Package 'FeNEU'

September 9, 2026

Type Package

Title Routines for Evaluating Forest Inventory Data

Version 3.0.2

Date 2026-08-27

Description Provides a collection of routines for evaluating data from larger forest management units, typically sample inventories, but also stand-wise inventories. The idea is to support modern forest planning approaches and to be open by design to new data sources and evaluation methods. For the methodological background of forest inventories see Kangas and Maltamo (2006) ``Forest Inventory: Methodology and Applications'' <doi:10.1007/1-4020-4381-3>.

License AGPL (>= 3)

Encoding UTF-8

Imports tibble, dplyr, purrr, rlang, Rdpack, tidyr, readr, ggplot2, tidyselect (>= 1.2.0), foreach, doSNOW, lifecycle, parallel, progressr, stringr, stringdist, sf

RdMacros Rdpack

Depends R (>= 4.4), ForestElementsR (>= 3.0.0)

Suggests knitr, kableExtra, rmarkdown, tinytex, testthat (>= 3.0.0)

SystemRequirements pandoc (>= 1.12.3) and a LaTeX distribution (e.g. TinyTeX, see ?tinytex::install_tinytex) with the booktabs, longtable, siunitx, and makecell LaTeX packages are required for the *_pdf() output functions.

Config/testthat/edition 3

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Peter Biber [aut, cre] (ORCID: <https://orcid.org/0000-0002-9700-8708>), Astor Toraño Caicoya [aut] (ORCID: <https://orcid.org/0000-0002-9658-8990>)

Maintainer Peter Biber <peter.biber@lwf.bayern.de>

Repository CRAN

Date/Publication 2026-09-09 15:00:10 UTC

Contents

`back_table_dclass` *Create a Background Table for All Mean Diameter Class Based Evaluations From an Inventory Tree Data Frame*

Description

Calculates the quadratic mean diameter on the nested levels of layer in species group in plot, attributes these values to categories which are the basis for further aggregation with other functions. The function was designed for producing often required information only once.

Usage

```
back_table_dclass(inv_trees_plus, d_q_interval = 10)
```

Arguments

`inv_trees_plus`	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an `fe_inventory` object with `pull_trees` and pre-treated with `trees_add_essentials` (see example).
`d_q_interval`	Integer indicating the interval breaks for the d_q (quadratic mean diameter) classes to be formed. The breaks are internally handed over to `cut` as its parameter `breaks`. Default is 10-cm-classes, where all d_q values over 60 cm are all put into the same class.

Value

A data frame which reports for each plot, species group and layer the actual quadratic mean diameter and the d_q-class (as an ordered factor) it belongs to.

Examples

```
# The prepared tree list ships with the package; see
# ?data_ex3_trees_essentials for the chain that builds it.
data_ex3_sample_trees_essentials |>
  back_table_dclass()
```

base_table_age_class *Species Group Information Table by Age Class*

Description

Inventory tree data are grouped by species group and age class. A data frame with group-wise aggregated inventory information is returned.

Usage

```
base_table_age_class(inv_trees_plus, tree_filter = !.data$removal)
```

Arguments

inv_trees_plus	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an fe_inventory object with pull_trees and pre-treated with trees_add_essentials (see example).
tree_filter	Expression describing which trees to use in the function, internally passed to filter. Default is !removal, i.e. trees that died or were removed are not included.

Value

A list containing four data frames. The first one, called *detail* contains the aggregated information by species group and age class. The second one, *total* is aggregated on species level, i.e. one level higher. These values are mostly sums, but not in all cases. Confidence intervals must be calculated differently and also n_plot in total will not always be the sum of the values in *detail*, as there might be plots which contain two layers with different d_q classes. The third data frame, *all_species*, holds the cross-species aggregation by age class (the body of the "Summe" block of the base table (Basistabelle)); the fourth, *all_total*, is the grand total across all species and all classes.

Attached attribute

The returned list carries a tree_selection attribute recording which tree cohort it represents: "mainstand" for the *_main_stand() functions, "alllayers" for the general functions called with their default tree_filter, or the raw tree_filter expression for a custom filter. output_base_table / output_structure_table carry this attribute through to their own output, where the *_pdf() renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: base_table_age_class_main_stand(), base_table_d_q_class(), base_table_d_q_class_mai
output_base_table(), output_increment_overall(), output_structure_table(), structure_table_age_class(),
structure_table_age_class_main_stand(), structure_table_d_q_class(), structure_table_d_q_class_main_s

Examples

```
# The prepared tree list ships with the package; see
# ?data_ex3_trees_essentials for the chain that builds it.
data_ex3_sample_trees_essentials |>
  base_table_age_class()
```

`base_table_age_class_main_stand` *Species Group Information Table by Age Class For the Main Stand Cohort*

Description

Species Group Information Table by Age Class For the Main Stand Cohort

Usage

```
base_table_age_class_main_stand(inv_trees_plus)
```

Arguments

`inv_trees_plus`	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an `fe_inventory` object with `pull_trees` and pre-treated with `trees_add_essentials` (see example).

Value

A list containing four data frames, similar as the output from `base_table_age_class`, however restricted to the main stand, but with information about the areas covered by species (sub-groups). The first data frame in the list, called *detail* contains the aggregated information by species group and age class. The second one, *total* is aggregated on species level, i.e. one level higher. These values are mostly sums, but not in all cases. Confidence intervals must be calculated differently. The third data frame, *all_species*, holds the cross-species aggregation by age class (the body of the "Summe" block of the base table (Basistabelle)); the fourth, *all_total*, is the grand total across all species and all classes.

Attached attribute

The returned list carries a `tree_selection` attribute recording which tree cohort it represents: `"mainstand"` for the `*_main_stand()` functions, `"alllayers"` for the general functions called with their default `tree_filter`, or the raw `tree_filter` expression for a custom filter. `output_base_table` / `output_structure_table` carry this attribute through to their own output, where the `*_pdf()` renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: base_table_age_class(), base_table_d_q_class(), base_table_d_q_class_main_stand(), output_base_table(), output_increment_overall(), output_structure_table(), structure_table_age_class(), structure_table_age_class_main_stand(), structure_table_d_q_class(), structure_table_d_q_class_main_s

Examples

```
# The prepared tree list ships with the package; see
# ?data_ex3_trees_essentials for the chain that builds it.
data_ex3_sample_trees_essentials |>
  base_table_age_class_main_stand()
```

---

base_table_d_q_class *Species Group Information Table by Mean Diameter Class*

Description

Inventory tree data are grouped by species group and mean diameter class. A data frame with group-wise aggregated inventory information is returned.

Usage

```
base_table_d_q_class(inv_trees_plus, dclass_back, tree_filter = !.data$removal)
```

Arguments

inv_trees_plus	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an fe_inventory object with pull_trees and pre-treated with trees_add_essentials (see example).
dclass_back	Data frame listing quadratic mean diameter classes per species group and layer on plot level. Typically the output of back_table_dclass.
tree_filter	Expression describing which trees to use in the function, internally passed to filter. Default is !removal, i.e. trees that died or were removed are not included.

Value

A list containing four data frames. The first one, called *detail* contains the aggregated information by species group and mean diameter class. The second data frame, *total* is aggregated on species level, i.e. one level higher. These values are mostly sums, but not in all cases. Confidence intervals must be calculated differently and also n_plot in total will not always be the sum of the values in *detail*, as there might be plots which contain two layers with different d_q classes. The third data frame, *all_species*, holds the cross-species aggregation by mean diameter class (the body of the "Summe" block of the base table (Basistabelle)); the fourth, *all_total*, is the grand total across all species and all classes.

Attached attribute

The returned list carries a `tree_selection` attribute recording which tree cohort it represents: `"mainstand"` for the `*_main_stand()` functions, `"alllayers"` for the general functions called with their default `tree_filter`, or the raw `tree_filter` expression for a custom filter. `output_base_table` / `output_structure_table` carry this attribute through to their own output, where the `*_pdf()` renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: `base_table_age_class()`, `base_table_age_class_main_stand()`, `base_table_d_q_class_mai`
`output_base_table()`, `output_increment_overall()`, `output_structure_table()`, `structure_table_age_class()`,
`structure_table_age_class_main_stand()`, `structure_table_d_q_class()`, `structure_table_d_q_class_main_s`

Examples

```
# The prepared tree list is shipped with the package: it is this inventory
# put through pull_trees() -> height_complete_inventory() ->
# fill_heights_back() -> pull_trees() -> trees_add_essentials(). See
# ?data_ex3_trees_essentials for that chain spelled out.
trees_with_heights <- data_ex3_sample_trees_essentials

peg_back_d <- back_table_dclass(trees_with_heights)

base_table_d_q_class(trees_with_heights, dclass_back = peg_back_d)
```

---

base_table_d_q_class_main_stand

Species Group Information Table by Mean Diameter Class and Single Tree Diameter Class For the Main Stand Cohort

---

Description

Very similar aggregation table to the one produced by `base_table_d_q_class`, but restricted to to the main stand (layer_key == 1) only. It contains, however, estimates of areas covered by species (sub-)groups and ha-related values o these cohorts. Due the methodological dubiousness of species area calculations in mixed stands, this is only done for the main stand (comparably to how this was handled in the 3rd German National Forest Inventory).

Usage

```
base_table_d_q_class_main_stand(inv_trees_plus, dclass_back)
```

Arguments

`inv_trees_plus`	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an `fe_inventory` object with `pull_trees` and pre-treated with `trees_add_essentials` (see example).

dclass_back — Data frame listing quadratic mean diameter classes per species group and layer on plot level. Typically the output of back_table_dclass.

Value

A list containing four data frames, similar as the output from base_table_d_q_class, however restricted to the main stand, but with information about the areas covered by species (sub-groups). The first data frame in the list, called *detail* contains the aggregated information by species group and mean diameter class. The second one, *total* is aggregated on species level, i.e. one level higher. These values are mostly sums, but not in all cases. Confidence intervals must be calculated differently. The third data frame, *all_species*, holds the cross-species aggregation by mean diameter class (the body of the "Summe" block of the base table (Basistabelle)); the fourth, *all_total*, is the grand total across all species and all classes.

Attached attribute

The returned list carries a tree_selection attribute recording which tree cohort it represents: "mainstand" for the *_main_stand() functions, "alllayers" for the general functions called with their default tree_filter, or the raw tree_filter expression for a custom filter. output_base_table / output_structure_table carry this attribute through to their own output, where the *_pdf() renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: base_table_age_class(), base_table_age_class_main_stand(), base_table_d_q_class(), output_base_table(), output_increment_overall(), output_structure_table(), structure_table_age_class(), structure_table_age_class_main_stand(), structure_table_d_q_class(), structure_table_d_q_class_main_s

Examples

```
# The prepared tree list is shipped with the package: it is this inventory
# put through pull_trees() -> height_complete_inventory() ->
# fill_heights_back() -> pull_trees() -> trees_add_essentials(). See
# ?data_ex3_trees_essentials for that chain spelled out.
trees_with_heights <- data_ex3_sample_trees_essentials

peg_back_d <- back_table_dclass(trees_with_heights)

base_table_d_q_class_main_stand(
  trees_with_heights, dclass_back = peg_back_d
)
```

---

check_pdf_dependencies

Check R Package and System Dependencies for PDF Rendering

Description

Gate function called at the top of every `*_pdf()` function. Throws an informative error when **rmarkdown**, **kableExtra**, or pandoc is missing, so users get a clear install message rather than a cryptic backtrace from inside `rmarkdown::render()`.

Usage

```
check_pdf_dependencies(fun_name)
```

Arguments

`fun_name`	Character string — the name of the calling `*_pdf()` function, used in the error message.

Value

`NULL` invisibly when all dependencies are present.

See Also

`pdf_dependencies`, `diagnose_pdf_toolchain`

Other pdf_output: `output_base_table_pdf()`, `output_increment_base_table_pdf()`, `output_increment_overall_pd`
`output_increment_overview_gnfi3_pdf()`, `output_increment_overview_ytables_pdf()`, `output_structure_table`
`pdf_dependencies`, `plot_info_sheet_pdf()`

Examples

```
# The gate every *_pdf() function calls first: silent when everything is in
# place, an informative error otherwise. Guarded here so that the example
# itself stays harmless on a machine without the toolchain.
if (requireNamespace("kableExtra", quietly = TRUE) &&
    rmarkdown::pandoc_available()) {
  check_pdf_dependencies("output_base_table_pdf")
}
```

`data_ex1_sample_raw` *Example raw sample inventory tables (Format 1, 10 inventory points)*

Description

Artificial but realistic raw forest inventory tables (tree, plot and regeneration) in FeNEU's raw data **Format 1** for sample inventories with concentric circles. They describe an inventory of only ten inventory points. The Gauss-Krueger coordinates in the point table are random numbers and do not refer to any real location. Only the columns the Format-1 converter consumes are kept (see `import_sample_concentric_format1_raw_to_pre`).

Format

data_ex1_sample_raw_points Plot table linked by `plot_id`.

data_ex1_sample_raw_trees Tree table linked by `plot_id`.

data_ex1_sample_raw_smalltrees Regeneration table linked by `plot_id`.

Source

Internal example data (artificial, Format 1)

See Also

Other example data: data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventor
data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory,
data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inventory, data_ex5_sample_fe_inventory,
data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview,
inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans,
processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

---

data_ex2_sample_raw *Example raw inventory tables in BaySF style (forestry-mask format, 10 inventory points)*

Description

Artificial but realistic raw forest inventory tables (plot and tree data) in FeNEU's raw data format 2 for sample inventories. They describe an inventory in the style of the Bavarian State Forest Enterprise (BaySF), comprising only ten inventory points. Any stored coordinates are random numbers and do not refer to any real location.

Format

data_ex2_sample_raw_points Plot table linked by `koord`.

data_ex2_sample_raw_trees Tree table linked by `koord`.

Source

Internal example data (artificial, BaySF style)

See Also

Other example data: data_ex1_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventor
data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory,
data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inventory, data_ex5_sample_fe_inventory,
data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview,
inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans,
processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

`data_ex3_increment_interim` *Interim Increment Stages of the ex3 Inventory Pair*

Description

The repeated-survey increment runs through several steps, each of them an exported function with its own help page. The first two steps are shipped here as ready-made objects, so that the examples of the later steps can start at their own subject instead of rebuilding the chain that leads up to it.

Usage

```
data_ex3_increment_matched

data_ex3_increment_fills
```

Format

`data_ex3_increment_matched` is a list of six elements: `tree_increments` (152 rows, 22 columns), `plot_matches`, `match_plot_stats`, `match_tree_stats`, `inv_period` and `method` (`"rep_classic"`).

`data_ex3_increment_fills` is a list of five estimate data frames – `fills_forward` (31 rows), `fills_backward` (47), `fills_backward_nomatch` (0), `fills_backward_implausible` (1) and `fills_backward_reptrees` (56) – plus `inv_period`, `plot_matches` and `method` carried through from the matching.

An object of class `list` of length 6.

An object of class `list` of length 8.

Details

`data_ex3_increment_matched` is what matching the two surveys yields: the per-tree increments derived from the **measured** data of both surveys, together with the plot and tree matching statistics and the survey period.

`data_ex3_increment_fills` holds the GNFI 3 **estimates that stand ready** to fill the gaps the matching leaves – trees found in only one of the surveys, and trees whose match was flagged implausible. It is not a filled tree list: which of these estimates is actually used is decided one step later by `inv_inc_tree_consolidate` through its `fill_option`. Because it was built with `include_reptrees = TRUE`, it also carries estimates for trees measured in *both* surveys, which have no gap at all; `fill_option = "all_estimates"` needs those.

Interim data for examples and tests

These objects exist for the documentation and the test suite, not as a processing stage of their own. They are built in `data-raw/data_ex3_increment_interim.R` from the shipped ex3 inventory pair and its prepared tree lists (`data_ex3_trees_essentials`):

```
data_ex3_increment_matched <- inv_increment_repsurv_ccirc(
  data_ex3_previous_sample_fe_inventory,
  data_ex3_sample_fe_inventory,
  data_ex3_previous_sample_trees_essentials,
  data_ex3_sample_trees_essentials,
  match_type = "plot_id", plot_id_style = "baysf"
)

data_ex3_increment_fills <- inv_inc_fill_gaps_gnfi3(
  data_ex3_increment_matched,
  data_ex3_previous_sample_trees_essentials,
  data_ex3_sample_trees_essentials,
  include_reptrees = TRUE
)
```

Those are the settings the examples use throughout. An example that deliberately shows a different one – another increment method, matching by centre coordinates, or `include_reptrees = FALSE` – calls the function itself, because that difference is its subject. Whenever the increment chain changes, these objects have to be rebuilt with the script above, just like the prepared tree lists.

See Also

`inv_increment_repsurv_ccirc` and `inv_inc_fill_gaps_gnfi3`, the functions that produce them; `inv_increment_repeated_survey`, which runs the whole chain in one call and is what most users want.

Other example data: `data_ex1_sample_raw`, `data_ex2_sample_raw`, `data_ex3_previous_sample_fe_inventory`, `data_ex3_sample_fe_inventory`, `data_ex3_trees_essentials`, `data_ex4_previous_sample_fe_inventory`, `data_ex4_sample_fe_inventory`, `data_ex5_previous_sample_fe_inventory`, `data_ex5_sample_fe_inventory`, `data_ex6_standwise_fe_inventory`, `data_ex7_standwise_fe_inventory`, `data_examples_overview`, `inc_sub10_rep_classic`, `inc_sub10_rep_end`, `inc_sub10_rep_mean`, `inc_sub10_rep_trans`, `processed_heights_bav_sub10`, `processed_heights_nfi_sub10`, `processed_pulled_sub10`

Examples

```
# Start at the consolidation step, without rebuilding what precedes it
inv_inc_tree_consolidate(
  data_ex3_increment_fills, fill_option = "standard"
)
```

`data_ex3_previous_sample_fe_inventory`

Example Inventory Data as an fe_inventory object (10 inventory points, earlier survey)

Description

Example Inventory Data as an fe_inventory object (10 inventory points, earlier survey)

Details

This fe_inventory object holds artificial but realistic inventory data in the style of the Bavarian State Forest Enterprise (BaySF). It comprises only ten inventory points and was generated with fe_inventory from tab-delimited text files. The stored Gauss-Krueger coordinates are random numbers and do not refer to any real location. It contains the same inventory points as data_ex3_sample_fe_inventory and represents the earlier of two successive surveys (used to demonstrate increment estimation).

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inventory, data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

---

data_ex3_sample_fe_inventory

Example Inventory Data as an fe_inventory object (10 inventory points)

Description

Example Inventory Data as an fe_inventory object (10 inventory points)

Details

This fe_inventory object holds artificial but realistic inventory data in the style of the Bavarian State Forest Enterprise (BaySF). It comprises only ten inventory points and was generated with fe_inventory from tab-delimited text files. The stored Gauss-Krueger coordinates are random numbers and do not refer to any real location. Five of the ten inventory points carry small-tree records, five do not.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventor
data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inventory, data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

data_ex3_trees_essentials

Prepared Tree Lists of the ex3 Inventory Pair

Description

The tree lists of the two ex3 surveys, ready for analysis: heights complete, volumes, standing areas and species groups added. They are shipped so that the increment examples can start from a prepared tree list instead of repeating the five-step preparation in every single `@examples` block.

Usage

```
data_ex3_sample_trees_essentials

data_ex3_previous_sample_trees_essentials
```

Format

Each a tibble with one row per tree and 23 columns, among them `plot_id`, `tree_id`, `species_id`, `species_group`, `age_yr`, `dbh_cm`, `height_m`, `n_rep_ha`, `area_rep_ha`, `layer_key`, `removal`, `g_m2`, `v_hub_m3` and `standing_area_m2`. `data_ex3_sample_trees_essentials` has 139 rows (second survey), `data_ex3_previous_sample_trees_essentials` 112 (first survey).

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 139 rows and 23 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 112 rows and 23 columns.

Details

These are *not* a fourth processing stage of the inventory taxonomy (raw -> pre -> `fe_inventory`, see `data_examples_overview`). They sit after the import, on the analysis side: an `fe_inventory` object carries the inventory, and pulling and preparing its trees is the first step of evaluating it.

How they were built

With the canonical chain, spelled out in `data-raw/data_ex3_trees_essentials.R`:

```
trees_with_heights <- inv |>
  pull_trees() |>
  height_complete_inventory()

inv |>
  fill_heights_back(trees_with_heights) |>
  pull_trees() |>
  trees_add_essentials(method = "BaySF")
```

The second `pull_trees` is what makes the chain canonical: `fill_heights_back` writes the estimated heights into the inventory object, so the re-pulled trees carry them in `height_m` and none of the estimation helper columns (`h_est_m`, `d_q_cm`, `h_q_m`) ride along. Folding `h_est_m` into

`height_m` by hand instead gives the same increments but a tree list of a different shape – a difference that has caused real breakage, see `inv_inc_tree_extend`.

See Also

`data_ex3_sample_fe_inventory` and `data_ex3_previous_sample_fe_inventory`, the inventories they were pulled from; `inv_increment_repeated_survey`, the usual consumer.

Other example data: `data_ex1_sample_raw`, `data_ex2_sample_raw`, `data_ex3_increment_interim`, `data_ex3_previous_sample_fe_inventory`, `data_ex3_sample_fe_inventory`, `data_ex4_previous_sample_fe_inver` `data_ex4_sample_fe_inventory`, `data_ex5_previous_sample_fe_inventory`, `data_ex5_sample_fe_inventory`, `data_ex6_standwise_fe_inventory`, `data_ex7_standwise_fe_inventory`, `data_examples_overview`, `inc_sub10_rep_classic`, `inc_sub10_rep_end`, `inc_sub10_rep_mean`, `inc_sub10_rep_trans`, `processed_heights_bav_sub10`, `processed_heights_nfi_sub10`, `processed_pulled_sub10`

Examples

```
# What the two objects are
dim(data_ex3_sample_trees_essentials)
dim(data_ex3_previous_sample_trees_essentials)
names(data_ex3_sample_trees_essentials)

# They are ready to use, no preparation needed. The usual consumer is
# inv_increment_repeated_survey(); its help page runs the whole chain on
# exactly this pair, so it is not repeated here.
head(data_ex3_sample_trees_essentials[, c("plot_id", "species_id",
                                          "dbh_cm", "height_m", "v_hub_m3")])
```

`data_ex4_previous_sample_fe_inventory`

Example Inventory Data as an fe_inventory object (earlier survey)

Description

Example Inventory Data as an fe_inventory object (earlier survey)

Details

This `fe_inventory` object holds artificial but realistic inventory data in the style of the Bavarian State Forest Enterprise (BaySF). It was generated with `fe_inventory` from tab-delimited text files. The stored Gauss-Krueger coordinates are random numbers and do not refer to any real location. It represents the earlier of two successive surveys and contains the same inventory points as `data_ex4_sample_fe_inventory`, except for three points that were newly established in the later survey; it therefore comprises 97 inventory points.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inventory, data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

`data_ex4_sample_fe_inventory`

Example Inventory Data as an fe_inventory object (100 inventory points)

Description

Example Inventory Data as an fe_inventory object (100 inventory points)

Details

This fe_inventory object holds artificial but realistic inventory data in the style of the Bavarian State Forest Enterprise (BaySF). It comprises one hundred inventory points and was generated with fe_inventory from tab-delimited text files. The stored Gauss-Krueger coordinates are random numbers and do not refer to any real location.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex5_previous_sample_fe_inventory, data_ex5_sample_fe_inver data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

`data_ex5_previous_sample_fe_inventory`

Simulated Inventory Data for Generic Matching Tests (10 plots, first survey)

Description

Simulated Inventory Data for Generic Matching Tests (10 plots, first survey)

Details

This fe_inventory object was derived from a simulated forest enterprise (pure spruce, 400 stands, 5 ha each). It contains a 10-plot subsample selected for unique within-plot tree IDs. The original simulation tree IDs are used as lfd_bhd, making the BaySF tree ID (paste(lfd_satz, lfd_bhd)) a permanent identifier that is stable across inventories. This enables testing of tree_id_style = "generic" (ID-based matching) in inv_increment_repsurv_ccirc.

Seven of the ten plots have both vanished and ingrown trees; three plots have all trees surviving between surveys. The dataset represents the first survey (year 2020). See data_ex5_sample_fe_inventory for the second survey.

Generated by data-raw/simulated_inventory_generic_matching.R.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

---

data_ex5_sample_fe_inventory

Simulated Inventory Data for Generic Matching Tests (10 plots, second survey)

Description

Simulated Inventory Data for Generic Matching Tests (10 plots, second survey)

Details

This fe_inventory object represents the second survey (year 2030) of the same 10 plots as data_ex5_previous_sample_fe
See that object's documentation for details on the data source and the permanent tree ID scheme used for testing tree_id_style = "generic" matching.

Generated by data-raw/simulated_inventory_generic_matching.R.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver
data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

`data_ex6_standwise_fe_inventory`

Stand-Wise Inventory as an fe_inventory object

Description

Stand-Wise Inventory as an fe_inventory object

Details

This `fe_inventory` object (`fe_stand` plots) was built via the two-stage import chain (`import_standwise_relascope_forma` + `import_standwise_relascope_pre_to_fe_inventory`) from the raw Silvarith-style example data shipped at `system.file("extdata", "data_ex6_standwise_raw", package = "FeNEU")`. It has two fictitious stands and carries no plot coordinates, as is typical for stand-wise inventories with angle-count (relascopic, "Winkelzählprobe") surveys.

See Also

Other example data: `data_ex1_sample_raw`, `data_ex2_sample_raw`, `data_ex3_increment_interim`, `data_ex3_previous_sample_fe_inventory`, `data_ex3_sample_fe_inventory`, `data_ex3_trees_essentials`, `data_ex4_previous_sample_fe_inventory`, `data_ex4_sample_fe_inventory`, `data_ex5_previous_sample_fe_inver` `data_ex5_sample_fe_inventory`, `data_ex7_standwise_fe_inventory`, `data_examples_overview`, `inc_sub10_rep_classic`, `inc_sub10_rep_end`, `inc_sub10_rep_mean`, `inc_sub10_rep_trans`, `processed_heights_bav_sub10`, `processed_heights_nfi_sub10`, `processed_pulled_sub10`

`data_ex7_standwise_fe_inventory`

Stand-Wise Inventory as an fe_inventory object

Description

Stand-Wise Inventory as an fe_inventory object

Details

This `fe_inventory` object (`fe_stand` plots) was built with `import_standwise_relascope_pre_to_fe_inventory` from the pre-processed Silvarith-style example data shipped at `system.file("extdata", "data_ex7_standwise_pre", package = "FeNEU")`. It has two fictitious stands and carries no plot coordinates, , as is typical for stand-wise inventories with angle-count (relascopic, "Winkelzählprobe") surveys.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

`data_examples_overview`

FeNEU Example Data

Description

FeNEU ships a set of small example inventories used throughout the documentation, examples and tests. They are fully anonymised and artificial: plot coordinates are randomised and all identifiers are re-assigned, so the data carry no reference to any real forest enterprise and must not be used to draw conclusions about one. The sets are named `data_ex1_...` to `data_ex7_...`. The data represent three different processing stages, and the inventory types currently supported by `FeNEU`.

Processing stages

An inventory passes through up to three stages in FeNEU, and each example is provided at whichever stages are useful for demonstration:

raw The unprocessed input tables as delivered by the field recording software, before any standardisation. Bundled as text files under `system.file("extdata", ...)`, which is what the `*_raw_to_pre()` functions read. For ex1 and ex2, the same tables are additionally bundled as `data/` tibbles (`data_ex1_*`, `data_ex2_*`) so that they can be inspected without reading a file; the text files are generated from those tibbles (`data-raw/`), so the two cannot drift apart.

pre (preprocessed) The intermediate, standardised form the `*_pre_to_fe_inventory()` step consumes – BaySF-style tab-delimited text files for the concentric sample type, or a canonical `WZP_Daten.txt` for the stand-wise type. Bundled as folders under `system.file("extdata", ...)`.

fe_inventory The ready-to-analyse fe_inventory object, bundled as a `data/` object.

Inventory types

FeNEU supports sample and stand-wise inventories. Currently, we cover sample inventories that follow a concentric circle design, and stand-wise inventories with several angle-count samples per stand. In the overview table below, we refer to these types as *sample*, and *standwise*.

sample (concentric circles) Sample-point inventory on concentric circles. Two raw input formats exist for sample inventories, "Format 1" and "Format 2"; both feed the same preprocessed BaySF-style form. Scope is counted in inventory points.

standwise (angle-count / relascope) Stand-wise inventory in which each stand is sampled with several relascope angle-count (Winkelzaehlprobe) points. Currently, only one raw data format is supported for standwise relascope inventories ("Format 1"); the raw and the preprocessed data follow the Silvarith-style. Scope is counted in stands; in the resulting object every angle-count point becomes one `fe_stand` row.

Sample circle geometry

The raw formats of the sample inventories do not carry the geometry of the concentric circles – it is part of the inventory design, not of the field records. It therefore has to be passed to the `*_raw_to_pre()` functions, through the arguments `dbh_cm_from` (the lower dbh threshold of each circle, in cm) and `radiuses_m` (the matching radii, in m). Getting it wrong scales the representation factors and hence every per-hectare figure, so the designs of the two raw example sets are stated here:

Set	`dbh_cm_from`	`radiuses_m`	resulting circle areas
ex1	`c(0, 12, 30, 48)`	`c(2.82, 5.64, 11.28, 17.84)`	25 / 100 / 400 / 1000 m²
ex2	`c(0, 12, 30)`	`c(2, 6.31, 12.62)`	12.6 / 125 / 500 m²

These are the designs of the inventories the anonymised examples were derived from; use them when running the raw import of ex1 or ex2. The sets bundled from the preprocessed stage onwards (ex3 to ex5) carry their circle definition in the data and need no such argument.

Overview

Set	Type	Stages bundled	Scope
ex1	sample (concentric)	raw (Format 1)	10 points
ex2	sample (concentric)	raw (Format 2)	10 points
ex3	sample (concentric)	pre, fe_inventory (+ previous)	10 points (previous 9)
ex4	sample (concentric)	pre, fe_inventory (+ previous)	100 points (previous 95)
ex5	sample (concentric)	pre (simulated), fe_inventory (+ previous)	10 points (previous 10)
ex6	standwise (relascope)	raw (Format 1), pre, fe_inventory	2 stands (17 points)
ex7	standwise (relascope)	pre, fe_inventory	2 stands (15 points)

Datasets in each set

ex1 – sample, Format 1, raw extdata folder `data_ex1_sample_raw` (`Baumschicht.txt`, `Inv_punkt.txt`, `Verjuengung.txt`); objects `data_ex1_sample_raw_trees` (133 rows), `data_ex1_sample_raw_points` (10), `data_ex1_sample_raw_smalltrees` (43). Circle design: `dbh_cm_from = c(0, 12, 30, 48)`, `radiuses_m = c(2.82, 5.64, 11.28, 17.84)`. A dead-wood table exists for developers only (deadwood import is deferred); it is not shipped with the package.

ex2 – sample, Format 2, raw extdata folder `data_ex2_sample_raw` (`02_probekreis.txt`, `01_root_entity.txt`); objects `data_ex2_sample_raw_trees` (177 rows), `data_ex2_sample_raw_points` (10). Circle design: `dbh_cm_from = c(0, 12, 30)`, `radiuses_m = c(2, 6.31, 12.62)`. A small-tree and a dead-wood table are not bundled yet (see note).

ex3 – sample, concentric extdata folder `data_ex3_sample_pre`; objects `data_ex3_sample_fe_inventory` and `data_ex3_previous_sample_fe_inventory` (an earlier survey of the same points, for increment estimation).

ex4 – sample, concentric extdata folder `data_ex4_sample_pre`; objects `data_ex4_sample_fe_inventory` and `data_ex4_previous_sample_fe_inventory`. Same design as ex3, larger.

ex5 – sample, concentric, simulated extdata folders `data_ex5_sample_pre` and `data_ex5_previous_sample_pre`; objects `data_ex5_sample_fe_inventory` and `data_ex5_previous_sample_fe_inventory`. Carries permanent tree ids (`tree_id_style = "generic"`).

ex6 – standwise, relascope extdata folders `data_ex6_standwise_raw` and `data_ex6_standwise_pre`; object `data_ex6_standwise_fe_inventory`.

ex7 – standwise, relascope extdata folder `data_ex7_standwise_pre`; object `data_ex7_standwise_fe_inventory`.

Derived test fixtures

A few further bundled objects are derived from ex3 and kept under their historical names; they are not part of the ex1–ex7 scheme and exist for regression testing: `inc_sub10_rep_classic`, `inc_sub10_rep_mean`, `inc_sub10_rep_end`, `inc_sub10_rep_trans` (increment reference results), `processed_pulled_sub10`, `processed_heights_nfi_sub10` and `processed_heights_bav_sub10` (intermediate tree-preparation results).

Note on the Format-2 example (ex2)

ex2 currently provides only the tree and inventory-point tables. A Format-2 dead-wood table (the "07" sheet) and a regeneration table are not bundled yet; they can be added later.

Stand register (Revierbuch)

Besides the raw -> pre -> fe_inventory chain, FeNEU offers a separate strand that produces a **stand register** (*Revierbuch*) from the inventory input data. It needs stand-level attributes that an `fe_inventory` object does not carry. For stand-wise inventories these are read from a metadata file with `import_standwise_relascope_format1_metadata()`; `import_standwise_relascope_format1_stand_register` orchestrates trees and metadata for `stand_register_pdf()`. Set **ex6** bundles such a metadata file (`stand_metadata.txt`) as an example; the three processing stages above never touch it.

See Also

The exported import functions that move an inventory between the stages illustrated by these example data:

- raw to pre: `import_sample_concentric_format1_raw_to_pre()`, `import_sample_concentric_format2_raw_to_` `import_standwise_relascope_format1_raw_to_pre()`
- pre to fe_inventory: `import_sample_concentric_pre_to_fe_inventory()`, `import_standwise_relascope_pre_t`
- the inventory-type-agnostic dispatcher for the pre to fe_inventory step: `read_and_convert_data()`
- the object constructor itself: `fe_inventory()`
- the stand register strand: `import_standwise_relascope_format1_metadata()`, `import_standwise_relascope_fo` `stand_register_pdf()`

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

`diagnose_pdf_toolchain` *Diagnose the PDF Rendering Toolchain*

Description

Checks whether the tools required for rendering PDF reports (pandoc and a LaTeX distribution) are available on the current system, and prints a message for each missing component explaining how to install it.

Usage

```
diagnose_pdf_toolchain()
```

Value

Invisibly, a named logical list with elements `pandoc` and `latex`.

See Also

setup_pdf_toolchain

Examples

```
diagnose_pdf_toolchain()
```

`fe_inventory` *Standard Construction of an* **fe_inventory** *Object*

Description

Standard Construction of an **fe_inventory** Object

Usage

```
fe_inventory(
  inventory_list,
  object_type = c("fe_stand", "fe_stand_spatial", "fe_ccircle_spatial")
)
```

Arguments

inventory_list	A tibble with the inventory data containing the necessary three columns with plot_id, the plot objects, and the inv_rep_area
object_type	Character to define what type of objects will be save as inventory points: fe_stand or children, fe_stand_spatial or fe_ccircle_spatial

Value

If the information provided allows to construct a valid fe_inventory object, this object will be returned. The function will terminate with an error otherwise

Examples

```
# fe_inventory() is the constructor at the end of the import pipeline. It
# is normally reached through the public entry point read_and_convert_data(),
# which builds the intermediate inventory list and calls fe_inventory()
# internally.
in_path <- system.file("extdata", "data_ex3_sample_pre", package = "FeNEU")
read_and_convert_data(in_path, inventory_type = "sample_concentric",
                      coord_sys = "gk4")
```

---

fill_heights_back *Write Estimated Heights at Inventory Level Back Into the Original fe_inventory Object*

Description

This procedure can be time consuming with large inventories, therefore parallel processing is used (see parameter free_cores).

Usage

```
fill_heights_back(
  inv_dat,
  pulled_est_height,
  small_trees = FALSE,
  .progress = TRUE,
  free_cores = 4
)
```

Arguments

inv_dat	Object of class fe_inventory
pulled_est_height	tibble which was pulled from an fe_inventory object with pull_trees, and where missing heights were completed with height_complete_inventory.

small_trees	**[Experimental]** logical, default is FALSE. Only users who absolutely know what they are doing, should use the setting TRUE. If TRUE also small trees (i.e. trees with heights < 1.3 m) will be dealt with.
.progress	Logical, if TRUE (default) a progress bar is displayed during execution.
free_cores	Integer, indicating the number of processor cores which will not be assigned to parallel processing tasks. If free_cores is equal or greater than the number of available cores, one core will be used (i.e. no parallel processing). Defaults to 4.

Value

The input fe_inventory object with updated heights

Examples

```
# Estimate heights - only the pulled tree tibble is affected so far
suppressWarnings(
# Warnings come from an intentional species code casts - no problem here
  inventory_est_heights <- data_ex3_sample_fe_inventory |>
    pull_trees() |>
    height_complete_inventory(method = "NFI")
)

# Now write the heights back into the original object or, as in this case,
# an otherwise identical copy of it
peg_fe_inventory_updated <- fill_heights_back(
  data_ex3_sample_fe_inventory, inventory_est_heights
)
```

---

generate_circle_definition

Generate a Circle-Definition File for a Concentric Sample Inventory

---

Description

Writes an fe_ikl-keyed circle-definition file (default Probekreisdefinition.txt) for the concentric sample import chain (import_sample_concentric_format1_raw_to_pre, import_sample_concentric_format2_
The concentric importers require a circle definition; when a ready-made one is not at hand, this helper produces a **uniform** definition – the same set of concentric circles for *every* plot – written as a single fe_ikl = "ALL" group. (For multiple circle classes, write the file yourself with one integer fe_ikl per class and link each inventory point to its class via its own fe_ikl column.)

Usage

```
generate_circle_definition(
  output_dir,
  dbh_cm_from,
```

```
  radiuses_m,
  filename = "Probekreisdefinition"
)
```

Arguments

`output_dir`	Folder the file is written to (character). Created if it does not exist.
`dbh_cm_from`	Numeric vector of lower DBH limits in **cm**, one per concentric circle (e.g. `c(0, 12, 30)`).
`radiuses_m`	Numeric vector of circle radii in **m**, the same length as `dbh_cm_from` (e.g. `c(2.82, 5.64, 12.62)`).
`filename`	Base file name with or without the `.txt` extension. Default `"Probekreisdefinition"`.

Details

There is deliberately no default geometry: the caller must supply `dbh_cm_from` and `radiuses_m`, because the evaluation results depend directly on the circle sizes. These parameter names match those of the concentric raw importers.

Value

The path of the written file (invisibly).

See Also

`import_sample_concentric_format1_raw_to_pre`, `import_sample_concentric_format2_raw_to_pre`

Examples

```
dir <- tempdir()
generate_circle_definition(
  dir,
  dbh_cm_from = c(0, 12, 30),
  radiuses_m  = c(2.82, 5.64, 12.62)
)
```

---

get_inv_year *Survey Year of an Inventory Object*

Description

Returns the unique survey year (`time_yr`) of an inventory object. At present a method is provided for `fe_inventory`, where uniformity of `time_yr` across all plots is guaranteed by the `fe_inventory` constructor; the method does a defensive check regardless.

Usage

```
get_inv_year(x, ...)

## S3 method for class 'fe_inventory'
get_inv_year(x, ...)
```

Arguments

x	An inventory object.
...	Currently unused, reserved for method-specific extensions.

Value

A single numeric value: the survey year.

Examples

```
get_inv_year(data_ex3_sample_fe_inventory)
```

---

harmonize_inv_for_repsurv

Harmonize First Inventory for Repeated Survey Increment Calculation

---

Description

For matched plot pairs from two subsequent inventories, this function ensures that the slope values in the first inventory's circle definitions match those of the second inventory. This is important because differing slope assessments lead to different n_rep_ha values for the same tree in the same concentric circle, which can distort increment calculations and cause false circle-transition detections.

Usage

```
harmonize_inv_for_repsurv(inv_1st, inv_2nd, plot_matches)
```

Arguments

inv_1st	An object of class fe_inventory representing the earlier inventory.
inv_2nd	An object of class fe_inventory representing the later inventory.
plot_matches	Output of match_2_inventories.

Details

The function modifies the `circle_definition$slope` of matched plots in `inv_1st` to the value found in the corresponding plot of `inv_2nd`. Missing (`NA`) slopes are treated as 0 (flat terrain). Plots where neither circle definition contains a `slope` column are skipped, as are plots with empty circle definitions (notrees plots). After the slope correction, `n_rep_ha` is recalculated for affected trees using `n_rep_ha`, and the modified plot objects are revalidated with `validate_fe_ccircle_spatial`.

After harmonization, the user should re-derive the tree data from the returned inventory object (e.g. via `pull_trees |> height_complete_inventory |> trees_add_essentials`).

Value

A modified copy of `inv_1st` where the slope values in matched plots have been set to the corresponding values from `inv_2nd` and `n_rep_ha` has been recalculated accordingly. Unmatched plots are returned unchanged.

See Also

Other increment: `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table`
`inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`,
`inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`,
`inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_su`
`inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`,
`match_2_inventories()`, `match_2_inventories_by_center_coord()`, `match_2_inventories_by_plot_id()`,
`match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`
`output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`,
`output_increment_overall_pdf()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pdf`
`output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Other repeated inventory: `inv_increment_repsurv_ccirc()`, `match_2_inventories()`, `match_2_inventories_by_cent`
`match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`,
`match_trees_on_plot_repsurv_ccirc()`, `reclassify_pseudo_ingrowth_ccirc()`

Other inventory matches: `match_2_inventories()`, `match_2_inventories_by_center_coord()`,
`match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`,
`match_trees_on_plot_repsurv_ccirc()`

Examples

```
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

plot_matches <- match_2_inventories(
  inv_a, inv_b, match_type = "plot_id", "baysf"
)

inv_a_harmonized <- harmonize_inv_for_repsurv(inv_a, inv_b, plot_matches)
```

height_complete_inventory *Complete Heights for Non-Measured Trees in an Inventory*

Description

Typically in an inventory, height measurements are done only for a part of the sampled trees. This function provides reasonable estimates for the missing heights.

Usage

```
height_complete_inventory(
  pulled_inventory,
  d_q_interval = 5,
  method = c("NFI", "Bavaria"),
  diagnostic = FALSE
)
```

Arguments

pulled_inventory	Tibble with tree list for the whole inventory, typically the output of pull_trees
d_q_interval	Integer indicating the interval breaks for the dq (quadratic mean diameter) classes to be formed. Default is 5-cm-classes, where all dq values over 60 cm are all put into the same class.
method	Height estimation method for trees with missing heights, "NFI" height curves are defined as default, alternatively "Bavaria" can be selected, but in this case an age must be provided for every tree (see Details)
diagnostic	Logical, if TRUE, the function provides diagnostic output, which is NOT compatible with the standard output and its usage. Default is FALSE. Use TRUE only, if you know exactly what you are doing.

Details

The function first identifies quadratic mean diameter (dq) classes grouped by species group (i.e. fe_species_tum_wwk_short), and stand layer level for each plot. These classes are then applied to the whole inventory (i.e. omitting the grouping by plot), to calculate a dq value for each dq class, species group, and stand layer. From the height-measured trees a corresponding hq (quadratic mean height) is calculated. In case there are no height measured trees in a group, so that there is no data-based estimate of hq possible the function h_q_from_d_q is used as a fallback. In this case a warning is issued.
Both, hq and dq are the entry points for the subsequent individual tree height estimates which are based on the individual trees' dbh. Note that for the height estimation method "Bavaria" also an age must be given for each tree (usually the stand or stand layer age).
The estimation methods "NFI", and "Bavaria" use the standard height curve systems h_standard_gnfi3, and h_standard_bv, respectively.

Value

A tibble in the same format as the input `pulled_inventory`, but with the additional columns `h_est_m`, i.e. height estimates for every tree, `species_group`, `d_q_cm`, and `h_q_m`.

Examples

```
# Pull tree data from an fe_inventory object
pulled_inventory <- data_ex3_sample_fe_inventory |>
  pull_trees()

# Use the NFI method. The h_q fallback warning is expected here where a
# species x layer group has no measured height.
suppressWarnings(
  height_complete_inventory(pulled_inventory, method = "NFI")
)

# ... and the Bavaria method on the same data
suppressWarnings(
  height_complete_inventory(pulled_inventory, method = "Bavaria")
)
```

---

import_sample_concentric_format1_raw_to_pre

Convert raw sample inventory data (format 1) to preprocessed BaySF style data

---

Description

Reads format 1 raw *sample* inventory data collected with the concentric-circle method and writes the five standard BaySF-style intermediate files (`fdinvbhd.txt`, `fdinvba.txt`, `fdinvkrs.txt`, `fcbestku.txt`, `fdvikrs.txt`) into `output_dir`.

Usage

```
import_sample_concentric_format1_raw_to_pre(
  input_path,
  output_dir,
  treelist_filename = "Baumschicht.txt",
  inv_punkt_filename = "Inv_punkt.txt",
  small_trees_filename = NULL,
  circle_def_filename = NULL,
  species_guess = FALSE,
  dbh_cm_from = NULL,
  radiuses_m = NULL,
  coord_sys = NULL,
  inventory_year = NULL,
  encoding = "auto"
)
```

Arguments

`input_path` Path to the folder holding the source files (character).

`output_dir` Folder the preprocessed BaySF-style files are written to (character, required). There is deliberately no default: the function writes files, and a default would write into the folder the raw data came from, i.e. into the user's own filespace. Created if it does not exist. Untouched if validation fails.

Should the plot ids of the tree list and the inventory-point file not match, two diagnostic files listing the offending plots are written here as well: `tree_not_in_inv.txt` (plots with trees but no inventory point – their trees are dropped further down the build) and `inv_not_in_tree.txt` (inventory points without trees). They go to the output folder, never next to the input data, because this is the folder you chose to be written to.

`treelist_filename`

Name of the tree-list file (with or without the `.txt` extension).

Required columns (German alias in parentheses):

`plot_id` Plot identifier (character or integer). Must match the `plot_id` values in `inv_punkt_filename` exactly.

`dbh` (`bhd`) Diameter at breast height, in **cm** (numeric, >= 0; use 0 for small-tree / ingrowth records).

`height` (`hoehe`) Total tree height, in **m** (numeric, 0-50; `NA` if not measured).

`species` (`baumart`) Species code in BaySF integer coding, or a species abbreviation when `species_guess = TRUE` (see the `species_guess` argument).

`angle` (`winkel`) Azimuth from plot centre to the tree stem, in decimal **degrees** (numeric, 0-360). May be `NA` for trees on the innermost circle, which need no position; a larger tree without a position is excluded (with a report warning).

`distance` (`entfernung`) Horizontal distance from plot centre to the tree stem, in **m** (numeric, > 0). Same `NA` rule as `angle`.

`layer` (`schicht`) Stand-layer code (integer): 1 = main stand (OS), 2 = understorey (US), 3 = advance regeneration (VVJ), 4 = residual stand (NHR), 5 = overstorey (UEH), 6 = veteran tree (ALT).

`mortality` (`mortal`) Mortality status (integer): 0 = alive, 1 = recently dead (Frischmortalitaet), 2 = older snag (Altmortalitaet).

`count` (`anzahl`) Number of trees this record represents (integer, usually 1; values > 1 are used for small-tree tally records where individual trees are not located).

Optional columns:

`age` (`alter`) **Stand** (or stand-layer) age in years – never an individual-tree age, even though it is stored redundantly per tree row (numeric). Optional; used if present. Missing or incomplete age limits which analyses are possible and triggers a warning on import – see the "Stand age" section in Details.

`tree_nr` (`baumnummer`) Tree number within the plot. It may be supplied but is **not processed further**: tree identification – in particular when linking two successive inventories – rests solely on the coordinates, and trees without coordinates (the innermost circle) are treated as unmatchable. So no tree number is ever consumed.

Any further columns are ignored – extra columns do not disturb the import, they are simply not used. As FeNEU develops, more columns may become supported (this applies to all raw import formats).

`inv_punkt_filename`

Name of the inventory-point file (with or without the `.txt` extension).

Required columns (German alias in parentheses):

`plot_id` Plot identifier (character or integer). Must match the values in `treelist_filename` exactly.

`coord_x` X coordinate (easting / longitude) of the plot centre, in the coordinate reference system declared either by the optional `coord_sys` column below or by the `coord_sys` argument (numeric).

`coord_y` Y coordinate (northing / latitude) of the plot centre (numeric).

`rep_area` (`repfl`) Area represented by this inventory plot, in **ha** (numeric, > 0).

`slope` (`neigung`) Terrain slope at the plot centre, in **percent** (numeric, >= 0; e.g. 27 means a 27 percent slope). Used for the horizontal-projection (area) correction of the concentric circles; use 0 for level ground. The BaySF-native name `neigung` is accepted as an alias; the pre-reader converts the percent value to the internal slope in degrees.

Optional columns:

`coord_sys` Coordinate reference system of `coord_x`/`coord_y`, one value for the whole file. Either a FeNEU key (`"gk4"`, `"lonlat"`, ... – see the `coord_sys` argument) or an EPSG declaration (`"EPSG:31468"`, `"31468"`). When absent, the `coord_sys` argument must supply it; when both are given they must agree (otherwise the import stops).

`survey_date` (`erfdat`) Survey date in any format parseable by `detect_date_format` (character or Date). Only its *year* is used (see `inventory_year`); the day and month are discarded. If `inventory_year` is given it wins and this column is not read; if it is absent, either `survey_date` or `inventory_year` must be present.

`fe_ikl` Circle-class key (integer). Only needed when the inventory uses more than one circle definition: it links each plot to its class in the per-class circle-definition file (`circle_def_filename`). Omit it when one circle definition applies to all plots (`dbh_cm_from`/`radiuses_m` or an `fe_ikl = "ALL"` file).

Further columns are ignored (they do not disturb the import).

`small_trees_filename`

Name of the small-tree file (with or without the `.txt` extension), or `NULL` (default) if none exists. Small trees are trees below the inventory's usual caliper threshold ("Kluppschwelle") – often even below breast height (1.3 m) – and are therefore recorded with somewhat different methods: un-located tally records (no azimuth/distance). They are carried into the `small_trees` slot of each inventory plot. A record needs at least one size measure: a diameter (`dbh > 0`) or a height. Records with `dbh > 0` are sorted into the regular trees slot; those with `dbh = 0` stay small trees and must then carry a height. **Required columns** (German alias in parentheses):

`plot_id` Plot identifier matching the tree-list.

dbh (bhd) DBH in **cm** (0 for small trees below the caliper threshold).

height (hoehe) Height in **m** (numeric, NA if not measured).

species (baumart) Species code (same coding as main tree list).

count (anzahl) Count of trees this record represents.

Optional columns:

layer (schicht) Stand-layer code (integer), same coding as the tree list; used if present to place the small-tree records in a layer.

age (alter) Stand (stand-layer) age in years; used if present.

tree_nr (baumnummer) Tree / tally number – may be supplied but is not processed further (see the tree-list note above).

Any further columns are ignored (they do not disturb the import).

`circle_def_filename`

Name of the circle-definition file (with or without the `.txt` extension), or `NULL` (default). An `fe_ikl`-keyed **Probekreisdefinition** file with columns `fe_ikl`, `dbh_cm_from` (lower DBH limit, cm) and `radiuses_m` (circle radius, m), one row per concentric circle (the older `bhd_cm_von` / `radius_m` names are still accepted as aliases). Use `fe_ikl = "ALL"` for a single definition applied to every plot (it must then be the only definition); otherwise give one integer `fe_ikl` per circle class and link each inventory point to its class via the point file's own `fe_ikl` column. Create a uniform file with `generate_circle_definition`. Supply **either** this file **or** `dbh_cm_from`/`radiuses_m`, never both.

`species_guess` Logical. If `TRUE` the function tries to map text species abbreviations to numeric BaySF codes using `species_abbreviations_bavrn_state`. Default is `FALSE`.

This is a **stopgap for real-world data** whose species column holds text abbreviations instead of codes. We **strongly recommend supplying the proper numeric BaySF species codes** (`species_guess = FALSE`) instead: guessing relies on exact, case-insensitive, and – as a last resort – fuzzy matching, and the fuzzy tier can silently mis-assign an ambiguous abbreviation. When guessing runs, always check the returned `species_mapping` (especially the `tier == "review"` rows, which also set `needs_review = TRUE`) before trusting the result.

`dbh_cm_from` Numeric vector of lower DBH limits in **cm** for each concentric circle, or `NULL`. Alternative to `circle_def_filename`; must have the same length as `radiuses_m`. A circle definition (file or these arguments) is required.

`radiuses_m` Numeric vector of circle radii in **m**, or `NULL`. Same length as `dbh_cm_from`.

`coord_sys` Coordinate reference system of `coord_x`/`coord_y` in the inventory-point file, or `NULL` (default) to take it from that file's own optional `coord_sys` column. One of `"lonlat"` (EPSG:4326), `"etrs89"` (4258), `"utm32"`/`"utm33"` (25832, 25833), `"gk2"`-`"gk5"` (31466-31469).

The system is never guessed: if neither the column nor this argument states it, the import fails and says so; if both do and they disagree, it fails and names both; and if the declaration contradicts the magnitude of the coordinates (geographic values declared as projected or vice versa), it fails with a concrete suggestion.

The preprocessed `fdinvkrs.txt` stores the plot centres in a **projected metric** CRS – the `gauss_rw`/`gauss_hw` columns carry UTM/Gauss-Krueger by convention, as real BaySF exports do. Geographic input (`lonlat`/`etrs89`) is therefore

reprojected to UTM32 (EPSG:25832) here, when the pre is written; already-projected input (`gk*`, `utm*`) is kept as is. The CRS of the written coordinates is recorded in the `coord_sys` column, so the pre reader picks it up without a second declaration.

`inventory_year` Optional single survey year (e.g. `2025`). It overrides any `survey_date` in the point file and is then the year for every plot; `survey_date` is not parsed. If omitted, the year is taken as the plain calendar year of each plot's `survey_date`. If neither is available the import stops and asks for `inventory_year`.

Downstream only the year is kept (each plot object carries a single `time_yr`). Set `inventory_year` deliberately by the growing season the survey belongs to: a survey done in, e.g., April counts for the previous season, since almost no growth has happened yet. This choice matters most when the increment between two successive inventories is computed later – the period length is the difference of the two inventory years, so an off-by-one year biases the whole increment rate.

`encoding` Character encoding of the source `.txt` files. Default `"auto"` detects the encoding per file (UTF-8, including a byte-order mark, vs. Latin-1/Windows-1252), so users normally never set this. Pass an explicit encoding string only to force a particular one in a pathological case.

Details

Column names in the source files are case-insensitive (converted to lower case internally). Both English canonical names and their German equivalents are accepted; the German alias is silently renamed to the English canonical before validation.

In its raw encoding, Format 1 records measurements in familiar units (DBH in cm, height in m), the tree position as a single azimuth `angle` (degrees) plus a `distance` (m), the stand layer directly as BaySF layer codes, and both living and dead trees. Format 2 encodes the same information differently (DBH in mm, a gon-or-degree back azimuth with a unit flag, its own layer code, living trees only); see `import_sample_concentric_format2_raw_to_pre`.

Stand age. Age in forest inventories always refers to the stand or stand layer, never to the individual tree, even though it is recorded redundantly in every tree row. It is **optional**: in structure-rich, uneven-aged stands – increasingly important in practice – a meaningful stand age often cannot be given. When the age is missing or incomplete, the analyses that group by **age class** are unavailable: base tables and structure tables by age class, and the age-class breakdown of increment from a repeated inventory – all of these can be run by **mean-diameter (dq) class** instead. Increment from yield tables and from the BWI3 (gnfi3) functions requires the age and is not possible without it. A warning is issued on import whenever the age is absent or has gaps, so an accidental omission can be noticed and corrected; users whose forest structure genuinely precludes a stand age can ignore it.

No height completion is performed here. Measured tree heights are carried through as given (missing heights stay `NA`); the converter only records which heights were measured (internal `hmb` flag) and normalises their unit where the raw format requires it. Estimating the missing heights is a deliberate, separate downstream step – `pull_trees() |> height_complete_inventory() |> fill_heights_back()` – so that measured and estimated heights are never silently mixed.

Value

A named list (the common raw-to-pre report): `ok` (logical), `errors` (character), `warnings` (character), `paths` (the written files, or `NULL` if `!ok`), `species_mapping` and `needs_review`. On success it writes `fdinvbhd.txt`, `fdinvba.txt`, `fdinvkrs.txt`, `fcbestku.txt`, and `fdvikrs.txt` into `output_dir`.

When `species_guess = TRUE`, `species_mapping` is a tibble with one row per raw species abbreviation (`raw`, `n_trees`, `canonical`, `species_id`, `name_ger`, `tier`). A `tier` of `"confirmed"` means an exact or case-insensitive match; `"review"` means the code was resolved only by substring / fuzzy / mojibake matching and is worth a manual check. `needs_review` is `TRUE` whenever any row is `"review"` (with a matching warning); both are `NULL`/`FALSE` when `species_guess = FALSE`.

Format 1 and Format 2

FeNEU currently supports two raw data formats for sample inventories with concentric circles. The two grew up in parallel with FeNEU itself and have no fundamental differences in content – they simply reflect different user preferences for how the data are exported. We call them "Format 1" and "Format 2". This function reads **Format 1**; Format 2 is read by `import_sample_concentric_format2_raw_to_pre`. Both produce the same BaySF-style preprocessed files, so everything downstream is identical.

Examples

```
# Use example raw input data shipped with the package
# and write them to a temporary folder.

td <- tempdir()

# Write example tables under the recommended Format-1 file names
readr::write_delim(
  data_ex1_sample_raw_trees,
  file = file.path(td, "Baumschicht.txt"),
  delim = "\t"
)

readr::write_delim(
  data_ex1_sample_raw_points,
  file = file.path(td, "Inv_punkt.txt"),
  delim = "\t"
)

readr::write_delim(
  data_ex1_sample_raw_smalltrees,
  file = file.path(td, "Verjuengung.txt"),
  delim = "\t"
)

# Definition of concentric sample circles
dbh_cm_from <- c(0, 12, 30, 48)
radiuses_m  <- c(2.82, 5.64, 11.28, 17.84)

# Create BaySF-style intermediate files in a separate temporary folder
# (function is called for its side effects). output_dir is mandatory --
```

```
# the converter never picks a target folder itself.
out_dir <- file.path(tempdir(), "pre_ex1")
import_sample_concentric_format1_raw_to_pre(
  input_path = td,
  output_dir = out_dir,
  treelist_filename = "Baumschicht",
  inv_punkt_filename = "Inv_punkt",
  small_trees_filename = "Verjuengung",
  species_guess = TRUE,
  dbh_cm_from = dbh_cm_from,
  radiuses_m = radiuses_m,
  coord_sys = "gk4"
)

# Check that one of the output files was created
file.exists(file.path(out_dir, "fdinvbhd.txt"))
```

`import_sample_concentric_format2_raw_to_pre` *Convert raw sample inventory data (format 2) to preprocessed BaySF style data*

Description

Reads format 2 raw sample inventory data collected with the concentric circle method and writes the standard preprocessed BaySF-style intermediate files (`fdinvbhd.txt`, `fdinvba.txt`, `fdinvkrs.txt`, `fcbestku.txt`, `fdvikrs.txt`) into `output_dir`.

Usage

```
import_sample_concentric_format2_raw_to_pre(
  input_path,
  output_dir,
  treelist_filename = "02_probekreis.txt",
  inv_punkt_filename = "01_root_entity.txt",
  circle_def_filename = NULL,
  species_guess = FALSE,
  dbh_cm_from = NULL,
  radiuses_m = NULL,
  coord_sys = NULL,
  inventory_year = NULL,
  encoding = "auto"
)
```

Arguments

`input_path`	Path to the folder holding the source files (character).

`output_dir` Folder the preprocessed BaySF-style files are written to (character, required). There is deliberately no default: the function writes files, and a default would write into the folder the raw data came from, i.e. into the user's own filespace. Created if it does not exist. Untouched if validation fails.

`treelist_filename`

Name of the tree-list file (Format 2 source table 02, with or without the `.txt` extension). Default `"02_probekreis.txt"` (the format's standard file name).

Required columns:

`koord` Plot identifier (integer). Must match `koord` in the inventory-point file exactly.

`ba_fe` Species code (BaySF `bavrn_state` integer coding).

`bhd` Diameter at breast height, in **mm** (cm × 10); converted to cm internally.

`hoehe_1` Total tree height, in **dm**; converted to m internally (`NA` if not measured).

`pol_entf` Horizontal distance from plot centre, in **cm**.

`pol_wink`**,** `rueckazimut_gon`**,** `rueckazimut_grad`**,** `einheit_azimut` Azimuth of the tree. `pol_wink` (degrees) is used if present; otherwise the back azimuth is read from the column that `einheit_azimut` declares – `1` = `rueckazimut_gon` (gon, converted × 0.9 to degrees), `2` = `rueckazimut_grad` (degrees). `einheit_azimut` is authoritative: if it does not match the populated column, the tree is treated as having *no* usable azimuth rather than reading a value in the wrong unit. Trees outside the innermost circle that end up without a usable position are excluded from the output, with a prominent warning in the report. An innermost-circle tree may legitimately carry no azimuth.

`bestku` Stand-layer code in the **Format-2** coding – *not* the BaySF coding; it is remapped internally. See the "Stand-layer coding" section below.

Optional columns:

`alter_ba` **Stand** (or stand-layer) age in years – never an individual-tree age, even though it is stored redundantly per tree row. Used if present. Missing or incomplete age limits which analyses are possible and triggers a warning on import – see the "Stand age" section in Details.

`baumnummer` A tree number within the plot. It may be supplied but is **not processed further**: tree identification – in particular when linking two successive inventories – rests solely on the coordinates, and trees without coordinates (the innermost circle) are treated as unmatchable, so no tree number is ever consumed.

Any further columns are ignored – extra columns do not disturb the import. Format 2 records only living trees (no mortality column); mortality is set to 0 for every record.

`inv_punkt_filename`

Name of the inventory-point file (Format 2 source table 01, with or without the `.txt` extension). Default `"01_root_entity.txt"` (the format's standard file name).

Required columns:

`koord` Plot key linking the point to its trees.

`lage_probekreismittelpunkt_x` / `lage_probekreismittelpunkt_y` Plot-centre coordinates. Their coordinate reference system is declared either by the file's own **optional** `lage_probekreismittelpunkt_srs` column (an EPSG declaration such as `"EPSG:4326"`, as written by the native export) or by the `coord_sys` argument – see there.

`repfl` Represented area (ha).

Optional columns:

`date_created` Survey date, in any format parseable by `detect_date_format`. Only its *year* is used (see `inventory_year`); the day and month are discarded. If `inventory_year` is given it wins and this column is not read; if it is absent, either `date_created` or `inventory_year` must be present.

`lage_probekreismittelpunkt_srs` CRS declaration; see `coord_sys`.

`neigung` **(or** `slope`**)** Terrain slope, in **percent**. Format 2's multi-table source usually carries no slope; when the column is absent the slope defaults to 0 (level ground, no horizontal-projection area correction of the concentric circles) and a warning is issued. Supply `neigung` only if the plots actually lie on a slope.

Any further columns are ignored – extra columns do not disturb the import.

`circle_def_filename`

Name of the circle-definition file (with or without the `.txt` extension), or `NULL` (default). An `fe_ikl`-keyed **Probekreisdefinition** file with columns `fe_ikl`, `dbh_cm_from` (lower DBH limit, cm) and `radiuses_m` (circle radius, m); use `fe_ikl = "ALL"` for one uniform definition. Create a uniform one with `generate_circle_definition`. Supply **either** this file **or** `dbh_cm_from`/`radiuses_m`, never both.

In practice Format 2 runs **uniform**: its source tables carry no circle-class key (the inventory-point table has no `fe_ikl` column), so one circle definition applies to every plot – give it via `dbh_cm_from`/`radiuses_m` or an `fe_ikl = "ALL"` file. The per-class (multi-`fe_ikl`) path exists only for the shared machinery and is reached only if a user deliberately adds an `fe_ikl` column to the inventory-point file.

`species_guess` Logical. Currently unused; reserved for future species-guessing support. Default is `FALSE`.

`dbh_cm_from` Numeric vector of lower DBH limits (cm) for each concentric circle, or `NULL`. Alternative to `circle_def_filename`; must have the same length as `radiuses_m`. A circle definition (file or these arguments) is required.

`radiuses_m` Numeric vector of circle radii (m), or `NULL`. Same length as `dbh_cm_from`.

`coord_sys` Coordinate reference system of the plot-centre coordinates, or `NULL` (default) to take it from the point file's own optional `lage_probekreismittelpunkt_srs` column. One of `"lonlat"` (EPSG:4326), `"etrs89"` (4258), `"utm32"`/`"utm33"` (25832, 25833), `"gk2"`-`"gk5"` (31466-31469).

The native Format-2 export stores **geographic** coordinates and declares them (`srs = "EPSG:4326"`), so it normally needs no argument here. Nothing is guessed, though: if neither the column nor this argument states the system, the import fails and says so; if both do and they disagree, it fails and names both; and if the declaration contradicts the magnitude of the coordinates, it fails with a concrete

suggestion. (Real exports do contain unusable `srs` values such as `"GUESS!!"`; those count as no declaration.)

The preprocessed `fdinvkrs.txt` stores the plot centres in a **projected metric** CRS – the `gauss_rw`/`gauss_hw` columns carry UTM/Gauss-Krueger by convention, as real BaySF exports do. Geographic input (`lonlat`/`etrs89`) is therefore reprojected to UTM32 (EPSG:25832) here, when the pre is written; already-projected input (`gk*`, `utm*`) is kept as is. The CRS of the written coordinates is recorded in the `coord_sys` column, so the pre reader picks it up without a second declaration.

`inventory_year` Optional single survey year (e.g. `2025`). It overrides any date in the point file and is then the year for every plot; the file's `date_created` is not parsed. If omitted, the year is taken as the plain calendar year of each plot's `date_created`. If neither is available the import stops and asks for `inventory_year`.

Downstream only the year is kept (each plot object carries a single `time_yr`). Set `inventory_year` deliberately by the growing season the survey belongs to: a survey done in, e.g., April counts for the previous season, since almost no growth has happened yet. This choice matters most when the increment between two successive inventories is computed later – the period length is the difference of the two inventory years, so an off-by-one year biases the whole increment rate. The native Format-2 `date_created` is the ambiguous `DD-MM-YY`; for such data passing `inventory_year` explicitly is the recommended path.

`encoding` Character encoding of the source `.txt` files. Default `"auto"` detects the encoding per file (UTF-8, incl. BOM, vs. Latin-1/Windows-1252); pass an explicit encoding only to force one.

Details

Format 2 defines **nine** source tables in total. This converter currently supports only the narrow core – two tables: the main tree list (02) and the inventory-point / coordinate table (01). Importers for the remaining tables are **under construction**: the tree-property data (03), regeneration incl. height classes (04-06, 08), and dead wood (07, 09) are not read yet. In particular, Format 2 has **its own dedicated small-tree / regeneration format**, for which a reader will be added in the future; until then no regeneration / small-tree records are produced, so this converter never yields the `bhd == 0` tally rows that the BaySF-style pre-reader would otherwise split off into small trees.

In its raw encoding, Format 2 records DBH in mm and heights in dm, the tree position as a back azimuth given in gon or degrees (disambiguated by a unit flag) plus a distance in cm, its own stand-layer code (remapped to BaySF codes, see above), and living trees only. Format 1 encodes the same information differently (cm and m, a single degree azimuth, BaySF layer codes directly, living and dead trees); see `import_sample_concentric_format1_raw_to_pre`.

Stand age. Age in forest inventories always refers to the stand or stand layer, never to the individual tree, even though it is recorded redundantly in every tree row. It is **optional**: in structure-rich, uneven-aged stands – increasingly important in practice – a meaningful stand age often cannot be given. When the age is missing or incomplete, the analyses that group by **age class** are unavailable: base tables and structure tables by age class, and the age-class breakdown of increment from a repeated inventory – all of these can be run by **mean-diameter (dq) class** instead. Increment from yield tables and from the BWI3 (gnfi3) functions requires the age and is not possible without it. A warning is issued on import whenever the age is absent or has gaps, so an accidental omission can

be noticed and corrected; users whose forest structure genuinely precludes a stand age can ignore it.

No height completion is performed here. Measured tree heights are carried through as given (missing heights stay NA); the converter only records which heights were measured (internal hmb flag) and normalises their unit where the raw format requires it (Format 2 stores heights in dm). Estimating the missing heights is a deliberate, separate downstream step – `pull_trees() |> height_complete_inventory() |> fill_heights_back()` – so that measured and estimated heights are never silently mixed.

Value

A named list (the common raw-to-pre report): ok (logical), errors (character), warnings (character), and paths (the written files, or NULL if !ok). On success it writes fdinvbhd.txt, fdinvba.txt, fdinvkrs.txt, fcbestku.txt, and fdvikrs.txt into output_dir.

Format 1 and Format 2

FeNEU currently supports two raw data formats for sample inventories with concentric circles. The two grew up in parallel with FeNEU itself and have no fundamental differences in content – they simply reflect different user preferences for how the data are exported. We call them "Format 1" and "Format 2". This function reads **Format 2**; Format 1 is read by import_sample_concentric_format1_raw_to_pre. Both produce the same BaySF-style preprocessed files, so everything downstream is identical.

Stand-layer coding (bestku)

Format 2 codes the stand layer in the tree-list column bestku **differently** from the BaySF-style pre format and from Format 1. The converter therefore **remaps** the Format-2 codes onto the BaySF layer codes that the rest of the pipeline (the fcbestku table and the internal layer_key) expects:

Format 2 (bestku)	**->**	**BaySF code**
1 Hauptschicht	->	1 Oberschicht (main stand)
2 Zwischenschicht	->	2 Unterstand
3 Unterschicht	->	2 Unterstand
4 Vorausverjuengung	->	3 Vorausverjuengung
5 Ueberhaelter/Nachhiebsrest	->	5 Ueberhaelter

Note in particular that the same number means different things in the two codings (e.g. Format-2 bestku = 4 is advance regeneration, whereas BaySF/Format-1 code 4 is residual stand). The main stand (code 1) is identical in both. Format 2's two sub-canopy layers (Zwischenschicht, Unterschicht) are both mapped to the single BaySF Unterstand, and Format 2's combined Ueberhaelter/Nachhiebsrest maps to Ueberhaelter.

Examples

```
# Use example raw input data shipped with the package
# and write them to a temporary folder under the default
# file names, so the converter can be called with its defaults.

td <- tempdir()
```

```
readr::write_delim(
  data_ex2_sample_raw_trees,
  file = file.path(td, "02_probekreis.txt"),
  delim = "\t"
)

readr::write_delim(
  data_ex2_sample_raw_points,
  file = file.path(td, "01_root_entity.txt"),
  delim = "\t"
)

# Definition of concentric sample circles
dbh_cm_from <- c(0, 12, 30)
radiuses_m  <- c(2., 6.31, 12.62)

# Create BaySF-style intermediate files in a separate temporary folder
# (function is called for its side effects; the file names default to the
# format's standard names "02_probekreis.txt" / "01_root_entity.txt").
# output_dir is mandatory -- the converter never picks a target itself.
# The example points carry their own srs (EPSG:4326), so no coord_sys is
# needed; inventory_year is passed because the native date_created is the
# ambiguous DD-MM-YY.
out_dir <- file.path(tempdir(), "pre_ex2")
import_sample_concentric_format2_raw_to_pre(
  input_path = td,
  output_dir = out_dir,
  dbh_cm_from = dbh_cm_from,
  radiuses_m = radiuses_m,
  inventory_year = 2025
)

# Check that one of the output files was created
file.exists(file.path(out_dir, "fdinvbhd.txt"))
```

`import_sample_concentric_pre_to_fe_inventory` *Concentric Sample Inventory: Preprocessed Files to fe_inventory*

Description

The pre-to-fe_inventory step of the concentric sample import chain. Reads a folder of preprocessed BaySF-style files (`fdinvbhd.txt`, `fdinvba.txt`, `fdinvkrs.txt`, `fcbestku.txt`, `fdvikrs.txt`) – produced by `import_sample_concentric_format1_raw_to_pre` / `import_sample_concentric_format2_raw_to_pre`, or supplied directly in that shape – and builds an `fe_inventory` of `fe_ccircle_spatial` plots.

Usage

```
import_sample_concentric_pre_to_fe_inventory(
  input_path,
  style = "baysf",
  coord_sys = NULL,
  check_envelope = TRUE,
  encoding = "auto"
)
```

Arguments

input_path	Folder holding the preprocessed BaySF-style files (character).
style	Format family of the preprocessed data. Currently only "baysf" (Bavarian State Forest style); the argument exists so future concentric styles slot in without changing the call shape.
coord_sys	Coordinate reference system of the plot coordinates; see read_and_convert_data. If NULL, a format-specific default is used ("gk4" for the BaySF style). Geographic coordinates ("lonlat", "etrs89") are accepted and are **reprojected to the matching UTM zone** on import – tree positions are built metrically from distance and azimuth around the plot centre, which degrees do not allow. The plot coordinates of the returned object are then in UTM, and a message states the zone. **State** coord_sys **explicitly for Gauss-Krueger data.** BaySF files normally store the easting *without* the zone prefix (a six-digit Rechtswert), which on its own is not valid Gauss-Krueger. When you declare a "gk*" system, the prefix is added to match your declaration and a message says so. When coord_sys was left NULL and merely defaulted, the same step would be a guess – UTM eastings occupy exactly the same six-digit range – so the import **stops** and asks you to name the system instead of silently altering your coordinates.
check_envelope	Logical, default TRUE. Warn when the coordinates, interpreted with the given coord_sys, fall outside German-speaking Europe – the usual sign of a coord_sys that does not match the data. It is a plausibility hint, not a restriction: set FALSE for inventories that genuinely lie elsewhere.
encoding	Character encoding of the preprocessed .txt files. Default "auto" detects the encoding per file (UTF-8, incl. BOM, vs. Latin-1/Windows-1252); pass an explicit encoding only to force one. The files are read entirely as text and only the consumed numeric columns are parsed, so a comma decimal is never misread as a thousands mark – the same decimal-robustness / encoding handling as the raw converters.

Details

Counterpart to import_standwise_relascope_pre_to_fe_inventory.

Building the inventory objects can take a while for large forest units (thousands of plots). Progress is reported through the **progressr** framework, one step per plot. By default nothing is shown; wrap the call to see a progress bar, and the same code drives a native Shiny progress bar inside shiny::withProgress() / progressr::withProgressShiny():

```
progressr::handlers("cli")
progressr::with_progress(
  inv <- import_sample_concentric_pre_to_fe_inventory(input_path)
)
```

No height completion is performed here either: the returned inventory carries the measured heights as they are (missing ones stay NA). Estimating them is a deliberate, separate downstream step – `pull_trees() |> height_complete_inventory() |> fill_heights_back()`.

Value

An `fe_inventory` of `fe_ccircle_spatial` plots.

A note on the BaySF-style column names

The plot-centre coordinate columns are named `gauss_rw`/`gauss_hw` (Gauss-Krueger Rechtswert / Hochwert) for **historical** reasons, but modern BaySF exports store **UTM32** (ETRS89 / UTM zone 32N) values in them – the "gauss" name is a legacy label, not a statement about the CRS (native files even carry a `GAUSS_KZ` zone field that no longer reflects the actual system). FeNEU therefore never infers the CRS from the column name: pre files it writes itself record the real CRS in a `coord_sys` column (self-describing, so no argument is needed here), while a genuine BaySF export has no such column and you must pass `coord_sys` explicitly – for current data that is almost always `"utm32"`.

See Also

`import_sample_concentric_format1_raw_to_pre`, `read_and_convert_data`

Examples

```
# The bundled example pre carries a coord_sys column (utm32), so it is
# self-describing and needs no coord_sys argument.
inv_path <- system.file("extdata", "data_ex3_sample_pre",
                        package = "FeNEU")
inv <- import_sample_concentric_pre_to_fe_inventory(inv_path)
inv
```

---

import_standwise_relascope_format1_raw_to_pre

Import a Stand-Wise Angle-Count Inventory: Silvarith-style Raw to Preprocessed

---

Description

The raw-to-pre step of the stand-wise relascope ("Winkelzählprobe" / angle-count) import chain. Reads a Silvarith-style raw tree file (`EingabedatenGesamt.txt`: German locale – decimal comma, umlaut headers, `DD.MM.YYYY` dates), validates it (collecting *every* problem rather than stopping at the first), and – if valid – writes the preprocessed `WZP_Daten.txt` (the canonical ASCII / decimal-point form) that `import_standwise_relascope_pre_to_fe_inventory` reads.

Usage

```
import_standwise_relascope_format1_raw_to_pre(
  input_path,
  output_dir,
  treelist_filename = "EingabedatenGesamt.txt",
  encoding = "auto"
)
```

Arguments

input_path	Folder holding the raw file (character).
output_dir	Folder the preprocessed WZP_Daten.txt is written to (character, required). There is deliberately no default: the function writes files, and a default would write into the folder the raw data came from, i.e. into the user's own filespace. Created if it does not exist. Untouched if validation fails.
treelist_filename	Raw file name, with or without the .txt. Default "EingabedatenGesamt", the standard Silvarith export name.
encoding	Character encoding of the source file. Default "auto" detects it (UTF-8, incl. BOM, vs. Latin-1/Windows-1252); pass an explicit encoding only to force one.

Details

This is the direct counterpart to import_sample_concentric_format1_raw_to_pre for concentric sample inventories: a single top-level converter that performs its own validation ("plausi") internally and returns a collected report. There is no separately exported plausi function.

"Silvarith-style" describes the *shape* of the data. Silvarith is a stand-wise angle-count software; this importer targets its export layout without claiming exact or official compatibility. A user who already holds a WZP_Daten.txt – Silvarith also produces one directly, or it was made earlier by this function – skips this step and goes straight to import_standwise_relascope_pre_to_fe_inventory.

Required columns (after tolower() of the header):

stpnr Plot number within the stand (integer).

lfdnrstp Sequential record number within the plot; renamed to lfd_nr.

bestand Stand identifier (character).

best.-fläche Stand area in ha (decimal comma); renamed to bestflaeche.

zf Angle-count factor; renamed to zaehlfaktor.

ba Species code in BaySF coding; renamed to baumart.

alter Age in years (integer).

schicht Silvarith layer code in {0, 1, 2, 3}: 0 = main stand (Hauptschicht), 1 = harvest remnants (Nachhiebsreste), 2 = veterans (Ueberhaelter) – 1 and 2 both fall into the remnant/veteran layer – 3 = understorey (Unter-/Zwischenschicht). Mapped to fe_stand layer_key in import_standwise_relascope_pre_ Silvarith has no code for advance regeneration.

anz. Tree count for this record; renamed to anzahl.

bhd DBH in cm (decimal comma).

höhe Height in m (decimal comma); renamed to hoehe.

stichtag Survey date in DD.MM.YYYY format.

Collected checks: bhd non-negative; schicht in {0, 1, 2, 3}; stichtag parseable as %d.%m.%Y; hoehe > 50 m raises a (non-blocking) warning.

No height completion is performed here, and missing heights are not rejected: measured tree heights are carried through as given (missing ones stay NA). Estimating the missing heights is a deliberate, separate downstream step – pull_trees() |> height_complete_inventory() |> fill_heights_back() – so that measured and estimated heights are never silently mixed. (Angle-count surveys usually measure a height for every tallied tree, so missing heights are uncommon here, but they are tolerated.)

Value

A named list (the common raw-to-pre report):

ok Logical. TRUE if the file passed validation and was written.

errors Character vector of validation problems; empty if ok.

warnings Character vector of non-blocking notices; may be non-empty even when ok.

paths Character vector of written files, or NULL if !ok.

See Also

import_standwise_relascope_pre_to_fe_inventory, import_sample_concentric_format1_raw_to_pre

Examples

```
# The bundled example uses the canonical Silvarith export name
# "EingabedatenGesamt.txt" (the default of `treelist_filename`).
sil_dir <- system.file("extdata", "data_ex6_standwise_raw", package = "FeNEU")
out_dir <- tempdir()
result  <- import_standwise_relascope_format1_raw_to_pre(
  sil_dir, output_dir = out_dir
)
result$ok
result$paths
```

---

import_standwise_relascope_pre_to_fe_inventory

Stand-Wise Relascope Inventory: Preprocessed WZP_Daten.txt to fe_inventory

---

Description

The pre-to-fe_inventory step of the stand-wise relascope ("Winkelzählprobe" / angle-count) import chain. Reads the preprocessed WZP_Daten.txt – produced by import_standwise_relascope_format1_raw_to_pre, or a Silvarith-style WZP_Daten.txt supplied directly – and builds an fe_inventory of fe_stand plots.

Usage

```
import_standwise_relascope_pre_to_fe_inventory(
  input_path,
  style = "silvarith",
  encoding = "auto"
)
```

Arguments

input_path	Folder holding the preprocessed WZP_Daten.txt (character).
style	Format family of the preprocessed data. Currently only "silvarith" (Silvarith-style); the argument exists so future stand-wise styles slot in without changing the call shape.
encoding	Character encoding of the preprocessed WZP_Daten.txt. Default "auto" detects it (UTF-8, incl. BOM, vs. Latin-1/Windows-1252); pass an explicit encoding only to force one. The file is read entirely as text and only the consumed numeric columns are parsed, so a comma decimal is never mis-read as a thousands mark – the same decimal-robustness / encoding handling as the raw converter.

Details

Counterpart to import_sample_concentric_pre_to_fe_inventory.

Building the inventory objects can take a while for large forest units (thousands of plots). Progress is reported through the **progressr** framework, one step per plot. By default nothing is shown; wrap the call to see a progress bar, and the same code drives a native Shiny progress bar inside shiny::withProgress() / progressr::withProgressShiny():

```
progressr::handlers("cli")
progressr::with_progress(
  inv <- import_standwise_relascope_pre_to_fe_inventory(input_path)
)
```

No height completion is performed here either, and missing heights are not rejected: the returned inventory carries the measured heights as they are (missing ones stay NA). Estimating them is a deliberate, separate downstream step – pull_trees() |> height_complete_inventory() |> fill_heights_back().

Value

An fe_inventory of fe_stand plots.

See Also

import_standwise_relascope_format1_raw_to_pre, read_and_convert_data

Examples

```
# Build a preprocessed WZP_Daten.txt from the bundled Silvarith-style example,
# then read it into an fe_inventory.
sil_dir <- system.file("extdata", "data_ex6_standwise_raw", package = "FeNEU")
out_dir <- tempdir()
import_standwise_relascope_format1_raw_to_pre(sil_dir, output_dir = out_dir)
inv <- import_standwise_relascope_pre_to_fe_inventory(out_dir)
inv
```

---

increment_base_table *Inventory-Level Increment Base Table by Species Group and Age or Diameter Class*

Description

Aggregates tree-level increments to the inventory level in the same four-element layout as the static `base_table_age_class` / `base_table_d_q_class` family, but for the annual volume increment (`iv_hub_m3_ha_yr`) instead of standing volume. It is the increment counterpart of the static base tables and the per-subunit breakdown analogue of `inv_inc_big_overview_matches_only`.

Usage

```
increment_base_table(
  tree_inc_extd,
  by_class = c("age", "dq"),
  tree_filter = TRUE
)
```

Arguments

`tree_inc_extd`	Either an increment bundle – the output of `inv_increment_repeated_survey` or of `inv_increment_gnfi3`, which is the usual case, its per-tree frame being taken from it – or a bare per-tree increment data frame.
`by_class`	Either `"age"` (default) or `"dq"`, selecting aggregation by age class or by quadratic mean diameter class. Passed to `inv_inc_tree_2_plot`.
`tree_filter`	Expression describing which trees to keep, internally passed to `filter` on `tree_inc_extd`. Default `TRUE`, i.e. all trees of all layers – including those removed between the two inventories, whose increment is part of the period's total production. Pass e.g. `!.data$removal` to restrict to standing trees.

Details

The function operates on the per-tree increment frame produced by `inv_increment_repeated_survey` (repeated survey) or `inv_increment_gnfi3` (single inventory). Both carry the same per-tree contract, so the same base table is produced regardless of the increment's origin. Internally the trees

are first aggregated to plot level with inv_inc_tree_2_plot, then area-weighted to the inventory level via the shared primitive se_area_weighted_grouped.

This is the "plain" aggregation: each plot contributes its single representation area (area_rep_ha); there is no split into virtual monospecific areas (that is the job of a separate _main_stand variant). The per-hectare values therefore relate to one hectare of total forest, and the per-ha denominator is the full area across all plots in the input. Because totals are linear in the per-plot values, the species rows of detail sum (in iv_m3_yr_total) to the matching all_species row, and total sums to all_total – the central sum-consistency invariant (confidence intervals are not additive and are computed independently per level).

Value

A list of four tibbles, mirroring the static base table family:

detail One row per species group and class.

total One row per species group, across all classes.

all_species One row per class, across all species groups (the body of the "Summe" block).

all_total One row, the grand total.

Each carries n_plot_class (the number of inventory points falling into the group), the annual increment as both an absolute total (iv_m3_yr_total) and per hectare of operation area (iv_m3_ha_yr) with their 95% confidence half-widths (ci95_*), and the counts n_iv_filled / n_iv_total (gnfi3-estimated vs. total tree increments in the group). The plain table carries no per-species area_ha (that is the main-stand variant's job). The list additionally carries a collective element (a list with kind, label, definition) describing the tree collective the table represents: "all_layers" for the default tree_filter, otherwise a "custom" collective reporting the filter. It is carried through to the output and PDF steps so the collective can be labelled without re-deriving it. A meta element carries the header meta attached by inv_inc_tree_extend (survey years, period, area, point count, Berechnungs-/Ergänzungsmethode), or NULL for a single-inventory source.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table_main_stand(), increment_ytables_base_
inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(),
inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(),
inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_su
inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(),
match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(),
match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc()
output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(),
output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd
output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Examples

```
inv_b <- data_ex3_sample_fe_inventory

# Prepare a single inventory's tree data (warnings/messages are ok here)
inv_b_trees <- data_ex3_sample_trees_essentials
```

```
# Single-inventory increment, then the inventory base table
inc <- inv_increment_gnfi3(inv_b, inv_b_trees)
bt  <- increment_base_table(inc, by_class = "age")
bt$detail
bt$all_total
```

`increment_base_table_main_stand` *Inventory-Level Main-Stand Increment Base Table With Virtual Species Areas*

Description

Main-stand counterpart of `increment_base_table`: it restricts the aggregation to the main stand (`layer_key == 1`, standing trees) and expresses the per-hectare increment relative to the cohort's *virtual monospecific area* rather than to one hectare of total forest. This mirrors the static `base_table_age_class_main_sta` / `base_table_d_q_class_main_stand` family. As there, species areas are only formed for the main stand, because area shares of species in mixed multi-layer stands are methodologically dubious (the same stance the Third German National Forest Inventory took).

Usage

```
increment_base_table_main_stand(tree_inc_extd, by_class = c("age", "dq"))
```

Arguments

`tree_inc_extd`	Either an increment bundle – the output of `inv_increment_repeated_survey` or of `inv_increment_gnfi3`, which is the usual case, its per-tree frame being taken from it – or a bare per-tree increment data frame.
`by_class`	Either `"age"` (default) or `"dq"`, passed to `inv_inc_tree_2_plot`.

Details

The virtual areas are built exactly as on the static side: a per-plot correction factor normalises the plot's main-stand standing area to one hectare, and each cohort receives its standing-area share of the plot's representation area. Unlike the static side, the main stand here includes trees removed between the two inventories: their increment counts, and their (inv-1) standing area counts towards the ideal species area, so numerator and denominator span the same collective.

Because every tree that contributes increment also carries a positive standing area (removed trees at inv-1, surviving/ingrowth trees at inv-2), every group that appears has a positive ideal area – so the "species extinct but residual increment" edge case (division by a zero area) cannot arise. It also keeps strong species-share shifts (e.g. after a storm) from producing implausibly large per-ideal-hectare increments, because the removed area enters the denominator as well.

Value

A list of four tibbles (`detail`, `total`, `all_species`, `all_total`) like `increment_base_table` (with `iv_m3_ha_yr` and its CI over the whole operation area), additionally holding `area_ha` (the ideal monospecific species area) and `iv_m3_ha_yr_virt` (the increment per that ideal area, without a CI). Unlike the static main-stand base tables it deliberately carries no per-species area-share column (`per_species`): in the increment context the ideal area is that of a period collective (including removed trees), so a percentage share would be more misleading than informative. Like `increment_base_table`, the list also carries a `collective` element (here fixed to the `"main_stand"` collective, "Hauptbestand") and a `meta` element with the header meta attached by `inv_inc_tree_extend` (`NULL` for a single-inventory source).

See Also

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_su`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center_coord()`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`, `output_increment_overall_pdf()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pdf`, `output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Examples

```
inv_b <- data_ex3_sample_fe_inventory
inv_b_trees <- data_ex3_sample_trees_essentials

inc <- inv_increment_gnfi3(inv_b, inv_b_trees)
bt  <- increment_base_table_main_stand(inc, by_class = "age")
bt$detail
bt$total
```

---

increment_ytables_base_table

Main-Stand Yield-Table Increment Base Table by Species Group and Age or Diameter Class

---

Description

Yield-table ("Ertragstafel") counterpart of `increment_base_table_main_stand`. It breaks the yield-table increment estimate down by species group and age or quadratic-mean-diameter class, for the main stand only, and returns the same four-element layout as the other increment base tables so it flows through `output_increment_base_table` and `output_increment_base_table_pdf` unchanged.

Usage

```
increment_ytables_base_table(
  inv_dat,
  ytable_selection,
  by_class = c("age", "dq"),
  plot_area_weight = TRUE,
  fe_inv = NULL
)
```

Arguments

inv_dat	Normally the result of inv_increment_ytables – the estimate is then taken from it and nothing is computed twice, which matters because determining the site index is the most expensive step of the increment side. Alternatively a tree data frame as pulled from an fe_inventory object after height completion (height_complete_inventory) and trees_add_essentials, in which case ytable_selection is required and the estimate is made here.
ytable_selection	A data frame assigning yield-table names to species, e.g. ytables_bavrn_state_var_1_feneu. Must cover all main-stand species in the data (see inv_increment_ytables). Not needed when inv_dat is an fe_increment_ytables bundle.
by_class	Either "age" (default) or "dq". Selects how the result is grouped, not what is estimated: a cohort has one age and one quadratic mean diameter, so both are labels on the same estimate and the two axes reach the same inventory total.
plot_area_weight	Logical; if TRUE (default) plots are weighted by their represented area, otherwise equally. Ignored when inv_dat is a bundle, which was built with its own setting.
fe_inv	Optional fe_inventory object. If supplied, the operation-wide per-ha increment (iv_m3_ha_yr) relates to the full operation area and the header meta (year, area, point count) is filled; otherwise the stocked main-stand ideal area is used as the reference. Taken from the bundle when one is supplied.

Details

Yield tables are age-indexed and their increment is expressed per hectare of a (virtual) monospecific stand, so this table exists *only* for the main stand and always relates to ideal species areas – there is no "all layers" variant. Each species-group cohort on a plot receives its yield-table increment (see inv_increment_ytables), which is then aggregated to the inventory level. For by_class = "dq" the cohorts are grouped by their quadratic mean diameter class (one per plot x species group, as in back_table_dclass); the underlying per-cohort increment is still the age-indexed yield-table value.

The confidence-interval columns are absent by design: only the repeated- inventory method carries an empirical CI.

Value

A list of four tibbles (detail, total, all_species, all_total) plus a collective element (fixed to "Hauptbestand"), a meta element (marking the source as "ytable" so the PDF is titled "Er-

tragstafelschätzung"), and a `ytables_used` element (the yield tables actually used, carried through so the PDF can list them). Each table carries `n_plot_class`, `iv_m3_yr_total`, `iv_m3_ha_yr` (over the operation area), `area_ha` (ideal species area), and `iv_m3_ha_yr_virt` (increment per ideal area). The per-species levels (`detail`, `total`) additionally carry `site_index` (mean Ertragsklasse); it is omitted from the cross-species `all_species` / `all_total` block, where averaging a site index would be meaningless.

See Also

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_su`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center_coord()`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`, `output_increment_overall_pdf()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pd`, `output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Examples

```
inc <- inv_increment_ytables(
  data_ex3_sample_fe_inventory, data_ex3_sample_trees_essentials,
  ytables_bavrn_state_var_1_feneu
)

# Both axes rest on the same estimate
increment_ytables_base_table(inc, by_class = "age")$all_total
increment_ytables_base_table(inc, by_class = "dq")$all_total
```

---

inc_sub10_rep_classic *Reference Tree Increment Data for Automated Tests (sub_10, rep_classic)*

Description

Reference Tree Increment Data for Automated Tests (sub_10, rep_classic)

Details

Provided for automated testing. Contains the `tree_increments` tibble produced by `inv_increment_repsurv_ccirc` with method `"rep_classic"`, applied to `data_ex3_previous_sample_fe_inventory` (first survey) and `data_ex3_sample_fe_inventory` (second survey), with `match_type = "plot_id"` and `plot_id_style = "baysf"`.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

---

inc_sub10_rep_end *Reference Tree Increment Data for Automated Tests (sub_10, rep_end)*

Description

Reference Tree Increment Data for Automated Tests (sub_10, rep_end)

Details

Provided for automated testing. Contains the `tree_increments` tibble produced by `inv_increment_repsurv_ccirc` with method `"rep_end"`. See `inc_sub10_rep_classic` for details on the input data.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

---

inc_sub10_rep_mean *Reference Tree Increment Data for Automated Tests (sub_10, rep_mean)*

Description

Reference Tree Increment Data for Automated Tests (sub_10, rep_mean)

Details

Provided for automated testing. Contains the `tree_increments` tibble produced by `inv_increment_repsurv_ccirc` with method `"rep_mean"`. See `inc_sub10_rep_classic` for details on the input data.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

---

inc_sub10_rep_trans *Reference Tree Increment Data for Automated Tests (sub_10, rep_trans)*

Description

Reference Tree Increment Data for Automated Tests (sub_10, rep_trans)

Details

Provided for automated testing. Contains the `tree_increments` tibble produced by inv_increment_repsurv_ccirc with method `"rep_trans"`. See inc_sub10_rep_classic for details on the input data.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, processed_heights_bav_sub10, processed_heights_nfi_sub10, processed_pulled_sub10

---

inventory_meta *Inventory-Wide Meta Information for the Table Headers*

Description

Derive the enterprise-level figures that the meta block of the static table PDFs (output_base_table_pdf, output_structure_table_pdf) displays: the inventory year, the total (represented) area, and the number of inventory points. The result is returned in exactly the `list(year, area_ha, n_plots)` shape those functions expect for their `meta` argument (and their `inventory` argument uses this function internally).

Usage

```
inventory_meta(x)
```

Arguments

x	A fe_inventory object.

Details

The figures are read off the whole fe_inventory object, which is the authoritative plot list - so treeless plots (and their area) are counted too, unlike a derivation from an already-filtered tree table. This mirrors how the increment tables derive their header meta from the second inventory.

Value

A named list with three elements: year (the survey year; the most recent one if the plots carry more than one, with a warning), area_ha (sum of the per-plot represented areas), and n_plots (number of inventory points).

Examples

```
inventory_meta(data_ex3_sample_fe_inventory)
```

---

inv_increment_gnfi3 *Estimate the Volume Increment of a Single Inventory With the BWI 3 Growth Functions*

Description

Covers the whole chain from a prepared tree list to a ready-to-aggregate increment result, using the single-tree growth functions of the third German National Forest Inventory (Riedel et al. 2017). It is the single-inventory counterpart of inv_increment_repeated_survey: both return the same kind of object, so everything downstream – the increment base tables, the overview, and their PDF reports – is served in exactly the same way.

Usage

```
inv_increment_gnfi3(
  inv,
  inv_trees,
  dt = 5,
  d_q_interval = 10,
  keep_steps = FALSE
)
```

Arguments

inv	An fe_inventory object – the inventory the tree list was pulled from. Supplies the authoritative plot list (including plots without trees) and the header figures of the reports.
inv_trees	Tree data frame as pulled from inv after height completion (height_complete_inventory, fill_heights_back) and trees_add_essentials.
dt	Time span in years the estimate spans. Positive values look forward, negative values backward. Default is five years; the further such an estimate reaches, the less it can account for removals, mortality and ingrowth.
d_q_interval	Width of the quadratic mean diameter classes in cm, passed to back_table_dclass. Default is 10.
keep_steps	Logical. If TRUE, the intermediate results are returned in the steps element. They are useful for inspection but can become large for real inventories, so the default is FALSE, which leaves steps as NULL.

Details

The increment is estimated *per tree* and for the whole population, not only for the collective that happens to be reported afterwards. Any collective can therefore be aggregated from the result: the main stand, all layers, or a selection of your own via the tree_filter argument of the base-table functions.

Because the estimate ages each tree by dt years, the age of every tree is required. If ages are missing, the function stops rather than silently evaluating the part of the population that happens to carry one.

Value

An object of class fe_increment_gnfi3, which inherits from fe_increment. A list with the elements

trees the per-tree increments, covering all layers,

overview the aggregation to inventory level, overall and by species group,

meta the figures the reports print in their header,

steps the intermediate results, or NULL (see keep_steps).

References

Riedel T, Hennig P, Kroiher F, Polley H, Schmitz F, F. S (2017). *Die dritte Bundeswaldinventur (BWI 2012). Inventur- und Auswertungsmethoden*. Thuenen Institut fuer Waldoekosysteme.

See Also

inv_increment_repeated_survey for the repeated-survey counterpart, increment_base_table and increment_base_table_main_stand for the class tables, output_increment_overview_gnfi3 for the species-group overview.

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(),

inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingu tree_inc_repsurv()

Examples

```
inc <- inv_increment_gnfi3(
  inv       = data_ex3_sample_fe_inventory,
  inv_trees = data_ex3_sample_trees_essentials
)

inc

# Increment of the whole inventory, m3 per hectare and year
inc$overview$overall$iv_total$iv_m3_ha_yr

# Broken down by species group and age class
increment_base_table(inc, by_class = "age")$all_total
```

`inv_increment_repeated_survey`

Increment From a Repeated Inventory – the Whole Chain in One Call

---

Description

Calculates the volume increment of a pair of subsequent inventories and returns everything the output functions need. This is the standard entry point for the repeated-survey increment ("Verfahren 1"): it runs the six-step chain `inv_increment_repsurv_ccirc` -> `inv_inc_fill_gaps_gnfi3` -> `inv_inc_tree_consolidate` -> `inv_inc_tree_extend` -> `inv_inc_tree_extend_combined` -> `inv_inc_big_overview` in the right order, with the tree lists handed on where they are needed, so that callers do not have to reproduce it.

Usage

```
inv_increment_repeated_survey(
  inv_1st,
  inv_2nd,
  inv_1st_trees,
  inv_2nd_trees,
  method = c("rep_classic", "rep_mean", "rep_end", "rep_trans"),
  fill_option = c("standard", "min_estimates", "all_estimates"),
```

```
  match_type = c("plot_id", "center_coord"),
  plot_id_style = c("baysf", "generic"),
  tree_id_style = c("baysf", "generic"),
  inner_circle_non_matchable = FALSE,
  shrinkage_rate_per_decade = 0.1,
  species_check = c("species_group", "species", "warn_only"),
  clamp_plausible_shrinkage = TRUE,
  coord_tolerance = 10,
  include_reptrees = TRUE,
  include_second_only = TRUE,
  d_q_interval = 10,
  dt_second_only = -5,
  style = c("baysf", "generic"),
  progress_bar = TRUE,
  keep_steps = FALSE
)
```

Arguments

`inv_1st`, `inv_2nd`
: The two `fe_inventory` objects, first and second survey. See *Which inventories are admissible*.

`inv_1st_trees`, `inv_2nd_trees`
: The matching tree data frames, as pulled from the two inventories and prepared with the canonical chain `pull_trees` -> `height_complete_inventory` -> `fill_heights_back` -> `pull_trees` -> `trees_add_essentials`.

`method`
: The repeated-survey increment method, one of `"rep_classic"` (default), `"rep_mean"`, `"rep_end"`, `"rep_trans"`. Passed to `inv_increment_repsurv_ccirc`.

`fill_option`
: How the gnfi3 gap-filling estimates are used, one of `"standard"` (default), `"min_estimates"`, `"all_estimates"`. Passed to `inv_inc_tree_consolidate`, where the three options are described. Note that the second-only plots always carry gnfi3 estimates, whatever this argument says – they have no measured increment that could be preferred over an estimate.

`match_type`, `plot_id_style`, `tree_id_style`, `coord_tolerance`
: Plot- and tree-matching arguments, passed to `inv_increment_repsurv_ccirc`.

`inner_circle_non_matchable`, `shrinkage_rate_per_decade`, `species_check`, `clamp_plausible_shrinkage`
: Further arguments of `inv_increment_repsurv_ccirc`, passed through unchanged.

`include_reptrees`
: Passed to `inv_inc_fill_gaps_gnfi3`. Default `TRUE`, which is required for `fill_option = "all_estimates"`.

`include_second_only`
: Logical, whether plots that exist only in the second survey are folded in. Default `TRUE`; see *Whole-enterprise coverage*.

`d_q_interval`
: Numeric, width (cm) of the quadratic mean diameter classes. Passed to `inv_inc_tree_extend`.

`dt_second_only`
: Time span (years, negative = backward) of the gnfi3 estimate for the second-only plots. Default `-5`. It reaches both places that estimate those plots – the

per-tree frame (via inv_inc_tree_extend_combined) and the enterprise-level overview (inv_inc_big_overview's dt_2nd_only) – so that the base tables and the enterprise level keep agreeing whatever value is chosen.

`style` Species grouping style of the enterprise-level overview, "baysf" (default) or "generic". Passed to inv_inc_big_overview.

`progress_bar` Logical, whether the function displays a console progress bar itself. Default TRUE. Set to FALSE when you supply your own **progressr** handler – in particular in a Shiny app, see *Progress reporting*. The progress bars of the inner steps are always suppressed, independently of this argument.

`keep_steps` Logical. If TRUE, the intermediate results of the chain are returned in the steps element. They are useful for inspection, but they hold several per-tree frames and grow with the inventory – for a real inventory they dominate the size of the returned object. The default is FALSE, which leaves steps as NULL.

Details

The individual step functions stay exported and are the expert-level interface: use them when a single stage has to be inspected, varied or replaced. For everything else this function is the intended route, and it is the one whose result is guaranteed to be internally consistent.

Value

An object of class fe_increment_repsurv (which inherits from fe_increment), a list with

trees The per-tree increment data frame covering the whole enterprise, carrying the "inc_meta" attribute. This is the input for increment_base_table and increment_base_table_main_stand.

overview The enterprise-level aggregate, output of inv_inc_big_overview, input for output_increment_overall.

summaric The summaric (*ertragsgeschichtlicher*) increment, output of inv_inc_summaric – the gold standard at enterprise level.

meta The header meta (survey years, period, area, point count, calculation method, fill option).

steps The intermediate results of the chain (tree_level, filled, consolidated, extended_matched) for inspection, or NULL – see keep_steps.

Which inventories are admissible

This is a **sample inventory** method and applies only to sample inventories in concentric circles (*Stichprobeninventur*, taxonomy axis *sample* x *concentric*). Both fe_inventory objects must consist *entirely* of fe_ccircle_spatial plots (its _notrees child included) – the tree matching relies on the plot geometry and the tree positions those objects carry. Stand-wise inventories (*standwise* x *relascope*, plain fe_stand plots) carry no tree positions and therefore have no repeated-survey increment; they are rejected with an error by inv_increment_repsurv_ccirc_check_input, which runs first. The two surveys must further be a genuine pair: the same enterprise, the first one earlier than the second, at least one year apart.

Whole-enterprise coverage

The per-tree result covers the whole enterprise, not just the plots measured twice. Plots that exist only in the second survey (new establishments, sample extensions) cannot have a repeated-survey increment by construction, so they receive a backward gnfi3 estimate via `inv_inc_tree_extend_combined`. This matters for more than completeness: the increment base tables express their per-hectare increment over the *full* operation area, so leaving these plots out would put their area into the denominator while their increment is missing from the numerator, and every per-hectare figure would come out too low. With them folded in, the base tables reproduce the enterprise level (`inv_inc_big_overview`'s `combined` rows) exactly. Set `include_second_only = FALSE` only to reproduce the matched-plots-only view, and be aware of the bias just described.

Progress reporting

The chain reports its seven phases through the **progressr** framework. With the default `progress_bar = TRUE` the function displays a console bar itself, so an interactive user needs no ceremony. With `progress_bar = FALSE` it only *signals* progress and the caller decides what to do with it – notably a Shiny app, which wraps the call in `progressr::withProgressShiny()` and gets its own progress bar without FeNEU knowing anything about Shiny. Always pass `progress_bar = FALSE` when supplying a handler yourself, otherwise two handlers compete. The phase labels are translated (`options(fe_lang = "de")`).

The progress bars of the inner steps are switched off deliberately (`inv_increment_repsurv_ccirc` is called with `progress_bar = FALSE`): the tree matching would otherwise draw its own per-plot bar over the phase bar of this function. Progress is reported at this level only.

See Also

`increment_base_table`, `increment_base_table_main_stand`, `output_increment_overall`, `output_increment_base_`

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center_coord()`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`, `output_increment_overall_pdf()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pdf`, `output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Examples

```
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# The prepared tree lists ship with the package; how they were built with
# the canonical chain is shown in ?data_ex3_trees_essentials
inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# The whole chain in one call
```

```
suppressWarnings(suppressMessages({
  inc <- inv_increment_repeated_survey(
    inv_a, inv_b, inv_a_trees, inv_b_trees,
    method = "rep_classic", fill_option = "standard",
    progress_bar = FALSE
  )
}))

# Enterprise level, and the base tables built on the same object
inc$overview$combined$overall$iv_total
increment_base_table(inc, by_class = "age")$all_total
```

`inv_increment_repsurv_ccirc` *Calculate Tree-Wise Increments From Two Subsequent Inventories Based on Concentric Circle Plots With Spatial Information*

Description

Main function to be used when calculating tree-level increments based on repeated inventories with concentric circle sample plots (`fe_ccircle_spatial`).

Usage

```
inv_increment_repsurv_ccirc(
  inv_1st,
  inv_2nd,
  inv_1st_trees,
  inv_2nd_trees,
  match_type = c("plot_id", "center_coord"),
  plot_id_style = c("baysf", "generic"),
  tree_id_style = c("baysf", "generic"),
  method = c("rep_classic", "rep_mean", "rep_end", "rep_trans"),
  inner_circle_non_matchable = FALSE,
  shrinkage_rate_per_decade = 0.1,
  species_check = c("species_group", "species", "warn_only"),
  clamp_plausible_shrinkage = TRUE,
  coord_tolerance = 10,
  progress_bar = TRUE
)
```

Arguments

`inv_1st`	`fe_inventory` object representing the earlier of the two inventories. All plots of this inventory must have the class `fe_ccircle_spatial`.
`inv_2nd`	`fe_inventory` object representing the later of the two inventories. All plots of this inventory must have the class `fe_ccircle_spatial`.

`inv_1st_trees` A data frame resulting from applying the functions `pull_trees`, `height_complete_inventory`, and `trees_add_essentials` to `inv_1st`. While this could be done internally inside this function, this has typically happened earlier already in an inventory evaluation workflow. As this is a considerably time-consuming process, we deem the redundancy acceptable.

`inv_2nd_trees` A data frame resulting from applying the functions `pull_trees`, `height_complete_inventory`, and `trees_add_essentials` to `inv_2nd`. While this could be done internally inside this function, this has typically happened earlier already in an inventory evaluation workflow. As this is a considerably time-consuming process, we deem the redundancy acceptable.

`match_type` Character string indicating the method for matching inventory plots from the two subsequent inventories. The option "center_coord" matches points whose center coordinates have a distance of no more than `coord_tolerance`, while the option "plot_id" (default) matches plots with the same id.

`plot_id_style` Character string indicating the plot_id style to be used. Only relevant when `match_type == "plot_id"`. The available options are "baysf" (default) and "generic". For a match, the latter ("generic") requires the full plot_ids of two plots to be equal in both inventories. The former ("baysf") only requires the last group of digits in the plot_id for a match due to the specific BaySF data format.

`tree_id_style` Character string that indicates how trees are uniquely identified on inventory plot level. This is required for matching trees on the same plot in both inventories. The options are "baysf" (default) and "generic". In the BaySF case, trees are matched by their polar coordinates, while in the generic case, the tree_ids must mark the same trees in both inventories.

`method` Character string, choices are `"rep_classic"` (default), `"rep_mean"`, `"rep_end"`, and `"rep_trans"`. See Details.

`inner_circle_non_matchable`
Logical, default `FALSE`. Controls the handling of pseudo-ingrowth trees — trees whose dbh crossed the inner circle's threshold between the two surveys and would otherwise be counted as fresh ingrowth although they did exist at the first survey on the inner circle. For `tree_id_style == "baysf"` inner-circle trees of the first inventory are inherently unmatchable, and pseudo-ingrowth identification (via Inv-2 position) and gnfi3 backward reconstruction always run. For `tree_id_style == "generic"` set this argument to `TRUE` when the survey design does not yield reliable tree identifiers on the inner circle (e.g. because very small trees cannot practically be tagged in the field). In that case, Inv-1 inner-circle trees are dropped from the match pool and pseudo-ingrowth identification runs analogously to the BaySF case. See `reclassify_pseudo_ingrowth_ccirc` for the reconstruction details.

`shrinkage_rate_per_decade`
Numeric, the maximum plausible volume shrinkage rate per decade for re-measured trees. Trees exceeding this threshold are flagged as implausible. Uses an exponential decay model internally, scaled to the actual survey interval. Default is `0.10` (i.e. 10 percent per decade).

`species_check` Character string controlling how species consistency is checked for re-measured trees. Options are `"species_group"` (default, flags trees whose species group

differs between surveys), "species" (flags trees whose species id differs), and "warn_only" (only issues a message, does not flag).

clamp_plausible_shrinkage
: Logical. If TRUE (default), re-measured trees with plausible volume shrinkage (i.e. below the implausibility threshold) receive a zero increment instead of a negative one. If FALSE, plausible shrinkage is passed through to the increment for rep_classic, rep_mean, and rep_end. Has no effect on rep_trans, which always clamps negative volume differences to zero. Trees flagged as implausible always receive a zero increment regardless of this setting.

coord_tolerance
: Maximum distance allowed for a match of two inventory plots in the distance unit of the coordinate reference system of inv_a and inv_b; in virtually all relevant cases it is m. Default is 10, i.e. inventory plots whose centers have a distance of up to 10 meters will be accepted as matches.

progress_bar
: Boolean, if TRUE (default), a progress bar is shown during time critical steps of the execution.

Details

Even though the volume increments are calculated on tree level here, they are upscaled to 1 ha. This is important, because otherwise the increment represented by trees that change their representation number per ha between both inventories cannot be calculated correctly. Trees present in the first survey only always get zero increment; such values can be replaced by model-based estimates in a subsequent step. The function allows to choose between four methods of increment calculation (parameter method):

- rep_classic: The volume increment, iv, of trees that are present in both inventories is calculated as $iv = n[2] \cdot v[2] - n[1] \cdot v[1]$, with n and v being the representation number per hectare and the volume of the tree at the first and the second survey, respectively. The volume increment of the ingrowth trees is $iv = n[2] * v[2]$, i.e. such trees contribute to the increment with their full volume. This method is equivalent to the classic stand level increment calculation (German "Ertragsgeschichtlicher Zuwachs"), however, some trees (and plots) may also have negative increments. This might look strange, but is fully valid from a statistical angle of view.
- rep_mean: The volume increment, iv, of trees that are present in both inventories is calculated as $iv = (n[2]+n[1])/2 \cdot (v[2]-v[1])$, and the volume increment of ingrowth trees is calculated as $iv = n[2]*(v[2]-v_t)$, where v_t is the tree's volume when it crossed the sampling threshold. This method is not compatible to the classic stand level increment calculation. Guarantees $iv >= 0$ when clamp_plausible_shrinkage = TRUE.
- rep_end: The volume increment, iv, of trees that are present in both inventories is calculated as $iv = n[2] \cdot (v[2] - v[1])$, and the volume increment of ingrowth trees is calculated as $iv = n[2] * (v[2] - v_t)$, where v_t is the tree's volume when it crossed the sampling threshold. This method is not compatible to the classic stand level increment calculation. Guarantees $iv >= 0$ when clamp_plausible_shrinkage = TRUE.
- rep_trans: The volume increment, iv, of trees that are present in both inventories, and did not cross a sampling threshold, is calculated as $iv = n \cdot (v[2] - v[1])$, with $n = n[1] = n[2]$. For trees that crossed a sampling threshold, the calculation is $iv = n[2] \cdot (v[2]-v_t)+n[1] \cdot (v_t-v[1])$. The volume increment of ingrowth trees is calculated as $iv = n[2]*(v[2]-v_t)$, where v_t

is the tree's volume when it crossed the sampling threshold. This method is not compatible to the classic stand level increment calculation and always guarantees $iv >= 0$ (negative volume differences are clamped to zero regardless of `clamp_plausible_shrinkage`).

Value

A list with six elements: i) `tree_increments` is a data frame that identifies all trees that were appropriate for increment calculation with their plot and tree id in both inventories, and contains their annual and periodic volume increment (in m³/ha under bark after harvest). Note that in case of decreasing representation numbers between both surveys, increments can be also become negative which is statistically correct. ii) `plot_matches` informs about the outcome of the required plot matching across both inventories. It is the output of `match_2_inventories` which is called internally. iii) `match_plot_stats` which gives an overview of the plot matching results in terms of plot numbers and representation areas (i.e. the output of an internal call to `match_plot_statistics`). iv) `match_tree_stats` provides similar overview statistics on tree level. v) `inv_period` is the output of `inv_period`, computed once here so downstream functions can access the survey years and the inter-survey period without having to carry the `fe_inventory` objects further. vi) `method` is the increment method that was used in this call (one of `"rep_classic"`, `"rep_mean"`, `"rep_end"`, `"rep_trans"`); carried through so downstream consumers can label outputs accordingly.

Position in the workflow

This is the *first* of six steps that lead from a pair of inventories to the increment tables and their PDFs. Unless a single stage has to be inspected or varied, call `inv_increment_repeated_survey` instead: it runs the whole chain in one call, hands the tree lists on where they are needed, and – importantly – includes the plots that exist only in the second survey, which a hand-built chain easily forgets (see the coverage warning in `inv_inc_tree_extend`).

See Also

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined(`, `inv_increment_gnfi3()`, `inv_increment_repeated_survey()`, `inv_increment_ytables()`, `is_fe_increment_repsur`, `match_2_inventories()`, `match_2_inventories_by_center_coord()`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`, `output_increment_overall_pdf()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pd`, `output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Other repeated inventory: `harmonize_inv_for_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `reclassify_pseudo_ingrowth_ccirc()`

Examples

```
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory
```

```
inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# The two match types. "center_coord" pairs plots by their centres within
# a tolerance, "plot_id" by their identifiers.
inv_increment_repsurv_ccirc(
  inv_a, inv_b, inv_a_trees, inv_b_trees,
  match_type = "center_coord", coord_tolerance = 10,
  method = "rep_classic"
)

# The four methods differ only in how the increment of a single tree is
# formed; "rep_classic" above and "rep_trans" here are the two that matter
# in practice, "rep_mean" and "rep_end" are set the same way. See the
# increment vignette for the formulae.
inv_increment_repsurv_ccirc(
  inv_a, inv_b, inv_a_trees, inv_b_trees,
  match_type = "plot_id", plot_id_style = "baysf",
  method = "rep_trans"
)
```

`inv_increment_ytables` *Estimate the Volume Increment of a Single Inventory With Yield Tables*

Description

Covers the whole chain from a prepared tree list to a ready-to-aggregate increment result. It is the yield-table counterpart of `inv_increment_repeated_survey` and `inv_increment_gnfi3`: all three return an object of class `fe_increment`, carry their report header in `meta`, and are consumed the same way.

Usage

```
inv_increment_ytables(
  inv,
  inv_trees,
  ytable_selection,
  plot_area_weight = TRUE
)
```

Arguments

`inv`	An `fe_inventory` object – the inventory the tree list was pulled from. Supplies the report header and the reference area for the operation-wide per-hectare increment.
`inv_trees`	Tree data frame as pulled from `inv` after height completion (`height_complete_inventory`, `fill_heights_back`) and `trees_add_essentials`.

`ytable_selection`
: A data frame that assigns names of existing `fe_yield_table` objects to species codes. Must have a column `species_id` which contains the species codes (in a valid species coding provided by the package `ForestElementsR`) and a column `ytable_name` with the name of the corresponding `fe_yield_table` object. See `yield_tables_for_species` for predefined table selections, and `ytables_bavrn_state_var_1_feneu` for the selection FeNEU ships. If `species_id` is not unique, the function terminates with an error.

`plot_area_weight`
: Logical, if TRUE (default), the single plots are weighted with the area they represent. If FALSE all plots are equally weighted.

Details

The symmetry stops where the method does. A yield-table increment is not estimated per tree but for a *cohort*: one species in the main stand of one inventory plot, on the ideal (virtual monospecific) area it occupies there. The estimate needs the cohort's age and its mean height, which give the site index; age and site index give the per-hectare increment of the yield table, which is then corrected by the stocking level (the cohort's basal area over the basal area the table expects). Consequently this bundle holds no per-tree increments, the method is confined to the main stand, and the class tables cannot be filtered to arbitrary collectives the way the per-tree strands can.

A cohort has one age and one quadratic mean diameter, so the class an evaluation is grouped by – age class or diameter class – is a *label* on the result, not part of the estimate. Both labels are carried, and `increment_ytables_base_table` groups by whichever is asked for without estimating anything again.

Value

An object of class `fe_increment_ytables`, which inherits from `fe_increment`. A list with the elements

cohorts one row per inventory plot and species: age, quadratic mean diameter and height, ideal area, site index, stocking level and the annual increment. This is the estimate itself,

overview the species-group overview of the whole inventory,

meta the figures the reports print in their header,

ytables_used the yield tables actually used, per species group,

collective the collective descriptor, always the main stand.

See Also

`increment_ytables_base_table` for the class tables, `output_increment_overview_ytables_pdf` for the overview report, `inv_increment_gnfi3` for the per-tree single-inventory counterpart.

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_survey()`, `inv_increment_repsurv_ccirc()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center_coord()`,

match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr tree_inc_repsurv()

Examples

```
inc <- inv_increment_ytables(
  inv       = data_ex3_sample_fe_inventory,
  inv_trees = data_ex3_sample_trees_essentials,
  ytable_selection = ytables_bavrn_state_var_1_feneu
)

inc

inc$overview
```

---

inv_inc_big_overview *Inventory-Wide Increment Overview for a Repeated Survey, Combining Matched and Second-Only Plots*

Description

Top-level wrapper that runs both inv_inc_big_overview_matches_only (for the plots that appear in both surveys) and inv_inc_big_overview_2nd_only (for the plots that only appear in the second survey, with a short-window backward gnfi3 estimate) and combines their results into one whole-enterprise view. Designed as the data source for user-facing summary tables of the whole forest enterprise.

Usage

```
inv_inc_big_overview(
  tree_inc_rep_gapfill,
  inv_1st_trees,
  inv_2nd_trees,
  inv_2nd,
  style = c("generic", "baysf"),
  inc_summaric,
  dt_2nd_only = -5
)
```

Arguments

tree_inc_rep_gapfill
: Output of inv_inc_fill_gaps_gnfi3.

inv_1st_trees, inv_2nd_trees
: Tree data frames as in inv_inc_big_overview_matches_only.

inv_2nd
: fe_inventory object of the second survey.

style
: Passed through to inv_inc_big_overview_matches_only.

inc_summaric
: Output of inv_inc_summaric, passed through to inv_inc_big_overview_matches_only.

dt_2nd_only
: Estimation period in years for the second-only backward gnfi3 projection (negative for backward). Default -5. Passed through to inv_inc_big_overview_2nd_only.

Details

Output layout (the central design decision): three sub-overview slots that all keep the same internal shape ($overall + $by_species_group), plus the inv_period and method info passed through so downstream renderers can label survey years and the increment method without re-deriving them.

matches_only Verbatim output of inv_inc_big_overview_matches_only.

second_only Verbatim output of inv_inc_big_overview_2nd_only.

combined Per-row addition of matches_only and second_only. One row per matched fill_option (combined_standard, combined_min_estimates, combined_all_estimates, combined_inc_summaric), each summing the matched fill_option result with the single gnfi3_backward second_only contribution. iv_m3_yr_total and area_total_ha are additive; iv_m3_ha_yr is the resulting ratio. Sum-consistency: each combined row's totals equal the corresponding matches_only row plus the second_only row.

inv_period The same inv_period list element that is already carried by inv_increment_repsurv_ccirc and inv_inc_fill_gaps_gnfi3, with the two survey years and the period length.

method The increment method used in the upstream inv_increment_repsurv_ccirc call (one of "rep_classic", "rep_mean", "rep_end", "rep_trans"); carried through so user-facing renderers can label the table accordingly.

CI95 in $combined: the ci95_* columns are kept for structural compatibility but are NA throughout. The second_only contribution carries no empirical CI by contract (see inv_inc_big_overview_2nd_only); an area-weighted pooling of variances with one variance unknown would require assumptions that we are not willing to make here. Users who need a CI on the combined value can fall back to the matches_only CI as a lower bound, and document the choice.

iv_by_groups **in** $combined is deliberately omitted. The matched side breaks down by id_able_tree × tree_status (categories that the second_only side does not share), so a mixed breakdown would have a heterogeneous schema. Users that need that detail should look at $matches_only$overall$iv_by_groups separately.

Value

A list with the three top-level slots described above.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(),

inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_su
inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(),
match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(),
match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc()
output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(),
output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf
output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Examples

```
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

suppressWarnings(suppressMessages({
  inv_a_trees <- data_ex3_previous_sample_trees_essentials
  inv_b_trees <- data_ex3_sample_trees_essentials

  # Matching and gap filling are shipped ready-made for this pair
  inc_rep <- data_ex3_increment_matched
  inc_fl  <- data_ex3_increment_fills
  inc_sum <- inv_inc_summaric(
    inc_rep$tree_increments, inc_fl$fills_forward,
    inv_a_trees, inv_b_trees
  )
}))

bo <- inv_inc_big_overview(
  inc_fl, inv_a_trees, inv_b_trees, inv_b,
  style = "generic", inc_summaric = inc_sum
)
bo$combined$overall$iv_total
```

---

inv_inc_big_overview_2nd_only

Inventory-Wide Increment Overview for Plots Only Present in the Second Inventory

---

Description

Counterpart to inv_inc_big_overview_matches_only for those plots that only appear in the second survey (typical case: sample extensions and newly established plots between two inventories). Their increment cannot be observed directly; it is estimated per tree by applying the third-German-NFI growth functions *backwards* over a short period (dt, default 5 years) and aggregating across plots with the same area-weighted machinery as in inv_inc_big_overview_matches_only.

Usage

```
inv_inc_big_overview_2nd_only(inc_repinv_fl, inv_2nd_trees, inv_2nd, dt = -5)
```

Arguments

inc_repinv_fl	Output of inv_inc_fill_gaps_gnfi3. Used for the inv_period and plot_matches elements only; the per-tree increment data frames inside are not consumed here.
inv_2nd_trees	Tree data frame from the second inventory, as produced by pull_trees() |> height_complete_inventory() |> trees_add_essentials().
inv_2nd	The corresponding fe_inventory object.
dt	Estimation period for the backward gnfi3 projection, in years, passed through to tree_inc_gnfi_2012. Default -5 — see the function description for the rationale. The sign matters: negative values mean backward projection (the only direction that makes sense here), positive values would project the trees into the future and are flagged via a warning. Magnitudes other than 5 are accepted but should be motivated explicitly.

Details

Why a short period and not the full inter-survey distance: on plots without a previous observation we do not know the mortality and removal history. A backward projection over ~10 years would attribute much of the standing volume to "still living" trees and miss the unknown fraction that would actually have died or been harvested in that span. Over a shorter window (~5 years) that bias stays small and the resulting per-year growth rate is a defensible estimate for the representative area of the second-only plots.

The output structure mirrors inv_inc_big_overview_matches_only so that the two can later be combined into one whole-enterprise table by the caller. There is, however, no fill_option dimension here: only one estimator is in play (backward gnfi3 over dt years). A single label "gnfi3_backward" is used in the fill_option column so that a row bind with the matched-plots output stays unambiguous.

Current scope and limitations (v1):

- Only living trees of the second inventory enter the estimate (filter !removal). Removal trees on second-only plots are ignored; their contribution to the period increment would need a separate mortality model and is left out for now.
- The estimate is therefore a "living-tree backward" growth quantity, not a full balance increment. Be careful when combining the per-ha results with the matched-plot inc_summaric reference value, which does include mortality.
- **The ci95_* columns are present for structural compatibility with inv_inc_big_overview_matches_only but contain NA throughout.** All per-plot increments here come from a model (backward gnfi3 over a short window), not from empirical re-measurement; the only source of plot-to-plot scatter is then what the model itself produces on top of the actual tree sample. A confidence interval computed from that scatter would suggest a precision the underlying numbers do not have. Downstream code that merges this output with the matched-plots overview must therefore treat the 2nd-only CI columns as missing-on-purpose.

Value

A list with the same top-level structure as inv_inc_big_overview_matches_only: $overall (containing iv_by_groups, iv_total, v_total, n_plots) and $by_species_group (containing

iv_by_groups, iv_total, v_total). All m3/ha values relate to the total area across all second-only plots, even when grouped by species; plots without trees of a given species_group contribute their area but a zero numerator, so sums of per-species values match the corresponding overall values.

If there are no second-only plots, an empty-shaped list is returned with zero rows in the tables and area_total_ha = 0 in n_plots; this lets downstream combination code merge the two overviews without special-casing.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_su inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc() output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Examples

```
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

suppressWarnings(suppressMessages({
  # Matching and gap filling are shipped ready-made for this pair
  inc_rep <- data_ex3_increment_matched
  inc_fl  <- data_ex3_increment_fills
}))

bo_2nd <- inv_inc_big_overview_2nd_only(inc_fl, inv_b_trees, inv_b)
bo_2nd$overall$iv_total
bo_2nd$by_species_group$iv_total
```

---

inv_inc_big_overview_matches_only

Inventory-Wide Increment Overview for the Matched Plots of a Repeated Survey

---

Description

Produces the area-weighted increment overview from those plots that appear in *both* inventories — i.e. the subset for which repeated tree-level measurements are available. The companion function `inv_inc_big_overview_2nd_only` covers the plots that only exist in the second survey (new establishments, sample extensions); the top-level `inv_inc_big_overview_matches_only` combines both.

Usage

```
inv_inc_big_overview_matches_only(
  tree_inc_rep_gapfill,
  inv_1st_trees,
  inv_2nd_trees,
  inv_2nd,
  style = c("generic", "baysf"),
  inc_summaric
)
```

Arguments

`tree_inc_rep_gapfill`
: Output object of `inv_inc_fill_gaps_gnfi3`

`inv_1st_trees`
: Tree data frame as pulled and extended from the the `fe_inventory` object that represents the first of two subsequent surveys. Should be the same as previously used as input for `inv_increment_repsurv_ccirc`.

`inv_2nd_trees`
: Tree data frame as pulled and extended from the the `fe_inventory` object that represents the second of two subsequent surveys. Should be the same as previously used as input for `inv_increment_repsurv_ccirc`.

`inv_2nd`
: `fe_inventory` object that represents the second of the two inventories of interest.

`style`
: Two options `"generic"` and `"baysf"`. The former groups the most important part of the output by the tree cohorts that are internally distinguished when calculating/estimating increments. The latter applies a similar grouping as common in the Bavarian State Forest. The two styles differ only in the `tree_status` categories used in the `iv_by_groups` breakdown:

  `"baysf"` `"living"`, `"missing"`, `"rem_mort"`, `"no_coord"` (table-63-like, kept stable).

  `"generic"` `"regular"`, `"ingrown"`, `"missing"`, `"rem_mort_regular"`, `"rem_mort_ingrown"`, `"no_track"`, plus `"pseudo_ingrown"` for trees that `reclassify_pseudo_ingrowth_ccirc` reclassified from ingrowth to re-measured (inner-circle Inv-1 cohort). They carry a regular increment but are split out here so the pseudo-ingrowth contribution is visible in the aggregate. Only the `"generic"` style breaks this cohort out; under `"baysf"` they remain part of `"living"`.

`inc_summaric`
: Output of `inv_inc_summaric`, i.e. a list with the elements `$overall` (per-plot summaric increment) and `$by_species_group` (per plot × species group). Its area-weighted mean is appended as an additional row `"inc_summaric"` in the `iv_total` elements of both `$overall` and `$by_species_group` (one row per

species group there). The summaric increment is the most reliable overall increment estimate available from a repeated survey, as it relies purely on volume balances. The tree-level methods exist because they enable breakdowns by layers, diameter class, etc., which a pure balance can also provide once it is partitioned (here: by species group).

Value

A list with three top-level elements:

overall A sub-list with

- **iv_by_groups** Increments aggregated by fill_option, tree identification status, and tree status category
- **iv_total** Total increments per fill_option, plus an additional row `"inc_summaric"` with the summaric (yield-history) increment as the most reliable reference value
- **v_total** Total standing volume at the second inventory
- **n_plots** Number of plots and total represented area

by_species_group A sub-list with the same structure as `overall` (minus `n_plots`, which would be redundant), but every tibble is additionally grouped by `species_group`. The total area (`area_total_ha`) stays at the area across all plots, so that plots without trees of a given species group still contribute their area to the denominator. Consequently, sums of per-species values (`iv_m3_yr_total`, `iv_m3_ha_yr`, `v_hub_m3_ha`) across species groups match the corresponding overall values, and the same applies row by row to the appended `"inc_summaric"` entries.

method The increment method used in the upstream `inv_increment_repsurv_ccirc` call (carried through `inv_inc_fill_gaps_gnfi3`).

See Also

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_su`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center_coord()`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`, `output_increment_overall_pdf()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pdf`, `output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Examples

```
# Identify two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# Matching and gap filling for this pair are shipped ready-made, see
```

```
# ?data_ex3_increment_interim for how they were built
inc_repinv    <- data_ex3_increment_matched
inc_repinv_fl <- data_ex3_increment_fills

# Compute the summaric increment per plot (required for big_overview).
# Note: summaric reads matched_trees and fills_forward directly — it
# does NOT go through the consolidate / extend per-tree pipeline,
# so its balance always sees the full measured population.
inc_sum <- inv_inc_summaric(
  inc_repinv$tree_increments, inc_repinv_fl$fills_forward,
  inv_a_trees, inv_b_trees
)

# Finally generate the matched-plots overview, use both styles
inv_inc_big_overview_matches_only(
  inc_repinv_fl, inv_a_trees, inv_b_trees, inv_b,
  style = "generic", inc_summaric = inc_sum
)
inv_inc_big_overview_matches_only(
  inc_repinv_fl, inv_a_trees, inv_b_trees, inv_b,
  style = "baysf", inc_summaric = inc_sum
)
```

`inv_inc_fill_gaps_gnfi3` *Close Tree Increment Gaps with Estimates from National Forest Inventory Functions*

Description

When calculating tree level increments from two subsequent inventories, there are typically trees that were present in one of the inventories only. For different types of such trees, we apply here the single tree growth functions from the third German National Forest Inventory (Riedel et al. 2017) as implemented in the Package `ForestElementsR`, namely the functions `d_age_gnfi3` and `h_age_gnfi3`.

Usage

```
inv_inc_fill_gaps_gnfi3(
  tree_inc_rep,
  inv_1st_trees,
  inv_2nd_trees,
  include_reptrees = TRUE
)
```

Arguments

tree_inc_rep	Output of inv_increment_repsurv_ccirc
inv_1st_trees	Extended tree data frame for the first inventory, exactly as it was used as an input to inv_increment_repsurv_ccirc in order to generate tree_inc_rep.
inv_2nd_trees	Extended tree data frame for the second inventory, exactly as it was used as an input to inv_increment_repsurv_ccirc in order to generate tree_inc_rep.
include_reptrees	If TRUE (default), backward increment estimates are also made for trees that are present in both inventories and therefore have "real" increments. If FALSE the columns for backward estimates will be set to NA. This might save computation time when dealing with extensive amounts of data.

Details

The increment estimates performed here are only applied to inventory plots that match in both inventories. This is guaranteed by requiring tree_inc_rep as input object. This is the output of inv_increment_repsurv_ccirc which ensures plot matching.

Value

A list comprising five data frames that have exactly the same structure, i.e. the same columns as tree_inc_rep$tree_increments, however with two additional columns,iv_hub_m3_per_gnfi3, and iv_hub_m3_yr_gnfi3). The data frame fills_forward relates to trees that were present in the first, but missing in the second inventory. Their increment is projected forward up to the mid of the period between both inventories. The second data frame fills_backward comprises trees that were present in the second, but not present in the first inventory. Their increment is estimated backwards for the whole period. The third data frame fills_backward_nomatch comprises trees present in the second inventory, which were, however, not recorded in a way that makes them identifyable in the first inventory (typically, such trees are under a certain dbh threshold). For such trees backward increment estimates are made for the whole period. The fourth data frame fills_backward_implausible comprises trees that were present in both inventories but flagged as implausible during tree matching (due to excessive volume shrinkage or species mismatch). These trees received a zero increment from the repeated measurement; their increment is estimated backwards from the second inventory's values for the whole period. The fifth data frame fills_backward_reptrees relates to plausible trees that were present in both inventories. While directly calculated increments are available for these trees, backward estimates are calculated if (include_reptrees) is TRUE. This might only be required under special circumstances, therefore, the default setting is FALSE for saving time. In the latter case, the columns reserved for the increment estimates are set to NA. In addition the list contains the elements inv_period (carried through from inv_increment_repsurv_ccirc), plot_matches (the plot match overview, also from inv_increment_repsurv_ccirc; downstream consumers such as inv_inc_big_overview_2nd_only use the in_b_only entry there to identify plots that only appear in the second inventory), and method (the increment method used in the upstream inv_increment_repsurv_ccirc call – one of "rep_classic", "rep_mean", "rep_end", "rep_trans").

References

Riedel T, Hennig P, Kroiher F, Polley H, Schmitz F, F. S (2017). *Die dritte Bundeswaldinventur (BWI 2012). Inventur- und Auswertungsmethoden*. Thuenen Institut fuer Waldoekosysteme.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidat inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_su inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc() output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Examples

```
# Identify two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# The matching this builds on is shipped ready-made for the pair, see
# ?data_ex3_increment_interim
inc_repinv <- data_ex3_increment_matched

# Finally, fill the gaps. include_reptrees = FALSE estimates only where
# there is a gap; the default TRUE also estimates for trees measured
# twice, which is what fill_option = "all_estimates" later feeds on.
inv_inc_fill_gaps_gnfi3(
  inc_repinv, inv_a_trees, inv_b_trees, include_reptrees = TRUE
)
```

`inv_inc_summaric` *Compute the Summaric Increment per Plot, Overall and by Species Group*

Description

Computes a pure balance increment per plot that does not require any tree matching assumptions. The formula is: `inc_summaric = (V_end - V_start + V_removed) / period`, where `V_end` is the total standing volume at the second inventory (excluding removed trees), `V_start` is the total standing volume at the first inventory, and `V_removed` is the forward-estimated volume of trees that were

removed between inventories. The forward estimation relies on gnfi3-based estimates from the "standard" fill option, which is why the input must come from inv_inc_tree_consolidate with fill_option = "standard", extended by inv_inc_tree_extend.

Usage

```
inv_inc_summaric(matched_trees, fills_forward, inv_1st_trees, inv_2nd_trees)
```

Arguments

matched_trees	Data frame, the $tree_increments element of inv_increment_repsurv_ccirc's output. Carries all rows needed by the summaric balance — including the unmatchable inner- circle cohort (is_unmatchable_1st = TRUE / NA on the counterpart side of tree_exists_*).
fills_forward	Data frame, the $fills_forward element of inv_inc_fill_gaps_gnfi3's output. Provides the half-period gnfi3 forward increment estimate (iv_hub_m3_ha_per_gnfi3) for the v_aus term. summaric joins the relevant value back to matched_trees by (plot_id_1st, tree_id_1st).
inv_1st_trees	Tree data frame as pulled and extended from the first fe_inventory object. Used for the species-group lookup of trees that exist only in the first inventory (removed trees).
inv_2nd_trees	Tree data frame for the second inventory. Used for the species-group lookup of trees that exist in the second inventory (the dominant case — both-exist trees, ingrowth, and unmatchable inv-2).

Details

The classical German forestry term for this quantity is "ertragsgeschichtlicher Zuwachs" (literally: yield-history increment). There is no established English equivalent; we use the term "summaric increment" here to express that this increment is derived purely from summary-level volume balances between two inventories, without relying on individual tree identity across surveys.

Methodological role. The ertragsgeschichtliche increment is the **Goldstandard** for the Betriebsebenen-Zuwachs — the best estimate of the enterprise-wide annual increment that two inventories can yield. Unlike the four single-tree methods (rep_classic, rep_mean, rep_end, rep_trans) it is a pure mass balance over the measured tree population; it does not depend on assumptions about which inventory's representation number to use for trees that change circles, and it does not need a per-tree pairing across inventories. For that reason it stands architecturally apart from the per-tree pipeline (inv_inc_tree_consolidate and inv_inc_tree_extend): summaric pulls its inputs (matched_trees, fills_forward, and the inv-trees lookups for species groups) directly, so its balance always sees the full measured population including the unmatchable inner-circle cohort — independent of any fill_option choice on the per-tree side.

Computational structure on each plot:

- v_hub_m3_ha_1st: sum of n_rep_ha_1st * v_hub_m3_1st over all inv-1 measurements present in matched_trees, including the rows flagged is_unmatchable_1st = TRUE (e.g. inner-circle trees recorded without polar coordinates).

- v_hub_m3_ha_2nd: sum of n_rep_ha_2nd * v_hub_m3_2nd over all living inv-2 measurements (no removal flag), including unmatchable inv-2 rows and the inv-2 records of pseudo-ingrowth- reclassified trees (the latter contribute only their inv-2 volume after the local undo described below).
- v_hub_m3_ha_aus: sum of n_rep_ha_1st * v_hub_m3_1st plus the half-period gnfi3 forward estimate, over the verified-ausscheider subset (tree_exists_1st is TRUE and either tree_exists_2nd is FALSE or the tree carries a removal flag at inv 2). The forward estimate is pulled from fills_forward. Unmatchable rows have NA on the counterpart side and are filtered out by NA-semantics so they never inflate v_aus.

Pseudo-ingrowth-reclassified trees have their reclassification *locally undone* inside summaric so that their inv-1 reconstruction (gnfi3 backward) does not enter the balance. They contribute only via their measured inv-2 volume — exactly the methodological treatment we want for the unmatchable inner-circle cohort.

Value

A list with two tibbles:

overall One row per plot, columns plot_id, v_hub_m3_ha_1st, v_hub_m3_ha_2nd, v_hub_m3_ha_aus, inc_summaric_m3_ha_yr.

by_species_group One row per (plot, species_group) combination, same volume / increment columns as overall plus species_group. Combinations where a species group is absent on a plot are not represented; consumers that need a complete grid (e.g. for area-weighted aggregation across all plots) must fill in zeros.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr tree_inc_repsurv()

Examples

```
# Identify two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# Pull and prepare tree data
inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials
```

```
# Matching and gap filling for this pair are shipped ready-made, see
# ?data_ex3_increment_interim for how they were built
inc_repinv    <- data_ex3_increment_matched
inc_repinv_fl <- data_ex3_increment_fills

# Summaric goes directly off matched_trees + fills_forward —
# no consolidate / extend in between.
inv_inc_summaric(
  inc_repinv$tree_increments,
  inc_repinv_fl$fills_forward,
  inv_a_trees, inv_b_trees
)
```

`inv_inc_tree_2_plot` *Aggregate Single Tree Increments to ha-Values per Species Group, Age- or Diameter Class and Layer on Plot Level*

Description

The ha-related values calculated here are to be understood as the contributions of each tree cohort (as defined by species group, (age- or diameter-) class, and layer) to the whole ha-wise increment of the plot. Importantly, they do not relate to virtual area shares of the cohorts. In an increment evaluation workflow, the intended use of this function is to be applied to the output of `inv_inc_tree_extend`

Usage

```
inv_inc_tree_2_plot(tree_inc_extd, by_class = c("age", "dq"))
```

Arguments

`tree_inc_extd`	Data frame, output of `inv_inc_tree_extend`
`by_class`	Two options "age" (default), and "dq", which will aggregate the increments by age-, and mean diameter classes, respectively.

Value

A data frame with increment sums grouped by `plot_id`, `species_group`, `d_q_class` or `age_class`, `layer_key`, `removal`

See Also

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_survey()`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_cente`

match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr, tree_inc_repsurv()

Examples

```
# Identify two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# Matching and gap filling for this pair are shipped ready-made, see
# ?data_ex3_increment_interim for how they were built
inc_repinv    <- data_ex3_increment_matched
inc_repinv_fl <- data_ex3_increment_fills

# Consolidate and extend tree increments
inc_repinv_ce <- inv_inc_tree_consolidate(inc_repinv_fl) |>
  inv_inc_tree_extend(inv_a_trees, inv_b_trees, inv_b)

# Finally, aggregate on plot level
inv_inc_tree_2_plot(inc_repinv_ce, "age")
inv_inc_tree_2_plot(inc_repinv_ce, "dq")
```

---

inv_inc_tree_consolidate

Consolidating a Mix of Real and Estimated Tree-Level Increments for Further Processing

---

Description

In an increment evaluation workflow, the intended use of this function is to be applied to the output of inv_inc_fill_gaps_gnfi3. In parallel to the increment values that were directly calculated, the latter has estimated the increment with the tree growth functions from the Third German National Forest Inventory (Riedel et al. 2017) for different cohorts of trees (in different ways, see inv_inc_fill_gaps_gnfi3). Here, the user decides which increments to use for which cohort in subsequent evaluations.

Usage

```
inv_inc_tree_consolidate(
  tree_inc_rep_gapfill,
  fill_option = c("standard", "min_estimates", "all_estimates")
)
```

Arguments

tree_inc_rep_gapfill	Output object of inv_inc_fill_gaps_gnfi3
fill_option	Three options, "standard" (default), "min_estimates", and "all_estimates". See Details.

Details

The fill_option "standard" takes the inventory increment for all trees that can be potentially identified across inventories and are present at least in the second survey (i.e. re-measured or ingrown), as long as they do not have a removal flag at the second survey. For such trees that are, however, present in the first survey but are absent or have a removal flag at the second survey, an NFI based forward increment estimate for half the period between the surveys is used. Trees that cannot be identified across inventories and are present in the second survey, obtain a backwards increment estimate for the whole period. The option "min_estimates" uses NFI estimates only for trees that cannot be identified across surveys. Therefore, trees that only exist in the first survey obtain a zero increment. The third option "all_estimates" uses NFI based estimates for all trees.

Value

A data frame with every line representing a tree, actually all output components from inv_inc_fill_gaps_gnfi3 combined into one data frame. Importantly, there are three additional columns: iv_hub_m3_per and iv_hub_m3_yr are the periodic and annual increment values to continue calculating with (coming either from the inventory or from the estimate), and iv_filled is logical; if TRUE, the increment is estimated with the German NFI growth functions. If FALSE the increment is directly calculated from the survey data.

References

Riedel T, Hennig P, Kroiher F, Polley H, Schmitz F, F. S (2017). *Die dritte Bundeswaldinventur (BWI 2012). Inventur- und Auswertungsmethoden*. Thuenen Institut fuer Waldoekosysteme.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr tree_inc_repsurv()

Examples

```
# Identify two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# Matching and gap filling for this pair are shipped ready-made, see
# ?data_ex3_increment_interim for how they were built
inc_repinv    <- data_ex3_increment_matched
inc_repinv_fl <- data_ex3_increment_fills

# Finally, consolidate the tree increments, try all fill options
inv_inc_tree_consolidate(inc_repinv_fl, fill_option = "standard")

inv_inc_tree_consolidate(inc_repinv_fl, fill_option = "min_estimates")

inv_inc_tree_consolidate(inc_repinv_fl, fill_option = "all_estimates")

# The option 'all_estimates' must throw an error if the gap filling was
# not done for the trees which have dbh values in both surveys
inc_repinv_fl <- inv_inc_fill_gaps_gnfi3(
  inc_repinv, inv_a_trees, inv_b_trees, include_reptrees = FALSE
)
try(
  inv_inc_tree_consolidate(inc_repinv_fl, fill_option = "all_estimates")
)
```

`inv_inc_tree_extend` *Extend Tree-Level Increments With Detailed Tree Information, Especially Age and Mean Diameter Classes for Further Aggregation*

Description

In an increment evaluation workflow, the intended use of this function is to be applied to the output of `inv_inc_tree_consolidate`

Usage

```
inv_inc_tree_extend(
  tree_inc_cons,
  inv_1st_trees,
  inv_2nd_trees,
  inv_2nd,
  d_q_interval = 10
)
```

Arguments

tree_inc_cons	A data frame with consolidated tree level increments, typically the output of inv_inc_tree_consolidate
inv_1st_trees	Tree data frame as pulled and extended from the the fe_inventory object that represents the first of two subsequent surveys. Should be the same as previously used as input for inv_increment_repsurv_ccirc.
inv_2nd_trees	Tree data frame as pulled and extended from the the fe_inventory object that represents the second of two subsequent surveys. Should be the same as previously used as input for inv_increment_repsurv_ccirc.
inv_2nd	fe_inventory object that represents the second of the two inventories of interest.
d_q_interval	Numeric value, represents the width (cm) of the quadratic mean diameter (d_q) classes to be added to the output data frame. This argument is passed to the function back_table_dclass which is internally called.

Value

A data frame where each row represents a tree, with extended information about increment and other tree level variables. The two columns time_yr_prev and time_yr_2nd are uniform across all rows and describe the survey years of the inventory pair (1st and 2nd survey, respectively); their difference is the inter-survey period. time_yr is the time of the tree's actual observation in the join partner inventory and equals time_yr_prev for trees that exist only in the first survey. The result also carries an "inc_meta" attribute (survey years, period, area, point count, Berechnungs-/Ergänzungsmethode) that the increment base tables promote into their header meta block.

Coverage – read this before aggregating

The result covers **only the plots measured in both inventories**. Plots that exist just in the second survey are not part of the repeated-survey matching and therefore absent here. Feeding this frame straight into increment_base_table or increment_base_table_main_stand *understates* every per-hectare figure, because those tables divide by the full operation area (from the "inc_meta" attribute) while the second-only plots contribute no increment – their area sits in the denominator with nothing in the numerator. Pass the result through inv_inc_tree_extend_combined first, or, simpler, use inv_increment_repeated_survey, which runs the whole chain including that step.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr tree_inc_repsurv()

Examples

```
# Identify two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# Matching and gap filling for this pair are shipped ready-made, see
# ?data_ex3_increment_interim for how they were built
inc_repinv    <- data_ex3_increment_matched
inc_repinv_fl <- data_ex3_increment_fills

# Consolidate tree increments
inc_repinv_cons <- inv_inc_tree_consolidate(inc_repinv_fl)

# Extend with tree information required for further aggregation
inv_inc_tree_extend(inc_repinv_cons, inv_a_trees, inv_b_trees, inv_b)
```

---

inv_inc_tree_extend_combined

Fold Second-Only Plots Into the Repeated-Survey Tree-Level Increments

---

Description

Extends the per-tree increment frame of a repeated survey – the output of `inv_inc_tree_extend`, which covers only the plots measured in *both* inventories – with the plots that exist only in the second survey (new establishments, sample extensions). These second-only plots receive a short-window backward gnfi3 estimate via `inv_inc_tree_extend_single` (default `dt = -5`, the same convention as `inv_inc_big_overview_2nd_only`), and the two per-tree frames are row-bound into one.

Usage

```
inv_inc_tree_extend_combined(
  tree_inc_extd,
  inc_repinv_fl,
  inv_2nd_trees,
  dt = -5,
  d_q_interval = 10
)
```

Arguments

`tree_inc_extd`	Per-tree increment frame of the matched plots, the output of `inv_inc_tree_extend`. Its `"inc_meta"` attribute is carried over to the result.

inc_repinv_fl	Output of inv_inc_fill_gaps_gnfi3. Only its plot_matches$in_b_only element is used, to identify the second-only plots.
inv_2nd_trees	Tree data frame of the second survey (as pulled, height-completed and passed through trees_add_essentials); the same object supplied to inv_inc_tree_extend as its inv_2nd_trees argument.
dt	Time span (years) for the second-only backward gnfi3 estimate, passed to inv_inc_tree_extend_single Negative for a backward projection. Default -5, matching inv_inc_big_overview_2nd_only.
d_q_interval	Numeric, width (cm) of the quadratic mean diameter (d_q) classes, passed on to inv_inc_tree_extend_single.

Details

The point of this function is the shared per-tree contract: both inv_inc_tree_extend and inv_inc_tree_extend_single emit the same columns consumed by inv_inc_tree_2_plot, so the combined frame runs through increment_base_table and increment_base_table_main_stand *unchanged*. The resulting base tables then cover the *whole enterprise* (matched plus second-only plots) instead of only the matched subset, so their per-hectare increment relates to the full operation area – consistent with the enterprise-level inv_inc_big_overview (where combined = matches_only + second_only).

Only living trees (!removal) of the second-only plots enter the estimate, mirroring inv_inc_big_overview_2nd_only: a plot without a predecessor survey has no between-inventory removals to capture, so a removal flag there marks a tree that is not part of the period's standing increment. The matched frame is left exactly as it is (it keeps its removed trees, whose increment is part of the period's production – the "Weg B" convention).

The header meta carried by tree_inc_extd already describes the whole enterprise (its area and point count come from the full second inventory passed to inv_inc_tree_extend), so it is simply re-attached to the combined frame (row-binding drops attributes).

Value

A per-tree increment data frame with the same contract as inv_inc_tree_extend, now spanning matched and second-only plots, carrying the (whole-enterprise) "inc_meta" attribute of tree_inc_extd. If there are no second-only plots (or none with living trees), tree_inc_extd is returned unchanged.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr, tree_inc_repsurv()

Examples

```
# Identify two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

suppressWarnings(suppressMessages({
  # Matching and gap filling are shipped ready-made for this pair
  inc_rep <- data_ex3_increment_matched
  inc_fl  <- data_ex3_increment_fills
  inc_extd <- inv_inc_tree_consolidate(inc_fl) |>
    inv_inc_tree_extend(inv_a_trees, inv_b_trees, inv_b)
}))

# Fold in the second-only plots -> whole-enterprise per-tree frame
inc_extd_all <- inv_inc_tree_extend_combined(
  inc_extd, inc_fl, inv_b_trees
)

# Base tables now cover matched + second-only plots (run unchanged)
increment_base_table(inc_extd_all, by_class = "age")$all_total
```

---

inv_period *Survey Years and Period Between Two Inventories*

Description

Convenience helper that extracts the survey years of two `fe_inventory` objects and returns them together with the resulting inter-survey period, after the same validation that `inv_increment_repsurv_ccirc` applies. Used by functions that combine two inventories so that downstream consumers (e.g. `inv_inc_summaric`) can access the period without having to carry the `fe_inventory` objects further.

Usage

```
inv_period(inv_1st, inv_2nd)
```

Arguments

`inv_1st`, `inv_2nd`
: Two `fe_inventory` objects in chronological order.

Value

A list with elements `time_yr_1st`, `time_yr_2nd`, and `period_yr` (the difference).

Examples

```
inv_period(
  data_ex3_previous_sample_fe_inventory,
  data_ex3_sample_fe_inventory
)
```

`is_fe_increment_gnfi3` *Test for the Class fe_increment_gnfi3*

Description

Test for the Class fe_increment_gnfi3

Usage

```
is_fe_increment_gnfi3(x)
```

Arguments

`x`	An object

Value

`TRUE` if `x` is an `fe_increment_gnfi3` object, `FALSE` otherwise

Examples

```
inc <- inv_increment_gnfi3(
  data_ex3_sample_fe_inventory, data_ex3_sample_trees_essentials
)
is_fe_increment_gnfi3(inc)
```

`is_fe_increment_repsurv`
Is an Object an fe_increment_repsurv?

Description

Is an Object an fe_increment_repsurv?

Usage

```
is_fe_increment_repsurv(x)
```

Arguments

x	Object to test.

Value

Logical.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr tree_inc_repsurv()

Examples

```
is_fe_increment_repsurv(1L)
```

---

is_fe_increment_ytables

Test for the Class fe_increment_ytables

Description

Test for the Class fe_increment_ytables

Usage

```
is_fe_increment_ytables(x)
```

Arguments

x	An object

Value

TRUE if x is an fe_increment_ytables object, FALSE otherwise

Examples

```
inc <- inv_increment_ytables(
  data_ex3_sample_fe_inventory, data_ex3_sample_trees_essentials,
  ytables_bavrn_state_var_1_feneu
)
is_fe_increment_ytables(inc)
```

`is_fe_inventory` *Check if an Object is an* **fe_inventory**

Description

Check if an Object is an **fe_inventory**

Usage

```
is_fe_inventory(x)
```

Arguments

x	an object

Value

`TRUE` if the object inherits from the `fe_stand` class

Examples

```
is_fe_inventory(data_ex4_sample_fe_inventory)
```

`match_2_inventories` *Match two fe_inventory Objects on Plot Level*

Description

Wraps the functions `match_2_inventories_by_center_coord`, and `match_2_inventories_by_plot_id` and decides which one to call.

Usage

```
match_2_inventories(
  inv_a,
  inv_b,
  match_type = c("plot_id", "center_coord"),
  plot_id_style = c("baysf", "generic"),
  coord_tolerance = 10
)
```

Arguments

inv_a	An object of class fe_inventory
inv_b	An object of class fe_inventory
match_type	Character string indicating the method for matching inventory plots from the two subsequent inventories. The option "center_coord" matches points whose center coordinates have a distance of no more than coord_tolerance, while the option "plot_id" (default) matches plots with the same id.
plot_id_style	Character string indicating the plot_id style to be used. Only relevant when match_type == "plot_id". The available options are "baysf" (default) and "generic". For a match, the latter ("generic") requires the full plot_ids of two plots to be equal in both inventories. The former ("baysf") only requires the last group of digits in the plot_id for a match due to the specific BaySF data format.
coord_tolerance	Maximum distance allowed for a match of two inventory plots in the distance unit of the coordinate reference system of inv_a and inv_b; in virtually all relevant cases it is m. Default is 10, i.e. inventory plots whose centers have a distance of up to 10 meters will be accepted as matches.

Value

A list with five elements: i) matches – the matching inventory plots, ii) in_a_only – plots that exist in inv_a only, iii) in_b_only – plots that exist in inv_b only, iv) multiple_matches – a logical vector identifying multiple matches (can only occur with match_type = "center_coord" and is undesired), and v) circle_def_mismatch – plot pairs that matched by id or coordinates but were excluded because their circle definitions (dbh_lower and/or c_area) differ between the two inventories. Such pairs are unsuitable for tree-level increment calculations from repeated surveys. The exact types of the objects (tibbles or sf) in this list depend on the match_type the function has been called with, but they always contain a column with the relevant plot ids.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingr tree_inc_repsurv()

Other repeated inventory: harmonize_inv_for_repsurv(), inv_increment_repsurv_ccirc(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), reclassify_pseudo_ingr

Other inventory matches: harmonize_inv_for_repsurv(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc()

Examples

```
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# both options for match_type work for this example
match_2_inventories(inv_a, inv_b, match_type = "plot_id", "baysf")
match_2_inventories(inv_a, inv_b, match_type = "center_coord", "baysf")
```

---

match_2_inventories_by_center_coord

Match Two fe_inventory Objects by the Center Coordinates of the Inventory Plots

---

Description

From two fe_inventory objects identify the matching and non- matching inventory plots based on their center coordinates. A warning is issued if the matching coordinates indicate spatial one-to-many or many-to-many relationships.

Usage

```
match_2_inventories_by_center_coord(inv_a, inv_b, tolerance = 0)
```

Arguments

inv_a	An object of class fe_inventory
inv_b	An object of class fe_inventory
tolerance	Maximum distance allowed for a match of two inventory plots in the distance unit of the coordinate reference system of inv_a and inv_b; in virtually all relevant cases it is m. Default is 0, i.e. exact matches are required.

Value

A list containing three sf objects, matches, in_a_only, and in_b_only, and one logical vector multiple_matches. The three sf objects indicate the inventory plots that are present in both, inv_a and inv_b, only in inv_a but not in inv_b, and only in inv_b but not in inv_a, respectively. The sf object matches has the columns plot_id.inv_a, plot_id.inv_b, and geometry (i.e. the point coordinates). Thereby, The column plot_id_inv_a contains the id of a plot it has in inv_a, and analogously with plot_id_inv_b. Note that spatially matching points do not necessarily need to have the same plot_id in both inventories. The sf object in_a_only has only two columns, plot_id, and geometry. The same applies to the sf object in_b_only. the plot_id entries for the

other inventories are NA. The vector multiple_matches relates to the matches object, and indicates entries with the same center coordinates. This indicates one-to-many or many-to-many relations between the two input inventories which is usually a problem. Therefore, multiple_matches should only contain FALSE entries.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_plot_ match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Other repeated inventory: harmonize_inv_for_repsurv(), inv_increment_repsurv_ccirc(), match_2_inventories(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), reclassify_pseudo_ingr

Other inventory matches: harmonize_inv_for_repsurv(), match_2_inventories(), match_2_inventories_by_plot_i match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(

Examples

```
# Construct a dummy example with partly non-matching plots from one
# fe_inventory object
inv_a <- data_ex3_sample_fe_inventory[3:10, ]
inv_b <- data_ex3_sample_fe_inventory[1:7,  ]

# As both dummy inventories were generated from the same set, we do not
# require any tolerance > 0
match_2_inventories_by_center_coord(inv_a, inv_b, tolerance = 0)


# Now a real example from two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# Requires a tolerance of 10.0 m in order to obtain the correct match
# (as indicated by the plot ids)
match_2_inventories_by_center_coord(inv_a, inv_b, tolerance = 10.0)

# If tolerance is set too high (here unrealistic 3000 m for demo purposes),
# we obtain duplicate matches and a warning message
match_2_inventories_by_center_coord(inv_a, inv_b, tolerance = 3000)
```

`match_2_inventories_by_plot_id` *Match Two fe_inventory Objects by the IDs of the Inventory Plots*

Description

From two `fe_inventory` objects identify the matching and non- matching inventory plots based on their center coordinates.

Usage

```
match_2_inventories_by_plot_id(inv_a, inv_b, style = c("generic", "baysf"))
```

Arguments

`inv_a`	An object of class `fe_inventory`
`inv_b`	An object of class `fe_inventory`
`style`	Character, two options "generic" and "baysf". The former requires that the column `plot_id` in both inventories, `inv_a`, and `inv_b` is the same for the same inventory unit. The latter assumes a BaySF (Bavarian State Forest) style plot_id which is an underscore ("_") separated string, where only the third substring (actually the original BaySF field 'koord") has to match for the the same inventory unit in subsequent inventories.

Value

A list that comprises three data frames (tibbles): The first one, `matches`, has two columns, `plot_id_inv_a`, and `plot_id_inv_b` which indicate the ids of the matching `plot_ids` in both inventories. The two columns are required because there may be situations (BaySF style) where the ids of matching points are not entirely the same. The second data frame, `in_a_only`, contains the ids of those plots that exist in `inv_a`, but not in `inv_b`, while the third one, `in_b_only`, contains the ids of the plots that are present in `inv_b`, but not in `inv_a`.

See Also

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_survey()`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`, `output_increment_overall_pdf()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pdf()`, `output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Other repeated inventory: harmonize_inv_for_repsurv(), inv_increment_repsurv_ccirc(), match_2_inventories(), match_2_inventories_by_center_coord(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), reclassify_pseudo_ingr

Other inventory matches: harmonize_inv_for_repsurv(), match_2_inventories(), match_2_inventories_by_center match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(

Examples

```
# Construct a dummy example with partly non-matching plots from one
# fe_inventory object
inv_a <- data_ex3_sample_fe_inventory[3:10, ]
inv_b <- data_ex3_sample_fe_inventory[1:7,  ]

# These are BaySF style inventory data, which means that even matching plot
# ids from subsequent inventories are equal only with regard to their last
# group of digits. However, as this example was constructed from only one
# inventory, both style options, "baysf" and "generic" are equally ok.
match_2_inventories_by_plot_id(inv_a, inv_b, style = "baysf")
match_2_inventories_by_plot_id(inv_a, inv_b, style = "generic")

# Now a real example with two subsequent inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# These are BaySF style inventory data, from real subsequent inventories.
# Therefore, we must choose that style for matching
match_2_inventories_by_plot_id(inv_a, inv_b, style = "baysf")


# The generic inventory style requires that corresponding plots have the
# same plot_id in both inventories. We can achieve that with a slight
# manipulations of the BaySF style data used above (i.e. reducing the
# original plot_ids to the last group of digits):
inv_a <- inv_a |>
  dplyr::mutate(
    plot_id = purrr::map_chr(
      plot_id, .f = function(.x) strsplit(.x, "_")[[1]][3]
    )
  )

inv_b <- inv_b |>
  dplyr::mutate(
    plot_id = purrr::map_chr(
      plot_id, .f = function(.x) strsplit(.x, "_")[[1]][3]
    )
  )

# Now we achieve the same result as above with generic-style matching
match_2_inventories_by_plot_id(inv_a, inv_b, style = "generic")
```

match_plot_statistics *Generate Plot Match Overview Statistics from the Output of* match_2_inventories

Description

Informs about the numbers of plots that match and do not match in both inventories, and the corresponding representation areas in ha.

Usage

```
match_plot_statistics(plot_matches, inv_a, inv_b)
```

Arguments

plot_matches	list, output of match_2_inventories
inv_a	An object of class fe_inventory nust be the first inventory relating to plot_matches
inv_b	An object of class fe_inventory nust be the second inventory relating to plot_matches

Value

A list containing two named vectors, plot_numbers, and rep_areas_ha. These contain the statistics relating to plot numbers and represented areas, respectively. Both include a count/area for plots excluded due to incompatible circle definitions (n_circle_def_mismatch, ha_circle_def_mismatch_1st, ha_circle_def_mismatch_2nd).

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente match_2_inventories_by_plot_id(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsu output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Other repeated inventory: harmonize_inv_for_repsurv(), inv_increment_repsurv_ccirc(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), reclassify_pseudo_ingr

Other inventory matches: harmonize_inv_for_repsurv(), match_2_inventories(), match_2_inventories_by_center match_2_inventories_by_plot_id(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsu

Examples

```
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# both options for match_type work for this example
match_pid <- match_2_inventories(
  inv_a, inv_b, match_type = "plot_id", "baysf"
)
match_crd <- match_2_inventories(
  inv_a, inv_b, match_type = "center_coord", "baysf"
)

# stats should be the same in both cases
match_plot_statistics(match_pid, inv_a, inv_b)
match_plot_statistics(match_crd, inv_a, inv_b)
```

---

match_trees_on_inventory_repsurv_ccirc

Match Trees on Subsequent Inventories on Inventory Level

Description

Typically called from within inv_increment_repsurv_ccirc. The consistency of the input to this function is not checked here. In essence, the function wraps serial calls of match_trees_on_plot_repsurv_ccirc.

Usage

```
match_trees_on_inventory_repsurv_ccirc(
  inv_point_matches,
  inv_1st,
  inv_2nd,
  inv_1st_trees,
  inv_2nd_trees,
  tree_id_style = c("baysf", "generic"),
  inner_circle_non_matchable = FALSE,
  shrinkage_rate_per_decade = 0.1,
  species_check = c("species_group", "species", "warn_only"),
  progress_bar = TRUE
)
```

Arguments

inv_point_matches	Output of match_2_inventories (typical) or match_2_inventories_by_center_coord, or match_2_inventories_by_plot_id
inv_1st	fe_inventory object representing the earlier of the two inventories. All plots of this inventory must have the class fe_ccircle_spatial.

inv_2nd — fe_inventory object representing the later of the two inventories. All plots of this inventory must have the class fe_ccircle_spatial.

inv_1st_trees — A data frame resulting from applying the functions pull_trees, height_complete_inventory, and trees_add_essentials to inv_1st. While this could be done internally inside this function, this has typically happened earlier already in an inventory evaluation workflow. As this is a considerably time-consuming process, we deem the redundancy acceptable.

inv_2nd_trees — A data frame resulting from applying the functions pull_trees, height_complete_inventory, and trees_add_essentials to inv_2nd. While this could be done internally inside this function, this has typically happened earlier already in an inventory evaluation workflow. As this is a considerably time-consuming process, we deem the redundancy acceptable.

tree_id_style — Character string that indicates how trees are uniquely identified on inventory plot level. This is required for matching trees on the same plot in both inventories. The options are "baysf" (default) and "generic". In the BaySF case, trees are matched by their polar coordinates, while in the generic case, the tree_ids must mark the same trees in both inventories.

inner_circle_non_matchable — Logical, default FALSE. Passed through to match_trees_on_plot_repsurv_ccirc. See its documentation for the full meaning.

shrinkage_rate_per_decade — Numeric, the maximum plausible volume shrinkage rate per decade. Default is 0.10. See inv_increment_repsurv_ccirc for details.

species_check — Character string controlling species consistency checks. Options: "species_group" (default), "species", "warn_only". See inv_increment_repsurv_ccirc for details.

progress_bar — Boolean, if TRUE (default), a progress bar is shown during time critical steps of the execution.

Value

A data frame identifying all matching trees with their tree and plot ids in both inventories, the calendar years of both inventories, and columns with flags that indicate in which of both inventories a tree exists (can be only one of them or both). Additionally contains the columns volume_change_rate_per_decade (the observed volume change rate scaled to a decade, based on an exponential model; positive values indicate growth, negative values indicate shrinkage), volume_change_implausible (logical, TRUE if shrinkage exceeds the threshold), and species_mismatch (logical, TRUE if species identity changed between surveys according to the species_check setting).

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente

match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Other repeated inventory: harmonize_inv_for_repsurv(), inv_increment_repsurv_ccirc(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_plot_repsurv_ccirc(), reclassify_pseudo_ingrowth_ccirc()

Other inventory matches: harmonize_inv_for_repsurv(), match_2_inventories(), match_2_inventories_by_center match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_plot_repsurv_ccirc()

Examples

```
# Prepare everything required
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# Match inventory points
inv_pt_mtch <- match_2_inventories(inv_a, inv_b, "plot_id", "baysf")

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# Finally, make the match
match_trees_on_inventory_repsurv_ccirc(
  inv_pt_mtch, inv_a, inv_b, inv_a_trees, inv_b_trees,
  tree_id_style = "baysf"
)
```

`match_trees_on_plot_repsurv_ccirc`

Match Trees on Subsequent Inventories of One Plot

Description

Identifying trees that do or do not occur in two subsequent observations of one inventory plot of class fe_ccircle_spatial. Takes into account only trees with an entry in the tree_positions slot of the fe_ccircle_spatial object. Trees that have a removal flag in the first observation are excluded, because they are not relevant for increment calculations.

Usage

```
match_trees_on_plot_repsurv_ccirc(
  plot_in_inv_1st,
  plot_in_inv_2nd,
  tree_id_style = c("baysf", "generic"),
  inner_circle_non_matchable = FALSE
)
```

Arguments

`plot_in_inv_1st`
: An inventory plot represented as an object of class `fe_ccircle_spatial` or its child class `fe_ccircle_spatial_notrees`.

`plot_in_inv_2nd`
: An inventory plot representing a later survey of plot `plot_in_inv_1st`. Must be an object of class `fe_ccircle_spatial` or its child class `fe_ccircle_spatial_notrees`.

`tree_id_style`
: Character string that indicates how trees are uniquely identified on inventory plot level. This is required for matching trees on the same plot in both inventories. The options are "baysf" (default) and "generic". In the BaySF case, trees are matched by their polar coordinates, while in the generic case, the tree_ids must mark the same trees in both inventories.

`inner_circle_non_matchable`
: Logical, default `FALSE`. Only relevant for `tree_id_style == "generic"`: set to `TRUE` when the survey design renders inner-circle tree identifiers unreliable — e.g. when very small trees cannot practically be individually tagged in the field. In that case, all trees of the first inventory that belong to the inner circle by dbh (`dbh_cm < min(circle_definition$dbh_lower[dbh_lower > 0])`) are removed from the match pool, so the matching behaves analogously to the BaySF case on the inner circle. Inner-circle trees of the second inventory are left untouched; they will simply not find a partner. Has no effect for `tree_id_style == "baysf"` (where inner-circle trees of the first inventory are inherently unmatchable already).

Details

Typically called from within `match_trees_on_inventory_repsurv_ccirc` which in turn is called by `inv_increment_repsurv_ccirc`. The function does not check whether the two input inventory plots (`plot_in_inv_1st`, `plot_in_inv_2nd`) are really matching, i.e. represent two subsequent surveys of the same plot. This must be made sure before calling it.

Value

A data frame with seven columns, `tree_id_1st`, `tree_id_2nd`, `tree_exists_1st`, `tree_exists_2nd`, `time_yr_1st`, and `time_yr_2nd`. The first two contain the ids of the plot and the matching trees in the first inventory. If a tree is not present in one inventory, its id value is `NA`. Similarly, the third and the fourth column have the value `TRUE` or `FALSE` if a tree exists in an inventory or not. The column `tree_removal_2nd` indicates whether a tree has been registed as removed (`TRUE`) or not (`FALSE`) in the second inventory. The tree_id and the tree_exists columns are somewhat redundant, but their different formats may be useful in subsequent evaluations. The sixth and the seventh column contain the years of both inventories. These two columns do never have NA values.

Trees sharing one position

With `tree_id_style == "baysf"` trees are matched on their exact polar coordinates. If two trees of the same plot and survey carry the same distance and azimuth, that position identifies neither of them, and an exact-coordinate join would pair every inv-1 tree at the spot with every inv-2 tree at the spot — a Cartesian product that lets a tree enter the increment more than once.

The rule is therefore: **a position shared by more than one tree of the same plot and survey is treated as no position at all**. The affected trees are handled exactly like trees recorded without coordinates — they are excluded from the match and flagged `is_unmatchable_1st` resp. `is_unmatchable_2nd`. Their measured volumes still reach the standing-volume aggregates, they simply contribute no individual increment, and they never become pseudo-ingrowth candidates (their `tree_exists_*` stays `NA`). Each occurrence raises a warning naming the plot and the trees.

Duplicated coordinates are a defect of the source data, not a modelling choice, and they are rare (single-digit counts per real inventory). This rule keeps the increment computation well-defined and conservative rather than resolving the ambiguity by guessing which tree is which.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Other repeated inventory: harmonize_inv_for_repsurv(), inv_increment_repsurv_ccirc(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), reclassify_pseudo_ingrowth_ccirc()

Other inventory matches: harmonize_inv_for_repsurv(), match_2_inventories(), match_2_inventories_by_center match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc()

Examples

```
# Standard case: Match two single plots, both with the same set of trees

## Select two matching plots from two subsequent inventories
plot_1st <- data_ex3_previous_sample_fe_inventory |>
  dplyr::filter(plot_id == "900100002_1_99900002") |>
  purrr::pluck("plot", 1)

plot_2nd <- data_ex3_sample_fe_inventory |>
  dplyr::filter(plot_id == "900000002_1_99900002") |>
  purrr::pluck("plot", 1)

## Here's the actual matching
match_trees_on_plot_repsurv_ccirc(
  plot_1st, plot_2nd, tree_id_style = "baysf"
)


# Standard case: Match two single plots, some trees however missing in
#                the first or the second inventory
```

```
## Construct the data from the example above by clipping out trees from
## plot_1st
plot_2nd   <- plot_1st

miss_index <- c(1, 2, 4, 5)
plot_1st$trees <- plot_1st$trees |>
  dplyr::arrange(tree_id) |>
  dplyr::slice(- miss_index)
plot_1st$tree_positions <- plot_1st$tree_positions |>
  dplyr::arrange(tree_id) |>
  dplyr::slice(- miss_index)

miss_index <- c(3, 7, 8)
plot_2nd$trees <- plot_2nd$trees |>
  dplyr::arrange(tree_id) |>
  dplyr::slice(- miss_index)
plot_2nd$tree_positions <- plot_2nd$tree_positions |>
  dplyr::arrange(tree_id) |>
  dplyr::slice(- miss_index)
## Matching works with both styles in this example
match_trees_on_plot_repsurv_ccirc(
  plot_1st, plot_2nd, tree_id_style = "baysf"
)
match_trees_on_plot_repsurv_ccirc(
  plot_1st, plot_2nd, tree_id_style = "generic"
)


# Special case: First survey has no trees (all ingrowth)

## Construct an artifical example
plot_1st <- ForestElementsR::spruce_pine_ccircle_spatial_notrees
plot_2nd <- ForestElementsR::spruce_pine_ccircle_spatial

## Matching works with both styles in this example
match_trees_on_plot_repsurv_ccirc(
  plot_1st, plot_2nd, tree_id_style = "baysf"
)
match_trees_on_plot_repsurv_ccirc(
  plot_1st, plot_2nd, tree_id_style = "generic"
)


# Special case: Second survey has no trees (all felled)

## Construct an artifical example
plot_1st <- ForestElementsR::spruce_pine_ccircle_spatial
plot_2nd <- ForestElementsR::spruce_pine_ccircle_spatial_notrees

## Matching works with both styles in this example
match_trees_on_plot_repsurv_ccirc(
  plot_1st, plot_2nd, tree_id_style = "baysf"
)
```

```
match_trees_on_plot_repsurv_ccirc(
  plot_1st, plot_2nd, tree_id_style = "generic"
)
```

`output_base_table` *Format Base Tables for Output (Basistabellen)*

Description

Format Base Tables for Output (Basistabellen)

Usage

```
output_base_table(base_table)
```

Arguments

`base_table`	the output from either `base_table_d_q_class`, `base_table_age_class`, `base_table_d_q_class_mai` or `base_table_age_class_main_stand`

Value

A data frame containing aggregated information by species group, classified by age or dq class (rows)

Attached attributes

The returned object carries a `class_type` attribute (`"age"` or `"dq"`) recording its class axis, and propagates the `tree_selection` attribute set by the `*_table_*()` functions. `output_base_table_pdf` / `output_structure_table_pdf` read both (via the internal helper `.table_pdf_meta`) instead of taking a `type` or a cohort/layer argument.

See Also

Other inventory tables: `base_table_age_class()`, `base_table_age_class_main_stand()`, `base_table_d_q_class()`, `base_table_d_q_class_main_stand()`, `output_increment_overall()`, `output_structure_table()`, `structure_table_age_class()`, `structure_table_age_class_main_stand()`, `structure_table_d_q_class()`, `structure_table_d_q_class_main_stand()`

Examples

```
# The prepared tree list ships with the package; see
# ?data_ex3_trees_essentials for the chain that builds it.
trees_complete <- data_ex3_sample_trees_essentials

base_table  <- trees_complete |>
   base_table_age_class_main_stand()
```

```
base_table |> output_base_table()
```

`output_base_table_pdf` *Render an Output Base Table in PDF Format*

Description

Transform an inventory output base table as generated by `output_base_table` into a pdf file that can be exported or displayed.

Usage

```
output_base_table_pdf(
  x,
  tab_title,
  output_dir = NA,
  meta = NULL,
  inventory = NULL,
  language = c("ger")
)
```

Arguments

`x`	Input object, must be an output base table as generated with `output_base_table`
`tab_title`	Character, used for the table title and the file name.
`output_dir`	Path to the directory where the information sheet will be stored in. Default is NA; in this case, the file will be written into the temporary directory of the current R session. This directory will, however, only available as long as the session is going on.
`meta`	Optional named list with inventory meta-information to display above the table. Recognised names: `year` (inventory year, integer), `area_ha` (total area in ha, numeric), `n_plots` (number of inventory points, integer). Any or all elements may be omitted or `NULL`. When `meta` is `NULL` (default) no meta block is rendered - unless `inventory` is supplied (see below).
`inventory`	Optional `fe_inventory` object. A convenience shortcut for the `meta` block: when `meta` is `NULL` and `inventory` is given, the meta list is derived automatically via `inventory_meta`. An explicit `meta` always takes precedence.
`language`	Language of the rendered output. Currently only `"ger"` (German) is implemented and accepted; passing any other value raises an error. Future languages will be added as separate template files (one template per language, keeping each template linguistically consistent).

Details

The user must make sure that the input object x has been generated with output_base_table, because the function does not check that. For other objects the function will either produce an error or at least strange results.

The file name is assembled as output_base_table_ger_<tab_title>_<type>_<cohort>.pdf. The <cohort> tag identifies the tree cohort the table represents and is taken from the tag the base_table_*() functions attach to their result ("mainstand" for the *_main_stand() functions, "alllayers" for the general functions called with their default tree_filter, or a sanitised form of a custom tree_filter expression). This keeps the four standard variants (main stand / all layers × age / diameter classes) in separate, self-describing files instead of overwriting one another.

The class axis (<type>, age or dq) is likewise read off the object (the class_type attribute stamped by output_base_table), and the same cohort tag also selects the disclaimer note below the table (main-stand note, all-layers note, or - for a custom tree_filter - a plain "Benutzerdefinierte Auswahl" subtitle without a note). Neither is a function argument any longer, so they can no longer be set inconsistently with the actual table content.

Value

The path to the rendered pdf file

See Also

The stages that produce this function's input: output_base_table, fed by one of the four base tables base_table_age_class, base_table_age_class_main_stand, base_table_d_q_class, and base_table_d_q_class_main_stand. The structure-table counterpart of this function is output_structure_table_pd

Other pdf_output: check_pdf_dependencies(), output_increment_base_table_pdf(), output_increment_overall_p output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), output_structure_table pdf_dependencies, plot_info_sheet_pdf()

Examples

```
# Build the table (the species cast warnings are a natural side effect of
# the required species grouping)
base_tab <- data_ex3_sample_trees_essentials |>
  base_table_age_class_main_stand() |>
  output_base_table()

# Render it; output_dir defaults to tempdir(), which is what we want here
pdf_path <- output_base_table_pdf(base_tab, "example_ex3")
basename(pdf_path)
```

output_increment_base_table *Format Increment Base Tables for Output*

Description

Increment counterpart of output_base_table. It restructures the four-element increment base table into the same display layout (variables to rows, age or diameter classes to columns, plus the appended "Summe" cross-species block) used for the static base tables.

Usage

```
output_increment_base_table(increment_base_table)
```

Arguments

increment_base_table	the output from either increment_base_table or increment_base_table_main_stand

Details

Because the increment base tables share the exact four-element layout (list(detail, total, all_species, all_total)) and the same species_group / class-column convention as the static ones, this function reuses the generic restructuring of output_base_table. It is exposed as its own increment-family entry point so the increment output API stays discoverable and can diverge later (e.g. combining a value with its confidence interval) without touching the static side.

Value

A data frame with one row per species group and variable, the class levels as columns plus a total column, and the cross-species "Summe" block appended at the bottom (tagged with an NA species_group). The input's collective descriptor and its meta element are carried through as "collective" and "meta" attributes on the returned data frame, so the PDF step can label the collective (Hauptbestand / Alle Schichten / custom) and fill the header meta block without re-deriving them from the columns.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(, inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente, match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd, output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Examples

```
inv_b <- data_ex3_sample_fe_inventory
inv_b_trees <- data_ex3_sample_trees_essentials

inc <- inv_increment_gnfi3(inv_b, inv_b_trees)

# Plain and main-stand increment base tables, restructured for output
increment_base_table(inc, by_class = "age") |>
  output_increment_base_table()

increment_base_table_main_stand(inc, by_class = "age") |>
  output_increment_base_table()
```

`output_increment_base_table_pdf`

Render an Increment Base Table in PDF Format

Description

Increment counterpart of `output_base_table_pdf`. Turns the restructured increment base table from `output_increment_base_table` into a formatted PDF, using the unified header (Forstbetrieb name, collective line, meta line) modelled on the increment overview PDFs.

Usage

```
output_increment_base_table_pdf(
  x,
  tab_title,
  type = c("age", "dq"),
  output_dir = NA,
  meta = NULL,
  collective = NULL,
  ytables_used = NULL,
  language = c("ger")
)
```

Arguments

`x`	Input object, the output of `output_increment_base_table`. Its `"collective"` attribute is used unless `collective` is given.
`tab_title`	Character; the Forstbetrieb (or collective) name shown prominently in the header and used for the file name.
`type`	Character, `"age"` (default) or `"dq"`, must match how the table was built. Used for the subtitle, the class-column header and the file name.
`output_dir`	Path to the directory the PDF is written to. Default `NA`: the R session's temporary directory (gone when the session ends).

`meta` Optional named list with inventory meta-information for the header meta block. Recognised names: `year` (current inventory year), `area_ha` (total area in ha), `n_plots` (number of inventory points). For repeated-inventory increments the block additionally shows, if present, `inv_year_prev` (previous inventory year), `period_yr` (period length in years), `method` (Berechnungsmethode) and `fill_option` (Ergänzungsmethode) – the same block as the "Zuwachs auf Betriebsebene" overview PDF. Any name may be omitted. `NULL` (default) uses the `"meta"` attribute that the increment pipeline carries on `x` (assembled in `inv_inc_tree_extend` for a repeated inventory); pass a list to override it, or when the source carries none.

`collective` Optional collective descriptor list (`kind`/`label`/`definition`) overriding the one carried on `x`. `NULL` (default) uses `attr(x, "collective")`; if that is also absent, an "Alle Schichten" collective is assumed.

`ytables_used` Optional data frame of the yield tables used (columns `species_group`, `name_orig`), printed as a "Verwendete Ertragstafeln" list below the table (one row per table). `NULL` (default) uses `attr(x, "ytables_used")`, which the yield-table class tables carry; for the repeated-survey and gnfi3 tables it is absent and nothing is printed.

`language` Language of the rendered output. Currently only `"ger"` is implemented and accepted; other values raise an error. Future languages are added as separate template files.

Details

The header is built in four parts: the table title, the prominent Forstbetrieb name (`tab_title`) right below it, and a flushleft meta block modelled on the "Zuwachs auf Betriebsebene" overview PDF – a collective line ("Hauptbestand" / "Alle Schichten" / "Benutzerdefiniertes Kollektiv: ...") taken from the table's `collective` descriptor, followed by the inventory meta (year, area, point count and, for repeated inventories, the previous inventory year, period, Berechnungsmethode and Ergänzungsmethode). A footnote states whether the table is restricted to the main stand (a partial collective that allows virtual per-species hectare areas) or not.

The user must make sure that `x` was produced by `output_increment_base_table`; the function does not check that.

Value

The path to the rendered PDF file.

See Also

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_survey()`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center()`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_overall()`,

output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pd
output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Other pdf_output: check_pdf_dependencies(), output_base_table_pdf(), output_increment_overall_pdf(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), output_structure_table_pdf_dependencies, plot_info_sheet_pdf()

Examples

```
# A gnfi3 increment of the ex3 inventory; the ready-made tree list is
# shipped with the package, so the height chain does not have to be run
inc <- inv_increment_gnfi3(
  data_ex3_sample_fe_inventory, data_ex3_sample_trees_essentials
)

bt <- increment_base_table_main_stand(inc, by_class = "age") |>
  output_increment_base_table()

# output_dir defaults to tempdir()
pdf_path <- output_increment_base_table_pdf(
  bt,
  tab_title = "Forstbetrieb Beispiel",
  meta      = list(year = 2024, area_ha = 1234, n_plots = 42)
)
basename(pdf_path)
```

---

output_increment_overall

Restructure an Increment Big-Overview Into a Render-Ready Object

---

Description

Companion to inv_inc_big_overview: takes its rich multi-list output and produces a render-ready intermediate object that downstream functions (output_increment_overall_pdf, Shiny tables, csv exports) can consume without redoing the restructuring.

Usage

```
output_increment_overall(
  bo,
  fill_option = c("standard", "min_estimates", "all_estimates"),
  user_subtitle = NULL
)
```

Arguments

bo	Either an fe_increment_repsurv bundle, the output of inv_increment_repeated_survey (the usual case – its overview element is taken from it, and fill_option defaults to the one the bundle was built with), or a bare inv_inc_big_overview result.
fill_option	Which matched fill_option to display in the "Single-tree based" block of table_matched and the matching column block of table_combined. Default "standard" – the practical recommendation. The "Summaric / Yield-history" block always uses the inc_summaric row from the matched overview.
user_subtitle	Optional character giving an extra subtitle for the output (e.g. "Forstbetrieb Neustadt"). Default NULL.

Details

The intermediate object collects three parallel tables – one for matched plots, one for second-only plots, one for the combined enterprise view – with identical column shape. This lets the renderer stack them vertically with column-aligned numbers. CI95 columns are present in all three tables but only filled for the matches_only table; the gnfi3-based second_only and combined contributions stay empty by contract (NA) because their per-plot scatter is a model artefact and an empirical CI would mislead.

Language is deliberately a render-time concern, not a restructuring-time one: the species_group column is passed through as the original fe_species_* S3 vector, and the rendering layer (output_increment_overall_pdf or any alternative consumer) sets options(fe_spec_lang = ...) plus format() to choose the display language. This keeps the intermediate object reusable for multi-language exports from the same data.

Value

A list with the elements:

meta A list with title (named character vector indexed by language code), user_subtitle, method (the increment method, pulled from bo$method), fill_option, inv_year_prev, inv_year_curr, period_yr, and plot_match_stats (a list with n_total_2nd, n_matched, n_2nd_only, area_total_2nd_ha, area_matched_ha, area_2nd_only_ha).

table_matched One row per species_group plus a summary row (is_total_row = TRUE), with the eight numeric columns described below.

table_2nd_only Same column shape as table_matched. The "Single-tree based" block holds the gnfi3-backward estimate; the "Summaric" block is all NA (there is no yield-history balance for plots with only one observation).

table_combined Same column shape; columns hold the per-row sum of table_matched and table_2nd_only. All CI95 columns are NA.

The eight numeric columns in each table: iv_m3_yr_total_st, ci95_iv_m3_yr_total_st, iv_m3_ha_yr_st, ci95_iv_m3_ha_yr_st for the single-tree block; suffix _sum (for "summaric") on the same four for the yield-history block.

See Also

Other inventory tables: base_table_age_class(), base_table_age_class_main_stand(), base_table_d_q_class(), base_table_d_q_class_main_stand(), output_base_table(), output_structure_table(), structure_table_age_class(), structure_table_age_class_main_stand(), structure_table_d_q_class(), structure_table_d_q_class_main_stand()

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente, match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf, output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Examples

```
# Matching and gap filling of the ex3 pair are shipped ready-made, so the
# overall figures can be built without running the whole chain first.
inv_b       <- data_ex3_sample_fe_inventory
inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials
inc_rep     <- data_ex3_increment_matched
inc_fl      <- data_ex3_increment_fills

inc_sum <- inv_inc_summaric(
  inc_rep$tree_increments, inc_fl$fills_forward, inv_a_trees, inv_b_trees
)
bo <- inv_inc_big_overview(
  inc_fl, inv_a_trees, inv_b_trees, inv_b,
  style = "generic", inc_summaric = inc_sum
)

out <- output_increment_overall(
  bo,
  fill_option   = "standard",
  user_subtitle = "Forstbetrieb Neustadt"
)
str(out, max.level = 2)
```

---

output_increment_overall_pdf

Render an Increment Overview Table in PDF Format

Description

Thin wrapper around `rmarkdown::render()` that turns the restructured intermediate from `output_increment_overall` into a nicely formatted PDF. The intermediate's `meta` block drives the title, survey-year line, plot-match summary, and the fill_option subtitle; the three tibbles (`table_matched`, `table_2nd_only`, `table_combined`) are rendered stacked below with shared column shape and decimal-point-bound numbers.

Usage

```
output_increment_overall_pdf(x, output_dir = NA, language = c("ger"))
```

Arguments

`x`	Input object, must be the output of `output_increment_overall`.
`output_dir`	Path to the directory where the PDF will be written. Default `NA`; in this case the file goes into the R session's temporary directory and is gone when the session ends.
`language`	Language of the rendered output. Currently only `"ger"` (German) is implemented and accepted; passing any other value raises an error. An English variant is planned and will be added as a separate template (`output_increment_overall_template_eng.Rmd`), keeping each template linguistically consistent rather than scattering if-switches across one template.

Details

The user must make sure that the input object `x` has been generated with `output_increment_overall`; the function does not check that. Other objects will either produce an error or strange results.

Value

The path to the rendered PDF file.

See Also

Other pdf_output: `check_pdf_dependencies()`, `output_base_table_pdf()`, `output_increment_base_table_pdf()`, `output_increment_overview_gnfi3_pdf()`, `output_increment_overview_ytables_pdf()`, `output_structure_table_pdf_dependencies`, `plot_info_sheet_pdf()`

Other increment: `harmonize_inv_for_repsurv()`, `increment_base_table()`, `increment_base_table_main_stand()`, `increment_ytables_base_table()`, `inv_inc_big_overview()`, `inv_inc_big_overview_2nd_only()`, `inv_inc_big_overview_matches_only()`, `inv_inc_fill_gaps_gnfi3()`, `inv_inc_summaric()`, `inv_inc_tree_2_plot()`, `inv_inc_tree_consolidate()`, `inv_inc_tree_extend()`, `inv_inc_tree_extend_combined()`, `inv_increment_gnfi3()`, `inv_increment_repeated_survey()`, `inv_increment_repsurv_ccirc()`, `inv_increment_ytables()`, `is_fe_increment_repsurv()`, `match_2_inventories()`, `match_2_inventories_by_center()`, `match_2_inventories_by_plot_id()`, `match_plot_statistics()`, `match_trees_on_inventory_repsurv_ccirc()`, `match_trees_on_plot_repsurv_ccirc()`, `output_increment_base_table()`, `output_increment_base_table_pdf()`, `output_increment_overall()`, `output_increment_overview_gnfi3()`, `output_increment_overview_gnfi3_pdf()`, `output_increment_overview_ytables_pdf()`, `reclassify_pseudo_ingrowth_ccirc()`, `tree_inc_repsurv()`

Examples

```
# See output_increment_overall() for what the intermediate is built from;
# matching and gap filling of the ex3 pair are shipped ready-made.
inv_b       <- data_ex3_sample_fe_inventory
inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials
inc_rep     <- data_ex3_increment_matched
inc_fl      <- data_ex3_increment_fills

inc_sum <- inv_inc_summaric(
  inc_rep$tree_increments, inc_fl$fills_forward, inv_a_trees, inv_b_trees
)
bo <- inv_inc_big_overview(
  inc_fl, inv_a_trees, inv_b_trees, inv_b,
  style = "generic", inc_summaric = inc_sum
)
out <- output_increment_overall(
  bo,
  fill_option   = "standard",
  user_subtitle = "Forstbetrieb Neustadt"
)

# output_dir defaults to tempdir()
pdf_path <- output_increment_overall_pdf(out)
basename(pdf_path)
```

`output_increment_overview_gnfi3`

Restructure a gnfi3 Single-Inventory Increment Overview for Output

---

Description

Builds the species-level increment overview for the increment estimated from a single inventory with the tree growth functions of the Third German National Forest Inventory (gnfi3). The layout corresponds to table B of `output_increment_overall`: one row per species group plus a total row, each carrying the annual volume increment as an absolute total (`zv_m3_yr`) and per hectare (`zv_m3_ha_yr`). Confidence intervals are not reported here – consistent with the package-wide convention that the gnfi3 estimate carries no empirical confidence interval (unlike the repeated survey).

Usage

```
output_increment_overview_gnfi3(increment_base_table)
```

Arguments

increment_base_table
: A gnfi3 increment base table, the output of increment_base_table (or increment_base_table_main_s built from a single-inventory source (inv_increment_gnfi3). Its total and all_total elements drive the overview.

Details

Like the increment base tables, this is the language-neutral intermediate; species_group stays an fe_species_* vector so the render step decides the language. The table's collective descriptor is carried through as a "collective" attribute.

Value

A tibble with one row per species group plus a total row, the columns species_group, is_total_row, zv_m3_yr and zv_m3_ha_yr, and a "collective" attribute carried over from the input.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Examples

```
inv_b <- data_ex3_sample_fe_inventory
inv_b_trees <- data_ex3_sample_trees_essentials

inc <- inv_increment_gnfi3(inv_b, inv_b_trees)
increment_base_table(inc, by_class = "age") |>
  output_increment_overview_gnfi3()
```

---

output_increment_overview_gnfi3_pdf
Render a gnfi3 Single-Inventory Increment Overview in PDF Format

---

Description

Renders the overview from output_increment_overview_gnfi3 as a PDF, using the unified header (Forstbetrieb name, collective line, meta line) of the increment table PDFs. The German heading names the method in full ("Zuwachsfunktionen der 3. Bundeswaldinventur"); a footnote states that all volume figures are harvested volume under bark (Erntefestmeter ohne Rinde).

Usage

```
output_increment_overview_gnfi3_pdf(
  x,
  tab_title,
  output_dir = NA,
  meta = NULL,
  collective = NULL,
  language = c("ger")
)
```

Arguments

x	The output of output_increment_overview_gnfi3. Its "collective" attribute is used unless collective is given.
tab_title	Character; the Forstbetrieb name shown prominently in the header and used for the file name.
output_dir	Path to the directory the PDF is written to. Default NA: the R session's temporary directory.
meta	Optional named list with header meta-information (year, area_ha, n_plots); see output_increment_base_table_pdf.
collective	Optional collective descriptor overriding the one carried on x.
language	Language of the rendered output. Currently only "ger".

Value

The path to the rendered PDF file.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(, inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente, match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Other pdf_output: check_pdf_dependencies(), output_base_table_pdf(), output_increment_base_table_pdf(), output_increment_overall_pdf(), output_increment_overview_ytables_pdf(), output_structure_table_pdf(), pdf_dependencies, plot_info_sheet_pdf()

Examples

```
# A gnfi3 increment of the ex3 inventory; the ready-made tree list is
# shipped with the package, so the height chain does not have to be run
ov <- inv_increment_gnfi3(
  data_ex3_sample_fe_inventory, data_ex3_sample_trees_essentials
) |>
  increment_base_table(by_class = "age") |>
  output_increment_overview_gnfi3()

# output_dir defaults to tempdir()
pdf_path <- output_increment_overview_gnfi3_pdf(
  ov, tab_title = "Forstbetrieb Beispiel"
)
basename(pdf_path)
```

---

output_increment_overview_ytables_pdf

Render a Yield-Table Increment Overview in PDF Format

Description

Renders the overview of an inv_increment_ytables result as a PDF with the unified header (Forstbetrieb name, collective line, meta line). Two tables are shown: the species-group overview (virtual area, mean site index, increment per ha and total) and a listing of the yield tables used per species group. A footnote states that all volumes are harvested volume under bark (Erntefestmeter ohne Rinde) and that the table is restricted to the main stand.

Usage

```
output_increment_overview_ytables_pdf(
  x,
  tab_title,
  output_dir = NA,
  meta = NULL,
  language = c("ger")
)
```

Arguments

x	The result of inv_increment_ytables, or any list carrying overview, ytables_used and collective. A bundle also supplies the header figures, so meta can be omitted.

tab_title	Character; the Forstbetrieb name shown prominently in the header and used for the file name.
output_dir	Path to the directory the PDF is written to. Default NA: the R session's temporary directory.
meta	Optional named list with header meta-information (year, area_ha, n_plots).
language	Language of the rendered output. Currently only "ger". The German table uses only the German yield-table names (name_orig).

Value

The path to the rendered PDF file.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), reclassify_pseudo_ingrowth_ccirc(), tree_inc_repsurv()

Other pdf_output: check_pdf_dependencies(), output_base_table_pdf(), output_increment_base_table_pdf(), output_increment_overall_pdf(), output_increment_overview_gnfi3_pdf(), output_structure_table_pdf(), pdf_dependencies, plot_info_sheet_pdf()

Examples

```
# FeNEU's own yield-table collection covers all current bavrn_state
# species codes; the ForestElementsR collection does not.
inc <- inv_increment_ytables(
  data_ex3_sample_fe_inventory,
  data_ex3_sample_trees_essentials,
  ytables_bavrn_state_var_1_feneu
)

# output_dir defaults to tempdir()
pdf_path <- output_increment_overview_ytables_pdf(
  inc, tab_title = "Forstbetrieb Beispiel"
)
basename(pdf_path)
```

output_structure_table *Format Structure Tables for Output (Strukturtabellen)*

Description

A data frame with group-wise aggregated inventory information separated per species is returned.

Usage

```
output_structure_table(structure_table)
```

Arguments

structure_table	structure table generated from structure_table_d_q_class_main_stand, structure_table_age_cla structure_table_d_q_class, or structure_table_age_class. This is a list with *detail*, *total*, *all_species*, *all_total*, *species_total*, *species_dbh_total*, *grand_total*, and *grand_dbh_total* data frames; a bare *detail* data frame is also accepted (in which case no cross-species "Summe" block or "Summe" rows are added, and the total column uses rowSums).

Value

A data frame containing aggregated information by species group, separated by age or dq class (rows) and classified by single tree diameter class (columns). When all_species is supplied, a trailing cross-species "Summe" group (with an NA species group) is appended. When species_total is supplied, each species' block gets its own trailing "Summe" row (grand total across all classes and dbh classes for that species); when species_dbh_total is also supplied, that same row's dbh_class columns are filled with the species' per-dbh_class sums/ weighted means across all classes (mirroring BaySF's own report), rather than being left blank. When grand_total/grand_dbh_total are supplied, the cross-species "Summe" block gets the equivalent trailing "Summe" row - the true grand total across every species, class, and dbh class combined.

Attached attributes

The returned object carries a class_type attribute ("age" or "dq") recording its class axis, and propagates the tree_selection attribute set by the *_table_*() functions. output_base_table_pdf / output_structure_table_pdf read both (via the internal helper .table_pdf_meta) instead of taking a type or a cohort/layer argument.

See Also

Other inventory tables: base_table_age_class(), base_table_age_class_main_stand(), base_table_d_q_class(), base_table_d_q_class_main_stand(), output_base_table(), output_increment_overall(), structure_table_age_class(), structure_table_age_class_main_stand(), structure_table_d_q_class(), structure_table_d_q_class_main_stand()

Examples

```
# The prepared tree list ships with the package; see
# ?data_ex3_trees_essentials for the chain that builds it. Three inventory
# points are enough to show the shape of the result, and they keep the
# example quick; the function takes a whole tree list alike.
trees_complete <- data_ex3_sample_trees_essentials |>
  dplyr::filter(plot_id %in% unique(plot_id)[1:3])

structure_table <- trees_complete |>
   structure_table_age_class_main_stand()

structure_table |> output_structure_table()
```

---

output_structure_table_pdf

Render an Output Structure Table in PDF Format

Description

Transform an inventory output structure table as generated by `output_structure_table` into a pdf file that can be exported or displayed.

Usage

```
output_structure_table_pdf(
  x,
  tab_title,
  output_dir = NA,
  meta = NULL,
  inventory = NULL,
  language = c("ger")
)
```

Arguments

`x`	Input object, must be an output structure table as generated with `output_structure_table`
`tab_title`	Character, used for the table title and the file name.
`output_dir`	Path to the directory where the information sheet will be stored in. Default is NA; in this case, the file will be written into the temporary directory of the current R session. This directory will, however, only available as long as the session is going on.
`meta`	Optional named list with inventory meta-information to display above the table. Recognised names: `year` (inventory year, integer), `area_ha` (total area in ha, numeric), `n_plots` (number of inventory points, integer). Any or all elements may be omitted or `NULL`. When `meta` is `NULL` (default) no meta block is rendered - unless `inventory` is supplied (see below).

`inventory` Optional `fe_inventory` object. A convenience shortcut for the `meta` block: when `meta` is `NULL` and `inventory` is given, the meta list is derived automatically via `inventory_meta`. An explicit `meta` always takes precedence. Supplying `inventory` additionally guards against nonsensical output: structure tables are only meaningful for sample (concentric-circle) inventories, so a standwise / stand-level inventory (any plot that is not an `fe_ccircle_spatial`) is rejected with a clear error.

`language` Language of the rendered output. Currently only `"ger"` (German) is implemented and accepted; passing any other value raises an error. Future languages will be added as separate template files (one template per language, keeping each template linguistically consistent).

Details

The user must make sure that the input object x has been generated with `output_structure_table`, because the function does not check that. For other objects the function will either produce an error or at least strange results.

The file name is assembled as `output_structure_table_ger_<tab_title>_<type>_<cohort>.pdf`. The `<cohort>` tag identifies the tree cohort the table represents and is taken from the tag the `structure_table_*()` functions attach to their result (`"mainstand"` for the `*_main_stand()` functions, `"alllayers"` for the general functions called with their default `tree_filter`, or a sanitised form of a custom `tree_filter` expression). This keeps the four standard variants (main stand / all layers × age / diameter classes) in separate, self-describing files instead of overwriting one another.

The class axis (`<type>`, age or dq) is likewise read off the object (the `class_type` attribute stamped by `output_structure_table`), and the same cohort tag also selects the disclaimer note (mainstand note, all-layers note, or - for a custom `tree_filter` - a plain "Benutzerdefinierte Auswahl" subtitle without a note). Neither is a function argument any longer, so they can no longer be set inconsistently with the actual table content.

Value

The path to the rendered pdf file

See Also

The stages that produce this function's input: `output_structure_table`, fed by one of the four structure tables `structure_table_age_class`, `structure_table_age_class_main_stand`, `structure_table_d_q_clas` and `structure_table_d_q_class_main_stand`. The base-table counterpart of this function is `output_base_table_pdf`.

Other pdf_output: `check_pdf_dependencies()`, `output_base_table_pdf()`, `output_increment_base_table_pdf()`, `output_increment_overall_pdf()`, `output_increment_overview_gnfi3_pdf()`, `output_increment_overview_ytabl` `pdf_dependencies`, `plot_info_sheet_pdf()`

Examples

```
# Build the table (the species cast warnings are a natural side effect of
# the required species grouping)
```

```
structure_tab <- data_ex3_sample_trees_essentials |>
  structure_table_age_class_main_stand() |>
  output_structure_table()

# Render it; output_dir defaults to tempdir(), which is what we want here
pdf_path <- output_structure_table_pdf(structure_tab, "example_ex3")
basename(pdf_path)
```

`pdf_dependencies` *Setup for PDF Output Functions*

Description

The `*_pdf()` functions in FeNEU render their tables through **rmarkdown** + **kableExtra** on top of pandoc and a LaTeX distribution. Pandoc and LaTeX are not R packages; they are external tools that must be installed once per machine.

R packages

The R-side dependencies are listed as *Suggests* in the FeNEU `DESCRIPTION`. They are not pulled in by `install.packages("FeNEU")`; users who want PDF output must install them explicitly:

```
install.packages(c("rmarkdown", "kableExtra", "tinytex"))
```

pandoc

- Check via `rmarkdown::pandoc_available()` (returns `TRUE` when a usable pandoc is on the path).
- RStudio bundles pandoc and rmarkdown picks it up automatically.
- A standalone installer for every platform is available at https://pandoc.org/installing.html. On macOS, also `brew install pandoc`; on Debian/Ubuntu, `apt install pandoc`.

LaTeX

- The easiest path for R users is **tinytex**:

  ```
  install.packages("tinytex")
  tinytex::install_tinytex()
  ```

  The second call downloads a small LaTeX distribution (~ 300 MB) to the user's home directory. From then on, missing LaTeX packages required by FeNEU's templates (siunitx, booktabs, makecell, longtable, ...) are fetched automatically when a PDF is rendered.
- Alternatively, any TeX Live or MiKTeX installation works as long as those LaTeX packages are available.

Diagnostic

To verify the environment before rendering a first PDF:

```
rmarkdown::pandoc_available()    # pandoc on PATH?
rmarkdown::pandoc_version()      # which version?
tinytex::is_tinytex()            # tinytex installed?
Sys.which("pdflatex")            # latex on PATH?
```

See Also

Other pdf_output: check_pdf_dependencies(), output_base_table_pdf(), output_increment_base_table_pdf(), output_increment_overall_pdf(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytabl output_structure_table_pdf(), plot_info_sheet_pdf()

Examples

```
# See whether everything is in place to render a FeNEU PDF
rmarkdown::pandoc_available()
requireNamespace("kableExtra", quietly = TRUE)
```

---

plot_info_sheet_pdf *Render an Information Sheet for an Inventory Plot as a PDF File*

Description

Render an Information Sheet for an Inventory Plot as a PDF File

Usage

```
plot_info_sheet_pdf(
  x,
  title = NA,
  output_dir = NA,
  dbh_scale = 4,
  language = c("ger")
)
```

Arguments

x	An object of class fe_ccircle_spatial
title	id or name for the title of the plot and file. Default is NA, which means that the entry stand_id of the input object x will be used. The actual file name will be "plot_info_ger_" followed by title and the appendix ".pdf".
output_dir	Path to the directory where the information sheet will be stored in. Default is NA; in this case, the file will be written into the temporary directory of the current R session. This directory will, however, only available as long as the session is going on.

dbh_scale	Factor for oversizing the trees' dbh in the plot
language	Language of the rendered output. Currently only "ger" (German) is implemented and accepted; passing any other value raises an error. Future languages will be added as separate template files (one template per language, keeping each template linguistically consistent).

Value

The path to the rendered output file

See Also

Other pdf_output: check_pdf_dependencies(), output_base_table_pdf(), output_increment_base_table_pdf(), output_increment_overall_pdf(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytabl output_structure_table_pdf(), pdf_dependencies

Examples

```
# One info sheet for the first inventory point of ex3.
# output_dir defaults to tempdir().
file_path <- data_ex3_sample_fe_inventory$plot[[1]] |>
  plot_info_sheet_pdf()
basename(file_path)

# Several sheets in one go (not run here -- one render per plot)
# data_ex3_sample_fe_inventory$plot[1:10] |> lapply(plot_info_sheet_pdf)
```

---

print.fe_increment_gnfi3

Print an fe_increment_gnfi3 Object

Description

Print an fe_increment_gnfi3 Object

Usage

```
## S3 method for class 'fe_increment_gnfi3'
print(x, ...)
```

Arguments

x	An fe_increment_gnfi3 object
...	Other parameters (not used)

Value

x, invisibly

Examples

```
inv_increment_gnfi3(
  data_ex3_sample_fe_inventory, data_ex3_sample_trees_essentials
)
```

`print.fe_increment_repsurv`
Print an fe_increment_repsurv Object

Description

Print an fe_increment_repsurv Object

Usage

```
## S3 method for class 'fe_increment_repsurv'
print(x, ...)
```

Arguments

`x`	An `inv_increment_repeated_survey` result.
`...`	Not used.

Value

`x`, invisibly.

`print.fe_increment_ytables`
Print an fe_increment_ytables Object

Description

Print an fe_increment_ytables Object

Usage

```
## S3 method for class 'fe_increment_ytables'
print(x, ...)
```

Arguments

`x`	An `fe_increment_ytables` object
`...`	Other parameters (not used)

Value

x, invisibly

Examples

```
inv_increment_ytables(
  data_ex3_sample_fe_inventory, data_ex3_sample_trees_essentials,
  ytables_bavrn_state_var_1_feneu
)
```

`processed_heights_bav_sub10`

Sample Inventory Trees With Estimated Heights (Bavarian)

Description

Sample Inventory Trees With Estimated Heights (Bavarian)

Details

These data is mainly provided for internal testing. It is a tibble that has been created by applying `height_complete_inventory` to the pulled tree data frame `processed_pulled_sub10`, with the Bavarian standard height curve system (see `h_standard_bv`).

See Also

Other example data: `data_ex1_sample_raw`, `data_ex2_sample_raw`, `data_ex3_increment_interim`, `data_ex3_previous_sample_fe_inventory`, `data_ex3_sample_fe_inventory`, `data_ex3_trees_essentials`, `data_ex4_previous_sample_fe_inventory`, `data_ex4_sample_fe_inventory`, `data_ex5_previous_sample_fe_inver` `data_ex5_sample_fe_inventory`, `data_ex6_standwise_fe_inventory`, `data_ex7_standwise_fe_inventory`, `data_examples_overview`, `inc_sub10_rep_classic`, `inc_sub10_rep_end`, `inc_sub10_rep_mean`, `inc_sub10_rep_trans`, `processed_heights_nfi_sub10`, `processed_pulled_sub10`

`processed_heights_nfi_sub10`

Sample Inventory Trees With Estimated Heights (NFI)

Description

Sample Inventory Trees With Estimated Heights (NFI)

Details

These data is mainly provided for internal testing. It is a tibble that has been created by applying `height_complete_inventory` to the pulled tree data frame `processed_pulled_sub10`, with the German National Forest Inventory (NFI) standard height curve system ((see `h_standard_gnfi3`).

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_pulled_sub10

---

processed_pulled_sub10 *Example Inventory as Pulled Trees (10 inventory points)*

Description

Example Inventory as Pulled Trees (10 inventory points)

Details

Mainly provided for internal testing. It is a tibble that has been created by applying pull_trees to the ten-point fe_inventory object data_ex3_sample_fe_inventory.

See Also

Other example data: data_ex1_sample_raw, data_ex2_sample_raw, data_ex3_increment_interim, data_ex3_previous_sample_fe_inventory, data_ex3_sample_fe_inventory, data_ex3_trees_essentials, data_ex4_previous_sample_fe_inventory, data_ex4_sample_fe_inventory, data_ex5_previous_sample_fe_inver data_ex5_sample_fe_inventory, data_ex6_standwise_fe_inventory, data_ex7_standwise_fe_inventory, data_examples_overview, inc_sub10_rep_classic, inc_sub10_rep_end, inc_sub10_rep_mean, inc_sub10_rep_trans, processed_heights_bav_sub10, processed_heights_nfi_sub10

---

pull_centers *Pull Center Point Coordinates From an fe_inventory Object*

Description

Pull Center Point Coordinates From an fe_inventory Object

Usage

```
pull_centers(x)
```

Arguments

x	An object of class fe_inventory

Value

An object of class `sf` with the column `plot_id` as taken from the input object `x`, and a geometry column that represents each plot's center point. In case the `plot` column of `x` does not contain objects that allow to extract meaningful center coordinates the resulting geometry will be empty.

Examples

```
data_ex3_sample_fe_inventory |> pull_centers()
```

---

pull_circle_definitions

Pull Circle Definitions From an fe_inventory Object

Description

Extracts per-plot slope and minimum circle area from the circle definition of an `fe_inventory` object. Useful for reconstructing small-tree expansion factors or for diagnostic purposes.

Usage

```
pull_circle_definitions(forest_inventory, .progress = TRUE)
```

Arguments

`forest_inventory`	Object of class `fe_inventory`.
`.progress`	Logical, if TRUE a progress bar is shown.

Value

Tibble with one row per plot, containing at least `plot_id`, `slope` and `min_c_area`.

Examples

```
pull_circle_definitions(data_ex3_sample_fe_inventory)
```

pull_sums *Generate Layer-Wise Stand Sum and Mean Values per Plot of an fe_inventory Object and Hand them Back as a Tibble*

Description

Basically, this function calls the function stand_sums_static. According to the behaviour of that function, If the height values provided with an inventory plot are not complete, all output values that require height for calculation (i.e. meam and dominant heights, tree volumes) will be NA.

Usage

```
pull_sums(forest_inventory, hd_dom_method = "Weise", .progress = TRUE)
```

Arguments

forest_inventory	An fe_inventory object
hd_dom_method	Method for calculating the dominant diameter and dominant height. The default choice is "Weise". See the documentation of stand_sums_static for more options and details.
.progress	Logical, if TRUE (default) a progress bar will be shown during execution. Check the documentation of map for more options.

Value

A tibble containing the calculated sum and mean values

Examples

```
# Very small example
fe_inv <- data_ex3_sample_fe_inventory[1:3, ]

# Example with incomplete heights
fe_inv |> pull_sums(.progress = FALSE)

# Complete all heights with estimates ...
suppressWarnings(
# Warnings come from intentional species code casts - no problem here
  all_trees_with_heights <- fe_inv |>
    pull_trees() |>
    height_complete_inventory()
)

# ... and run pull_sums() again
fe_inv |>
  fill_heights_back(all_trees_with_heights, .progress = FALSE) |>
  pull_sums(.progress = FALSE)
```

pull_trees *Pull All Trees From an fe_inventory Object Into a Tibble*

Description

While it has many advantages to keep inventory data in an object of class `fe_inventory`, having one large data frame comprising all trees is more convenient for many standard evaluation purposes, This function pulls such a data frame (tibble) from an `fe_inventory` object.

Usage

```
pull_trees(forest_inventory, small_trees = FALSE, .progress = TRUE)
```

Arguments

`forest_inventory`	Object of class `fe_inventory`
`small_trees`	**[Experimental]** logical, default is FALSE. Only users who absolutely know what they are doing, should use the setting TRUE. If TRUE also small trees (i.e. trees with heights < 1.3 m) will be pulled. This is still highly experimental, recommended for insiders only.
`.progress`	Logical, if TRUE (default) a progress bar will be shown during execution. Check the documentation of map for more options.

Value

A tibble with one row per tree, holding the trees of all inventory plots of `forest_inventory` in a single flat table. It is not an object of a FeNEU class; the plot each tree belongs to is carried in the column `plot_id` instead. The leading columns are `plot_id`, `area_rep_ha` (the area in ha the plot represents), `layer_key`, and `species_id` (an `fe_species` vector of whatever coding the inventory uses); they are followed by the tree attributes of the plots' `trees` slots, among them `tree_id`, `time_yr`, `age_yr`, `dbh_cm`, `height_m`, `removal`, `ingrowth`, and `h_m_tree` (whether the height was measured or estimated).

Two columns are added by this function and are not tree attributes as such. `n_rep_ha` is the number of trees per hectare the tree represents; it comes from the plot but is multiplied here by the tree's `tree_count`, so that a record standing for more than one tree is correctly weighted (`tree_count` itself is dropped afterwards). `dbh_trshld` is the lower DBH limit of the concentric circle the tree was sampled in; it is `NA` for plots that are not `fe_ccircle_spatial`, i.e. for stand-wise inventories.

With `small_trees = TRUE` the small trees of all plots are appended as further rows, with `crown_base_height_m`, `crown_radius_m`, and `dbh_trshld` set to `NA`.

Examples

```
data_ex3_sample_fe_inventory |> pull_trees()
```

`read_and_convert_data` *Read Preprocessed Inventory Data Into an fe_inventory (expert dispatcher)*

Description

Expert-level function – standard users do not need it. The normal import path is two calls: the matching `raw_to_pre` converter for your raw format, followed by the matching `pre_to_fe_inventory` reader, which alone turns the preprocessed on-disk files into a finished `fe_inventory` object (see `import_sample_concentric_pre_to_fe_inventory` and `import_standwise_relascope_pre_to_fe_inventory`).

Usage

```
read_and_convert_data(
  input_path,
  inventory_type = c("sample_concentric", "standwise_relascope"),
  style = NULL,
  coord_sys = NULL,
  check_envelope = TRUE
)
```

Arguments

`input_path`	Path of the folder holding the preprocessed input files (character). The required filenames depend on `inventory_type`; see the routed `pre_to_fe_inventory` functions above.
`inventory_type`	Character. Which inventory chain to read. One of `"sample_concentric"` or `"standwise_relascope"`.
`style`	Character or `NULL`. Format family of the preprocessed data within the chosen chain. If `NULL` (default), the chain's only current style is used: `"baysf"` (Bavarian State Forest style) for `"sample_concentric"`, `"silvarith"` (Silvarith-style) for `"standwise_relascope"`. The argument is explicit (not auto-detected): the right style follows from which raw format was imported.
`coord_sys`	Character string specifying the coordinate reference system of the input plot coordinates (used only by `"sample_concentric"`; ignored for stand-wise data). Supported values: `"lonlat"` WGS84 geographic, EPSG:4326 (degrees) `"etrs89"` ETRS89 geographic, EPSG:4258 (degrees) `"utm32"`, `"utm33"` ETRS89 / UTM 32N, 33N (EPSG:25832, 25833) `"gk2"` **...** `"gk5"` DHDN / Gauss-Krueger zones 2-5 (EPSG:31466-31469) If `NULL`, a format-specific default is used (`"gk4"` for the BaySF style). The **geographic** systems (`"lonlat"`, `"etrs89"`) are accepted, but their coordinates are **reprojected to the matching UTM zone** during import (a message names the zone). This is not cosmetic: tree positions are built metrically from distance and azimuth around the plot centre, which is impossible in degrees.

The plot coordinates of the returned object are therefore in UTM, not in the geographic system supplied.

Note that `coord_sys` must be stated **explicitly** whenever the eastings are stored without the Gauss-Krueger zone prefix (the usual BaySF form): adding the prefix follows from your declaration, but would be a mere guess if the system had only been defaulted – so the import stops and asks. See `import_sample_concentric_pre_to_fe_inven`

`check_envelope` Logical, default `TRUE`; see `import_sample_concentric_pre_to_fe_inventory`.

Details

This function is a convenience dispatcher for **expert / programmatic** use – e.g. when the inventory chain has to be chosen at run time from an argument. It routes to the matching `pre_to_fe_inventory` step of the two import chains and derives the correct `object_type` from the inventory type, so callers cannot mismatch them:

`"sample_concentric"` concentric-circle sample inventory -> `fe_ccircle_spatial` plots, via `import_sample_concentri`

`"standwise_relascope"` stand-wise relascope ("Winkelzählprobe" / angle-count) inventory -> `fe_stand` plots, via `import_standwise_relascope_pre_to_fe_inventory`.

The input must already be in the preprocessed on-disk form of the respective chain (a folder of BaySF-style files for `"sample_concentric"`, a `WZP_Daten.txt` for `"standwise_relascope"`). Raw vendor formats are brought into that form first by the matching `raw_to_pre` converter – see `import_sample_concentric_format1_raw_to_pre`, `import_sample_concentric_format2_raw_to_pre` and `import_standwise_relascope_format1_raw_to_pre`.

Value

An `fe_inventory` object (`fe_ccircle_spatial` plots for `"sample_concentric"`, `fe_stand` plots for `"standwise_relascope"`).

See Also

`import_sample_concentric_pre_to_fe_inventory`, `import_standwise_relascope_pre_to_fe_inventory`

Examples

```
# Concentric-circle sample inventory in BaySF-style preprocessed form
inv_path <- system.file("extdata", "data_ex3_sample_pre",
                        package = "FeNEU")
read_and_convert_data(inv_path, inventory_type = "sample_concentric",
                      coord_sys = "gk4")
```

reclassify_pseudo_ingrowth_ccirc *Reclassify Pseudo-Ingrowth Trees on Concentric-Circle Plots*

Description

In repeated inventories with concentric sampling circles, trees that grow across the dbh threshold of the innermost circle (typically 12 cm) between two surveys are often mis-classified as ingrowth — even though they physically existed during the first survey, just on the inner circle, where they may not have been individually identifiable. This function corrects such pseudo-ingrowth trees so that subsequent steps of the increment calculation (`tree_inc_repsurv` and `inv_inc_summaric`) treat them as re-measured trees with a circle transition rather than as fresh ingrowth.

Usage

```
reclassify_pseudo_ingrowth_ccirc(
  matched_trees,
  inv_2nd,
  inv_2nd_trees,
  tree_id_style = c("baysf", "generic"),
  inner_circle_non_matchable = FALSE
)
```

Arguments

`matched_trees`	A data frame as produced by `match_trees_on_inventory_repsurv_ccirc`.
`inv_2nd`	`fe_inventory` object representing the later of the two inventories. Used to access `tree_positions` (for the polar distance `R`) and `circle_definition` (for the inner radius and the per-circle representation numbers).
`inv_2nd_trees`	Extended tree data frame for the second inventory (same object that was passed to `match_trees_on_inventory_repsurv_ccirc`). Provides `species_id`, `age_yr`, `dbh_cm`, `height_m`, and `v_hub_m3` for the gnfi3 backward estimate.
`tree_id_style`	Character string indicating how trees are uniquely identified on inventory plot level. Must be the same value that was used when matching. Options are `"baysf"` (default) and `"generic"`.
`inner_circle_non_matchable`	Logical, default `FALSE`. Only relevant for `tree_id_style == "generic"`: must be `TRUE` if the survey design renders inner-circle identifiers unreliable. Ignored for `tree_id_style == "baysf"`, where inner-circle trees of the first inventory are always treated as unmatchable.

Details

A tree is classified as pseudo-ingrowth if all of the following hold:

- It is currently classified as ingrowth (`tree_exists_1st == FALSE` and `tree_exists_2nd == TRUE`).

- Its polar distance from the plot centre at the second survey (`R` from `tree_positions`) is smaller than the radius of the innermost circle of the plot's `circle_definition`.

The innermost circle is treated as perfectly concentric. In practice it is sometimes laid out slightly off-centre (to avoid trampling small trees), but the displacement is random in direction and magnitude, so individual mis-classifications average out across many plots.

For pseudo-ingrowth trees, the first-survey volume is reconstructed via the gnfi3 growth functions (`d_age_gnfi3`, `h_age_gnfi3`, `v_gri`, and `v_red_harvest_ubark`) over the full inter-survey period (`dt = -(time_yr_2nd - time_yr_1st)`). The estimated volume is clamped at zero. The first-survey representation number is always set to the inner circle's value, because the very premise of pseudo-ingrowth identification is that the tree was sampled on the inner circle at the first survey (Inv-2 position inside r_inner, plus unmatchable). The gnfi3 back-estimated dbh is only used for the volume; the sampling circle is a fact of the survey design, not a model output.

The function leaves all other trees in `matched_trees` untouched. It always adds the column `pseudo_ingrowth_corrected` (boolean), with `TRUE` for the reclassified trees and `FALSE` for all others. This flag is meant to remain visible downstream so consumers can analyse the corrected subset (e.g. for a separate tree-status category in aggregation overviews).

Downstream consumer contract:

- The per-tree methods (`tree_inc_repsurv` with `rep_classic`, `rep_mean`, `rep_end`, `rep_trans`) consume the corrected state and benefit from the reclassification — pseudo-ingrowth trees are processed through the standard both-exist circle-transition branches.
- `inv_inc_summaric` explicitly undoes the reclassification on its working copy via the `pseudo_ingrowth_corrected` flag, because the ertragsgeschichtliche balance is by definition built on actually measured volumes and the inv_1 inner-circle cohort was never measured. There the affected trees enter only via the inv_2 volume sum, as they did before the reclassification.

The reclassification is only meaningful when inner-circle trees of the first inventory are inherently unmatchable:

- For `tree_id_style == "baysf"` this is always the case (BaySF data have no identifiers for inner-circle trees in the first survey).
- For `tree_id_style == "generic"` only when the survey design marks inner-circle identifiers as unreliable. In that case the caller must set `inner_circle_non_matchable = TRUE` both here and in the upstream `match_trees_on_plot_repsurv_ccirc` call so that Inv-1 inner-circle trees are dropped from the match.

In all other situations the function simply adds the `pseudo_ingrowth_corrected` column as all-`FALSE` and leaves the data untouched.

Value

The input `matched_trees` with two changes:

- For pseudo-ingrowth trees: `tree_exists_1st` flipped to `TRUE`; `v_hub_m3_1st` set to the gnfi3-back-estimated volume (clamped at 0); `n_rep_ha_1st` set to the representation number corresponding to the back-estimated dbh on the same plot.
- A new column `pseudo_ingrowth_corrected` (logical), with `TRUE` for reclassified trees and `FALSE` for all others.

Note that tree_id_1st deliberately stays NA on these rows. The reclassification asserts that the tree *existed* at inv 1, not that an individual inv-1 record was identified — there is none, which is exactly why the tree was unmatchable. A reclassified row therefore carries tree_exists_1st = TRUE together with tree_id_1st = NA, and any downstream count that keys on tree_id_1st will not see it. This is why n_in_both_1st and n_in_both_2nd of match_tree_statistics differ by exactly the number of pseudo-ingrowth trees.

The existing v_hub_m3_at_thrsh value is preserved as-is — it was already correctly computed for the ingrowth case by calc_v_hub_m3_at_thrsh, and that same threshold volume is what the circle-transition branch of tree_inc_repsurv needs after reclassification.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), tree_inc_repsurv()

Other repeated inventory: harmonize_inv_for_repsurv(), inv_increment_repsurv_ccirc(), match_2_inventories(), match_2_inventories_by_center_coord(), match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc()

Examples

```
# Prepare two matching inventories
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

inv_pt_mtch <- match_2_inventories(inv_a, inv_b, "plot_id", "baysf")
mtch_trs <- match_trees_on_inventory_repsurv_ccirc(
  inv_pt_mtch, inv_a, inv_b, inv_a_trees, inv_b_trees,
  tree_id_style = "baysf"
)

# Reclassify pseudo-ingrowth — default tree_id_style "baysf"
mtch_trs_corr <- reclassify_pseudo_ingrowth_ccirc(
  mtch_trs, inv_b, inv_b_trees, tree_id_style = "baysf"
)
sum(mtch_trs_corr$pseudo_ingrowth_corrected)
```

setup_pdf_toolchain *Install Missing PDF Rendering Tools*

Description

Installs TinyTeX if no LaTeX distribution is found on the current system. For pandoc, installation must be done manually (see diagnose_pdf_toolchain for instructions).

Usage

```
setup_pdf_toolchain()
```

Value

Invisibly, the result of diagnose_pdf_toolchain after the setup attempt.

See Also

diagnose_pdf_toolchain

Examples

```
## Not run:
# Deliberately not run, and deliberately not merely skipped on the check
# farm either: this installs TinyTeX into the user's own filespace. That
# must never happen unasked, least of all during a package check.
setup_pdf_toolchain()

## End(Not run)
```

se_area_weighted *Area-Weighted Standard Error and Confidence Interval*

Description

Computes the standard error and confidence interval for an area-weighted mean and its corresponding total, based on the ratio estimator approach. This is the standard method for forest inventory data where plots have different representation areas (e.g. due to slope correction in concentric circle designs).

Usage

```
se_area_weighted(plot_values, area_weights, conf_level = 0.95)
```

Arguments

`plot_values`	Numeric vector of per-hectare values, one per plot (e.g. volume in m3/ha, increment in m3/ha/yr).
`area_weights`	Numeric vector of representation areas in hectares, same length as `plot_values`. Must be strictly positive.
`conf_level`	Confidence level for the interval, default 0.95.

Details

The area-weighted mean is computed as a ratio estimator:

$$\hat{R} = \frac{\sum a_i x_i}{\sum a_i}$$

Its variance is estimated as:

$$\mathrm{Var}(\hat{R}) = \frac{n}{(n-1)\,(\sum a_i)^2} \sum a_i^2 (x_i - \hat{R})^2$$

The total is $\hat{Y} = \hat{R} \cdot \sum a_i$, with $\mathrm{SE}(\hat{Y}) = \mathrm{SE}(\hat{R}) \cdot \sum a_i$.

Confidence intervals use the t-distribution with `n - 1` degrees of freedom. No finite population correction (FPC) is applied. For typical forest inventories on systematic grids, the sampling fraction n/N is very small and the FPC is negligible. Omitting it is conservative, i.e. the standard error is slightly overestimated.

Value

A tibble (one row) with:

n_plots Number of plots used

mean_ha Area-weighted mean (per hectare)

se_mean Standard error of the mean

margin_mean Margin of error for the mean (t * SE)

ci_lo_mean Lower confidence bound of the mean

ci_up_mean Upper confidence bound of the mean

total Area-weighted total (mean_ha * total area)

se_total Standard error of the total

margin_total Margin of error for the total (t * SE)

ci_lo_total Lower confidence bound of the total

ci_up_total Upper confidence bound of the total

ci_pct Margin of error as percentage of the mean

Examples

```
# Simple example with equal areas
se_area_weighted(c(10, 12, 8, 11), area_weights = c(1, 1, 1, 1))

# Unequal representation areas (e.g. from slope correction)
se_area_weighted(
  c(10, 12, 8, 11),
  area_weights = c(0.95, 1.00, 0.88, 1.02)
)
```

`se_area_weighted_grouped`

Group-Wise Area-Weighted Aggregation

Description

Convenience wrapper around `se_area_weighted` that takes a (possibly grouped) data frame with one row per plot per group, a value column, and an area column, and returns one row per group with all the columns of `se_area_weighted`.

Usage

```
se_area_weighted_grouped(data, value_col, area_col, conf_level = 0.95)
```

Arguments

`data`	A data frame. If grouped via `group_by`, the aggregation runs per group; otherwise a single row is returned.
`value_col`	Character. Name of the per-plot per-hectare value column (e.g. `"iv_m3_ha_yr_plot"`).
`area_col`	Character. Name of the plot's representation area column, in hectares.
`conf_level`	Confidence level for the CI, default 0.95.

Details

Used as the shared primitive behind several aggregation pipelines in this package (back tables, big_overview, increment summaries). The data frame is expected to be pre-aggregated to one row per plot per group — the caller decides what "per plot" means (e.g. sum over trees in a species group, or a single per-plot quantity).

Value

A tibble with one row per group, the group keys preserved, and all output columns of `se_area_weighted` attached.

Examples

```
df <- tibble::tibble(
  group = rep(c("a", "b"), each = 4),
  value = c(10, 12,  8, 11,  20, 22, 18, 21),
  area  = c( 1,  1,  1,  1,   1,  1,  1,  1)
)
df |>
  dplyr::group_by(group) |>
  se_area_weighted_grouped("value", "area")
```

`structure_table_age_class` *Species Group Information Table by Age Class and Single Tree Diameter Class*

Description

Inventory tree data are grouped by species group, age class, and single tree diameter class. A data frame with group-wise aggregated inventory information is returned.

Usage

```
structure_table_age_class(inv_trees_plus, tree_filter = !.data$removal)
```

Arguments

`inv_trees_plus`	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an `fe_inventory` object with `pull_trees` and pre-treated with `trees_add_essentials` (see example).
`tree_filter`	Expression describing which trees to use in the function, internally passed to `filter`. Default is `!removal`, i.e. trees that died or were removed are not included.

Value

A list of six data frames. *detail* holds the aggregated information by species group, age class, and single tree diameter class. *total* aggregates across single tree diameter classes, providing correct weighted `d_q_cm`/`h_q_m` for the total column of each species row. *all_species* holds the cross-species aggregation by age class and single tree diameter class (body of the "Summe" block). *all_total* aggregates *all_species* across single tree diameter classes, providing correct weighted `d_q_cm`/`h_q_m` for the total column of the Summe block. *species_total* aggregates across age classes and single tree diameter classes (one level up from *total*), i.e. the species-wise grand total used for the trailing "Summe" row `output_structure_table` appends within each species' block. *species_dbh_total* aggregates across age classes only (keeping the single tree diameter class breakdown), providing the per-dbh_class values shown in that same "Summe" row.

Attached attribute

The returned list carries a `tree_selection` attribute recording which tree cohort it represents: `"mainstand"` for the `*_main_stand()` functions, `"alllayers"` for the general functions called with their default `tree_filter`, or the raw `tree_filter` expression for a custom filter. `output_base_table` / `output_structure_table` carry this attribute through to their own output, where the `*_pdf()` renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: `base_table_age_class()`, `base_table_age_class_main_stand()`, `base_table_d_q_class()`, `base_table_d_q_class_main_stand()`, `output_base_table()`, `output_increment_overall()`, `output_structure_table()`, `structure_table_age_class_main_stand()`, `structure_table_d_q_class()`, `structure_table_d_q_class_main_stand()`

Examples

```
# The prepared tree list ships with the package; see
# ?data_ex3_trees_essentials for the chain that builds it. Three inventory
# points are enough to show the shape of the result, and they keep the
# example quick; the function takes a whole tree list alike.
data_ex3_sample_trees_essentials |>
  dplyr::filter(plot_id %in% unique(plot_id)[1:3]) |>
  structure_table_age_class()
```

---

structure_table_age_class_main_stand

Species Group Information Table by Age Class and Single Tree Diameter Class for the Main Stand Cohort

---

Description

Inventory tree data are grouped by species group, age class, and single tree diameter class. A data frame with group-wise aggregated inventory information is returned.

Usage

```
structure_table_age_class_main_stand(inv_trees_plus)
```

Arguments

`inv_trees_plus`	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an `fe_inventory` object with `pull_trees` and pre-treated with `trees_add_essentials` (see example).

Value

A list of six data frames. *detail* holds the aggregated information by species group, age class, and single tree diameter class, restricted to the main stand, including area estimates. *total* aggregates across single tree diameter classes, providing correct weighted `d_q_cm`/`h_q_m` for the total column of each species row. *all_species* holds the cross-species aggregation by age class and single tree diameter class (body of the "Summe" block). *all_total* aggregates *all_species* across single tree diameter classes, providing correct weighted `d_q_cm`/`h_q_m` for the total column of the Summe block. *species_total* aggregates across age classes and single tree diameter classes (one level up from *total*), i.e. the species-wise grand total used for the trailing "Summe" row `output_structure_table` appends within each species' block. *species_dbh_total* aggregates across age classes only (keeping the single tree diameter class breakdown), providing the per-dbh_class values shown in that same "Summe" row.

Attached attribute

The returned list carries a `tree_selection` attribute recording which tree cohort it represents: `"mainstand"` for the `*_main_stand()` functions, `"alllayers"` for the general functions called with their default `tree_filter`, or the raw `tree_filter` expression for a custom filter. `output_base_table` / `output_structure_table` carry this attribute through to their own output, where the `*_pdf()` renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: `base_table_age_class()`, `base_table_age_class_main_stand()`, `base_table_d_q_class()`, `base_table_d_q_class_main_stand()`, `output_base_table()`, `output_increment_overall()`, `output_structure_table()`, `structure_table_age_class()`, `structure_table_d_q_class()`, `structure_table_d_q_class_main_stand()`

Examples

```
# The prepared tree list ships with the package; see
# ?data_ex3_trees_essentials for the chain that builds it. Three inventory
# points are enough to show the shape of the result, and they keep the
# example quick; the function takes a whole tree list alike.
data_ex3_sample_trees_essentials |>
  dplyr::filter(plot_id %in% unique(plot_id)[1:3]) |>
  structure_table_age_class_main_stand()
```

---

structure_table_d_q_class

Species Group Information Table by Mean Diameter Class and Single Tree Diameter Class

---

Description

Inventory tree data are grouped by species group, mean diameter class, and single tree diameter class. A data frame with group-wise aggregated inventory information is returned.

Usage

```
structure_table_d_q_class(
  inv_trees_plus,
  dclass_back,
  tree_filter = !.data$removal
)
```

Arguments

inv_trees_plus	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an fe_inventory object with pull_trees and pre-treated with trees_add_essentials (see example).
dclass_back	Data frame listing quadratic mean diameter classes per species group and layer on plot level. Typically the output of back_table_dclass.
tree_filter	Expression describing which trees to use in the function, internally passed to filter. Default is !removal, i.e. trees that died or were removed are not included.

Value

A list of six data frames. *detail* holds the aggregated information by species group, mean diameter class, and single tree diameter class. *total* aggregates across single tree diameter classes, providing correct weighted d_q_cm/h_q_m for the total column of each species row. *all_species* holds the cross-species aggregation by mean diameter class and single tree diameter class (body of the "Summe" block). *all_total* aggregates *all_species* across single tree diameter classes, providing correct weighted d_q_cm/h_q_m for the total column of the Summe block. *species_total* aggregates across mean diameter classes and single tree diameter classes (one level up from *total*), i.e. the species-wise grand total used for the trailing "Summe" row output_structure_table appends within each species' block. *species_dbh_total* aggregates across mean diameter classes only (keeping the single tree diameter class breakdown), providing the per-dbh_class values shown in that same "Summe" row.

Attached attribute

The returned list carries a tree_selection attribute recording which tree cohort it represents: "mainstand" for the *_main_stand() functions, "alllayers" for the general functions called with their default tree_filter, or the raw tree_filter expression for a custom filter. output_base_table / output_structure_table carry this attribute through to their own output, where the *_pdf() renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: base_table_age_class(), base_table_age_class_main_stand(), base_table_d_q_class(), base_table_d_q_class_main_stand(), output_base_table(), output_increment_overall(), output_structure_table(), structure_table_age_class(), structure_table_age_class_main_stand(), structure_table_d_q_class_main_stand()

Examples

```
# The prepared tree list is shipped with the package: it is this inventory
# put through pull_trees() -> height_complete_inventory() ->
# fill_heights_back() -> pull_trees() -> trees_add_essentials(). See
# ?data_ex3_trees_essentials for that chain spelled out.
# Three inventory points are enough to show the shape of the result, and
# they keep the example quick; the function takes a whole tree list alike.
trees_with_heights <- data_ex3_sample_trees_essentials |>
  dplyr::filter(plot_id %in% unique(plot_id)[1:3])

peg_back_d <- back_table_dclass(trees_with_heights)

structure_table_d_q_class(trees_with_heights, dclass_back = peg_back_d)
```

---

structure_table_d_q_class_main_stand

Species Group Information Table by Mean Diameter Class and Single Tree Diameter Class

---

Description

Inventory tree data are grouped by species group, mean diameter class, and single tree diameter class. A data frame with group-wise aggregated inventory information is returned.

Usage

```
structure_table_d_q_class_main_stand(inv_trees_plus, dclass_back)
```

Arguments

inv_trees_plus	Data frame which covers trees from an inventory (each row is a tree), typically obtained from an fe_inventory object with pull_trees and pre-treated with trees_add_essentials (see example).
dclass_back	Data frame listing quadratic mean diameter classes per species group and layer on plot level. Typically the output of back_table_dclass.

Details

Very similar aggregation table to the one produced by structure_table_d_q_class, but restricted to to the main stand (layer_key == 1) only. It contains, however, estimates of areas covered by species (sub-)groups and ha-related values o these cohorts. Due the methodological dubiousness of species area calculations in mixed stands, this is only done for the main stand (comparably to how this was handled in the 3rd German National Forest Inventory).

Value

A list of six data frames. *detail* holds the aggregated information by species group, mean diameter class, and single tree diameter class, restricted to the main stand, including area estimates. *total* aggregates across single tree diameter classes, providing correct weighted `d_q_cm`/`h_q_m` for the total column of each species row. *all_species* holds the cross-species aggregation by mean diameter class and single tree diameter class (body of the "Summe" block). *all_total* aggregates *all_species* across single tree diameter classes, providing correct weighted `d_q_cm`/`h_q_m` for the total column of the Summe block. *species_total* aggregates across mean diameter classes and single tree diameter classes (one level up from *total*), i.e. the species-wise grand total used for the trailing "Summe" row `output_structure_table` appends within each species' block. *species_dbh_total* aggregates across mean diameter classes only (keeping the single tree diameter class breakdown), providing the per-dbh_class values shown in that same "Summe" row.

Attached attribute

The returned list carries a `tree_selection` attribute recording which tree cohort it represents: `"mainstand"` for the `*_main_stand()` functions, `"alllayers"` for the general functions called with their default `tree_filter`, or the raw `tree_filter` expression for a custom filter. `output_base_table` / `output_structure_table` carry this attribute through to their own output, where the `*_pdf()` renderers read it for a self-describing file name and the cohort disclaimer.

See Also

Other inventory tables: `base_table_age_class()`, `base_table_age_class_main_stand()`, `base_table_d_q_class()`, `base_table_d_q_class_main_stand()`, `output_base_table()`, `output_increment_overall()`, `output_structure_table()`, `structure_table_age_class()`, `structure_table_age_class_main_stand()`, `structure_table_d_q_class()`

Examples

```
# The prepared tree list is shipped with the package: it is this inventory
# put through pull_trees() -> height_complete_inventory() ->
# fill_heights_back() -> pull_trees() -> trees_add_essentials(). See
# ?data_ex3_trees_essentials for that chain spelled out.
# Three inventory points are enough to show the shape of the result, and
# they keep the example quick; the function takes a whole tree list alike.
trees_with_heights <- data_ex3_sample_trees_essentials |>
  dplyr::filter(plot_id %in% unique(plot_id)[1:3])

peg_back_d <- back_table_dclass(trees_with_heights)

structure_table_d_q_class_main_stand(trees_with_heights, dclass_back = peg_back_d)
```

`trees_add_essentials` *Add Essential Information Like Species Groups, DBH-Classes, Basal Areas, and Volumes to a Trees Data Frame*

Description

Function for intermediate calculations between pulling a trees data frame from an `fe_inventory` object with `pull_trees` and making aggregated evaluations and plots. Made for avoiding double calculations of tree volume, basal area, and species groups.

Usage

```
trees_add_essentials(
  inv_trees,
  method = c("WWK", "BaySF"),
  dbh_interval = 5,
  age_interval = 20
)
```

Arguments

`inv_trees`	Data frame which covers trees from an inventory (each row is a tree), obtained from an `fe_inventory` object with `pull_trees` and then **height-completed**. The column `height_m` must be present and must not contain any missing values. Note that `pull_trees` alone may legitimately return trees with missing heights (they are not filled in during import); complete them first, e.g. with `height_complete_inventory` followed by `fill_heights_back` (or the `h_est_m` fallback shown in the examples). A missing `height_m` column, or any `NA` height, is rejected with an informative error.
`method`	Character string indicating the method to be used for the species groups calculation. Default is "WWK", which is the method used for the species coding tum_wwk_short. Additionally "BaySF" is available, for classifying species according to the groups of the Bavarian State Forest Enterprise (bavrn_state_short).
`dbh_interval`	Integer indicating the interval breaks for the dbh classes to be formed. The breaks are internally handed over to `cut`as its parameter `breaks`. Default is 5-cm-classes, where diameters over 60 cm are all put into one class.
`age_interval`	Integer indicating the interval breaks for the age classes to be formed. The breaks are internally handed over to `cut` as its parameter `breaks`. Default is 20-years-classes, where ages over 160 years are all put into one class.

Value

The original data frame but with the additional columns

species_group: Translation of the original species ids into the `fe_species_tum_wwk_short`coding, which is useful for aggregated evaluations,

dbh_class: The dbh class each tree belongs to according to `dbh_breaks` as an ordered factor,

age_class: The age class each tree belongs to according to `age_breaks` as an ordered factor,

g_m2: Basal area of each tree in m²,

v_m3: Standing wood volume of each tree over bark in m³ (calculated with `v_gri`),

v_hub_m3: Harvested wood volume of each tree under bark in m³ (calculated with `v_red_harvest_ubark`).

standing_area_m2: Tree specific standing areas, calculated with the function `standing_area_gnfi3`

Examples

```
# Complete heights are necessary for trees_add_essentials. The h_q fallback
# warning is expected here (some species x layer groups lack a measured height).
trees_with_heights <- suppressWarnings(
  data_ex3_sample_fe_inventory |>
    pull_trees() |>
    height_complete_inventory(method = "Bavaria") |>
    dplyr::mutate(height_m = ifelse(!h_m_tree, h_est_m, height_m))
)

# Actual application
trees_add_essentials(trees_with_heights)
```

`tree_inc_repsurv` *Calculate the Volume Increment of the Single Trees in a Repeated Inventory with a Concentric Sampling Circle Design*

Description

Provided for convenience, but typically not called directly but from within the function `inv_increment_repsurv_ccirc`.

Usage

```
tree_inc_repsurv(
  matched_trees,
  method = c("rep_classic", "rep_mean", "rep_end", "rep_trans"),
  clamp_plausible_shrinkage = TRUE
)
```

Arguments

`matched_trees`	A data frame being the output of `match_trees_on_inventory_repsurv_ccirc`.
`method`	Character string, choices are `"rep_classic"` (default), `"rep_mean"`, `"rep_end"`, and `"rep_trans"`. These methods differ in how sampling thresholds are handled (see Details.)

clamp_plausible_shrinkage
: Logical. If TRUE (default), re-measured trees with plausible volume shrinkage (below the implausibility threshold set in match_trees_on_inventory_repsurv_ccirc) receive a zero increment. If FALSE, plausible shrinkage is passed through to the increment for rep_classic, rep_mean, and rep_end. Has no effect on rep_trans, which always clamps negative volume differences to zero (a warning is issued in that case). Trees flagged as implausible always receive a zero increment regardless of this setting.

Details

Even though the volume increments are calculated on tree level here, they are upscaled to 1 ha. This is important, because otherwise the increment represented by trees that change their representation number per ha between both inventories cannot be calculated correctly. Trees present in the first survey only always get zero increment; such values can be replaced by model-based estimates in a subsequent step. The function allows to choose between four methods of increment calculation (parameter method):

- rep_classic: The volume increment, iv, of trees that are present in both inventories is calculated as $iv = n[2] \cdot v[2] - n[1] \cdot v[1]$, with n and v being the representation number per hectare and the volume of the tree at the first and the second survey, respectively. The volume increment of the ingrowth trees is $iv = n[2] * v[2]$, i.e. such trees contribute to the increment with their full volume. This method is equivalent to the classic stand level increment calculation (German "Ertragsgeschichtlicher Zuwachs"), however, some trees (and plots) may also have negative increments. This might look strange, but is fully valid from a statistical angle of view.
- rep_mean: The volume increment, iv, of trees that are present in both inventories is calculated as $iv = (n[2]+n[1])/2 \cdot (v[2]-v[1])$, and the volume increment of ingrowth trees is calculated as $iv = n[2]*(v[2]-v_t)$, where v_t is the tree's volume when it crossed the sampling threshold. This method is not compatible to the classic stand level increment calculation. Guarantees $iv >= 0$ when clamp_plausible_shrinkage = TRUE.
- rep_end: The volume increment, iv, of trees that are present in both inventories is calculated as $iv = n[2] \cdot (v[2] - v[1])$, and the volume increment of ingrowth trees is calculated as $iv = n[2] * (v[2] - v_t)$, where v_t is the tree's volume when it crossed the sampling threshold. This method is not compatible to the classic stand level increment calculation. Guarantees $iv >= 0$ when clamp_plausible_shrinkage = TRUE.
- rep_trans: The volume increment, iv, of trees that are present in both inventories, and did not cross a sampling threshold, is calculated as $iv = n \cdot (v[2] - v[1])$, with $n = n[1] = n[2]$. For trees that crossed a sampling threshold, the calculation is $iv = n[2] \cdot (v[2]-v_t)+n[1] \cdot (v_t-v[1])$. The volume increment of ingrowth trees is calculated as $iv = n[2]*(v[2]-v_t)$, where v_t is the tree's volume when it crossed the sampling threshold. This method is not compatible to the classic stand level increment calculation and always guarantees $iv >= 0$ (negative volume differences are clamped to zero regardless of clamp_plausible_shrinkage).

Value

The input data frame with two additional columns: iv_hub_m3_ha_per_rep and iv_hub_m3_ha_yr_rep. They represent the periodic and the (mean) annual tree volume increment in m³/ha under bark and after harvest.

See Also

Other increment: harmonize_inv_for_repsurv(), increment_base_table(), increment_base_table_main_stand(), increment_ytables_base_table(), inv_inc_big_overview(), inv_inc_big_overview_2nd_only(), inv_inc_big_overview_matches_only(), inv_inc_fill_gaps_gnfi3(), inv_inc_summaric(), inv_inc_tree_2_plot(), inv_inc_tree_consolidate(), inv_inc_tree_extend(), inv_inc_tree_extend_combined(), inv_increment_gnfi3(), inv_increment_repeated_survey(), inv_increment_repsurv_ccirc(), inv_increment_ytables(), is_fe_increment_repsurv(), match_2_inventories(), match_2_inventories_by_cente, match_2_inventories_by_plot_id(), match_plot_statistics(), match_trees_on_inventory_repsurv_ccirc(), match_trees_on_plot_repsurv_ccirc(), output_increment_base_table(), output_increment_base_table_pdf(), output_increment_overall(), output_increment_overall_pdf(), output_increment_overview_gnfi3(), output_increment_overview_gnfi3_pdf(), output_increment_overview_ytables_pdf(), reclassify_pseudo_ingu

Examples

```
# When called directly, considerable preparation is required
inv_a <- data_ex3_previous_sample_fe_inventory
inv_b <- data_ex3_sample_fe_inventory

# Match inventory points
inv_pt_mtch <- match_2_inventories(inv_a, inv_b, "plot_id", "baysf")

inv_a_trees <- data_ex3_previous_sample_trees_essentials
inv_b_trees <- data_ex3_sample_trees_essentials

# Now, match both inventories on tree level
mtch_trs <- match_trees_on_inventory_repsurv_ccirc(
  inv_pt_mtch, inv_a, inv_b, inv_a_trees, inv_b_trees,
  tree_id_style = "baysf"
)

# Finally, calculate the increments
tree_inc_repsurv(mtch_trs, method = "rep_classic")
tree_inc_repsurv(mtch_trs, method = "rep_mean")
tree_inc_repsurv(mtch_trs, method = "rep_end")
tree_inc_repsurv(mtch_trs, method = "rep_trans")
```

---

validate_fe_inventory *Validate an* **fe_inventory** *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor new_fe_inventory. Regular users, please construct fe_inventory objects with fe_inventory.

Usage

```
validate_fe_inventory(x)
```

Arguments

x	an object that is expected to be a correct fe_inventory object

Value

Returns x, but this function is mainly called for its side effect which is pointing out any violations of the fe_inventory object specifications. In case of such violations, the function will terminate with an error.

Examples

```
validate_fe_inventory(data_ex4_sample_fe_inventory)
```

---

ytables_bavrn_state_var_1_feneu

Complete Bavarian-State-Forest Yield-Table Selection

Description

A yield-table selection assigning a ForestElementsR yield table to every bavrn_state tree species code, for use with inv_increment_ytables and its overview. It is FeNEU's completed counterpart to ForestElementsR::ytables_bavrn_state_var_1: that FER dataset predates the additional bavrn_state tree species codes introduced in ForestElementsR 3.0.0 and does not cover them, so inv_increment_ytables aborts on data containing such species (e.g. recent Bavarian State Forest inventories). This selection covers all current bavrn_state tree species codes and is therefore the recommended default for the bavrn_state coding.

Usage

```
ytables_bavrn_state_var_1_feneu
```

Format

A tibble with one row per species, and two columns:

species_id An fe_species_bavrn_state vector (ForestElementsR species coding "bavrn_state") identifying the tree species.

ytable_name (character) The name of the ForestElementsR yield-table object assigned to that species (e.g. "fe_ytable_spruce_gehrhardt_moderate_1921").

Details

The 20 species not present in the FER selection are assigned a surrogate yield table by genus / species-group analogy to an already-assigned congener (e.g. the additional oaks use the oak table, the elms the ash table). Three assignments are silvicultural judgement calls: the cedars (Atlaszeder, Libanonzeder) use the larch table, because the Bavarian State Forest counts them in the larch species group; the Baumhasel uses the birch table, as a minor broadleaf ("sonstiges Laubholz"). The remaining 45 species keep exactly the assignments of ForestElementsR::ytables_bavrn_state_var_1.

Source

Extends `ForestElementsR::ytables_bavrn_state_var_1` with surrogate assignments for the tree species codes it does not cover.

See Also

`inv_increment_ytables`, `ForestElementsR::ytables_bavrn_state_var_1`

Index