Package 'basifoR'

August 26, 2026

Type Package

Title Retrieval and Processing of the Spanish National Forest Inventory

Version 0.7.9

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. Spatial extensions can attach plot geometries, preserve geometry sidecars through metric workflows, and return georeferenced sf outputs for mapping and remote-sensing integration.

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, ggplot2

Imports curl, foreign, Hmisc, httr, measurements, RODBC, rvest, sf, stats, tools, 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 2026-08-26 08:50:02 UTC

Contents

addNFIcoords . . . 3
addNFIsf . . . 5
asNFI_spatial_sf . . . 7
copyNFIboundary_spatial . . . 8
copyNFIgeometry_spatial . . . 9
copyNFIspatial_sidecars . . . 9
dbhMetric . . . 10
default_dominant_height_methods . . . 11
default_external_volume_methods . . . 12
default_snfi_volume_equations . . . 13
dendroMetrics . . . 13
dominant_height_method_registry . . . 16
externalMetrics . . . 17
externalMetrics2Vol . . . 19
external_dendroMetrics . . . 22
external_volume_method_registry . . . 27
fetchNFI . . . 28
filter_gadm_province_spatial . . . 29
gadm_spatial . . . 31
getNFI . . . 32
getNFIboundary_spatial . . . 33
getNFIgeometry_spatial . . . 34
hasNFIboundary_spatial . . . 34
hasNFIgeometry_spatial . . . 35
inventoryMetrics . . . 35
inventoryMetrics_spatial . . . 38
listNFI_tables . . . 44
metrics2Vol . . . 46
metrics2Vol_spatial . . . 48
new_concentric_design . . . 54
new_dominant_height_method . . . 55
new_external_schema . . . 57
new_inventory_design . . . 58
new_volume_method . . . 60
nfiMetrics . . . 62
nfiMetrics_spatial . . . 64
plot.inventoryMetrics_spatial . . . 68
plot.metrics2Vol_spatial . . . 69
plot.nfiMetrics_spatial . . . 69
plot.readNFI_spatial . . . 70
print.concentric_design . . . 71
print.external_schema . . . 72
print.inventory_design . . . 73
readNFI . . . 74
readNFIcoords . . . 75
readNFIsf . . . 78

readNFI_spatial 80
snfi_design 85
snfi_volume_method_registry 86
trees_per_ha 87
trees_per_ha.concentric_design 88
trees_per_ha.inventory_design 89
update.dendroMetrics 90
update.external_dendroMetrics 91
update.inventoryMetrics 92
update.list 93

Index **94**

---

addNFIcoords *addNFIcoords*

Description

addNFIcoords: attach NFI plot coordinates as ordinary columns.

Join a processed Spanish National Forest Inventory table with a plot-level coordinate table and append UTM coordinate metadata. This function is intentionally independent from `readNFI()`, so it can enrich already loaded tables without creating circular calls.

Usage

```
addNFIcoords(nfi, coords,
    nfi.nr = attr(nfi,
        "nfi.nr"), x.col = NULL,
    y.col = NULL, huso.col = NULL,
    x.name = "x", y.name = "y",
    huso.name = "huso",
    huso.source.name = "huso_source",
    datum.name = "datum",
    epsg.name = "epsg",
    crs.name = "crs",
    overwrite = FALSE,
    keep.raw.coords = FALSE,
    coord.factor = NULL,
    validate = TRUE,
    infer.huso = FALSE)
```

Arguments

`nfi`	`data.frame`. Inventory table, commonly an object returned by `readNFI`.
`coords`	`data.frame` or one-table `list`. Coordinate table. For IFN2 this is usually `DATESTXX.DBF` or the parcel layer table with `CX`/`CY`. For IFN3/IFN4 this is usually `PCDatosMap` or `Listado definitivo`.

nfi.nr	integer. Inventory stage: 2, 3, or 4. IFN2 coordinates are converted from kilometres to metres by default.
x.col	Optional source X-coordinate column in coords.
y.col	Optional source Y-coordinate column in coords.
huso.col	Optional source UTM-zone column in coords.
x.name	Output X-coordinate column name, in metres.
y.name	Output Y-coordinate column name, in metres.
huso.name	Output UTM-zone column name when available.
huso.source.name	Output column describing whether Huso comes from the coordinate table or was inferred from province code.
datum.name	Output geodetic datum column name.
epsg.name	Output EPSG code column name when known.
crs.name	Output CRS label column name when known.
overwrite	logical. Allow overwriting existing output coordinate columns.
keep.raw.coords	logical. Keep raw source coordinate values before unit conversion.
coord.factor	Numeric multiplier applied to raw coordinates. Defaults to 1000 for IFN2 and 1 for IFN3/IFN4.
validate	logical. Warn about duplicate coordinate keys and unmatched plots.
infer.huso	logical. Fill missing UTM zones from the province code. A Huso column in coords has precedence when present. Province-filled values are marked in huso_source.

Details

The function uses province and plot identifiers to match rows in nfi with records in coords. It searches common IFN column names such as pr, Provincia, ESTADILLO, Estadillo, and NUMPAR. The main table keeps its original row order and number of rows; tree-level tables therefore receive repeated plot coordinates.

Coordinate values are treated as UTM coordinates. IFN2 coordinate columns, normally COORDEX/COORDEY or CX/CY, are interpreted as kilometres and multiplied by 1000 by default. IFN3 and IFN4 coordinate columns, normally CoorX/CoorY or CoorXC/CoorYC, are interpreted as metres.

A Huso column in the coordinate table always has priority. When infer.huso = TRUE, missing UTM zones are filled from the province code and marked as province_inferred in huso_source; otherwise the corresponding CRS fields remain NA.

Value

A data frame with the same rows and row order as nfi, plus coordinate columns. Output x and y are always in metres. Attributes record the source coordinate columns, source units, and multiplier used during conversion.

Author(s)

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>)

Examples

```
trees <- data.frame(
    nfi.nr = 2,
    pr = 45,
    ESTADILLO = c(1, 1, 2),
    ESPECIE = c(21, 21, 25)
)
attr(trees, "nfi.nr") <- 2

coords <- data.frame(
    PROVINCIA = 45,
    ESTADILLO = c(1, 2),
    COORDEX = c(412, 413),
    COORDEY = c(4411, 4412)
)

x <- addNFIcoords(trees, coords, infer.huso = TRUE)
x[, c("pr", "ESTADILLO", "x", "y", "huso", "datum", "epsg")]

## Keep raw IFN2 kilometre coordinates for checking.
addNFIcoords(trees, coords, keep.raw.coords = TRUE)
```

`addNFIsf` *addNFIsf*

Description

addNFIsf: convert an NFI table and coordinate table to sf points.

Join an already loaded inventory table to an already loaded coordinate table and return an `sf` point data frame. The inventory table is preserved as the attribute table, while coordinates are used to build a geometry column instead of being returned as ordinary x/y columns.

Usage

```
addNFIsf(nfi, coords,
    nfi.nr = attr(nfi,
        "nfi.nr"), x.col = NULL,
    y.col = NULL, huso.col = NULL,
    crs = NULL, keep.coord.meta = FALSE,
    mixed.crs = c("na",
        "error"), na.action = c("keep",
        "drop", "error"),
```

```
  overwrite = FALSE,
  keep.raw.coords = FALSE,
  coord.factor = NULL,
  validate = TRUE,
  infer.huso = NULL)
```

Arguments

nfi	data.frame. Inventory table, commonly returned by readNFI.
coords	data.frame or one-table list. Coordinate table.
nfi.nr	integer. Inventory stage: 2, 3, or 4.
x.col	Optional source X-coordinate column in coords.
y.col	Optional source Y-coordinate column in coords.
huso.col	Optional source UTM-zone column in coords.
crs	Optional CRS passed to sf::st_as_sf. When NULL, the function uses the unique EPSG derived from coordinate metadata, if one is available.
keep.coord.meta	Keep Huso/datum/EPSG metadata columns in the sf attribute table. When FALSE, these are stored only in attributes.
mixed.crs	What to do when rows imply more than one EPSG code. The default creates sf with CRS NA and warns.
na.action	Handling of rows that do not match a coordinate record.
overwrite	Allow overwriting coordinate metadata columns when keep.coord.meta = TRUE.
keep.raw.coords	Passed to addNFIcoords; mainly useful for debugging and ignored in the final sf output unless keep.coord.meta = TRUE.
coord.factor	Numeric multiplier applied to raw coordinates. Defaults to 1000 for IFN2 and 1 for IFN3/IFN4.
validate	logical. Warn about duplicate coordinate keys and unmatched plots.
infer.huso	Fill missing UTM zones from province code. By default, TRUE for IFN2 and FALSE for IFN3/IFN4.

Details

This helper is the spatial analogue of addNFIcoords(). It first creates temporary coordinate and CRS metadata columns, then converts the result to an sf point object with sf::st_as_sf(). Temporary coordinate columns are removed from the final object unless keep.coord.meta = TRUE.

An sf geometry column can store only one active CRS. If rows imply more than one EPSG code, mixed.crs = "na" creates the object with an undefined CRS and stores the detected metadata in attr(x, "coord_reference"); mixed.crs = "error" stops instead.

Rows without matching coordinates can be kept as missing point geometries, dropped, or rejected with na.action. This is useful when external CSV files contain partial plot coverage.

Value

An `sf` data frame with point geometries. The geometry coordinates are UTM metres. CRS is assigned only when a unique EPSG is available or when `crs` is supplied explicitly.

Author(s)

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>)

Examples

```
trees <- data.frame(
    nfi.nr = 2,
    pr = 45,
    ESTADILLO = c(1, 1, 2),
    ESPECIE = c(21, 21, 25)
)
attr(trees, "nfi.nr") <- 2

coords <- data.frame(
    PROVINCIA = 45,
    ESTADILLO = c(1, 2),
    COORDEX = c(412, 413),
    COORDEY = c(4411, 4412)
)

if (requireNamespace("sf", quietly = TRUE)) {
    x <- addNFIsf(trees, coords, infer.huso = TRUE)
    sf::st_crs(x)
    sf::st_geometry(x)
}
```

---

asNFI_spatial_sf *asNFI spatial sf*

Description

Reconstruct an sf object from tabular NFI data and its geometry sidecar.

Usage

```
asNFI_spatial_sf(x, registry = getNFIgeometry_spatial(x),
    na.action = c("keep",
        "drop", "error"),
    validate = TRUE)
```

Arguments

x	Tabular NFI object to convert to an sf object.
registry	Plot-geometry registry used to match one geometry to each row.
na.action	Action for rows whose plot key has no matching geometry: retain, remove, or raise an error.
validate	Logical; whether to validate registry keys and geometry consistency before reconstruction.

Author(s)

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>)

---

copyNFIboundary_spatial

copyNFIboundary spatial

Description

Copy the administrative-boundary sidecar between objects.

Usage

```
copyNFIboundary_spatial(from,
    to)
```

Arguments

from	Source object carrying an administrative-boundary sidecar.
to	Target object that will receive the boundary sidecar.

Author(s)

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>)

copyNFIgeometry_spatial *copyNFIgeometry spatial*

Description

Copy the plot-geometry sidecar between NFI objects.

Usage

```
copyNFIgeometry_spatial(from,
    to)
```

Arguments

from	Source object carrying plot-geometry and related spatial attributes.
to	Target object that will receive the spatial attributes.

Author(s)

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>)

copyNFIspatial_sidecars *copyNFIspatial sidecars*

Description

Copy both plot-geometry and administrative-boundary sidecars.

Usage

```
copyNFIspatial_sidecars(from,
    to)
```

Arguments

from	Source object carrying plot-geometry and administrative-boundary sidecars.
to	Target object that will receive both spatial sidecars.

Author(s)

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>)

`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 `m` for `met = "h"`. Returns `NA_real_` when all supplied values are missing or become missing after zero replacement.

Author(s)

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>)

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_dominant_height_methods

Default dominant-height methods

Description

Return the bundled dominant-height method specifications used by `nfiMetrics()` and external metric workflows.

Usage

```
default_dominant_height_methods()
```

Details

The default registry contains the backward-compatible method

`"Hd"` plus stricter explicit alternatives. The default

`"Hd"` mirrors the previous internal `domheight()`

behaviour, so existing analyses keep the same numerical convention

while the equation and fallback rule become visible.

Value

A named `list` of `"dominant_height_method"` objects.

Author(s)

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>)

Examples

```
methods <- default_dominant_height_methods()
names(methods)
methods$Hd$equation
```

`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 [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>)

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 includes descriptive metadata together with the target column name, equation function, unit conversion, argument builder, and fallback rule.

Usage

```
default_snfi_volume_equations()
```

Value

A named list with the default methods `V`, `VCC`, and `VSC`.

Each entry includes `label`, `equation`, `reference`, and

`description` fields for documentation, in addition to the

computational fields used by `metrics2Vol()`.

Author(s)

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>)

Examples

```
methods <- default_snfi_volume_equations()
names(methods)
methods$VCC$output
methods$VCC$unit
```

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), domheight_method = "Hd_strict",
    domheight_registry = dominant_height_method_registry(),
    ...)
```

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. For grouped summaries, use n_tot to filter total stand density. 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.
domheight_method	Dominant-height method used when Hd is recomputed during summary preparation. The default preserves the previous strict dendrometric behavior.
domheight_registry	Named dominant-height registry created with dominant_height_method_registry().
...	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'), total stand density n_tot ('ha-1'), 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. The tree-level expansion

factor n is used internally for weighting and summation; its group total is returned as n_tot. 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. With summ.vr = NULL, tree-level output keeps the expansion-factor column n. With grouped summaries, total stand density is returned as n_tot. See Details.

Author(s)

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>)

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(7.1, 9.4, 13.2),         # m
        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  = "m",
        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)

## The tree-level expansion factor n is summed and returned as n_tot
names(dendromet_toy)
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.
```

`dominant_height_method_registry` *Build the active dominant-height method registry*

Description

Return the registry of dominant-height `methods` used by `nfiMetrics()` and related metric workflows.

Usage

```
dominant_height_method_registry(methods = get0("dominant_height_methods",
    inherits = TRUE,
    ifnotfound = NULL))
```

Arguments

`methods`	Optional named `list` of method definitions. Each element should follow the structure returned by `new_dominant_height_method()`.

Details

The returned registry is built in three steps. First,
`default_dominant_height_methods()` provides the bundled
definitions. Second, `methods` replaces or extends those
defaults. Third, named entries in option
`"basifoR.dominant_height_methods"` override both. This
mirrors the volume-method registry pattern and lets users inspect
or replace the dominant-height method without editing package code.

Value

A named `list` of dominant-height method definitions.

Author(s)

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>)

Examples

```
reg <- dominant_height_method_registry()
names(reg)
reg$Hd$equation

custom <- list(
    Hd = new_dominant_height_method(
        fun_name = "domheight_strict",
        equation = "Hd = sum(h_i * n_i) / sum(n_i)",
        selection_rule = "Use the largest trees until n reaches 100 trees ha-1.",
        fallback = "Return NA if the threshold is not reached."
    )
)
dominant_height_method_registry(custom)$Hd$fallback
```

`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",
    "Dd"), levels = NULL,
    design, colmap = NULL,
    d_unit = c("mm",
        "cm")[1], h_unit = c("m",
        "dm", "cm")[1],
    keep_cols = NULL,
    domheight_fun = NULL,
    domdiameter_fun = NULL,
    domheight_method = "Hd_strict",
    domheight_registry = dominant_height_method_registry())
```

Arguments

`x`	Input data.frame with one row per tree or stem.
`var`	Requested metrics to return. Supported values are `"d"`, `"h"`, `"ba"`, `"n"`, `"Hd"`, and `"Dd"`. Requesting `"Hd"` requires `"d"`, `"h"`, and `"n"`; requesting `"Dd"` requires `"d"` and `"n"`.
`levels`	Columns defining the groups within which dominant metrics are computed.
`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 without changing the groups defined by levels.

domheight_fun — Optional function used when "Hd" is requested. If supplied, it overrides domheight_method.

domdiameter_fun — Optional function used when "Dd" is requested together with a custom domheight_fun. It must accept d and n.

domheight_method — Dominant-height method code resolved through domheight_registry when domheight_fun = NULL.

domheight_registry — Named registry created with dominant_height_method_registry.

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 and Dd, metres 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. Dominant height and dominant diameter are computed within each resolved group defined only by levels; keep_cols preserves source columns without changing those groups. Both metrics use the diameter ordering, threshold, and fallback convention selected through domheight_registry; Dd uses valid diameter and expansion-factor values and therefore may be available when Hd is missing because heights are unavailable. When both metrics are requested with custom functions, supply both domheight_fun and domdiameter_fun so the pair remains explicit.

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", "Hd", or "Dd", the result also stores sampling design metadata in attr(out, "design_meta").

Author(s)

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>)

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	Input data.frame or object inheriting from "externalMetrics". Standardized columns such as d, h, dnm, or v must have named unit metadata in the "units" attribute of x.
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	Alias list for raw diameter and height columns used by externalMetrics().
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	Default alias list used to resolve standardized inputs and contextual columns such as species, region, and equation set.
...	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.

Author(s)

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>)

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", "Dd"),
    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,
    domdiameter_fun = NULL,
    domheight_method = "Hd_strict",
    domheight_registry = dominant_height_method_registry(),
    ...)
```

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 after metrics or summaries have been computed. Use `n_tot` for grouped stand density.

`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"`, `"Hd"`, and `"Dd"`; 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` Aliases used to resolve raw diameter and height columns during internal metric computation.

`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` Aliases used by the optional volume stage to locate metrics, species, region, and equation-set selectors.

`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` Optional function used to compute dominant height. When supplied, it overrides `domheight_method` during internal metric computation.

`domdiameter_fun`
Optional function used to compute dominant diameter. Supply it with a custom `domheight_fun` when `"Dd"` is requested; it must accept standardized `d` and `n` arguments.

`domheight_method`
Dominant-height method code resolved through `domheight_registry`.

`domheight_registry`
Named registry created with `dominant_height_method_registry()`.

`...` 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`, `Hd`, and `Dd`, arithmetic sums for `ba` and volume variables, total stand density as `n_tot`, and quadratic mean diameter `dg` when diameter and expansion factors 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 classes "external_dendroMetrics", "dendroMetrics", and "data.frame".

With summ.vr = NULL, the returned rows represent tree-level records and retain the expansion-factor column n. With summ.vr supplied, grouped summaries return total stand density as n_tot, together with standardized 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", "dominant_height_meta", "dominant_diameter_meta", and "volume_meta" attributes from upstream processing, which makes the result suitable for downstream inspection and update methods.

Author(s)

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>)

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`	Optional named `list` of method definitions. Each element should follow the structure returned by `new_volume_method()` and should usually be named with the requested output code (for example `"V"`, `"VCC"`, or `"VSC"`). When `NULL`, the function first looks for an object named `external_volume_methods` in the calling environment and, if it does not exist, falls back to the package defaults.

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()`.

Author(s)

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>)

Examples

```
reg <- external_volume_method_registry()
names(reg)

custom <- list(
    V = new_volume_method(
        output = "v",
        unit = "m3",
        raw_unit = "m3",
        scale_to_m3 = 1,
        build_args = function(ctx, pars, resolved) list(),
        fallback = function(ctx, pars, resolved) 0,
        match_by = character(0),
        required_inputs = "v"
    )
)

reg2 <- external_volume_method_registry(custom)
reg2$V$output
```

`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 [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>)

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)
```

---

filter_gadm_province_spatial

filter gadm province spatial

Description

Filter a Spanish GADM layer using a `strict` province identity resolver.

Usage

```
filter_gadm_province_spatial(adm,
    prov, province_table = NULL,
    level = attr(adm,
        "gadm_level",
        exact = TRUE),
    strict = TRUE)
```

Arguments

adm	An sf object returned by gadm_spatial() or a user-provided GADM-like boundary layer.
prov	Province code or name. Numeric codes are matched directly. Non-numeric names are resolved against a clean GADM-oriented province-name lookup and, only as a last resort, against basifoR's procods code resolver.
province_table	Optional province lookup table used only by the fallback basifoR code resolver. It is not used as a list of GADM aliases.
level	GADM level of adm. Level 2 returns one province. Higher levels return features whose NAME_2 belongs to the province, for example municipalities inside a province.
strict	logical. If TRUE, level-2 matches must return one province. This prevents false matches from short aliases such as "ZA", which could match both Zamora and Zaragoza.

Details

This function deliberately does not use all aliases stored in procods for GADM filtering. The procods table can contain operational curation and province redirections used by basifoR/MITECO workflows. Boundary filtering needs geographic province identity. Therefore procods is used, at most, to resolve the input to a numeric province code; the final GADM match uses exact normalized matching against NAME_2 and a clean province-name lookup. This avoids false matches such as Zamora also selecting Zaragoza through a short alias.

Value

A filtered sf object. For GADM level 2, the object should contain one province feature. For higher levels, the object may contain several subprovince features. Attributes store the resolved province code, GADM name, and aliases tested. Exact normalized matching only. Do not use grepl() with short aliases.

Author(s)

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>)

See Also

gadm_spatial, readNFI_spatial

Examples

```
if (interactive()) {
    adm <- gadm_spatial("ESP", level = 2, path = tempdir())
    zamora <- filter_gadm_province_spatial(adm, 49)
    nrow(zamora)
}
```

gadm_spatial *gadm spatial*

Description

Download and read GADM boundaries without depending on geodata.

Usage

```
gadm_spatial(country = "ESP",
    level = 2, path = tools::R_user_dir("basifoR",
        "cache"), ext = c("json",
        "gpkg"), version = "4.1",
    quiet = TRUE)
```

Arguments

country	ISO3 country code. The default downloads Spain.
level	Administrative level to download. For Spain, level 2 is commonly useful as a province boundary layer; higher levels can be used for municipal or local cartographic context.
path	Cache directory used to store downloaded GADM files.
ext	GADM file format. The default "json" downloads the level-specific zipped GeoJSON.
version	GADM version string used to build the download URL.
quiet	Passed to sf::st_read().

Details

This helper downloads GADM directly from the public GADM distribution endpoint, caches the file in path, and reads it with **sf**. It is intended as an optional cartographic support layer for plotting and spatial quality checks. GADM boundaries are not Spanish NFI products and should not be interpreted as official inventory geometries.

Value

An sf object containing the requested GADM administrative boundary layer. Attributes record country, level, version, and URL.

Author(s)

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>)

See Also

readNFI_spatial, getNFIboundary_spatial

Examples

```
if (interactive()) {
    adm <- gadm_spatial("ESP", level = 2, path = tempdir())
    plot(sf::st_geometry(adm))
}
```

---

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 [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>)

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)
```

---

getNFIboundary_spatial

getNFIboundary spatial

Description

Return the optional administrative-boundary sidecar.

Usage

```
getNFIboundary_spatial(x)
```

Arguments

x	Object from which to retrieve the administrative-boundary sidecar stored in the `nfi_boundary` attribute.

Author(s)

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>)

getNFIgeometry_spatial

getNFIgeometry spatial

Description

Return the plot-geometry sidecar stored in a spatial NFI object.

Usage

```
getNFIgeometry_spatial(x)
```

Arguments

x	Object from which to retrieve the plot-geometry sidecar stored in the `nfi_geometry_registry` attribute..

Author(s)

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>)

hasNFIboundary_spatial

hasNFIboundary spatial

Description

Test whether an object contains a valid boundary sidecar.

Usage

```
hasNFIboundary_spatial(x)
```

Arguments

x	Object to test for a valid administrative-boundary sidecar.

Author(s)

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>)

hasNFIgeometry_spatial *hasNFIgeometry spatial*

Description

Test whether an object contains a valid plot-geometry sidecar.

Usage

```
hasNFIgeometry_spatial(x)
```

Arguments

x	Object to test for a valid, non-empty plot-geometry sidecar.

Author(s)

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>)

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 "Estadillo" for Spanish NFI and the schema level or detected plot identifier for external inventories.
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. Tree-level outputs use n for the per-record expansion factor; grouped outputs use n_tot for total stand density.

Author(s)

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>)

See Also

dendroMetrics, nfiMetrics, metrics2Vol, update.inventoryMetrics NULL is meaningful here: it requests tree-level output. Add it after compacting optional arguments so it is not discarded. NULL is meaningful here: it requests tree-level output.

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')

## Downloadable Toledo workflow. It requires internet access and an
## Access backend (mdbtools on Unix-like systems, ODBC on Windows),
## so it is not run during package checks.
## Not run:
cache <- file.path(tempdir(), 'basifoR-cache')
dir.create(cache, showWarnings = FALSE)
toledo <- inventoryMetrics(
  'toledo',
  nfi.nr = 4,
  dir = cache,
  summ.vr = 'Estadillo'
)
head(toledo)
attr(toledo, 'units')

## End(Not run)
```

`inventoryMetrics_spatial` *Complete inventory metrics with optional sf reconstruction Complete inventory workflow carrying a spatial sidecar*

Description

Run the complete inventory workflow while preserving optional plot geometry

This function is a conservative `spatial` companion to `inventoryMetrics`. It keeps the original calculation engines for Spanish NFI and external inventories, but prepares or reuses a plot-level geometry registry and carries that registry in attributes. The geometry column is reconstructed only at the end when `spatial = "sf"`. The original `inventoryMetrics()` function is not replaced.

Usage

```
inventoryMetrics_spatial(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,
    domheight_method = "Hd_strict",
    domheight_registry = dominant_height_method_registry(),
    parameter_table = NULL,
    ..., spatial = c("attribute",
        "sf", "none",
        "inherit"), spatial.summary = c("auto",
        "centroid", "none"),
    keep.geometry.meta = TRUE,
    geometry.role.name = NULL,
    geometry.dt.nm = NULL,
    coord.nm = NULL,
    coords = NULL, x.col = NULL,
    y.col = NULL, pr.col = NULL,
    plot.col = NULL,
    crs = NULL, coord.units = "m",
    geometry.source = c("auto",
        "snfi", "external",
        "none"), file_ext = NULL,
    file_name = NULL,
    coord.factor = NULL,
    huso.method = c("auto",
        "candidate",
        "xgap", "province",
        "none"), huso.candidates = NULL,
    target.crs = NULL,
    boundary = TRUE,
    boundary.source = c("auto",
        "gisco", "gadm",
        "user", "none"),
    boundary.object = NULL,
    boundary.level = 2,
```

```
    boundary.path = tools::R_user_dir("basifoR",
        "cache"), boundary.ext = "json",
    boundary.version = "4.1",
    boundary.crs = NULL,
    infer.huso = NULL,
    validate = TRUE,
    na.action = c("keep",
        "drop", "error"))
```

Arguments

nfi	Inventory source accepted by the selected backend. For Spanish NFI this can be a province code/name, URL, ZIP file, decompressed files, a readNFI object, a readNFI_spatial object, or an sf object. For external inventories it can be the same input accepted by external_dendroMetrics().
backend	Backend selector. The default infers "external" when schema or parameter_table is supplied, otherwise uses "snfi".
summ.vr	Grouping variable passed to the selected backend. When omitted, Spanish NFI uses "Estadillo" and external inventories use the schema level or detected plot identifier. Supply a vector such as plot and species for finer groups, or NULL for tree-level output.
cut.dt	Logical expression used by the backend to subset the metric output.
report	Request backend reports when supported.
mc.cores	Number of cores passed to the selected backend.
design	Optional sampling design passed to the backend.
schema	Optional external schema. If it contains defaults$spatial, it can also define external coordinates for the spatial sidecar.
method_registry	Optional volume method registry passed to the selected backend.
domheight_method	Dominant-height method passed to dendroMetrics() for SNFI inputs. For external inputs, this method is resolved to a function and passed as domheight_fun.
domheight_registry	Named dominant-height registry created with dominant_height_method_registry().
parameter_table	Optional parameter table for external inventories.
...	Additional arguments forwarded to the selected backend. A safe subset such as nfi.nr, dt.nm, dir, and timeOut is also used by readNFI_spatial() when a sidecar must be prepared from a source.
spatial	Spatial policy. "attribute" returns a tabular result with attr(x, "nfi_geometry_registry"). "sf" returns an sf object when geometry can be restored. "none" runs the original tabular workflow. "inherit" returns sf for sf input, "attribute" for sidecar input, and tabular output otherwise.

`spatial.summary`
: Geometry rule after summarization. `"auto"` preserves observed plot points when plot identity remains and uses centroids for multi-plot summaries. `"centroid"` forces centroid construction for multi-plot summaries. `"none"` leaves the result tabular while preserving the sidecar attribute.

`keep.geometry.meta`
: Store geometry role and spatial-summary metadata as attributes.

`geometry.role.name`
: Optional visible column containing the geometry role, for example `"plot_point"` or `"summary_centroid"`.

`geometry.dt.nm`
: Optional table name used by `readNFI_spatial()` to build the sidecar.

`coord.nm`
: Optional coordinate table name for Spanish NFI.

`coords`
: Optional external coordinate table.

`x.col`
: External X/easting/longitude coordinate column.

`y.col`
: External Y/northing/latitude coordinate column.

`pr.col`
: Optional province/region key for coordinate joins.

`plot.col`
: Optional plot identifier key for coordinate joins.

`crs`
: CRS for external coordinates, for example EPSG 25830.

`coord.units`
: Coordinate units recorded in the sidecar.

`geometry.source`
: Geometry source passed to `readNFI_spatial()`.

`file_ext`
: Optional file extension passed to `readNFI_spatial()`.

`file_name`
: Optional file name filter passed to `readNFI_spatial()`.

`coord.factor`
: Optional coordinate multiplier for Spanish NFI tables. IFN2 kilometre coordinates are converted to metres by default in `readNFI_spatial()`.

`huso.method`
: UTM-zone assignment method passed to `readNFI_spatial()`.

`huso.candidates`
: Optional candidate UTM zones passed to `readNFI_spatial()`, for example `c(29, 30)` for IFN2 provinces that span Huso 29 and 30 when no Huso columns are available in the coordinate table.

`target.crs`
: Optional common CRS for Spanish NFI geometries.

`boundary`
: `logical`. If `TRUE` (default), attach an optional administrative `boundary` sidecar for plot maps.

`boundary.source`
: Boundary source passed to `readNFI_spatial()`. For external inventories, `"auto"` uses only a supplied `boundary.object`; it does not infer a Spanish built-in `boundary`. For Spanish inputs, `"auto"` tries GISCO/NUTS before GADM.

`boundary.object`
: Optional user-provided `sf` polygon or multipolygon layer. This is the supported `boundary` route for external inventories.

`boundary.level`
: GADM administrative level, used when `boundary.source` resolves to `"gadm"`.

`boundary.path`
: Cache directory for optional `boundary` downloads.

boundary.ext	GADM extension used by gadm_spatial().
boundary.version	GADM version used by gadm_spatial().
boundary.crs	CRS assigned to boundary.object when missing. It is required when the supplied object has no CRS metadata.
infer.huso	Backward-compatible Huso inference argument passed to readNFI_spatial().
validate	Warn about sidecar or geometry-restoration problems.
na.action	Handling of metric rows that do not match the plot geometry registry when spatial = "sf".

Details

External inventories keep their original backend path. When external coordinates are supplied through an sf input, coords, or schema$defaults$spatial, the geometry registry is carried in attributes and may be restored if the output keeps compatible grouping keys. Without external coordinate information, external outputs remain tabular. When summ.vr is omitted for an external inventory, the function uses schema$levels when available and otherwise detects the plot identifier from the prepared input. External boundaries are never inferred from province-like columns. Supply an sf polygon or multipolygon through boundary.object; its CRS must be present or supplied through boundary.crs.

Value

A result equivalent to the selected backend output, augmented with class "inventoryMetrics_spatial". With spatial = "attribute", the output is tabular and carries the plot geometry in attr(x, "nfi_geometry_registry"). With spatial = "sf", the result is an sf object when geometry restoration succeeds. Preserve explicit summ.vr = NULL; NULL selects tree-level output. Preserve explicit summ.vr = NULL; NULL selects tree-level output.

Author(s)

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>)

Examples

```
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),
    x = c(-3.70, -3.70, -3.69),
    y = c(40.40, 40.40, 40.41)
)

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"),
    defaults = list(
        spatial = list(plot = "plot", x = "x", y = "y", crs = 4326)
    )
)

dsg <- new_inventory_design(
    sample_area_m2 = 1000,
    min_dbh_cm = 7.5,
    name = "Square 0.1-ha plot"
)

x <- inventoryMetrics_spatial(
    ext,
    backend = "external",
    schema = sch,
    design = dsg,
    summ.vr = "plot",
    var = c("d", "h", "ba", "n"),
    spatial = "attribute",
    geometry.source = "external",
    boundary = FALSE
)

inherits(x, "inventoryMetrics")
hasNFIgeometry_spatial(x)

## Tabular sidecar output for Spanish NFI:
## x <- inventoryMetrics_spatial(28, nfi.nr = 2, dir = tempdir())
## hasNFIgeometry_spatial(x)

## Direct sf output:
## xsf <- inventoryMetrics_spatial(28, nfi.nr = 2, dir = tempdir(), spatial = "sf")
```

```
## inherits(xsf, "sf")
## attr(xsf, "geometry_role")

## Dominant-height calculations use the same inspectable registry exposed
## by nfiMetrics(). Metadata is preserved in attr(x, "dominant_height_meta")
## when Hd is computed.
##
 ## External inventories keep the original external backend. If the schema
 ## contains defaults$spatial or the input is sf, the sidecar can be carried.
```

`listNFI_tables` *List available raw SNFI tables*

Description

Discover the raw tables or files available in Spanish National Forest Inventory (SNFI) downloads, local archives, or already decompressed files. The function helps users inspect available table names before reading data with `readNFI()`.

Usage

```
listNFI_tables(nfi, nfi.nr = 4,
    dir = tempdir(),
    file_ext = NULL,
    file_name = NULL,
    ...)
```

Arguments

`nfi`	`character` or `numeric`. Inventory source to inspect. 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 `.dbf`, `.mdb`, `.accdb`, or `.csv` files.
`nfi.nr`	`integer`. SNFI stage used when `nfi` is given as a province identifier. Use `2`, `3`, or `4`. The value selects the internal URL resolver `nfi2()`, `nfi3()`, or `nfi4()`.
`dir`	`character`. Directory used by `fetchNFI()` to store downloaded archives and extracted files. Use the same `dir` in later calls to `readNFI()` to reuse cached files and avoid unnecessary downloads.
`file_ext`	`character` or `NULL`. Optional file extension or extensions forwarded to `fetchNFI`. Leave `NULL` to use the default extensions defined by `fetchNFI()`.
`file_name`	`character` or `NULL`. Optional file name or bare stem forwarded to `fetchNFI` to keep only specific files inside a compressed archive.
`...`	Additional arguments passed to `fetchNFI`, such as `timeOut = httr::timeout(120)`.

Details

The input nfi can be a province name or province code, a local or remote .zip archive, or one or more already decompressed .dbf, .mdb, .accdb, or .csv files.

When nfi is a province identifier, the function resolves it with the internal URL resolver selected by nfi.nr: nfi2(), nfi3(), or nfi4(). It then calls fetchNFI() to download or reuse the corresponding local files.

For second-stage SNFI data and other DBF-based sources, the function lists one row per DBF file and uses the DBF file stem as dt.nm. For CSV sources, it lists one row per CSV file and also uses the file stem as dt.nm. For Access sources, it lists one row per table found inside each .mdb or .accdb file.

The function does not read full data tables. It only discovers table or file names. However, it must inspect local files. Therefore, when nfi is remote or province-based, at least one download or extraction step can be necessary unless the files already exist in the same dir cache.

The arguments file_ext and file_name are forwarded to fetchNFI() only when they are not NULL. This keeps fetchNFI() as the single authority for default downloadable extensions and matches the forwarding style used by readNFI().

Access table listing is platform dependent. On Windows it requires **RODBC** and a Microsoft Access ODBC driver. Package **odbc** is optional and is used only to check whether such a driver is visible. On Unix-like systems, table listing requires the external mdbtools command mdb-tables.

Value

A data.frame with one row per discovered raw table or file. The columns are source, which identifies the backend as "DBF", "CSV", or "Access"; dt.nm, the table or file stem that users can pass to readNFI() when appropriate; file, the local file basename; and path, the normalized local file path. The function returns an empty data frame with these columns when no matching files or tables are found.

Author(s)

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>)

Examples

```
## Self-contained example using a temporary DBF file.
tmp <- tempfile(fileext = ".dbf")
foreign::write.dbf(
    data.frame(
        plot = 1:2,
        tree = 1:2,
        dbh = c(15.2, 31.8)
    ),
    file = tmp
)

tabs <- listNFI_tables(tmp)
```

```
tabs[, c("source", "dt.nm", "file")]

unlink(tmp)

## Typical SNFI use with a persistent cache directory.
## This can download data and may require mdbtools or an Access driver,
## so it is intentionally left as commented example code.
## cache <- tools::R_user_dir("basifoR", "cache")
## tabs4 <- listNFI_tables(28, nfi.nr = 4, dir = cache)
## head(tabs4)
## x4 <- readNFI(28, nfi.nr = 4, dt.nm = "PCMayores", dir = cache)
```

`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 descriptive metadata, its equation function, output column name, units, and fallback rule.
`track_provenance`	Add per-row provenance columns and audit metadata, including the method label, equation description, reference, and output description.
`...`	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. With `track_provenance = TRUE`, the `volume_meta` attribute also records each method's label, equation description, reference, units, scale, and output description.

Value

A data.frame with the requested volume outputs added. When `track_provenance = TRUE`, attribute `volume_meta` contains computational and descriptive method metadata.

Author(s)

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>)

See Also

nfiMetrics, snfi_volume_method_registry,

default_snfi_volume_equations 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. The legacy helper evaluates old equations with h in dm. Current basifoR tree metrics expose h in m, so make that metadata explicit before entering the compatibility path. 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.

Examples

```
x <- structure(
    data.frame(
        nfi.nr = 4,
        pr = 28,
```

```
        especie = 21,
        d = c(180, 260),
        h = c(9.5, 14.2),
        n = c(31.83, 14.15)
    ),
    class = c("nfiMetrics", "data.frame"),
    units = c(d = "mm", h = "m", n = "ha-1"),
    nfi.nr = 4
)

demo_registry <- snfi_volume_method_registry(list(
    VCC = list(
        output = "vcc_demo",
        fun = function(dbh_mm, h_m, pars) {
            pars$k[1] * dbh_mm^2 * h_m
        },
        raw_unit = "m3 tree-1",
        unit = "m3 tree-1",
        scale_to_m3 = 1,
        build_args = function(ctx, pars, resolved) {
            list(dbh_mm = ctx$d_mm, h_m = ctx$h_m, pars = pars)
        },
        fallback = function(ctx, pars, resolved) NA_real_,
        pars = data.frame(
            nfi.nr = 4,
            pr = 28,
            especie = 21,
            k = 1e-7
        )
    )
))

y <- metrics2Vol(
    x,
    parametro = "VCC",
    method_registry = demo_registry,
    keep.var = FALSE
)

y[, c("nfi.nr", "pr", "especie", "vcc_demo")]

## Real SNFI workflows may require local Access/mdbtools support or
## previously cached data, so they are intentionally not run here.
## z <- nfiMetrics(28, nfi.nr = 4, dir = tools::R_user_dir("basifoR", "cache"))
## metrics2Vol(z, parametro = c("VCC", "VSC"))
```

---

metrics2Vol_spatial *Tree-level volumes carrying a spatial sidecar*

Description

Compute tree-level SNFI volume variables while preserving plot geometry

This function mirrors the public arguments and calculations of `metrics2Vol` but adds an optional `spatial` sidecar. The volume computation itself is delegated to the original `metrics2Vol()`, so the equation registry, fallback logic, provenance, units, and volume metadata remain consistent with the non-spatial workflow.

Spatial information is prepared before the volume call by invoking `nfiMetrics_spatial()` with `spatial = "attribute"` when the input is not already an `"nfiMetrics"` object, and is copied back to the volume output after `metrics2Vol()` finishes. Thus the geometry column never travels through the volume-equation calculations. If `spatial = "sf"`, the function reconstructs an `sf` point object only at the end.

Usage

```
metrics2Vol_spatial(nfi,
    cub.met = "freq",
    parametro = c("VCC"),
    keep.var = TRUE,
    keep.legacy = FALSE,
    method_registry = snfi_volume_method_registry(),
    track_provenance = FALSE,
    ..., spatial = c("attribute",
        "sf", "none",
        "inherit"), var = c("d",
        "h", "ba", "n",
        "Hd", "Dd"),
    levels = c("esta",
        "espe"), design = snfi_design(),
    domheight_method = "Hd",
    domheight_registry = dominant_height_method_registry(),
    geometry.dt.nm = NULL,
    schema = NULL, coords = NULL,
    x.col = NULL, y.col = NULL,
    pr.col = NULL, plot.col = NULL,
    crs = NULL, coord.units = "m",
    geometry.source = c("auto",
        "snfi", "external",
        "none"), validate = TRUE,
    coord.nm = NULL,
    coord.factor = NULL,
    huso.method = c("auto",
        "candidate",
        "province", "none"),
    target.crs = NULL,
    boundary = TRUE,
    boundary.source = c("auto",
        "gisco", "gadm",
        "user", "none"),
```

```
    boundary.object = NULL,
    boundary.level = 2,
    boundary.path = tools::R_user_dir("basifoR",
        "cache"), boundary.ext = "json",
    boundary.version = "4.1",
    boundary.crs = NULL,
    infer.huso = NULL,
    na.action = c("keep",
        "drop", "error"))
```

Arguments

nfi	Input accepted by metrics2Vol. This can be a province code/name, a file path, a "readNFI" object, a "readNFI_spatial" object, a "nfiMetrics" object, or an "sf" object carrying plot-level SNFI records.
cub.met	Same as metrics2Vol(). Cubication selector used when several coefficient rows match.
parametro	Same as metrics2Vol(). One or more volume outputs to compute.
keep.var	Same as metrics2Vol(). Keep auxiliary coefficient columns when available.
keep.legacy	Same as metrics2Vol(). Also return the legacy volume estimate for backward compatibility.
method_registry	Same as metrics2Vol(). Registry that maps each requested output to its equation function, output column name, units, and fallback rule.
track_provenance	Same as metrics2Vol(). Add per-row provenance columns and audit metadata.
...	Additional arguments passed to nfiMetrics_spatial() when metrics must be computed before volume calculation. Typical examples are nfi.nr, dt.nm, dir, file_ext, and timeOut.
spatial	Spatial output mode. "attribute" keeps the ordinary tabular metrics2Vol() output and stores the plot geometry registry in attributes. "sf" reconstructs point geometry at the end. "none" runs the original non-spatial workflow. "inherit" returns "sf" for sf input, "attribute" for objects already carrying a sidecar, and "none" otherwise.
var	Forwarded to nfiMetrics_spatial() if the input is not already an "nfiMetrics" object.
levels	Forwarded to nfiMetrics_spatial(). Keeping the plot identifier is important for final sf reconstruction.
design	Forwarded to nfiMetrics_spatial().
domheight_method	Dominant-height method forwarded to nfiMetrics_spatial() or, when metrics are computed inside metrics2Vol(), to nfiMetrics().
domheight_registry	Named dominant-height registry created with dominant_height_method_registry().

`geometry.dt.nm` Optional table name used to build the geometry registry. When `NULL`, the default of `readNFI_spatial()` is used, normally the same table requested by `dt.nm`.

`schema` Optional external schema forwarded to `nfiMetrics_spatial()` and `readNFI_spatial()`.

`coords` Optional coordinate table forwarded to `readNFI_spatial()` when coordinates are stored outside the main table.

`x.col` Optional X/easting/longitude coordinate column for external coordinate tables.

`y.col` Optional Y/northing/latitude coordinate column for external coordinate tables.

`pr.col` Optional province, region, or stratum column used in `spatial` joins.

`plot.col` Optional plot identifier column used in `spatial` joins.

`crs` Optional CRS for external coordinates, for example an EPSG code such as `25830`.

`coord.units` Coordinate units recorded in the `spatial` sidecar when external coordinates are supplied.

`geometry.source`
Forwarded to `readNFI_spatial()`. Use `"snfi"` to force Spanish NFI coordinate discovery and `"external"` when coordinates come from the main table or a user-supplied coordinate table.

`validate` Warn about missing `spatial` helpers, failed sidecar creation, or failed sf reconstruction. Volume calculations still follow `metrics2Vol()`.

`coord.nm` Optional SNFI coordinate table name forwarded to `readNFI_spatial()`.

`coord.factor` Optional coordinate multiplier for SNFI coordinate tables. By default IFN2 kilometre coordinates are detected and converted to metres.

`huso.method` UTM-zone assignment method forwarded to `readNFI_spatial()`. `"auto"` uses direct `Huso`, then historical `huso1`/`huso2`/`huso3` + `CoorX`, then province fallback.

`target.crs` Optional common CRS for SNFI geometries. When `NULL`, mixed ED50 zones are transformed to EPSG:23030.

`boundary` `logical`. If `TRUE` (default), attach an optional administrative `boundary` sidecar for plot maps.

`boundary.source`
Boundary source passed to `readNFI_spatial()`. `"auto"` tries GISCO/NUTS before GADM for Spanish province-like inputs.

`boundary.object`
Optional user-provided `sf` polygon layer.

`boundary.level` GADM administrative level, used when `boundary.source` resolves to `"gadm"`.

`boundary.path` Cache directory for optional `boundary` downloads.

`boundary.ext` GADM extension used by `gadm_spatial()`.

`boundary.version`
GADM version used by `gadm_spatial()`.

`boundary.crs` CRS assigned to `boundary.object` when missing.

`infer.huso` Backward-compatible fallback argument forwarded to `readNFI_spatial()`.

`na.action` Used only when `spatial = "sf"`. It controls rows that do not match the plot geometry registry during final `sf` reconstruction.

Details

The default `spatial = "attribute"` returns a normal `data.frame` with `metrics2vol` metadata intact and an extra `attr(x, "nfi_geometry_registry")` sidecar. This is the safest mode for developing `dendroMetrics_spatial()` and `inventoryMetrics_spatial()` because no geometry column is passed through the volume internals. Use `spatial = "sf"` only when direct `spatial` output is needed at this stage.

Value

A `metrics2vol` object. With `spatial = "attribute"`, the result remains tabular and carries a plot-level geometry registry in `attr(x, "nfi_geometry_registry")` for later `*_spatial()` functions. With `spatial = "sf"`, the function attempts to return an `sf` object by joining plot geometries back to the tree-level volume rows. If the volume output no longer contains a plot identifier, the function warns and returns the tabular output with the sidecar instead.

Author(s)

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>)

Examples

```
toy <- data.frame(
    nfi.nr = 4,
    pr = 28,
    esta = c("P1", "P2"),
    especie = c(21, 21),
    Dn = c(180, 260),
    altura = c(9.5, 14.2),
    x = c(-3.70, -3.69),
    y = c(40.40, 40.41)
)
attr(toy, "nfi.nr") <- 4

demo_registry <- snfi_volume_method_registry(list(
    VCC = list(
        output = "vcc_demo",
        fun = function(dbh_mm, h_m, pars) {
            pars$k[1] * dbh_mm^2 * h_m
        },
        raw_unit = "m3 tree-1",
        unit = "m3 tree-1",
        scale_to_m3 = 1,
        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_,
        pars = data.frame(
            nfi.nr = 4,
            pr = 28,
            especie = 21,
            k = 1e-7
        )
    )
))

x <- metrics2Vol_spatial(
    toy,
    parametro = "VCC",
    method_registry = demo_registry,
    var = c("d", "h", "ba", "n"),
    levels = c("esta", "especie"),
    spatial = "attribute",
    geometry.source = "external",
    plot.col = "esta",
    x.col = "x",
    y.col = "y",
    crs = 4326,
    boundary = FALSE
)

inherits(x, "metrics2vol")
hasNFIgeometry_spatial(x)

## Spanish NFI: compute tree metrics, volumes, and carry geometry as an
## attribute sidecar.
## x <- metrics2Vol_spatial(28, nfi.nr = 3, dir = tempdir(),
##                          parametro = c("VCC", "VSC"))
## inherits(x, "metrics2vol")
## hasNFIgeometry_spatial(x)
## attr(x, "units")
## attr(x, "volume_meta")
## attr(x, "dominant_height_meta")

## Direct sf output for visual checks.
## xsf <- metrics2Vol_spatial(28, nfi.nr = 3, dir = tempdir(),
##                            parametro = c("VCC", "VSC"),
##                            spatial = "sf")
## inherits(xsf, "sf")

## Continue from an existing spatial nfiMetrics object.
## nm <- nfiMetrics_spatial(28, nfi.nr = 3, dir = tempdir())
## vv <- metrics2Vol_spatial(nm, parametro = "VCC")
## hasNFIgeometry_spatial(vv)

## Disable spatial behavior and recover the ordinary metrics2Vol workflow.
## y <- metrics2Vol_spatial(28, nfi.nr = 3, dir = tempdir(),
```

```
##                          spatial = "none")
```

`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 [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>)

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_dominant_height_method

Define one dominant-height computation method

Description

Create a compact method specification for registry-based dominant-height calculations.

Usage

```
new_dominant_height_method(output = "Hd",
    fun = NULL, fun_name = NULL,
    diameter_fun = NULL,
    diameter_fun_name = NULL,
    diameter_unit = "mm",
    diameter_equation = NULL,
    unit = "m", threshold = 100,
    equation = NULL,
    selection_rule = NULL,
    fallback = NULL,
    variables = c(h = "tree height in metres",
        d = "diameter at breast height in millimetres",
        n = "tree expansion factor in trees per hectare"),
    required_inputs = c("h",
        "d", "n"), reference = NULL,
    description = NULL)
```

Arguments

output	character(1). Name of the output column produced by the method.
fun	Optional function. Direct function used to compute dominant height.
fun_name	Optional character(1). Name of a function to resolve at run time when fun is not supplied.
diameter_fun	Optional paired function used to compute dominant diameter from d and n.
diameter_fun_name	Optional run-time name of the paired dominant-diameter function.
diameter_unit	character(1). Unit reported for dominant diameter.
diameter_equation	character(1). Human-readable dominant-diameter equation.
unit	character(1). Unit reported for the returned dominant-height value.
threshold	numeric(1). Reference number of trees per hectare used to define the dominant-height subset.
equation	character(1). Human-readable equation used by the method.
selection_rule	character(1). Human-readable rule used to select trees before applying the equation.
fallback	character(1). Human-readable description of what the method returns when the threshold cannot be reached.
variables	Named character. Meaning and expected units of the variables used by equation.
required_inputs	character. Standardized inputs required by the method.
reference	Optional character(1). Source or implementation reference.
description	Optional character(1). Short description of the method.

Details

The object does not evaluate dominant height by itself. It only

records the output name, function or function name, unit,

dominant-tree threshold, selection rule, equation, fallback rule,

variable definitions, and optional reference.

Value

A named list with class "dominant_height_method".

Author(s)

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>)

Examples

```
m <- new_dominant_height_method(
    fun_name = "domheight",
    equation = "Hd = sum(h_i * n_i) / sum(n_i)",
    selection_rule = "Sort trees by decreasing diameter and select dominant trees.",
    fallback = "Use the weighted mean height of all valid trees."
)
m$equation
```

---

new_external_schema *Define a schema for external inventory workflows*

Description

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

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 [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>)

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

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 [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>)

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

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	function. Returns a fallback value when the method cannot compute a raw result, for example because the function, parameters, or arguments are missing.
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 [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>)

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",
    "Dd"), levels = c("esta",
    "espe"), design = snfi_design(),
    domheight_method = "Hd",
    domheight_registry = dominant_height_method_registry(),
    ...)
```

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 m), 'ba' (basal area per tree in m^2), 'n' (trees per hectare), 'Hd' (dominant height in m), and 'Dd' (dominant diameter in mm). 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.
domheight_method	character(1). Dominant-height method code resolved through domheight_registry when 'Hd' is requested.
domheight_registry	Named registry created with dominant_height_method_registry.
...	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'. The active method is resolved through domheight_registry; inspect dominant_height_method_registry()$Hd to see the equation, selection rule, threshold, variables, and fallback.

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', 'Hd', or 'Dd' is requested. When 'Hd' is requested, inspect attr(x, 'dominant_height_meta') to see the method code, equation, selection rule, threshold, and fallback.

Author(s)

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>)

See Also

dendroMetrics, dbhMetric, readNFI, snfi_design, trees_per_ha, dominant_height_method_registry

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")
```

`nfiMetrics_spatial` *Tree-level metrics carrying a spatial sidecar*

Description

Compute tree-level Spanish NFI metrics while preserving plot geometry

This function mirrors the public arguments and calculations of `nfiMetrics` but adds an optional `spatial` sidecar. The metric computation itself is delegated to the original `nfiMetrics()`, so the returned diameter, height, basal-area, trees-per-hectare, dominant-height, dominant-diameter, unit, and `design` metadata remain consistent with the non-spatial workflow.

Spatial information is prepared before the metric call by invoking `readNFI_spatial()` with `spatial = "attribute"` and copied back to the metric output after `nfiMetrics()` finishes. Thus the geometry column never travels through the tree-metric calculations. If `spatial = "sf"`, the function reconstructs an `sf` point object only at the end.

Usage

```
nfiMetrics_spatial(nfi,
    var = c("d", "h",
        "ba", "n", "Hd",
        "Dd"), levels = c("esta",
        "espe"), design = snfi_design(),
    domheight_method = "Hd",
    domheight_registry = dominant_height_method_registry(),
    ..., spatial = c("attribute",
        "sf", "none",
        "inherit"), geometry.dt.nm = NULL,
    schema = NULL, coords = NULL,
    x.col = NULL, y.col = NULL,
    pr.col = NULL, plot.col = NULL,
    crs = NULL, coord.units = "m",
    geometry.source = c("auto",
        "snfi", "external",
        "none"), validate = TRUE,
    coord.nm = NULL,
    coord.factor = NULL,
    huso.method = c("auto",
        "candidate",
        "province", "none"),
    target.crs = NULL,
    boundary = TRUE,
    boundary.source = c("auto",
```

```
    "gisco", "gadm",
    "user", "none"),
boundary.object = NULL,
boundary.level = 2,
boundary.path = tools::R_user_dir("basifoR",
    "cache"), boundary.ext = "json",
boundary.version = "4.1",
boundary.crs = NULL,
infer.huso = NULL,
na.action = c("keep",
    "drop", "error"))
```

Arguments

nfi	character(1), "readNFI", "readNFI_spatial", or "sf" object. Province codes/names and file paths are first read through readNFI_spatial() when spatial output is requested. Objects that already carry attr(x, "nfi_geometry_registry") reuse that sidecar.
var	Same as nfiMetrics(). Metrics to compute: diameter 'd', height 'h', basal area 'ba', trees per hectare 'n', dominant height 'Hd', and dominant diameter 'Dd'.
levels	Same as nfiMetrics(). Column-name patterns used to keep grouping variables in the tree-level output. Spatial reconstruction to sf requires a plot identifier to remain in the output, so the default should usually be kept while developing the spatial chain.
design	Same as nfiMetrics(). Sampling design used to compute expansion factors for 'n', dominant height, and dominant diameter.
domheight_method	Dominant-height method code resolved through domheight_registry when 'Hd' is requested. The default matches nfiMetrics().
domheight_registry	Named dominant-height registry created with dominant_height_method_registry().
...	Additional arguments passed to readNFI_spatial() and then to nfiMetrics(). Typical examples are nfi.nr, dt.nm, dir, file_ext, and timeOut.
spatial	Spatial output mode. "attribute" keeps the ordinary tabular nfiMetrics() output and stores the plot geometry registry in attributes. "sf" reconstructs point geometry at the end. "none" runs the original non-spatial workflow. "inherit" returns "sf" for sf input, "attribute" for objects already carrying a sidecar, and "none" otherwise.
geometry.dt.nm	Optional table name used to build the geometry registry. When NULL, the default of readNFI_spatial() is used, normally the same table requested by dt.nm.
schema	Optional external schema forwarded to readNFI_spatial(). This lets compatible non-standard inventories define spatial metadata in schema$defaults$spatial while keeping the metric calculation unchanged.
coords	Optional coordinate table forwarded to readNFI_spatial() when coordinates are stored outside the main table.

x.col	Optional X/easting/longitude coordinate column for external coordinate tables.
y.col	Optional Y/northing/latitude coordinate column for external coordinate tables.
pr.col	Optional province, region, or stratum column used in spatial joins.
plot.col	Optional plot identifier column used in spatial joins.
crs	Optional CRS for external coordinates, for example an EPSG code such as 25830.
coord.units	Coordinate units recorded in the spatial sidecar when external coordinates are supplied.
geometry.source	Forwarded to readNFI_spatial(). Use "snfi" to force Spanish NFI coordinate discovery and "external" when coordinates come from the main table or a user-supplied coordinate table.
validate	Warn about missing spatial helpers, failed sidecar creation, or failed sf reconstruction. Metric calculations still follow nfiMetrics().
coord.nm	Optional SNFI coordinate table name forwarded to readNFI_spatial().
coord.factor	Optional coordinate multiplier for SNFI coordinate tables. By default IFN2 kilometre coordinates are detected and converted to metres.
huso.method	UTM-zone assignment method forwarded to readNFI_spatial(). "auto" uses direct Huso, then historical huso1/huso2/huso3 + CoorX, then province fallback.
target.crs	Optional common CRS for SNFI geometries. When NULL, mixed ED50 zones are transformed to EPSG:23030.
boundary	logical. If TRUE (default), attach an optional administrative boundary sidecar for plot maps.
boundary.source	Boundary source passed to readNFI_spatial(). "auto" tries GISCO/NUTS before GADM for Spanish province-like inputs.
boundary.object	Optional user-provided sf polygon layer.
boundary.level	GADM administrative level, used when boundary.source resolves to "gadm".
boundary.path	Cache directory for optional boundary downloads.
boundary.ext	GADM extension used by gadm_spatial().
boundary.version	GADM version used by gadm_spatial().
boundary.crs	CRS assigned to boundary.object when missing.
infer.huso	Backward-compatible fallback argument forwarded to readNFI_spatial().
na.action	Used only when spatial = "sf". It controls rows that do not match the plot geometry registry during final sf reconstruction.

Details

The default spatial = "attribute" returns a normal

data.frame with the nfiMetrics metadata intact

and an extra `attr(x, "nfi_geometry_registry")` sidecar.

This is the safest mode for developing

`metrics2Vol_spatial()` and `dendroMetrics_spatial()`

because no geometry column is passed through the metric

internals. Use `spatial = "sf"` only when direct `spatial`

output is needed at this stage.

Value

A `nfiMetrics` object. With `spatial = "attribute"`, the result remains tabular and carries a plot-level geometry registry in `attr(x, "nfi_geometry_registry")` for later `*_spatial()` functions. With `spatial = "sf"`, the function attempts to return an `sf` object by joining plot geometries back to the tree-level metric rows. If the metric output no longer contains a plot identifier, the function warns and returns the tabular output with the sidecar instead.

Author(s)

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>)

Examples

```
toy <- data.frame(
    nfi.nr = 4,
    pr = 28,
    esta = c("P1", "P1", "P2"),
    espe = c(21, 21, 25),
    Dn = c(120, 185, 260),
    altura = c(7.1, 9.4, 13.2),
    x = c(-3.70, -3.70, -3.69),
    y = c(40.40, 40.40, 40.41)
)
attr(toy, "nfi.nr") <- 4

x <- nfiMetrics_spatial(
    toy,
    var = c("d", "h", "ba", "n"),
    levels = c("esta", "espe"),
    spatial = "attribute",
    geometry.source = "external",
    plot.col = "esta",
    x.col = "x",
    y.col = "y",
    crs = 4326,
    boundary = FALSE
)

inherits(x, "nfiMetrics")
hasNFIgeometry_spatial(x)
```

```
 ## Spanish NFI: compute metrics and carry geometry as an attribute sidecar.
 ## x <- nfiMetrics_spatial(28, nfi.nr = 3, dir = tempdir())
 ## inherits(x, "nfiMetrics")
 ## hasNFIgeometry_spatial(x)
## reg <- getNFIgeometry_spatial(x)
## reg$geometry
## attr(x, "dominant_height_meta")

## Direct sf output for visual checks.
## xsf <- nfiMetrics_spatial(28, nfi.nr = 3, dir = tempdir(),
##                           spatial = "sf")
## inherits(xsf, "sf")

## Disable spatial behavior and recover the ordinary nfiMetrics workflow.
 ## y <- nfiMetrics_spatial(28, nfi.nr = 3, dir = tempdir(),
 ##                         spatial = "none")
```

`plot.inventoryMetrics_spatial` *plot inventoryMetrics spatial*

Description

Plot a spatial `inventoryMetrics` object.

Usage

```
## S3 method for class 'inventoryMetrics_spatial'
plot(x,
    y = NULL, ...)
```

Arguments

`x`	A spatial `inventoryMetrics_spatial` object to plot.
`y`	Optional character vector of variable names or numeric vector of column positions to plot; `NULL` uses the default numeric variables.
`...`	Additional arguments passed to the internal spatial plotting engine, including options such as `vars`, `boundary`, and `engine`.

Author(s)

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>)

plot.metrics2Vol_spatial

plot metrics2Vol spatial

Description

Plot a spatial `metrics2Vol` object.

Usage

```
## S3 method for class 'metrics2Vol_spatial'
plot(x,
    y = NULL, ...)
```

Arguments

x	A spatial `metrics2Vol_spatial` object to plot.
y	Optional character vector of variable names or numeric vector of column positions to plot; `NULL` uses the default numeric variables.
...	Additional arguments passed to the internal spatial plotting engine, including options such as `vars`, `boundary`, and `engine`.

Author(s)

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>)

plot.nfiMetrics_spatial

plot nfiMetrics spatial

Description

Plot a spatial `nfiMetrics` object.

Usage

```
## S3 method for class 'nfiMetrics_spatial'
plot(x,
    y = NULL, ...)
```

Arguments

x	A spatial nfiMetrics_spatial object to plot.
y	Optional character vector of variable names or numeric vector of column positions to plot; NULL uses the default numeric variables.
...	Additional arguments passed to the internal spatial plotting engine, including options such as vars, boundary, and engine.

Author(s)

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>)

---

plot.readNFI_spatial *plot readNFI spatial*

Description

Plot a spatial readNFI object.

Usage

```
## S3 method for class 'readNFI_spatial'
plot(x,
    y = NULL, ...)
```

Arguments

x	A spatial readNFI_spatial object to plot.
y	Optional character vector of variable names or numeric vector of column positions to plot; NULL uses the default numeric variables.
...	Additional arguments passed to the internal spatial plotting engine, including options such as vars, boundary, and engine.

Author(s)

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>)

print.concentric_design *Print a concentric plot design*

Description

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 [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>)

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*

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 [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>)

Examples

```
sch <- new_external_schema(
    colmap = list(plot = "plot_id", d = "dbh_mm", h = "height_m"),
    units = list(d = "mm", h = "m"),
    levels = "plot_id"
)

print.external_schema(sch)
```

print.inventory_design *Print a generic inventory plot design*

Description

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 [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>)

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

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 [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>)

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)
```

`readNFIcoords` *readNFIcoords*

Description

readNFIcoords: read an NFI table and append coordinate columns.

Read one Spanish National Forest Inventory table, locate the matching plot-coordinate table, and append UTM coordinates and CRS metadata as ordinary columns. The wrapper discovers files with `listNFI_tables()` once and then reuses the local files present in `dir`; it does not intentionally download the same source twice.

Usage

```
readNFIcoords(nfi, nfi.nr = 4,
    dt.nm = "PCMayores",
    coord.nm = NULL,
    file_ext = NULL,
    file_name = NULL,
    validate = TRUE,
    ..., x.name = "x",
    y.name = "y", huso.name = "huso",
    huso.source.name = "huso_source",
    datum.name = "datum",
    epsg.name = "epsg",
    crs.name = "crs",
    overwrite = FALSE,
    keep.raw.coords = FALSE,
    infer.huso = FALSE)
```

Arguments

`nfi`	Input accepted by `listNFI_tables` and `readNFI`: province identifier, zip archive, decompressed files, or a previously loaded data frame.
`nfi.nr`	`integer`. Inventory stage: 2, 3, or 4.
`dt.nm`	Table to import and preserve as the main output.
`coord.nm`	Optional coordinate table name. If `NULL`, the function validates `DATEST*` in `listNFI_tables()` but passes the generic `DATEST` stem for IFN2, and uses `PCDatosMap/Listado definitivo` for IFN3/IFN4 when available.
`file_ext`	Optional file extension passed to `listNFI_tables()`.
`file_name`	Optional file name filter for the main-table read. Coordinate discovery ignores this argument.
`validate`	`logical`. Validate coordinate-table discovery and warn about unmatched plot keys.
`...`	Additional arguments passed to `listNFI_tables()`, such as `dir` or `timeOut`. The resulting local paths are then passed to `readNFI()`.
`x.name`	Output X-coordinate column name, in metres.
`y.name`	Output Y-coordinate column name, in metres.
`huso.name`	Output UTM-zone column name when available.

`huso.source.name`
: Output column describing whether Huso comes from the coordinate table or was inferred from province code.

`datum.name`
: Output geodetic datum column name.

`epsg.name`
: Output EPSG code column name when known.

`crs.name`
: Output CRS label column name when known.

`overwrite`
: Allow overwriting existing coordinate columns.

`keep.raw.coords`
: Keep raw source coordinates before conversion.

`infer.huso`
: Fill missing UTM zones from province code only when requested. A Huso column in the coordinate table has precedence. Province-filled values are marked in huso_source.

Details

The wrapper first calls `listNFI_tables()` to discover and, when necessary, fetch the available files. After that discovery step, the main table and the coordinate table are read from the local paths returned by `listNFI_tables()`, respecting the package cache philosophy and avoiding a second intentional download.

The argument `dt.nm` controls the main table. The coordinate table only supplies plot positions. For IFN2 the coordinate table is detected as the generic `DATEST` stem after validating available `DATESTXX` files. For IFN3 and IFN4 the function looks for `PCDatosMap` or `Listado definitivo`, unless `coord.nm` is supplied explicitly.

The output remains a regular `data.frame` enriched with coordinate and CRS metadata columns. Use `readNFIsf()` when an `sf` geometry column is preferred.

Value

A `readNFI` data frame enriched with plot-level spatial coordinates. The imported table named by `dt.nm` remains the main output; coordinate fields are joined by province and plot identifier.

Author(s)

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>)

Examples

```
## Synthetic example for the joining helper.
trees <- data.frame(
    nfi.nr = 3,
    pr = 28,
    Estadillo = c(10, 10, 11),
    Especie = c(21, 21, 25)
)
attr(trees, "nfi.nr") <- 3
```

```
coords <- data.frame(
    Provincia = 28,
    Estadillo = c(10, 11),
    CoorX = c(440000, 441000),
    CoorY = c(4488000, 4489000),
    Huso = 30
)

addNFIcoords(trees, coords)

## Real use with a persistent cache directory. This may download data
## the first time and reuse the local files later.
## cache <- tools::R_user_dir("basifoR", "cache")
## x <- readNFIcoords(45, nfi.nr = 4, dt.nm = "PCMayores", dir = cache)
## unique(x[, c("huso", "huso_source", "datum", "epsg")])
```

`readNFIsf` *readNFIsf*

Description

readNFIsf: read a Spanish NFI table as an sf point data frame.

Read one Spanish National Forest Inventory table and return an `sf` object with plot geometries. The table requested by `dt.nm` becomes the attribute table. Coordinates come from the appropriate plot-coordinate table detected from `listNFI_tables()`.

Usage

```
readNFIsf(nfi, nfi.nr = 4,
    dt.nm = "PCMayores",
    coord.nm = NULL,
    file_ext = NULL,
    file_name = NULL,
    validate = TRUE,
    ..., crs = NULL,
    keep.coord.meta = FALSE,
    mixed.crs = c("na",
        "error"), na.action = c("keep",
        "drop", "error"),
    infer.huso = NULL)
```

Arguments

`nfi`	Input accepted by `listNFI_tables`: province identifier, zip archive, URL, or decompressed files.
`nfi.nr`	`integer`. Inventory stage: 2, 3, or 4.
`dt.nm`	Table to import and preserve as the sf attribute table.

coord.nm	Optional coordinate table name. If NULL, the function detects DATEST for IFN2 and PCDatosMap/Listado definitivo for IFN3/IFN4.
file_ext	Optional file extension passed to listNFI_tables().
file_name	Optional file name filter for the main-table read. Coordinate discovery ignores this argument.
validate	logical. Validate coordinate-table discovery and warn about unmatched plot keys.
...	Additional arguments passed to listNFI_tables(), such as dir or timeOut.
crs	Optional CRS passed to sf::st_as_sf. When NULL, the function uses the unique EPSG derived from metadata.
keep.coord.meta	Keep Huso/datum/EPSG metadata columns in the sf attribute table. Otherwise they are stored in attr(x, "coord_reference").
mixed.crs	What to do when rows imply more than one EPSG code.
na.action	Handling of rows that do not match a coordinate record.
infer.huso	Fill missing UTM zones from province code. By default, TRUE for IFN2 and FALSE for IFN3/IFN4.

Details

This is the high-level spatial reader. It mirrors the usual readNFI() workflow for the selected dt.nm table, but returns an sf point data frame. It should complement rather than replace readNFI(), because the return type changes and a coordinate table must be available.

The function calls listNFI_tables() once, reuses the local files in dir, reads the requested table as attributes, reads the coordinate table separately, and delegates the geometry construction to addNFIsf().

By default, infer.huso is TRUE for IFN2 and FALSE for IFN3/IFN4. This prevents silently assigning uncertain UTM zones for inventories where a province-level fallback may be less precise than the original map-sheet reference.

Value

An sf data frame. The imported table named by dt.nm remains the attribute table; plot coordinates are stored in the geometry column.

Author(s)

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>)

Examples

```
if (interactive()) {
    ## Real use with a persistent cache directory. This may download data
    ## the first time and reuse the local files later.
```

```
    cache <- tools::R_user_dir("basifoR", "cache")
    x <- readNFIsf(45, nfi.nr = 4, dt.nm = "PCMayores", dir = cache)
    sf::st_crs(x)
    attr(x, "coord_reference")
}

## Visual check against a province boundary, if optional packages exist.
## if (interactive() &&
##     requireNamespace("geodata", quietly = TRUE) &&
##     requireNamespace("terra", quietly = TRUE)) {
##     gadm <- geodata::gadm("ESP", level = 2, path = tempdir())
##     gadm_sf <- sf::st_as_sf(gadm)
##     plot(sf::st_geometry(gadm_sf))
##     plot(sf::st_geometry(sf::st_transform(x, sf::st_crs(gadm_sf))),
##          add = TRUE, pch = 16, cex = 0.3)
## }
```

`readNFI_spatial` *readNFI spatial*

Description

Read SNFI data and store plot geometry as an attribute sidecar

This function is a conservative `spatial` companion to `readNFI()`. It keeps the ordinary tabular output of `readNFI()` and stores a plot-level geometry registry in `attr(x, "nfi_geometry_registry")`. By default, the returned object remains a data frame and carries geometry as an attribute sidecar. If `spatial = "sf"`, the function returns an `sf` object by joining the sidecar geometry back to each row. The attribute-sidecar default lets downstream metric functions carry `spatial` information without pushing a geometry column through tabular routines.

Usage

```
readNFI_spatial(nfi,
    nfi.nr = 4, dt.nm = "PCMayores",
    file_ext = NULL,
    file_name = NULL,
    ..., spatial = c("attribute",
        "sf", "none"),
    geometry.dt.nm = dt.nm,
    coord.nm = NULL,
    schema = NULL, coords = NULL,
    x.col = NULL, y.col = NULL,
    pr.col = NULL, plot.col = NULL,
    crs = NULL, coord.units = "m",
    coord.accuracy.note = NULL,
    geometry.source = c("auto",
        "snfi", "external",
        "none"), validate = TRUE,
```

```
    coord.factor = NULL,
    huso.method = c("auto",
        "candidate",
        "xgap", "province",
        "none"), huso.candidates = NULL,
    repair.table.huso = TRUE,
    target.crs = NULL,
    boundary = FALSE,
    boundary.source = c("auto",
        "gisco", "gadm",
        "user", "none"),
    boundary.object = NULL,
    boundary.level = 2,
    boundary.path = tools::R_user_dir("basifoR",
        "cache"), boundary.ext = "json",
    boundary.version = "4.1",
    boundary.crs = NULL,
    infer.huso = NULL)
```

Arguments

nfi	Inventory source passed to `readNFI()`: province code or name, local/remote zip archive, decompressed files, or a data frame.
nfi.nr	Spanish NFI stage. Use `2`, `3`, or `4`.
dt.nm	Main table to read and return as tabular data.
file_ext	Optional file extension passed to `readNFI()` and `readNFIsf()` when the input must be fetched or filtered.
file_name	Optional file-name filter passed to `readNFI()`. The `spatial` registry reads coordinates independently.
...	Additional arguments passed to `readNFI()` and `readNFIsf()`, such as `dir` or `timeOut`.
spatial	`"attribute"` stores a plot-level `spatial` registry in an attribute and keeps a tabular output. `"sf"` returns an `sf` object by joining the sidecar geometry to each row. `"none"` returns the ordinary `readNFI()` output.
geometry.dt.nm	Table used to build the geometry registry. In most SNFI workflows this can match `dt.nm`.
coord.nm	Optional SNFI coordinate table name. If `NULL`, the function detects `DATEST` for IFN2 and `PCDatosMap`/`Listado definitivo` for IFN3/IFN4.
schema	Optional `external_schema`. For external inventories, `spatial` metadata can be stored in `schema$defaults$spatial`, for example a named `list` containing plot, x, y, and `crs` entries. Explicit coordinate arguments below override `schema` values.
coords	Optional external coordinate table. Use this for non- Spanish inventories when coordinates live in a separate table. It can be a data frame or an `sf` object.
x.col	Optional X/easting/longitude column for external coordinate tables. If `NULL`, common names such as `x`, `coord_x`, `easting`, and `longitude` are detected.

`y.col` Optional Y/northing/latitude column for external coordinate tables. If `NULL`, common names such as `y`, `coord_y`, `northing`, and `latitude` are detected.

`pr.col` Optional province, region, or stratum key for external coordinate matching. It is not required for one-region external inventories.

`plot.col` Optional plot identifier column for external coordinate matching. If `NULL`, common plot names such as `plot_id`, `sample_id`, and `site_id` are detected.

`crs` Optional CRS for external coordinates, passed to `sf::st_as_sf()`. Use an EPSG code such as `25830` or `4326`.

`coord.units` Coordinate units recorded in the external `spatial` sidecar metadata.

`coord.accuracy.note`
Optional note describing the accuracy or provenance of user-supplied coordinates. This is most useful when replacing public inventory coordinates with field-validated plot-centre coordinates. Spanish NFI public coordinates receive an automatic accuracy note.

`geometry.source`
Source used to build the sidecar. `"auto"` first uses explicit external coordinates when available, then tries SNFI geometry only for province-like sources.

`validate` Warn when duplicate plot geometries are found or when geometry cannot be attached.

`coord.factor` Optional coordinate multiplier for SNFI coordinate tables. By default IFN2 kilometre coordinates are detected and converted to metres.

`huso.method` Method for assigning SNFI UTM zones when the coordinate table does not contain a direct `Huso` column. `"auto"` uses table Huso, then the historical `huso1`/`huso2`/`huso3` + `CoorX` rule, then province fallback.

`huso.candidates`
Optional candidate UTM zones for SNFI Huso inference. Use, for example, `c(29, 30)` for provinces that span zones 29 and 30 when no Huso columns are available.

`repair.table.huso`
`logical`. If `TRUE`, direct table-supplied Huso values are checked against the candidate-zone `CoorX` threshold rule before building geometry. This repairs boundary-zone cases such as plots stored in the neighbouring UTM zone.

`target.crs` Optional common CRS for projected SNFI geometries. When `NULL`, a single source EPSG is kept, or mixed ED50 zones are transformed to EPSG:23030.

`boundary` `logical`. If `TRUE`, attach an optional `boundary` sidecar in `attr(x, "nfi_boundary")`. Spanish NFI province-like inputs can use built-in GISCO/GADM download helpers; external inventories must provide `boundary.object`.

`boundary.source`
Boundary source. `"auto"` uses a user object when supplied and otherwise tries GISCO/NUTS before GADM for Spanish province-like inputs.

`boundary.object`
Optional user-provided `sf` polygon or multipolygon layer. External inventories must use this route; automatic built-in `boundary` selection is limited to Spanish NFI inputs.

boundary.level	GADM administrative level used when boundary.source = "gadm". GISCO uses NUTS-3.
boundary.path	Cache directory used for optional boundary downloads.
boundary.ext	GADM extension used by gadm_spatial().
boundary.version	GADM version used by gadm_spatial().
boundary.crs	CRS assigned to boundary.object when it lacks CRS metadata. It is required when the supplied object has no CRS.
infer.huso	Backward-compatible argument forwarded to the fallback readNFIsf() route. The new SNFI table route prefers huso.method.

Details

For Spanish NFI sources, the coordinate note is stage-specific and is stored in attr(x, "nfi_geometry_registry")$coord_
IFN2 coordinates are treated as official public locations derived from the kilometre UTM sampling grid. IFN3 coordinates are treated as official cartographic plot-location coordinates in UTM metres; when their Huso is not explicit, CRS metadata may be reconstructed from candidate Huso fields, coordinate rules, or fallback methods. IFN4 CoorX/ CoorY are treated as official public coordinates available before fieldwork for the plot centre. These stage notes document coordinate interpretation; they do not claim field-validated exact plot-centre precision.

External inventories or user-supplied coordinate tables do not receive an automatic coordinate-accuracy note because basifoR cannot know their measurement protocol or precision. If better coordinates become available, for example field GPS or institutional plot-centre coordinates, pass them through coords, x.col, y.col, plot.col, and optionally pr.col, with geometry.source = "external" and provide coord.accuracy.note only when the coordinate source is known. This replaces the spatial sidecar while keeping the ordinary readNFI() table and downstream metric calculations unchanged.

Value

A data frame with the same tabular content returned by readNFI(). When spatial = "attribute", the object also carries a plot-level geometry sidecar in attr(x, "nfi_geometry_registry"). The sidecar stores one geometry per inventory plot, CRS metadata, coordinate provenance, and normalized join keys for later *_spatial() metric functions.

Author(s)

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>)

Examples

```
toy <- data.frame(
    nfi.nr = 4,
    pr = 28,
    esta = c("P1", "P1", "P2"),
    espe = c(21, 21, 25),
```

```
    Dn = c(120, 185, 260),
    altura = c(7.1, 9.4, 13.2),
    x = c(-3.70, -3.70, -3.69),
    y = c(40.40, 40.40, 40.41)
)
attr(toy, "nfi.nr") <- 4

x <- readNFI_spatial(
    toy,
    spatial = "attribute",
    geometry.source = "external",
    plot.col = "esta",
    x.col = "x",
    y.col = "y",
    crs = 4326,
    boundary = FALSE
)

hasNFIgeometry_spatial(x)
names(getNFIgeometry_spatial(x))

## Keep the ordinary readNFI() output but carry plot geometry in attributes.
## cache <- tools::R_user_dir("basifoR", "cache")
## x <- readNFI_spatial(28, nfi.nr = 3, dt.nm = "PCMayores", dir = cache)
## is.data.frame(x)
## hasNFIgeometry_spatial(x)
## reg <- getNFIgeometry_spatial(x)
## reg$geometry

## Return a true sf object when direct spatial output is wanted.
## xsf <- readNFI_spatial(28, nfi.nr = 3, dt.nm = "PCMayores",
##                        dir = cache, spatial = "sf")
## inherits(xsf, "sf")

## External inventory: keep tabular data but store geometry sidecar.
## ext <- data.frame(plot_id = 1:2, x = c(440000, 441000),
##                   y = c(4488000, 4489000), d = c(20, 30))
## y <- readNFI_spatial(ext, spatial = "attribute",
##                      geometry.source = "external",
##                      plot.col = "plot_id", x.col = "x", y.col = "y",
##                      crs = 25830)
## hasNFIgeometry_spatial(y)
##
## External inventory with schema-stored spatial metadata:
## sch <- new_external_schema(
##     colmap = list(plot = "plot_id", d = "d"),
##     units = list(d = "cm"),
##     levels = "plot_id",
##     defaults = list(spatial = list(plot = "plot_id", x = "x", y = "y", crs = 25830))
## )
## y2 <- readNFI_spatial(ext, spatial = "attribute", schema = sch)
## hasNFIgeometry_spatial(y2)
##
```

```
## Replacing public SNFI coordinates with improved coordinates uses the
## same external sidecar mechanism.  The improved table must contain the
## inventory plot keys and coordinate columns.
## improved <- data.frame(pr = 28, Estadillo = c(1, 2),
##                        x_gps = c(440000, 441000),
##                        y_gps = c(4488000, 4489000))
 ## z <- readNFI_spatial(28, nfi.nr = 4, spatial = "attribute",
 ##                      geometry.source = "external", coords = improved,
 ##                      pr.col = "pr", plot.col = "Estadillo",
 ##                      x.col = "x_gps", y.col = "y_gps", crs = 25830,
 ##                     coord.accuracy.note = "Field-validated plot-centre coordinates.")
```

`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 [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>)

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`	Optional named list of SNFI method definitions to merge with the defaults; top-level names identify outputs such as `"V"`, `"VCC"`, or `"VSC"`.

Details

Each registry entry is a named list that typically contains descriptive fields such as `label`, `equation`, `reference`, and `description`, together with computational 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.

When overriding a computational method, also override its descriptive metadata so reported `equations` and references remain accurate.

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:

`label` Human-readable name of the volume method.

`equation` Concise description of the equation or computation.

`reference` Source associated with the method.

`description` Short explanation of the returned quantity.

`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 [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>)

Examples

```
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

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 [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>)

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

Description

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 [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>)

Examples

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

---

trees_per_ha.inventory_design

Compute trees per hectare for generic inventory designs

Description

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 [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>)

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 [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>)

`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(`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 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 [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>)

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*

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 [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>)

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

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 [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>)

Index

addNFIcoords, 3
addNFIsf, 5
asNFI_spatial_sf, 7

copyNFIboundary_spatial, 8
copyNFIgeometry_spatial, 9
copyNFIspatial_sidecars, 9

dbhMetric, 10
default_dominant_height_methods, 11
default_external_volume_methods, 12
default_snfi_volume_equations, 13
dendroMetrics, 13, *35*, *90*, *93*
dominant_height_method_registry, 16, *18*, *62*

external_dendroMetrics, 22, *91*
external_volume_method_registry, *12*, 27
externalMetrics, 17
externalMetrics2Vol, *12*, 19

fetchNFI, 28, *32*, *44*, *74*
filter_gadm_province_spatial, 29

gadm_spatial, 31
getNFI, 32
getNFIboundary_spatial, 33
getNFIgeometry_spatial, 34

hasNFIboundary_spatial, 34
hasNFIgeometry_spatial, 35

inventoryMetrics, 35, *39*, *68*, *92*
inventoryMetrics_spatial, 38

listNFI_tables, 44, *76*, *78*
Logic, *15*

metrics2Vol, *13, 14*, 46, *49, 50*, *69*
metrics2Vol_spatial, 48

new_concentric_design, 54, *71*, *88*
new_dominant_height_method, 55
new_external_schema, 57, *72*
new_inventory_design, *54*, 58, *89*
new_volume_method, *12*, 60
nfiMetrics, *13, 14*, 62, *64*, *69*
nfiMetrics_spatial, 64

plot.inventoryMetrics_spatial, 68
plot.metrics2Vol_spatial, 69
plot.nfiMetrics_spatial, 69
plot.readNFI_spatial, 70
print.concentric_design, 71
print.external_schema, 72
print.inventory_design, 73

readNFI, *3*, *6*, *13, 14*, *32*, *62*, *70*, 74, *76*
readNFI_spatial, 80
readNFIcoords, 75
readNFIsf, 78

snfi_design, *62*, 85, *88*
snfi_volume_method_registry, 86

timeout, *28*
trees_per_ha, *62*, *85*, 87
trees_per_ha.concentric_design, 88
trees_per_ha.inventory_design, 89

update, *35*
update.dendroMetrics, 90
update.external_dendroMetrics, 91
update.inventoryMetrics, 92
update.list, 93