, measured continuously on a desiccating leaf for determination of minimum conductance, by fitting the fresh weight values to a combined exponential and linear model

Usage

```
FittedFW(data, sample = "sample", fresh.weight = "fresh.weight",
  time.since.start = "time.since.start", graph = TRUE,
  show.legend = TRUE)
```

Arguments

`data`	data frame, at least with a column containing numeric fresh weight (g) and time since start (min) values, ordered by sample by descending fresh.weight. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample IDs, default: "sample"
`fresh.weight`	optional name of the column in data containing the fresh weight values (g), default: "fresh weight"
`time.since.start`	optional name of the column in data containing the time since start (min) values, default: "time.since.start"
`graph`	set FALSE if no plots are to be returned
`show.legend`	set FALSE if no legend is to be be shown in the plots

Details

Determination of minimum conductance via a leaf drying curve requires fresh weight to be measured continuously on a desiccating leaf. The fresh weight values are then used to calculate leaf conductance. If several leaves are measured on one scale, the measurements are prone to noises, which influence conductance values largely. Here, the fresh weight data is corrected for noises by fitting it to a combined linear and exponential model using the port algorithm of nls().
Before using this function, check the raw data for an initial plateau. If the exponential decline does not onset directly, fitting might not succeed.

Value

the original data frame (data) extended by a numeric column containing the fitted fresh weight values ("fitted.fw")

Examples

```
# get example data
df <- TimeSinceStart(leaf_drying_data)
# remove plateauing data
df <- df[df$fw.plateau != "yes",]
# extend the data frame by saturated fresh weight
df <- FittedFW(df)
```

`FWSaturated` *Saturated fresh weight estimation*

Description

Calculates saturated fresh weight by fitting fresh weight values above the turgor loss point linearly to water potential values.

Usage

```
FWSaturated(data, sample = "sample",
  water.potential = "water.potential", fresh.weight = "fresh.weight",
  dry.weight = "dry.weight")
```

Arguments

`data`	data frame, at least with a column containing numeric water potential (bar), fresh.weight (g) and dry.weight (g) values, ordered by sample by descending water potential. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample IDs, default: "sample"
`water.potential`	optional name of the column in data containing the water potential values (bar), default = "water.potential"
`fresh.weight`	optional name of the column in data containing the fresh weight values (g), default: "fresh weight"
`dry.weight`	optional name of the column containing the dry weight values (g), default: "dry.weight"

Details

Above the turgor loss point, a linear relationship between water content and water potential exists. Based on this premise, saturated water content is found where water potential is zero. First, turgor loss point is calculated based on the relative leaf water loss (fresh weight minus dry weight relativized by the maximum leaf water content value). Then, data above the turgor loss point is extracted and fresh weight is fitted linearly to water potential. The point where water potential of the linear regression line is zero is the saturated water content.

Value

the original data frame (data) extended by a numeric column containing the saturated fresh weight values ("fresh.weight.saturated")

Examples

```
# get example data
df <- pressure_volume_data
# extend the data frame by saturated fresh weight
df <- FWSaturated(df)
```

Gmin *Minimum leaf conductance*

Description

Determines mean minimum leaf conductance and minimum leaf conducance at full turgidity and stomatal closure in experimentally obtained water loss curves.

Usage

```
Gmin(data, sample = "sample", time.since.start = "time.since.start",
  conductance = "conductance", RWD = "RWD.interval",
  stom.clos.threshold = FALSE, graph = TRUE, show.legend = TRUE)
```

Arguments

`data`	data frame containing columns of equal lengths giving the coordinates of the curve: time since start (minutes), conductance (mmol m^-2 s^-1) and RWD (%), ordered by sample by ascending time since start. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional column name in data containing the sample ID, default: sample
`time.since.start`	optional column name in data containing the numerical values for time since start of the experiment (min), default: time.since.start
`conductance`	optional column name in data containing the numerical conductance values (mmol m^-2 s^-1), default: "conductance"
`RWD`	optional column name in data containing the numerical relative water deficit values (%), default: "RWD.interval" (RWD average of an interval, as outputted by RWDInterval)
`stom.clos.threshold`	threshold value for stomatal closure. Automatic determination by default.
`graph`	set FALSE if no plots are to be returned
`show.legend`	set FALSE if no legend is to be shown in the plots

Details

The coordinates of stomatal closure are determined via the function StomatalClosure().
Conductance data including and following stomatal closure are then extracted and the average is taken (mean.gmin). A linear regression is applied to the data and the y axis intercept (gmin.full.sat) and the coordinate at the RWD point of stomatal closure (lin.gmin) are calculated from the function.

Before using this function, check the raw data for an initial plateau. If the exponential decline does not onset directly, fitting might not succeed.

Value

List splitted by sample consisting of

`gmin`	mean minimum conductance (mean.gmin) (mmol m^-2 s^-1) after stomatal closure of the measurement interval, minimum conductance at full saturation (gmin.full.sat) (mmol m^-2 s^-1) and minimum conductance at stomatal closure based on the linear fit (lin.gmin) (mmol m^-2 s^-1)
`formula`	formula of the linear regression of gmin vs. RWD
`coef`	coefficients of linear fit
`conf_int`	upper (97.5 %) and lower (2.5 %) border of 95 % confidence interval of model parameters

If graph = TRUE, the plotted original data is displayed with the x-axis intercept of the point of stomatal closure and the linear regression line of gmin showing the point of y-intercept (gminfullsat).

Examples

```
# get example data
df <- WeatherAllocation(leaf_drying_data, weather_data)  # allocate weather to weight loss data
df <- TimeSinceStart(df) # calculate time since start
df <- df[df$fw.plateau != "yes",] # remove plateauing data
df <- FittedFW(df, graph = FALSE) # correct noises in fresh weight
df <- RWDInterval(df, fresh.weight = "fitted.fw") # calculate RWD based in the intervals
df <- Conductance(df, fresh.weight = "fitted.fw") # calculate conductance

# calculate gmin and plot graphs
gmin <- Gmin(df)
```

`leaf_drying_data` *Experimentally determined leaf drying data*

Description

A dataset containing repeatedly measured fresh weights of transpiring leaves subjected to different soil moisture conditions during their growth (n = 6) and their saturated fresh weight and dry weight. togeher with their saturated fresh weight, their fresh weight before saturation, their dry weight and leaf area.

Usage

```
leaf_drying_data
```

Format

A data frame with 172 rows and 9 variables:

Details

- treatment: Soil moisture conditions during the last 6 days of Kohlrabi growth (10-30
- sample: Sample ID (1 - 12)
- dry.weight: Dry weight of the sample in gramm (0.3922 - 0.6692)
- fresh.weight.harvest: Fresh weight directly after cutting before over night saturation of the sample in gramm (2.8062 - 7.1009)
- fresh.weight.saturated: Saturated fresh weight of the sample in gramm (4.2186 - 7.3179)
- leaf.area: leaf area of the sample in cm^2 (91 - 148)
- date.and.time: Time (and date) of the fresh weight measurement of the transpiring sample (2019-03-26 09:48:00 - 2019-03-27 09:51:00)
- fresh.weight. Fresh weight of the transpiring sample in gramm (3.5112 - 7.3179)
- fw.plateau: Indicates plateauing fresh.weight values before the onset of the exponential decline. Remove values for analysis where fw.plateau = yes

---

`MergeDf` *Merges data to data frames*

Description

merges data frames containing all neccessary informations for plotting with PlotOutput()

Usage

```
MergeDf(x, y, y2 = FALSE, y3 = FALSE, legend, legend.y2 = FALSE,
  legend.y3 = FALSE)
```

Arguments

`x`	vector containing the x values
`y`	vector containing the y values
`y2`	optional vector containing the values for the second y coordinates
`y3`	optional vector containing the values for the third y coordinates
`legend`	name of the y values in the legend
`legend.y2`	optional name of the second y values in the legend
`legend.y3`	optional name of the third y values in the legend

Value

data frame with columns containing all above information in equalized length as requested by gglot

---

`ModElasticity` *Modulus of elasticity*

Description

Determines pressure potential and the modulus of elasticity

Usage

```
ModElasticity(data, sample = "sample",
  water.potential = "water.potential", RWD = "RWD", graph = TRUE,
  show.legend = TRUE)
```

Arguments

`data`	data frame containing columns of equal lengths giving the numerical coordinates of the curve: water potential (bar) and RWD (%), ordered by sample by descending water potential. A column containing the sample IDs is optionally required if several samples were measured
`sample`	optional column name in data containing the sample ID, default: "sample"
`water.potential`	optional column name in data containing the water potential values of the leaf (bar), default: "water.potential"
`RWD`	optional column name in data containing the relative water deficit values (%), default: "RWD"
`graph`	set FALSE if no plots are to be returned
`show.legend`	set FALSE if no legend is to be shown in the plots

Details

Relative water deficit at turgor loss point is determined via the function TurgorLossPoint() and osmotic potential is calculated via the function OsmoticPot().

Pressure potential is derived by subtracting osmotic potential from water potential. The part of the pressure potential prior the turgor loss point is then fitted linearly and the modulus of elasticity (M.Elasticity) equals the slope of the fitted line.

Before using this function, check the raw data for an initial plateau. If the exponential decline does not onset directly, fitting might not succeed.

Value

List splitted by sample consisting of

`modulus.elasticity`	modulus of elasticity (bar)
`formula`	formula of the transformed linear osmotic potential fit (1/-bar) and the pressure potential (bar) fit
`coef`	coefficients of the osmotic (1/-bar) and pressure potential (bar) fit
`conf_int`	upper (97.5 %) and lower (2.5 %) border of 95 % confidence interval of model parameters

If graph = TRUE, the original data is displayed with the x- and y-axis intercepts of the turgor loss point, the osmotic potential fit and the linear regression line of the pressure potential.

Examples

```
#get example data, calculate Relative Water Deficit
data <- RelativeWaterDeficit(pressure_volume_data)[pressure_volume_data$sample == 1, ]

# determine modulus of elasticity and the fitting parameters. Do not plot results.
m_elasticity <- ModElasticity(data, graph = FALSE)
```

`OrderCheck` *Order Check*

Description

Checks for the correct ordering of the data: increasing for date.and.time and time.since start, decreasing for fresh.weight and water.potential. Done separatly for each sample. An individualized warning is printed if not ordered correctly.

Usage

```
OrderCheck(data, sample = FALSE, date.and.time = FALSE,
  fresh.weight = FALSE, water.potential = FALSE,
  time.since.start = FALSE)
```

Arguments

`data`	data frame containing the data to be checked
`sample`	name of the column containing the sample IDs, if present in data
`date.and.time`	name of the column containing the date and time (POSIXct) of the measurements, if present in data
`fresh.weight`	name of the column containing the numeric fresh weight values, if present in data

`water.potential`	name of the column containing the numeric water potential values, if present in data
`time.since.start`	name of the column containing the numeric time since start values, if present in data

Value

no return value

`OsmoticPot` *Pressure Volume Curve Analysis*

Description

Determines the coordinates of the turgor loss point, osmotic potential at full hydration and apoplastic fraction

Usage

```
OsmoticPot(data, sample = "sample",
  water.potential = "water.potential", RWD = "RWD", graph = TRUE,
  show.legend = TRUE)
```

Arguments

`data`	data frame containing columns of equal lengths giving the numerical coordinates of the curve: water potential (bar) and RWD (%), ordered by sample by descending water potential. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional column name in data containing the sample ID, default: "sample"
`water.potential`	optional column name in data containing the numeric water potential values (bar), default: "water.potential"
`RWD`	optional column name in data containing the relative water deficit values (%), default: "RWD"
`graph`	set FALSE if no plots are to be returned
`show.legend`	set FALSE if no legend is to be shown in the plots

Details

RWD at turgor loss point is derived by the function TurgorLossPoint().

The pressure-volume curve data is converted to -1/bar. The osmotic potential is then derived by fitting a linear regression line with the Gauss-Newton algorithm of nls() to the water potential data following the turgor loss point. The y- and x-axis intercept of the regression line gives the osmotic potential at full hydration (op.full.sat) and the apoplastic fraction (apo.fract), respectively. Water potential at turgor loss point equals the value of the osmotic potential fit at the relative water deficit at turgor loss point.

Value

List splitted by sample consisting of

`turgor.loss.point`	x and y coordinates of the turgor loss point (RWD (%) and water.potential (bar), respectively)
`osmotic.potential`	x and y intercepts of the osmotic potential fit (apoplasic fraction (apo.fract) (%) and op.full.sat (bar), respectively)
`formula`	formula of the linear osmotic potential fit
`coef`	coefficients of the linear model
`conf_int`	upper (97.5 %) and lower (2.5 %) border of 95 % confidence interval of model parameters

If graph = TRUE, the plotted tranformed data is displayed with the x- and y-axis intercepts of the turgor loss point and the linear regression line of the osmotic potential showing the point of y-intercept (op.full.sat) and x-intercept (apo.fract).
Before using this function, check the raw data for an initial plateau. If the exponential decline does not onset directly, fitting might not succeed.

Examples

```
# get example data, calculate Relative Water Deficit
data <- RelativeWaterDeficit(pressure_volume_data)

# calculate pressure volume curve characteristics and plot graphs
pv_analysis <- OsmoticPot(data)
```

`PlotOutput` *Plot Output*

Description

plots the data as specified

Usage

```
PlotOutput(sub.sample, x, y, y2 = FALSE, y3 = FALSE, legend.y,
  legend.y2 = FALSE, legend.y3 = FALSE, x.axis, y.axis,
  x.intercept = FALSE, y.intercept = FALSE,
  legend.x.intercept = FALSE, line.x, line.y, line.y2 = FALSE,
  line.y3 = FALSE, legend.line.y, legend.line.y2 = FALSE,
  legend.line.y3 = FALSE, show.legend = show.legend)
```

Arguments

`sub.sample`	sample ID
`x`	vector containing the x coordinates of the data to be plotted as points
`y`	vector containing the y coodinates of the data to be plotted as points
`y2`	optional vector containing the second y coordinates of the data to be plotted as points
`y3`	optional vector containing the third y coordinates of the data to be plotted as points
`legend.y`	string, name of data points to be printed in the legend
`legend.y2`	string, optional name of second set of data points to be printed in the legend
`legend.y3`	string, optional name of third set of data points to be printed in the legend
`x.axis`	string, label of x axis
`y.axis`	sring, label of y axis
`x.intercept`	vector containg the x coordinate of the intercept
`y.intercept,`	optional vector containg the y coordinate of the intercept
`legend.x.intercept`	string, name of x.intercept to be printed in the legend
`line.x`	vector containing the x coordinate for the lines
`line.y`	vector containing the y coordinates for the line
`line.y2`	vector containing the y coordinates for the second line
`line.y3`	vector containing the y coordinates for the second line
`legend.line.y`	string, name of line to be printed in the legend
`legend.line.y2`	string, name of second line to be printed in the legend
`legend.line.y3`	string, name of third line to be printed in the legend
`show.legend`	boolean, specifies whether a legend is to be printed

Value

graphic

`pressure_volume_data` *Pressure volume curve data*

Description

A dataset containing water potential and fresh weight measurements of repeatedly measured drying kohlrabi leaves subjected to different soil moisture conditions during their growth (n = 6) and their saturated fresh weight and dry weight.

Usage

```
pressure_volume_data
```

Format

A data frame with 160 rows and 8 variables

Details

- date: Date of measurement
- treatment: Soil moisture conditions during the last 6 days of Kohlrabi growth (10-30
- sample: Sample ID (1 - 12)
- fresh.weight.harvest: Fresh weight measured at harvest (12 h prior measurement of fresh.weight.saturated) (2.9813 - 7.1557)
- fresh.weight.saturated: Saturated fresh weight of the leaf in gramms (4.1276 - 7.0867)
- fresh.weight: Fresh weight of the leaf in gramms (2.7215 - 6.8246)
- dry.weight: Dry weight of the leaf in gramms (0.2937 - 0.7267)
- water.potential: Water potential of the leaf in bar (-16.2 - -2.4)

`RelativeWaterContent` *Relative Water Content (RWC)*

Description

Calculates relative water content (RWC, %)

Usage

```
RelativeWaterContent(data, fresh.weight = "fresh.weight",
  dry.weight = "dry.weight",
  fresh.weight.saturated = "fresh.weight.saturated")
```

Arguments

`data`	data frame with columns of equal length containing at least columns with the fresh weight (g), the dry weight (g) and the saturated fresh weight (g)
`fresh.weight`	optional name of the column in data containing the numeric fresh weight values (g); default: "fresh.weight"
`dry.weight`	optional name of the column in data containing the numeric dry weight values (g); default: "dry.weight"
`fresh.weight.saturated`	optional name of the column in data containing the numeric saturated fresh weight values (g); default: "fresh.weight.saturated"

Details

Relative water content (%) is calculated as:

$$RWC = 100 * ((FW - DW)(FWs - DW)^-1)$$

whereas FW = fresh weight, DW = dry weight and FWs = fresh weight at water saturation.

Value

the original data frame extended by a numeric column with the relative water content (RWC) (%).

Examples

```
# get example data
df <- leaf_drying_data

# extend df by RWC
df_with_RWC <- RelativeWaterContent(df)
```

`RelativeWaterDeficit` *Relative Water Deficit (RWD)*

Description

Calculates relative water deficit (%)

Usage

```
RelativeWaterDeficit(data, fresh.weight = "fresh.weight",
  dry.weight = "dry.weight",
  fresh.weight.saturated = "fresh.weight.saturated")
```

Arguments

`data`	data frame with columns of equal length containing at least columns with the fresh weight (g), the dry weight (g) and the saturated fresh weight (g)
`fresh.weight`	optional name of the column in data containing the numeric fresh weight values (g); default: fresh.weight
`dry.weight`	optional name of the column in data containing the numeric dry weight values (g); default: dry.weight
`fresh.weight.saturated`	optional name of the column in data containing the numeric saturated fresh weight values (g); default: fresh.weight.saturated

Details

Relative water deficit (%) is calculated as:

$$RWD = 100 - 100 * ((FW - DW)(FWs - DW)^-1)$$

whereas FW = fresh weight, DW = dry weight and FWs = fresh weight at water saturation.

Value

the original data frame extended by a numeric column with the relative water deficit (RWD) (%).

Examples

```
# get example data
df <- leaf_drying_data

# extend df by RWD
df_with_RWD <- RelativeWaterDeficit(df)
```

`RWDInterval` *Mean relative water deficit (RWD) of an interval*

Description

Calculates relative water deficit (%) as mean value of a measurement interval

Usage

```
RWDInterval(data, sample = "sample", fresh.weight = "fitted.fw",
  dry.weight = "dry.weight",
  fresh.weight.saturated = "fresh.weight.saturated")
```

Arguments

`data`	data frame with columns of equal length containing at least columns with the fresh weight (g), the dry weight (g) and the saturated fresh weight (g), ordered by sample by descending by fresh weight. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample IDs, default: "sample"
`fresh.weight`	optional name of the column in data containing the numeric fresh weight values (g); default: "fitted.fw"
`dry.weight`	optional name of the column in data containing the numeric dry weight values (g); default: "dry.weight"
`fresh.weight.saturated`	optional name of the column in data containing the numeric saturated fresh weight values (g); default: "fresh.weight.saturated"

Details

First, the mean fresh weight is calculated for each measurement interval. Relative water deficit (%) is then calculated as:

$$RWD = 100 - 100 * ((mFW - DW)(FWs - DW)^-1)$$

whereas mFW = mean fresh weight, DW = dry weight and FWs = fresh weight at water saturation.

Value

the original data frame extended by a numeric column with the mean relative water deficit for the measurement interal (RWD.interval) (%).

Examples

```
# get example data
df <- leaf_drying_data

# extend df by RWD
df_with_RWD <- RWDInterval(df, fresh.weight = "fresh.weight")
```

`SaturationVaporPressure` *Saturation vapor pressure (VPsat)*

Description

Calculates saturation vapor pressure (kPa) using the Arden Buck equation

Usage

```
SaturationVaporPressure(data, temperature = "temperature")
```

Arguments

`data`	data frame with at least a numeric column containing temperature (degree Celsius)
`temperature`	optional name of the column in data containing the temperature values; default: "temperature"

Details

Calculates saturation vapor pressure (kPa) over liquid by temperature using the Arden Buck equation:

$$VPsat = 0.61121 exp((18.678 - T234.5^-1)(T257.14 + T))$$

where T = air temperature (degree Celsius)

Value

the original data frame extended by a numeric column with the saturation vapor pressure (kPa).

Examples

```
# generate example data frame
df <- data.frame(temperature = c(23.1, 23.2))

# extend df by saturation vapor pressure
df_with_VPsat <- SaturationVaporPressure(df)
```

`StomatalClosure` *Point of stomatal closure*

Description

Determines the point of stomatal closure in a set of experimentally obtained leaf drying curves. Stomatal closure happens when the curve irreversibly settles to linear water loss.

Usage

```
StomatalClosure(data, sample = "sample",
  time.since.start = "time.since.start", conductance = "conductance",
  RWD = "RWD.interval", threshold = FALSE, graph = TRUE,
  show.legend = TRUE)
```

Arguments

`data`	data frame containing columns of equal length giving the numerical coordinates of the curve: time since start (minutes), conductance (mmol m^-2 s^-1) and RWD (%), ordered by sample by ascending time since start. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample ID (if available), default: "sample"
`time.since.start`	optional name of the column in data containing numeric time since start values (min), default: "time.since.start"
`conductance`	optional name of the column in data containing numeric leaf conductance values (mmol m^-2 s^-1), default: "conductance"
`RWD`	optional name of the column in data containing numeric relative water deficit (%) values, default: "RWD.interval" (RWD average of an interval as outputted by RWDInterval)
`threshold`	sensitivity for the determination of stomatal closure. 60 by default.
`graph`	set FALSE if no plots are to be returned
`show.legend`	set FALSE if no legend is to be be shown in the plots

Details

Before using this function, check the raw data for an initial plateau. If the exponential decline does not onset directly, fitting might not succeed.
The conductances by time since start curves are fitted using the Gauss-Newton algorithm of nls() to a combined exponential and linear model. The exponential and linear parts are extracted and time since start at stomatal closure is localized at the point where the slope of the exponential part of the fit is higher than a threshold value. The threshold value is calculated by the use of the parameter b of the exponential part of the fit (a * exp(b * x)): -(b^2 * sens). The sensitivity constant (sens) is 60 by default and can be specified individually by the argument 'threshold'.
Minimum conductance (gmin) at stomatal closure is the conductance value of the overall fit at stomatal closure. RWD at stomatal closure is then calculated by linear regression of RWD and time since start.

Value

List splitted by sample consisting of

`stomatal.closure`	coordinates of the point of stomatal closure (time.since.start, RWD, conductance)
`formula`	formula of the exponential and linear part of the combined fits
`coef`	coefficients of combined model
`conf_int`	upper (97.5 %) and lower (2.5 %) border of 95 % confidence interval of model parameters

If graph = TRUE, the plotted original data is displayed with the exponential and linear fit of the combined model as well as the x-coordinate (time.since.start) of the point of stomatal closure.

Examples

```
# get example data
df <- WeatherAllocation(leaf_drying_data, weather_data)  # allocate weather to weight loss data
df <- TimeSinceStart(df) # calculate time since start
df <- df[df$fw.plateau != "yes",] # remove plateauing data
df <- FittedFW(df, graph = FALSE) # correct noises in fresh weight
df <- RWDInterval(df, fresh.weight = "fitted.fw") # calculate RWD based in the intervals
df <- Conductance(df, fresh.weight = "fitted.fw") # calculate conductance

# identify stomatal closure in curve and get graphs
sc <- StomatalClosure(df)
```

`TimeInterval` *Time Interval*

Description

Calculates time intervals (min) between temporally repeated measurements

Usage

```
TimeInterval(data, sample = "sample", date.and.time = "date.and.time")
```

Arguments

`data`	data frame containing at least a column giving the time (and date) (class POSIXct) of the measurements ordered by sample and chronologically. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample ID, default: "sample"
`date.and.time`	optional name of the column in data containing the time (and date) as class POSIXct, default: "date.and.time"

Value

the original data frame extended by a numerical vecor containing the time intervals (min) between the measurements of a sample. The first values of each sample is NA since time intervals are computed from row i and i-1.

Examples

```
# get example data
df <- leaf_drying_data

# extend df by time interval
df_with_ti <- TimeInterval(df)
```

`TimeSinceStart` *Time Since Start*

Description

Calculates time since start (min) of measurement for temporally repeated measurements

Usage

```
TimeSinceStart(data, sample = "sample",
  date.and.time = "date.and.time")
```

Arguments

`data`	data frame containing at least a column giving the the time (and date) of the measurements ordered by sample and chronologically. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample IDs, default: "sample".
`date.and.time`	optional name of the column in data containing the time (and date) as class POSIXct, default: "date.and.time"

Value

The original data frame extended by a numerical vector containing time since start (min) of the measurements.

Examples

```
# get example data frame
df <- leaf_drying_data

# extend df by time since start
df_with_tss <- TimeSinceStart(df)
```

`Transpiration` *Leaf transpiration*

Description

Calculates transpiration of a plant part from experimentally determined weight loss per time unit and double-sided leaf area.

Usage

```
Transpiration(data, sample = "sample", date.and.time = "date.and.time",
  fresh.weight = "fitted.fw", leaf.area = "leaf.area",
  output.unit = "mmol")
```

Arguments

`data`	data frame with columns of equal length containg at least columns with the time (and date) of the fresh weight measurements as well as columns with the measured fresh weights (g) and the single-sided leaf area (cm^2) of the sample. The data is to be ordered chronologically by sample. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample ID; default: "sample"
`date.and.time`	optional name of the column in data containing the time of the fresh weight measurements as class POSIXct; default: "date.and.time"
`fresh.weight`	optional name of the column in data containing the numeric fresh weight values (g); default: "fitted.fw"
`leaf.area`	optional name of the column in data containing the numeric single-sided leaf area values (cm^2); default: "leaf.area"
`output.unit`	optional; possible values: "mg" or "mmol"; defines whether transpiration is given in mmol m^-2 s^-1 (Default) or in mg m^-2 s^-1

Details

Transpiration (mmol s^-1 m^-2) is calculated as:

$$T = \Delta FM * 1000 * (\Delta t * 60 * LA * 2/10000 * 18.01528)^{-}1$$

whereas T = transpiration, ΔFW = reduction of fresh weight (g), Δt = time unit (min), LA = single-sided leaf area (cm^2)

Value

The original data frame extended by a numeric column with the transpiration (mg s^-1 m^-2 or mmol s^-1 m^-2) of the double-sided leaf area. The first value of each sample is NA, since transpiration values are computed from row i and i-1.

Examples

```
# get example data and allocate
df <- WeatherAllocation(leaf_drying_data, weather_data)

# extend df by transpiration in mmol s^-1 m^-2
df_with_transpiration <- Transpiration(df, fresh.weight = "fresh.weight")

# extend df by transpiration in mg s^-1 m^-2
df_with_transpiration <- Transpiration(df, fresh.weight = "fresh.weight", output.unit = "mg")
```

TurgorLossPoint *Turgor Loss Point*

Description

Determines the x coordinate (RWD) of the turgor loss point in a set of experimentally obtained pressure volume curves.

Usage

```
TurgorLossPoint(data, sample = "sample",
  water.potential = "water.potential", RWD = "RWD", graph = TRUE,
  show.legend = TRUE)
```

Arguments

`data`	data frame containing columns of equal lengths giving at least the numerical coordinates of the curve: water potential (bar) and RWD (%), ordered by sample by descending water potential. A column containing the sample IDs is optionally required if several samples were measured.
`sample`	optional name of the column in data containing the sample ID, default: "sample"
`water.potential`	optional name of the column in data containing the numeric water potential values (bar), default: "water.potential"
`RWD`	optional name of the column in data containing numeric relative water deficit values (%), default: "RWD"
`graph`	set FALSE if no plots are to be returned
`show.legend`	set FALSE if no legend is to be shown in the plots

Details

Before using this function, check the raw data for an initial plateau. If the exponential decline does not onset directly, fitting might not succeed.

The data is fitted using the Gauss-Newton algorithm of nls() to a combined exponential and linear model. The exponential and linear parts are extracted and RWD at turgor loss point is localized at their point of minimum distance.

Value

List splitted by sample consisting of

`turgor.loss.point`	coordinates of the turgor loss point (RWD)
`formula`	formula of the exponential and linear part of the combined fits
`coef`	coefficients of combined model

`conf_int`	upper (97.5 %) and lower (2.5 %) border of 95 % confidence interval of model parameters

If graph = TRUE, the plotted original data is displayed with the exponential and linear fit of the combined model as well as the x-coordinate (RWD) of the turgor loss point.

Examples

```
# get sample data
data <- RelativeWaterDeficit(pressure_volume_data)[pressure_volume_data$sample == 1, ]

# identify turgor loss point in curve
turgor_loss_point <- TurgorLossPoint(data)
```

`ValidityCheck` *Ensures the validity of the input data*

Description

Ensures the validity of the input data

Usage

```
ValidityCheck(data, sample = FALSE, leaf.area = FALSE,
  date.and.time = FALSE, dry.weight = FALSE,
  fresh.weight.saturated = FALSE, fresh.weight = FALSE,
  water.potential = FALSE, RWD = FALSE, conductance = FALSE,
  time.since.start = FALSE, temperature = FALSE, humidity = FALSE)
```

Arguments

`data`	data frame containing the data to be checked
`sample`	name of column containing the sample ID (default: sample)
`leaf.area`	name of column containing the leaf area (cm^2) (default: leaf.area)
`date.and.time`	name of column containing the date and time (default: date and time)
`dry.weight`	name of column containing the dry weight (g) (default: dry weight)
`fresh.weight.saturated`	name of column containing the saturated fresh weight (g) (default: fresh.weight.saturated)
`fresh.weight`	name of column containing the fresh weight (g) (default: fresh.weight)
`water.potential`	name of column containing the water potential (bar) (default: water.potential)
`RWD`	name of column containing the relative water deficit (default: RWD)
`conductance`	name of column containing the conductance values (default: conductance)
`time.since.start`	name of column containing the time since start values (default: time.since.start)
`temperature`	name of column containing the temperature values (default: temperature)
`humidity`	name of column containing the humidity values (default: humidity)

Value

no return value

`ValidityCheckDetail` *Checks if column exists in data, is numeric and has the same lenghts as the others existence*

Description

Checks if column exists in data, is numeric and has the same lenghts as the others existence

Usage

```
ValidityCheckDetail(data_in, value)
```

Arguments

`data_in`	data frame to be checked
`value`	column in data

Value

no return value

`VaporPressureDeficit` *Vapor Pressure Deficit (VPD)*

Description

Calculates mole fraction vapor pressure deficit (mol * mol^-1) of the air.

Usage

```
VaporPressureDeficit(data, humidity = "humidity",
  temperature = "temperature", atmospheric.pressure = 101.325)
```

Arguments

`data`	data frame at least with two numeric columns of equal length containg humidity (%) and temperature (degree Celsius)
`humidity`	optional name of the column in data containing the humidity values; default: "humidity"
`temperature`	optional name of the column in data containing the temperature values; default: "temperature"
`atmospheric.pressure`	optional; default = 101.325 (atmospheric pressure at sea level)

Details

Mole fraction vapor pressure deficit is calculated as:

$$(1 - RH/100) * (VPsat/AP)$$

whereas RH = relative humidity (%), VPsat = saturation vapor pressure (kPA), AP = atmospheric pressure (kPA), whereas:

$$VPsat = 0.61121 exp((18.678 - T234.5^-1)(T257.14 + T))$$

where T = air temperature (degree Celsius)

Value

The original data frame extended by a numeric column with the vapor pressure deficit (mol * mol^-1).

Examples

```
# get example data
df <- weather_data

# calculate vapor pressure deficit from weather data measured at sea level
df_with_VPD <- VaporPressureDeficit(df)

# calculate vapor pressure deficit from weather data measured at 2000 m altitude
df_with_VPD <- VaporPressureDeficit(df, atmospheric.pressure = 79.495)
```

`WeatherAllocation` *Weather Allocation*

Description

Calculates average weather (humidity, temperature) values within a measurement period.

Usage

```
WeatherAllocation(weight_loss_data, weather_data)
```

Arguments

`weight_loss_data`	data frame containing at least a column named "date.and.time" of the class POSIXct with the time (and date) of the measuring events. A column named "sample" containing the sample IDs is optionally required if several samples were measured.
`weather_data`	data frame containing at least a column named "date.and.time" of the class POSIXct with the time (and date) of the weather measurements and two columns named "humidity" and "temperature" containing the numerical weather data

Details

Averages within a measurement period are determined by approximate integration and normalization of the weather as a function of time.

Value

The original weight loss data frame extended by the approximatively integrated and normalized weather data for each period between two weight measurements. The first value of each sample is NA since weather values are averaged from row i to i-1.

Examples

```
# get example data
weight_loss_data <- leaf_drying_data
weather_data <- weather_data

# allocate averaged weather data to weight loss data
weight_loss_data_with_weather <- WeatherAllocation(weight_loss_data, weather_data)
```

`weather_data` *Weather data*

Description

A dataset containing weather data obtained by a data logger at the site of the measurement of the leaf drying data (leaf_drying_data)

Usage

```
weather_data
```

Format

A data frame with 253 rows and 3 variables

Details

- date.and.time: Time (and date) of the weather logging event (2019-03-25 00:00:00 - 2019-03-27 15:00:00)
- temperature: Air temperature in degree Celsius (18.05 - 26.81)
- humidity: Relative air humidity in percentage (35.1 - 65.07)

WeightDifference *Weight Difference*

Description

Calculates weight changes between temporally repeated measurements

Usage

```
WeightDifference(data, sample = "sample",
  fresh.weight = "fresh.weight")
```

Arguments

data	data frame containing at least a numeric column containing the measured weights (g), ordered chronologically by sample. A column containing the sample IDs is optionally required if several samples were measured.
sample	optional name of the column in data containing the sample ID, default: "sample"
fresh.weight	optional name of the column in data containing the numeric fresh weight (g) values, default: "fresh.weight"

Value

the original data frame extended by a numeric column containing the absolute differences in fresh weight (g) beween the measurements of a sample. The first value of each sample is NA since weight differences are computed from row i and i-1.

Examples

```
# get example data
df <- leaf_drying_data

# extend df by weight difference
df_with_WD <- WeightDifference(df)
```

Index