Package: forrescalc (via r-universe)

October 23, 2024

Title Calculation of Aggregated Values on Dendrometry, Regeneration and Vegetation of Forests, Starting from Individual Tree Measures from Fieldmap

Version 0.0.1

Description A collection of functions to load and aggregate measurements related to dendrometry, rejuvenation and vegetation, and to access plot level results from Flemish forest reserves in data package forresdat.

License GPL-3

URL https://inbo.github.io/forrescalc/, https://github.com/inbo/forrescalc

BugReports https://github.com/inbo/forrescalc/issues

Imports assertthat, DBI, dplyr, frictionless, fs, git2r, httr, jsonlite, lubridate, odbc, plyr, purrr, readr, readxl, rlang, RSQLite, stats, stringr, tidyr (>= 1.0.0), tidyselect (>= 1.0.0), tools

Suggests knitr, rmarkdown, testthat, tibble

VignetteBuilder knitr

Config/checklist/communities inbo

Config/checklist/keywords strict forest reserves; monitoring; forestry; Flanders; Belgium; dendrometry; rejuvenation; carbon; temperate deciduous forests; regeneration; vegetation

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://inbo.r-universe.dev

RemoteUrl https://github.com/inbo/forrescalc

RemoteRef HEAD

RemoteSha 4325507252548c7cc3f30800f6c4e09b4532a3b8

Contents

`add_zeros` *Add records with value zero for missing variable combinations in a given dataset*

Description

Datasets for which this package has been developed, typically contain measurements of observations. Absence is often not reported explicitly (e.g. there exists no record of a species that is not observed in a plot), while it can be important to include these zero values in an analysis (e.g. mean coverage per species in a certain forest reserve; mean stem number per diameter class in a forest reserve). This function automatically adds missing combinations with value zero to the dataset for each combination of values of the variables given in `comb_vars` (within each value of `grouping_vars`). All variables that are not mentioned in `comb_vars` or `grouping_vars`, are considered to be numerical variables and will get value 0 (zero). Note that if a certain value is not present in the dataset (or in one of the subsets defined by `grouping_vars`), it will not be added automatically; at least one record should be added manually for this value (e.g. a plot or `diameterclass` that doesn't exist in the given dataset, but has to be included in the output). The data in `forresdat` already contain one record with zeros per plot (with NA value for `species` and/or `diameterclass`), resulting in records to be added automatically if 'plot_id' is added to `comb_vars`.

Usage

```
add_zeros(
  dataset,
  comb_vars,
  grouping_vars,
  add_zero_no_na = NA,
  remove_na_records_in_comb_vars = NA,
  defaults_to_na = NA
)
```

Arguments

`dataset`	data.frame in which records should be added

comb_vars — variables (given as a vector of strings) of which all combinations of their values should have a record in the dataset.

grouping_vars — one or more variables for which the combination of values of the variables given in comb_vars should be made for each value, e.g. if grouping_vars = "forest_reserve" and comb_vars = c("plot", "species"), all combinations of the values in "plot" and "species" are made within each value of "forest_reserve".

add_zero_no_na — variable indicating which records of the grouping_vars should get a zero value (variable should be TRUE) or a NA value (variable should be FALSE). E.g. a variable indicating whether or not observations are done. If no variable name is given (default NA), all added records get zero values.

remove_na_records_in_comb_vars — In which of the given comb_vars should records with NA values be removed after adding the records with zero values for all combinations? In some cases, e.g. if no species are observed in a plot, the dataset in forresdat has records with species NA and zeros for measured variables to make sure zero values for all species are added for each plot when using this function. But after adding zero records for all missing species, the records with species NA have become superfluous. They can be removed by adding argument remove_na_records_in_comb_vars = "species". This argument defaults to NA (= no NA records are removed).

defaults_to_na — Columns in which the function should add NA instead of zero in the records that are added to complete the dataset.

Value

dataframe based on dataset to which records are added with value 0 (zero) for each measurement.

Examples

```
library(forrescalc)
library(dplyr)
dendro_by_plot_species <-
  read_forresdat_table(tablename = "dendro_by_plot_species") %>%
  select(
    -year, -plottype, -starts_with("survey_"), -data_processed,
    -starts_with("game_")
  )
add_zeros(
  dataset = dendro_by_plot_species,
  comb_vars = c("plot_id", "period", "species"),
  grouping_vars = c("forest_reserve")
)
add_zeros(
  dataset = dendro_by_plot_species,
  comb_vars = c("plot_id", "period", "species"),
  grouping_vars = c("forest_reserve"),
  remove_na_records_in_comb_vars = "species"
)
add_zeros(
  dataset = dendro_by_plot_species,
```

```
  comb_vars = c("plot_id", "period", "species"),
  grouping_vars = c("forest_reserve"),
  defaults_to_na = "stems_per_tree"
)
```

`calculate_dendrometry` *Calculates and aggregates parameters on dendrometry of trees*

Description

This function calculates additional variables and makes aggregations of individual tree measures on the levels of

- plot and year
- plot, tree species and year
- diameter class, plot and year
- diameter class, plot, tree species and year

and it makes aggregations of volume data on logs on the levels of

- decay stage, plot and year
- decay stage, plot, tree species and year

Usage

```
calculate_dendrometry(
  data_dendro,
  data_deadwood,
  data_shoots,
  height_model,
  plotinfo
)
```

Arguments

`data_dendro`	dataframe on tree measures with variables `plot_id`, `plottype`, `tree_measure_id`, `date_dendro`, `dbh_mm`, `height_m`, `species`, `alive_dead`, `decaystage`, `period`, `old_id`, `year`, `subcircle`, `plotarea_ha`,... (output of function `load_data_dendrometry()`)
`data_deadwood`	dataframe on logs with variables `plot_id`, `plottype`, `date_dendro`, `species`, `decaystage`, `calc_volume_m3`, `period` and `year` (output of function `load_data_deadwood()`)
`data_shoots`	dataframe on shoots as given from the function `load_data_shoots()`
`height_model`	dataframe with `model` containing 'exp' or 'ln', coefficients `P1` and `P2` to calculate height model for each combination of `species`, `forest_reserve`, `period` and `plottype`. Height models in .xlsx generated by `Fieldmap` can be grouped in a dataframe using function `load_height_models()`
`plotinfo`	dataframe on surveyed plots with variables `plot_id`, `plottype`, `forest_reserve`, `survey_trees`, `survey_deadw`, `period` and `year_dendro` (output of function `load_plotinfo()`)

Value

List of dataframes that are mentioned in the above description

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
calculate_dendrometry(
  data_dendro, data_deadwood, data_shoots, height_model, plotinfo)
```

---

calculate_regeneration

aggregates parameters on regeneration of trees

Description

This function makes aggregations of tree generation data on the levels of

- plot and year (and subplot for core area)
- plot, height class and year (and subplot for core area)
- plot, tree species and year (and subplot for core area)
- plot, height class, tree species and year (and subplot for core area)

For core area plots it makes additional aggregations on the levels of

- core area, tree species and year
- core area, height class, tree species and year

Usage

```
calculate_regeneration(data_regeneration)
```

Arguments

`data_regeneration`
: dataframe on tree regeneration with variables `plot_id`, `plottype`, `subplot_id`, `height_class`, `species`, `nr_of_regeneration`, `rubbing_damage_number`, `period`, `year`, `subcircle`, `plotarea_ha`, `min_number_of_regeneration` and `max_number_of_regeneration`.

Value

List of dataframes that are mentioned in the above description

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_regeneration <- load_data_regeneration(path_to_fieldmapdb)
calculate_regeneration(data_regeneration)
```

---

calculate_vegetation *aggregates parameters on vegetation of forests*

Description

This function makes aggregations of vegetation data on the levels of

- plot and year
- subplot and year (only for plot type 'core area')
- plot, species and year (only for plot type 'core area')

Usage

```
calculate_vegetation(data_vegetation, data_herblayer)
```

Arguments

data_vegetation	dataframe on vegetation with variables ...
data_herblayer	dataframe on vegetation in the species level ('herb layer') with variables ...

Value

List of dataframes that are mentioned in the above description

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_vegetation <- load_data_vegetation(path_to_fieldmapdb)
data_herblayer <- load_data_herblayer(path_to_fieldmapdb)
calculate_vegetation(data_vegetation, data_herblayer)
```

`calc_deadw_decay_plot` *aggregate parameters by decay stage, plot and year*

Description

This function calculates for each plot and year the volume logs and standing dead wood per hectare and per decay stage.

Usage

```
calc_deadw_decay_plot(plotinfo, data_deadwood = NA, data_dendro_calc = NA)
```

Arguments

`plotinfo`	dataframe on surveyed plots with variables `plot_id`, `plottype`, `forest_reserve`, `survey_trees`, `survey_deadw`, `period` and `year_dendro` (output of function `load_plotinfo()`)
`data_deadwood`	dataframe on logs with variables `plot_id`, `plottype`, `date_dendro`, `species`, `decaystage`, `calc_volume_m3`, `period` and `year` (output of function `load_data_deadwood()`)
`data_dendro_calc`	dataframe on stems (shoots and trees) as given from the function `calc_variables_stem_level()`

Value

dataframe with columns `plot`, `year`, `decaystage`, `vol_log_m3_ha`

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
data_stems_calc <- calc_variables_stem_level(data_stems, height_model)
data_dendro_calc <- calc_variables_tree_level(data_dendro, data_stems_calc)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
calc_deadw_decay_plot(plotinfo, data_deadwood, data_dendro_calc)
```

calc_deadw_decay_plot_species
aggregate parameters by decay stage, plot, tree species and year

Description

This function calculates for each plot, tree species and year the volume logs and standing dead wood per hectare and per decay stage.

Usage

```
calc_deadw_decay_plot_species(
  plotinfo,
  data_deadwood = NA,
  data_dendro_calc = NA
)
```

Arguments

plotinfo	dataframe on surveyed plots with variables plot_id, plottype, forest_reserve, survey_trees, survey_deadw, period and year_dendro (output of function load_plotinfo())
data_deadwood	dataframe on logs with variables plot_id, plottype, date_dendro, species, decaystage, calc_volume_m3, period and year (output of function load_data_deadwood())
data_dendro_calc	dataframe on stems (shoots and trees) as given from the function calc_variables_stem_level()

Value

dataframe with columns plot, year, tree_species, decaystage, vol_log_m3_ha

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
data_stems_calc <- calc_variables_stem_level(data_stems, height_model)
data_dendro_calc <- calc_variables_tree_level(data_dendro, data_stems_calc)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
calc_deadw_decay_plot_species(plotinfo, data_deadwood, data_dendro_calc)
```

`calc_dendro_plot` *aggregate parameters by plot and year*

Description

This function calculates for each plot and year some values per hectare: number of tree species, number of trees, basal area and volume.

Usage

```
calc_dendro_plot(data_dendro_calc, data_deadwood, plotinfo)
```

Arguments

`data_dendro_calc`	dataframe on tree measures with variables `plot_id`, `plottype`, `tree_measure_id`, `date_dendro`, `dbh_mm`, `height_m`, `species`, `alive_dead`, `decaystage`, `vol_tot_m3`, `basal_area_m2`, `period`, `old_id`, `year`, `subcircle`, `plotarea_ha`,... (output of function `calc_variables_tree_level()`)
`data_deadwood`	dataframe on logs with variables `plot_id`, `plottype`, `date_dendro`, `species`, `decaystage`, `calc_volume_m3`, `period` and `year` (output of function `load_data_deadwood()`)
`plotinfo`	dataframe on surveyed plots with variables `plot_id`, `plottype`, `forest_reserve`, `survey_trees`, `survey_deadw`, `period` and `year_dendro` (output of function `load_plotinfo()`)

Value

dataframe with columns plot, year, number_of_tree_species, number_of_trees_ha, basal_area_m2_ha, volume_m3_ha

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
data_stems_calc <- calc_variables_stem_level(data_stems, height_model)
data_dendro_calc <- calc_variables_tree_level(data_dendro, data_stems_calc)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
calc_dendro_plot(data_dendro_calc, data_deadwood, plotinfo)
```

calc_dendro_plot_species *aggregate parameters by plot and tree species*

Description

This function calculates for each plot, tree species and year some values per hectare: number of trees, basal area and volume.

Usage

```
calc_dendro_plot_species(data_dendro_calc, data_deadwood, plotinfo)
```

Arguments

data_dendro_calc	dataframe on tree measures with variables plot_id, plottype, tree_measure_id, date_dendro, dbh_mm, height_m, species, alive_dead, decaystage, vol_tot_m3, basal_area_m2, period, old_id, year, subcircle, plotarea_ha,... (output of function calc_variables_tree_level())
data_deadwood	dataframe on logs with variables plot_id, plottype, date_dendro, species, decaystage, calc_volume_m3, period and year (output of function load_data_deadwood())
plotinfo	dataframe on surveyed plots with variables plot_id, plottype, forest_reserve, survey_trees, survey_deadw, period and year_dendro (output of function load_plotinfo())

Value

dataframe with columns plot, year, tree_species, number_of_trees_ha, basal_area_m2_ha, volume_m3_ha

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
data_stems_calc <- calc_variables_stem_level(data_stems, height_model)
data_dendro_calc <- calc_variables_tree_level(data_dendro, data_stems_calc)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
calc_dendro_plot_species(data_dendro_calc, data_deadwood, plotinfo)
```

calc_diam_plot *aggregate parameters by diameter class, plot and year*

Description

This function calculates for each plot, year and diameter class some values per hectare: number of stems, basal area and volume of standing trees (for coppice based on data on shoot level), and volume of logs (= lying deadwood).

Usage

```
calc_diam_plot(data_stems_calc, data_deadwood, plotinfo)
```

Arguments

data_stems_calc	dataframe on stem level measurements with variables plot_id, plottype, tree_measure_id, date_dendro, dbh_mm, height_m, species, alive_dead, decaystage, period, year, subcircle, plotarea_ha,... (output of function calc_variables_stem_level())
data_deadwood	dataframe on logs with variables plot_id, plottype, date_dendro, species, decaystage, calc_volume_m3, period and year (output of function load_data_deadwood())
plotinfo	dataframe on surveyed plots with variables plot_id, plottype, forest_reserve, survey_trees, survey_deadw, period and year_dendro (output of function load_plotinfo())

Value

dataframe with columns plot, year, dbh_class_5cm, number_of_trees_ha, basal_area_m2_ha, volume_m3_ha

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
data_stems_calc <- calc_variables_stem_level(data_stems, height_model)
data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
calc_diam_plot(data_stems_calc, data_deadwood, plotinfo)
```

`calc_diam_plot_species` *aggregate parameters by diameter class, plot, tree species and year*

Description

This function calculates for each plot, tree species, year and diameter class some values per hectare: number of stems, basal area and volume of standing trees (for coppice based on data on shoot level), and volume of logs (= lying deadwood).

Usage

```
calc_diam_plot_species(data_stems_calc, data_deadwood, plotinfo)
```

Arguments

`data_stems_calc`	dataframe on stem level measurements with variables `plot_id`, `plottype`, `tree_measure_id`, `date_dendro`, `dbh_mm`, `height_m`, `species`, `alive_dead`, `decaystage`, `period`, `year`, `subcircle`, `plotarea_ha`,... (output of function `calc_variables_stem_level()`)
`data_deadwood`	dataframe on logs with variables `plot_id`, `plottype`, `date_dendro`, `species`, `decaystage`, `calc_volume_m3`, `period` and `year` (output of function `load_data_deadwood()`)
`plotinfo`	dataframe on surveyed plots with variables `plot_id`, `plottype`, `forest_reserve`, `survey_trees`, `survey_deadw`, `period` and `year_dendro` (output of function `load_plotinfo()`)

Value

dataframe with columns `plot`, `year`, `tree_species`, `dbh_class_5cm`, `basal_area_m2_ha`, `volume_m3_ha`

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
data_stems_calc <- calc_variables_stem_level(data_stems, height_model)
data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
calc_diam_plot_species(data_stems_calc, data_deadwood, plotinfo)
```

calc_diam_statistics_species

Give diameter statistics by forest reserve, species and year

Description

This function calculates the diameter distribution on the level of forest reserve, species and year

Usage

```
calc_diam_statistics_species(data_stems)
```

Arguments

data_stems	dataframe on stems (shoots and trees) as given from the function compose_stem_data()

Value

dataframe with columns forest_reserve, species, year and measures on diameter distribution (min, max, mean, median, Q1, Q3)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
calc_diam_statistics_species(data_stems)
```

calc_intact_deadwood *Calculate bole and crown volume of intact deadwood*

Description

In Core Areas, some lying deadwood is marked as 'complete tree' by giving variable `intact_fragm` value 10 (intact) instead of 20 (fragment) to save time (while in general all fragments are measured separately). This function calculates the total volume (sum of bole and crown volume) for this intact deadwood and keeps the initial volume in case of fragments.

Usage

```
calc_intact_deadwood(data_deadwood)
```

Arguments

data_deadwood	dataframe on logs with variables plot_id, plottype, date_dendro, species, decaystage, intact_fragm, calc_volume_m3, period and year (output of function load_data_deadwood()), in which calc_volume_m3 should be replaced by a more precise calculation

Value

A similar dataframe (data_deadwood) in which the volume of intact deadwood is replaced by a volume calculated based on tariffs. Intermediate results vol_crown_m3 and vol_bole_m3 are added as columns (which are NA in case of deadwood fragments).

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
calc_intact_deadwood(data_deadwood)
```

---

calc_reg_core_area_height_spec

aggregate regeneration parameters by plot, height, species and year

---

Description

This function calculates for each plot, height, species and year the number of regeneration per ha (or interval with mean and confidence interval using a log transformation), the number and percentage of subplots in which the species is regenerating and the approximate rubbing damage percentage per hectare. This calculation is designed for core areas, that consist of different subplots.

Usage

```
calc_reg_core_area_height_spec(data_regeneration)
```

Arguments

data_regeneration	dataframe on tree regeneration with variables plot_id, plottype, subplot_id, height_class, species, nr_of_regeneration, rubbing_damage_number, period, year, subcircle, plotarea_ha, min_number_of_regeneration and max_number_of_regeneration.

Value

dataframe with columns plot, species, year, height, nr_of_subplots_with_regeneration, perc_subplots_with_regeneration, approx_rubbing_damage_perc, mean_number_of_regeneration_ha, lci_number_of_regeneration_ha, uci_number_of_regeneration_ha and approx_nr_regeneration_ha.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_regeneration_CA <-
  load_data_regeneration(path_to_fieldmapdb, plottype = "CA")
calc_reg_core_area_height_spec(data_regeneration_CA)
```

---

calc_reg_core_area_species

aggregate regeneration parameters by plot, species and year

Description

This function calculates for each plot, species and year the number of seedlings and established regeneration per ha (or interval with mean and confidence interval using a log transformation), the number and percentage of subplots in which the species is regenerating and the approximate rubbing damage percentage per hectare. This calculation is designed for core areas,that consist of different subplots.

Usage

```
calc_reg_core_area_species(data_regeneration)
```

Arguments

`data_regeneration`
: dataframe on tree regeneration with variables `plot_id`, `plottype`, `subplot_id`, `height_class`, `species`, `nr_of_regeneration`, `rubbing_damage_number`, `period`, `year`, `subcircle`, `plotarea_ha`, `min_number_of_regeneration` and `max_number_of_regeneration`.

Value

dataframe with columns `plot`, `species`, `year`, `nr_of_subplots_with_regeneration`, `perc_subplots_with_regenerati` `mean_number_established_ha`, `lci_number_established_ha`, `uci_number_established_ha`, `mean_number_seedlings_ha`, `lci_number_seedlings_ha`, `uci_number_seedlings_ha`, `mean_rubbing_damage_perc_es` `lci_rubbing_damage_perc_established`, `uci_rubbing_damage_perc_established`, `mean_rubbing_damage_perc_se` `lci_rubbing_damage_perc_seedlings`, `uci_rubbing_damage_perc_seedlings`, `approx_nr_established_ha`, `approx_nr_seedlings_ha`, `approx_rubbing_damage_perc_established`, `approx_rubbing_damage_perc_seedlings`.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
```

```
data_regeneration_CA <-
  load_data_regeneration(path_to_fieldmapdb, plottype = "CA")
calc_reg_core_area_species(data_regeneration_CA)
```

---

calc_reg_plot *calculate species number by plot and year*

Description

This function calculates for each plot and year the number of species, total number of seedlings and established regeneration (or interval with mean and confidence interval using a log transformation) and approximate rubbing damage percentage for seedlings and established regeneration. For core area plots, these variables are calculated for each subplot.

Usage

```
calc_reg_plot(data_regeneration)
```

Arguments

data_regeneration	dataframe on tree regeneration with variables plot_id, plottype, subplot_id, height_class, species, nr_of_regeneration, rubbing_damage_number, period, year, subcircle, plotarea_ha, min_number_of_regeneration and max_number_of_regeneration.

Value

dataframe with columns plot, subplot, year, period, number_of_tree_species, nr_of_tree_species_established, mean_number_established_ha, lci_number_established_ha, uci_number_established_ha, mean_number_seedlings_ha, lci_number_seedlings_ha, uci_number_seedlings_ha, mean_rubbing_damage_perc_es lci_rubbing_damage_perc_established, uci_rubbing_damage_perc_established, mean_rubbing_damage_perc_see lci_rubbing_damage_perc_seedlings, uci_rubbing_damage_perc_seedlings, approx_nr_established_ha, approx_nr_seedlings_ha, approx_rubbing_damage_perc_established, approx_rubbing_damage_perc_seedlings.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_regeneration <- load_data_regeneration(path_to_fieldmapdb)
calc_reg_plot(data_regeneration)
```

calc_reg_plot_height *calculate species number by plot, tree height class and year*

Description

This function calculates for each plot, tree height class and year the number of species, total number of regeneration (or interval with mean and confidence interval using a log transformation) and approximate rubbing damage percentage for regeneration. For core area plots, these variables are calculated for each subplot.

Usage

```
calc_reg_plot_height(data_regeneration)
```

Arguments

data_regeneration	dataframe on tree regeneration with variables plot_id, plottype, subplot_id, height_class, species, nr_of_regeneration, rubbing_damage_number, period, year, subcircle, plotarea_ha, min_number_of_regeneration and max_number_of_regeneration.

Value

dataframe with columns plot, subplot, year, period, height_class, number_of_tree_species, approx_rubbing_damage_perc, mean_number_of_regeneration_ha, lci_number_of_regeneration_ha, uci_number_of_regeneration_ha and approx_nr_regeneration_ha.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_regeneration <- load_data_regeneration(path_to_fieldmapdb)
calc_reg_plot_height(data_regeneration)
```

calc_reg_plot_height_species *aggregate parameters by plot, tree height class, species and year*

Description

This function calculates for each plot, tree height class, species and year the number of regeneration (or interval with mean and confidence interval using a log transformation) and the approximate rubbing damage percentage per hectare for regeneration. For core area plots, these variables are calculated for each subplot.

Usage

```
calc_reg_plot_height_species(data_regeneration)
```

Arguments

data_regeneration
: dataframe on tree regeneration with variables plot_id, plottype, subplot_id, height_class, species, nr_of_regeneration, rubbing_damage_number, period, year, subcircle, plotarea_ha, min_number_of_regeneration and max_number_of_regeneration.

Value

dataframe with columns plot, subplot, year, height_class, species, approx_rubbing_damage_perc, mean_number_of_regeneration_ha, lci_number_of_regeneration_ha, uci_number_of_regeneration_ha and approx_nr_regeneration_ha.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_regeneration <- load_data_regeneration(path_to_fieldmapdb)
calc_reg_plot_height_species(data_regeneration)
```

---

calc_reg_plot_species *calculate regeneration parameters by plot, species and year*

Description

This function calculates for each plot, species and year the number of seedlings and established regeneration per ha (or interval with mean and confidence interval using a log transformation), and the approximate rubbing damage percentage for seedlings and established regeneration. For core area plots, these variables are calculated for each subplot.

Usage

```
calc_reg_plot_species(data_regeneration)
```

Arguments

data_regeneration
: dataframe on tree regeneration with variables plot_id, plottype, subplot_id, height_class, species, nr_of_regeneration, rubbing_damage_number, period, year, subcircle, plotarea_ha, min_number_of_regeneration and max_number_of_regeneration.

Value

dataframe with columns plot, subplot, species, year, period, mean_number_established_ha, lci_number_established_ha, uci_number_established_ha, mean_number_seedlings_ha, lci_number_seedlings_ha uci_number_seedlings_ha, mean_rubbing_damage_perc_established, lci_rubbing_damage_perc_established, uci_rubbing_damage_perc_established, mean_rubbing_damage_perc_seedlings, lci_rubbing_damage_perc_seedl uci_rubbing_damage_perc_seedlings, approx_nr_established_ha, approx_nr_seedlings_ha, approx_rubbing_damage_perc_established, approx_rubbing_damage_perc_seedlings.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_regeneration <- load_data_regeneration(path_to_fieldmapdb)
calc_reg_plot_species(data_regeneration)
```

---

calc_variables_stem_level

Calculate additional variables on stem level

Description

This function calculates additional variables based on measurements, such as

- calc_height_m: calculated height based on dbh_mm and a species specific diameter-height model
- basal_area_m2
- vol_bole_m3: calculated based on dbh_mm, calc_height_m and species specific tariffs
- vol_crown_m3: calculated based on dbh_mm and species specific tariffs
- vol_tot_m3: sum of vol_bole_m3 and vol_crown_m3
- basal_area_alive_m2_ha
- basal_area_dead_m2_ha
- vol_alive_m3_ha
- vol_dead_standing_m3_ha
- vol_bole_alive_m3_ha
- vol_bole_dead_m3_ha

Usage

```
calc_variables_stem_level(data_stems, height_model)
```

Arguments

data_stems	dataframe on stems (shoots and trees) as given from the function compose_stem_data()
height_model	dataframe with model containing 'exp' or 'ln', coefficients P1 and P2 to calculate height model for each combination of species, forest_reserve, period and plottype. Height models in .xlsx generated by Fieldmap can be grouped in a dataframe using function load_height_models()

Value

Dataframe with ...

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
calc_variables_stem_level(data_stems, height_model)
```

---

calc_variables_tree_level

Calculate additional variables on tree level

Description

This function calculates additional variables based on measurements, such as

- nr_of_stems: the number of shoots in the tree (= 1 for an individual tree; >= 1 when coppice)
- individual: true for individual tree or coppice, false if record is a secondary shoot
- calc_height_m: calculated height based on dbh_mm and a species specific diameter-height model
- basal_area_m2
- vol_bole_m3: calculated based on dbh_mm, calc_height_m and species specific tariffs
- vol_crown_m3: calculated based on dbh_mm and species specific tariffs
- vol_tot_m3: sum of vol_bole_m3 and vol_crowwn_m3
- dbh_mm (based on average for coppice trees)
- decaystage (based on average for coppice trees)
- basal_area_alive_m2_ha

- basal_area_dead_m2_ha
- vol_alive_m3_ha
- vol_dead_standing_m3_ha
- vol_bole_alive_m3_ha
- vol_bole_dead_m3_ha

Usage

```
calc_variables_tree_level(data_dendro, data_stems_calc)
```

Arguments

data_dendro	dataframe on tree measures with variables plot_id, plottype, tree_measure_id, date_dendro, dbh_mm, height_m, species, alive_dead, decaystage, period, old_id, year, subcircle, plotarea_ha,... (output of function load_data_dendrometry())
data_stems_calc	dataframe on stem level measurements with variables plot_id, plottype, tree_measure_id, date_dendro, dbh_mm, height_m, species, alive_dead, decaystage, period, year, subcircle, plotarea_ha,... (output of function calc_variables_stem_level())

Value

Dataframe with ...

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
data_stems <- compose_stem_data(data_dendro, data_shoots)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
data_stems_calc <- calc_variables_stem_level(data_stems, height_model)
calc_variables_tree_level(data_dendro, data_stems_calc)
```

---

calc_veg_core_area_species

aggregate vegetation parameters by core area, species and year

---

Description

This function calculates for each plot, species and year the percentage of subplots in which the species is present and the percentage of subplots where the species is browsed (relative to the plots where it is present). A difference is made between browsed (which contains all damage) and seriously browsed, which is reported if the damage is more than 1/20. This calculation is designed for core areas, that consist of different subplots. Year refers to year of recording of that specific species (source is table `data_herblayer`), and is possibly different for spring flora than for other species in the same subplot.

Usage

```
calc_veg_core_area_species(data_herblayer)
```

Arguments

`data_herblayer`	dataframe on vegetation in the species level ('herb layer') with variables ...

Value

dataframe with columns `plot`, `species`, `year` (year of recording of specific species, possibly different for spring flora), `number_of_subplots` (= number of subplots where the species occurs), `perc_of_subplots` (= percentage of subplots with species), `number_of_subplots_browsed`, `perc_of_subplots_browsed`, `number_of_subplots_seriously_browsed`, `perc_of_subplots_seriously_browsed` and `mean_coverage_class_avera`

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_herblayer_CA <- load_data_herblayer(path_to_fieldmapdb, plottype = "CA")
calc_veg_core_area_species(data_herblayer_CA)
```

---

calc_veg_plot *aggregate vegetation parameters by plot and year*

Description

This function calculates for each plot (subplot in case of core area) and year the total coverage and the number of species in the vegetation layer. Year refers to year of the main vegetation survey (source is table "data_vegetation"), and will in some cases differ from the year of the spring flora survey.

Usage

```
calc_veg_plot(data_vegetation, data_herblayer)
```

Arguments

data_vegetation	dataframe on vegetation with variables ...
data_herblayer	dataframe on vegetation in the species level ('herb layer') with variables ...

Value

dataframe with columns plot, subplot, date, year (year of main vegetation survey, possible deviating year of spring survey not taken into account), number_of_tree_species and min/max/mid cover of the different vegetation layers (moss, herb, shrub, tree), the waterlayer and since 2015 also of the soildisturbance by game.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_vegetation <- load_data_vegetation(path_to_fieldmapdb)
data_herblayer <- load_data_herblayer(path_to_fieldmapdb)
calc_veg_plot(data_vegetation, data_herblayer)
```

---

check_data_deadwood *check table Deadwood from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of table Deadwood (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_deadwood(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_deadwood(path_to_fieldmapdb)
check_data_deadwood(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

---

check_data_fmdb *check tables from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of tables Trees, Shoots, Deadwood, Regeneration, Regspecies, Vegetation, Herblayer, Plots and Plotdetails (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_fmdb(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with layer, ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_fmdb(path_to_fieldmapdb)
check_data_fmdb(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

check_data_herblayer *check table* Herblayer *from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of table Herblayer (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_herblayer(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_herblayer(path_to_fieldmapdb)
check_data_herblayer(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

check_data_plotdetails *check table* Plotdetails *from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of table Plotdetails (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_plotdetails(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_plotdetails(path_to_fieldmapdb)
check_data_plotdetails(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

---

check_data_plots *check table* Plots *from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of table Plots from the given database and checks for missing data or wrong input.

Usage

```
check_data_plots(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_plots(path_to_fieldmapdb)
check_data_plots(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

---

check_data_regeneration

check table `Regeneration` *from* `Fieldmap` *database for inconsistencies*

Description

This function retrieves the important fields of table `Regeneration` (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_regeneration(database, forest_reserve = "all")
```

Arguments

`database`	name of `Fieldmap`/Access database (with specific `Fieldmap` structure) including path
`forest_reserve`	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns `aberrant_field` (which column is wrong) and `anomaly` (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_regeneration(path_to_fieldmapdb)
check_data_regeneration(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

check_data_regspecies *check table* RegSpecies *from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of tables HeightClass and RegSpecies (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_regspecies(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_regspecies(path_to_fieldmapdb)
check_data_regspecies(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

check_data_shoots *check table* Shoots *from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of table Shoots (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_shoots(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_shoots(path_to_fieldmapdb)
check_data_shoots(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

---

check_data_trees *check table* Trees *from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of table Trees (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_trees(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_trees(path_to_fieldmapdb)
check_data_trees(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

---

check_data_vegetation *check table* Vegetation *from* Fieldmap *database for inconsistencies*

Description

This function retrieves the important fields of table Vegetation (of all periods) from the given database and checks for missing data or wrong input.

Usage

```
check_data_vegetation(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_data_vegetation(path_to_fieldmapdb)
check_data_vegetation(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

check_trees_evolution *check table* Trees *from* Fieldmap *database for inconsistencies between periods*

Description

This function retrieves the important fields of table Trees (of all periods) from the given database and checks for anomalies between periods, such as zombies, shifters, outlier_height, outlier_diameter or walkers.

Usage

```
check_trees_evolution(database, forest_reserve = "all")
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
forest_reserve	name of forest reserve for which the records in the database should be checked (defaults to "all")

Value

Dataframe with inconsistent data with ID's and additional columns aberrant_field (which column is wrong) and anomaly (what is wrong with the input)

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
check_trees_evolution(path_to_fieldmapdb)
check_trees_evolution(path_to_fieldmapdb, forest_reserve = "Everzwijnbad")
```

compare_periods_per_plot

compare parameters between periods (years that parameters are measured)

Description

This function compares for each plot (and other provided variables) the differences between periods/years for the column names given in parameter `measure_vars`. It gives results for differences between subsequent measures (based on `period`) and between the last and the first measure. All column names of the dataset that are not added to parameter `measure_vars`, are considered as grouping variables, except for `period`. If the result is not as expected, please verify that the dataset only consists of grouping variables, variables added to `measure_vars` and `period`.

Usage

```
compare_periods_per_plot(dataset, measure_vars, replace_na_in_vars = NA)
```

Arguments

`dataset`	dataframe with values for each `period`, `plot` and `year`, in addition to grouping variables and `measure_vars`
`measure_vars`	column names of variables that should be compared between periods (including year)
`replace_na_in_vars`	column names of variables (`measure_vars`) in which NA should be replaced by 0 in case at least one record is measured in the same `plot_id` in the same `period`. Similar to function `add_zeros()`, this argument allows to add zeros to end up with records for all species or heights in which measures were taken, even if this specific species or height was absent in the period (and thus the tree volume or number of trees being 0 for all periods and plots in which observations were done). Care should be taken to use this argument: it should only be used for aggregated results of measures (e.d. tree volume, tree number,...) and not for measures of individual trees (which will lead to unwanted additional records) or id's referring to coded tables. It is useless to use this argument if only one result per plot is given.

Value

dataframe with columns `plot`, `year_diff`, `n_years`, grouping variables and differences between periods for each column of `measure_vars`

Examples

```
library(forrescalc)
library(dplyr)
treenr_by_plot <-
  read_forresdat_table(tablename = "dendro_by_plot") %>%
  select(
    period, year, plot_id, number_of_tree_species, number_of_trees_ha
  ) %>%
  distinct()
compare_periods_per_plot(
  treenr_by_plot, c("year", "number_of_tree_species", "number_of_trees_ha")
)
```

compose_stem_data *Combine dendro data and shoot data to give detailed stem data*

Description

This function replaces in the given dendrometric data (result from function `load_data_dendrometry()`) the diameters, height, decay stage and info on intact/snag from coppice trees by their separate stems given in the shoot data (result from function `load_data_shoots()`).

Usage

```
compose_stem_data(data_dendro, data_shoots, extra_variables = FALSE)
```

Arguments

`data_dendro`	dataframe on tree measures with variables `plot_id`, `plottype`, `tree_measure_id`, `date_dendro`, `dbh_mm`, `height_m`, `species`, `alive_dead`, `decaystage`, `period`, `old_id`, `year`, `subcircle`, `plotarea_ha`,... (output of function `load_data_dendrometry()`)
`data_shoots`	dataframe on shoots as given from the function `load_data_shoots()`
`extra_variables`	Should additional variables such as `iufro_hght`, `iufro_vital`, `iufro_socia`, `remark` and `common_remark` be added? Default is FALSE (no). ATTENTION: some variables as IUFRO-classes and (common-)remark are • for coppice - collected at shoot level. To include these extra variables, it is necessary to indicate this argument in both load-functions (`load_data_dendrometry()` and `load_data_shoots()`): extra_variables = TRUE.

Value

Dataframe with shoot data

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
compose_stem_data(data_dendro, data_shoots)

#to include iufro-classes and other additional variables:
data_dendro <-
  load_data_dendrometry(path_to_fieldmapdb, extra_variables = TRUE)
data_shoots <-
  load_data_shoots(path_to_fieldmapdb, extra_variables = TRUE)
compose_stem_data(data_dendro, data_shoots, extra_variables = TRUE)
```

`create_statistics` *Calculate statistics for the given dataset*

Description

This function calculates statistics for the given data (e.g. from the git-repository `forresdat`) on the specified level (e.g. forest_reserve, period and species) and for the specified variables (e.g. basal_area and volume). Calculated statistics include number of observations, mean, variance and confidence interval with lower and upper limit (lci and uci).

These summary statistics are calculated on the given data, not taking into account absence of observations unless explicitly added as a record with value zero. E.g. if a certain species only occurs in 3 plots out of 10 and no records are added for the 7 remaining plots, the summary statistics (e.g. mean coverage) are calculated on 3 plots. Records with value zero for certain variables (e.g. coverage of a certain species or number of trees for a certain diameter class) can automatically be added using the function add_zeros().

In case of intervals, the variance and confidence interval are calculated based on the minimum and maximum values of the intervals of the individual records (which is considered a CI, so lci and uci can serve as min and max). For this, `dataset` must contain columns with minimum and maximum values, `variables` must contain a name for the output of this variable, and `interval_information` must contain the variable names for minimum, maximum and output that should be used. In `interval_information` it can be specified if a logarithmic transformation is needed to compensate of unequal interval widths. In this case, mean and the confidence interval are transformed back, but variance is not, as this result would be confusing rather than useful. For typical `forresdat` variables, the default value of `interval_information` can be used and in this case, the variable mentioned in `variables` should be named after the values in `forresdat`, omitting `min_`, `_min`, `max_` or `_max` (see example on interval data).

Usage

```
create_statistics(
  dataset,
  level = c("period", "forest_reserve"),
  variables,
  include_year_range = FALSE,
  na_rm = FALSE,
 interval_information = suppressMessages(read_csv2(system.file("extdata/class_data.csv",
    package = "forrescalc")))
)
```

Arguments

`dataset`	dataset with data to be summarised with at least columns year and period, e.g. table from git repository `forresdat`
`level`	grouping variables that determine on which level the values should be calculated (e.g. forest_reserve, year and species), given as a string or a vector of strings. Defaults to forest_reserve & period.

`variables` variable(s) of which summary statistics should be calculated (given as a string or a vector of strings)

`include_year_range`
Should min_year and max_year be calculated based on a given column year in dataset? Defaults to FALSE.

`na_rm` Should NA values in the dataset be ignored? Defaults to FALSE. If TRUE, levels without any non NA data are kept (resulting in NA values).

`interval_information`
overview of names for interval data, including columns `var_name` (= name for output), `var_min` and `var_max` (= names for minimum and maximum value in input dataset), and `preferred_transformation` (= "log" if log-transformation is desired). Defaults to a table containing all interval variables in `forresdat`, where log transformation is applied in variables where class widths differ. (In cover data in the Longo scale, log transformation is only applied in variables where most observations have a low coverage, e.g. moss cover, in congruence with the fact that class widths only differ in the lower part of the Longo scale.)

Value

dataframe with the columns chosen for level, a column variable with the chosen variables, and the columns `n_obs`, `mean`, `variance`, `lci` (lower limit of confidence interval) and `uci` (upper limit of confidence interval)

Examples

```
library(forrescalc)
dendro_by_plot <- read_forresdat_table(tablename = "dendro_by_plot")
create_statistics(
  dataset = dendro_by_plot,
  level = c("forest_reserve", "period"),
  variables = "vol_alive_m3_ha"
)
dendro_by_diam_plot_species <-
  read_forresdat_table(tablename = "dendro_by_diam_plot_species")
create_statistics(
  dataset = dendro_by_diam_plot_species,
  level = c("forest_reserve", "year", "species", "dbh_class_5cm"),
  variables = c("basal_area_alive_m2_ha", "basal_area_dead_m2_ha")
)
#example on interval data (shrub_cover and tree_cover)
veg_by_plot <- read_forresdat_table(tablename = "veg_by_plot")
create_statistics(dataset = veg_by_plot,
  level = c("forest_reserve", "period", "plottype"),
  variables = c("number_of_species", "shrub_cover", "tree_cover")
)
# example on data with confidence interval (number_established_ha and
# number_seedlings_ha)
reg_by_plot <-
  read_forresdat_table(tablename = "reg_by_plot")
create_statistics(dataset = reg_by_plot,
```

```
  level = c("forest_reserve", "period", "plot_id"),
  variables = c("number_established_ha", "number_seedlings_ha")
)
```

`create_unique_tree_id` *create_unique_tree_id for each individual tree over different years*

Description

This function creates a unique ID for each tree, that allows to group (f.e. by use of `make_table_wide()`) all given information on the life stages of an individual tree during different measures.

Usage

```
create_unique_tree_id(data_dendro)
```

Arguments

`data_dendro`	dataframe on tree measures with variables `plot_id`, `plottype`, `tree_measure_id`, `date_dendro`, `dbh_mm`, `height_m`, `species`, `alive_dead`, `decaystage`, `period`, `old_id`, `year`, `subcircle`, `plotarea_ha`,... (output of function `load_data_dendrometry()`)

Value

a dataset with 1 record per tree measurement, containing the given data of each tree in different years (= `data_dendro`) and a link to a unique `tree_id`.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
data_dendro <-
  load_data_dendrometry(path_to_fieldmapdb, extra_variables = TRUE)
create_unique_tree_id(data_dendro)
```

`from_access_to_forresdat` *copy table(s) from access database to git repository* `forresdat`

Description

This function loads one or more tables from the access database (or an SQLite database) and saves them in the git repository `forresdat`. Table names in camel case in the database are renamed to snake case before saving in `forresdat`.

Usage

```
from_access_to_forresdat(
  database,
  tables,
  repo_path,
  metadata_path,
  push = FALSE,
  strict = TRUE,
  branch = "develop"
)
```

Arguments

`database`	name of `Fieldmap`/Access database (with specific `Fieldmap` structure) including path
`tables`	vector with table names of tables that should be moved
`repo_path`	name and path of local `forresdat` repository in which results/tables should be saved
`metadata_path`	path including .xlsx file in which the metadata are stored
`push`	push commits directly to the remote on GitHub? Default is FALSE (no). (This option can only be used with SSH.)
`strict`	keep default TRUE to update data without structural changes, change to FALSE only if tables are structurally changed (e.g. additional column, change in sorting order,...)
`branch`	branch from repository `forresdat` to which the new version should be committed. Default is 'develop'.

Value

No value is returned, the tables are saved in the git repository.

Examples

```
## Not run:
#make a local clone of forresdat and change path before running
library(forrescalc)
# add path to your local clone of forresdat
path_to_forresdat <- "xxx/forresdat"
# if you don't have a local clone yet, make it:
git2r::clone("https://github.com/inbo/forresdat.git", path_to_forresdat)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
# add path to metadata here
temp <- tempfile(fileext = ".xlsx")
dl <- googledrive::drive_download(
         googledrive::as_id("12x2H9lp86R-AFPdN2JXB9nqwJ2_A6PF6"),
         path = temp, overwrite = TRUE
       )

from_access_to_forresdat(
  database = path_to_fieldmapdb,
  tables = c("qCoverHerbs", "qtotalCover"),
  repo_path = path_to_forresdat,
  metadata_path = temp
)

## End(Not run)
```

`from_forresdat_to_access` *copy table(s) from git repository* `forresdat` *to access database*

Description

This function loads one or more tables from git repository `forresdat` and saves them in an Access (or SQLite) database.

Usage

```
from_forresdat_to_access(
  tables,
  database,
  remove_tables = FALSE,
  plottype = NA,
  join_plotinfo = TRUE
)
```

Arguments

tables	vector with table names of tables that should be moved
database	name of (empty) Access database including path in which results should be saved
remove_tables	overwrite existing tables in database? Default is FALSE, which means tables are not overwritten/deleted unless this parameter is explicitly put on TRUE.
plottype	possibility to select only data for a certain plot type, e.g. 'CP' for Circular plot or 'CA' for Core area (the default NA means that data from all plots are retrieved)
join_plotinfo	should table plotinfo be joined to the chosen table to add columns forest_reserve, survey_dendro/deadw/reg/veg (TRUE or FALSE) and data_processed (TRUE or FALSE)? Default is TRUE. (This is only possible if the given table contains a column plot_id, so this parameter should be put FALSE if this column is absent.) Must be FALSE if you want to load the table plotinfo itself.

Value

No value is returned, the tables are saved in the access database.

Examples

```
library(forrescalc)
# (add path to your own database here)
path_to_database <- "my-db.sqlite"
from_forresdat_to_access(
  tables = "dendro_by_plot",
  database = path_to_database
)
# if tables don't contain column plot_id, or it is not relevant to add
# information on the plots, add argument join_plotinfo = FALSE
from_forresdat_to_access(
  tables = c("qalive_dead", "qdecaystage"),
  database = path_to_database,
  join_plotinfo = FALSE
)

file.remove("my-db.sqlite")
```

---

give_diamclass_5cm *Calculate the diameter class of diameter measure data*

Description

This function returns a factor with diameter classes of 5 cm for a given vector with diameter data in mm.

Usage

```
give_diamclass_5cm(diameterdata)
```

Arguments

diameterdata	vector with diameter data in millimetre

Value

vector with factors in diameter classes of 5 cm

Examples

```
library(forrescalc)
give_diamclass_5cm(c(80, 1512, 2222))
```

---

load_data_deadwood *retrieves data on logs from* Fieldmap *database*

Description

This function queries the given database to retrieve data on deadwood (logs) (ready for use in calculate_dendrometry() function).

Usage

```
load_data_deadwood(
  database,
  plottype = NA,
  forest_reserve = NA,
  extra_variables = FALSE,
  processed = TRUE
)
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
plottype	possibility to select only data for a certain plot type, e.g. 'CP' for Circular plot or 'CA' for Core area (the default NA means that data from all plots are retrieved)
forest_reserve	possibility to select only data for 1 forest reserve by giving the name of the forest reserve (the default NA means that data from all plots are retrieved)
extra_variables	Should additional variables such as remark and common_remark be added? Default is FALSE (no).
processed	Should only processed and surveyed data be added? Defaults to TRUE (yes).

Value

Dataframe with data on logs

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
load_data_deadwood(path_to_fieldmapdb)
```

---

load_data_dendrometry *retrieve dendrometry data from* Fieldmap *database*

Description

This function queries the given database to retrieve data on dendrometry (ready for use in `calculate_dendrometry()` function).

Usage

```
load_data_dendrometry(
  database,
  plottype = NA,
  forest_reserve = NA,
  extra_variables = FALSE,
  processed = TRUE
)
```

Arguments

`database`	name of `Fieldmap`/Access database (with specific `Fieldmap` structure) including path
`plottype`	possibility to select only data for a certain plot type, e.g. 'CP' for Circular plot or 'CA' for Core area (the default NA means that data from all plots are retrieved)
`forest_reserve`	possibility to select only data for 1 forest reserve by giving the name of the forest reserve (the default NA means that data from all plots are retrieved)
`extra_variables`	Should additional variables such as `x_m`, `y_m`, `coppice_id`, `iufro_hght`, `iufro_vital`, `iufro_socia`, `remark` and `common_remark` be added? Default is FALSE (no).
`processed`	Should only processed and surveyed data be added? Defaults to TRUE (yes).

Value

Dataframe with dendrometry data

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
load_data_dendrometry(path_to_fieldmapdb)
```

---

load_data_herblayer *retrieve species specific vegetation data from* Fieldmap *database*

Description

This function queries the given database to retrieve data on vegetation (ready for use in `calculate_vegetation()` function). `year_main_survey` refers to year of the main vegetation survey (source is table `vegetation`), while `year` refers to year of recording of that specific species (possibly different for spring flora; source is table `herblayer`)

Usage

```
load_data_herblayer(
  database,
  plottype = NA,
  forest_reserve = NA,
  processed = TRUE
)
```

Arguments

`database`	name of `Fieldmap`/Access database (with specific `Fieldmap` structure) including path
`plottype`	possibility to select only data for a certain plot type, e.g. 'CP' for Circular plot or 'CA' for Core area (the default NA means that data from all plots are retrieved)
`forest_reserve`	possibility to select only data for 1 forest reserve by giving the name of the forest reserve (the default NA means that data from all plots are retrieved)
`processed`	Should only processed and surveyed data be added? Defaults to TRUE (yes).

Value

Dataframe with vegetation data on the species level ('herb layer'), containing columns as species, coverage_id, browse_index_id, date_vegetation (= date of survey of specific species, different for spring flora and other flora in the same plot), year (= year of survey of specific species, possibly different for spring flora and other flora), ...

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
load_data_herblayer(path_to_fieldmapdb)
```

`load_data_regeneration`

retrieve regeneration data from `Fieldmap` *database*

Description

This function queries the given database to retrieve data on regeneration (ready for use in `calculate_regeneration()` function).

Usage

```
load_data_regeneration(
  database,
  plottype = NA,
  forest_reserve = NA,
  processed = TRUE
)
```

Arguments

`database`	name of `Fieldmap`/Access database (with specific `Fieldmap` structure) including path
`plottype`	possibility to select only data for a certain plot type, e.g. 'CP' for Circular plot or 'CA' for Core area (the default NA means that data from all plots are retrieved)
`forest_reserve`	possibility to select only data for 1 forest reserve by giving the name of the forest reserve (the default NA means that data from all plots are retrieved)
`processed`	Should only processed and surveyed data be added? Defaults to TRUE (yes).

Value

Dataframe with regeneration data

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
load_data_regeneration(path_to_fieldmapdb)
load_data_regeneration(path_to_fieldmapdb, plottype = "CP")
```

load_data_shoots *retrieve data on shoots from* Fieldmap *database*

Description

This function queries the given database to retrieve additional data on shoots to use with dendrometry data.

Usage

```
load_data_shoots(database, extra_variables = FALSE)
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
extra_variables	Should additional variables such as iufro_hght, iufro_vital, iufro_socia, remark and common_remark be added? Default is FALSE (no).

Value

Dataframe with shoot data

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
load_data_shoots(path_to_fieldmapdb)
```

load_data_vegetation *retrieve vegetation data from* Fieldmap *database*

Description

This function queries the given database to retrieve data on vegetation (ready for use in calculate_vegetation function).

Usage

```
load_data_vegetation(
  database,
  plottype = NA,
  forest_reserve = NA,
  processed = TRUE
)
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
plottype	possibility to select only data for a certain plot type, e.g. 'CP' for Circular plot or 'CA' for Core area (the default NA means that data from all plots are retrieved)
forest_reserve	possibility to select only data for 1 forest reserve by giving the name of the forest reserve (the default NA means that data from all plots are retrieved)
processed	Should only processed and surveyed data be added? Defaults to TRUE (yes).

Value

Dataframe with vegetation data, containing columns as total_herb_cover, total_shrub_cover, total_tree_cover, total_soildisturbance_game, date_vegetation (= date of vegetation survey), year_main_survey (= year of vegetation survey), ...

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
load_data_vegetation(path_to_fieldmapdb)
```

---

load_height_models *retrieve height model data from git repository* forresheights

---

Description

This function groups the information on height models from the .csv files in the git repository forresheights together in one dataframe.

Usage

```
load_height_models(example_dataset = FALSE)
```

Arguments

example_dataset
: Should a (limited) example dataset be loaded? Defaults to FALSE, loading the whole dataset from the git repository. If TRUE, only height models needed for the example database will be loaded (to be used in the examples).

Value

Dataframe with height model data

Examples

```
## Not run:
# example ignored during checks due to high elapsed time
library(forrescalc)
load_height_models()

## End(Not run)
```

---

load_plotinfo *retrieve generic plot data from* Fieldmap *database*

Description

This function queries the given database to retrieve additional data on plots to save in forresdat and link with the datasets that are saved there.

Usage

```
load_plotinfo(database, plottype = NA, forest_reserve = NA, processed = TRUE)
```

Arguments

database	name of Fieldmap/Access database (with specific Fieldmap structure) including path
plottype	possibility to select only data for a certain plot type, e.g. 'CP' for Circular plot or 'CA' for Core area (the default NA means that data from all plots are retrieved)
forest_reserve	possibility to select only data for 1 forest reserve by giving the name of the forest reserve (the default NA means that data from all plots are retrieved)
processed	Should only processed and surveyed data be added? Defaults to TRUE (yes).

Value

Dataframe with columns plot_id, plottype, forest_reserve, period, year of dendrometric survey and information on (1) whether there has been a dendro, deadwood, regeneration and/or vegetation survey and (2) whether the data have been processed or not.

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
load_plotinfo(path_to_fieldmapdb)
```

`make_table_wide` *change dataframe from long to wide format*

Description

This function changes a dataframe from long to wide format, e.g. to show data from 2 different periods in different columns.

Usage

```
make_table_wide(table_long, column_to_repeat, columns_for_comparison)
```

Arguments

`table_long`	dataframe with data in long format. It is important that all columns that are not mentioned in the variables `column_to_repeat` or `columns_for_comparison`, are grouping variables that have the same value for each `column_to_repeat` (see second example).
`column_to_repeat`	name of the column of which the values have to be added to the column headings
`columns_for_comparison`	(vector with) name(s) of the column(s) you want to repeat for each value of `column_to_repeat`

Value

the dataframe in long format

Examples

```
library(forrescalc)
library(dplyr)
table_long <- read_forresdat_table(tablename = "dendro_by_plot_species") %>%
  filter(plot_id < 110) %>%
  select(plot_id, species, period, number_of_trees_ha, vol_alive_m3_ha)
table_wide <-
  make_table_wide(
    table_long, column_to_repeat = "period",
    columns_for_comparison = c("number_of_trees_ha", "vol_alive_m3_ha"))
#if number_of_trees_ha is not mentioned in columns_for_comparison, it is
```

```
#considered as a grouping variable while it has different values for each
#period.
#This gives an unwanted result with still many rows and a lot of NA values:
table_wide <-
  make_table_wide(table_long, column_to_repeat = "period",
                  columns_for_comparison = c("vol_alive_m3_ha"))
```

`read_forresdat` *load data package from git repository* `forresdat`

Description

This function reads the data package from git repository `forresdat` (and saves the `forresdat` data to a local temp directory to avoid unneeded downloading in the future). This data package contains both data and metadata and can be explored using functions of the frictionless package.

Data available in `forresdat` only contain observations, so no records with zero values are added for for instance species that were not observed and hence absent. These zero value records can easily be added by using the function `add_zeros()`.

The different tables of this dataset contain data that are collected using 2 different methods (plot types): circular plots (CP) and core areas (CA). It is advised to only use one of them for analyses, as the data are likely to differ due to method related differences.

General information on the plot level is available in table `plotinfo`, which can easily be joined to other tables on `plot_id` and `period` (or only `plot_id` if `period` is absent).

Usage

```
read_forresdat()
```

Value

A `frictionless` data package with all tables and metadata from GitHub repository `forresdat`, which can be explored using package `frictionless`. To be able to recall the version of the data, this data package contains an attribute with the version number of the release of `forresdat` from which the data are taken.

Examples

```
library(forrescalc)
datapackage <- read_forresdat()
frictionless::resources(datapackage)
attr(datapackage, "forresdat")
```

read_forresdat_table *load tables from git repository* forresdat

Description

This function reads a table in .csv format from git repository forresdat (and saves the forresdat data to a local temp directory to avoid unneeded downloading in the future). Data available in forresdat only contains observations, so no records with zero values are added for for instance species that were not observed and hence absent. These zero value records can easily be added by using the function add_zeros(). To load table plotinfo, set argument join_plotinfo = FALSE.

Usage

```
read_forresdat_table(
  tablename,
  join_plotinfo = TRUE,
  plottype = c("CP", "CA", "all")
)
```

Arguments

tablename	name of the table that should be read
join_plotinfo	should table plotinfo be joined to the chosen table to add columns forest_reserve, survey_dendro/deadw/reg/veg (TRUE or FALSE) and data_processed (TRUE or FALSE)? Default is TRUE. (This is only possible if the given table contains a column plot_id, so this parameter should be put FALSE if this column is absent.) Must be FALSE if you want to load the table plotinfo itself.
plottype	Data of which plottype (used method) should be retrieved? Default is 'CP' or 'circle plot', alternatively 'CA' or 'core area', or 'all' (retrieve both circle plots and core areas) could be chosen.

Value

A dataframe with the specified table, default columns plottype, forest_reserve, survey_dendro/deadw/reg/veg (TRUE or FALSE) and data_processed (TRUE or FALSE). To be able to recall the version of the data, this dataframe contains an attribute with the version number of the release of forresdat from which the data are taken.

Examples

```
library(forrescalc)
data_dendro <- read_forresdat_table(tablename = "dendro_by_plot")
data_dendro
attr(data_dendro, "forresdat")
```

remove_last_commit_forresdat

remove last local change from git repository forresdat

Description

This function removes the last commit from the active branch of the specified git repository. ONLY USE THIS FUNCTION IF YOUR COMMIT IS NOT YET PUSHED TO THE REMOTE!!! This function is meant for users that are not familiar with Git to easily remove an automatically generated commit in forresdat after they discovered mistakes in it.

Usage

```
remove_last_commit_forresdat(repo_path)
```

Arguments

repo_path	name and path of local git repository in which last commits should be removed

Value

A dataframe with the specified table

Examples

```
## Not run:
#change paths before running
library(forrescalc)
# add path to your local clone of forresdat
path_to_forresdat <- "xxx/forresdat"

# only run this after writing a commit with `save_results_forresdat()` or
# `from_access_to_forresdat()` that has not yet been pushed to Github!
remove_last_commit_forresdat(repo_path = path_to_forresdat)

## End(Not run)
```

remove_table_forresdat

remove table(s) from data package forresdat

Description

This function removes one or more tables from the data package forresdat by making a commit on a local clone of the git repository. While removing the table(s), it also updates the metadata (.json file)

Usage

```
remove_table_forresdat(tables, repo_path, push = FALSE, branch = "develop")
```

Arguments

tables	vector with table names of tables that should be removed
repo_path	name and path of local forresdat repository in which results/tables should be saved
push	push commits directly to the remote on GitHub? Default is FALSE (no). (This option can only be used with SSH.)
branch	branch from repository forresdat to which the new version should be committed. Default is 'develop'.

Value

No value is returned, the tables are removed from the git repository.

Examples

```
## Not run:
#make a local clone of forresdat and change path before running
library(forrescalc)
# add path to your local clone of forresdat
path_to_forresdat <- "xxx/forresdat"
# if you don't have a local clone yet, make it:
git2r::clone("https://github.com/inbo/forresdat.git", path_to_forresdat)

remove_table_forresdat(
  tables = c("qCoverHerbs", "qtotalCover"),
  repo_path = path_to_forresdat
)

## End(Not run)
```

---

save_results_access *save results of calculations in Access database*

Description

This function saves the results from calculations in the forrescalc package (or any other named list with dataframes) in an Access database. List item names will be used to name each of the tables, which contain as a content the different dataframes.

Usage

```
save_results_access(results, database, remove_tables = FALSE)
```

Arguments

results	results from calculations in package forrescalc as a named list
database	name of (empty) Access database including path in which results should be saved
remove_tables	overwrite existing tables in database? Default is FALSE, which means tables are not overwritten/deleted unless this parameter is explicitly put on TRUE.

Value

No value is returned, data are saved in the specified database

Examples

```
library(forrescalc)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")

# do calculations
data_dendro <- load_data_dendrometry(path_to_fieldmapdb)
data_deadwood <- load_data_deadwood(path_to_fieldmapdb)
data_shoots <- load_data_shoots(path_to_fieldmapdb)
# omit argument 'example_dataset = TRUE' below to use all height models
height_model <- load_height_models(example_dataset = TRUE)
plotinfo <- load_plotinfo(path_to_fieldmapdb)
result_dendro <-
  calculate_dendrometry(
    data_dendro, data_deadwood, data_shoots, height_model, plotinfo)

# save the results
save_results_access(result = result_dendro, database = path_to_fieldmapdb)
# Repeating the previous line of code will give an error, because you try to
# overwrite a table that was already saved in the database on the first run.
# To overwrite previously saved tables, use this command:
save_results_access(
  result = result_dendro, database = path_to_fieldmapdb, remove_tables = TRUE
)
# To remove the tables again in the example database (undo the changes),
# use this code:
con <- forrescalc:::connect_to_database(path_to_fieldmapdb)
for (tablename in names(result_dendro)) {
  DBI::dbRemoveTable(con, tablename)
}
DBI::dbDisconnect(con)
```

save_results_csv *save results of calculations as* .csv*-files*

Description

This function saves the results from calculations in the forrescalc package (or any other named list with dataframes) in a predefined folder. List item names will be used to name each of the tables, which contain as a content the different dataframes.

Usage

```
save_results_csv(results, output_dir)
```

Arguments

results	results from calculations in package forrescalc as a named list
output_dir	name of output folder including path in which results should be saved

Value

No value is returned, data are saved in the specified folder

Examples

```
library(forrescalc)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
path_to_plotlevel_csv <- getwd()
data_regeneration <- load_data_regeneration(database = path_to_fieldmapdb)
regeneration <- calculate_regeneration(data_regeneration)
save_results_csv(results = regeneration, output_dir = path_to_plotlevel_csv)

files <- list.files()
files <- files[grepl("^reg_by.*\\.csv$", files)]
file.remove(files)
```

save_results_forresdat

save results of calculations in git repository forresdat

Description

This function saves the results from calculations by the forrescalc package (or any other named list with dataframes) in git repository forresdat. List item names will be used to name each of the tables, which contain as a content the different dataframes.

Usage

```
save_results_forresdat(
  results,
  repo_path,
  metadata_path,
  push = FALSE,
  strict = TRUE,
  branch = "develop"
)
```

Arguments

results	results from calculations in package forrescalc as a named list of dataframes
repo_path	name and path of local forresdat repository in which results/tables should be saved
metadata_path	path including .xlsx file in which the metadata are stored
push	push commits directly to the remote on GitHub? Default is FALSE (no). (This option can only be used with SSH.)
strict	keep default TRUE to update data without structural changes, change to FALSE only if tables are structurally changed (e.g. additional column, change in sorting order,...)
branch	branch from repository forresdat to which the new version should be committed. Default is 'develop'.

Value

No value is returned, data are saved in the specified git repository

Examples

```
## Not run:
#make a local clone of forresdat and change path before running
library(forrescalc)
# add path to your local clone of forresdat
path_to_forresdat <- "xxx/forresdat"
# if you don't have a local clone yet, make it:
git2r::clone("https://github.com/inbo/forresdat.git", path_to_forresdat)
# (add path to your own fieldmap database here)
path_to_fieldmapdb <-
  system.file("example/database/mdb_bosres.sqlite", package = "forrescalc")
# add path to metadata here
temp <- tempfile(fileext = ".xlsx")
dl <- googledrive::drive_download(
         googledrive::as_id("17M_TfOyjpqLzsFqQ_w1DXitzI7tnULR6"),
         path = temp, overwrite = TRUE
       )

data_regeneration <- load_data_regeneration(path_to_fieldmapdb)
result_regeneration <- calculate_regeneration(data_regeneration)
```

```
save_results_forresdat(
  results = result_regeneration,
  repo_path = path_to_forresdat,
  metadata_path = temp
)

## End(Not run)
```

Index