Package 'basifoR'

May 7, 2026

Type Package

Title Retrieval and Processing of the Spanish National Forest Inventory

Version 0.7.7

Maintainer Wilson Lara <wilarhen@gmail.com>

Description Fetches, harmonizes, and analyses data from the Spanish National Forest Inventory for reproducible, design-aware forest inventory workflows. Computes tree- and stand-level metrics, applies sampling-based expansion factors, estimates volume, and supports extensible processing for external inventory designs with custom sampling schemes and volume equations.

URL https://www.miteco.gob.es/es/biodiversidad/temas/inventarios-nacionales/inventario-forestal-nacional.html

License GPL-3

Depends R (>= 4.4.0)

Suggests odbc

Imports curl, foreign, Hmisc, httr, measurements, RODBC, rvest, stats, utils

SystemRequirements Microsoft Access driver on Windows for Access backends; mdbtools on Unix-like systems

Encoding UTF-8

NeedsCompilation no

Author Wilson Lara [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Repository CRAN

Date/Publication 2026-04-09 21:00:02 UTC

Contents

dbhMetric . . . 2
default_external_volume_methods . . . 4
default_snfi_volume_equations . . . 5
dendroMetrics . . . 5
externalMetrics . . . 7
externalMetrics2Vol . . . 9
external_dendroMetrics . . . 13
external_volume_method_registry . . . 17
fetchNFI . . . 18
getNFI . . . 19
get_snfi_iavc . . . 21
get_snfi_vcc . . . 22
get_snfi_vle . . . 23
get_snfi_vsc . . . 24
inventoryMetrics . . . 25
metrics2Vol . . . 28
new_concentric_design . . . 30
new_external_schema . . . 31
new_inventory_design . . . 33
new_volume_method . . . 34
nfiMetrics . . . 37
print.concentric_design . . . 38
print.external_schema . . . 40
print.inventory_design . . . 40
readNFI . . . 41
snfi_design . . . 43
snfi_volume_method_registry . . . 44
trees_per_ha . . . 46
trees_per_ha.concentric_design . . . 47
trees_per_ha.inventory_design . . . 48
update.dendroMetrics . . . 49
update.external_dendroMetrics . . . 50
update.inventoryMetrics . . . 51
update.list . . . 52

Index **53**

`dbhMetric` *Compute diameter, basal area, tree density, or height from tree measurements*

Description

Convert raw tree diameter or height inputs into the compact metric formats used across basifoR workflows.

Usage

```
dbhMetric(dbh, met = "d",
    design = snfi_design())
```

Arguments

dbh	numeric. Diameter at breast height in mm, or tree height in m when met = "h". Non-numeric inputs are coerced, zeros are treated as missing, and vectors are averaged after that replacement.
met	character(1). Metric to compute: mean diameter at breast height ("d"), basal area ("ba"), trees per hectare ("n"), or height ("h").
design	Object inheriting from "inventory_design", used only when met = "n" to derive the trees-per-hectare expansion factor. The default is the Spanish National Forest Inventory concentric subplot design.

Details

The sampling design affects only met = "n". For "d", "ba", and "h", the returned value does not depend on design.

Value

A single numeric value. Returns mean diameter in mm for met = "d", basal area in m^2 per tree for met = "ba", trees per hectare for met = "n", and height in dm for met = "h". Returns NA_real_ when all supplied values are missing or become missing after zero replacement.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
dbhMetric(300, "d")
dbhMetric(300, "ba")
dbhMetric(18, "h")

dsg <- new_concentric_design(
    radii_m = c(4, 8, 12),
    min_dbh_cm = c(5, 15, 30),
    name = "3-subplot design"
)

dbhMetric(130, "n", design = dsg)
```

`default_external_volume_methods` *Return bundled external volume methods*

Description

Return the built-in external volume-method specifications used as the starting registry for `externalMetrics2Vol` and `external_volume_method_registry`.

Usage

```
default_external_volume_methods()
```

Details

`"V"` passes through an already available total-volume field. `"VCC"` computes merchantable volume over bark from diameter and height, and `"VSC"` computes merchantable volume under bark from the resolved `vcc` value.

Value

A named list of default external volume-method specifications. The list currently contains entries named `"V"`, `"VCC"`, and `"VSC"`, each stored as the named list returned by `new_volume_method`.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
methods <- default_external_volume_methods()
names(methods)
methods$V$output
methods$VCC$required_inputs
```

default_snfi_volume_equations *Default SNFI volume-equation methods*

Description

Return the default SNFI volume-equation registry. Creates the default set of method definitions used by ‘metrics2Vol()‘ to compute tree-level volume variables. Each registry entry describes how one output is computed, including the target column name, equation function, unit conversion, argument builder, and fallback rule.

Usage

```
default_snfi_volume_equations()
```

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

dendroMetrics *Summarize dendrometrics*

Description

This function summarizes dendrometric data from the Spanish National Forest Inventory (SNF). It primarily accepts a province name or number, a local compressed SNF file, or a URL to a compressed SNF file hosted by `www.miteco.gob.es`. It can also process data frames previously returned by `readNFI`, `nfiMetrics`, or `metrics2Vol`. Dendrometric variables in the output are transformed into stand units, see Details section.

Usage

```
dendroMetrics(nfi, summ.vr = "Estadillo",
    metric_levels = NULL,
    cut.dt = "d == d",
    report = FALSE, mc.cores = getOption("mc.cores",
        1L), ...)
```

Arguments

nfi	character, data.frame, or list. A province name or province number used to locate SNF data; a local path or URL to a compressed SNF file (.zip), including ZIP files hosted by www.miteco.gob.es; a data frame such as that returned by readNFI, nfiMetrics, or metrics2Vol; or a list of such objects.
summ.vr	character or NULL. Name of a Categorical variables in the SNF data used to summarize the outputs. If NULL then output from metrics2Vol is returned. Default 'Estadillo' processes sample plots.
metric_levels	character or NULL. Grouping variables used to compute tree-level metrics such as Hd before final summarization. When NULL, metric computation follows summ.vr.
cut.dt	character. Logical condition used to subset the output. Default 'd == d' avoids subsetting.
report	logical. Print a report of the output in the current working directory.
mc.cores	integer. Number of cores used when several inputs are processed.
...	Additional arguments passed to readNFI, nfiMetrics, or metrics2Vol, including nfi.nr when required.

Details

Dendrometric variables are summarized according to the levels of argument summ.vr. Summary outputs include the categorical columns defined by summ.vr together with the quantitative variables available after processing with nfiMetrics and metrics2Vol.

These variables may include tree basal area ba ('m2 ha-1'), mean diameter at breast height d ('cm'), quadratic mean diameter dg ('cm'), mean tree height h ('m'), number of trees per hectare n ('dimensionless'), and over-bark volume v ('m3 ha-1').

When summ.vr = NULL, the function returns tree-level outputs from metrics2Vol after applying the filter defined in cut.dt.

When summ.vr is not NULL, the function converts supported variables to stand units, splits the data by the requested grouping variable, and computes summaries by group. Extensive variables are multiplied by n before summation. Variables d, h, and Hd, when present, are returned as averages weighted by n.

If both ba and n are available, the function also derives the quadratic mean diameter dg.

Output subsets are extracted using the logical expression supplied in cut.dt, see syntax in Logic.

The function accepts one input object or several inputs. When several inputs are supplied, each one is processed independently and the results are merged into a single output data frame. Parallel processing is controlled by mc.cores. Values greater than 1 process several inputs in parallel.

Value

data.frame. Depending on summ.vr = NULL, an output from metrics2Vol, or a summary of the variables, see Details section.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
## Minimal precomputed metrics object with units
toy_metrics <- structure(
    data.frame(
        Estadillo = c("plot1", "plot1", "plot2"),
        Especie   = c("sp1", "sp2", "sp1"),
        d         = c(120, 185, 260),          # mm
        h         = c(71, 94, 132),            # dm
        ba        = c(0.0113, 0.0269, 0.0531), # m2 tree-1
        n         = c(127.32, 31.83, 14.15),
        stringsAsFactors = FALSE
    ),
    class = c("nfiMetrics", "data.frame"),
    units = c(
        d  = "mm",
        h  = "dm",
        ba = "m2 tree-1",
        n  = "ha-1"
    ),
    nfi.nr = 4
)

## Summarize by plot
dendromet_toy <- dendroMetrics(toy_metrics, summ.vr = "Estadillo")

## Display output structure
str(dendromet_toy)

## Check returned units
attr(dendromet_toy, "units")

## Return tree-level processed data
tree_toy <- dendroMetrics(toy_metrics, summ.vr = NULL, cut.dt = "h > 8")
head(tree_toy)

## Alternatively, download data from 'www.miteco.gob.es'
## Specify province name/number and nfi number to compute dendrometrics.
```

`externalMetrics` *Compute tree-level metrics from external inventory data*

Description

Standardize external tree measurements for external inventory workflows and return requested tree-level metrics in basifoR units.

Usage

```
externalMetrics(x, var = c("d",
    "h", "ba", "n", "Hd"),
    levels = NULL, design,
    colmap = NULL, d_unit = c("mm",
        "cm")[1], h_unit = c("m",
        "dm", "cm")[1],
    keep_cols = NULL,
    domheight_fun = NULL)
```

Arguments

x	Input data.frame with one row per tree or stem.
var	Supported values are "d", "h", "ba", Requested metrics to return. "n", and "Hd". Requesting "Hd" also Requested metrics to return. requires "d", "h", and "n". Requested metrics to return.
levels	Grouping columns to keep in the output.
design	Inventory design used to compute expansion factors for n.
colmap	Named list of candidate raw column names for diameter and height. Matching is case-insensitive and also accepts numeric suffixes such as diameter_1 or height.2.
d_unit	Unit of raw diameter columns in colmap$d.
h_unit	Unit of raw height columns in colmap$h.
keep_cols	Additional source columns to carry into the result.
domheight_fun	Function used when "Hd" is requested.

Details

The function first resolves measurement columns from colmap. Exact matches are preferred, then case-insensitive matches with optional numeric suffixes are considered. When several repeated measurement columns are found for the same variable, row-wise non-missing means are used.

Zero values in resolved diameter or height columns are treated as missing before aggregation. Returned units are standardized to millimetres for d, decimetres for h and Hd, square metres per tree for ba, and trees per hectare for n.

When var includes "n", the function uses design to obtain expansion factors. Fixed-area designs call trees_per_ha(), while concentric designs choose the proper factor from the design thresholds. When var includes "Hd", dominant height is computed within each resolved group defined by levels and keep_cols.

Value

A data.frame containing the requested tree-level metrics, optionally preceded by resolved grouping columns.

The returned object inherits from classes "externalMetrics" and "nfiMetrics". Unit metadata are stored in attr(out, "units"). When var includes "n" or "Hd", the result also stores sampling design metadata in attr(out, "design_meta").

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
sq_0.1ha <- new_inventory_design(
    sample_area_m2 = 1000,
    min_dbh_cm = 7.5,
    name = "Square 0.1-ha plot",
    metadata = list(shape = "square", side_m = sqrt(1000))
)

x <- data.frame(
    plot = c("P1", "P1", "P2"),
    species = c("sp1", "sp1", "sp2"),
    diameter_mm = c(120, 185, 260),
    height_m = c(7.1, 9.4, 13.2),
    stringsAsFactors = FALSE
)

externalMetrics(
    x = x,
    var = c("d", "h", "ba", "n"),
    levels = c("plot", "species"),
    design = sq_0.1ha,
    colmap = list(d = "diameter_mm", h = "height_m"),
    d_unit = "mm",
    h_unit = "m"
)
```

---

externalMetrics2Vol *Compute tree-level volume outputs from external inventory data*

Description

Compute one or more tree-level volume outputs from external inventory data or from a precomputed external metrics table.

Usage

```
externalMetrics2Vol(x,
    parametro = c("V"),
    parameter_table = NULL,
    method_registry = external_volume_method_registry(),
    colmap = NULL, selector = c("first",
        "priority")[1],
    track_provenance = FALSE,
    compute_metrics_if_needed = TRUE,
    design = NULL, var = NULL,
    metric_var = NULL,
    levels = NULL, metric_levels = NULL,
    keep_cols = NULL,
    metric_keep_cols = NULL,
    metric_colmap = list(d = c("d",
        "dbh", "diameter",
        "diameter_mm"),
        h = c("h", "height",
            "height_m")),
    d_unit = NULL, metric_d_unit = c("mm",
        "cm")[1], h_unit = NULL,
    metric_h_unit = c("m",
        "dm", "cm")[1],
    volume_colmap = list(d = c("d"),
        h = c("h"), dnm = c("dnm",
            "d_nm", "D.n.m."),
        v = c("v"), species = c("species",
            "spec", "especie"),
        region = c("region",
            "pr"), equation_set = c("equation_set",
            "eqset",
            "tariff",
            "model_set")),
    ...)
```

Arguments

x	Standardized columns such as d, h, dnm, or Input data.frame or object inheriting from "externalMetrics". v must have named unit metadata in the "units" attribute of x. Input data.frame or object inheriting from "externalMetrics".
parametro	Requested volume outputs or method names.
parameter_table	Optional data.frame of coefficients or parameter rows used by the selected methods. Rows may be filtered by columns such as parameter, parametro, or method.
method_registry	Named list of method definitions, usually created with external_volume_method_registry() and new_volume_method().

colmap	Optional alias list used to resolve contextual inputs for the volume methods. If NULL, volume_colmap is used.
selector	Rule used when several parameter rows remain after filtering.
track_provenance	If TRUE, append provenance columns and store volume_meta.
compute_metrics_if_needed	If TRUE, derive missing standardized inputs with externalMetrics().
design	Inventory design passed to externalMetrics() when metric derivation is needed.
var	Legacy alias for metric_var.
metric_var	Metric variables requested during automatic derivation.
levels	Legacy alias for metric_levels.
metric_levels	Grouping or identifier columns kept during metric derivation.
keep_cols	Legacy alias for metric_keep_cols.
metric_keep_cols	Extra columns kept during metric derivation.
metric_colmap	metric_colmap
d_unit	Legacy alias for metric_d_unit.
metric_d_unit	Unit of raw diameter columns used during metric derivation.
h_unit	Legacy alias for metric_h_unit.
metric_h_unit	Unit of raw height columns used during metric derivation.
volume_colmap	volume_colmap
...	Additional arguments passed only to externalMetrics() when metric derivation is triggered.

Details

Required inputs are inferred from the selected methods in method_registry. When one or more standardized inputs are missing and compute_metrics_if_needed = TRUE, the function calls externalMetrics() to derive them, using design, metric_colmap, units, grouping columns, and retained columns from the corresponding arguments.

Each requested output is evaluated with its method definition. A method can resolve its own parameter rows with get_pars, use embedded parameters in pars, or fall back to parameter_table. Raw outputs are converted to cubic metres with scale_to_m3.

When track_provenance = TRUE, the result stores per-output provenance columns together with a volume_meta attribute describing input units, returned units, required inputs, and method settings.

Value

A data.frame containing the original columns in x plus the requested volume outputs.

The returned object inherits from classes "externalMetrics2Vol" and "metrics2vol". Named unit metadata are stored in attr(out, "units"), and the result may also preserve design_meta and volume_meta. Use exact alias matching only here. This avoids false positives such as matching the requested standardized metric "n" to source columns like "CAMPAGNE" during the preflight check that decides whether metrics need to be computed.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
x <- data.frame(
    species = c("sp1", "sp2"),
    d = c(120, 260),
    h = c(7.1, 13.2)
)
attr(x, "units") <- c(d = "mm", h = "m")

pars <- data.frame(
    parameter = "V",
    species = c("sp1", "sp2"),
    a = c(0.00002, 0.00003),
    b = c(2.30, 2.10),
    model = c("demo_sp1", "demo_sp2")
)

ext_methods <- external_volume_method_registry(list(
    V = new_volume_method(
        output = "v",
        fun = function(dbh_mm, h_m, pars) {
            dbh_cm <- dbh_mm / 10
            pars$a + pars$b * (dbh_cm^2) * h_m
        },
        raw_unit = "cm3",
        unit = "m3",
        scale_to_m3 = 1 / 1e6,
        build_args = function(ctx, pars, resolved) {
            list(dbh_mm = ctx$d_mm, h_m = ctx$h_m, pars = pars)
        },
        fallback = function(ctx, pars, resolved) NA_real_,
        match_by = "species",
        required_inputs = c("d", "h")
    )
))

out <- externalMetrics2Vol(
    x,
    parametro = "V",
    parameter_table = pars,
    method_registry = ext_methods,
    track_provenance = TRUE
)

out[, c("species", "v", "v_source", "v_status")]
```

`external_dendroMetrics` *Summarize external inventory tree data and optional volume outputs*

Description

Process external tree data through a unified workflow that standardizes measurements, computes missing dendrometric variables, optionally derives volume outputs, and returns either tree-level records or grouped stand-level summaries. The function can work from already standardized inputs or from raw external tables, and it supports repeated processing of several tables with optional parallel execution.

Usage

```
external_dendroMetrics(x,
    summ.vr = NULL, cut.dt = "d == d",
    report = FALSE, mc.cores = getOption("mc.cores",
        1L), var = c("d",
        "h", "ba", "n",
        "Hd"), parametro = NULL,
    design = NULL, parameter_table = NULL,
    method_registry = external_volume_method_registry(),
    levels = NULL, metric_levels = NULL,
    keep_cols = NULL,
    metric_keep_cols = NULL,
    colmap = NULL, metric_colmap = list(d = c("d",
        "dbh", "diameter",
        "diameter_mm"),
        h = c("h", "height",
            "height_m")),
    d_unit = NULL, metric_d_unit = c("mm",
        "cm")[1], h_unit = NULL,
    metric_h_unit = c("m",
        "dm", "cm")[1],
    tree_d_unit_out = NULL,
    tree_h_unit_out = NULL,
    volume_colmap = list(d = c("d"),
        h = c("h"), dnm = c("dnm",
            "d_nm", "D.n.m."),
        v = c("v"), species = c("species",
            "spec", "especie"),
        region = c("region",
            "pr"), equation_set = c("equation_set",
            "eqset",
            "tariff",
            "model_set")),
    selector = c("first",
```

```
    "priority")[1],
  track_provenance = FALSE,
  compute_metrics_if_needed = TRUE,
  schema = NULL, domheight_fun = null_or(get0("domheight_strict",
    mode = "function",
    inherits = TRUE),
    get0("domheight",
        mode = "function",
        inherits = TRUE)),
  ...)
```

Arguments

x	External input table, processed table, or list of such objects. Raw inputs must contain enough columns to resolve the requested metrics and, if needed, the volume selectors.
summ.vr	Grouping variable or character vector of grouping variables. Leave NULL to keep tree-level output; supply one or more column names to obtain grouped summaries.
cut.dt	Character filter evaluated on the final table. The expression is evaluated with eval(parse(\code{...})) after metrics or summaries have been computed.
report	Whether to write the returned table to 'report.csv'. The file is written in the working directory only when report = TRUE.
mc.cores	Number of worker processes used when x is a list. Values smaller than 1 are reset to 1.
var	Requested variables. Typical metric requests are "d", "h", "ba", "n", and "Hd"; volume-like names can also trigger volume processing.
parametro	Optional volume-method codes, for example "V". If NULL, the function tries to infer required methods from var and method_registry.
design	Inventory design used when missing metrics must be computed. Supply an object inheriting from "inventory_design" when the function must derive metrics such as n from raw external data.
parameter_table	Optional parameter table for the volume stage. Passed to externalMetrics2Vol() for method-specific coefficients or selection metadata.
method_registry	Volume method registry. Usually created by external_volume_method_registry() or a custom registry following the same structure.
levels	Grouping variables forwarded to schema resolution. When schema is not used, these serve as defaults for workflow grouping metadata.
metric_levels	Grouping variables used during internal metric computation. These are forwarded specifically to externalMetrics() when missing metrics are derived internally.
keep_cols	Source columns to preserve in the output or schema defaults. These columns are retained in the returned data whenever possible.

`metric_keep_cols`
: Columns to preserve during internal metric computation. These are passed to `externalMetrics()` when metrics are derived internally.

`colmap`
: Optional aliases that update the default mappings. Use this when source names differ from the expected volume or metric names.

`metric_colmap`
: metric_colmap

`d_unit`
: Optional diameter unit override shared with `schema` resolution. Accepted values are `"mm"` and `"cm"`.

`metric_d_unit`
: Diameter unit expected by `metric_colmap`. Used when metrics must be computed internally.

`h_unit`
: Optional height unit override shared with `schema` resolution. Accepted values are `"m"`, `"dm"`, and `"cm"`.

`metric_h_unit`
: Height unit expected by `metric_colmap`. Used when metrics must be computed internally.

`tree_d_unit_out`
: Optional output unit for tree-level diameter values. Only used when `summ.vr = NULL`. Accepted values are `"mm"` and `"cm"`.

`tree_h_unit_out`
: Optional output unit for tree-level height values. Only used when `summ.vr = NULL`. Accepted values are `"m"`, `"dm"`, and `"cm"`.

`volume_colmap`
: volume_colmap

`selector`
: Rule used by `externalMetrics2Vol()` when several matches remain. `"first"` keeps the first surviving row; `"priority"` uses the highest numeric `priority` value when available.

`track_provenance`
: Whether to keep provenance columns from the volume workflow. When `TRUE`, provenance columns are retained until the final summary stage removes them from grouped output.

`compute_metrics_if_needed`
: Whether to compute missing metrics from raw inputs. If `FALSE`, the function stops when required standardized columns are absent.

`schema`
: Optional `"external_schema"` object created by `new_external_schema()`. It centralizes column aliases, units, grouping defaults, and kept columns for repeated workflows.

`domheight_fun`
: domheight_fun

`...`
: Additional arguments passed to downstream helpers. These are mainly useful for custom options in the internal metric or volume-processing steps.

Details

This wrapper mirrors the arguments and behaviour of `dendroMetrics()`.

When `summ.vr = NULL`, the function returns tree-level records after applying `cut.dt`. When `summ.vr` contains one or more grouping variables, the function aggregates by those groups, reporting weighted means for `d`, `h`, and `Hd`, arithmetic sums for `ba`, `n`, and volume variables, and quadratic mean diameter `dg` when diameter and trees-per-hectare are available.

The function can work from already standardized inputs or from raw external tables. If requested metrics are missing and compute_metrics_if_needed = TRUE, it calls externalMetrics() internally, so raw inputs usually need a valid design plus diameter and, when relevant, height mappings. When schema is supplied, it provides reusable defaults for column aliases, units, grouping variables, and retained columns, while explicit arguments supplied in the call override those defaults.

Volume outputs are optional. They are computed only when parametro is supplied explicitly or can be inferred from var and method_registry. This lets the same entry point handle metric-only workflows, mixed metric-plus-volume workflows, and repeated processing of several input tables with optional parallel execution through mc.cores.

Value

A data.frame with class c("external_dendroMetrics", "dendroMetrics", \code{...}).

With summ.vr = NULL, the returned rows represent tree-level records, optionally converted to tree_d_unit_out and tree_h_unit_out. With summ.vr supplied, the returned rows represent grouped summaries and include standardized summary units such as "cm", "m", "m2 ha-1", "ha-1", and "m3 ha-1" when those variables are present.

The returned object stores the matched call in attr(out, "call"). When available, it also preserves "units", "design_meta", and "volume_meta" attributes from upstream processing, which makes the result suitable for downstream inspection and update methods.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
sq_0.1ha <- new_inventory_design(
    sample_area_m2 = 1000,
    min_dbh_cm = 7.5,
    name = "Square 0.1-ha plot",
    metadata = list(shape = "square", side_m = sqrt(1000))
)

x <- data.frame(
    plot = c("P1", "P1", "P2"),
    species = c("sp1", "sp1", "sp2"),
    diameter_mm = c(120, 185, 260),
    height_m = c(7.1, 9.4, 13.2),
    stringsAsFactors = FALSE
)

external_dendroMetrics(
    x = x,
    summ.vr = "plot",
    var = c("d", "h", "ba", "n"),
```

```
    design = sq_0.1ha,
    metric_colmap = list(d = "diameter_mm", h = "height_m"),
    metric_d_unit = "mm",
    metric_h_unit = "m"
)
```

`external_volume_method_registry` *Build the active registry of external volume methods*

Description

Return the registry of external volume `methods` used by `externalMetrics2Vol()` and related external-inventory workflows. The function starts from the package defaults, optionally uses a user-supplied registry, then applies named overrides stored in option `"basifoR.external_volume_methods"`.

Usage

```
external_volume_method_registry(methods = get0("external_volume_methods",
    inherits = TRUE,
    ifnotfound = NULL))
```

Arguments

`methods`	methods

Details

The function only assembles and validates the registry. It does not

evaluate volume equations by itself.

Value

A named `list` of external volume-method definitions. The list

always includes the package defaults, updated by any entries supplied

in `methods` and then by any entries found in option

`"basifoR.external_volume_methods"`. Each element is expected

to be compatible with `new_volume_method()`. examples« reg <- external_volume_method_registry() examples« names(reg) examples« examples« custom <- list(examples« V = new_volume_method(examples« output = "v", examples« unit = "m3", examples« raw_unit = "m3", examples« scale_to_m3 = 1, examples« build_args = function(ctx, pars, resolved) list(), examples« fallback = function(ctx, pars, resolved) 0, examples« match_by = character(0), examples« required_inputs = "v" examples«) examples«) examples« reg2 <- external_volume_method_registry(custom) examples« reg2Voutput

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

`fetchNFI` *Fetch SNF Data*

Description

This function can download and decompress data sets from the Spanish National Forest Inventory (SNF) stored in `.zip` files, whether they are local or remote (URL-based).

Usage

```
fetchNFI(url., dir = tempdir(),
    file_ext = c("mdb",
        "DBF", "accdb"),
    file_name = NULL,
    timeOut = timeout(60))
```

Arguments

`url.`	`character`. Specifies the URL/path to a compressed SNF (.zip).
`dir`	`character`. Directory where the fetched file will be stored.
`file_ext`	`character`. Supported file extensions.
`file_name`	`character`. Optional file names inside the decompressed content to keep. It accepts either full file names or bare stems without extensions.
`timeOut`	`request`. Maximum request time, see `timeout`. Default is `timeout(60)`.

Details

The data should be in files with extensions specified in the `file_ext` argument. Use `file_name` to restrict the returned files to specific names inside the decompressed content. It accepts either full file names or bare stems without extensions.

Value

`character`. Returns the path to the fetched and decompressed NFI data (.mdb, .DBF, .accdb, or other extensions requested in `file_ext`) stored in a temporary file.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
## Process SNF data for Toledo stored locally
# Path to Toledo data file in 'basifoR' package
ifn4p45 <- system.file("Ifn4_Toledo.zip", package="basifoR")

# Decompress SNF data from the specified file path or URL
fetch_ifn4p45 <- fetchNFI(ifn4p45)
print(fetch_ifn4p45)

## French NFI tree table read from the official web resource
## f <- "https://inventaire-forestier.ign.fr/dataifn/data/export_dataifn_2024_en.zip"
## fnfi <- fetchNFI(f, file_ext = "csv", file_name = "ARBRE")
## print(fnfi)
```

---

getNFI *Read raw SNFI tables (deprecated; use* `readNFI()`*)*

Description

Deprecated wrapper around `readNFI`. `getNFI()` is kept for backward compatibility, but it now delegates internally to `readNFI()` and returns imported tables instead of acting as a file-fetching helper.

Usage

```
getNFI(provincia, ...)
```

Arguments

`provincia`	Input accepted by `readNFI`. This can be a province identifier, a local or remote `.zip` archive, one or more decompressed inventory file paths, or a data frame already loaded in memory.
`...`	Additional arguments passed to `readNFI`.

Details

Deprecated. Use `readNFI` in new code. This compatibility wrapper preserves the historical function name while redirecting all supported inputs to `readNFI()`, including province identifiers, local or remote `.zip` archives, direct paths to decompressed inventory files, and data frames already loaded in memory.

Because `getNFI()` now calls `readNFI()` internally, its behavior follows `readNFI()` semantics. In particular, the function returns imported data rather than extracted file paths. Users who still need archive extraction without import should call `fetchNFI` directly.

The argument names are kept for compatibility with older code,

but users should migrate to `readNFI`.

A deprecation warning is emitted on each call to make that transition explicit.

Value

The same return value as `readNFI`. Depending on the input and selected files, this is typically a data frame or a named list of data frames containing imported SNFI tables.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
## Minimal self-contained example using a temporary CSV file
tmp <- tempfile(fileext = ".csv")

utils::write.table(
    data.frame(
        plot = 1:2,
        species = c("sp1", "sp2"),
        dbh_cm = c(12.5, 18.0)
    ),
    file = tmp,
    sep = ";",
    row.names = FALSE,
    quote = FALSE
)

x <- suppressWarnings(getNFI(tmp))
str(x)

unlink(tmp)
```

`get_snfi_iavc` *SNFI annual increment of volume over bark*

Description

Compute SNFI IAVC using the variables required by the selected model.

Usage

```
get_snfi_iavc(dbh_mm = NULL,
    dnm_mm = NULL, h_t = NULL,
    vcc = NULL, pars)
```

Arguments

`dbh_mm`	Tree diameter at breast height in millimetres, when required by the model.
`dnm_mm`	Mean plot diameter in millimetres, when required by the model.
`h_t`	Tree height in metres, when required by the model.
`vcc`	Merchantable volume over bark in dm3, when required by the model.
`pars`	One-row `data.frame` of SNFI coefficients. The function « assumes exactly one row with a `Modelo` field and the « coefficients required by that model.

Details

This function evaluates the SNFI equation for annual increment

of volume over bark (IAVC) and returns the result in dm3.

It supports models 8, 13, 14, 16, 17, 18, 19, 20, 21, and 25.

The required inputs depend on the selected model.

Value

Numeric IAVC in dm3. Returns `NA_real_` when `pars`

is empty, when required inputs are missing, or when the model

is not recognised. Missing required inputs also trigger a

printed message. print("Model not recognized for IAVC.")

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

`get_snfi_vcc` *SNFI merchantable volume over bark*

Description

Compute SNFI VCC from tree diameter, height, and one coefficient row.

Usage

```
get_snfi_vcc(dbh_mm,
    h_t, pars)
```

Arguments

`dbh_mm`	Tree diameter at breast height in millimetres.
`h_t`	Tree height in metres.
`pars`	One-row `data.frame` of SNFI coefficients. The function « assumes exactly one row with a `Modelo` field and the « coefficients required by that model.

Details

This function evaluates the SNFI equation for merchantable

volume over bark (VCC) and returns the result in dm3.

It currently supports models 1 and 11 from the SNFI

coefficient table.

Value

Numeric VCC in dm3. Returns `NA_real_` when `pars`

is empty or when the model is not recognised. print("Model not recognized for VCC.")

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

get_snfi_vle *SNFI coarse woody volume*

Description

Compute SNFI VLE from tree diameter or VCC and one coefficient row.

Usage

```
get_snfi_vle(dbh_mm = NULL,
    vcc = NULL, pars)
```

Arguments

dbh_mm	Tree diameter at breast height in millimetres, when required by the model.
vcc	Merchantable volume over bark in dm3, when required by the model.
pars	One-row data.frame of SNFI coefficients. The function « assumes exactly one row with a Modelo field and the « coefficients required by that model.

Details

This function evaluates the SNFI equation for coarse woody

volume (VLE) and returns the result in dm3.

It supports models 10 and 12.

Value

Numeric VLE in dm3. Returns NA_real_ when pars

is empty, when required inputs are missing, or when the model

is not recognised. Missing required inputs also trigger a

printed message. print("Model not recognized for VLE.")

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

`get_snfi_vsc` *SNFI merchantable volume under bark*

Description

Compute SNFI VSC from VCC and one coefficient row.

Usage

```
get_snfi_vsc(vcc, pars)
```

Arguments

`vcc`	Merchantable volume over bark in dm3.
`pars`	One-row `data.frame` of SNFI coefficients. The function « assumes exactly one row with a `Modelo` field and the « coefficients required by that model.

Details

This function evaluates the SNFI equation for merchantable

volume under bark (VSC) and returns the result in dm3.

It currently supports model 7.

Value

Numeric VSC in dm3. Returns `NA_real_` when `pars`

is empty or when the model is not recognised. print("Model not recognized for VSC.")

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

`inventoryMetrics` *Dispatch complete workflows for SNFI and external inventories Unified dispatcher for inventory workflows*

Description

Run a complete inventory workflow by dispatching to either `dendroMetrics` for Spanish National Forest Inventory inputs or to `external_dendroMetrics()` for external inventories. The function is intentionally thin: it decides which backend to use, forwards the relevant arguments, and stores the original call so the result can be updated later with `update`.

Usage

```
inventoryMetrics(nfi,
    backend = c("auto",
        "snfi", "external"),
    summ.vr = "Estadillo",
    cut.dt = "d == d",
    report = FALSE, mc.cores = getOption("mc.cores",
        1L), design = NULL,
    schema = NULL, method_registry = NULL,
    parameter_table = NULL,
    ...)
```

Arguments

`nfi`	`character`, `data.frame`, or `list`. One « inventory input or several inputs accepted by the selected « backend.
`backend`	`character(1)`. « Backend selector. « Use `"auto"` « to infer the backend « from the supplied « arguments.
`summ.vr`	`character` or `NULL`. Grouping « variable passed to the selected backend. « When this argument is not supplied, « `inventoryMetrics()` uses a backend- « aware default: `"Estadillo"` for « backend = `"snfi"` and the backend « default for backend = `"external"`.
`cut.dt`	`character`. Logical expression used « to subset the backend output.
`report`	`logical`. If `TRUE`, request a « CSV `report` from the selected backend.
`mc.cores`	`integer`. Number « of cores used when the « backend supports several « inputs.
`design`	Optional sampling `design`. Passed to the « selected backend when relevant.
`schema`	Optional external `schema` object. Used only by « backend = `"external"`.
`method_registry`	Optional method registry passed to the « selected backend. This may define « custom volume or summarization methods.
`parameter_table`	Optional table of parameters required « by the external backend.
`...`	Additional named arguments forwarded to the selected « backend.

Details

The dispatcher keeps computation inside the backend functions. It only validates the backend choice, forwards arguments, stores the reconstructed call in `attr(x, "call")`, and records the chosen backend in `attr(x, "backend")`. That makes the wrapper easier to maintain while preserving a single public entry point.

Value

A backend result augmented with class `"inventoryMetrics"`, a stored call in `attr(x, "call")`, and the selected backend in `attr(x, "backend")`. The returned object otherwise preserves the class and attributes produced by the backend.

Note

Keep backend-specific computation in the backend functions, not in this dispatcher.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

See Also

dendroMetrics, nfiMetrics, metrics2Vol, update.inventoryMetrics

Examples

```
    ## External workflows require an external backend function and the
    ## corresponding schema, design, and parameter objects.

ext <- data.frame(
  plot = c("P1", "P1", "P2"),
  species = c("sp1", "sp1", "sp2"),
  diameter_mm = c(120, 185, 260),
  height_m = c(7.1, 9.4, 13.2),
  stringsAsFactors = FALSE
)

sch <- new_external_schema(
```

```
  colmap = list(
    plot = "plot",
    species = "species",
    d = "diameter_mm",
    h = "height_m"
  ),
  units = list(d = "mm", h = "m"),
  levels = "plot",
  keep_cols = c("plot", "species")
)

dsg <- new_inventory_design(
  sample_area_m2 = 1000,
  min_dbh_cm = 7.5,
  name = "Square 0.1-ha plot",
  metadata = list(shape = "square", side_m = sqrt(1000))
)

pars <- data.frame(
  species = c("sp1", "sp2"),
  a = c(0.00002, 0.00003),
  b = c(2.30, 2.10),
  stringsAsFactors = FALSE
)

reg <- external_volume_method_registry(list(
  V = new_volume_method(
    output = "v",
    fun = function(dbh_mm, h_m, pars) {
      dbh_cm <- dbh_mm / 10
      pars$a + pars$b * (dbh_cm^2) * h_m
    },
    raw_unit = "cm3",
    unit = "m3",
    scale_to_m3 = 1 / 1e6,
    build_args = function(ctx, pars, resolved) {
      list(dbh_mm = ctx$d_mm, h_m = ctx$h_m, pars = pars)
    },
    fallback = function(ctx, pars, resolved) NA_real_,
    match_by = "species",
    required_inputs = c("d", "h")
  )
))

out <- inventoryMetrics(
  ext,
  backend = "external",
  schema = sch,
  design = dsg,
  parameter_table = pars,
  method_registry = reg,
  summ.vr = "plot",
  var = c("d", "h", "ba", "n", "v"),
```

```
  parametro = "V"
)

out
attr(out, "backend")
attr(out, "units")
```

`metrics2Vol` *Compute tree-level volume variables from NFI metrics*

Description

Computes one or more tree-level volume variables from Spanish National Forest Inventory data. The function standardizes the input, selects the appropriate volume model for each tree, applies equation-based methods defined in a method registry, converts results to cubic metres, and optionally keeps legacy outputs (from previous versions of the package) and provenance information.

Usage

```
metrics2Vol(nfi, cub.met = "freq",
    parametro = c("VCC"),
    keep.var = TRUE,
    keep.legacy = FALSE,
    method_registry = snfi_volume_method_registry(),
    track_provenance = FALSE,
    ...)
```

Arguments

`nfi`	Input accepted by ‘nfiMetrics()‘, or a precomputed ‘"nfiMetrics"‘ object with tree-level metrics.
`cub.met`	Cubication selector used when several coefficient rows match.
`parametro`	One or more volume outputs to compute.
`keep.var`	Keep auxiliary coefficient columns when available.
`keep.legacy`	Also return the legacy volume estimate for backward compatibility.
`method_registry`	Registry that maps each requested output to its equation function, output column name, units, and fallback rule.
`track_provenance`	Add per-row provenance columns and audit metadata.
`...`	Passed to ‘nfiMetrics()‘ when ‘nfi‘ is not already an ‘"nfiMetrics"‘ object.

Details

Computes requested volume outputs from standardized NFI tree metrics using registry-based methods and optional fallback to legacy estimates.

Value

A data.frame with the requested volume outputs added.

Note

Registry-based dispatcher for volume equations. Return early for 'NULL' input. Build dendrometric metrics when the input is not already standardized. Resolve the inventory edition from the object attribute first and from a column only as a fallback. Match the first available column name among accepted aliases. Identify the core columns needed by the volume equations. Validate and standardize measurement-unit metadata before dispatching any equation. Validate requested outputs against the method registry. Preallocate all requested output columns. Compute the legacy volume once so modern methods can fall back to it when needed. Decide whether legacy estimates can be computed from the input. need_legacy <- keep.legacy || any(parametro %in% c("V", "VCC")) Load and normalize the coefficient table used by registry-based methods. Convert relevant inputs and coefficient keys to numeric values. Cache row matches to avoid repeated filtering of the same coefficient subset. Resolve the parameter row for one method and one tree. Resolve the equation function for one registry entry. Evaluate one registry method and return both value and provenance. Evaluate methods row by row. 'VCC' is computed first because later outputs may reuse it. Drop auxiliary coefficient columns when requested. Keep only the requested computed outputs. Reorder identifying columns to the front of the output. Rebuild units from surviving input columns plus returned volume outputs. 'nfiMetrics' stores units as a named vector: names = variable names, values = unit strings. Add units for the computed outputs that are returned. Computed outputs override any original units with the same name. Keep units only for columns present in the final output.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

See Also

nfiMetrics, snfi_volume_method_registry

Examples

```
if (FALSE) {
  x <- nfiMetrics('toledo')
  y <- metrics2Vol(x, parametro = c('VCC', 'VSC'))
  head(y[c('nfi.nr', 'pr', 'especie', 'vcc', 'vsc')])
}
```

new_concentric_design *Construct a concentric subplot design*

Description

Define a concentric inventory design from subplot radii and minimum diameter thresholds.

The constructor orders subplot tiers by minimum DBH, converts radii to sampled area in square metres, computes trees-per-hectare expansion factors, and returns an object of class `"concentric_design"` inheriting from `"inventory_design"`.

Usage

```
new_concentric_design(radii_m,
    min_dbh_cm, name = "custom",
    metadata = NULL)
```

Arguments

`radii_m`	`numeric`. Subplot radii in metres, one value per « tally tier.
`min_dbh_cm`	`numeric`. Minimum diameter at breast height in « `cm` required for a tree to be tallied in each « subplot. Must have the same length as « `radii_m`. The function sorts these thresholds in « ascending order and reorders the paired radii « accordingly.
`name`	`character(1)`. Human-readable design `name` « stored in the returned object.
`metadata`	Optional `list`. Additional design `metadata` « merged with default entries for « `shape = "circular"` and the ordered « `radii_m`.

Details

The function delegates to `new_inventory_design`

and then appends the ordered `radii_m` vector plus the

class `"concentric_design"`. The `metadata`

argument is merged with default entries for circular plot

shape and ordered radii using

`utils::modifyList()`.

Value

An object of class

`c("concentric_design", "inventory_design")`, returned

as a named list with components `name`,

`min_dbh_cm`, `sample_area_m2`, `sf`,

`metadata`, and `radii_m`. The

sample_area_m2 component stores subplot areas in square metres, sf stores the corresponding trees-per-hectare expansion factors, and metadata contains the merged design metadata.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
dsg <- new_concentric_design(
    radii_m = c(5, 10, 15, 25),
    min_dbh_cm = c(7.5, 12.5, 22.5, 42.5),
    name = "SNFI"
)
dsg
```

---

new_external_schema *Define a schema for external inventory workflows*

Description

Create a reusable "external_schema" object that records how Create a reusable schema with column aliases, units, default grouping settings, retained columns, and optional defaults for the external basifoR workflow. source columns in an external inventory correspond to the standardized Create a reusable schema with column aliases, units, default grouping settings, retained columns, and optional defaults for the external basifoR workflow. field names used by the basifoR external workflow. The schema can also Create a reusable schema with column aliases, units, default grouping settings, retained columns, and optional defaults for the external basifoR workflow. store declared measurement units, default grouping levels, columns to Create a reusable schema with column aliases, units, default grouping settings, retained columns, and optional defaults for the external basifoR workflow. preserve during processing, and auxiliary defaults that wrapper Create a reusable schema with column aliases, units, default grouping settings, retained columns, and optional defaults for the external basifoR workflow. functions may reuse across repeated calls. Create a reusable schema with column aliases, units, default grouping settings, retained columns, and optional defaults for the external basifoR workflow.

Usage

```
new_external_schema(colmap,
    units = list(), levels = NULL,
    keep_cols = NULL,
    defaults = list())
```

Arguments

colmap	Named list mapping standardized variables to one or more candidate source column names in the input data.
units	Named list of declared units for standardized variables, usually entries such as list(d = "mm", h = "m").
levels	Optional character vector of default grouping variables for downstream summaries.
keep_cols	Optional character vector naming source columns that should be retained in downstream outputs.
defaults	Optional named list of auxiliary defaults or metadata that wrappers may reuse.

Details

The returned object is lightweight. It validates the basic structure of the inputs, assigns class c("external_schema", "list"), and leaves semantic interpretation to downstream helpers such as externalMetrics(), externalMetrics2Vol(), or external_dendroMetrics().

Value

An object of class "external_schema" containing normalized colmap, units, levels, keep_cols, and defaults components, ready to pass to external workflow wrappers.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
sch <- new_external_schema(
    colmap = list(
        plot = c("plot_id", "plot"),
        species = c("species_code", "sp"),
        d = c("dbh_mm", "diameter_mm"),
        h = c("height_m", "h")
    ),
    units = list(d = "mm", h = "m"),
    levels = "plot",
    keep_cols = c("plot_id", "species_code"),
    defaults = list(selector = "priority")
```

```
)

class(sch)
sch$colmap$d
sch$units
sch$levels
```

`new_inventory_design` *Create a generic inventory sampling design*

Description

Build a generic inventory design from tally areas and minimum DBH Construct a generic `"inventory_design"` object from sampled areas and minimum DBH thresholds. The function computes trees-per-hectare expansion factors for each tally tier and can represent circular, square, strip, ring, or custom layouts through `metadata`. thresholds. The function stores each tally tier, computes its Construct a generic `"inventory_design"` object from sampled areas and minimum DBH thresholds. The function computes trees-per-hectare expansion factors for each tally tier and can represent circular, square, strip, ring, or custom layouts through `metadata`. trees-per-hectare expansion factor, and returns an object of class Construct a generic `"inventory_design"` object from sampled areas and minimum DBH thresholds. The function computes trees-per-hectare expansion factors for each tally tier and can represent circular, square, strip, ring, or custom layouts through `metadata`. `"inventory_design"` for use in inventory workflows. Construct a generic `"inventory_design"` object from sampled areas and minimum DBH thresholds. The function computes trees-per-hectare expansion factors for each tally tier and can represent circular, square, strip, ring, or custom layouts through `metadata`.

Usage

```
new_inventory_design(sample_area_m2,
    min_dbh_cm = 0, name = "custom",
    metadata = NULL)
```

Arguments

`sample_area_m2`	`numeric`. Positive sampled area, in square « metres, for each tally tier. Supply one value « per diameter threshold.
`min_dbh_cm`	`numeric`. Minimum diameter at breast height, « in `cm`, required for a tree to enter each « tally tier. Must have the same length as « `sample_area_m2`. The function sorts these « thresholds in ascending order and reorders the « paired sampled areas accordingly.
`name`	`character(1)`. Human-readable label stored « in the returned design object.
`metadata`	Optional `list`. Extra design `metadata` stored « unchanged, for example plot shape, subplot radii, « side length, strip dimensions, or field notes.

Details

The object is shape-agnostic. Use `metadata` to record how the sampled area was obtained, for example from circular radii, square side lengths, strip dimensions, or protocol notes. The constructor stores `metadata` as supplied and does not validate or reorder its contents.

Value

An object of class `"inventory_design"`, returned as a named list with components `name`, `min_dbh_cm`, `sample_area_m2`, `sf`, and `metadata`. `sf` gives the trees-per-hectare expansion factor for each tally tier.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
dsg <- new_inventory_design(
    sample_area_m2 = c(400, 100),
    min_dbh_cm = c(20, 0),
    name = "Nested square design",
    metadata = list(shape = "square", side_m = c(20, 10))
)

dsg$min_dbh_cm
dsg$sf
```

---

new_volume_method *Define one external volume-computation method*

Description

Creates a compact method specification for the external volume registry. A method definition tells Create a method specification for registry-based volume calculations in external inventory workflows. `externalMetrics2Vol` which `output` to compute, where to find or resolve parameters, how to Create a method specification for registry-based volume calculations in external inventory workflows. build the argument list for the equation, how to scale the raw result to cubic metres,

and which Create a method specification for registry-based volume calculations in external inventory workflows. fallback to return when a direct computation is not possible. Create a method specification for registry-based volume calculations in external inventory workflows.

Usage

```
new_volume_method(output,
    fun = NULL, fun_name = NULL,
    unit = "m3", raw_unit = unit,
    scale_to_m3 = 1,
    build_args = function(ctx,
        pars, resolved) list(),
    fallback = function(ctx,
        pars, resolved) null_or(resolved$preexisting_v_m3,
        NA_real_), match_by = character(0),
    get_pars = NULL,
    pars = NULL, filter_pars = NULL,
    required_inputs = NULL)
```

Arguments

output	character(1). Name of the output column produced by the method, for example "v", "vcc", or "vsc".
fun	Optional function. Direct function used to compute the raw output value.
fun_name	Optional character(1). Name of a function to resolve at run time when fun is not supplied.
unit	character(1). Unit reported for the returned value after scaling.
raw_unit	character(1). Unit produced by the raw method before applying scale_to_m3.
scale_to_m3	numeric(1). Multiplicative factor used to convert the raw result to cubic metres.
build_args	function. Builds the argument list passed to the equation function. It receives the current row context, the selected parameter row, and already resolved outputs.
fallback	fallback
match_by	character. Column names used to match candidate parameter rows against the current row context.
get_pars	Optional function. Custom resolver that returns the parameter rows to use for the current tree or observation.
pars	Optional embedded parameter table stored inside the method definition.
filter_pars	Optional function. Additional filter applied after parameter matching and before row selection.
required_inputs	Optional character. Standardized inputs that must be available before the method can run, for example c("d", "h").

Details

The `fallback` function should return a scalar numeric value, typically `NA_real_` or a pre-existing volume estimate already present in the data. Define `required_inputs` with the standardized variable names expected by the method so the workflow can resolve prerequisites before evaluation.

Value

A named `list` describing one external volume method.

The list contains the components `output`, `fun`,

`fun_name`, `unit`, `raw_unit`,

`scale_to_m3`, `build_args`, `fallback`,

`match_by`, `get_pars`, `pars`,

`filter_pars`, and `required_inputs`.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
## Minimal standalone example
vm <- new_volume_method(
    output = "vcc",
    fun = function(dbh_cm, h_m, pars) dbh_cm * h_m * pars$k[1],
    unit = "m3",
    raw_unit = "cm3",
    scale_to_m3 = 1 / 1e6,
    build_args = function(ctx, pars, resolved) {
        list(dbh_cm = ctx$d_cm, h_m = ctx$h_m, pars = pars)
    },
    fallback = function(ctx, pars, resolved) NA_real_,
    match_by = "species",
    pars = data.frame(species = "sp1", k = 2500),
    required_inputs = c("d", "h")
)

names(vm)
vm$output
vm$build_args(
    ctx = list(d_cm = 20, h_m = 12),
    pars = vm$pars,
    resolved = list()
)
```

nfiMetrics *Tree-level metrics for Spanish NFI inputs*

Description

Compute tree-level diameter, height, basal area, trees per hectare, and optional dominant height from Spanish National Forest Inventory inputs. Supply either an object returned by readNFI or a path/URL that readNFI can import. The function returns the requested metrics together with the matched grouping columns and attaches unit metadata to the result.

Usage

```
nfiMetrics(nfi, var = c("d",
    "h", "ba", "n", "Hd"),
    levels = c("esta",
        "espe"), design = snfi_design(),
    ...)
```

Arguments

nfi	character(1) or a "readNFI" object. « Supply either a path/URL that readNFI can « import or an already imported object returned by « readNFI. The input should contain the SNFI « diameter and height fields, usually aliases such as « Dn and altura.
var	character. Metrics to « compute. Supported values are « 'd' (diameter in « mm), 'h' « (height in dm), « 'ba' (basal area per « tree in m^2), « 'n' (trees per hectare), « and 'Hd' (dominant « height in dm). « Request 'h', « 'd', and 'n' « together when you request « 'Hd'.
levels	character. Column-name « patterns used to keep grouping « variables in the output. Matching « ignores case and accepts partial « matches. The default usually keeps « plot and species identifiers when « those fields are present.
design	Sampling design used to derive « 'n' and any dependent « 'Hd' calculation. Supply the « default snfi_design(), « another "concentric_design", « or any "inventory_design" « supported by trees_per_ha. « The returned object stores a summary of « the design in attr(x,<< "design_meta") when relevant.
...	Additional arguments passed to readNFI when « nfi is not already a "readNFI" object.

Details

If you request 'Hd', the function computes

dominant height within the groups selected by

levels by using 'h', 'd', and

'n'.

Value

`data.frame` with the matched identifier and grouping columns plus the metrics requested in `var`. The output inherits from `'nfiMetrics'` and `'data.frame'` and stores `attr(x, 'nfi.nr')` when available. Inspect `attr(x, 'units')` for the returned metric units and `attr(x, 'design_meta')` for the sampling `design` summary attached when `'n'` or `'Hd'` is requested.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

See Also

dendroMetrics, dbhMetric, readNFI, snfi_design, trees_per_ha

Examples

```
## Minimal reproducible example with a small object that mimics
## readNFI() output
toy_ifn <- structure(
    data.frame(
        esta = c("plot1", "plot1", "plot2"),
        espe = c("sp1", "sp2", "sp1"),
        Dn = c(120, 185, 260),
        altura = c(7.1, 9.4, 13.2),
        stringsAsFactors = FALSE
    ),
    class = c("readNFI", "data.frame"),
    nfi.nr = 4
)

x <- nfiMetrics(
    toy_ifn,
    var = c("d", "h", "ba", "n"),
    levels = c("esta", "espe")
)

x
attr(x, "units")
```

`print.concentric_design`

Print a concentric plot design

Description

Display the main components of a "concentric_design" object: Display the main components of a "concentric_design" object. The printed summary reports the design name together with subplot radii, minimum DBH thresholds, and expansion factors for each tally tier. design name, subplot radii, minimum diameters, and expansion factors. Display the main components of a "concentric_design" object. The printed summary reports the design name together with subplot radii, minimum DBH thresholds, and expansion factors for each tally tier.

Usage

```
## S3 method for class 'concentric_design'
print(x,
    ...)
```

Arguments

x	Object of class "concentric_design", typically « created by new_concentric_design.
...	Further arguments passed to methods. Currently unused.

Details

The function does not modify the design and does not currently use additional arguments passed through ....

Value

The input "concentric_design" object, returned invisibly.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
dsg <- new_concentric_design(
    radii_m = c(5, 10, 15),
    min_dbh_cm = c(7.5, 22.5, 42.5),
    name = "Example concentric design"
)

print.concentric_design(dsg)
invisible(NULL)
```

print.external_schema *Print an external schema summary Print an external schema summary*

Description

Display the column mappings, declared units, and default grouping levels stored in an "external_schema" object.

Usage

```
## S3 method for class 'external_schema'
print(x,
    ...)
```

Arguments

x	An "external_schema" object created by new_external_schema.
...	Additional arguments accepted for S3 compatibility and ignored by this method.

Value

The input "external_schema" object, returned invisibly.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

print.inventory_design
Print a generic inventory plot design

Description

Display the main stored components of an "inventory_design" Print a compact summary of an "inventory_design" object. The method reports the stored design name, minimum DBH thresholds, sampled areas, and trees-per-hectare expansion factors. object in a compact human-readable summary. Print a compact summary of an "inventory_design" object. The method reports the stored design name, minimum DBH thresholds, sampled areas, and trees-per-hectare expansion factors.

Usage

```
## S3 method for class 'inventory_design'
print(x,
    ...)
```

Arguments

x	Object of class "inventory_design".
...	Further arguments passed to methods. Currently unused.

Details

The method does not modify the object.

Value

The input "inventory_design" object, returned invisibly.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
dsg <- new_inventory_design(
    sample_area_m2 = c(400, 100),
    min_dbh_cm = c(20, 0),
    name = "Nested square design"
)

print(dsg)
```

---

readNFI *Read raw SNFI tables from archives, URLs, or local files*

Description

Read raw tables from the Spanish National Forest Inventory (SNFI) Import raw inventory tables from the Spanish National Forest Inventory (SNFI) or compatible tabular exports. Use readNFI() when you need the original table structure before computing metrics with higher-level workflows. and compatible inventory exports. The function accepts province Import raw inventory tables from the Spanish National Forest Inventory (SNFI) or compatible tabular exports. Use readNFI() when you need the original table structure before computing metrics with higher-level workflows. identifiers that are resolved to official SNFI download URLs, local Import raw inventory tables from the

Spanish National Forest Inventory (SNFI) or compatible tabular exports. Use `readNFI()` when you need the original table structure before computing metrics with higher-level workflows. or remote `.zip` archives, and direct paths to decompressed Import raw inventory tables from the Spanish National Forest Inventory (SNFI) or compatible tabular exports. Use `readNFI()` when you need the original table structure before computing metrics with higher-level workflows. `.csv`, `.dbf`, `.mdb`, or `.accdb` files. Import raw inventory tables from the Spanish National Forest Inventory (SNFI) or compatible tabular exports. Use `readNFI()` when you need the original table structure before computing metrics with higher-level workflows.

Usage

```
readNFI(nfi, nfi.nr = 4,
    dt.nm = "PCMayores",
    file_ext = NULL,
    file_name = NULL,
    ...)
```

Arguments

`nfi`	`character`. Inventory source to read. Accepted values are: (i) a province name or province code to be resolved to an official SNFI download URL; (ii) a local or remote `.zip` archive; or (iii) one or more direct paths to decompressed `.csv`, `.dbf`, `.mdb`, or `.accdb` files.
`nfi.nr`	`integer`. SNFI stage used when `nfi` is given as a province identifier or when the inventory stage cannot be inferred from file names. Use `2`, `3`, or `4`.
`dt.nm`	`character`. Table name or names to import from the selected inventory source. For second-stage DBF inputs, `"PCMayores"` is internally remapped to `"PIESMA"`. For many third- and fourth-stage tree workflows, `"PCMayores"` is the main tree table.
`file_ext`	`character`. Optional file extension or extensions forwarded to `fetchNFI` when `nfi` is a province identifier or a `.zip` archive. Leave `NULL` to use the default Access/DBF extensions handled by `fetchNFI()`. Use `"csv"` for zipped CSV exports.
`file_name`	`character`. Optional file name or stem passed to `fetchNFI` to keep only specific files inside a `.zip` archive.
`...`	Additional arguments passed to `fetchNFI`, such as `dir` or `timeOut`.

Details

The input `nfi` can be supplied in three main forms. First, it can be a province name or code; in that case, `readNFI()` resolves the identifier to an official SNFI download URL according to `nfi.nr`. Second, it can be a local or remote `.zip` archive; the function then delegates extraction to `fetchNFI`. Third, it can be one or more already decompressed file paths.

When the selected files are `.csv`, the function detects the field separator automatically and returns either one data frame or a named list of data frames. When the selected files are Access or DBF tables from the SNFI, the function reads the requested table, converts numeric-looking factors back to numeric values, preserves character columns, and adds province and inventory-stage metadata.

Access backends are platform dependent. On Windows, reading `.mdb` or `.accdb` files requires package **RODBC** and an installed Microsoft Access driver. On Unix-like systems, it requires package **Hmisc** together with the external `mdbtools` utilities. Use `file_ext = "csv"` to bypass those dependencies when you work with zipped CSV exports.

Value

Returns one of three object types, depending on the input. First, when `nfi` resolves to `.csv` file paths, the function returns a single `data.frame` for one file or a named `list` of `data.frames` for several files. Second, when it reads Access or DBF tables from the SNFI, it returns a `data.frame` of class `c("readNFI", "data.frame")`. This object includes leading columns `nfi.nr` and `pr`, stores the province vector in `attr(x, "pr.")`, and stores the inferred inventory stage in `attr(x, "nfi.nr")`. Third, the function returns `NULL` when fetching, extraction, or import fails, or when no requested file can be read.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
## Minimal example using a local CSV file created on the fly
tmp <- tempfile(fileext = ".csv")
utils::write.table(
    data.frame(
        plot = 1:2,
        species = c("sp1", "sp2"),
        dbh_cm = c(12.5, 18.0)
    ),
    file = tmp,
    sep = ";",
    row.names = FALSE,
    quote = FALSE
)

x <- readNFI(tmp)
str(x)

unlink(tmp)
```

`snfi_design` *Return the default SNFI concentric subplot design*

Description

Return the predefined concentric sampling design used by `basifoR` for Spanish National Forest Inventory workflows.

Usage

```
snfi_design()
```

Details

Use this function when you want the default SNFI design object explicitly, for example before calling `trees_per_ha` or when inspecting the design used by SNFI-oriented workflows.

Value

An object of class `"concentric_design"` that also inherits from `"inventory_design"`. It stores the default SNFI subplot radii, minimum DBH thresholds, sampled areas, expansion factors, design name, and metadata.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
snfi_design()
```

---

snfi_volume_method_registry

Assemble the active SNFI volume-method registry

Description

Return the final registry of SNFI volume methods after merging package defaults, optional user definitions, and session-level option overrides.

Usage

```
snfi_volume_method_registry(equations = get0("snfi_volume_equations",
    inherits = TRUE,
    ifnotfound = NULL))
```

Arguments

equations	equations

Details

Each registry entry is a named list that typically contains fields such as `output`, `fun_name`, `unit`, `raw_unit`, `scale_to_m3`, `build_args`, and `fallback`. The function checks only that the supplied registry is a named list; downstream functions validate and use the individual fields.

Value

A named list of SNFI volume-method definitions.

Names usually correspond to requested parameters such as `"V"`, `"VCC"`, and `"VSC"`.

Each element describes one computation pathway and commonly includes:

`output` Name of the output column returned by the method.

`fun_name` Name of the equation helper called at run time, or `NULL` for direct passthrough methods.

`unit` Returned unit after scaling, usually `"m3 tree-1"`.

`raw_unit` Unit returned by the underlying equation before scaling.

`scale_to_m3` Multiplier applied to raw outputs to express them in cubic metres.

`build_args` Function that builds the argument list for the equation helper.

`fallback` Function used when coefficients are missing or a method cannot compute the requested output.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
c(
    "reg <- snfi_volume_method_registry()",
    "names(reg)",
    "reg$VCC$output",
    "custom <- list(VCC = list(output = 'vcc_m3'))",
    "reg2 <- snfi_volume_method_registry(custom)",
    "reg2$VCC$output"
)
```

`trees_per_ha` *Compute trees-per-hectare expansion factors from inventory designs*

Description

Dispatch on the sampling-design class to return the expansion factor, Return the expansion factor, expressed as trees per hectare, associated with a tree of a given diameter at breast height under a supported inventory `design`. expressed as trees per hectare, associated with a tree of a given Return the expansion factor, expressed as trees per hectare, associated with a tree of a given diameter at breast height under a supported inventory `design`. diameter at breast height. Return the expansion factor, expressed as trees per hectare, associated with a tree of a given diameter at breast height under a supported inventory `design`.

Usage

```
trees_per_ha(design,
    dbh_cm)
```

Arguments

`design`	Sampling `design` object. Supported methods currently include `"inventory_design"` and `"concentric_design"`.
`dbh_cm`	`numeric`. Diameter at breast height in `cm`. When several values are supplied, methods first coerce them to numeric and then use their mean after removing missing values.

Details

Current methods return `NA_real_` when diameter is

missing after preprocessing or falls below the smallest

supported threshold in the supplied `design`.

Value

A single `numeric` value giving the trees-per-hectare

expansion factor for the supplied diameter and `design`, or

`NA_real_` when no valid factor can be assigned.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
dsg <- new_inventory_design(sample_area_m2 = c(100, 400),
                            min_dbh_cm = c(0, 20))
trees_per_ha(dsg, 13)
```

`trees_per_ha.concentric_design`

Compute trees per hectare for concentric subplot designs Compute trees per hectare for concentric subplot designs

---

Description

Match a tree diameter to the appropriate subplot of a Return the trees-per-hectare expansion factor for a tree measured under a concentric subplot `design`. `"concentric_design"` object and return the corresponding Return the trees-per-hectare expansion factor for a tree measured under a concentric subplot `design`. trees-per-hectare expansion factor. Return the trees-per-hectare expansion factor for a tree measured under a concentric subplot `design`.

Usage

```
## S3 method for class 'concentric_design'
trees_per_ha(design,
    dbh_cm)
```

Arguments

`design`	Object of class `"concentric_design"` created by helpers such as `new_concentric_design` or `snfi_design`.
`dbh_cm`	`numeric`. Diameter at breast height in `cm`. When several values are supplied, the method uses their mean after removing missing values.

Details

It returns `NA_real_` when all supplied diameters are

missing or when the resulting diameter is smaller than the

smallest subplot threshold.

Value

A single `numeric` expansion factor in trees per hectare

taken from `design$sf`, or `NA_real_` when the tree

does not belong to any subplot tier.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
trees_per_ha(snfi_design(), 13)
```

`trees_per_ha.inventory_design` *Compute trees per hectare for generic inventory designs Compute trees per hectare for generic inventory designs*

Description

Match a tree diameter to the tally tier of a generic Return the trees-per-hectare expansion factor for a tree measured under a generic inventory `design`. `"inventory_design"` object and return the corresponding Return the trees-per-hectare expansion factor for a tree measured under a generic inventory `design`. trees-per-hectare expansion factor. Return the trees-per-hectare expansion factor for a tree measured under a generic inventory `design`.

Usage

```
## S3 method for class 'inventory_design'
trees_per_ha(design,
    dbh_cm)
```

Arguments

`design`	Object of class `"inventory_design"` created by helpers such as `new_inventory_design`.
`dbh_cm`	`numeric`. Diameter at breast height in `cm`. When several values are supplied, the method uses their mean after removing missing values.

Details

It returns `NA_real_` when all supplied diameters are missing or when the resulting diameter is smaller than the minimum threshold supported by the `design`.

Value

A single `numeric` expansion factor in trees per hectare taken from `design$sf`, or `NA_real_` when the tree does not belong to any tally tier.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
dsg <- new_inventory_design(sample_area_m2 = c(100, 400),
                            min_dbh_cm = c(0, 20),
                            name = "Example design")
trees_per_ha(dsg, 13)
```

---

update.dendroMetrics *Update a stored dendroMetrics call Update a dendroMetrics result*

Description

Re-run dendroMetrics from the call stored in a previous "dendroMetrics" result while replacing one or more named arguments in .... This method supports reproducible re-evaluation of the original workflow and can either return the updated result or the reconstructed call.

Usage

```
## S3 method for class 'dendroMetrics'
update(object,
    ..., evaluate = TRUE)
```

Arguments

object	A "dendroMetrics" object created by the updatable dendroMetrics definition.
...	Named arguments used to replace entries in the stored call before evaluation.
evaluate	logical. If TRUE, evaluate the updated call and return the resulting object. If FALSE, return the reconstructed call without evaluation.

Value

A new "dendroMetrics" object when evaluate = TRUE; otherwise the updated call.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

`update.external_dendroMetrics` *Update a stored external_dendroMetrics call*

Description

Rebuild the call stored in a previous `"external_dendroMetrics"` result, optionally replace named arguments, and either `evaluate` the updated call or return it unevaluated.

Usage

```
## S3 method for class 'external_dendroMetrics'
update(object,
    ..., evaluate = TRUE)
```

Arguments

`object`	Object returned by `external_dendroMetrics()`.
`...`	Named arguments used to replace entries in the stored call.
`evaluate`	If `TRUE`, `evaluate` the updated call; otherwise return the call.

Details

The method checks that `object` inherits from `"external_dendroMetrics"` and that the original matched call is stored in `attr(\code{object}, "call")`. It then replaces any named arguments supplied in `...` inside that stored call.

Use `evaluate = FALSE` to inspect the reconstructed call before execution. This is useful when debugging filters, grouping variables, schemas, or volume-method options.

The method checks that `object` inherits from `"external_dendroMetrics"` and that it stores the original matched call in `attr(\code{object}, "call")`. It then replaces any named arguments supplied in `...` inside that stored call.

Use `evaluate = FALSE` to inspect the reconstructed call before execution, which is useful when debugging filters, grouping variables, schemas, or volume-method options.

The method only changes arguments supplied explicitly in `...`; all other arguments remain as stored in the original call.

Value

A new `"external_dendroMetrics"` `object` when `evaluate = TRUE`; otherwise the updated call.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Examples

```
x <- structure(
    data.frame(plot = "P1", d = 12, h = 8),
    class = c("external_dendroMetrics", "dendroMetrics", "data.frame")
)

attr(x, "call") <- quote(
    external_dendroMetrics(
        x = data.frame(plot = "P1", d = 12, h = 8),
        var = c("d", "h"),
        summ.vr = NULL
    )
)

update(x, var = c("d", "h", "ba"), evaluate = FALSE)
```

`update.inventoryMetrics` *Update a stored inventoryMetrics call Update a stored inventoryMetrics call*

Description

Re-run `inventoryMetrics` from the call stored in a previous `"inventoryMetrics"` result while replacing one or more named arguments in `....` This method supports reproducible re-evaluation of the selected backend and can either return the updated result or the reconstructed call.

Usage

```
## S3 method for class 'inventoryMetrics'
update(object,
    ..., evaluate = TRUE)
```

Arguments

`object`	A `"inventoryMetrics"` `object` created by the « updatable `inventoryMetrics` definition.
`...`	Named arguments used to replace entries in the stored « call before evaluation.
`evaluate`	`logical`. If `TRUE`, `evaluate` the « updated call and return the resulting `object`. « If `FALSE`, return the reconstructed call « without evaluation.

Value

A new `"inventoryMetrics"` `object` when

`evaluate = TRUE`; otherwise the updated call.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

`update.list` *Guard raw list inputs in* `update` *Guard against raw dendroMetrics inputs*

Description

Catch attempts to call `update` on raw input lists intended for `dendroMetrics` and return a workflow-specific error message. When the input does not look like a `dendroMetrics` workflow, this method falls back to `update.default`.

Usage

```
## S3 method for class 'list'
update(object, ...,
    evaluate = TRUE)
```

Arguments

`object`	A `list` passed to `update()`.
`...`	Additional arguments passed to `update()`.
`evaluate`	`logical`. Passed through to `update.default` when no `dendroMetrics`-specific guard is triggered.

Value

Either an error with a `dendroMetrics`-specific message or the result of `update.default()`.

Author(s)

Wilson Lara <wilarhen@gmail.com> [aut, cre] (ORCID: <https://orcid.org/0000-0003-3527-1380>), Cristobal Ordonez <angelcristobal.ordonez@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-5354-3760>), Aitor Vázquez-Veloso <aitor.vazquez.veloso@uva.es> [aut] (ORCID: <https://orcid.org/0000-0003-0227-506X>), Felipe Bravo <felipe.bravo@uva.es> [aut] (ORCID: <https://orcid.org/0000-0001-7348-6695>)

Index

dbhMetric, 2
default_external_volume_methods, 4
default_snfi_volume_equations, 5
dendroMetrics, 5, *25*, *49*, *52*

external_dendroMetrics, 13
external_volume_method_registry, *4*, 17
externalMetrics, 7
externalMetrics2Vol, *4*, 9

fetchNFI, 18, *20*, *42*

get_snfi_iavc, 21
get_snfi_vcc, 22
get_snfi_vle, 23
get_snfi_vsc, 24
getNFI, 19

inventoryMetrics, 25, *51*

Logic, *6*

metrics2Vol, *5, 6*, 28

new_concentric_design, 30, *39*, *47*
new_external_schema, 31, *40*
new_inventory_design, *30*, 33, *48*
new_volume_method, *4*, 34
nfiMetrics, *5, 6*, 37

print.concentric_design, 38
print.external_schema, 40
print.inventory_design, 40

readNFI, *5, 6*, *19, 20*, *37*, 41

snfi_design, *37*, 43, *47*
snfi_volume_method_registry, 44

timeout, *18*
trees_per_ha, *37*, *44*, 46
trees_per_ha.concentric_design, 47
trees_per_ha.inventory_design, 48

update, *25*
update.dendroMetrics, 49
update.external_dendroMetrics, 50
update.inventoryMetrics, 51
update.list, 52