R topics documented:

avg_tree_curve .................................................. 3
bdq_meyer .......................................................... 5
bias_per ............................................................ 6
check_names ......................................................... 7
classify_site ......................................................... 8
class_center ........................................................ 9
diameter_class ...................................................... 10
dom_height ........................................................... 12
est_clutter ........................................................... 14
exfm1 ................................................................. 16
exfm10 ................................................................. 17
exfm11 ................................................................. 18
exfm12 ................................................................. 18
exfm13 ................................................................. 19
exfm14 ................................................................. 20
exfm15 ................................................................. 20
exfm16 ................................................................. 21
exfm17 ................................................................. 21
exfm18 ................................................................. 22
exfm19 ................................................................. 23
exfm2 ................................................................. 23
exfm20 ................................................................. 24
exfm21 ................................................................. 25
exfm22 ................................................................. 26
exfm3 ................................................................. 26
exfm4 ................................................................. 27
exfm5 ................................................................. 27
exfm6 ................................................................. 28
exfm7 ................................................................. 29
exfm8 ................................................................. 30
exfm9 ................................................................. 30
fit_clutter .............................................................. 31
forest_structure ..................................................... 33
graybill_f .............................................................. 34
guide_curve .......................................................... 36
huberwb .............................................................. 38
huberwob ............................................................ 39
ident_model .......................................................... 41
inv ................................................................. 43
lm_resid ............................................................. 44
lm_resid_group ....................................................... 45
lm_table .............................................................. 47
nls_table ............................................................. 49
npv_irr ............................................................... 52
outliersiqr ........................................................... 54
plot_summarise ....................................................... 54
avg_tree_curve

Generate the curve of a forest's average tree using the Kozak taper model

Description

Generate a ggplot curve of a forest’s average tree using the Kozak taper model (Kozak, Munro and Smith, 1969).

Usage

```r
avg_tree_curve(
  df,
  d,
  dbh,
  h,
  th,
  facet = NA,
  color = NA,
  eq = TRUE,
  mirror = TRUE
)
```
Arguments

- **df**: A data frame.
- **d**: Quoted name of the section diameter variable, in cm.
- **dbh**: Quoted name of the diameter at breast height variable, in cm.
- **h**: Quoted name of the section height variable, in meters.
- **th**: Quoted name of the total height variable, in meters.
- **facet**: Optional argument. If supplied with the Quoted name of a factor variable(s), this variable is used to divide the plot into facets. Default: NA.
- **color**: Quoted name of a variable. If supplied, this variable will be used to classify the data by color. Default: NA.
- **eq**: if TRUE, Kozak’s taper model is adjusted and the equation is shown on the plot. Default: TRUE
- **mirror**: if TRUE, the plot will be mirrored, to resemble the shape of a tree. Default: TRUE

Value

A ggplot object.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


Examples

```r
library(forestmangr)
data("exfm7")
head(exfm7)

# Get the data's average tree curve inserting the section diameter and height, total height and dbh variables:
avg_tree_curve(df=exfm7,d="di_wb",dbh="DBH",h="hi",th="TH",eq=FALSE)

# It's possible to get the average tree curve of each strata with the facet arg., and divide the data by color with the color argument:
avg_tree_curve(exfm7,"di_wb","DBH","hi","TH","STRATA","GENCODE",FALSE)
```
bdq_meyer

Classify a forest for selective cutting using the Meyer BDq method

Description
This function can be used to plan and execute selective cuttings of a native forest, without damaging the forest’s natural structure.

Usage
bdq_meyer(
  df,
  plot,
  dbh,
  plot_area,
  class_interval = 5,
  dbh_min = 5,
  licourt_index = 2,
  output = "table"
)

Arguments
- df: A data frame.
- plot: Quoted name of the plot variable. used to differentiate the plot’s trees, and calculate the number of sampled plots.
- dbh: Quoted name of the diameter at breast height variable, in cm.
- plot_area: Quoted name of the plot area variable, or a numeric vector with the plot area value. The plot area value must be in square meters.
- class_interval: Numeric value for the class interval used to classify the data. Default: 5.
- dbh_min: Numeric value for minimum diameter value to be considered in the classifications. dbh values smaller than this will be dismissed from the classification. Default: 5.
- licourt_index: Numeric value for the start licourt index used. Default: 2.
- output: Character value for the desired output. Can be either "table" for the classified data table, "model" to get a lm object with the linear model fitted, "coefs" to get a vector with the Meyer coefficients, or "full", to get a list with all restuls. Default: "table".

Value
- a data frame, a lm object, a vector or a list, according to the output argument.

Author(s)
Eric Bastos Gorgens <e.gorgens@gmail.com>
References

Examples
library(forrestmangr)
data("exfm20")
head(exfm20)

# To get the table with the regulated forest:
bdq_meyer(exfm20, "transect", "dbh", 1000)

# Use different class interval values to get different results:
bdq_meyer(exfm20, "transect", "dbh", 1000, class_interval = 10)

# It's possible to get different outputs:
bdq_meyer(exfm20, "transect", "dbh", 1000, output="model")
bdq_meyer(exfm20, "transect", "dbh", 1000, output="coefs")
bdq_meyer(exfm20, "transect", "dbh", 1000, output="full")

bias_per

Bias of an estimator in percentage

Description
Function for calculating the bias of an estimator.

Usage
bias_per(df, y, yhat, na.rm = TRUE)

Arguments
df a data frame.
y Quoted name of the variable representing the observed values in the data frame. If a data frame is not provided, y can also be a numeric vector.
yhat Quoted name of the variable representing the estimated values in the data frame. If a data frame is not provided, yhat can also be a numeric vector.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds. default: TRUE

Details
Function for calculating the bias of an estimator, given the observed values, and the estimated values.
Value

Numeric vector with the bias value, in percentage.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

See Also

other statistics to evaluate estimators: \texttt{rmse_per} for the Root mean square error of an estimator

Examples

library(forestmangr)
data("exfm11")
head(exfm11)

# Bias of an estimator, given the data frame and quoted variable names:
bias_per(exfm11, "TH", "TH_EST3")

# Bias of an estimator, given the vectors for observed and estimated values:
bias_per(y = exfm11$TH, yhat = exfm11$TH_EST3)

check_names

\textit{Check if character vector contains variable names}

Description

Function used to check if a string, or a character vector contains variable names of a given data frame.

Usage

check_names(df, var_names, boolean = TRUE)

Arguments

df a data frame.
var_names Character vector to be compared with the data frame names.
boolean Boolean object used to define if the output is going to be a boolean object \texttt{TRUE}, or a string \texttt{FALSE}. Default: \texttt{TRUE}.

Details

Function used to check if a string, or a character vector contains variable names of a given data frame. This functions is mainly used to error-proof other functions of this package.
Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

Examples

```r
tidyverse
classify_site <- function(df, site, nc = 3, plot, .groups = NA) {
  # Function to classify inventory data based on site index

  # Check if the site variable is present
  if (!colSums(!is.na(df[[site]]) & !is.na(df[[plot]])) > 0) {
    stop("Site variable does not exist or is not numeric.
  
    Example:

```
library(forestmangr)

check_names(iris, "Species")
check_names(iris, "Species", boolean = FALSE )

check_names(iris, c("Especies", "Setal.Width") )
check_names(iris, c("Especies", "Setal.Width"), boolean = FALSE)

classify_site <- Classify inventory data based on site index

Usage
classify_site(df, site, nc = 3, plot, .groups = NA)

Arguments
df A data frame.
site Quoted name for the site variable.
nc number of categories used to classify the data. If 3, a additional column will be
 created with levels Lower, Middle and Upper, referencing the 3 categories. If
 not, only numbers will be used to differentiate the categories. Default: 3.
plot Quoted name for the plot variable.
.groups Optional argument. Quoted name(s) of grouping variables used to fit multiple
 regressions, one for each level of the provided variable(s). Default NA.

Value
A data frame classified based on the site index.

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

See Also
other sampling functions: fit_clutter for fitting Clutter’s Growth and Yield, and est_clutter
for estimating Clutter’s Growth and Yield model variables.
Examples

```
library(forestmangr)
data("exfm17")
head(exfm17)

# Classify data into 3 classes:
ex_class <- classify_site(exfm17, "S", 3, "plot")
head(ex_class, 15)
```

---

**class_center**

*Classify a given variable and get center of class*

### Description

This function can be used to divide the data into classes, based on minimum value and class interval of a given variable, and create a column with the center of each class.

### Usage

```
class_center(df, y, ci = 3, ymin = 5)
```

### Arguments

- **df**
  A data frame.
- **y**
  Quoted name of a variable, or a vector to be classified.
- **ci**
  Numeric value for the class interval used to classify the data. Default: 3.
- **ymin**
  Numeric value for minimum value value to be considered in the classifications. dbh values smaller than this will be dismissed from the classification. Default: 5.

### Value

if `df` is supplied, a data frame containing the supplied data with a new column for the center of classes; if `df` is missing, a vector with the center of class.

### Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>
Examples

library(forestmangr)
library(dplyr)
data("exfm20")
head(exfm20)

# n
# Number of individuals per ha per diameter class
class_center(df = exfm20, y = "dbh", ci = 10, ymin = 10)

exfm20 %>%
  mutate(CC = class_center(y = dbh, ci = 10, ymin = 10))

---

**diameter_class**  
*Divide data into diameter classes, and get number of observations*

**Description**

This function can be used to divide data into diameter classes, get the number of observations, number of observations per ha, and check number of species individuals, volume and G in each diameter class. It’s also possible to spread the diameter classes as columns.

**Usage**

```r
diameter_class(
  df, dbh, plot = NA, plot_area, ci = 5, dbhmin = 5, species = NA, volume = NA, NI_label = "NI",
  cc_to_column = FALSE, G_to_cc = FALSE, cctc_ha = TRUE, keep_unused_classes = FALSE)
```

**Arguments**

- **df**  
  A data frame.
- **dbh**  
  Quoted name of the diameter at breast height variable, in cm.
- **plot**  
  Optional parameter. Quoted name of the plot variable. used to differentiate the plots trees, and calculate the number of sampled plots. Default NA.
plot_area  Optional parameter. Quoted name of the plot area variable, or a numeric vector with the plot area value. The plot area value must be in square meters. Default NA.

ci  Numeric value for the class interval used to classify the data. Default: 5.

dbhmin  Numeric value for minimum diameter value to be considered in the classifications.dbh values smaller than this will be dismissed from the classification. Default: 5.

species  Optional parameter. Quoted name of the scientific names variable, or any variable used to differentiate the different species found in data. If supplied, will be used to classify the species in the diameter data. Default NA.

volume  Optional parameter. Quoted name of the volume variable. If supplied, will be used to classify the volume variable in the different diameter classes. Also, if cc_to_column is TRUE, the center of class columns will be filled with volume values, instead of number of individuals. Default NA.

NI_label  Label used for Species not identified. This parameter works along with species. The level supplied here will not be considered in the classification. Default "NI".

cc_to_column  If TRUE, will spread the center class column as multiple columns, one for each class. The value that fills these columns, by default is the number of individuals found in each class, but this can be changed by using other arguments. Default FALSE.

G_to_cc  If TRUE, and cc_to_column is also TRUE, will fill the center of class columns with basal area values, instead of number of individuals. Default FALSE.

cctc_ha  If TRUE, will calculate values per hectare for number of individuals per class, basal area per class and volume per class (if supplied). These values will also be used to fill the center of class columns, if cc_to_column is TRUE. Default TRUE.

keep_unused_classes  Some diameter classes may end up empty, depending on the maximum value of diameter and the class interval used. If this is TRUE, those classes will not be removed from the final data frame. Default FALSE.

Value
A data frame containing the supplied data divided into diameter classes.

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

Examples
library(forestmangr)
data("exfm20")
head(exfm20)

# n
# Number of individuals per ha per diameter class
diameter_class(df = exfm20, dbh = "dbh", ci = 10, dbhmin = 10, volume = "vol")

# Number of individuals per ha per diameter class per species
diameter_class(exfm20,"dbh", "transect", 10000, 10, 10, "scientific.name")

# Number of individuals per ha per diameter class, with each diameter class as a column
diameter_class(exfm20,"dbh","transect",10000,10,10,"scientific.name",cc_to_column=TRUE)

# G
# Basal area per ha per diameter class, with each diameter class as a column
# diameter_class(exfm20,"dbh","transect",10000,10,10,"scientific.name", cc_to_column=TRUE,G_to_cc=FALSE)

# Volume
# Volume per ha per diameter class
# diameter_class(exfm20,"dbh","transect",10000,10,10,"scientific.name",volume = "vol")

# Volume per ha per diameter class, with each diameter class as a column
# diameter_class(exfm20,"dbh","transect",10000,10,10,"scientific.name","vol",cc_to_column=TRUE)

---

**dom_height**

*Calculate the dominant height of forest inventory data plots*

**Description**

This function is used to get a data frame with Dominant height values for each plot in a forest inventory data.

**Usage**

```
dom_height(
  df,  # A data frame.
  th,  # Quoted name of the total height variable.
  dbh,
  plot,
  obs,
  dom,  # .groups,
  merge_data = FALSE,
  dh_name = "DH"
)
```

**Arguments**

- `df`  
- `th`  
- `dbh`  
- `plot`  
- `obs`  
- `dom`  
- `.groups`  
- `merge_data`  
- `dh_name`
**dom_height**

- **dbh**: Quoted name of the diameter at breast height variable. Used to filter out trees with no diameter measurement.
- **plot**: Quoted name of the plot variable. Used to differentiate the data's plots. If this argument is missing, the defined groups in the data frame will be used. If there are no groups in the data, the function will fail.
- **obs**: Quoted name of the observations variable. This will be used to tell which trees are dominant, i.e., it’s the variable that tells the type of tree; if it is normal, dominant, suppressed, etc. If this argument is not supplied, the function will calculate the average value of 2 trees with bigger height values in each plot, and use that as the dominant value.
- **dom**: Character value for the dominant tree code used in the observations variable supplied in the `obs` argument. This is used alongside the `obs` argument to differentiate dominant trees from the others.
- **.groups**: Optional argument. Quoted name(s) of grouping variables that can be added to differentiate subdivisions of the data. Default: NA.
- **merge_data**: If TRUE, will merge the original data frame with the dominant height table. Default: FALSE.
- **dh_name**: Character value for the name of the dominant height variable created. Default: "DH"

**Value**

A data frame with the the dominant height values by plot.

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**Examples**

```r
library(forestmangr)
data("exfm9")
head(exfm9)

# Let's say we need to get the dominant height (DH) values for a forest inventory data.
# If we don't have a variable that tells which trees are dominant, it's ok. We can
# still estimate DH using this function. It will average the top 2 trees of each plot:
dom_height(df=exfm9, th="TH", dbh="DBH", plot="PLOT")

# Of course, if we do have a variable that differentiate the dominant trees, it's
# best we use it. For that we use the obs argument, and the dom argument.
# In this data, the OBS variable is used to tell the type of tree.
# Let's check the levels in our OBS variable, to see which one is associated
# with dominant trees.
levels(as.factor(exfm9$OBS))

# So, the "D" level must be the one that tells which trees are dominant. Let's use it:
dom_height(df=exfm9, th="TH", dbh="DBH", plot="PLOT", obs="OBS", dom="D")
```
```
# If there are subdivisions of the data, like different strata, they can be included in the
# .groups argument:
dom_height(df=exfm9,th="TH",dbh="DBH",plot="PLOT",obs="OBS",dom="D",.groups="STRATA")

# It's possible to automatically bind the dominant heights table to the original data,
# using the merge_data argument:
dom_height(df=exfm9,th="TH",dbh="DBH",plot="PLOT",obs="OBS",dom="D",.groups="STRATA", merge_data=TRUE)
```

---

**est_clutter**

*Estimate future and present basal area, volume, TCA, CMI and MMI values of the Clutter Growth and Yield Model*

---

**Description**

This function estimates the present value of basal area for each class using either the
class mean, or a linear quadratic model, and then uses it’s value to calculate the basal area from
Clutter’s growth and yield model.

**Usage**

```r
est_clutter(
  df, # A data frame.
  age, # A numeric vector with the desired age range to be used in the estimation, or a
  basal_area, # Quoted name for the basal area variable.
  site, # Quoted name for the average site variable.
  category, # Quoted name for the category variable.
  coeffs, # Linear quadratic coefficients.
  method = "average", # Method for estimating basal area.
  annual_increment = FALSE, # Annual increment is included in the calculation.
  gray_scale = TRUE, # Gray scale is used for plotting.
  output = "table" # Output format.
)
```

**Arguments**

- **df**: A data frame.
- **age**: A numeric vector with the desired age range to be used in the estimation, or a
  Quoted name for the age variable.
- **basal_area**: Quoted name for the basal area variable.
- **site**: Quoted name for the average site variable.
- **category**: Quoted name for the category variable.
**est_clutter**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coeffs</td>
<td>Numeric vector or a data frame with the fitted values of Clutter’s growth and yield model. It must be a named vector, with b0,b1,b2,b3,a0 and a1 as names. a1 is not obligatory.</td>
</tr>
<tr>
<td>method</td>
<td>Method used for estimating the present basal area of each class. It can either be the class’ average basal area “average”, or an estimated value from a linear quadratic model of site as a function of basal area “model”. Default: “average”.</td>
</tr>
<tr>
<td>annual_increment</td>
<td>If TRUE, changes the labels from Mean Monthly Increment (MMI) and Current Monthly Increment (CMI) to Mean Annual Increment (MAI) and Current Annual Increment (CAI). Default FALSE.</td>
</tr>
<tr>
<td>gray_scale</td>
<td>If TRUE, the plot will be rendered in a gray scale. Default: “TRUE”.</td>
</tr>
<tr>
<td>output</td>
<td>Type of output the function should return. This can either be “plot”, for the estimation plots, “table”, for a data frame with the estimated values, and “full” for a list with the plot and 2 more data frames. “table”.</td>
</tr>
</tbody>
</table>

**Value**

A data frame, a ggplot object or a list, according to output.

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**See Also**

other sampling functions: `fit_clutter` for fitting the clutter growth and Yield model, and `classify_site` for classifying data according to site.

**Examples**

```r
library(forestmangr)
data("exfm17")
head(exfm17)

clutter <- fit_clutter(exfm17, "age", "DH", "B", "V", "S", "plot")
clutter

# Classify data into 3 classes:
ex_class <- classify_site(exfm17, "S", 3, "plot")
head(ex_class ,15)

# Estimate basal area using the average basal area as the initial basal area, volume, Mean Monthly Increment (MMI) and Current Monthly Increment (CMI) values using Clutter’s model:
est_clutter(ex_class,20:125, "B","S","category_",clutter,"average")

# For a more detailed output, including a plot, use output="full":
est_clutter(ex_class,20:125, "B","S","category_",clutter, output="full")
```
# Estimate basal area using an estimated basal area as the initial basal area:
est_clutter(ex_class, 20:125, "B", "S", "category_", clutter, "model")

# age can be a variable:
est_clutter(ex_class, "age", "B", "S", "category_", clutter, "model")

---

**exfm1**  
*Stratified random inventory pilot data*

**Description**

In this data, each observation is a plot.

**Usage**

data(exfm1)

**Format**

A data frame with 22 observations and 4 variables:

- **STRATA** stratum number
- **STRATA_AREA** area of each strata, in hectares
- **PLOT_AREA** area of plots, in square meters
- **VWB** volume with bark, in cubic meters
- **VWB_m3ha** volume with bark, in cubic meters per hectare

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**Source**

Inventory data of an eucalyptus forest Brazil with dominant height variable

Description

In this data, each observation is a tree.

Usage

data(exfm10)

Format

A data frame with 900 observations and 14 variables:

MAP  map numbers
PROJECT project number
STRATA stratum number
GENCODE genetic code of plots
STRATAAREA area of each strata, in hectares
PLANTING_DATE date of planting
SPACING Spacing used in the plots, in meters
PLOT plot number
MEASUREMENT_DATE date of measurement
PLOT_AREA area of plots, in square meters
PIT pit number
DBH diameter at breast height, in meters
TH total height, in meters
OBS quality of trees, N = normal tree, D = dominant tree, F = a failure, or dead tree
DH dominant height, in meters

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>
Observed height values of trees, and estimated values using 3 different models

Description

In this data, each observation is a tree.

Usage

data(exfm11)

Format

A data frame with 199 observations and 6 variables:

- STRATA  stratum number
- PLOT   plot number
- TH   total height, in meters
- TH_EST1 total estimated height with model 1, in meters
- TH_EST2 total estimated height with model 2, in meters
- TH_EST3 total estimated height with model 3, in meters

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Inventory data of an eucalyptus forest in Brazil, with age and site variables

Description

In this data, each observation is a plot. The site was estimated using the Chapman & Richards model.

Usage

data(exfm12)
**Format**

A data frame with 139 observations and 8 variables:

- **strata**: stratum number
- **plot**: plot number
- **age**: average age of plots, in months
- **DH**: dominant height, in meters
- **N**: number of individuals
- **V**: volume of trees, in cubic meters
- **B**: basal area, in square meters
- **S**: site variable, in meters

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

---

**Description**

In this data, each observation is a tree.

**Usage**

```r
data(exfm13)
```

**Format**

A data frame with 36 observations and 7 variables:

- **species**: species common name
- **trat**: treatment number
- **esp**: species number
- **N**: quantity of nitrogen used, in grams
- **N2**: squared quantity of nitrogen used, in squared grams
- **block**: block number
- **dbh**: diameter at breast height, in meters

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>
Inventory data of an eucalyptus forest in Brazil, with age and dominant height variables

Description
In this data, each observation is a plot.

Usage
data(exfm14)

Format
A data frame with 859 observations and 4 variables:

- strata  stratum number
- plot    plot number
- age     average age of plots, in months
- dh      dominant height, in meters

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

Simplified Inventory data of an eucalyptus forest in Brazil

Description
In this data, each observation is a tree.

Usage
data(exfm15)

Format
A data frame with 900 observations and 7 variables:

- STRATA  stratum number
- STRATA_AREA area of each strata, in hectares
- PLOT    plot number
- PLOT_AREA area of plots, in square meters
- DBH     diameter at breast height, in meters
- TH      total height, in meters
- OBS     quality of trees, N = normal tree, D = dominant tree, F = a failure, or dead tree
**Description**

In this data, each observation is a plot.

**Usage**

```r
data(exfm16)
```

**Format**

A data frame with 139 observations and 8 variables:

- `strata` stratum number
- `plot` plot number
- `age` average age of plots, in months
- `DH` dominant height, in meters
- `N` number of individuals
- `V` volume of trees, in cubic meters
- `B` basal area, in square meters

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

---

**Description**

In this data, each observation is a plot. The site was estimated using the Schumacher model.

**Usage**

```r
data(exfm17)
```
**Format**

A data frame with 139 observations and 8 variables:

- **strata**  stratum number
- **plot**  plot number
- **age**  average age of plots, in months
- **DH**  dominant height, in meters
- **N**  number of individuals
- **V**  volume of trees, in cubic meters
- **B**  basal area, in square meters
- **S**  site variable, in meters

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

---

**Description**

In this data, each observation is a tree.

**Usage**

data(exfm18)

**Format**

A data frame with 877 observations and 6 variables:

- **Plot**  plot number
- **Species**  species scientific name
- **Tree**  tree number
- **Trunk**  trunk number
- **CBH**  circumference at breast height, in meters
- **DBH**  diameter at breast height, in meters

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>
exfm19

Volume of felled trees, measured in the Smalian method

Description
In this data, each observation is a section of a tree.

Usage
data(exfm19)

Format
A data frame with 16 observations and 3393 variables:

- STRATA  stratum number
- TREE    number of trees
- DBH     diameter at breast height, in meters
- TH      total height, in meters
- CSA     cross section area with bark, in square meters
- VWB     volume with bark, in cubic meters
- FFWB    form factor for each section
- FFWB\_mean average form factor value

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

exfm2

Stratified random inventory definite data

Description
In this data, each observation is a plot.

Usage
data(exfm2)
Format

A data frame with 57 observations and 4 variables:

- **STRATA**  stratum number
- **STRATA_AREA**  area of each strata, in hectares
- **PLOT_AREA**  area of plots, in square meters
- **VWB**  volume with bark, in cubic meters
- **VWB_m3ha**  volume with bark, in cubic meters per hectare

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Source


---

Inventory data of a natural forest in Brazil

Description

In this data, each observation is a tree.

Usage

data(exfm20)

Format

A data frame with 12295 observations and 18 variables:

- **cod**  area code
- **transect**  plot number
- **tree**  tree number
- **common.name**  species common name
- **scientific.name**  species scientific name
- **family**  species family name
- **dbh**  diameter at breast height, in meters
- **canopy.pos**  canopy position
- **light**  level of light received by the tree
- **dead**  tells if the tree is dead or not
**Description**

In this data, each observation is a tree.

**Usage**

```r
data(exfm21)
```

**Format**

A data frame with 900 observations and 13 variables:

- **STRATA**: stratum number
- **STRATA_AREA**: area of each strata, in hectares
- **PLOT**: plot number
- **PLOT_AREA**: area of plots, in square meters
- **DBH**: diameter at breast height, in meters
- **TH**: total height, in meters
- **OBS**: quality of trees, N = normal tree, D = dominant tree, F = a failure, or dead tree
- **DH**: dominant height, in meters
- **TH_EST**: estimated total height, in meters
- **CSA**: cross sectional area, in square meters
- **AGE**: average age of plots, in months
- **VWB**: volume with bark, in cubic meters
- **VWOB**: volume without bark, in cubic meters

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>
Revenue data if an eucalyptus forest

Description
In this data, each observation is the year’s revenue.

Usage
data(exfm22)

Format
A data frame with 8 observations and 3 variables:

<table>
<thead>
<tr>
<th>year</th>
<th>Revenue year</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>cost values for that year, in dollars</td>
</tr>
<tr>
<td>revenue</td>
<td>revenue values for that year, in dollars</td>
</tr>
</tbody>
</table>

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

Simple random inventory pilot data

Description
In this data, each observation is a plot.

Usage
data(exfm3)

Format
A data frame with 10 observations and 3 variables:

<table>
<thead>
<tr>
<th>TOTAL_AREA</th>
<th>total area, in hectares</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLOT_AREA</td>
<td>area of plots, in square meters</td>
</tr>
<tr>
<td>VWB</td>
<td>volume with bark, in cubic meters</td>
</tr>
<tr>
<td>VWB_m3ha</td>
<td>volume with bark, in cubic meters per hectare</td>
</tr>
</tbody>
</table>

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>
Source

exfm4

Simple random inventory definite data

Description
In this data, each observation is a plot.

Usage
data(exfm4)

Format
A data frame with 25 observations and 3 variables:

- **TOTAL_AREA**: total area, in hectares
- **PLOT_AREA**: area of plots, in square meters
- **VWB**: volume with bark, in cubic meters

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

Source

exfm5

Systematic inventory data

Description
In this data, each observation is a plot.

Usage
data(exfm5)
Format

A data frame with 18 observations and 3 variables:

- **TOTAL_AREA**: total area, in hectares
- **PLOT_AREA**: area of plots, in square meters
- **VWB**: volume with bark, in cubic meters
- **VWB_m3ha**: volume with bark, in cubic meters per hectare

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Source


---

**Description**

In this data, each observation is a plot.

**Usage**

`data(exfm6)`

**Format**

A data frame with 10 observations and 14 variables:

- **GENCODE**: genetic code of plots
- **MAP**: map numbers
- **STRATA**: stratum number
- **PLOT**: plot number
- **AGE**: average age of plots, in months
- **STRATA_AREA**: area of each strata, in hectares
- **PLOT_AREA**: area of plots, in square meters
- **DBH**: diameter at breast height, in meters
- **q**: quadratic diameter, in meters
- **TH**: total height, in meters
- **DH**: dominant height, in meters
- **G**: basal area, in square meters
- **VWB**: volume with bark, in cubic meters
- **VWB_m3ha**: volume with bark, in cubic meters per hectare
- **VWOB**: volume without bark, in cubic meters
Data of felled trees sections, measured in the Smalian method

Description

In this data, each observation is a section of a tree.

Usage

data(exfm7)

Format

A data frame with 11 observations and 3393 variables:

MAP  map numbers
PROJECT  Project’s name
SPACING  Spacing used in the plots, in meters
STRATA  stratum number
GENCODE  genetic code of plots
TREE  number of trees
DBH  diameter at breast height, in meters
TH  total height, in meters
hi  height of sections, in meters
di_wb  diameter of sections with bark, in centimeters
bark_t  bark of thickness, in millimeters
Description

In this data, each observation is a section of a tree.

Usage

data(exfm8)

Format

A data frame with 10 observations and 596 variables:

- **CLONE** Clone number
- **STRATA** stratum number
- **TREE** number of trees
- **LOG** log number
- **DBH** diameter at breast height, in meters
- **TH** total height, in meters
- **hi** height of sections, in meters
- **di_wb** diameter of sections with bark, in centimeters
- **bark_t** bark of thickness, in millimeters
- **sec_length** section length, in meters

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Description

In this data, each observation is a tree.

Usage

data(exfm9)
**fit_clutter**  

**Format**

A data frame with 900 observations and 14 variables:

- **MAP**  map numbers
- **PROJECT**  project number
- **STRATA**  stratum number
- **GENCODE**  genetic code of plots
- **STRATA_AREA**  area of each strata, in hectares
- **PLANTING_DATE**  date of planting
- **SPACING**  Spacing used in the plots, in meters
- **PLOT**  plot number
- **MEASUREMENT_DATE**  date of measurement
- **PLOT_AREA**  area of plots, in square meters
- **PIT**  pit number
- **DBH**  diameter at breast height, in meters
- **TH**  total height, in meters
- **OBS**  quality of trees, N = normal tree, D = dominant tree, F = a failure, or dead tree

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

---

**Description**

Fit the Clutter model for growth and yield using the two stage least squares method (2SLS).

**Usage**

```r
fit_clutter(
  df,
  age,
  dh,
  basal_area,
  volume,
  site,
  plot,
  .groups = NA,
  model = "full",
  keep_model = FALSE
)
```
Arguments

- **df**: A data frame.
- **age**: Quoted name for the age variable.
- **dh**: Quoted name for the dominant height variable.
- **basal_area**: Quoted name for the basal area variable.
- **volume**: Quoted name for the volume area variable.
- **site**: Quoted name for the site variable.
- **plot**: Quoted name for the plot variable.
- **.groups**: Optional argument. Quoted name(s) of grouping variables used to fit multiple regressions, one for each level of the provided variable(s). Default NA.
- **model**: Character variable for the type of the model fitted. If "full", the full model will be used. If "mod", a modified model will be fitted, where the X3 variable is excluded from the regression. Default: full.
- **keep_model**: If TRUE a variable with the regression model will be kept in the data frame. Default: FALSE.

Value

A data frame with the regression coefficients.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


See Also

other sampling functions: `est_clutter` for estimating the Clutter growth and yield model variables, and `classify_site` for classifying data according to site.

Examples

```r
library(forestmangr)
data("exfm17")
head(exfm17)

# To fit the Clutter model we just need to define the data, and age, dominant height, # basal area, volume, site and plot variables:
fit_clutter(exfm17, "age", "DH", "B", "V", "S", "plot")
```
To fit the alternate model (without a1) just use model="mod":

```r
fit_clutter(exfm17, "age", "DH", "B", "V", "S", "plot", model="mod")
```

To keep the regression model, use keep_model=TRUE:

```r
fit_clutter(exfm17, "age", "DH", "B", "V", "S", "plot", keep_model=TRUE)
```

---

### forest_structure

*Get the forest horizontal, vertical and internal structure*

#### Description

This function calculates the horizontal structure of a given forest inventory data, with information like absolute frequency, relative frequency, absolute density, relative density, absolute dominance, relative dominance, importance value index, and coverage value index. If additional variables are supplied, the vertical and internal structures are also provided.

#### Usage

```r
forest_structure(
  df,
  species,
  dbh,
  plot,
  plot_area,
  vertical_est = NA,
  internal_est = NA,
  NI_label = ""
)
```

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>A data frame.</td>
</tr>
<tr>
<td>species</td>
<td>Quoted name of the scientific names variable, or any variable used to differentiate the different species found in data.</td>
</tr>
<tr>
<td>dbh</td>
<td>Quoted name of the diameter at breast height variable, in cm.</td>
</tr>
<tr>
<td>plot</td>
<td>Quoted name of the plot variable. Used to differentiate the plot's trees, and calculate the number of sampled plots.</td>
</tr>
<tr>
<td>plot_area</td>
<td>Quoted name of the plot area variable, or a numeric vector with the plot area value. The plot area value must be in square meters.</td>
</tr>
<tr>
<td>vertical_est</td>
<td>Optional argument. Quoted name of the vertical strata variable, or the height variable. If this is a factor variable, its levels will be used to classify the forest vertically. If it's a height variable, the vertical strata will be created based on its mean and standard deviation values. Default: NA.</td>
</tr>
<tr>
<td>internal_est</td>
<td>Optional argument. Quoted name of the internal strata variable. Default: NA.</td>
</tr>
<tr>
<td>NI_label</td>
<td>Label used for Species not identified. This parameter works along with species. The level supplied here will not be considered in the classification. Default &quot;&quot;.</td>
</tr>
</tbody>
</table>
Value

a data frame with the forest’s structure.

Author(s)

Eric Bastos Gorgens <e.gorgens@gmail.com>

References


Examples

library(forestmangr)
data("exfm20")
head(exfm20)

# Get the forest's horizontal structure:
forest_structure(exfm20, "scientific.name", "dbh", "transect", 10000)

# area plot as a variable name:
forest_structure(exfm20, "scientific.name", "dbh", "transect", "plot.area")

# Get the forest's horizontal and vertical structure.
# The vertical structure variable can either be the height variable,
# or a factor variable with the horizontal strata:
forest_structure(exfm20, "scientific.name", "dbh", "transect", 10000, "canopy.pos")

# Get the forest's horizontal, vertical and internal structure:
forest_structure(exfm20, "scientific.name", "dbh", "transect", 10000, "canopy.pos", "light")

graybill_f

Graybill F Test

Description

Hypothesis test as described by Graybill (1976).

Usage

graybill_f(df, Y1, Yj, signif = 0.05, output = "simple")
graybill_f

Arguments

df A data frame.
Y1 Quoted name of the standard variable.
Yj Quoted name of the proposed variable.
signif Numeric value for the significance level used in the test. Default: 0.05.
output Defines the type of output. If "simple", a simple data frame is created, with only essential information about the test. If "table", more information is provided, and if "full", a data frame with informations about the test and both variables is created. Default: "simple".

Details

This test is used to compare two variables, usually a proposed method, and a standard variable. This test is popular among forestry engineers, specially because, since it considers all data in it's analysis, it's usually more precise than a standard mean t-test. If the data has outliers, the mean may not represent the data correctly, so Graybill F test is specially useful for heterogeneous data.

A simple model regression is applied, and it's significance is evaluated by applying Graybill F test for the parameters estimate, according to the methodology described by Graybill (1976).

Value

A data frame. Its dimensions will vary, according to the output argument.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


Examples

library(forestmangr)
data("exfm11")
head(exfm11)

# The data frame exfm11 contains a height variable called "TH". This will be our standard value. We'll compare it to height estimated using different hypsometric equations.
# These are variables "TH_EST1" and "TH_EST2":
graybill_f( exfm11,"TH", "TH_EST1")

# TH_EST1 is statistically different from "TH".
# It's possible to alter the test's significance level using the signif argument:
graybill_f( exfm11,"TH", "TH_EST1", signif = 0.01)

# Different output options are available through the output argument:
graybill_f( exfm11,"TH", "TH_EST2", output="table")
graybill_f( exfm11,"TH", "TH_EST2", output="full")

**guide_curve**

*Get the guide curve plot for growth and yield analysis of inventory data*

**Description**

Get the guide curve for growth and yield analysis of inventory data using the factor method, and different statistical models.

**Usage**

```r
guide_curve(
  df,  # A data frame.
  dh,  # Quoted name for the dominant height variable.
  age,  # Quoted name for the age variable.
  age_index,  # Numeric value for the age index.
  n_class = 4,  # Numeric value for the number of classes used to divide the data. Default 4.
  model = "Schumacher",  # model used to fit dh as a function of age. The models available are "Schumacher", "Curtis", "Chapman-Richards" and "Bailey-Clutter". Default: "Schumacher".
  start_chap = c(b0 = 23, b1 = 0.03, b2 = 1.3),  # Numeric vector with the start values for the Chapman-Richards model. This must be a named vector, with b0, b1 and b2 as parameter names. Default: c(b0=23, b1=0.03, b2 = 1.3).
  start_bailey = c(b0 = 3, b1 = -130, b2 = 1.5),
  round_classes = FALSE,  # Numeric value for the number of classes used to divide the data. Default 4.
  font = "serif",  # font = "serif",
  gray_scale = TRUE,
  output = "plot"
)
```

**Arguments**

- `df`: A data frame.
- `dh`: Quoted name for the dominant height variable.
- `age`: Quoted name for the age variable.
- `age_index`: Numeric value for the age index.
- `n_class`: Numeric value for the number of classes used to divide the data. Default 4.
- `model`: model used to fit dh as a function of age. The models available are "Schumacher", "Curtis", "Chapman-Richards" and "Bailey-Clutter". Default: "Schumacher".
- `start_chap`: Numeric vector with the start values for the Chapman-Richards model. This must be a named vector, with b0, b1 and b2 as parameter names. Default: c(b0=23, b1=0.03, b2 = 1.3).
- `start_bailey`: Numeric vector with the start values for the Bailey-Clutter model. This must be a named vector, with b0, b1 and b2 as parameter names. Default: c(b0=3, b1=-130, b2 = 1.5).
- `round_classes`: Numeric value for the number of classes used to divide the data. Default 4.
- `font`: font = "serif",
- `gray_scale`: TRUE,
- `output`: "plot"
**start_bailey**  Numeric vector with the start values for the Bailey-Clutter model. This must be a named vector, with b0, b1 and b2 as parameter names. Default: \(c( b0=3, b1=-130, b2 = 1.5)\).

**round_classes**  If TRUE, class values will be rounded to the nearest 5. Default TRUE.

**font**  Type of font used in the plot. Default: "serif".

**gray_scale**  If TRUE, the plot will be rendered in a gray scale. Default: "TRUE".

**output**  Type of output the function should return. This can either be "plot", for the guide curve plot, "table", for a data frame with the data used on the guide curve plot, and "full" for a list with 2 ggplot2 objects, one for residual plot and other for plot curves, a lm object for the regression, a data frame with quality of fit variables, the dominant height index, the class table used, and the table used for the guide curve plot. Default "plot".

**Value**

A data frame, a ggplot object, or a list, varying according to the "output" argument.

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**Examples**

```r
data("exfm14")
head(exfm14)

# To get a guide curve plot for this data, we simply need to input
# dominant height and age variables, age index, and number of classes to be used:
guide_curve(exfm14, "dh", "age", 72, 5)

# if we want to get the table used to get the plot, we can choose the output "table":
guide_curve(exfm14, "dh", "age", 72, 5, output = "table")

# Other models are available for use, such as Curtis, Chapman Richards, and Bailey:
# CR and BC models are non linear, and thus need start values. There are default values,
# but they may fail, depending on the data used, so it's recommended to try start values that
# are ideal for the data used:
guide_curve(exfm14, "dh", "age", 72, 5,
model = "Chapman-Richards", start_chap = c(b0=23, b1=0.03, b2 = 1.3))

# Or, to get more information on the analysis, such as details on the regression,
# bias, rmse, plot for residuals and more (cpu taxing):
## Not run:
guide_curve(exfm14, "dh", "age", 72, 5, output = "full")

## End(Not run)
```
huberwb  

*Calculate the volume with bark of trees using the Huber method*

**Description**

Function used to calculate the volume with bark of trees using the Huber method. This function has integration with dplyr, so it can be used inside a pipe, along with the `group_by` function.

**Usage**

```
huberwb(df, di, section_length, tree, .groups = NA, di_mm_to_cm = FALSE)
```

**Arguments**

- `df`  
  A data frame.

- `di`  
  Quoted name of the section diameter variable, in centimeters.

- `section_length`  
  Quoted name of the section length variable, in meters

- `tree`  
  Quoted name of the tree variable. used to differentiate the trees’ sections. If this argument is `NA`, the defined groups in the data frame will be used. Default: `NA`.

- `.groups`  
  Optional argument. Quoted name(s) of additional grouping variables that can be added to differentiate subdivisions of the data. If this argument is `NA`, the defined groups in the data frame will be used. Default: `NA`.

- `di_mm_to_cm`  
  Boolean argument that, if `TRUE`, converts the `di` argument from milliliters to centimeters. Default: `FALSE`.

**Value**

Data frame with volume values by section.

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**References**


**See Also**

Complementary functions: `huberwob`, for calculation of volume without bark using the Huber method, `smalianwb`, for calculation of volume with bark using the Smalian method, `smalianwob`, for calculation of volume without bark the Smalian method.
Examples

```r
library(forestmangr)
data("exfm8")
head(exfm8)

# Calculate the volume with bark using the Huber method:
huberwb(exfm8, "di_wb", "sec_length", "TREE")

# Using pipes:
library(dplyr)

exfm8 %>%
group_by(TREE) %>%
huberwb("di_wb", "sec_length")
```

---

**huberwob**

*Calculate the volume without bark of trees using the Huber method*

---

**Description**

Function used to calculate the volume without bark of trees using the Huber method. This function has integration without dplyr, so it can be used inside a pipe, along with the `group_by` function.

**Usage**

```r
huberwob(
  df,
  di,
  section_length,
  bt,
  tree,
  .groups = NA,
  di_mm_to_cm = FALSE,
  bt_mm_to_cm = FALSE
)
```

**Arguments**

- `df` A data frame.
- `di` Quoted name of the section diameter variable, in centimeters.
- `section_length` Quoted name of the section length variable, in meters
- `bt` Quoted name of the bark thickness variable, in centimeters.
- `tree` Quoted name of the tree variable. used to differentiate the trees’ sections. If this argument is NA, the defined groups in the data frame will be used. Default: NA.
huberwob

.groups
Optional argument. Quoted name(s) of additional grouping variables that can be added to differentiate subdivisions of the data. If this argument is NA, the defined groups in the data frame will be used. Default: NA.

di_mm_to_cm
Boolean argument that, if TRUE, converts the di argument from milliliters to centimeters. Default: FALSE.

bt_mm_to_cm
Boolean argument that, if TRUE, converts the bt argument from milliliters to centimeters. Default: FALSE.

Value
Data frame with volume values by section.

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

References

See Also
Complementary functions: huberwb, For calculation of volume with bark using the Huber method, smalianwb, for calculation of volume with bark using the Smalian method, smalianwob, for calculation of volume without bark the Smalian method.

Examples
library(forestmangr)
data("exfm8")
head(exfm8)

# Calculate the volume without bark using the Huber method:
huberwob(exfm8,"di_wb", "sec_length", "bark_t", "TREE")

# Using pipes:
library(dplyr)

exfm8 %>%
group_by(TREE) %>%
huberwob("di_wb", "sec_length", "bark_t")
**Description**

Function for using the Identity of a Model test, as described by Regazzi (1999).

**Usage**

```r
ident_model(
  df, 
  factor, 
  reduced_model, 
  filter = NA, 
  gray_scale = TRUE, 
  signif = 0.05, 
  font = "serif", 
  output = "table", 
  eq = TRUE
)
```

**Arguments**

- `df`: a data frame.
- `factor`: Quoted name of the factor variable used to differentiate the data projects in the test.
- `reduced_model`: Quoted or unquoted reduced model used in the test. The variables mentioned in the model must exist in the provided data frame. X and Y sides of the model must be separated by "~".
- `filter`: Optional argument. If supplied with levels present in factor, only these levels will be used in the test. NA.
- `gray_scale`: If TRUE a gray scale will be used in the plots. Default: FALSE.
- `signif`: Numeric value for the significance level used in the test. Default: 0.05.
- `font`: font family used in the plots. Can be either "serif" for Times New Roman or "sans" for arial unicode MS. Default: "serif".
- `output`: Defines the type of output. If "table" an anova table with the identity of model test is provided.
- `eq`: if TRUE, A model is adjusted and the equation is shown on the plot. Default TRUE if "plot" a ggplot plot/object representing the test is created, if "table_plot", both anova table and plot are provided, and if "full", a list is provided, with details on the dummies variables created, the reduced and complete models, the anova table and the plot. Default: "table"

**Value**

A data frame, a ggplot object, or a list, varying according to the output argument.
Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>
Marcio leles Romarco de Oliveira <marciormarco@gmail.com>

References


Examples

```r
library(forestmangr)
data("exfm13")
head(exfm13)

# The objective is to know if the diameter's behavior is similar among 3 species.
# For this we'll use a quadratic model. We'll use nitrogen (N) as our X variable.
ident_model(exfm13, "species", dbh ~ N + N2)

# This test shows that there are differences between the species.
# We can get more details on this using a different output, that will also
# give us a plot:
ident_model(exfm13, "species", dbh ~ N + N2, output = "table_plot",eq=FALSE)

# This gives us only the plot:
ident_model(exfm13, "species", dbh ~ N + N2, output = "table_plot",eq=FALSE)

# And this gives us additional information on the test:
ident_model(exfm13, "species", dbh ~ N + N2, output = "full",eq=FALSE)

# Looking at the plot, it seems that 2 species are behaving very similar, while
# the Pequi species is different from the other 2. We can confirm this by running
# the test in a paired fashion, using the filter argument:
ident_model(exfm13, "species", dbh ~ N + N2,
  filter = c("PEQUI", "SUCUPIRA-PRETA"), output = "table_plot",eq=FALSE)
ident_model(exfm13, "species", dbh ~ N + N2,
  filter = c("PEQUI", "VINHATICO"), output = "table_plot",eq=FALSE)
ident_model(exfm13, "species", dbh ~ N + N2,
  filter = c("SUCUPIRA-PRETA", "VINHATICO"), output = "table_plot",eq=FALSE)

# As we imagined, a single model can be used to describe the behavior of
# the "Sucupira-preta" and "Vinhatico" species,
# and a second model is needed to explain the Pequi Variable.

# It's possible to apply a color scale to the plots, and also change it's font to arial:
```

"arial"
inv

Calculate the inverse of a number

Description

This function returns the inverse of a numeric vector.

Usage

inv(x)

Arguments

x A numeric vector

Details

This function is mainly used when fitting statistical models. If one of the variables in a model is an inverse of a vector, the `lm` function does not properly compute the variable, if 1/vector is inserted directly in the model, leading to the need of creating a separate variable. This function allows the user to get the inverse of a given numeric vector inside the model, without the need to create a new variable.

Value

a numeric vector containing the inverse of x.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Examples

```r
library(forestmangr)
data("exfm15")
head(exfm15)

# Get the inverse of a vector
inv(iris$Petal.Length)

# Fit a model that contains the inverse of a variable, without the need to
# create a new variable for the inverse:
lm(log(TH) ~ inv(DBH), exfm15 )
# or
lm_table(exfm15, log(TH) ~ inv(DBH) )
```
lm_resid  

Fit linear regressions, with the option of removing outliers using a interactive plot of residuals.

Description

With this function it’s possible to fit linear regressions by a grouping variable, and evaluate each equation via an interactive plot of residuals, and get a data frame with each column as a coefficient and quality of fit variables, and other output options. Works with dplyr grouping functions.

Usage

lm_resid(
  df,
  model,
  output_mode = "table",
  est.name = "est",
  keep_model = FALSE,
  rmoutliers = FALSE,
  fct_to_filter = NA,
  rmlevels = NA,
  onlyfiteddata = FALSE,
  group_print = NA
)

Arguments

df  A data frame.
model  A linear regression model, with or without quotes. The variables mentioned in the model must exist in the provided data frame. X and Y sides of the model must be separated by "~".
output_mode  Selects different output options. Can be either "table", "merge", "merge_est" and "nest". See details for explanations for each option. Default: "table".
est.name  Name of the estimated y value. Used only if est.name = TRUE. Default: "est".
keep_model  If TRUE, a column containing lm object(s) is kept in the output. Useful if the user desires to get more information on the regression. Default: FALSE.
rmoutliers  If TRUE, outliers are filtered out using the IQR method. Default: FALSE.
fct_to_filter  Name of a factor or character column to be used as a filter to remove levels. Default: NA.
rmlevels  Levels of the fct_to_filter variable to be removed from the fit. Default: NA.
onlyfiteddata  If TRUE, the output data will be the same as the fitted (and possibly filtered) data. Default: FALSE.
group_print  This argument is only used internally by another function. Please ignore.
**Details**

this function uses `lm_table` as a basis, but calls a plot of residuals for each fitted model, for the user to evaluate. If one decides to remove any of the points, one can click and drag, and then click on the 'remove points’ button. After that, one must simply click 'done’ and the coefficients will be printed out.

It's possible to use the `output` argument to get a merged table if `output="merge"`, that binds the original data frame and the fitted coefficients. If `output="merge_est"` we get a merged table as well, but with y estimated using the coefficients. If the fit is made using groups, this is taken into account, i.e. the estimation is made by group.

If `output="nest"`, a data frame with nested columns is provided. This can be used if the user desires to get a customized output.

**Value**

A data frame. Different data frame options are available using the output argument.

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**Examples**

```r
if (interactive() ){
  library(forestmangr)
  library(dplyr)

  data("exfm19")

  # Fit SH model:
  lm_resid(exfm19, log(VWB) ~ log(DBH) + log(TH))
}
```

**lm_resid_group**

Fit linear regressions by group, with the option of removing outliers using a interactive plot of residuals.

**Description**

With this function it’s possible to fit linear regressions by a grouping variable, and evaluate each equation via a interactive plot of residuals, and get a data frame, with each column as a coefficient and quality of fit variables, and other output options. Works with `dplyr` grouping functions.
Usage

```r
lm_resid_group(
  df,
  model,
  .groups = NA,
  output_mode = "table",
  est.name = "est",
  keep_model = FALSE,
  rmoutliers = FALSE,
  fct_to_filter = NA,
  rmlevels = NA,
  onlyfiteddata = FALSE
)
```

Arguments

df A data frame.
model A linear regression model, with or without quotes. The variables mentioned in the model must exist in the provided data frame. X and Y sides of the model must be separated by "~".
.groups Quoted name(s) of grouping variables used to fit multiple regressions, one for each level of the provided variable(s). Default NA.
output_mode Selects different output options. Can be either "table", "merge", "merge_est" and "nest". See details for explanations for each option. Default: "table".
est.name Name of the estimated y value. Used only if est.name = TRUE. Default: "est".
keep_model If TRUE, a column containing lm object(s) is kept in the output. Useful if the user desires to get more information on the regression. Default: FALSE.
rmoutliers If TRUE, outliers are filtered out using the IQR method. Default: FALSE.
fct_to_filter Name of a factor or character column to be used as a filter to remove levels. Default: NA.
rmlevels Levels of the fct_to_filter variable to be removed from the fit Default: NA.
onlyfiteddata If TRUE, the output data will be the same as the fitted (and possibly filtered) data. Default: FALSE.

Details

This function uses lm_table as a basis, but calls a plot of residuals for each fitted model, for the user to evaluate. If one decides to remove any of the points, one can click and drag, and then click on the 'remove points’ button. After that, one must simply click 'done’ and the coefficients will be printed out.

It’s possible to use the output argument to get a merged table if output="merge", that binds the original data frame and the fitted coefficients. If output="merge_est" we get a merged table as well, but with y estimated using the coefficients. If the fit is made using groups, this is taken into account, i.e. the estimation is made by group.

If output="nest", a data frame with nested columns is provided. This can be used if the user desires to get a customized output.
**lm_table**

*Fit linear regressions by group, and get different output options.*

**Value**

A data frame. Different data frame options are available using the output argument.

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**Examples**

```r
if (interactive() ){  
  library(forestmangr)  
  library(dplyr)  
  data("exfm19")  
  # Fit SH model by group:  
  lm_resid_group(exfm19, log(VWB) - log(DBH) + log(TH), "STRATA")
}
```

**Description**

With this function it's possible to fit linear regressions by a grouping variable, and get a data frame with each column as a coefficient and quality of fit variables, and other output options. Works with dplyr grouping functions.

**Usage**

```r
lm_table(df,  
  model,  
  .groups = NA,  
  output = "table",  
  est.name = "est",  
  keep_model = FALSE,  
  rmoutliers = FALSE,  
  fct_to_filter = NA,  
  rmlevels = NA,  
  boolean_filter = NA,  
  onlyfitteddata = FALSE,  
  del_boolean = FALSE
)
```
Arguments

- **df**: A data frame.
- **model**: A linear regression model, with or without quotes. The variables mentioned in the model must exist in the provided data frame. X and Y sides of the model must be separated by "~".
- **.groups**: Optional argument. Quoted name(s) of grouping variables used to fit multiple regressions, one for each level of the provided variable(s). Default: NA.
- **output**: Selects different output options. Can be either "table", "merge", "merge_est" and "nest". See details for explanations for each option. Default: "table".
- **est.name**: Name of the estimated y value. Used only if est.name = TRUE. Default: "est".
- **keep_model**: If TRUE, a column containing lm object(s) is kept in the output. Useful if the user desires to get more information on the regression. Default: FALSE.
- **rmoutliers**: If TRUE, outliers are filtered out using the IQR method. Default: FALSE.
- **fct_to_filter**: Name of a factor or character column to be used as a filter to remove levels. Default: NA.
- **rmlevels**: Levels of the fct_to_filter variable to be removed from the fit. Default: NA.
- **boolean_filter**: Name of a Boolean column to be used as a filter to remove data. Default: NA.
- **onlyfitteddata**: If TRUE, the output data will be the same as the fitted (and possibly filtered) data. Default: FALSE.
- **del_boolean**: If TRUE, the Boolean column supplied will be deleted after use. Default: FALSE.

Details

With this function there’s no more need to use the do function when fitting a linear regression in a pipeline. It’s also possible to easily make fit multiple regressions, specifying a grouping variable. In addition to that, the default output sets each coefficient as a column, making it easy to call coefficients by name or position when estimating values.

It’s possible to use the output argument to get a merged table if output="merge", that binds the original data frame and the fitted coefficients. If output="merge_est" we get a merged table as well, but with y estimated using the coefficients. If the fit is made using groups, this is taken into account, i.e. the estimation is made by group.

If output="nest", a data frame with nested columns is provided. This can be used if the user desires to get a customized output.

Value

A data frame. Different data frame options are available using the output argument.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>
Examples

library(forestmangr)
library(dplyr)
data("exfm19")
head(exfm19)

# Fit Schumacher and Hall model for volume estimation, and get
# coefficient, R2 and error values:
lm_table(exfm19, log(VWB) ~ log(DBH) + log(TH))

# Fit SH model by group:
lm_table(exfm19, log(VWB) ~ log(DBH) + log(TH), "STRATA")

# This can also be done using dplyr::group_by:
exfm19 %>%
  group_by(STRATA) %>%
  lm_table(log(VWB) ~ log(DBH) + log(TH))

# It's possible to merge the original data with the table containg the coefficients
# using the output parameter:
fit <- lm_table(exfm19, log(VWB) ~ log(DBH) + log(TH), "STRATA", output = "merge")
head(fit)

# It's possible to merge the original data with the table,
# and get the estimated values for this model:
fit <- lm_table(exfm19, log(VWB) ~ log(DBH) + log(TH), "STRATA",
  output = "merge_est", est.name = "VWB_EST")
head(fit)

# It's possible to further customize the output,
# unnesting the nested variables provided when output is defined as "nest":
lm_table(exfm19, log(VWB) ~ log(DBH) + log(TH), "STRATA", output = "nest")

nls_table

**Fit non-linear regressions by group, using LM algorithm and get different output options.**

Description

With this function it's possible to fit non-linear regressions using Levenberg-Marquardt or Gauss-Newton algorithms by a grouping variable, and get a data frame with each column as a coefficient and quality of fit variables, and other output options. Works with dplyr grouping functions.

Usage

nls_table(
df, model, mod_start, .groups = NA, output = "table", est.name = "est", replace = FALSE, keep_model = FALSE, global_start, algorithm = "LM"
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>A data frame.</td>
</tr>
<tr>
<td>model</td>
<td>A linear regression model, with or without quotes. The variables mentioned in the model must exist in the provided data frame. X and Y sides of the model must be separated by &quot;~&quot;.</td>
</tr>
<tr>
<td>mod_start</td>
<td>A vector or data frame, with start values for coefficients used in the model. This can be a data frame containing the same group variables used in the .groups argument, and the start values.</td>
</tr>
<tr>
<td>.groups</td>
<td>Optional argument. Quoted name(s) of grouping variables used to fit multiple regressions, one for each level of the provided variable(s). Default NA.</td>
</tr>
<tr>
<td>output</td>
<td>Selects different output options. Can be either &quot;table&quot;, &quot;merge&quot;, &quot;merge_est&quot; and &quot;nest&quot;. See details for explanations for each option. Default: &quot;table&quot;.</td>
</tr>
<tr>
<td>est.name</td>
<td>Name of the estimated y value. Used only if est.name = TRUE. Default: &quot;est&quot;.</td>
</tr>
<tr>
<td>replace</td>
<td>If TRUE, models that don’t converge on a grouped regression fit will be replaced by coefficients fitted using all data. Default: FALSE.</td>
</tr>
<tr>
<td>keep_model</td>
<td>If TRUE, a column containing lm object(s) is kept in the output. Useful if the user desires to get more information on the regression. Default: FALSE.</td>
</tr>
<tr>
<td>global_start</td>
<td>Optional argument. A vector or data frame, with start values for the global fit regression used when &quot;replace&quot; is TRUE.</td>
</tr>
<tr>
<td>algorithm</td>
<td>Algorithm to be used in the non-linear regression. It can be &quot;LM&quot; (Levenberg-Marquardt, more robust) or &quot;GN&quot; (Gauss-Newton, less robust, uses nls default algorithm). Default: &quot;LM&quot;.</td>
</tr>
</tbody>
</table>

Details

This function Levenberg-Marquardt algorithm as default for fitting non-linear regression models. Also, with this function there no more need to use the do function when fitting a linear regression in a pipe line. It’s also possible to easily make fit multiple regressions, specifying a grouping variable. In addition to that, the default output sets each coefficient as a column, making it easy to call coefficients by name or position when estimating values. The Levenberg-Marquardt fit uses nlsLM.

Value

A data frame. Different data frame options are available using the output argument.
Examples

library(forestmangr)
library(dplyr)
data("exfm14")
head(exfm14)

# Fit Chapman & Richards non-linear model for dominant Height:
nls_table(exfm14, dh ~ b0 * (1 - exp( -b1 * age ) )^b2,
    mod_start = c( b0=23, b1=0.03, b2 = 1.3 ) )

# Fit CR model by strata:
nls_table(exfm14,dh ~ b0 * (1 - exp( -b1 * age ) )^b2,
    mod_start = c( b0=23, b1=0.03, b2 = 1.3 ),
    .groups = "strata") %>%
    as.data.frame

# or, using group_by

exfm14 %>%
group_by(strata) %>%
nls_table(dh ~ b0 * (1 - exp( -b1 * age ) )^b2,
    mod_start = c( b0=23, b1=0.03, b2 = 1.3 ) )

# If there are multiple start values, for each strata, they can be supplied like so:
tab_coef <- data.frame(strata = c(1:20, 24,25,27,28,30,31,33,35,36,37),
    rbind(
        data.frame(b0 = rep(23, 20), b1 = rep(0.03, 20), b2 = rep(1.3, 20) ),
        data.frame(b0 = rep(23, 10), b1 = rep(0.03, 10), b2 = rep(.5, 10 )))
tab_coef

nls_table(exfm14, dh ~ b0 * (1 - exp( -b1 * age ) )^b2,
    mod_start = tab_coef,
    .groups = "strata" )

# mod_start needs to be a data frame in this case.

# It's possible to bind the coefficients to the original data,
# to estimate y. We'll also estimate bias and rmse for this estimation.

# This can also be done directly using "merge_est" as output:
nls_table(exfm14,dh ~ b0 * (1 - exp( -b1 * age ) )^b2,
    mod_start = tab_coef,
    .groups = "strata",
    output = "merge_est",
    est.name = "dh_est" ) %>%
    mutate(
        bias = bias_per(y = dh, yhat = dh_est),
        rmse = rmse_per(y = dh, yhat = dh_est) ) %>%
head(15)

# It’s possible to further customize the output, using nested columns:
# It’s possible to use Gauss–Newton’s algorithm. In this case,
# some regressions will not converge.

exfm14 %>%
group_by(strata) %>%
nls_table(dh ~ b0 * (1 - exp(-b1 * age))^b2,
  mod_start = c( b0=23, b1=0.03, b2 = 1.3 ),algorithm="GN" )

# If some regressions don’t converge, it’s possible to fill those NAs with
# regression coefficients from a general fit, using the entire data:

npv_irr

### Calculate Net Present Value and other economic variables

#### Description

Get the net present value, internal rate of return, and other economic variables, given cost and revenue values.

#### Usage

```r
npv_irr(
  df, 
  year, 
  cost, 
  revenue, 
  rate, 
  output = "full", 
  sens_limits = c(1, 30), 
  big_mark = ",", 
  dec_mark = ".", 
  prefix = "$"
)
```
Arguments

- **df**: A data frame.
- **year**: Quoted name of the year variable.
- **cost**: Quoted name of the costs variable.
- **revenue**: Quoted name of the revenue variable.
- **rate**: Numeric value of the yearly rate to be used.
- **output**: Selects different output options. It can be either "full" for a list containing a sensibility plot and a data frame with a single observation and one column for each variable, or "simple" for a two column data frame with one observation for each calculated variable. Default: "simple".
- **sens_limits**: Selects the rate rage used in the sensibility plot. This is a numeric vector with two elements, the initial and final rate to be used as range. These can vary between 0 and 100. Default: c(1,30).
- **big_mark**: Selects thousands separator. Can be either ",", " " or ",". Default: ",".
- **dec_mark**: Selects decimal separator. Can be either "," or ".". Default: ".".
- **prefix**: selects the prefix for the y axis in the sensibility plot. Can be either "$" or "R$". Default: "$".

Value

A data frame, or a list, according to output.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Examples

```r
## Not run:
library(forestmangr)
data(exfm22)

npv_irr(exfm22,"year","cost","revenue",rate=8.75)

# To also get a sensibility plot, use

npv_irr(exfm22,"year","cost","revenue",rate=8.75, output="full")

## End(Not run)
```
outliersiqr  
*Calculate interquartile range*

**Description**  
Calculate interquartile range for a given vector

**Usage**  
```r  
outliersiqr(x)  
```

**Arguments**  
- **x** a vector.

**Author(s)**  
Sollano Rabelo Braga <sollanorb@gmail.com>

**Examples**  
```r  
library(forestmangr)  
outliersiqr(iris$Sepal.Length)  
```

---

**plot_summarise**  
*Summarize forest inventory data*

**Description**  
Get informations about forest inventory plots, like number of individuals, mean DBH, q, height, basal area, volume, etc.

**Usage**  
```r  
plot_summarise(  
  df,  
  plot,  
  plot_area,  
  dbh,  
  th,  
  .groups,  
  total_area,  
  vwb,  
  vwb,  
)  
```

---
```r
plot_summarise(
  dh,
  age,
  dec_places = 4
)
```

### Arguments

- **df**: A data frame.
- **plot**: Quoted name of the plot variable. Used to differentiate the data’s plots. If this argument is missing, the defined groups in the data frame will be used. If there are no groups in the data, the function will fail.
- **plot_area**: Quoted name of the plot area variable, or a numeric vector with the plot area value. The plot area value must be in square meters.
- **dbh**: Optional parameter. Quoted name of the diameter at breast height variable. If supplied, will be used to calculate the mean diameter per plot, quadratic diameter (q), basal area and basal area per hectare. Default `NA`.
- **th**: Optional parameter. Quoted name of the total height variable. If supplied, will be used to calculate the mean total height, and the dominant height variable, if the `dbh` is `NA`. Default `NA`.
- **.groups**: Optional argument. Quoted name(s) of grouping variables that can be added to differentiate subdivisions of the data. Default: `NA`.
- **total_area**: Optional argument. Quoted name of the total area variable, or a numeric vector with the total area value. The total area value must be in hectares. Default: `NA`.
- **vwb**: Optional parameter. Quoted name of the volume with bark variable. If supplied, will be used to calculate the total vwb per plot, and vwb per hectare per plot. Default `NA`.
- **vwob**: Optional parameter. Quoted name of the volume without bark variable. If supplied, will be used to calculate the total vwob per plot, and vwob per hectare per plot. Default `NA`.
- **dh**: Optional parameter. Quoted name of the dominant height variable. If supplied, will be used to calculate the mean dominant height per plot. If not, the `ht` variable supplied will be used to calculate the average of the top two trees of each plot, and use that as `dh`. Default: `NA`.
- **age**: Optional parameter. Quoted name of the age variable. If supplied, will be used to calculate the average age per plot. Default: `NA`.
- **dec_places**: Numeric value for the number of decimal places to be used in the output tables. Default: 4.

### Value

A data frame with informations per plot.

### Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>
Examples

library(forestmangr)
data("exfm21")
head(exfm21)

# Obligatory arguments. Basic informations about the plot.
plot_summarise(exfm21, "PLOT", 810)

# Area values can be numeric, or a variable name
plot_summarise(exfm21, "PLOT", "PLOT_AREA")

# With DBH supplied, we get the mean diameter, quadratic diameter,
# basal area and basal area per hectare:
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH")

# With TH supplied, we get the mean total height and dominant height
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH", "TH_EST")

# With strata supplied, we divide the data into 2 strata
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH", "TH_EST", "STRATA")

# The strata area can also be supplied
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH", "TH_EST", "STRATA", "STRATA_AREA")

# With VWB supplied, we get the total vw, and vw per hectare
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH", "TH_EST", "STRATA", "STRATA_AREA", "VWB")

# With VWOB supplied, we get the total vwob, and vwob per hectare
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH", "TH_EST", "STRATA", "STRATA_AREA", "VWOB")

# If the data already has a dominant height variable, it can also be supplied here
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH", "TH_EST", "STRATA", "STRATA_AREA", "VWOB", "DH")

# With the AGE variable supplied, we get the average age of each plot
plot_summarise(exfm21, "PLOT", "PLOT_AREA", "DBH", "TH_EST", "STRATA", "STRATA_AREA", "VWOB", "DH", "AGE")

---

**pow**

Raise a numeric vector to a given power

**Description**

This function returns a numeric vector raised to a given power.

**Usage**

pow(x, y)
Arguments

- **x**: A numeric vector.
- **y**: A numeric value for the power \( x \) should be raised to.

Details

This function is mainly used when fitting statistical models. If one of the variables in a model is a variable raised to a given power, the \( \text{lm} \) function does not properly compute the variable, if \( \text{vector}^\text{power} \) is inserted directly in the model, leading to the need of creating a separate variable. This function allows the user to get the power of a given numeric vector to \( y \) inside the model, without the need to create a new variable.

Value

A numeric vector containing \( x \) to the power of \( y \).

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Examples

```r
library(forestmangr)
data("exfm15")
head(exfm15)

# Raise a numeric vector to the power of 2:
pow(iris$Petal.Length, 2)

# Fit a model that contains the dbh squared, without the need to create a new variable:
\text{lm}(\ln(\text{TH}) \sim \text{DBH} + \text{pow} (\text{DBH}, 2), \text{exfm15})
\text{or}
\text{lm_table}(\text{exfm15}, \ln(\text{TH}) \sim \text{DBH} + \text{pow}(\text{DBH}, 2))
```

---

**resid_plot**

*Calculate residual values and create plots*

Description

Function for creating plots and tables for residual values from observed and estimated values.
Usage

resid_plot(
  df,
  obs,
  ...,  
  type = "scatterplot",
  point_size = 3,
  color = NA,
  nrow = NA,
  ncol = NA,
  lim_y = NA,
  xlab = "Observed values",
  clab = NA,
  font = "serif",
  legend_pos = "bottom",
  gray_scale = TRUE,
  res_table = FALSE
)

Arguments

  df   A data frame.
  obs  Quoted name of the observed values variable.
  ...  Quoted name(s) for the estimated values variable(s). Multiple variables must be
        separated by comma.
  type Character object for the type of plot created. The available plots are: "scatterplot",
          "histogram", "histogram_curve" and "versus". Default: "scatterplot".
  point_size Numeric value for the point size in scatter plots. Default: 3.
  color Quoted name of a variable. If supplied, this variable will be used to classify the
          data by color. Default: NA.
  nrow Numeric value for number of rows in the plot matrix. If not supplied, the plots
          will be automatically sorted. Default: NA.
  ncol Numeric value for number of columns in the plot matrix. If not supplied, the
          plots will be automatically sorted. Default: NA.
  lim_y Numeric value for the y axis upper and lower limit. If NA, the biggest residual
           value is used. Default: NA.
  xlab Character value for the x label used in some plots. Default: "Observed values".
  clab Character value for the color label used, if a color variable is supplied. If not
           supplied, the color variable name will be used. Default: NA.
  font Type of font used in the plot. Default: "serif".
  legend_pos Position of legend, when a color variable is supplied. This can either be "left",
                 "right", "top" or "bottom". Default: "bottom".
  gray_scale If TRUE, the plot will be rendered in a gray scale. Default: "TRUE".
  res_table If TRUE, the function will return a data frame with observed, estimated, and
             residual values. Default: FALSE.
rmse_per

Value

A ggplot object, or, if res_table = TRUE, a data frame.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Examples

library(forestmangr)
data("exfm11")
head(exfm11)

# Specifying the observed and estimated variables, we get a scatter plot
# for the percentage residuals:
resid_plot(exfm11, "TH", "TH_EST1")

# It's possible to get other types of plots, with the type argument:
resid_plot(exfm11, "TH", "TH_EST1", type = "histogram_curve")
resid_plot(exfm11, "TH", "TH_EST1", type = "versus")

# It’s possible to add a factor variable as color in the plots:
resid_plot(exfm11, "TH", "TH_EST1", "TH_EST2", color="STRATA",
xlab="Total Height (m)", clab="Strata", gray_scale=FALSE)

# If there are more estimated values variables, they can also be used
# in the comparison:
resid_plot(exfm11, "TH", "TH_EST1", "TH_EST2", "TH_EST3")

# It's possible to rearrange the plots with ncol and nrow:
resid_plot(exfm11, "TH", "TH_EST1", "TH_EST2", "TH_EST3", ncol=1)

# It's possible to get the residuals table used to generate these plots, with res_table=TRUE:
head( resid_plot(exfm11, "TH", "TH_EST1", "TH_EST2", res_table = TRUE) )

---

rmse_per

RMSE of an estimator in percentage

Description

Function for calculating the Root-Mean-Square-Error of an estimator.

Usage

rmse_per(df, y, yhat, na.rm = TRUE)
Arguments

- **df**: a data frame.
- **y**: Quoted name of the variable representing the observed values in the data frame. If a data frame is not provided, `y` can also be a numeric vector.
- **yhat**: Quoted name of the variable representing the estimated values in the data frame. If a data frame is not provided, `yhat` can also be a numeric vector.
- **na.rm**: a logical value indicating whether NA values should be stripped before the computation proceeds. Default: `TRUE`.

Details

Function for calculating the Root-Mean-Square-Error of an estimator, given the observed values, and the estimated values.

Value

Numeric vector with the RMSE value, in percentage.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

See Also

other statistics to evaluate estimators: `bias_per` for the bias of an estimator

Examples

```r
library(forestmangr)
data("exfm11")
head(exfm11)

# RMSE of an estimator, given the data frame and quoted variable names:
rmse_per(exfm11, "TH", "TH_EST3")

# RMSE of an estimator, given the vectors for observed and estimated values:
rmse_per(y = exfm11$TH, yhat = exfm11$TH_EST3)
```

-----

**rm_empty_col**                                Remove empty columns

Description

This function removes columns filled with NA or 0 from a data frame.
**round_df**

**Usage**

```r
rm_empty_col(x)
```

**Arguments**

- `x` A dataframe

**Details**

This function is mainly used inside other functions, to remove optional variables when they are not opted in.

**Value**

A dataframe.

**Author(s)**

Sollano Rabelo Braga <sollanorb@gmail.com>

**Examples**

```r
library(forestmangr)
library(dplyr)
data("exfm15")
head(exfm15)

exfm15 %>%
mutate(emptycol=NA) %>%
rm_empty_col
```

---

**round_df**

*Round all numeric variables of a data frame to a given digit*

**Description**

This function allows the user to round all numeric values of a data frame, directly, even if the data frame contains non-numeric variables (which would throw an error in the `round` function).

**Usage**

```r
round_df(df, digits, rf = "round")
```
similarity_matrix

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>A data frame.</td>
</tr>
<tr>
<td>digits</td>
<td>Numeric vector for the desired number of digits.</td>
</tr>
<tr>
<td>rf</td>
<td>Type of round to be used. It can either be &quot;ceiling&quot;, &quot;floor&quot;, &quot;trunc&quot;, &quot;signif&quot;, or &quot;round&quot;. Default &quot;round&quot;.</td>
</tr>
</tbody>
</table>

Value

A data frame, with all the numeric variables rounded up to the number given to digits.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

Examples

```r
library(forestmangr)

# Round all numeric variables
round_df(iris)

# Round all numeric variables using the floor function
round_df(iris, rf="floor")

# Do not run
# trying this with the the base function throws an error:

# round(iris)
```

similarity_matrix  

Get the similarity matrix of an area

Description

Calculates the Jaccard similarity index and Sorensen similarity index.

Usage

```r
similarity_matrix(
  df, 
  species, 
  comparison, 
  NI_label = "", 
  index = "Sorensen", 
  dendrogram = FALSE, 
  n_groups = 3 
)
```
Arguments

df: A data frame.
species: Quoted name of the scientific names variable, or any variable used to differentiate the different species found in data. If supplied, will be used to classify the species in the diameter data.
comparison: Quoted name of the variable containing levels to be compared with each other.
NI_label: Label used for Species not identified. This parameter works along with species. The level supplied here will not be considered in the classification. Default "".
index: Character value for the desired index to be used. Can be either "Jaccard", for a matrix based on the Jaccard index of similarity, "Sorensen", for a matrix based the Sorensen index of similarity, or "all", for a list with matrices for both indexes. Default: "Sorensen".
dendrogram: If TRUE, a dendrogram will also be created. Default: FALSE.
n_groups: Number of groups in the dendrogram. Default 3.

Value

A matrix object with a similarity matrix, or a list, according to the "index" and "dendrogram" arguments.

Author(s)

Eric Bastos Gorgens <e.gorgens@gmail.com>

References


Examples

library(forestmangr)
data("exfm20")
head(exfm20)

# To get the similarity matrix of an area, we simply need to provide
# the species variable name, and a subdivision variable name, like
# transect. By default we get a a matrix based on the Sorensen index:
similarity_matrix(exfm20, "scientific.name", "transect")

# To get the similarity matrix of Jaccard, use the index argument:
similarity_matrix(exfm20, "scientific.name", "transect", index = "Jaccard")

# To get a dendrogram with the matrix, use dendrogram=TRUE:
similarity_matrix(exfm20, "scientific.name", "transect", index = "Jaccard", dendrogram = TRUE)

# To get a list with both matrices, use index="all":
similarity_matrix(exfm20, "scientific.name", "transect", index = "all")
# If the data supplied only has 2 levels, a paired comparison is made instead:
ex_pair <- exfm20[exfm20$transect %in% c("T01", "T02") , ]
ex_pair

similarity_matrix(ex_pair, "scientific.name", "transect", index = "all")

---

### smalianwb

*Calculate the volume with bark of trees using the Smalian method*

**Description**

Function used to calculate the volume with bark of trees using the Smalian method. This function has integration with dplyr, so it can be used inside a pipe, along with the group_by function.

**Usage**

```r
smalianwb(
  df,
  di,
  hi,
  tree,
  .groups = NA,
  di_mm_to_cm = FALSE,
  hi_cm_to_m = FALSE
)
```

**Arguments**

- `df` A data frame.
- `di` Quoted name of the section diameter variable, in centimeters.
- `hi` Quoted name of the section height variable, in meters.
- `tree` Quoted name of the tree variable, used to differentiate the trees’ sections. If this argument is `NA`, the defined groups in the data frame will be used.
- `.groups` Optional argument. Quoted name(s) of additional grouping variables that can be added to differentiate subdivisions of the data. If this argument is not supplied, the defined groups in the data frame will be used. Default: `NA`.
- `di_mm_to_cm` Boolean argument that, if `TRUE`, converts the `di` argument from milliliters to centimeters. Default: `FALSE`.
- `hi_cm_to_m` Boolean argument that, if `TRUE`, converts the `hi` argument from centimeters to meters. Default: `FALSE`.

**Value**

Data frame with volume values by section.
smalianwob

Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

References

See Also
Complementary functions: smalianwob, For calculation of volume without bark using the Smalian method, huberwb, for calculation of volume with bark using the Huber method, huberwob, for calculation of volume without bark the Huber method.

Examples
library(forestmangr)
data("exfm7")
head(exfm7)

# Calculate the volume with bark using the Smalian method:
smalianwb(exfm7,"di_wb", "hi", "TREE")

# Using pipes:
library(dplyr)

exfm7 %>%
group_by(TREE) %>%
smalianwb("di_wb", "hi")

---

smalianwob Calculate the volume without bark of trees using the Smalian method

Description
Function used to calculate the volume without bark of trees using the Smalian method. This function has integration with dplyr, so it can be used inside a pipe, along with the group_by function.

Usage
smalianwob(
  df,
  di,
  hi,
  bt,
  tree,
  .groups = NA,
Arguments

df
A data frame.

di
Quoted name of the section diameter variable, in centimeters.

hi
Quoted name of the section height variable, in meters

bt
Quoted name of the bark thickness variable, in centimeters.

tree
Quoted name of the tree variable. used to differentiate the trees’ sections. If this argument is NA, the defined groups in the data frame will be used. Default: NA.

.groups
Optional argument. Quoted name(s) of additional grouping variables that can be added to differentiate subdivisions of the data. If this argument is NA, the defined groups in the data frame will be used. Default: NA.

di_mm_to_cm
Boolean argument that, if TRUE, converts the di argument from milliliters to centimeters. Default: FALSE.

hi_cm_to_m
Boolean argument that, if TRUE, converts the hi argument from centimeters to meters. Default: FALSE.

bt_mm_to_cm
Boolean argument that, if TRUE, converts the bt argument from milliliters to centimeters. Default: FALSE.

Value

Data frame with volume values by section.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


See Also

Complementary functions: smalianwb, For calculation of volume with bark using the Smalian method, huberwb, for calculation of volume with bark using the Huber method, huberwob, for calculation of volume without bark the Huber method.
Examples

```r
library(forestmangr)
data("exfm7")
head(exfm7)

# Calculate the volume without bark using Smalian's method:
smalianwob(exfm7,"di_wb", "hi", "bark_t", "TREE",bt_mm_to_cm=TRUE)

# Using pipes:
library(dplyr)

exfm7 %>%
group_by(TREE) %>%
  smalianwob("di_wb", "hi", "bark_t")
```

species_aggreg

```
Get the aggregation state of species
```

Description

Get the aggregation state of species according to the Payandeh, Hazen and Morista methods.

Usage

```r
species_aggreg(df, species, plot, NI_label = "")
```

Arguments

- **df**: A data frame.
- **species**: Quoted name of the scientific names variable, or any variable used to differentiate the different species found in data.
- **plot**: Quoted name of the plot variable. used to differentiate the plots trees, and calculate the number of sampled plots.
- **NI_label**: Label used for Species not identified. This parameter works along with species. The level supplied here will not be considered in the classification. Default "".

Value

A data frame with the aggregation classification.

Author(s)

Eric Bastos Gorgens <e.gorgens@gmail.com>

References

species_diversity

Examples

```r
library(forestmangr)
data("exfm20")
head(exfm20)

# Get the aggregation indexes of species:
species_aggreg(exfm20, "scientific.name", "transect")
```

---

**species_diversity**  
*Get the species diversity indexes*

**Description**

Calculate the diversity of species for the following indexes: Shannon, Simpson, Equitability, Pielou and Jentsch.

**Usage**

```r
species_diversity(df, species, plot = NA, NI_label = "", index = "all")
```

**Arguments**

- `df`  
  A data frame.

- `species`  
  Quoted name of the scientific names variable, or any variable used to differentiate the different species found in data. If supplied, will be used to classify the species in the diameter data.

- `plot`  
  Optional parameter. Quoted name of the plot variable. used to differentiate the plots, and calculate the indexes by plot, or other subdivision variable.

- `NI_label`  
  Label used for Species not identified. This parameter works along with species. The level supplied here will not be considered in the classification. Default "".

- `index`  
  Character value for the desired index to be used. Can be either "H" for Shannon's diversity index, "S" for Total number of species in the community, "Hmax" for the maximum equitability, "J" for Pielou evenness, "QM" for the mixture coefficient of Jentsch, or "all", to get all indexes. Default: "all".

**Value**

- a data frame with the indexes, or a numeric value of the desired index specified in the index argument.

**Author(s)**

Eric Bastos Gorgens <e.gorgens@gmail.com>
References

Examples
library(forestmangr)
data("exfm20")
head(exfm20)

# By default, the function returns all indexes:
species_diversity(exfm20, "scientific.name")

# It's possible to use a subdivision variable, like plot, to get
# the indexes for each subdivision:
species_diversity(exfm20, "scientific.name", "transect")

# To only get one specific index, use the index argument:
species_diversity(exfm20, "scientific.name", index = "H")
species_diversity(exfm20, "scientific.name", index = "S")
species_diversity(exfm20, "scientific.name", index = "Hmax")
species_diversity(exfm20, "scientific.name", index = "J")

---

sprs  Simple Random Sampling

Description
Function for processing forest inventory data using simple random sampling.

Usage
sprs(
  df,
  Yi,
  plot_area,
  total_area,
  m3ha = FALSE,
  age = NA,
  .groups = NA,
  alpha = 0.05,
  error = 10,
  dec_places = 4,
  pop = "inf",
  tidy = TRUE
)
Arguments

df a data frame.

Yi Quoted name of the volume variable, or other variable one desires to evaluate, in quotes.

plot_area Quoted name of the plot area variable, or a numeric vector with the plot area value. The plot area value must be in square meters.

total_area Quoted name of the total area variable, or a numeric vector with the total area value. The total area value must be in hectares.

m3ha Boolean value. If TRUE Yi variable is treated in m3/ha, else, in m3. Default: FALSE.

age Optional parameter. Quoted name of the age variable. Calculates the average age supplied. NA.

.groups Optional argument. Quoted name(s) of additional grouping variable(s) that, if supplied, will be used to run multiple surveys, one for each level. If this argument is NA, the defined groups in the data frame will be used, if they exist. Default: NA.

alpha Numeric value for the significance value used in the t-student estimation. Default: 0.05.

error Numeric value for the minimum admitted error value in the survey, in percentage. Default: 10.

dec_places Numeric value for the number of decimal places to be used in the output tables. Default: 4.

pop Character value for the type of population considered in the calculations. This can be either infinite ("inf") or finite ("fin"). Default: "inf".

tidy Boolean value that defines if the output tables should be tidied up or not. Default: TRUE.

Details

This function allows the user to processes inventory data using simple random sampling for finite or infinite populations. It’s possible to run multiple sampling analysis using a factor variable indicated in the .groups() parameter.

Value

A data frame with the sampling results.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


ss_diffs

See Also

other sampling functions: strs for stratified random sampling, and ss_diffs for Systematic Sampling.

Examples

library(forestmangr)
data("exfm2")
data("exfm3")
data("exfm4")

# The objective is to sample an area, with an error of 20%.
# First we run a pilot inventory, considering a 20% error and a finite population:
head(exfm3)
sprs(exfm3, "VWB", "PLOT_AREA", "TOTAL_AREA", error = 20, pop = "fin")

# With these results, in order to obtain the desired error, we'll need to sample new
# plots, and run the definitive inventory. Again, we aim for a 20% error, and consider
# the population as finite:
exfm4

sprs(exfm4, "VWB", "PLOT_AREA", "TOTAL_AREA", error = 20, pop = "fin")

# The desired error was met

# area values can be numeric
sprs(exfm4, "VWB", 3000, 46.8, error = 20, pop = "fin")

# Here we run a simple random sampling inventory for each forest subdivision,
# using the STRATA variable as a group variable:
exfm2

sprs(exfm2, "VWB", "PLOT_AREA", "STRATA_AREA",.groups = "STRATA",error = 20, pop = "fin")

# If the volume variable is in m3ha, you should set m3ha to TRUE:
sprs(exfm3, "VWB_m3ha", "PLOT_AREA", "TOTAL_AREA",m3ha = TRUE,error = 20, pop = "fin")

---

ss_diffs  Systematic Sampling

Description

Function for processing forest inventory data using systematic sampling.
Usage

```r
ss_diffs(
  df,
  Yi,
  plot_area,
  total_area,
  m3ha = FALSE,
  age = NA,
  .groups = NA,
  alpha = 0.05,
  error = 10,
  dec_places = 4,
  tidy = TRUE
)
```

Arguments

- `df`: a data frame.
- `Yi`: Quoted name of the volume variable, or other variable one desires to evaluate, in quotes.
- `plot_area`: Quoted name of the plot area variable, or a numeric vector with the plot area value. The plot area value must be in square meters.
- `total_area`: Quoted name of the total area variable, or a numeric vector with the total area value. The total area value must be in hectares.
- `m3ha`: Boolean value. If `TRUE` `Yi` variable is treated in m3/ha, else, in m3. Default: `FALSE`.
- `age`: Optional parameter. Quoted name of the age variable. Calculates the average age supplied. `NA`.
- `groups`: Optional argument. Quoted name(s) of additional grouping variable(s) that, if supplied, will be used to run multiple surveys, one for each level. If this argument is `NA`, the defined groups in the data frame will be used, if they exist. Default: `NA`.
- `alpha`: Numeric value for the significance value used in the t-student estimation. Default: `0.05`.
- `dec_places`: Numeric value for the number of decimal places to be used in the output tables. Default: `4`.
- `tidy`: Boolean value that defines if the output tables should be tidied up or not. Default: `TRUE`.

Details

This function allows the user to process inventory data using simple random sampling for finite or infinite populations. It’s possible to run multiple sampling analysis using a factor variable indicated in the `.groups()` parameter.
Value

A data frame with the sampling results.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


See Also

other sampling functions: sprs for Simple Random Sampling, and strs for stratified random sampling, and

Examples

library(forestmangr)
data("exfm2")data("exfm5")

# We’re trying to run a inventory for an area This data was collected systematically, # but we’ll try to run the data using simple random sampling, # to show the difference between the two methods:
head(exfm5)
sprs(exfm5, "VWB", "PLOT_AREA", "TOTAL_AREA")

# We get a 22% error value. Now, we run this same data # considering the data as a systematic inventory, using the # successive differences method:
exfm5

ss_diffs(exfm5, "VWB", "PLOT_AREA", "TOTAL_AREA")

# The error was significantly lowered.

# Area Values can be numeric;
ss_diffs(exfm5, "VWB", 200, 18)

# Here we run a systematic sampling inventory for each forest subdivision, # using the STRATA variable as a group variable:
exfm2

ss_diffs(exfm2, "VWB", "PLOT_AREA", "STRATA_AREA", .groups = "STRATA")
Description

Function for processing forest inventory data using stratified random sampling.

Usage

```r
strs(
df,
Yi,
plot_area,
strata_area,
strata,
m3ha = FALSE,
.groups = NA,
age = NA,
alpha = 0.05,
.error = 10,
dec_places = 4,
.pop = "inf",
tidy = TRUE
)
```

Arguments

df  a data frame.
Yi  Quoted name of the volume variable, or other variable one desires to evaluate, in quotes.
plot_area Quoted name of the plot area variable, or a numeric vector with the plot area value. The plot area value must be in square meters.
strata_area Quoted name of the strata area variable, or a numeric vector with the plot strata values. If there are more than 1 area values, it’s possible to use a vector with all area values, like `c(14.4, 16.4, 14.2)`. The strata area values must be in hectares.
strata Quoted name of the subdivision variable(s), also known as strata. If this argument is not supplied, the defined groups in the data frame will be used, if they exist.
m3ha Boolean value. If TRUE Yi variable is treated in m3/ha, else, in m3. Default: FALSE.
.groups Optional argument. Quoted name(s) of additional grouping variable(s) that, if supplied, will be used to run multiple surveys, one for each level. If this argument is NA, the defined groups in the data frame will be used, if they exist. Default: NA.
strs

age Optional parameter. Quoted name of the age variable. Calculates the average age supplied. NA.

alpha Numeric value for the significance value used in the t-student estimation. Default: 0.05.

error Numeric value for the minimum admitted error value in the survey, in percentage. Default: 10.

dec_places Numeric value for the number of decimal places to be used in the output tables. Default: 4.

pop Character value for the type of population considered in the calculations. This can be either infinite ("inf") or finite ("fin"). Default: "inf".

tidy Boolean value that defines if the output tables should be tidied up or not. Default: TRUE.

Details

This function allows the user to processes inventory data using stratified random sampling for forest subdivisions (strata), for finite or infinite populations. It’s possible to run multiple sampling analysis using a factor variable indicated in the .groups() parameter.

Value

A list containing two data frames, one with information for each strata, and one with the stratified sampling results.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


See Also

other sampling functions: sprs for Simple Random Sampling, and ss_diffs for Systematic Sampling.

Examples

library(forestmangr)
data("exfm1")
data("exfm2")
data("exfm6")

# The objective is to sample an area, with an error of 5%.
# First we run a pilot inventory, considering a 5% error and a finite population:
head(exfm1)

strs(exfm1, "VWB", "PLOT_AREA", "STRATA_AREA", strata = "STRATA", error = 5, pop = "fin")

# With these results, in order to meet the desired error of 5%, we'll need to sample 24 more plots,
# 4 in stratum 1, 8 in stratum 2, and 12 in stratum 3.
# After sampling the necessary plots, we now run a definitive inventory,
# considering an 5% error and a finite population:
exfm2

strs(exfm2, "VWB", "PLOT_AREA", "STRATA_AREA", strata = "STRATA", error = 5, pop = "fin")

# The desired error value was met.

# Area values can be numeric:
strs(exfm2, "VWB", 1000, c(14.4, 16.4,14.2), strata = "STRATA", error = 5, pop = "fin")

# Optional variable age, and one stratified sampled inventory for each map:
exfm6

strs(exfm6, "VWB", "PLOT_AREA", "STRATA_AREA", strata="STRATA", .groups = "MAP", age = "AGE")

tree_summarise

*Calculate the equivalent diameter of trees with more than one trunk*

**Description**

This function uses takes the square root of the diameters squared sum, in order to estimate the equivalent diameter of trees. Other supplied variables are summed up, or averaged, depending on the variable.

**Usage**

```
tree_summarise(df, dbh, tree, .groups = NA, vwb = NA, vwob = NA)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>A data frame.</td>
</tr>
<tr>
<td>dbh</td>
<td>Quoted name of the diameter at breast height variable.</td>
</tr>
<tr>
<td>tree</td>
<td>Quoted name of the tree variable. used to differentiate the trees’ sections. If this argument is missing, the defined groups in the data frame will be used. If there are no groups in the data, the function will fail.</td>
</tr>
<tr>
<td>.groups</td>
<td>Optional argument. Quoted name(s) of grouping variables that can be added to differentiate subdivisions of the data. Default: NA.</td>
</tr>
<tr>
<td>vwb</td>
<td>Optional argument. Quoted name of the volume with bark variable, in cubic meters. Default: NA.</td>
</tr>
</tbody>
</table>
vertical_stratum

vwob Optional argument. Quoted name of the volume without bark variable, in cubic meters. Default: NA.

Value

A data frame with the the equivalent diameter calculated.

Author(s)

Sollano Rabelo Braga <sollanorb@gmail.com>

References


Examples

library(forestmangr)
data("exfm18")
head(exfm18)

# Calculate the equivalent diameter of trees with more than one trunk:
eq_diam <- tree_summarise(exfm18, "DBH", tree="Tree", .groups=c("Plot", "Species") )
head(eq_diam, 10)

verticall_stratum

Divide data into 3 vertical strata

Description

Get the vertical strata of data based on the height variable. The data will be divided into inferior, medium and superior strata.

Usage

vertical_stratum(df, th)

Arguments

df A data frame.

th Quoted name of the total height variable.

Value

A data frame.
Author(s)

Eric Bastos Gorgens <e.gorgens@gmail.com>

References


Examples

```r
library(forestmangr)
data("exfm10")
head(exfm10)

# To classify the data, supply the data frame and the height variable name:
vertical_stratum(exfm10, "TH")
```

---

### vol_summarise

**Summarize volume of trees**

**Description**

This function can be used to summarize volume with and without bark of trees in a data frame.

**Usage**

```r
vol_summarise(df, dbh, th, vwb, tree, .groups = NA, vwob = NA)
```

**Arguments**

- `df`: A data frame.
- `dbh`: Quoted name of the diameter at breast height variable, in cm.
- `th`: Quoted name of the total height variable, in meters.
- `vwb`: Quoted name of the volume with bark variable, in cubic meters.
- `tree`: Quoted name of the tree variable, used to differentiate the trees’ sections. If this argument is NA, the defined groups in the data frame will be used. Default: NA.
- `.groups`: Optional argument. Quoted name(s) of additional grouping variables that can be added to differentiate subdivisions of the data.
- `vwob`: Optional argument. Quoted name of the volume without bark variable, in cubic meters. Default: NA. If this argument is NA, the defined groups in the data frame will be used. Default: NA.

**Value**

A data frame summarized by the `.groups` variable(s).
Author(s)
Sollano Rabelo Braga <sollanorb@gmail.com>

See Also
Complementary functions: `smalianwb`, for calculation of volume with bark using the Smalian method, `smalianwob`, for calculation of volume without bark using the Smalian method, `huberwb`, for calculation of volume with bark using the Huber method, `huberwob`, for calculation of volume without bark the Huber method.

Examples

```r
library(forestmangr)
data("exfm7")
head(exfm7)

# In order to calculate the volume of each tree, first we
# Calculate the volume by tree section using the Smalian method:
sec_data_vol <- exfm7 %>%
  smalianwb("di_wb", "hi", "TREE") %>%
  smalianwob("di_wb", "hi", "bark_t", "TREE", bt_mm_to_cm = TRUE)

sec_data_vol

# Now, we summarize the tree's volume:
vol_summarise(sec_data_vol, dbh = "DBH", th = "TH", vwb = "VWB",
              tree = "TREE", .groups = "STRATA", vwob = "VWOB")

# It's possible to do everything using pipes:
exfm7 %>%
  smalianwb("di_wb", "hi", "TREE") %>%
  smalianwob("di_wb", "hi", "bark_t", "TREE", bt_mm_to_cm = TRUE) %>%
  vol_summarise("DBH", "TH", "VWB", "TREE", "STRATA", "VWOB")
```

---

**Pipe**

Pipe an object forward into a function or call expression. See `magrittr` for more details.

Search for quasiquotation for more details.

---

**Tee**

Pipe an object forward into a function or call expression. See `magrittr` for more details.
Index

* **2SLS**
  * fit_clutter, 31

* **ClutterModel**
  * fit_clutter, 31

* **Random**
  * sprs, 69
  * strs, 74

* **Root-Mean-Square-Error**
  * rmse_per, 59

* **Sampling**
  * sprs, 69
  * ss_diffs, 71
  * strs, 74

* **Simple**
  * sprs, 69

* **Stratified**
  * strs, 74

* **Systematic**
  * ss_diffs, 71

* **bias**
  * bias_per, 6

* **data**
  * exfm1, 16
  * exfm10, 17
  * exfm11, 18
  * exfm12, 18
  * exfm13, 19
  * exfm14, 20
  * exfm15, 20
  * exfm16, 21
  * exfm17, 21
  * exfm18, 22
  * exfm19, 23
  * exfm2, 23
  * exfm20, 24
  * exfm21, 25
  * exfm22, 26
  * exfm3, 26
  * exfm4, 27

* exfm5, 27
* exfm6, 28
* exfm7, 29
* exfm8, 30
* exfm9, 30

:= (%>%), 79
>%%, 79
%T>%, 79

avg_tree_curve, 3
bdq_meyer, 5
bias_per, 6, 60

check_names, 7
class_center, 9
classify_site, 8, 15, 32
diameter_class, 10
dom_height, 12

est_clutter, 8, 14, 32
exfm1, 16
exfm10, 17
exfm11, 18
exfm12, 18
exfm13, 19
exfm14, 20
exfm15, 20
exfm16, 21
exfm17, 21
exfm18, 22
exfm19, 23
exfm2, 23
exfm20, 24
exfm21, 25
exfm22, 26
exfm3, 26
exfm4, 27
exfm5, 27
exfm6, 28
INDEX

exfm7, 29
exfm8, 30
exfm9, 30

fit_clutter, 8, 15, 31
forest_structure, 33

graybill_f, 34
guide_curve, 36

huberwb, 38, 40, 65, 66, 79
huberwob, 38, 39, 65, 66, 79
ident_model, 41
inv, 43

lm_resid, 44
lm_resid_group, 45
lm_table, 47

magrittr, 79

nls_table, 49
nlsLM, 50
npv_irr, 52

outliersiqr, 54

plot_summarise, 54
pow, 56

resid_plot, 57
rm_empty_col, 60
rmse_per, 7, 59
round, 61
round_df, 61

similarity_matrix, 62
smalianwb, 38, 40, 64, 66, 79
smalianwob, 38, 40, 65, 65, 79
species_aggreg, 67
species_diversity, 68
sprs, 69, 73, 75
ss_diffs, 71, 71, 75
strs, 71, 73, 74

tree_summarise, 76

vertical_stratum, 77
vol_summarise, 78