Package 'ForestElementsR'

February 7, 2025

Title Data Structures and Functions for Working with Forest Data

Version 2.1.0

Date 2025-02-07

Description Provides generic data structures and algorithms for use with forest mensuration data in a consistent framework. The functions and objects included are a collection of broadly applicable tools. More specialized applications should be implemented in separate packages that build on this foundation. Documentation about 'ForestElementsR' is provided by three vignettes included in this package. For an introduction to the field of forest mensuration, refer to the textbooks by Kershaw et al. (2017) <doi:10.1002/9781118902028>, and van Laar and Akca (2007) <doi:10.1007/978-1-4020-5991-9>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports magrittr, ggplot2, tibble, dplyr, purrr, rlang, Rdpack, tidyr, vctrs, stringr, tidyselect (>= 1.2.0), doBy

RdMacros Rdpack

Depends R (>= 4.3), sf

LazyData true

Suggests knitr, rmarkdown, testthat (>= 3.2.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Peter Biber [aut, cre, cph] (<https://orcid.org/0000-0002-9700-8708>), Astor Toraño Caicoya [aut] (<https://orcid.org/0000-0002-9658-8990>), Torben Hilmers [ctb] (<https://orcid.org/0000-0002-4982-8867>)

Maintainer Peter Biber <p.biber@tum.de>

Repository CRAN

Date/Publication 2025-02-07 13:00:07 UTC

Contents

`age_d_gnfi3` *Inverse Stem Diameter Growth Model of the 3rd German National Forest Inventory (2012)*

Description

Inverse tree diameter growth model of the third German National Forest Inventory of 2012 (Riedel et al. 2017). Allows to estimate a tree's age at any dbh if its dbh is known at a given age.

Usage

```
age_d_gnfi3(species_id, dbh_cm, dbh_cm_known, age_yr_known)
```

Arguments

`species_id`	Vector of species id's preferably following the *ger_nfi_2012* species coding. Ideally, these species_id's are provided as a `fe_species_ger_nfi_2012` object. See Details for how other species codings are handled.
`dbh_cm`	Single numeric value or vector of dbh (cm) for which the age is to be estimated
`dbh_cm_known`	Vector of known dbh (cm) values at age `age_yr_known`
`age_yr_known`	Vector of ages (years) for which the dbh `d_cm_known` is known

Details

Originally, the function was parameterized for species and species groups corresponding to the national forest inventory's species coding (fe_species_ger_nfi_2012). We have attributed in addition these the original parameters also to the species codings fe_species_tum_wwk_short, and fe_species_bavrn_state_short. When called with a given species coding, the function will try to use the "nearest" of these three alternatives. Fallback option is the attempt to use fe_species_tum_wwk_short.

Value

A single age value or vector of age values corresponding to dbh_cm

References

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

See Also

Other growth functions: age_h_gnfi3(), d_age_gnfi3(), h_age_gnfi3()

Examples

```
# A Norway spruce has a diameter of 17.5 cm at age 45. Estimate
# its age at dbh 20.6 cm
age_d_gnfi3(10, 20.6, 17.5, 45) # 10 is ger_nfi_2012 code for Norway spruce

# Do the same backward in time, age at dbh 12.4 cm
age_d_gnfi3(10, 12.4, 17.5, 45)

# Apply for more than one tree, different species, same age
d_known   <- c(23.1, 16.2, 35.2, 19.3, 21.8)
d_age     <- c(27.0, 19.0, 40.8, 22.9, 25.8)
species   <- as_fe_species_tum_wwk_short(c(3, 3, 3, 6, 6))
age_d_gnfi3(
  species, dbh_cm = d_age, dbh_cm_known = d_known, age_yr_known = 40
)
```

---

age_h_gnfi3 *Inverse Tree Height Growth Model of the 3rd German National Forest Inventory (2012)*

Description

Inverse tree height growth model of the third German National Forest Inventory of 2012 (Riedel et al. 2017). Allows to estimate a tree's age at any height if its height is known at a given age.

Usage

```
age_h_gnfi3(species_id, h_m, h_m_known, age_yr_known)
```

Arguments

species_id	Vector of species id's preferably following the *ger_nfi_2012* species coding. Ideally, these species_id's are provided as a fe_species_ger_nfi_2012 object. See Details for how other species codings are handled.
h_m	Single numeric value or vector of tree heights (m) for which the age is to be estimated
h_m_known	Vector of known height (m) values at age age_yr_known
age_yr_known	Vector of ages (years) for which the height h_m_known is known

Details

Originally, the function was parameterized for species and species groups corresponding to the national forest inventory's species coding (fe_species_ger_nfi_2012). We have attributed in addition these the original parameters also to the species codings fe_species_tum_wwk_short, and fe_species_bavrn_state_short. When called with a given species coding, the function will try to use the "nearest" of these three alternatives. Fallback option is the attempt to use fe_species_tum_wwk_short.

Value

A single age value or vector of age values corresponding to h_m

References

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

See Also

Other growth functions: age_d_gnfi3(), d_age_gnfi3(), h_age_gnfi3()

Examples

```
# A European beech has a height of 25.2 m at age 75. Estimate
# its age at height 26.8 m
age_h_gnfi3(100, 26.8, 25.2, 75) # 100 is ger_nfi_2012 code for E. beech

# Do the same backward in time, age at height 21.7 m
age_h_gnfi3(100, 21.7, 25.2, 75)

# Apply for more than one tree, different species, same age
h_known   <- c(23.1, 16.2, 35.2, 19.3, 21.8)
h_age     <- c(25.5, 18.2, 38.0, 21.6, 24.2)
species   <- as_fe_species_tum_wwk_short(c(3, 3, 3, 6, 6))
age_h_gnfi3(
```

```
  species, h_m = h_age, h_m_known = h_known, age_yr_known = 60
)
```

`assmann_layers` *Attribute Tree Heights to Layers After Ernst Assmann*

Description

Tree heights are attributed to three layers as proposed by Assmann (1961). The layers are called Top, T, Middle, M, and Bottom, B, and correspond to >80%, >50%, and >0% of a reference height (usually the height of the highest tree in the stand of interest).

Usage

```
assmann_layers(heights, reference_height = NULL)
```

Arguments

`heights`	Vector of tree heights
`reference_height`	Reference height for the 100% level. If `NULL` (default), the maximum of `heights` will be used as reference height.

Value

An ordered factor of T, M, B values, corresponding to `heights` in the order as `heights` was provided.

References

Assmann E (1961). *Waldertragskunde. Organische Produktion, Struktur, Zuwachs und Ertrag von Waldbestaenden*. BLV Verlagsgesellschaft, Muenchen, Bonn, Wien.

See Also

Other structure and diversity: `shannon_index()`, `species_profile()`

Examples

```
# Monospecific stand
trees <- norway_spruce_1_fe_stand$trees
assmann_layers(trees$height_m)

# Selection forest
trees <- selection_forest_1_fe_stand$trees
assmann_layers(trees$height_m)
```

`as_fe_species_bavrn_state` *Cast Appropriate Objects Into a* **fe_stand_bavrn_state** *Species Class Object*

Description

If the cast is forward ambiguous, the function terminates with an error. "Forward ambiguous" means that one code in the original object corresponds to more than one codes in the goal coding. If the cast loses information, a warning is raised, but the cast is performed. "Information loss" in this context means that several codes from the orginal coding correspond to only one code in the goal coding.

Usage

```
as_fe_species_bavrn_state(x)
```

Arguments

`x`	The object to be cast, either a vector of types `integer`, `double`, or `character` or an object of one of the supported **fe_species** classes

Details

Note that a cast where only one species id from the original coding translates in a goal coding which represents a group of species is NOT considered losing information (i.e. backward ambiguous), because of the 1:1 match in the constellation of the specific cast.

Value

If a meaningful cast is possible, an `fe_species_bavrn_state` object is returned

Examples

```
as_fe_species_bavrn_state(c(10L, 40L, 40L, 20L)) # integer
as_fe_species_bavrn_state(c(10, 40, 40, 20)) # double
as_fe_species_bavrn_state(c("10", "40", "40", "20")) # character

# cast other fe_species classes
as_fe_species_bavrn_state(
  fe_species_tum_wwk_short(as.character(c(1, 1, 1, 3, 3, 5)))
)
as_fe_species_bavrn_state(
  fe_species_ger_nfi_2012(as.character(c(20, 20, 10, 30, 30, 100)))
)

# display the casting result in terms of scientific species names
as_fe_species_bavrn_state(c(10L, 40L, 40L, 20L)) |> format("sci")
```

as_fe_species_bavrn_state_short *Cast Appropriate Objects Into a* **fe_stand_bavrn_state_short** *Species Class Object*

Description

If the cast is forward ambiguous, the function terminates with an error. "Forward ambiguous" means that one code in the original object corresponds to more than one codes in the goal coding. If the cast loses information, a warning is raised, but the cast is performed. "Information loss" in this context means that several codes from the orginal coding correspond to only one code in the goal coding.

Usage

```
as_fe_species_bavrn_state_short(x)
```

Arguments

x	The object to be cast, either a vector of types `integer`, `double`, or `character` or an object of one of the supported **fe_species** classes

Details

Note that a cast where only one species id from the original coding translates in a goal coding which represents a group of species is NOT considered losing information (i.e. backward ambiguous), because of the 1:1 match in the constellation of the specific cast.

Value

If a meaningful cast is possible, an `fe_species_bavrn_state_short` object is returned

Examples

```
as_fe_species_bavrn_state_short(c(1L, 4L, 4L, 2L)) # integer
as_fe_species_bavrn_state_short(c(1, 4, 4, 2)) # double
as_fe_species_bavrn_state_short(c("1", "4", "4", "2")) # character

# cast other fe_species classes
as_fe_species_bavrn_state_short(
  fe_species_tum_wwk_short(as.character(c(1, 1, 1, 3, 3, 5)))
)
as_fe_species_bavrn_state_short(
  fe_species_ger_nfi_2012(as.character(c(20, 20, 10, 30, 30, 100)))
)

# display the casting result in terms of scientific species names
as_fe_species_bavrn_state_short(c(1L, 4L, 4L, 2L)) |> format("sci")
```

`as_fe_species_ger_nfi_2012` *Cast Appropriate Objects Into a* **fe_stand_ger_nfi_2012** *Species Class Object*

Description

If the cast is forward ambiguous, the function terminates with an error. "Forward ambiguous" means that one code in the original object corresponds to more than one codes in the goal coding. If the cast loses information, a warning is raised, but the cast is performed. "Information loss" in this context means that several codes from the orginal coding correspond to only one code in the goal coding.

Usage

```
as_fe_species_ger_nfi_2012(x)
```

Arguments

`x`	The object to be cast, either a vector of types `integer`, `double`, or `character` or an object of one of the supported **fe_species** classes

Details

Note that a cast where only one species id from the original coding translates in a goal coding which represents a group of species is NOT considered losing information (i.e. backward ambiguous), because of the 1:1 match in the constellation of the specific cast.

Value

If a meaningful cast is possible, an `fe_species_ger_nfi_2012` object is returned

Examples

```
as_fe_species_ger_nfi_2012(c(10L, 40L, 40L, 20L)) # integer
as_fe_species_ger_nfi_2012(c(10, 40, 40, 20)) # double
as_fe_species_ger_nfi_2012(c("10", "40", "40", "20")) # character

# cast other fe_species classes
as_fe_species_ger_nfi_2012(
  fe_species_tum_wwk_short(as.character(c(1, 1, 1, 3, 3, 5)))
)
as_fe_species_ger_nfi_2012(
  fe_species_bavrn_state(as.character(c(20, 20, 10, 30, 30, 60)))
)
as_fe_species_ger_nfi_2012(
  fe_species_tum_wwk_long(as.character(c(83, 83, 10, 31, 40, 61)))
)
```

```
# display the casting result in terms of scientific species names
as_fe_species_ger_nfi_2012(c(10L, 40L, 40L, 20L)) |> format("sci")
```

`as_fe_species_master` *Cast Appropriate Objects Into a* **fe_stand_master** *Species Class Object*

Description

If the cast is forward ambiguous, the function terminates with an error. "Forward ambiguous" means that one code in the original object corresponds to more than one codes in the goal coding. If the cast loses information, a warning is raised, but the cast is performed. "Information loss" in this context means that several codes from the orginal coding correspond to only one code in the goal coding.

Usage

```
as_fe_species_master(x)
```

Arguments

`x`	The object to be cast, either a vector of types `integer`, `double`, or `character` or an object of one of the supported **fe_species** classes

Details

Note that a cast where only one species id from the original coding translates in a goal coding which represents a group of species is NOT considered losing information (i.e. backward ambiguous), because of the 1:1 match in the constellation of the specific cast.

Value

If a meaningful cast is possible, an `fe_species_master` object is returned

Examples

```
as_fe_species_master(c("abies_001", "fagus_001")) # character

# cast other fe_species classes
as_fe_species_master(
  fe_species_tum_wwk_short(as.character(c(1, 1, 1, 3, 3, 5)))
)
as_fe_species_master(
  fe_species_ger_nfi_2012(as.character(c(20, 20, 10, 30, 30, 100)))
)
as_fe_species_master(
  fe_species_bavrn_state(as.character(c(10L, 40L, 40L, 20L)))
)
```

```
as_fe_species_master(
  fe_species_tum_wwk_long(as.character(c(10L, 30L, 88L, 87L)))
)

# display the casting result in terms of scientific species names
as_fe_species_master(c("abies_001", "fagus_001")) |> format("sci")
```

`as_fe_species_tum_wwk_long` *Cast Appropriate Objects Into a* **fe_species_tum_wwk_long** *Species Class Object*

Description

If the cast is forward ambiguous, the function terminates with an error. "Forward ambiguous" means that one code in the original object corresponds to more than one codes in the goal coding. If the cast loses information, a warning is raised, but the cast is performed. "Information loss" in this context means that several codes from the orginal coding correspond to only one code in the goal coding.

Usage

```
as_fe_species_tum_wwk_long(x)
```

Arguments

`x`	The object to be cast, either a vector of types `integer`, `double`, or `character` or an object of one of the supported **fe_species** classes

Details

Note that a cast where only one species id from the original coding translates in a goal coding which represents a group of species is NOT considered losing information (i.e. backward ambiguous), because of the 1:1 match in the constellation of the specific cast.

Value

If a meaningful cast is possible, an `fe_species_ger_nfi_2012` object is returned

Examples

```
as_fe_species_tum_wwk_long(c(10L, 41L, 41L, 31L)) # integer
as_fe_species_tum_wwk_long(c(10, 41, 41, 31)) # double
as_fe_species_tum_wwk_long(c("10", "41", "41", "31")) # character

# cast other fe_species classes
as_fe_species_tum_wwk_long(
  fe_species_tum_wwk_short(as.character(c(1, 1, 1, 3, 3, 5)))
```

```
)
as_fe_species_tum_wwk_long(
  fe_species_bavrn_state(as.character(c(20, 20, 10, 30, 30, 60)))
)

# display the casting result in terms of scientific species names
as_fe_species_tum_wwk_long(c(10L, 41L, 41L, 31L)) |> format("sci")
```

`as_fe_species_tum_wwk_short` *Cast Appropriate Objects Into a* **fe_stand_tum_wwk_short** *Species Class Object*

Description

If the cast is forward ambiguous, the function terminates with an error. "Forward ambiguous" means that one code in the original object corresponds to more than one codes in the goal coding. If the cast loses information, a warning is raised, but the cast is performed. "Information loss" in this context means that several codes from the orginal coding correspond to only one code in the goal coding.

Usage

```
as_fe_species_tum_wwk_short(x)
```

Arguments

`x`	The object to be cast, either a vector of types `integer`, `double`, or `character` or an object of one of the supported **fe_species** classes

Details

Note that a cast where only one species id from the original coding translates in a goal coding which represents a group of species is NOT considered losing information (i.e. backward ambiguous), because of the 1:1 match in the constellation of the specific cast.

Value

If a meaningful cast is possible, an `fe_species_tum_wwk_short` object is returned

Examples

```
as_fe_species_tum_wwk_short(c(1L, 4L, 4L, 2L, 6L, 7L)) # integer
as_fe_species_tum_wwk_short(c(1, 4, 4, 2, 6, 7)) # double
as_fe_species_tum_wwk_short(c("1", "4", "4", "2", "6")) # character

# cast other fe_species classes
as_fe_species_tum_wwk_short(
```

```
  fe_species_ger_nfi_2012(as.character(c(20, 20, 10, 30, 30, 100)))
)
as_fe_species_tum_wwk_short(
  fe_species_bavrn_state(as.character(c(20, 20, 10, 30, 30, 60)))
)
as_fe_species_tum_wwk_short(
  fe_species_tum_wwk_long(as.character(c(10, 20, 10, 31, 31, 812, 87)))
)

# display the casting result in terms of scientific species names
as_fe_species_tum_wwk_short(c(1, 4, 4, 2, 6, 7, 5, 5)) |> format("sci")
```

`crown_diameter_silva` *Estimate a tree's crown diameter*

Description

This function can be used for estimating a tree's crown diameter, given its species, its stem diameter at breast height, and its total height. This is the crown diameter function which is implemented in the forest growth simulator SILVA (Pretzsch et al. 2002). The crown diameter in this context is defined as the average diameter of the crown at its greatest lateral extension. The crown diameter equations are available for exactly the species (groups) defined in the coding *tum_wwk_short*. If they are called with another species coding supported by the package **ForestElementsR**, `crown_diameter_silva` will attempt to convert them accordingly.

Usage

```
crown_diameter_silva(species_id, dbh_cm, height_m)
```

Arguments

`species_id`	Vector of species id's following the *tum_wwk_short* species coding. Ideally, these species_id's are provided as a `fe_species_tum_wwk_short` object. If they are provided as another object, `crown_diameter_silva` will make an attempt to convert them. If this is not possible, the function will terminate with an error. The species id's can also be provided as numeric values (`double` or `integer`) or `character`. These will be internally converted to `fe_species_tum_wwk_short`. If this fails (i.e. the user provided species codes not defined in the *tum_wwk_short* coding), an error is thrown and the function terminates.
`dbh_cm`	Vector of tree dbh values in cm (dbh = stem diameter at breast height, i.e. 1.3 m)
`height_m`	Vector of tree height values in m

Value

An estimate of the tree's diameter of the crown at its greatest lateral extension in m.

References

Pretzsch H, Biber P, Dursky J (2002). "The single tree-based stand simulator SILVA: construction, application and evaluation." *Forest Ecology and Management*, **162**(1), 3–21. ISSN 0378-1127, doi:10.1016/S03781127(02)000476.

Examples

```
# Estimate the crown diameter of a Scots pine with a stem diameter
# at breast height of 45.2 cm and a total height of 29.2 m:
crown_diameter_silva(
  species_id = "3",    # will be internally converted to tum_wwk_short
  dbh_cm = 45.2,
  height_m = 29.2
) # 6.1 m (rounded)

# Crown diameter estimate for a European beech with
# the same height and diameter:
crown_diameter_silva(
  species_id = "5",   # will be internally converted to tum_wwk_short
  dbh_cm = 45.2,
  height_m = 29.2
) # 9.6 m (rounded)

# Run vectorized
spec <- mm_forest_1_fe_stand_spatial$trees$species_id
d <- mm_forest_1_fe_stand_spatial$trees$dbh_cm
h <- mm_forest_1_fe_stand_spatial$trees$height_m
crown_diameter_silva(spec, d, h)
```

---

d_100 *Dominant Diameter d100*

Description

The dominant diameter d100 was conceptuated by Ernst Assmann and Friedrich Franz in order obtain a mean diameter value for those trees which usually dominate a stand throughout its whole life.

Usage

```
d_100(d, n_rep_ha)
```

Arguments

d	vector of diameter values to calculate the dominant diameter d_100 of
n_rep_ha	vector of representation numbers per ha for each diameter in d. Must have the same length as d or the length 1 (in which case it is recycled to the length of d). Otherwise, the function terminates with an error.

Details

The d100 is defined as the quadratic mean diameter of the hundred thickest trees per ha. If there are only 100 trees or less on one ha, the d100 is the same as the quadratic mean diameter d_q. While the d100 is well defined and useful in monospecific stands, it is less so in mixed stands.

Value

The dominant diameter d100 value resulting from the input data

See Also

Other stand diameters: d_dom_weise(), d_q()

Examples

```
# A sample of trees from an angle count sample, where each
# tree represents a basal area of 4 m²/ha
d_cm <- c(12, 13, 25, 27, 28, 26, 26.1, 32, 35, 31, 42)
n_rep_ha <- 4 / ((d_cm / 100)^2 * pi / 4) # representation number of each tree
d_100(d_cm, n_rep_ha)
d_q(d_cm, n_rep_ha) # quadratic mean diameter for comparison


# Typical application to a set of single tree data grouped by survey
# time and species
# (note that everyone is applying d_100 mixed stands, but you should do it
# only, if you know exactly what you are doing)
library(dplyr)
oldopt <- options(fe_spec_lang = "eng") # display colloquial species names
# for d_100 in mixed stands, we require species shares
spec_shares <- species_shares(
  mm_forest_1_fe_stand_spatial,
  tree_filter = !removal, # include remaining trees only
  method = "ba_wd"
)
# extract the tree data to allow insights into the mechanics
trees <- mm_forest_1_fe_stand_spatial$trees |> filter(!removal)
# join with the shares
trees |>
  left_join(spec_shares) |>
  group_by(species_id, time_yr) |>
  summarise(
    n_ha     = round(sum(n_rep_ha)),
    d_q      = d_q(dbh_cm, n_rep_ha), # For comparison
    d_100_cm = d_100(dbh_cm, n_rep_ha / species_share)
  ) |>
  print(n = Inf)
options(oldopt) # set species name display to previous value
```

d_age_gnfi3 *Estimate Stem Diameter Growth With the 3rd German National Forest Inventory Growth Model (2012)*

Description

Tree diameter growth model of the third German National Forest Inventory of 2012 (Riedel et al. 2017). Allows to estimate a tree's dbh at any age if its dbh is known at a given age.

Usage

```
d_age_gnfi3(species_id, age_yr, dbh_cm_known, age_yr_known)
```

Arguments

species_id	Vector of species id's preferably following the *ger_nfi_2012* species coding. Ideally, these species_id's are provided as a fe_species_ger_nfi_2012 object. See Details for how other species codings are handled.
age_yr	Single numeric value or vector of ages (in years) for which the diameter is to be calculated
dbh_cm_known	Vector of known dbh (cm) values at age age_yr_known
age_yr_known	Vector of ages (years) for which the dbh d_cm_known is known

Details

Originally, the function was parameterized for species and species groups corresponding to the national forest inventory's species coding (fe_species_ger_nfi_2012). We have attributed in addition these the original parameters also to the species codings fe_species_tum_wwk_short, and fe_species_bavrn_state_short. When called with a given species coding, the function will try to use the "nearest" of these three alternatives. Fallback option is the attempt to use fe_species_tum_wwk_short.

Value

A single diameter value or vector of diameter values corresponding to age_yr

References

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

See Also

Other growth functions: age_d_gnfi3(), age_h_gnfi3(), h_age_gnfi3()

Examples

```
# A Norway spruce has a diameter of 17.5 cm at age 45. Estimate
# its diameter at age 55
d_age_gnfi3(10, 55, 17.5, 45) # 10 is ger_nfi_2012 code for Norway spruce

# Do the same but 10 years backward in time (dbh at age 35)
d_age_gnfi3(10, 35, 17.5, 45)

# Apply for more than one tree, different species, same age
d_known   <- c(23.1, 16.2, 35.2, 19.3, 21.8)
species   <- as_fe_species_tum_wwk_short(c(3, 3, 3, 6, 6))
d_age_gnfi3(
  species, age_yr = 50, dbh_cm_known = d_known, age_yr_known = 40
)
```

---

d_dom_weise *Weise's Dominant Diameter*

Description

The dominant diameter after Weise is the quadratic mean diameter of the 20% biggest trees in a stand. In contrast to the dominant diameter d_100 it is well defined not only in monospecific stands, but also in mixed stands.

Usage

```
d_dom_weise(d, n_rep = 1)
```

Arguments

d	vector of diameter values to calculate Weise's dominant diameter of
n_rep	vector of representation numbers (typically the number of trees per ha corresponding to the diameter at the same position), will be used as individual weights for each diameter. If n_rep has length 1, it will be recycled to the length of d. Otherwise, if the length of n_rep does not correspond to the length of d, the function will terminate with an error.

Value

The value of Weise's dominant diameter resulting from the input data

See Also

Other stand diameters: d_100(), d_q()

Examples

```
# A sample of trees from an angle count sample, where each
# tree represents a basal area of 4 m²/ha
d_cm <- c(12, 13, 25, 27, 28, 26, 26.1, 32, 35, 31, 42)
n_rep_ha <- 4 / ((d_cm / 100)^2 * pi / 4) # representation number of each tree
d_dom_weise(d_cm, n_rep_ha)
d_100(d_cm, n_rep_ha) # dominant diameter d100 for comparison
d_q(d_cm, n_rep_ha) # quadratic mean diameter for comparison

# if 20% of the trees are 100 stems/ha, Weise's dominant diameter and
# d100 are equal
d_cm <- rnorm(n = 500, mean = 35, sd = 7)
d_dom_weise(d_cm, 1)
d_100(d_cm, 1)

# Weise's dominant diameter is greater than d100, if 20% of the trees
# represent less than 100 trees/ha
d_cm <- rnorm(n = 200, mean = 35, sd = 7)
d_dom_weise(d_cm, 1)
d_100(d_cm, 1)

# Weise's dominant diameter is smaller than d100, if 20% of the trees
# represent more than 100 trees/ha
d_cm <- rnorm(n = 800, mean = 35, sd = 7)
d_dom_weise(d_cm, 1)
d_100(d_cm, 1)
```

---

d_q *Quadratic Mean Diameter*

Description

Function for calculating the quadratic mean of a vector. The typical application in forestry is to calculate the quadratic mean diameter.

Usage

```
d_q(d, n_rep = 1)
```

Arguments

d	vector of (stem diameter at breast height) values to calculate the quadratic mean of
n_rep	vector of representation numbers (typically the number of trees per ha corresponding to the diameter at the same position), will be used as individual weights for each diameter. If n_rep has length 1, it will be recycled to the length of d. Otherwise, if the length of n_rep does not correspond to the length of d, the function will terminate with an error.

Value

the quadratic mean of d

See Also

Other stand diameters: d_100(), d_dom_weise()

Examples

```
# Evaluate a sample of equally weighted tree diameters
d_cm <- c(12, 13, 25, 27, 28, 26, 26.1, 32, 35, 31, 42)
d_q(d_cm) # quadratic mean diameter
mean(d_cm) # the arithmetic mean is not the same!

# Assume, the same sample comes from an angle count sample, where each
# tree represents a basal area of 4 m²/ha
n_rep_ha <- 4 / ((d_cm / 100)^2 * pi / 4) # representation number of each tree
d_q(d_cm, n_rep_ha)


# Typical application to a set of single tree data grouped by survey
# time and species
library(dplyr)
oldopt <- options(fe_spec_lang = "eng") # display colloquial species names
# extract the tree data to allow insights into the mechanics
trees <- mm_forest_1_fe_stand_spatial$trees |> filter(!removal)
trees |>
  group_by(species_id, time_yr) |>
  summarise(
    n_ha   = round(sum(n_rep_ha)),
    d_q_cm = d_q(dbh_cm, n_rep_ha)
  ) |>
  print(n = Inf)
options(oldopt) # set species name display to previous value
```

---

example_data *Example Stands*

Description

ForestelementsR comes with five example stands that exist each in a 'raw' format that could have been taken from a user data base, and one of the package's specific stand object classes (**fe_stand**, **fe_stand_spatial**, ,**fe_ccircle_spatial**). Four of the example stands correspond to the **fe_stand** class, one to the **fe_stand_spatial** and one to **fe_ccircle_spatial**class. The **fe_stand**, **fe_stand_spatial** and **fe_ccircle_spatial** objects have been constructed from the raw data using the functions fe_stand, fe_stand_spatial and fe_ccircle_spatial (see the examples section of the functions' documentation).

fe_stand compatible examples: The first stand is a 53 years-old Norway spruce stand, the second one a 79 year old European beech stand, the third an even-aged mixed stand of Norway spruce and European beech (75 and 88 years, respectively). The fourth stand is a selection forest comprising the species Norway spruce, silver fir, European beech, and sycamore. All these stands have an area of 0.49 ha. The raw data (see name suffix) are data.frames (tibbles) comprising the following columns

`stand` Name of the stand

`species` Integer numbers following the `TUM_WWK_short` convention

`no` Tree id

`age` Tree age (i.e. stand age, in these examples) in years; note that the selection forest data do not provide age information, as this is usually not available in practice and mostly not meaningful for this forest type

`d` Breast height (1.3 m) stem diameter in cm

`h` Tree height in m

`hcb` Crown base height in m

`crad` Crown radius in m

The fe_stand objects (see name suffix) resulted from converting the raw data with `fe_stand`

fe_stand_spatial compatible example: mm_forest_1_raw represents raw data from a research plot in a mixed mountain forest that has been surveyed several times. **mm_forest_1_fe_stand_spatial** is the representation of this plot as an object of class `fe_stand_spatial`

fe_ccircle_spatial compatible examples spruce_pine_ccircle_raw represents raw data that could come from a typical inventory plot of the Bavarian State Forest, while **spruce_pine_ccircle_spatial** is an object of class `fe_ccircle_spatial`.

Usage

```
norway_spruce_1_raw

norway_spruce_1_fe_stand

european_beech_1_raw

european_beech_1_fe_stand

spruce_beech_1_raw

spruce_beech_1_fe_stand

selection_forest_1_raw

selection_forest_1_fe_stand

mm_forest_1_raw

mm_forest_1_fe_stand_spatial

spruce_pine_ccircle_raw
```

spruce_pine_ccircle_spatial

spruce_pine_ccircle_spatial_notrees

Format

An object of class tbl_df (inherits from tbl, data.frame) with 498 rows and 8 columns.

An object of class fe_stand of length 4.

An object of class tbl_df (inherits from tbl, data.frame) with 268 rows and 8 columns.

An object of class fe_stand of length 4.

An object of class tbl_df (inherits from tbl, data.frame) with 225 rows and 8 columns.

An object of class fe_stand of length 4.

An object of class tbl_df (inherits from tbl, data.frame) with 283 rows and 7 columns.

An object of class fe_stand of length 4.

An object of class list of length 5.

An object of class fe_stand_spatial (inherits from fe_stand) of length 6.

An object of class list of length 8.

An object of class fe_ccircle_spatial (inherits from fe_stand_spatial, fe_stand) of length 6.

An object of class fe_ccircle_spatial_notrees (inherits from fe_ccircle_spatial, fe_stand_spatial, fe_stand) of length 6.

---

fe_ccircle_spatial *User Friendly Construction of an* **fe_ccircle_spatial** *Object from a List of Data Frames*

Description

fe_ccircle_spatial() provides a user-friendly interface for the constructor new_fe_ccircle_spatial. While the constructor does not prevent users from creating malformed fe_ccircle_spatial objects, fe_ccircle_spatial does everything to achieve a well-defined object mostly based on an initial list of data.frames that might be, e.g. drawn out of a user's own data base.

Usage

```
fe_ccircle_spatial(
  x,
  method = c("strict", "flexible"),
  tree_frame_name = "trees",
  tree_pos_frame_name = "tree_positions",
  circle_frame_name = "circle_definition",
  center_coord = "center_coordinate",
```

```
  small_trees_name = "small_trees",
  time_yr_name = "time_yr",
  tree_id_col,
  species_id_col,
  time_yr_col,
  dbh_cm_col,
  radius_col,
  angle_col,
  stand_id = "my_fe_ccircle_spatial",
  layer_key_col = NA,
  age_yr_col = NA,
  height_m_col = NA,
  crown_base_height_m_col = NA,
  crown_radius_m_col = NA,
  removal_col = NA,
  ingrowth_col = NA,
  n_rep_ha_col = NA,
  verbose = TRUE
)
```

Arguments

x	named list of two data frames to be coerced into the goal object. One data frame must contain the single tree data; it has the same requirements as for the input data frame x of the function fe_stand or fe_stand_spatial. Another data frame must contain the tree positions.
method	name of the method for dealing with trees without positions. If method = "flexible" the validator issues a warning. By default (method = "strict"), all trees must have defined positions and the validator issues an error. Trees without positions are typically trees below a certain dbh threshold. Note that this check applies only to trees that actually have a dbh, not to trees in the small_trees slot of an fe_ccircle_spatial object.
tree_frame_name	name of the data frame in x that contains the single tree data (default: "trees")
tree_pos_frame_name	name of the data frame in x that contains the tree positions (default: "tree_positions"), it must contain the tree_id the radius and the angle in polar coordinates. The angle must be defined in degrees with the Y-Axis as North (anti-clockwise) and the radius in meters.
circle_frame_name	name of the data frame in x that contains the definition of the concentric circles. It must contain the lower and the upper dbh limits, the circle area in Ha, and the slope in degrees. If the slope is not given it will be set to 0.
center_coord	name of the sf object that contains the center coordinate of the circles with coordinate reference system (either Gauss Kruger or UTM). If it is not provided the center will be c(0,0) by the default

small_trees_name
: name of the object that contains the information regarding the small trees (generally those trees that do no have any dbh because of having a height below 1.3 m). This object is still experimental, so the only requirement for it is that it has to be a data frame with at least one row with column names. Useful standard information in this data frame are the tree id, the height and representation area.

time_yr_name
: name of the element of x which contains the required time information, i.e. a single (calendar) year or vector of years in order to give the object a time relation. If small tree information is present (i.e. the small tree data frame is not empty), time_yr must absolutely match with the time_yr column of this data frame.

tree_id_col
: name of the column in the trees and tree_positions data frames which contains the tree id's (character, required, must not contain missing values)

species_id_col
: name of the column in trees data frame which contains the species id's. Must be an object of one of the fe_species classes supported by this package. This column is required, must not contain missing values.

time_yr_col
: name of the column in the trees data frame which provides time information in years (character, required, must not contain missing values)

dbh_cm_col
: name of the column in the trees data frame which contains the dbh in cm (character, required, must not contain missing values)

radius_col
: name of the column in the tree positions data frame that contains the distance to the center of the plot (character).

angle_col
: name of the column in the tree positions data frame that contains the angle coordinate of the tree in polar coordinates. The angle must be measured respect to the x axis and anticlockwise (character).

stand_id
: arbitrary id of the stand (character, default: "my_fe_ccircle_spatial")

layer_key_col
: name of the column in x that contains codes for the stand layer a given tree belongs to. These codes are whole numbers. The following values are allowed: 1 - Main stand, 2 - Understorey, 3 - Pregeneration (ger: "Vorausverjuengung"), 4 - Remnant trees, hold over trees, veteran trees (ger: "Nachhiebsreste", "Ueberhaelter", "Altbaeume"). Must not contain missing values if provided. If not provided, it will be set to 1 (main stand) for every tree.

age_yr_col
: name of the column in the trees data frame which provides the tree ages in years (character, optional, may contain missing values)

height_m_col
: name of the column in the trees data frame which provides the tree heights in m (character, optional, may contain missing values)

crown_base_height_m_col
: name of the column in the trees data frame which provides the crown base heights in m (character, optional, may contain missing values)

crown_radius_m_col
: name of the column in the trees data frame which provides the crown radii in m (character, optional, may contain missing values)

removal_col
: name of the column in the trees data frame which provides the tree's removal status. If TRUE, this indicates the tree was removed or dead at the time indicated in age_yr_col (character, optional, must not contain missing values if

provided). If not provided, the removal status will be FALSE for all trees in the resulting fe_ccircle_spatial object.

ingrowth_col — name of the column in the trees data frame which provides the tree's ingrowth status. If TRUE, this indicates a tree that grew newly in at the time indicated in age_yr_col (character, optional, must not contain missing values if provided). If not provided, the ingrowth status will be FALSE for all trees in the resulting fe_ccircle_spatial object.

n_rep_ha_col — name of the column in the trees data frame which provides each tree's representation number per ha. n_rep_ha will be always recalculated based on the representation area of each of the concentric circles.

verbose — name of the column in the trees data frame which provides the tree's ingrowth status. If TRUE, this indicates a tree that grew newly in at the time indicated in age_yr_col (character, optional, must not contain missing values if provided). If not provided, the ingrowth status will be FALSE for all trees in the resulting fe_ccircle_spatial object.

Details

An object of class fe_ccircle_spatial is a child object of fe_stand_spatial which, however, contains information about the horizontal positions of the trees in a concentric circle representation scheme. All spatial information and its processing is based on the R-package sf.

The input object x to fe_ccircle_spatial must be a list that comprises two and an optional third data frame(s):

- a data frame containing the single tree information (same requirements as for fe_stand). This data frame must contain a a minimum set of columns (tree id, species id, time variable, diameter at breast height). These columns must not contain missing values. Other columns (containing tree height, height to crown base, crown radius, tree age) are optional for the user to provide. If provided, they may contain missing values. If not provided, these columns will only contain missing values in the fe_ccircle_spatial object. The columns about the trees' removal and ingrowth status are also optional, but if provided, they must *not* contain missing values. If not provided, both columns will be filled with FALSE in the resulting fe_ccircle_spatial object. fe_ccircle_spatial will automatically add a column n_rep_ha which contains for each tree the number of trees it represents per ha. This may seem redundant if looking at fe_ccircle objects alone, but it allows a broad range of evaluation functions to be applied to different objects containing trees. In addition to the input object of fe_ccircle_spatial, there is one additional list element needed which defines a single (calendar) year or a vector of years in order to give the object a time relation. If small tree information is present (i.e. the small tree data frame is not empty), time_yr must absolutely match with the time_yr column of this data frame.
- a data frame that contains information about the tree positions. This is not part of the first data frame, because the latter could contain several observations (at different times) of the same tree, which would lead to redundant coordinate representation. This data frame must contain a column with tree id's, and the x and y coordinates of the stem center points. NA values are not allowed in this data frame.

Value

If the user input allows to construct a well-defined `fe_ccircle_spatial` object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Transform the example data collection mm_forest_1_raw (could e.g. have
# been drawn out of a user's data base) into an fe_ccircle_spatial object

spruce_pine_ccircle_sp <- spruce_pine_ccircle_raw |>
  fe_ccircle_spatial(
    method = "flexible",
    tree_id_col = "tree_id",
    species_id_col = "species_id",
    time_yr_col = "time_yr",
    dbh_cm_col = "dbh_cm",
    radius_col = "R",
    angle_col = "angle",
    stand_id = mm_forest_1_raw$stand_id,
    height_m_col = "height_m",
    removal_col = "removal",
    time_yr_name = "time_yr"
  )
# Show a little summary, display scientific species names
options(fe_spec_lang = "sci")
spruce_pine_ccircle_sp |> summary()
```

---

fe_ccircle_spatial_notrees

User Friendly Construction of an **fe_ccircle_spatial_notrees** *Object from a List of Data Frames*

Description

`fe_ccircle_spatial_notrees()` provides a user-friendly interface for the constructor `new_fe_ccircle_spatial_notrees`. While the constructor does not prevent users from creating malformed `new_fe_ccircle_spatial_notrees` objects, `new_fe_ccircle_spatial_notrees` does everything to achieve a well-defined object mostly based on an initial list of data.frames that might be, e.g. drawn out of a user's own data base.

Usage

```
fe_ccircle_spatial_notrees(
  x,
  time_yr_name = "time_yr",
  circle_frame_name = "circle_definition",
  center_coord = "center_coordinate",
  small_trees_name = "small_trees",
  stand_id = "my_fe_ccircle_spatial_notrees"
)
```

Arguments

x	Named list to be coerced into the goal object.
time_yr_name	name of the element of x which contains the required time information, i.e. a single (calendar) year or vector of years in order to give the object a time relation. If small tree information is present (i.e. the small tree data frame is not empty), time_yr must absolutely match with the time_yr column of this data frame.
circle_frame_name	Name of the data frame in x that contains the definition of the concentric circles. It must contain the lower and the upper dbh limits, the circle area in Ha, and the slope in degrees. If the slope is not given it will be set to 0.
center_coord	Name of the sf object that contains the center coordinate of the circles with coordinate reference system (either Gauss Kruger or UTM). If it is not provided the center will be c(0,0) by the default
small_trees_name	Name of the object that contains the information regarding the small trees (generally those trees that do no have any dbh because of having a height below 1.3 m). This object is still experimental, so the only requirement for it is that it has to be a data frame without any further specification. Useful standard information in this data frame are the tree id, the height and represenation area.
stand_id	arbitrary id of the stand (character, default: "my_fe_ccircle_spatial")

Details

An object of class fe_ccircle_spatial_notrees is a child object of fe_ccircle_spatial which, however does not contain trees which are big enough to have a dbh. So, in contrast to fe_ccircle_spatial objects, where the elements trees, and tree_positions must be data frames, both are NULL in this class. The class fe_ccircle_spatial_notrees has been designed for covering a comparably rare case, i.e. an inventory point where no regular trees (trees that are big enough to have a dbh), but possibly small trees are present. The input object x to fe_ccircle_spatial must be a list that follows the same conventions as the input object of fe_ccircle_spatial, except that no tree data frame is required. If it does exist, it will be ignored. In addition to the input object of fe_ccircle_spatial, there is one additional list element needed which defines a single (calendar) year or a vector of years in order to give the object a time relation. If small tree information is present (i.e. the small tree data frame is not empty), time_yr must absolutely match with the time_yr column of this data frame.

Value

If the user input allows to construct a well-defined fe_ccircle_spatial_notrees object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Transform the example data collection mm_forest_1_raw (could e.g. have
# been drawn out of a user's data base) into an fe_ccircle_spatial_notrees
# object. This means, that in the resulting data frame the trees and their
# positions are ignored.
notrees_example <- spruce_pine_ccircle_raw |>
```

```
  fe_ccircle_spatial_notrees()

plot(notrees_example)
```

`fe_species_bavrn_state` *Construct a* fe_species_bavrn_state *Species Code Vector*

Description

User interface for constructing a vector of species codes follwing the *fe_species_bavrn_state* convention

Usage

```
fe_species_bavrn_state(x = character())
```

Arguments

`x`	Input vector to become a vector of tree species codes by the definition *bavrn_state*. Any type of vector (typically `integer`) which, after conversion with `as.character`, adheres to that definition is acceptable. If `x` is provided as a character vector, leading and trailing white spaces will be trimmed.

Details

The *bavrn_state* species coding is the species coding used by the Bavarian State Forest Service. See the example section for how to look up the coding.

Value

If the user input allows to construct a well-defined `fe_species_bavrn_state` object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Libraries required for the following two examples
library(dplyr)
library(purrr)

# Look up the bavrn_state species codes for all supported species
# the column species_id contains the bavrn_state codes
species_codings |>
  filter(species_coding == "bavrn_state") |>
  pluck(2, 1) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)
```

```
# Display a summary table which shows the number of single species behind
# each bavrn_state species code
species_codings |>
  filter(species_coding == "bavrn_state") |>
  pluck(2, 1) |>
  group_by(name_eng, species_id) |> # display english names
  summarise(n = n()) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)


# Make an fe_species_bavrn_state vector from a vector of integer codes
spec_ids <- fe_species_bavrn_state(
  c(10, 10, 10, 60, 60, 60, 60, 30, 30, 80, 86, 80)
)
```

`fe_species_bavrn_state_short` *Construct a* fe_species_bavrn_state_short *Species Code Vector*

Description

User interface for constructing a vector of species codes follwing the *fe_species_bavrn_state_short* convention

Usage

```
fe_species_bavrn_state_short(x = character())
```

Arguments

`x`	Input vector to become a vector of tree species codes by the definition *bavrn_state_short*. Any type of vector (typically `integer`) which, after conversion with `as.character`, adheres to that definition is acceptable. If `x` is provided as a character vector, leading and trailing white spaces will be trimmed.

Details

The *bavrn_state_short* species coding is the species coding used by the Bavarian State Forest Service for aggregated data evaluations. It is actually a grouped version of the detailed *bavrn_state* coding. See the example section for how to look up the coding.

Value

If the user input allows to construct a well-defined `fe_species_bavrn_state_short` object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Libraries required for the following two examples
library(dplyr)
library(purrr)

# Look up the bavrn_state_short species codes for all supported species
# the column species_id contains the bavrn_state_short codes
species_codings |>
  filter(species_coding == "bavrn_state_short") |>
  pluck(2, 1) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)

# Or, use an even easier access with
fe_species_get_coding_table("bavrn_state_short")

# Display a summary table which shows the number of single species behind
# each bavrn_state_short species code
fe_species_get_coding_table("bavrn_state_short") |>
  group_by(name_eng, species_id) |> # display english names
  summarise(n = n()) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)


# Make an fe_species_bavrn_state_short vector from a vector of integer codes
spec_ids <- fe_species_bavrn_state_short(
  c(1, 1, 1, 6, 6, 6, 6, 3, 3, 8, 8, 8)
)
```

`fe_species_ger_nfi_2012`

Construct a fe_species_ger_nfi_2012 *Species Code Vector*

Description

User interface for constructing a vector of species codes follwing the *fe_species_ger_nfi_2012* convention

Usage

```
fe_species_ger_nfi_2012(x = character())
```

Arguments

x	Input vector to become a vector of tree species codes by the definition *ger_nfi_2012*. Any type of vector (typically `integer`) which, after conversion with `as.character`, adheres to that definition is acceptable. If `x` is provided as a character vector, leading and trailing white spaces will be trimmed.

Details

The *ger_nfi_2012* species coding is the species coding used by the German National Forest Inventory of 2012. See the example section for how to look up the coding.

Value

If the user input allows to construct a well-defined `fe_species_ger_nfi_2012` object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Libraries required for the following two examples
library(dplyr)
library(purrr)

# Look up the ger_nfi_2012 species codes for all supported species
# the column species_id contains the ger_nfi_2012 codes
species_codings |>
  filter(species_coding == "ger_nfi_2012") |>
  pluck(2, 1) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)

# Display a summary table which shows the number of single species behind
# each ger_nfi_2012 species code
species_codings |>
  filter(species_coding == "ger_nfi_2012") |>
  pluck(2, 1) |>
  group_by(name_eng, species_id) |> # display english names
  summarise(n = n()) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)


# Make an fe_species_ger_nfi_2012 vector from a vector of integer codes
spec_ids <- fe_species_ger_nfi_2012(
  c(10, 10, 10, 100, 100, 100, 100, 20, 20, 190, 290, 190)
)
```

`fe_species_get_coding` *Get Name of the Coding Belonging to an* **fe_species** *Object*

Description

Get Name of the Coding Belonging to an **fe_species** Object

Usage

```
fe_species_get_coding(x)
```

Arguments

x	An object of one of the supported fe_species classes

Value

The requested coding name (character)

Examples

```
spec_ids <- fe_species_ger_nfi_2012(c("10", "10", "30"))
fe_species_get_coding(spec_ids)
```

---

fe_species_get_coding_table

Get the Coding Table of a Supported **fe_species** *Coding*

Description

Get the Coding Table of a Supported **fe_species** Coding

Usage

```
fe_species_get_coding_table(coding)
```

Arguments

coding	A character string representing one of the supported codings as can be requested for a given **fe_species** object with fe_species_get_coding, or one of the entries in the column species_coding of the data species_codings

Value

A data.frame (tibble) representing the requested coding

Examples

```
fe_species_get_coding_table("ger_nfi_2012")
fe_species_get_coding_table("bavrn_state")
fe_species_get_coding_table("tum_wwk_short")
```

`fe_species_master` *Construct a* fe_species_master *Species Code Vector*

Description

User interface for constructing a vector of species codes follwing the *fe_species_master* convention

Usage

```
fe_species_master(x = character())
```

Arguments

`x`	Input vector to become a vector of tree species codes by the definition *master*. Any type of vector (typically `integer`) which, after conversion with `as.character`, adheres to that definition is acceptable. If `x` is provided as a character vector, leading and trailing white spaces will be trimmed.

Details

The *master* species coding is the original species coding used by the package **ForestElementsR**. It contains each species from the `species_master_table` and no species groups. See the example section for how to look up the coding.

Value

If the user input allows to construct a well-defined `fe_species_master` object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Libraries required for the following two examples
library(dplyr)
library(purrr)

# Look up the master species codes for all supported species
# the column species_id contains the master codes
fe_species_get_coding_table("master") |>
  print(n = Inf)

# Display a summary table which shows the number of single species behind
# each master species code (must be 1 with no exception)
fe_species_get_coding_table("master") |>
  group_by(name_eng, species_id) |> # display english names
  summarise(n = n()) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)

# Make an fe_species_master vector from a character vector of appropriate
```

```
# codes
spec_ids <- fe_species_master(
  c("pinus_002", "sorbus_002", "sorbus_002", "quercus_002", "prunus_001")
)
```

`fe_species_tum_wwk_long` *Construct a* fe_species_tum_wwk_long *Species Code Vector*

Description

User interface for constructing a vector of species codes follwing the *fe_species_tum_wwk_long* convention

Usage

```
fe_species_tum_wwk_long(x = character())
```

Arguments

`x`	Input vector to become a vector of tree species codes by the definition *tum_wwk_long*. Any type of vector (typically `integer`) which, after conversion with `as.character`, adheres to that definition is acceptable. If `x` is provided as a character vector, leading and trailing white spaces will be trimmed.

Details

The *tum_wwk_long* species coding is one of two codings in use at the Chair of Forest Growth and Yield Science (see `species_codings` for more information). See the example section for how to look up the coding.

Value

If the user input allows to construct a well-defined `fe_species_ger_nfi_2012` object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Libraries required for the following two examples
library(dplyr)
library(purrr)

# Look up the tum_wwk_long species codes for all supported species
# the column species_id contains the tum_wwk_long codes
species_codings |>
  filter(species_coding == "tum_wwk_long") |>
  pluck(2, 1) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
```

```
  print(n = Inf)

# Display a summary table which shows the number of single species behind
# each tum_wwk_long species code
species_codings |>
  filter(species_coding == "tum_wwk_long") |>
  pluck(2, 1) |>
  group_by(name_eng, species_id) |> # display english names
  summarise(n = n()) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)


# Make an fe_species_tum_wwk_long vector from a vector of integer codes
spec_ids <- fe_species_tum_wwk_long(
  c(10, 10, 10, 20, 20, 20, 50, 50, 811, 811, 811, 891)
)
```

`fe_species_tum_wwk_short` *Construct a* fe_species_tum_wwk_short *Species Code Vector*

Description

User interface for constructing a vector of species codes follwing the *fe_species_tum_wwk_short* convention

Usage

```
fe_species_tum_wwk_short(x = character())
```

Arguments

`x`	Input vector to become a vector of tree species codes by the definition *tum_wwk_short*. Any type of vector (typically `integer`) which, after conversion with `as.character`, adheres to that definition is acceptable. If `x` is provided as a character vector, leading and trailing white spaces will be trimmed.

Details

The *tum_wwk_short* species coding is one of two codings in use at the Chair of Forest Growth and Yield Science. It defines only a small set of single species explicitly (the most important ones in Central Europe), while all other species are attributed to three large container groups. See the example section for how to look up the coding.

Value

If the user input allows to construct a well-defined `fe_species_tum_wwk_short` object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Libraries required for the following two examples
library(dplyr)
library(purrr)

# Look up the tum_wwk_short species codes for all supported species
# the column species_id contains the tum_wwk_short codes
species_codings |>
  filter(species_coding == "tum_wwk_short") |>
  pluck(2, 1) |>
  print(n = Inf)

# Display a summary table which shows the number of single species behind
# each tum_wwk_short species code
species_codings |>
  filter(species_coding == "tum_wwk_short") |>
  pluck(2, 1) |>
  group_by(name_eng, species_id) |> # display english names
  summarise(n = n()) |>
  arrange(as.numeric(species_id)) |> # just for the look of it
  print(n = Inf)


# Make an fe_species_tum_wwk_short vector from a vector of integer codes
spec_ids <- fe_species_tum_wwk_short(c(1, 1, 1, 5, 5, 5, 5, 3, 3, 8, 9, 8))
```

---

fe_stand *User Friendly Construction of an* **fe_stand** *Object from a Data Frame*

Description

`fe_stand()` provides a user-friendly interface for the constructor `new_fe_stand`. While the constructor does not prevent users from creating malformed `fe_stand` objects, `fe_stand` does everything to achieve a well-defined object mostly based on an initial `data.frame` that might be, e.g. drawn out of a user's own data base.

Usage

```
fe_stand(
  x,
  tree_id_col,
  species_id_col,
  time_yr_col,
  dbh_cm_col,
  area_ha,
  stand_id = "my_fe_stand",
  layer_key_col = NA,
```

```
  age_yr_col = NA,
  height_m_col = NA,
  crown_base_height_m_col = NA,
  crown_radius_m_col = NA,
  removal_col = NA,
  ingrowth_col = NA,
  n_rep_ha_col = NA,
  small_trees = data.frame(),
  verbose = TRUE
)
```

Arguments

x	data.frame to be coerced into the goal object. Each line of x must represent a single tree. As a minimum requirement, there must be a column for the tree id, the species id, the time (in years), the diameter at breast height (dbh), each.
tree_id_col	name of the column in x which contains the tree id's (character, required, must not contain missing values)
species_id_col	name of the column in x which contains the species id's. Must be an object of one of the fe_species classes supported by this package. This column is required, must not contain missing values.
time_yr_col	name of the column in x which provides time information in years (character, required, must not contain missing values)
dbh_cm_col	name of the column in x which contains the dbh in cm (character, required, must not contain missing values)
area_ha	size of the stand area in ha (numeric. It is possible to have no defined stand area; in this case area_ha must be NULL, and n_rep_ha_col must be given (with no missing values).
stand_id	arbitrary id of the stand (character, default: "my_fe_stand")
layer_key_col	name of the column in x that contains codes for the stand layer a given tree belongs to. These codes are whole numbers. The following values are allowed: 1 - Main stand, 2 - Understorey, 3 - Pregeneration (ger: "Vorausverjuengung"), 4 - Remnant trees, hold over trees, veteran trees (ger: "Nachhiebsreste", "Ueberhaelter", "Altbaeume"). Must not contain missing values if provided. If not provided, it will be set to 1 (main stand) for every tree.
age_yr_col	name of the column in x which provides the tree ages in years (character, optional, may contain missing values)
height_m_col	name of the column in x which provides the tree heights in m (character, optional, may contain missing values)
crown_base_height_m_col	name of the column in x which provides the crown base heights in m (character, optional, may contain missing values)
crown_radius_m_col	name of the column in x which provides the crown radii in m (character, optional, may contain missing values)

removal_col — name of the column in x which provides the tree's removal status. If TRUE, this indicates the tree was removed or dead at the time indicated in age_yr_col (character, optional, must not contain missing values if provided). If not provided, the removal status will be FALSE for all trees in the resulting fe_stand object.

ingrowth_col — name of the column in x which provides the tree's ingrowth status. If TRUE, this indicates a tree that grew newly in at the time indicated in age_yr_col (character, optional, must not contain missing values if provided). If not provided, the ingrowth status will be FALSE for all trees in the resulting fe_stand object.

n_rep_ha_col — name of the column in the trees data frame which provides each tree's representation number per ha. Not required if a stand area is provided under area_ha. If a stand outline is given, n_rep_ha will be always recalculated based on the outline and the tree positions.

small_trees — An fe_stand object does contain an extra slot for small trees, defined as trees which are too small to have an own dbh (i.e. having a height > 1.3 m). So far, this slot is still experimental. The only requirement is that it is a data.frame. Such a data.frame can be provided via this parameter, it will be directly put into the goal object's small_trees slot. The default is a data.frame with zero rows and zero columns (data.frame()).

verbose — logical, if TRUE (default) the tree size variables will be checked for plausible orders of magnitude after successful construction of the fe_stand object. In case of a potential implausibility, a warning will be raised. The purpose of this mechanism is to avoid unit mismatches.

Details

The initial data.frame (or even nicer, tibble) provided by the user must contain a a minimum set of columns (tree id, species id, time variable, diameter at breast height). These columns must not contain missing values. Other columns (containing tree height, height to crown base, crown radius, tree age) are optional for the user to provide. If provided, they may contain missing values. If not provided these columns will only contain missing values in the fe_stand object. The columns about the trees' removal and ingrowth status are also optional, but if provided, they must *not* contain missing values. If not provided, both columns will be filled with FALSE in the resulting fe_stand object.

The columns from the user's data.frame that correspond to the columns defined in fe_stand objects will turn up in the object under standard names. All other columns that might be in the data.frame will be transferred to the fe_stand object with their orignal names. It is the user's responsibility to take care of them.

fe_stand will automatically add a column n_rep_ha which contains for each tree the number of trees it represents per ha. This may seem redundant if looking at fe_stand objects alone, but it allows a broad range of evaluation functions to be applied to different objects containing trees.

Value

If the user input allows to construct a well-defined fe_stand object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Constructing an fe_stand object based on the minimum required information
# - make data.frame (or, nicer, a tibble) with stand information from
#   scratch
candidate_stand <- tibble::tibble(
  tree_no    = as.character(c(1:100)),
  species_id = as_fe_species_tum_wwk_short(c(rep("1", 30), rep("5", 70))),
  time_yr    = 2022,
  dbh        = c(rnorm(30, 45, 12), rnorm(70, 38, 9))
)

# - call fe_stand
goal_fe_stand_object <- fe_stand(
  x = candidate_stand,
  tree_id_col = "tree_no",
  species_id_col = "species_id",
  time_yr_col = "time_yr",
  dbh_cm_col = "dbh",
  area_ha = 0.33
)


# Using raw data that could come out of a user's data bases; here one
# example stands (spruce_beech_1_raw) provided with the ForestElementsR
# package
spruce_beech_1_raw$year <- 2022 # No time information in the data frame
spruce_beech_1_raw$species <- as_fe_species_tum_wwk_short(
  spruce_beech_1_raw$species
)

spruce_beech_stand <- fe_stand(
  spruce_beech_1_raw,
  tree_id_col = "no",
  species_id_col = "species",
  time_yr_col = "year",
  dbh_cm_col = "d",
  area_ha = 0.49,
  stand_id = spruce_beech_1_raw[1, ]$stand,
  age_yr_col = "age",
  height_m_col = "h",
  crown_base_height_m_col = "hcb",
  crown_radius_m_col = "crad"
)

# Little summary
spruce_beech_stand |> summary()
```

`fe_stand_spatial` *User Friendly Construction of an* **fe_stand_spatial** *Object from a List of Data Frames*

Description

`fe_stand_spatial()` provides a user-friendly interface for the constructor `new_fe_stand_spatial`. While the constructor does not prevent users from creating malformed `fe_stand_spatial` objects, `fe_stand_spatial` does everything to achieve a well-defined object mostly based on an initial list of data.frames that might be, e.g. drawn out of a user's own data base.

Usage

```
fe_stand_spatial(
  x,
  tree_frame_name = "trees",
  tree_pos_frame_name = "tree_positions",
  outline_frame_name = "outline",
  orientation = NA,
  tree_id_col,
  species_id_col,
  time_yr_col,
  dbh_cm_col,
  x_m_col,
  y_m_col,
  stand_id = "my_fe_stand_spatial",
  layer_key_col = NA,
  age_yr_col = NA,
  height_m_col = NA,
  crown_base_height_m_col = NA,
  crown_radius_m_col = NA,
  removal_col = NA,
  ingrowth_col = NA,
  n_rep_ha_col = NA,
  small_trees = data.frame(),
  verbose = TRUE
)
```

Arguments

`x`	named list of two or three data frames to be coerced into the goal object. One data frame must contain the single tree data; it has the same requirements as for the input data frame `x` of the function `fe_stand`. Another data frame must contain the tree positions. The third data frame is optional, if provided it must contain a coordinate description of the stand/plot outline. See the Details section for all requirements.
`tree_frame_name`	name of the data frame in `x` that contains the single tree data (default: "trees")
`tree_pos_frame_name`	name of the data frame in `x` that contains the tree positions (default: "tree_positions")
`outline_frame_name`	name of the data frame in `x` that contains the stand outline information (default: "outline"). It is possible to have no defined stand outline; in this case

outline_frame_name must be NULL, and n_rep_ha_col must be given (with no missing values).

orientation — counterclockwise angle in degrees between the y-axis of the coordinate system of the stand outline and the tree positions and the north direction. The value NA is allowed.

tree_id_col — name of the column in the trees and tree_positions data frames which contains the tree id's (character, required, must not contain missing values)

species_id_col — name of the column in trees data frame which contains the species id's. Must be an object of one of the fe_species classes supported by this package. This column is required, must not contain missing values.

time_yr_col — name of the column in the trees data frame which provides time information in years (character, required, must not contain missing values)

dbh_cm_col — name of the column in the trees data frame which contains the dbh in cm (character, required, must not contain missing values)

x_m_col — name of the column in the tree positions data frame that contains the x-coordinates of the tree stem centers in m. The coordinates must adhere to the same coordinate system as the coordinates in the plot outline data frame. If the latter is provided, its x column must also have the name provided with x_m_col.

y_m_col — name of the column in the tree positions data frame that contains the y-coordinates of the tree stem centers in m. The coordinates must adhere to the same coordinate system as the coordinates in the plot outline data frame. If the latter is provided, its x column must also have the name provided with y_m_col.

stand_id — arbitrary id of the stand (character, default: "my_fe_stand_spatial")

layer_key_col — name of the column in x that contains codes for the stand layer a given tree belongs to. These codes are whole numbers. The following values are allowed: 1 - Main stand, 2 - Understorey, 3 - Pregeneration (ger: "Vorausverjuengung"), 4 - Remnant trees, hold over trees, veteran trees (ger: "Nachhiebsreste", "Ueberhaelter", "Altbaeume"). Must not contain missing values if provided. If not provided, it will be set to 1 (main stand) for every tree.

age_yr_col — name of the column in the trees data frame which provides the tree ages in years (character, optional, may contain missing values)

height_m_col — name of the column in the trees data frame which provides the tree heights in m (character, optional, may contain missing values)

crown_base_height_m_col — name of the column in the trees data frame which provides the crown base heights in m (character, optional, may contain missing values)

crown_radius_m_col — name of the column in the trees data frame which provides the crown radii in m (character, optional, may contain missing values)

removal_col — name of the column in the trees data frame which provides the tree's removal status. If TRUE, this indicates the tree was removed or dead at the time indicated in age_yr_col (character, optional, must not contain missing values if provided). If not provided, the removal status will be FALSE for all trees in the resulting fe_stand object.

ingrowth_col — name of the column in the trees data frame which provides the tree's ingrowth status. If TRUE, this indicates a tree that grew newly in at the time indicated in age_yr_col (character, optional, must not contain missing values if provided). If not provided, the ingrowth status will be FALSE for all trees in the resulting fe_stand object.

n_rep_ha_col — name of the column in the trees data frame which provides each tree's representation number per ha. Not required if a stand outline is provided in x. If a stand outline is given, n_rep_ha will be always recalculated based on the outline and the tree positions.

small_trees — An fe_stand_spatial object does contain an extra slot for small trees, defined as trees which are too small to have an own dbh (i.e. having a height > 1.3 m). So far, this slot is still experimental. The only requirement is that it is a data.frame. Such a data.frame can be provided via this parameter, it will be directly put into the goal object's small_trees slot. The default is a data.frame with zero rows and zero columns (data.frame()).

verbose — name of the column in the trees data frame which provides the tree's ingrowth status. If TRUE, this indicates a tree that grew newly in at the time indicated in age_yr_col (character, optional, must not contain missing values if provided). If not provided, the ingrowth status will be FALSE for all trees in the resulting fe_stand object.

Details

An object of class fe_stand_spatial is a child object of fe_stand which, however, contains information about the horizontal positions of the trees and the horizontal outline of the stand. All spatial information and its processing is based on the R-package sf.

The input object x to fe_stand_spatial must be a list that comprises two and an optional third data frame(s):

- a data frame containing the single tree information (same requirements as for fe_stand). This data frame must contain a a minimum set of columns (tree id, species id, time variable, diameter at breast height). These columns must not contain missing values. Other columns (containing tree height, height to crown base, crown radius, tree age) are optional for the user to provide. If provided, they may contain missing values. If not provided, these columns will only contain missing values in the fe_stand_spatial object. The columns about the trees' removal and ingrowth status are also optional, but if provided, they must *not* contain missing values. If not provided, both columns will be filled with FALSE in the resulting fe_stand_spatial object. fe_stand_spatial will automatically add a column n_rep_ha which contains for each tree the number of trees it represents per ha. This may seem redundant if looking at fe_stand objects alone, but it allows a broad range of evaluation functions to be applied to different objects containing trees. Note that in a fe_stand_spatial object trees are allowed which are outside the actual stand outine (e.g. "buffer zone trees"). Such trees will be automatically identified by their coordinates and not taken into account for calculating the stand's growth and yield characteristics. However, they can be included in visual displays.
- a data frame that contains information about the tree positions. This is not part of the first data frame, because the latter could contain several observations (at different times) of the same tree, which would lead to redundant coordinate representation. This data frame must contain a column with tree id's, and the x and y coordinates of the stem center points according to the

same coordinate system as the stand outline (see below). NA values are not allowed in this data frame.

- The optional third data frame must comprise the corner points of a polygon that describes the stand's outline. The polygon must not be self-intersection, therefore, the points must be given in correct sequence. The first point does not need to be repeated - fe_stand_spatial will take care for closing the polygon. This data frame needs to contain only the x and y coordinates of the corner points (in m), no NA values allowed.

Value

If the user input allows to construct a well-defined fe_stand_spatial object, this object will be returned. If not, the function will terminate with an error.

Examples

```
# Transform the example data collection mm_forest_1_raw (could e.g. have
# been drawn out of a user's data base) into an fe_stand_spatial object

mm_forest_spatial <- mm_forest_1_raw |>
  fe_stand_spatial(
    orientation = mm_forest_1_raw$north_dev,
    tree_id_col = "tree_id",
    species_id_col = "species_id",
    time_yr_col = "time_yr",
    dbh_cm_col = "dbh_cm",
    x_m_col = "x",
    y_m_col = "y",
    stand_id = mm_forest_1_raw$stand_id,
    height_m_col = "height_m",
    crown_base_height_m_col = "crown_base_height_m",
    crown_radius_m_col = "crown_radius_m",
    removal_col = "removal"
  )

# Show a little summary, display scientific species names
options(fe_spec_lang = "sci")
mm_forest_spatial |> summary()
```

---

fe_yield_table *User Friendly Construction of an* **fe_yield_table** *Object from a Data Frame*

Description

User Friendly Construction of an **fe_yield_table** Object from a Data Frame

Usage

```
fe_yield_table(
  x,
  name_orig,
  name_int,
  si_variable = "h_q_m",
  mai_variable = NA
)
```

Arguments

x	Data frame to become an fe_yield_table object. The minimum required columns (all numeric) and their names are: • age: Stand age in years • si: Site index following the yield table's definition • ba_m2_ha: The stand's standing basal area in m²/ha • v_m3_ha: The stand's standing volume in m³/ha • pai_m3_ha_yr: The periodic annual volume increment in m³/ha/yr • mai_m3_ha_yr: The mean annual volume increment in m³/ha/yr The variable(s) named as the input parameters si_variable and mai_variable must be present in addition. NA values are allowed. All other columns of x will also be taken into the yield table object and will be accessible by all functions that request values from yield tables.
name_orig	Name of the table in the language it was originally published
name_int	Internatonalized (i.e. English) version of the table name
si_variable	Character (vector), name(s) of the variable(s) (i.e. column(s) in x) which is/are used for site indexing. Usually, this will be stand heights. Some yield tables contain different definitions of stand heights (e.g. mean and dominant height). In order to enable site indexing by more than one height definition in such a case, more than one variable name can be provided here. At least one site index variable must be given to obtain a valid fe_yield_table object.
mai_variable	Character (vector), name(s) of the variable(s) (i.e. column(s) in x) which can be used for mai-site indexing. Clearly, this will be variables containing a mean annual increment (mai). Some yield tables contain different definitions of mai (e.g. over- and under bark). Therefore, more than one variable name can be provided here. If no mai variable is provided (mai_variable = NA, default), mai site indexing is not possible with the yield table of interest.

Value

An object of class fe_yield_table, if the input allows to build a valid one. If this is not the case, the function will terminate with an error.

See Also

Other yield table functions: plot.fe_yield_table(), si_to_mai_age(), si_to_mai_max(), site_index(), stocking_level(), yield_tables_for_species, ytable_age_slice(), ytable_lookup(), ytable_max_slice()

Examples

```
# Make fe_yield_table object from a very original-table-like data frame
ytable_pine_wiedemann_moderate_1943_raw |>
  fe_yield_table(
    name_orig = "Kiefer Wiedemann 1943 Maessige Durchforstung",
    name_int  = "Scots Pine Wiedemann 1943 Moderate Thinning",
    si_variable = "h_q_m",
    mai_variable = c("mai_m3_ha_yr", "red_mai_m3_ha_yr")
  )
```

`format.fe_species_bavrn_state` *Formatted Output of an* **fe_species_bavrn_state** *Vector*

Description

Usually, this function is not required to be called explicitly. It Will always be used automatically, when an object of type `fe_species_bavrn_state` is printed, be it alone, be it as part of another object (e.g. a tibble)

Usage

```
## S3 method for class 'fe_species_bavrn_state'
format(x, spec_lang = options("fe_spec_lang")$fe_spec_lang, ...)
```

Arguments

`x`	An object of type `fe_species_bavrn_state`
`spec_lang`	Choice of how species (group) names or id's are displayed. Supported choices are "code" (default, displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("spec_lang")`. If this option is not set, the choice "code" is used.
`...`	Other parameters (not used)

Value

A `character` vector either displaying the original species codes provided in `x`, or the species (group) names in the desired language

Examples

```
# Create an fe_species_bavrn_state object
spec_ids <- fe_species_bavrn_state(
  as.character(c(10, 30, 20, 40, 50, 60, 87, 63, 73, 72, 86, 80))
)

# Display in default style, scientific names, English, and German names
format(spec_ids)
format(spec_ids, spec_lang = "sci")
format(spec_ids, spec_lang = "eng")
format(spec_ids, spec_lang = "ger")

# Usual application: Set option for species code output
# Any print of an fe_species object will use the last setting of the option
options(fe_spec_lang = "sci")
spec_ids
```

---

format.fe_species_bavrn_state_short

Formatted Output of an **fe_species_bavrn_state_short** *Vector*

Description

Usually, this function is not required to be called explicitly. It Will always be used automatically, when an object of type `fe_species_bavrn_state_short` is printed, be it alone, be it as part of another object (e.g. a tibble)

Usage

```
## S3 method for class 'fe_species_bavrn_state_short'
format(x, spec_lang = options("fe_spec_lang")$fe_spec_lang, ...)
```

Arguments

`x`	An object of type `fe_species_bavrn_state_short`
`spec_lang`	Choice of how species (group) names or id's are displayed. Supported choices are "code" (default, displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("spec_lang")`. If this option is not set, the choice "code" is used.
`...`	Other parameters (not used)

Value

A `character` vector either displaying the original species codes provided in `x`, or the species (group) names in the desired language

Examples

```
# Create an fe_species_bavrn_state_short object
spec_ids <- fe_species_bavrn_state_short(
  as.character(1:9)
)

# Display in default style, scientific names, English, and German names
format(spec_ids)
format(spec_ids, spec_lang = "sci")
format(spec_ids, spec_lang = "eng")
format(spec_ids, spec_lang = "ger")

# Usual application: Set option for species code output
# Any print of an fe_species object will use the last setting of the option
options(fe_spec_lang = "sci")
spec_ids
```

---

format.fe_species_ger_nfi_2012

Formatted Output of an **fe_species_ger_nfi_2012** *Vector*

Description

Usually, this function is not required to be called explicitly. It Will always be used automatically, when an object of type `fe_species_ger_nfi_2012` is printed, be it alone, be it as part of another object (e.g. a tibble)

Usage

```
## S3 method for class 'fe_species_ger_nfi_2012'
format(x, spec_lang = options("fe_spec_lang")$fe_spec_lang, ...)
```

Arguments

`x`	An object of type `fe_species_ger_nfi_2012`
`spec_lang`	Choice of how species (group) names or id's are displayed. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.
`...`	Other parameters (not used)

Value

A `character` vector either displaying the original species codes provided in `x`, or the species (group) names in the desired language

Examples

```
# Create an fe_species_ger_nfi_2012 object
spec_ids <- fe_species_ger_nfi_2012(
  as.character(c(10, 20, 30, 40, 50, 100, 110, 120, 130, 140, 150, 170))
)

# Display in default style, scientific names, English, and German names
format(spec_ids)
format(spec_ids, spec_lang = "sci")
format(spec_ids, spec_lang = "eng")
format(spec_ids, spec_lang = "ger")

# Usual application: Set option for species code output
# Any print of an fe_species object will use the last setting of the option
options(fe_spec_lang = "eng")
spec_ids
```

`format.fe_species_master`

Formatted Output of an **fe_species_master** *Vector*

Description

Usually, this function is not required to be called explicitly. It Will always be used automatically, when an object of type `fe_species_master` is printed, be it alone, be it as part of another object (e.g. a tibble)

Usage

```
## S3 method for class 'fe_species_master'
format(x, spec_lang = options("fe_spec_lang")$fe_spec_lang, ...)
```

Arguments

`x`	An object of type `fe_species_master`
`spec_lang`	Choice of how species (group) names or id's are displayed. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.
`...`	Other parameters (not used)

Value

A `character` vector either displaying the original species codes provided in `x`, or the species (group) names in the desired language

Examples

```
# Create an fe_species_master object
spec_ids <- fe_species_master(
  c("picea_001", "fagus_001", "quercus_002", "quercus_001")
)

# Display in default style, scientific names, English, and German names
format(spec_ids)
format(spec_ids, spec_lang = "sci")
format(spec_ids, spec_lang = "eng")
format(spec_ids, spec_lang = "ger")

# Usual application: Set option for species code output
# Any print of an fe_species object will use the last setting of the option
options(fe_spec_lang = "eng")
spec_ids
```

`format.fe_species_tum_wwk_long`
Formatted Output of an **fe_species_tum_wwk_long** *Vector*

Description

Usually, this function is not required to be called explicitly. It Will always be used automatically, when an object of type `fe_species_tum_wwk_long` is printed, be it alone, be it as part of another object (e.g. a tibble)

Usage

```
## S3 method for class 'fe_species_tum_wwk_long'
format(x, spec_lang = options("fe_spec_lang")$fe_spec_lang, ...)
```

Arguments

`x`	An object of type `fe_species_tum_wwk_long`
`spec_lang`	Choice of how species (group) names or id's are displayed. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.
`...`	Other parameters (not used)

Value

A `character` vector either displaying the original species codes provided in `x`, or the species (group) names in the desired language

Examples

```
# Create an fe_species_tum_wwk_long object
spec_ids <- fe_species_tum_wwk_long(
  as.character(c(70, 61, 88, 88, 10, 971, 32))
)

# Display in default style, scientific names, English, and German names
format(spec_ids)
format(spec_ids, spec_lang = "sci")
format(spec_ids, spec_lang = "eng")
format(spec_ids, spec_lang = "ger")

# Usual application: Set option for species code output
# Any print of an fe_species object will use the last setting of the option
options(fe_spec_lang = "sci")
spec_ids
```

---

format.fe_species_tum_wwk_short

Formatted Output of an **fe_species_tum_wwk_short** *Vector*

Description

Usually, this function is not required to be called explicitly. It Will always be used automatically, when an object of type `fe_species_tum_wwk_short` is printed, be it alone, be it as part of another object (e.g. a tibble)

Usage

```
## S3 method for class 'fe_species_tum_wwk_short'
format(x, spec_lang = options("fe_spec_lang")$fe_spec_lang, ...)
```

Arguments

`x`	An object of type `fe_species_tum_wwk_short`
`spec_lang`	Choice of how species (group) names or id's are displayed. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.
`...`	Other parameters (not used)

Value

A `character` vector either displaying the original species codes provided in `x`, or the species (group) names in the desired language

Examples

```
# Create an fe_species_tum_wwk_short object
spec_ids <- fe_species_tum_wwk_short(as.character(1:10))

# Display in default style, scientific names, English, and German names
format(spec_ids)
format(spec_ids, spec_lang = "sci")
format(spec_ids, spec_lang = "eng")
format(spec_ids, spec_lang = "ger")

# Usual application: Set option for species code output
# Any print of an fe_species object will use the last setting of the option
options(fe_spec_lang = "sci")
spec_ids
```

---

get_area_ha *Get the Area in ha of a Compatible Object*

Description

The function get_area_ha will return the correct area for all appropriate objects provided by **ForestElementsR**. While the fe_stand class contains this information directly, this is not the case for the fe_stand_spatial class. However, get_area_ha will work for both (and other objects to be implemented) in the same way. In case the object x does not contain sufficient information, the function returns NULL.

Usage

```
get_area_ha(x)
```

Arguments

x	Object provided by **ForestElementsR** for which an area can be meaningfully given, typically fe_stand or fe_stand_spatial

Value

The object's area in ha. In case the object x does not contain sufficient information, the function returns NULL.

Examples

```
# Example for an fe_stand object
selection_forest_1_fe_stand |> get_area_ha()

# Example for an fe_stand_spatial object
mm_forest_1_fe_stand_spatial |> get_area_ha()
```

`height_crown_base_silva` *Estimate a tree's height to crown base*

Description

This function can be used to estimate a tree's height to crown base, given its stem diameter at breast height, its height and its species. This is the height to crown base function implemented in the forest growth simulator SILVA (Pretzsch et al. 2002). Height to crown base in this context is defined as the height where the lowest living branch of the tree's primary crown branches of the stem.

Usage

```
height_crown_base_silva(species_id, dbh_cm, height_m)
```

Arguments

`species_id`	Vector of species id's following the *tum_wwk_short* species coding. Ideally, these species_id's are provided as a `fe_species_tum_wwk_short` object. If they are provided as another `fe_species` object, `height_crown_base_silva` will make an attempt to convert them. If this is not possible, the function will terminate with an error. The species id's can also be provided as numeric values (`double` or `integer`) or `character`. These will be internally converted to `fe_species_tum_wwk_short`. If this fails (i.e. the user provided species codes not defined in the *tum_wwk_short* coding), an error is thrown and the function terminates.
`dbh_cm`	Vector of tree dbh values in cm (dbh = stem diameter at breast height, i.e. 1.3 m)
`height_m`	Vector of tree height values in m

Value

An estimate of the tree's height to crown base in m.

References

Pretzsch H, Biber P, Dursky J (2002). "The single tree-based stand simulator SILVA: construction, application and evaluation." *Forest Ecology and Management*, **162**(1), 3–21. ISSN 0378-1127, doi:10.1016/S03781127(02)000476.

Examples

```
# Estimate the height to crown base of a Scots pine with a stem diameter
# at breast height of 45.2 cm and a total height of 29.2 m:
height_crown_base_silva(
  species_id = "3",     # will be internally converted to tum_wwk_short
  dbh_cm = 45.2,
```

```
  height_m = 29.2
) # 20.9 m (rounded)

# Height to crown base estimate for a European beech with
# the same height and diameter:
height_crown_base_silva(
  species_id = "5",     # will be internally converted to tum_wwk_short
  dbh_cm = 45.2,
  height_m = 29.2
) # 15.0 m (rounded)

# Run on vectors
spec <- mm_forest_1_fe_stand_spatial$trees$species_id
h <- mm_forest_1_fe_stand_spatial$trees$height_m
d <- mm_forest_1_fe_stand_spatial$trees$dbh_cm
height_crown_base_silva(spec, d, h)
```

---

h_100 *Dominant Height h100*

Description

The dominant height h100 was conceptuated by Ernst Assmann and Friedrich Franz in order obtain a mean height value for those trees which usually dominate a stand throughout its whole life.

Usage

```
h_100(h, d, n_rep_ha)
```

Arguments

h	vector of tree height values to calculate the dominant height h100 from
d	vector of stem diameters at breast height corresponding to h, must therefore have the same length as h
n_rep_ha	vector of representation numbers per ha for each diameter in d. Must have the same length as d or the length 1 (in which case it is recycled to the length of d). Otherwise, the function terminates with an error.

Details

The h100 is defined as the quadratic mean height of the hundred thickest trees per ha. If there are only 100 trees or less on one ha, the h100 is the same as the quadratic mean height d_q. While the h100 is well defined and useful in monospecific stands, it is less so in mixed stands.

Value

the dominant height value h100 resulting from the input data

See Also

Other stand heights: h_dom_weise(), h_q(), h_q_from_d_q()

Examples

```
# A sample of trees from an angle count sample , where each
# tree represents a basal area of 4 m²/ha
d_cm <- c(12, 13, 25, 27, 28, 26, 26.1, 32, 35, 31, 42)
n_rep_ha <- 4 / ((d_cm / 100)^2 * pi / 4) # representation number of each tree
dq_cm <- d_q(d_cm, n_rep_ha)
h_m <- 0.9 * dq_cm * (d_cm / dq_cm)^0.8 # quick plausible height estim.
h_100(h_m, d_cm, n_rep_ha)
h_q(h_m, d_cm) # quadratic mean height for comparison


# Typical application to a set of single tree data grouped by survey
# time and species
# (note that everyone is applying h_100 mixed stands, but you should do it
# only, if you know exactly what you are doing)
library(dplyr)
oldopt <- options(fe_spec_lang = "eng") # display colloquial species names
# for d_100 in mixed stands, we require species shares
spec_shares <- species_shares(
  mm_forest_1_fe_stand_spatial,
  tree_filter = !removal, # remaining trees only
  method = "ba_wd"
)
# extract the tree data to allow insights into the mechanics
trees <- mm_forest_1_fe_stand_spatial$trees |> filter(!removal)
# join with the shares
trees |>
  left_join(spec_shares) |>
  group_by(species_id, time_yr) |>
  summarise(
    n_ha     = round(sum(n_rep_ha)),
    d_q      = d_q(dbh_cm, n_rep_ha),
    d_100_cm = d_100(dbh_cm, n_rep_ha / species_share),
    h_q      = h_q(height_m, dbh_cm, n_rep_ha),
    h_100_m  = h_100(height_m, dbh_cm, n_rep_ha / species_share)
  ) |>
  print(n = Inf)
options(oldopt) # set species name display to previous value
```

---

h_age_gnfi3 *Estimate Tree Height Growth With the 3rd German National Forest Inventory Growth Model (2012)*

Description

Tree height growth model of the third German National Forest Inventory of 2012 (Riedel et al. 2017). Allows to estimate a tree's height at any age if its height is known at a given age.

Usage

```
h_age_gnfi3(species_id, age_yr, h_m_known, age_yr_known)
```

Arguments

species_id	Vector of species id's preferably following the *ger_nfi_2012* species coding. Ideally, these species_id's are provided as a fe_species_ger_nfi_2012 object. See Details for how other species codings are handled.
age_yr	Single numeric value or vector of ages (in years) for which the height is to be calculated
h_m_known	Vector of known height (m) values at age age_yr_known
age_yr_known	Vector of ages (years) for which the height h_m_known is known

Details

Originally, the function was parameterized for species and species groups corresponding to the national forest inventory's species coding (fe_species_ger_nfi_2012). We have attributed in addition these the original parameters also to the species codings fe_species_tum_wwk_short, and fe_species_bavrn_state_short. When called with a given species coding, the function will try to use the "nearest" of these three alternatives. Fallback option is the attempt to use fe_species_tum_wwk_short.

Value

A single height value or vector of height values corresponding to age_yr

References

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

See Also

Other growth functions: age_d_gnfi3(), age_h_gnfi3(), d_age_gnfi3()

Examples

```
# A European beech tree has a height of 25.2 m at age 75. Estimate
# its height at age 83
h_age_gnfi3(100, 83, 25.2, 75) # 100 is ger_nfi_2012 code for E. beech

# Do the same backward in time, height at age 67
h_age_gnfi3(100, 67, 25.2, 75)
```

```
# Apply for more than one tree, different species, same age
h_known   <- c(23.1, 16.2, 35.2, 19.3, 21.8)
species   <- as_fe_species_tum_wwk_short(c(3, 3, 3, 6, 6))
h_age_gnfi3(
  species, age_yr = 70, h_m_known = h_known, age_yr_known = 60
)
```

---

h_dom_weise *Weise's Dominant Height*

Description

The dominant height after Weise is the quadratic mean height of the 20% biggest trees in a stand. In contrast to the dominant height h_100 it is well defined not only in monospecific stands, but also in mixed stands.

Usage

```
h_dom_weise(h, d, n_rep = 1)
```

Arguments

h	vector of tree height values to calculate Weise's dominant height from
d	vector of stem diameters at breast height corresponding to h, must therefore have the same length as h
n_rep	vector of representation numbers (typically the number of trees per ha corresponding to the diameter at the same position), will be used as individual weights for each height, together with the squared diameters d. If n_rep has length 1, it will be recycled to the length of d. Otherwise, if the length of n_rep does not correspond to the length of d, the function will terminate with an error.

Value

The value of Weise's dominant diameter resulting from the input data

See Also

Other stand heights: h_100(), h_q(), h_q_from_d_q()

Examples

```
# A sample of trees from an angle count sample , where each
# tree represents a basal area of 4 m²/ha
d_cm <- c(12, 13, 25, 27, 28, 26, 26.1, 32, 35, 31, 42)
n_rep_ha <- 4 / ((d_cm / 100)^2 * pi / 4) # representation number of each tree
dq_cm <- d_q(d_cm, n_rep_ha)
h_m <- 0.9 * dq_cm * (d_cm / dq_cm)^0.8 # quick plausible height estim.
```

```
h_dom_weise(h_m, d_cm, n_rep_ha)
h_100(h_m, d_cm, n_rep_ha) # dominant height h100 for comparison
h_q(h_m, d_cm) # quadratic mean height for comparison

# if 20% of the trees are 100 stems/ha, Weise's dominant diameter and
# d100 are equal
d_cm <- rnorm(n = 500, mean = 35, sd = 7)
dq_cm <- d_q(d_cm)
h_m <- 0.8 * dq_cm * (d_cm / dq_cm)^0.8 # quick plausible height estim.
h_dom_weise(h_m, d_cm, 1)
h_100(h_m, d_cm, 1)

# Weise's dominant diameter is greater than d100, if 20% of the trees
# represent less than 100 trees/ha
d_cm <- rnorm(n = 200, mean = 35, sd = 7)
dq_cm <- d_q(d_cm)
h_m <- 0.8 * dq_cm * (d_cm / dq_cm)^0.8 # quick plausible height estim.
h_dom_weise(h_m, d_cm, 1)
h_100(h_m, d_cm, 1)

# Weise's dominant diameter is smaller than d100, if 20% of the trees
# represent more than 100 trees/ha
d_cm <- rnorm(n = 800, mean = 35, sd = 7)
dq_cm <- d_q(d_cm)
h_m <- 0.8 * dq_cm * (d_cm / dq_cm)^0.8 # quick plausible height estim.
h_dom_weise(h_m, d_cm, 1)
h_100(h_m, d_cm, 1)
```

---

h_q *Quadratic Mean Height*

Description

The quadratic mean height is the stand height corresponding to the quadratic mean diameter d_q. It is actually the basal area weighted average height of a tree in the stand of interest. Insofar, its name is somewhat misleading.

Usage

```
h_q(h, d, n_rep = 1, na_h.rm = FALSE)
```

Arguments

h	vector of tree height values to calculate the quadratic mean height from
d	vector of stem diameters at breast height corresponding to h, must therefore have the same length as h

n_rep — vector of representation numbers (typically the number of trees per ha corresponding to the diameter at the same position), will be used as individual weights for each height, together with the squared diameters d. If n_rep has length 1, it will be recycled to the length of d. Otherwise, if the length of n_rep does not correspond to the length of d, the function will terminate with an error.

na_h.rm — logical; if TRUE, records with NA heights will be excluded from h, d, and n_rep. If FALSE, NA height values are kept and will lead to the function returning NA.

Value

the quadratic mean height value resulting from the input data

See Also

Other stand heights: h_100(), h_dom_weise(), h_q_from_d_q()

Examples

```
# Evaluate a sample of equally weighted trees
d_cm <- c(12, 13, 25, 27, 28, 26, 26.1, 32, 35, 31, 42)
dq_cm <- d_q(d_cm)
h_m <- 0.9 * dq_cm * (d_cm / dq_cm)^0.8 # quick plausible height estim.
h_q(h_m, d_cm) # quadratic mean height
mean(h_m) # the arithmetic mean height is not the same!

# Assume, the same sample are trees from an angle count sample, where each
# tree represents a basal area of 4 m²/ha
n_rep_ha <- 4 / ((d_cm / 100)^2 * pi / 4) # representation number of each tree
h_q(h_m, d_cm, n_rep_ha)
# Interestingly, the h_q obtained here is the same as the unweighted
# arithmetic mean height above. The reason: the n_rep_ha used here act as
# inverse basal area weights, which is exactly compensated by the basal
# area weights coming from d_cm.


# Typical application to a set of single tree data grouped by survey
# time and species
library(dplyr)
oldopt <- options(fe_spec_lang = "eng") # display colloquial species names
# extract the tree data to allow insights into the mechanics
trees <- mm_forest_1_fe_stand_spatial$trees |> filter(!removal)
trees |>
  group_by(species_id, time_yr) |>
  summarise(
    n_ha   = round(sum(n_rep_ha)),
    d_q_cm = d_q(dbh_cm, n_rep_ha),
    h_q_m  = h_q(height_m, dbh_cm, n_rep_ha)
  ) |>
  print(n = Inf)
options(oldopt) # set species name display to previous value
```

h_q_from_d_q *Estimate Quadratic Mean Height from Quadratic Mean Diameter*

Description

Fallback function to be used in the unfortunate case when a mean stand (cohort) height value is required but nothing useful is contained in the available data. The function has originally been developed for and used in the forest growth simulator SILVA (Pretzsch et al. 2002). The version here is slightly simplified for the species Norway spruce and soft deciduous woods (codes 1 and 9 in the species coding `fe_species_tum_wwk_short`), because the original version requires non-standard information which is not available outside the simulator.

Usage

```
h_q_from_d_q(species_id, d_q_cm)
```

Arguments

`species_id`	vector of species codes in any format that can be converted into the tum_wwk_short species coding (see examples)
`d_q_cm`	vector of quadratic mean stand (cohort) diameters in cm for which a corresponding mean height estimate is required

Details

A typical application of this function would be to calculate the quadratic mean diameter, `d_q` of the tree cohort of interest. If height information is totally lacking, a reasonable fallback value of the corresponding quadratic mean height, h_q, can be obtained from this function. Both values could then be used as the entry point of a standard height curve system (like `h_standard_gnfi3` and `h_standard_bv`) in order to get plausible height estimates for the single trees in the cohort. Be aware that, though the height estimates obtained from this function are plausible, they should be interpreted with care, because the variation of mean stand height at a given mean diameter is very high in reality.

Value

vector of estimated quadratic mean heights corresponding to the given values of `d_q_cm`

References

Pretzsch H, Biber P, Dursky J (2002). "The single tree-based stand simulator SILVA: construction, application and evaluation." *Forest Ecology and Management*, **162**(1), 3–21. ISSN 0378-1127, doi:10.1016/S03781127(02)000476.

See Also

Other stand heights: `h_100()`, `h_dom_weise()`, `h_q()`

Examples

```
# Species codes handed to h_q_from_d_q will be attempted to convert
# into tum_wwk_short
h_q_from_d_q(5, 25.4) # Apply to a common beech stand width d_q = 25.4 cm
h_q_from_d_q(fe_species_ger_nfi_2012(100), 25.4)

# Works also vectorized, i.e. when vectors of d_q values and species codes
# are given
species <- fe_species_tum_wwk_short(c(1, 3, 5))
d_qmean <- c(23.2, 47.2, 12.7)
h_q_from_d_q(species, d_qmean)

# Typical application: From diameter values to single tree height estimates
# when no height information is available
# - single tree diameters in a stand/cohort
dbh <- c(10.2, 43.3, 37.5, 28.8, 12.4, 19.2, 25.4, 27.3, 32.0)
# - quadratic mean diameter
d_qmean <- d_q(dbh)
species <- fe_species_tum_wwk_short(3) # we assume it's a Scots pine stand
# - quadrativ mean height fallback value
h_qest <- h_q_from_d_q(species, d_qmean)
# - single tree height estimates with the German National Inventory height
#   curve system
h_standard_gnfi3(species, dbh, d_qmean, h_qest)
```

`h_standard_bv` *Calculate Tree Heights with the Bavarian Standard Height Curve System*

Description

Implementation of the standard height curve system used by the Bavarian State Forest Service (Kennel 1972). The structure of the approach was developed by R. Kennel together with a parameterisation for European beech (Fagus sylvatica). Later, anonymous scientists have extended the parameters for all species (groups) covered by the *tum_wwk_short* species coding. The standard height curve system allows to estimate a tree's height when its dbh is given together with the quadratic mean diameter, the corresponding quadratic mean height, and the age of the stand it belongs to.

Usage

```
h_standard_bv(species_id, dbh_cm, age_yr, d_q_cm, h_q_m)
```

Arguments

`species_id`	Vector of species id's following the *tum_wwk_short* or the *bavrn_state_short* species coding. If another coding is provided, an attempt will be made to convert it into the "nearest" of the two codings mentioned above. The default is a

	conversion attempt to *tum_wwk_short*. The species id's can also be provided as numeric values (`double` or `integer`) or `character`. These will be interpreted as and converted to `fe_species_tum_wwk_short`. If all conversion attempts fail, the function will terminate with an error.
`dbh_cm`	Vector of tree dbh values in cm (dbh = stem diameter at breast height, i.e. 1.3 m)
`age_yr`	Vector of stand age values in years (will be recycled following the rules for tibbles)
`d_q_cm`	Vector of quadratic mean stand diameters (will be recycled following the rules for tibbles)
`h_q_m`	Vector of quadratic mean stand heights (will be recycled following the rules for tibbles)

Details

In order to provide maximum flexibility in applying the function h_standard_bv, the stand values (age, mean height, mean diameter) can be provided with each tree diameter individually. This allows estimating heights for trees from different stands at the same time. In the same way, the provided species codes are not required to be the same for each tree.

Value

A vector of the estimated heights

References

Kennel R (1972). *Die Buchendurchforstungsversuche in Bayern von 1870 bis 1970. Mit dem Modell einer Strukturertragstafel für die Buche*, volume 7 of *Forstliche Forschungsberichte München*. Forstwissenschaftliche Fakultät der Universität München und Bayerische Forstliche Versuchs- und Forschungsanstalt.

See Also

Other standard height curve systems: `h_standard_gnfi3()`

Examples

```
species_id <- fe_species_tum_wwk_short(rep(3, 7))  # Seven Scots pines
dbh <- c(10.1, 27.4, 31.4, 35.5, 39.8, 45.2, 47.2) # and their diameters
# Estimate the heights of these trees assuming they are from a 100 year old
# stand with a mean diameter of 35.5 cm, and a corresponding mean height
# of 28 m.
h_standard_bv(species_id, dbh, age_yr = 100, d_q_cm = 35.5, h_q_m = 28.0)

# Compare with sister function h_standard_gnfi3 which does not require
# stand age
h_standard_gnfi3(species_id, dbh, d_q_cm = 35.5, h_q_m = 28.0)
```

`h_standard_gnfi3` *Calculate Tree Heights with the Bavarian Standard Height Curve System of the 3rd German National Forest Inventory (2012)*

Description

Implementation of the standard height curve developed during the the 3nd German National Forest Inventory (Riedel et al. 2017). Structurally, this is a height curve system after Sloboda (Sloboda et al. 1993) which allows to estimate a tree's height when its species, diameter, and the quadratic mean diameter and height of (the species in) the stand is given.

Usage

```
h_standard_gnfi3(species_id, dbh_cm, d_q_cm, h_q_m)
```

Arguments

`species_id`	Vector of species id's preferably following the *ger_nfi_2012* species coding. Ideally, these species_id's are provided as a `fe_species_ger_nfi_2012` object. Second best, they are provided in the *tum_wwk_short* coding. For any other type of object, an attempt will be made to convert into *ger_nfi_2012* (and use the corresponding parameters); if that fails, conversion to *tum_wwk_short* will be attempted (see details).
`dbh_cm`	Vector of tree dbh values in cm (dbh = stem diameter at breast height, i.e. 1.3 m)
`d_q_cm`	Vector of quadratic mean stand diameters (will be recycled following the rules for tibbles)
`h_q_m`	Vector of quadratic mean stand heights (will be recycled following the rules for tibbles)

Details

Originally, the height curve system was parameterized for species and species groups corresponding to the national forest inventory's species coding (`fe_species_ger_nfi_2012`). We have attributed in addition these the original parameters also to the species codings `fe_species_tum_wwk_short`, and `fe_species_bavrn_state_short`. When called with a given species coding, the function will try to use the "nearest" of these three alternatives. Fallback option is the attempt to use `fe_species_tum_wwk_short`.

In order to provide maximum flexibility in applying the function h_standard_gnfi3, the stand values (mean height, mean diameter) can be provided with each tree diameter individually. This allows estimating heights for trees from different stands at the same time. In the same way, the provided species codes are not required to be the same for each tree.

Value

A vector of the estimated heights

References

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

Sloboda B, Gaffrey D, Matsamura N (1993). "Regionale und lokale Systeme von Hoehenkurven gleichartiger Waldbestaende." *Allgmeine Forst- und Jagdzeitung*, **164**(12), 225–228.

See Also

Other standard height curve systems: `h_standard_bv()`

Examples

```
# Three examples for single tree applications with species codes given
# as integers (but following the ger_nfi_2012 coding)

# European beech, dbh_cm < dq_cm
h_standard_gnfi3(species_id = 100, dbh_cm = 14.8, d_q_cm = 25, h_q_m = 22)

# Scots pine, dbh_cm == dq_cm
h_standard_gnfi3(species_id = 20, dbh_cm = 25, d_q_cm = 25, h_q_m = 22)

# Douglas fir, dbh_cm > dq_cm
h_standard_gnfi3(species_id = 40, dbh_cm = 45, d_q_cm = 25, h_q_m = 22)

# Same Douglas fir but species_id = 7 (i.e. tum_wwk_short),
# note the message, because numeric 7 is not convertible into ger_nfi_2012
h_standard_gnfi3(species_id = 7, dbh_cm = 45, d_q_cm = 25, h_q_m = 22)

# But no message, when species_id = 7 is made a tum_wwk_short object first,
# because this can be unambiguously converted into ger_nfi_2012
h_standard_gnfi3(
  fe_species_tum_wwk_short(7), dbh_cm = 45, d_q_cm = 25, h_q_m = 22
)

# Usually, applications will be vectorized
species_id <- fe_species_ger_nfi_2012(rep(20, 7))  # Seven Scots pines
dbh <- c(10.1, 27.4, 31.4, 35.5, 39.8, 45.2, 47.2) # and their diameters
# Estimate the heights of these trees, assuming they are from a
# stand with a mean diameter of 35.5 cm, and a corresponding mean height
# of 28 m.
h_standard_gnfi3(species_id, dbh, d_q_cm = 35.5, h_q_m = 28.0)

# Compare with sister function h_standard_bv, assuming a stand age of
# 100 years
h_standard_bv(species_id, dbh, age_yr = 100, d_q_cm = 35.5, h_q_m = 28.0)
```

is_fe_ccircle_spatial *Check if an Object is an* **fe_ccircle_spatial**

Description

Check if an Object is an **fe_ccircle_spatial**

Usage

```
is_fe_ccircle_spatial(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_ccircle_spatial class

Examples

```
strange <- "xyzabc"
is_fe_ccircle_spatial(strange)
```

is_fe_ccircle_spatial_notrees *Check if an Object is an* **fe_ccircle_spatial_notrees** *object*

Description

Check if an Object is an **fe_ccircle_spatial_notrees** object

Usage

```
is_fe_ccircle_spatial_notrees(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_ccircle_spatial_notrees class

Examples

```
strange <- "xyzabc"
is_fe_ccircle_spatial_notrees(strange)
```

`is_fe_species_bavrn_state` *Check if an Object is a* **fe_species_bavrn_state** *species code vector*

Description

Check if an Object is a **fe_species_bavrn_state** species code vector

Usage

```
is_fe_species_bavrn_state(x)
```

Arguments

`x`	An object

Value

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

Examples

```
spec_ids <- new_fe_species_bavrn_state(
  as.character(c(40, 40, 30, 30, 60, 60, 60))
)
is_fe_species_bavrn_state(spec_ids)
```

`is_fe_species_bavrn_state_short` *Check if an Object is a* **fe_species_bavrn_state_short** *species code vector*

Description

Check if an Object is a **fe_species_bavrn_state_short** species code vector

Usage

```
is_fe_species_bavrn_state_short(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_species_bavrn_state_short class

Examples

```
spec_ids <- new_fe_species_bavrn_state_short(
  as.character(1:9)
)
is_fe_species_bavrn_state_short(spec_ids)
```

---

is_fe_species_ger_nfi_2012

Check if an Object is a **fe_species_ger_nfi_2012** *species code vector*

Description

Check if an Object is a **fe_species_ger_nfi_2012** species code vector

Usage

```
is_fe_species_ger_nfi_2012(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_species_ger_nfi_2012 class

Examples

```
spec_ids <- new_fe_species_ger_nfi_2012(
  as.character(c(50, 50, 20, 20, 100, 100, 100))
)
is_fe_species_ger_nfi_2012(spec_ids)
```

is_fe_species_master *Check if an Object is a* **fe_species_master** *species code vector*

Description

Check if an Object is a **fe_species_master** species code vector

Usage

```
is_fe_species_master(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_species_master class

Examples

```
spec_ids <- new_fe_species_master(
  c("picea_001", "fagus_001", "quercus_002", "quercus_001")
)
is_fe_species_master(spec_ids)
```

is_fe_species_tum_wwk_long *Check if an Object is a* **fe_species_tum_wwk_long** *species code vector*

Description

Check if an Object is a **fe_species_tum_wwk_long** species code vector

Usage

```
is_fe_species_tum_wwk_long(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_species_tum_wwk_long class

Examples

```
spec_ids <- new_fe_species_tum_wwk_long(
  as.character(c(70, 61, 88, 88, 10, 971, 32))
)
is_fe_species_tum_wwk_long(spec_ids)
```

`is_fe_species_tum_wwk_short` *Check if an Object is a* **fe_species_tum_wwk_short** *species code vector*

Description

Check if an Object is a **fe_species_tum_wwk_short** species code vector

Usage

```
is_fe_species_tum_wwk_short(x)
```

Arguments

`x`	An object

Value

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

Examples

```
spec_ids <- new_fe_species_tum_wwk_short(
  as.character(c(4, 4, 3, 3, 5, 5, 5))
)
is_fe_species_tum_wwk_short(spec_ids)
```

`is_fe_stand` *Check if an Object is an* **fe_stand**

Description

Check if an Object is an **fe_stand**

Usage

```
is_fe_stand(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_stand class

Examples

```
strange <- "xyzabc"
is_fe_stand(strange)

is_fe_stand(norway_spruce_1_fe_stand)
```

---

is_fe_stand_spatial *Check if an Object is an* **fe_stand_spatial**

Description

Check if an Object is an **fe_stand_spatial**

Usage

```
is_fe_stand_spatial(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_stand_spatial class

Examples

```
strange <- "xyzabc"
is_fe_stand_spatial(strange)
```

is_fe_yield_table *Check if an Object is an* **fe_yield_table**

Description

Check if an Object is an **fe_yield_table**

Usage

```
is_fe_yield_table(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the fe_yield_table class

Examples

```
x <- "I want to be a yield table!"  # strange object
is_fe_yield_table(x)                # sorry

is_fe_yield_table(ytable_pine_wiedemann_moderate_1943_raw) # Nope
is_fe_yield_table(fe_ytable_pine_wiedemann_moderate_1943)  # That's better
```

new_fe_ccircle_spatial *Constructor for the* **fe_ccircle_spatial** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function fe_stand for creating an object of that class.

Usage

```
new_fe_ccircle_spatial(x = list(), ..., class = character())
```

Arguments

x	An appropriate list object
...	Additional arguments required for enabling subclasses of fe_ccircle_spatial
class	A Character string required for enabling subclasses of fe_ccircle_spatial

Value

An object of class `fe_ccircle_spatial`

Examples

```
#' # Constructing a minimal fe_ccircle_spatial object from scratch
# Use fe_ccircle_spatial() if you are not absolutely sure

trees <- data.frame(
  tree_id = as.character(c(1:10)),
  species_id = as_fe_species_tum_wwk_short(rep("5", 10)),
  layer_key = 1,
  time_yr = 2024,
  dbh_cm = rnorm(10, 50, 8),
  age_yr = NA_real_,
  height_m = NA_real_,
  crown_base_height_m = NA_real_,
  crown_radius_m = NA_real_,
  removal = FALSE,
  ingrowth = FALSE,
  n_rep_ha = 1 / 0.75
)

# define a circle definition with three concentric circles
circle_def <- data.frame(
  dbh_lower = c(0,12,30),
  dbh_upper = c(11.9,29.9,999.0),
  c_area = c(0.0025, 0.0060, 0.0500)
)

# generate tree positions in polar coordinates for the 10 trees

tree_positions <- data.frame(
 tree_id = as.character(c(1:10)),
 R = runif(10, 1, 12),
 angle = runif(10, 0, 360)) |>
 dplyr::mutate(
 #convert the polar coordinates to cartesian and into an sf object
 x_pos = R*sin(angle*pi / 180),
 y_pos = R*cos(angle*pi / 180)
 ) |> sf::st_as_sf(coords = c("x_pos", "y_pos"))

# generate a NA dummy for small_trees

  small_trees <- data.frame(
  tree_id = NA_character_,
  species_id = NA_character_,
  layer_key = NA_real_,
  time_yr = NA_real_,
  dbh_cm = NA_real_,
  age_yr = NA_real_,
  height_m = NA_real_
```

```
  )

fe_ccircle_spatial_candidate <- list(
  stand_id = "my_interesting_stand",
  small_trees = small_trees,
  trees = trees,
  circle_definition = circle_def,
  tree_positions = tree_positions,
  time_yr = 2024
)

fe_ccircle_object <- new_fe_ccircle_spatial(fe_ccircle_spatial_candidate)

# Better validate it
fe_ccircle_object |> validate_fe_ccircle_spatial()
```

`new_fe_ccircle_spatial_notrees` *Constructor for the* **fe_ccircle_spatial_notrees** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_stand` for creating an object of that class.

Usage

```
new_fe_ccircle_spatial_notrees(x = list(), ..., class = character())
```

Arguments

`x`	An appropriate `list` object
`...`	Additional arguments required for enabling subclasses of `fe_ccircle_spatial_notrees`
`class`	A Character string required for enabling subclasses of `fe_ccircle_spatial_notrees`

Details

The class `fe_ccircle_spatial_notrees` has been designed for covering a comparably rare case, i.e. an inventory point where no regular trees (trees that are big enough to have a dbh), but possibly small trees are present.

Value

An object of class `fe_ccircle_spatial_notrees`

Examples

```
#' # Constructing a minimal fe_ccircle_spatial object from scratch
# Use fe_ccircle_spatial() if you are not absolutely sure

trees <- NULL
tree_positions <- NULL

# define a circle definition with three concentric circles
circle_def <- data.frame(
  dbh_lower = c(0,12,30),
  dbh_upper = c(11.9,29.9,999.0),
  c_area = c(0.0025, 0.0060, 0.0500)
)


# generate a NA dummy for small_trees

  small_trees <- data.frame(
  tree_id = NA_character_,
  species_id = NA_character_,
  layer_key = NA_real_,
  time_yr = NA_real_,
  dbh_cm = NA_real_,
  age_yr = NA_real_,
  height_m = NA_real_
  )

fe_ccircle_spatial_notrees_candidate <- list(
  stand_id = "my_interesting_stand",
  small_trees = small_trees,
  trees = trees,
  circle_definition = circle_def,
  tree_positions = tree_positions,
  time_yr = 2024
)

fe_ccircle_notrees_object <-
new_fe_ccircle_spatial_notrees(fe_ccircle_spatial_notrees_candidate)

# Better validate it
fe_ccircle_notrees_object |> validate_fe_ccircle_spatial_notrees()
```

`new_fe_species_bavrn_state` *Constructor for the* **fe_species_bavrn_state** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_species_bavrn_state` for creating an object of that class.

Usage

```
new_fe_species_bavrn_state(x = character())
```

Arguments

x	An appropriate `character` vector

Value

An object of class `fe_species_bavrn_state`

Examples

```
# Constructing a fe_species_bavrn_state object from scratch
# Use fe_species_bavrn_state() if you are not absolutely sure
spec_ids <- new_fe_species_bavrn_state(
  as.character(c(40, 40, 30, 30, 60, 60, 60))
)
```

---

new_fe_species_bavrn_state_short

Constructor for the **fe_species_bavrn_state_short** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_species_bavrn_state_short` for creating an object of that class.

Usage

```
new_fe_species_bavrn_state_short(x = character())
```

Arguments

x	An appropriate `character` vector

Value

An object of class `fe_species_bavrn_state_short`

Examples

```
# Constructing a fe_species_bavrn_state_short object from scratch
# Use fe_species_bavrn_state_short() if you are not absolutely sure
spec_ids <- new_fe_species_bavrn_state_short(
  as.character(1:9)
)
```

`new_fe_species_ger_nfi_2012`
Constructor for the **fe_species_ger_nfi_2012** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_species_ger_nfi_2012` for creating an object of that class.

Usage

```
new_fe_species_ger_nfi_2012(x = character())
```

Arguments

`x`	An appropriate `character` vector

Value

An object of class `fe_species_ger_nfi_2012`

Examples

```
# Constructing a fe_species_ger_nfi_2012 object from scratch
# Use fe_species_ger_nfi_2012() if you are not absolutely sure
spec_ids <- new_fe_species_ger_nfi_2012(
  as.character(c(50, 50, 20, 20, 100, 100, 100))
)
```

new_fe_species_master *Constructor for the* **fe_species_master** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_species_master` for creating an object of that class.

Usage

```
new_fe_species_master(x = character())
```

Arguments

x	An appropriate `character` vector

Value

An object of class `fe_species_master`

Examples

```
# Constructing a fe_species_master object from scratch
# Use fe_species_master() if you are not absolutely sure
spec_ids <- new_fe_species_master(
  c("picea_001", "fagus_001", "quercus_002", "quercus_001")
)
```

new_fe_species_tum_wwk_long *Constructor for the* **fe_species_tum_wwk_long** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_species_tum_wwk_long` for creating an object of that class.

Usage

```
new_fe_species_tum_wwk_long(x = character())
```

Arguments

x	An appropriate `character` vector

Value

An object of class `fe_species_tum_wwk_long`

Examples

```
# Constructing a fe_species_tum_wwk_long object from scratch
# Use fe_species_tum_wwk_long() if you are not absolutely sure
spec_ids <- new_fe_species_tum_wwk_long(
  as.character(c(70, 61, 88, 88, 10, 971, 32))
)
```

`new_fe_species_tum_wwk_short`
Constructor for the **fe_species_tum_wwk_short** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_species_tum_wwk_short` for creating an object of that class.

Usage

```
new_fe_species_tum_wwk_short(x = character())
```

Arguments

`x`	An appropriate `character` vector

Value

An object of class `fe_species_tum_wwk_short`

Examples

```
# Constructing a fe_species_tum_wwk_short object from scratch
# Use fe_species_tum_wwk_short() if you are not absolutely sure
spec_ids <- new_fe_species_tum_wwk_short(
  as.character(c(4, 4, 3, 3, 5, 5, 5))
)
```

new_fe_stand *Constructor for the* **fe_stand** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_stand` for creating an object of that class.

Usage

```
new_fe_stand(x = list(), ..., class = character())
```

Arguments

`x`	An appropriate `list` object
`...`	Additional arguments required for enabling subclasses of `fe_stand`
`class`	A Character string required for enabling subclasses of `fe_stand`

Value

An object of class `fe_stand`

Examples

```
# Constructing a minimal fe_stand object from scratch
# Use fe_stand() if you are not absolutely sure

trees <- data.frame(
  tree_id = as.character(c(1:100)),
  species_id = as_fe_species_tum_wwk_short(rep("5", 100)),
  layer_key = 1,
  time_yr = 2022,
  dbh_cm = rnorm(100, 50, 8),
  age_yr = NA_real_,
  height_m = NA_real_,
  crown_base_height_m = NA_real_,
  crown_radius_m = NA_real_,
  removal = FALSE,
  ingrowth = FALSE,
  n_rep_ha = 1 / 0.75
)

fe_stand_candidate <- list(
  stand_id = "my_interesting_stand",
  area_ha = 0.75,
  small_trees = data.frame(),
  trees = trees
)
```

```
fe_stand_object <- new_fe_stand(fe_stand_candidate)

# Better validate it
fe_stand_object |> validate_fe_stand()
```

`new_fe_stand_spatial` *Constructor for the* **fe_stand_spatial** *Class*

Description

Should be used by expert users only who know exactly what they are doing. Other users please take the function `fe_stand` for creating an object of that class.

Usage

```
new_fe_stand_spatial(x = list(), ..., class = character())
```

Arguments

`x`	An appropriate `list` object
`...`	Additional arguments required for enabling subclasses of `fe_stand_spatial`
`class`	A Character string required for enabling subclasses of `fe_stand_spatial`

Value

An object of class `fe_stand_spatial`

Examples

```
#' # Constructing a minimal fe_stand_spatial object from scratch
# Use fe_stand_spatial() if you are not absolutely sure

trees <- data.frame(
  tree_id = as.character(c(1:50)),
  species_id = as_fe_species_tum_wwk_short(rep("5", 50)),
  layer_key = 1,
  time_yr = 2024,
  dbh_cm = rnorm(50, 50, 8),
  age_yr = NA_real_,
  height_m = NA_real_,
  crown_base_height_m = NA_real_,
  crown_radius_m = NA_real_,
  removal = FALSE,
  ingrowth = FALSE,
  n_rep_ha = 1 / 0.75
)

# generate tree positions
```

```
tree_positions <- data.frame(
 tree_id = as.character(c(1:50)),
 x_pos = runif(50, 1, 49),
 y_pos = runif(50, 1, 49)
 ) |> sf::st_as_sf(coords = c("x_pos", "y_pos"))


fe_stand_spatial_candidate <- list(
  stand_id = "my_interesting_stand",
  outline = NULL,
  orientation = 0,
  small_trees = data.frame(),
  trees = trees,
  tree_positions = tree_positions
)

fe_stand_object <- new_fe_stand_spatial(fe_stand_spatial_candidate)

# Better validate it
fe_stand_object |> validate_fe_stand_spatial()
```

`new_fe_yield_table` *Constructor for the* **fe_yield_table** *Class*

Description

Provided for expert users only who know exactly what they are doing. Other users please take the function `fe_yield_table` for creating an object of that class.

Usage

```
new_fe_yield_table(x = list())
```

Arguments

`x`	An appropriate `list` object

Value

An object of class `fe_yield_table`

Examples

```
# Some highly motivated object, even a list
x <- list(my_dream = "I want to be a yield table!")
x <- new_fe_yield_table(x) # let's help the guy
is_fe_yield_table(x)       # nice - it worked?!

# But here's the error. That's why you should not use the bare constructor
# if you don't know what you are doing.
try(
  validate_fe_yield_table(x)
)
```

`n_rep_ha` *Calculate or Return the Representation Number per ha for the Trees Contained in a Compatible Object*

Description

The idea behind creating this function was to allow for using the very same evaluation algorithms for ha-based values for a broad range of different objects, e.g. stands/research plots, inventory plots, etc.

Usage

```
n_rep_ha(x)
```

Arguments

`x`	An object provided by **ForestElementsR** containing trees for which a representation number per ha can be meaningfully given. Such an object must contain a data frame called 'trees'. Typically, this is an `fe_stand` or `fe_stand_spatial` object.

Details

If the object `x` contains information about its area, the representation numbers for each tree will be calculated in the following way:

While `n_rep_ha` will return a vector of equal numbers for `fe_stand` objects, this is less trivial for the class `fe_stand_spatial`. The latter might contain 'buffer zone trees' beyond the actual stand outline. Such trees will obtain a zero representation number in contrast to the trees inside the outline. If `x` does not contain sufficient information about its area, the function will simply hand back the `n_rep_ha` column of the `x$trees` data frame.

Value

A vector of representation numbers corresponding to the order of the trees data frame in `x`.

Examples

```
# example for an fe_stand object
spruce_beech_1_fe_stand |> n_rep_ha()

# example for an fe_stand_spatial object
mm_forest_1_fe_stand_spatial |> n_rep_ha()
```

`plot.fe_ccircle_spatial` *Plot an fe_ccircle_spatial Object*

Description

Plot an fe_ccircle_spatial Object

Usage

```
## S3 method for class 'fe_ccircle_spatial'
plot(x, tree_filter = TRUE, dbh_scale = 1, show_labels = FALSE, ...)
```

Arguments

`x`	An `fe_ccircle_spatial` object
`tree_filter`	A `data-masking` expression that applies to the data.frame `x$trees`. It must return a logical value, and is defined in terms of the variables in `x$trees`. In this function, it is used internally in order to define the cohort of trees which is to be evaluated by this function (within a call to `dplyr::filter()`). While many meaningful filterings are conceivable, distinctions between total stand, removal stand, and remaining stand are the most probable applications. Defaults to `TRUE`, i.e. all trees are included. See examples.
`dbh_scale`	Scaling factor for plotting tree dbh in order to allow oversized representations. Defaults to 1 (correct scaling)
`show_labels`	Logical value. If TRUE, labels for species, dbh, R and angle are displayed.
`...`	Additional arguments, not used in `plot.fe_ccircle_spatial`

Value

A map (ggplot2) of the plot layout including the trees with coordinates

Examples

```
opt_old <- getOption("fe_spec_lang") # store user's current setting
options(fe_spec_lang = "eng")      # choose Englisch species name display
spruce_pine_ccircle_spatial |> plot()
spruce_pine_ccircle_spatial |> plot(dbh_scale = 4)
spruce_pine_ccircle_spatial |>
  plot(
    dbh_scale = 4,
    tree_filter = species_id == fe_species_tum_wwk_long(30) & dbh_cm > 35
  )
options(fe_spec_lang = opt_old)
```

`plot.fe_ccircle_spatial_notrees`
Plot an fe_ccircle_spatial_notrees Object

Description

For an object of classfe_ccircle_spatial_notrees only the concentric circles are plotted at their correct positions.

Usage

```
## S3 method for class 'fe_ccircle_spatial_notrees'
plot(x, ...)
```

Arguments

x	An fe_ccircle_spatial_notrees object
...	Additional arguments, not used in plot.fe_ccircle_spatial_notrees

Value

A map (ggplot2) of the plot layout including the trees with coordinates

Examples

```
# Abuse input data, that would actually allow for a full fe_ccircle_spatial
# object to construct a fe_ccircle_spatial_notrees object.
x <- spruce_pine_ccircle_raw
x <- x |> fe_ccircle_spatial_notrees()

# Plot it
plot(x)
```

`plot.fe_stand` *Plot an* **fe_stand** *Object*

Description

Diameter distributions in number of trees per ha, one diagrame by year of survey, tree cohort to be displayed can be filtered.

Usage

```
## S3 method for class 'fe_stand'
plot(x, tree_filter = TRUE, ...)
```

Arguments

`x`	an `fe_stand` object
`tree_filter`	A `data-masking` expression that applies to the data.frame `x$trees`. It must return a logical value, and is defined in terms of the variables in `x$trees`. In this function, it is used internally in order to define the cohort of trees which is to be evaluated by this function (within a call to `dplyr::filter()`). While many meaningful filterings are conceivable, distinctions between total stand, removal stand, and remaining stand are the most probable applications. Defaults to `TRUE`, i.e. all trees are included. See examples.
`...`	additional arguments, not used in `plot.fe_stand`

Details

The diagram(s) are made with `ggplot2::geom_bar`, the colours for the species, the number and width of the diameter bins correspond to the default settings in ggplot.

Value

A plot (ggplot2) of the diameter distribution

Examples

```
# display scientific species names in all examples
old_opt <- getOption("fe_spec_lang") # store current user sertting
options(fe_spec_lang = "sci") # display scientific names

# mixed mountain forest - all trees
mm_forest_1_fe_stand_spatial |> plot()
# ... remaining trees only
mm_forest_1_fe_stand_spatial |> plot(tree_filter = !removal)
# ... removal only
mm_forest_1_fe_stand_spatial |> plot(tree_filter = removal)
# ... all trees with dbh > 30 cm
mm_forest_1_fe_stand_spatial |> plot(tree_filter = dbh_cm > 20)
```

```
# other example stands
selection_forest_1_fe_stand |> plot()
norway_spruce_1_fe_stand |> plot()
spruce_beech_1_fe_stand |> plot()

# reset to previous species name settings
options(fe_spec_lang = old_opt)
```

`plot.fe_yield_table` *Plot an fe_yield_table Object*

Description

Plot an fe_yield_table Object

Usage

```
## S3 method for class 'fe_yield_table'
plot(x, variable = NA, ...)
```

Arguments

`x`	An object of class `fe_yield_table`
`variable`	Character, name of the variable to be plotted, default is `NA`, which plots the variable listed first in the fe_yield_table object's `site_index_variable` `slot`.
`...`	Other parameters, not used

Value

An object of class ggplot

See Also

Other yield table functions: `fe_yield_table()`, `si_to_mai_age()`, `si_to_mai_max()`, `site_index()`, `stocking_level()`, `yield_tables_for_species`, `ytable_age_slice()`, `ytable_lookup()`, `ytable_max_slice()`

Examples

```
fe_ytable_pine_wiedemann_moderate_1943 |> plot()
fe_ytable_pine_wiedemann_moderate_1943 |> plot(variable = "ba_m2_ha")
fe_ytable_pine_wiedemann_moderate_1943 |> plot(variable = "mai_m3_ha_yr")

# Modify plot post hoc ...
fe_ytable_pine_wiedemann_moderate_1943 |> plot(variable = "n_ha")
# ... better to read on log scale
fe_ytable_pine_wiedemann_moderate_1943 |> plot(variable = "n_ha") +
  ggplot2::scale_y_log10()
```

`se_tests` *Convenient Information About the Precision of a Random Sample*

Description

Given an arbitrary random sample, t-statistics are calculated in order to obtain information about the precision of the sample mean or sum. A list of useful statistics is returned, most importantly the standard error and the confidence boundaries according to the confidence level provided by the user.

Usage

```
se_tests(
  x,
  mu = 0,
  ref = c("mean", "sum"),
  alternative = c("two.sided", "greater", "less"),
  conf.level = 0.95
)
```

Arguments

`x`	vector representing the sample to be evaluated
`mu`	the hypothesized mean for the null hypothesis, standard = 0
`ref`	the reference value for the satistic, whether it is calculated from the mean or the sum, default = 'mean'
`alternative`	a character string specifying the alternative hypothesis for the t-statistics: "two-sided", "greater" or "less". Default is "two-sided"
`conf.level`	confidence level of the interval, Default = 0.95

Value

A list containing various statistics such as standard error, t-statistic, degrees of freedom, p-value, and confidence interval and margin of error as a percentage of the mean or sum, respectively.

Examples

```
set.seed(123)
data <- rnorm(100, mean = 5, sd = 2)

# Test for sum
se_tests(
  data, mu = 0, ref = "sum", alternative = "two.sided", conf.level = 0.95
)
# Test for mean
se_tests(
  data, mu = 0, ref = "mean", alternative = "two.sided", conf.level = 0.95
```

```
)
```

---

shannon_index *Shannon Diversity Index for Tree Species*

Description

Species diversity index after Shannon and Weaver (1948). Note that this function calculates comparable output only when the same species coding is used for the input parameter `species_id`.

Usage

```
shannon_index(species_id, weights = 1, n_rep = 1)
```

Arguments

`species_id`	A vector of species codes, each vector element representing a tree. Preferably, `species_id` is defined in one of the species codings supported by this package, but technically, this is not even a requirement.
`weights`	A vector of weights for each tree, default = 1, i.e. all trees are equally weighted. Must be of length 1 or the same length as `species_id`. Useful if e.g. trees should be weighted by their basal area.
`n_rep`	A vector of representation numbers for each tree, typically the number of trees represented per ha by each tree. Does only make a difference if it differs among the trees. Default = 1, i.e. all trees have the same representation number.

Value

The Shannon Index value resulting from the input data

References

Shannon CE, Weaver W (1948). *The Mathematical Theory of Communication*. University of Illinois Press.

See Also

Other structure and diversity: `assmann_layers()`, `species_profile()`

Examples

```
# Monospecific stand
trees <- norway_spruce_1_fe_stand$trees
shannon_index(trees$species_id)

# Two-species mixed stand
trees <- spruce_beech_1_fe_stand$trees
```

```
shannon_index(trees$species_id)

# Selection forest
trees <- selection_forest_1_fe_stand$trees
shannon_index(trees$species_id)

# weigh with basal area (i.e. dbh^2)
shannon_index(trees$species_id, weights = trees$dbh_cm^2)

# weigh with inverse basal area (i.e. 1 / dbh^2)
shannon_index(trees$species_id, weights = 1 / trees$dbh_cm^2)
```

`site_index` *Find Site Indexes With a Yield Table*

Description

Find Site Indexes With a Yield Table

Usage

```
site_index(age, size, ytable, si_variable)
```

Arguments

`age`	Age (years) of the stand to be site indexed
`size`	Size value, typically a height (m), of the stand to be site indexed. Must correspond to the parameter `si_variable` (see below)
`ytable`	A yield table, must be an `fe_yield_table` object
`si_variable`	Name of the stand size variable, typically a height (m), to be used for site indexing. Must correspond to the parameter `size` (see above). If `si_variable` is not part of the yield table object's slot `$site_index_variable`, the function will terminate with an error.

Value

The site index resulting from `age` and `height`

See Also

Other yield table functions: `fe_yield_table()`, `plot.fe_yield_table()`, `si_to_mai_age()`, `si_to_mai_max()`, `stocking_level()`, `yield_tables_for_species`, `ytable_age_slice()`, `ytable_lookup()`, `ytable_max_slice()`

Examples

```
site_index(72, 19.7, fe_ytable_pine_wiedemann_moderate_1943, "h_q_m")
```

si_to_mai_age *Convert a Standard Site Index Into an MAI(age) Site Index*

Description

A useful way of site indexing is to give the site index of a stand in terms of a mean annual increment (mai) at a given age, typically 100 years. This function converts a standard site index into such an mai site index. See si_to_mai_max for an alternative mai based site indexing method.

Usage

```
si_to_mai_age(si, mai_variable, age, ytable)
```

Arguments

si	Standard site index to be converted, must correspond to the site index nomenclature of the yield table to be used (param ytable, see below).
mai_variable	Character, name of the mai_variable to be used. Must be one if the mai variables listed in the fe_yield_table object provided with the parameter ytable.
age	The stand age (years) for which the mai site index is to be defined, typically 100 years.
ytable	An object of class fe_yield_table

Value

The requested mai value corresponding to the given standard site index at the given age

See Also

Other yield table functions: fe_yield_table(), plot.fe_yield_table(), si_to_mai_max(), site_index(), stocking_level(), yield_tables_for_species, ytable_age_slice(), ytable_lookup(), ytable_max_slice()

Examples

```
age <- 100
mai_var <- "mai_m3_ha_yr" # mai in volume over bark before harvest

si_to_mai_age(2.3, mai_var, age, fe_ytable_larch_schober_moderate_1946)
si_to_mai_age(0.7, mai_var, age, fe_ytable_larch_schober_moderate_1946)
si_to_mai_age(2.3, mai_var, age, fe_ytable_beech_wiedemann_moderate_1931)
si_to_mai_age(0.7, mai_var, age, fe_ytable_beech_wiedemann_moderate_1931)

mai_var <- "red_mai_m3_ha_yr" # mai in vol. under bark minus harvest losses
si_to_mai_age(2.3, mai_var, age, fe_ytable_larch_schober_moderate_1946)
si_to_mai_age(0.7, mai_var, age, fe_ytable_larch_schober_moderate_1946)
si_to_mai_age(2.3, mai_var, age, fe_ytable_beech_wiedemann_moderate_1931)
si_to_mai_age(0.7, mai_var, age, fe_ytable_beech_wiedemann_moderate_1931)
```

si_to_mai_max *Convert a Standard Site Index Into an maximum MAI Site Index*

Description

A less common, but sometimes useful way of site indexing is to give the site function converts a standard site index into such an maximum mai site index. Typically, the stand age where the maximum mai is obtained increases from better to lesser site index classes. See si_to_mai_age for an alternative mai based site indexing method.

Usage

```
si_to_mai_max(si, mai_variable, ytable)
```

Arguments

si	Standard site index to be converted, must correspond to the site index nomenclature of the yield table to be used (param ytable, see below).
mai_variable	Character, name of the mai_variable to be used. Must be one if the mai variables listed in the fe_yield_table object provided with the parameter ytable.
ytable	An object of class fe_yield_table

Value

The requested maximum mai value corresponding to the given standard site index

See Also

Other yield table functions: fe_yield_table(), plot.fe_yield_table(), si_to_mai_age(), site_index(), stocking_level(), yield_tables_for_species, ytable_age_slice(), ytable_lookup(), ytable_max_slice()

Examples

```
mai_var <- "mai_m3_ha_yr" # mai in volume over bark before harvest

si_to_mai_max(2.3, mai_var, fe_ytable_larch_schober_moderate_1946)
si_to_mai_max(0.7, mai_var, fe_ytable_larch_schober_moderate_1946)
si_to_mai_max(2.3, mai_var, fe_ytable_beech_wiedemann_moderate_1931)
si_to_mai_max(0.7, mai_var, fe_ytable_beech_wiedemann_moderate_1931)

mai_var <- "red_mai_m3_ha_yr" # mai in vol. under bark minus harvest losses
si_to_mai_max(2.3, mai_var, fe_ytable_larch_schober_moderate_1946)
si_to_mai_max(0.7, mai_var, fe_ytable_larch_schober_moderate_1946)
si_to_mai_max(2.3, mai_var, fe_ytable_beech_wiedemann_moderate_1931)
si_to_mai_max(0.7, mai_var, fe_ytable_beech_wiedemann_moderate_1931)
```

species_codings *Supported Species Codings in the Package* **ForestElementsR**

Description

Data of all supported species codings in the package `ForestElementsR`. Currently supported codings are

master The *master* species coding is the original species coding used by the package **ForestElementsR**. It contains each species from the `species_master_table` and no species groups. This coding corresponds directly to the `species_master_table`. Its species_id's (see below) are the master table's columns `genus` and `species` combined into one character string, separated by an underscore.

tum_wwk_short The *tum_wwk_short* species coding is one of two codings in use at the Chair of Forest Growth and Yield Science. It defines only a small set of single species explicitly (the most important ones in Central Europe), while all other species are attributed to a few large container groups.

tum_wwk_long The *tum_wwk_long* species coding is one of two codings in use at the Chair of Forest Growth and Yield Science. It defines a larger set of single species than the *tum_wwk_short* coding. In its original version, this coding contains several species groups, but most of these groups are ambiguous as they include species which also have a single coding. These ambiguous groups were not included in this package.

bavrn_state The *bavrn_state* species coding is the species coding used by the Bavarian State Forest Service.

bavrn_state_short The *bavrn_state_short* is a coding that combines the species of *bavrn_state* into groups. These groups are typically used by the Bavarian State Forest Service in aggregated evaluations.

ger_nfi_2012 The *ger_nfi_2012* species coding is the species coding used by the German National Forest Inventory of 2012 (Riedel et al. 2017)

Usage

```
species_codings
```

Format

A tibble containing the supported species codings together with the coding tables (which are tibbles themselves). Its columns are:

species_coding name of the coding

code_table tibble describing the species coding with the columns

species_id the species code (character)

species_name_sci the scientific species name (for species groups english terms are used)

species_name_eng English species names

species_name_ger German species names

References

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

Examples

```
# Get specific coding tables out of the data 'species_codings'
fe_species_get_coding_table("master")
fe_species_get_coding_table("tum_wwk_short")
fe_species_get_coding_table("tum_wwk_long")
fe_species_get_coding_table("bavrn_state")
fe_species_get_coding_table("bavrn_state_short")
fe_species_get_coding_table("ger_nfi_2012")

# Check number of species behind each code in a given coding
fe_species_get_coding_table("tum_wwk_short") |>
  dplyr::group_by(species_id) |>
  dplyr::summarise(n = dplyr::n()) |>
  dplyr::arrange(as.numeric(species_id)) # just for the look of it

fe_species_get_coding_table("bavrn_state_short") |>
  dplyr::group_by(species_id) |>
  dplyr::summarise(n = dplyr::n()) |>
  dplyr::arrange(as.numeric(species_id)) # just for the look of it
```

---

species_master_table *List of Supported Species in the Package* **ForestElementsR**

Description

A `data.frame` (`tibble`) where each row represents a single species which is - in principle - currently supported by the package **ForestElementsR**. Each species which is coded (as a single species or a group of species) in any of the supported `species_codings`, must also be represented in this data frame. No specific species coding, however, must necessarily use all of the species listed here.

The internal universal species coding used in this package is defined in the two columns `genus` and `species_no` (see below).

Usage

```
species_master_table
```

Format

A tibble with the following columns are:

deciduous_conifer `Character` column with allowed values `conif`, and `decid`for confifer and deciduous species, respectively

genus Character, the genus of the species in plaintext, but always lowercase

species_no Character, a three digit number, always with leading zeroes, indicating the species inside the genus. A number was chosen instead of plaintext, because spelling and naming must be considered somewhat instable on that level. The order of the numbers reflects (very) roughly the importance of the species inside a given genus in Central Europe.

name_sci Scientific species names (character)

name_eng Colloquial English species names (character)

name_ger Colloquial German species names (character)

---

species_profile *Species Profile Index After Pretzsch*

Description

As an extension of the Shannon Index (shannon_index), the species profile index by Pretzsch (2009) takes into account the vertical structure of a forest stand. For doing so, the function assmann_layers is called in the background.

Usage

```
species_profile(
  species_id,
  heights,
  weights = 1,
  n_rep = 1,
  reference_height = NULL
)
```

Arguments

species_id	A vector of species codes, each vector element representing a tree. Preferably, species_id is defined in one of the species codings supported by this package, but technically, this is not even a requirement.
heights	A vector of tree heights, must have the same length as species_id
weights	A vector of weights for each tree, default = 1, i.e. all trees are equally weighted. Must be of length 1 or the same length as species_id. Useful if e.g. trees should be weighted by their basal area.
n_rep	A vector of representation numbers for each tree, typically the number of trees represented per ha by each tree. Does only make a difference if it differs among the trees. Default = 1, i.e. all trees have the same representation number.
reference_height	Reference height for the 100% level of the stand height profile. Internally passed to assmann_layers. If NULL (default), the maximum of heights will be used as the reference height.

Details

Note that this function calculates comparable output only when the same species coding is used for the input parameter `species_id`.

Value

The Species Profile Index value resulting from the input data

References

Pretzsch H (2009). *Forest Dynamics, Growth and Yield: From Measurement to Model*, 2010 edition. Springer, Berlin. ISBN 978-3-540-88306-7.

See Also

Other structure and diversity: `assmann_layers()`, `shannon_index()`

Examples

```
# Monospecific stand
trees <- norway_spruce_1_fe_stand$trees
species_profile(trees$species_id, trees$height_m)

# Two-species mixed stand
trees <- spruce_beech_1_fe_stand$trees
species_profile(trees$species_id, trees$height_m)

# Selection forest
trees <- selection_forest_1_fe_stand$trees
species_profile(trees$species_id, trees$height_m)

# weigh with basal area (i.e. dbh^2)
species_profile(trees$species_id, trees$height_m, weights = trees$dbh_cm^2)

# weigh with inverse basal area (i.e. 1 / dbh^2)
species_profile(
  trees$species_id, trees$height_m, weights = 1 / trees$dbh_cm^2
)
```

---

species_shares *species_shares*

Description

Calculate tree species shares for a `fe_stand` object. Different methods and scopes are available.

Usage

```
species_shares(
  x,
  tree_filter = TRUE,
  method = c("ba_wd", "ba", "n"),
  include_ingrowth = TRUE
)
```

Arguments

x	An fe_stand object
tree_filter	A data-masking expression that applies to the data.frame x$trees. It must return a logical value, and is defined in terms of the variables in x$trees. In this function, it is used internally in order to define the cohort of trees which is to be evaluated by this function (within a call to dplyr::filter()). While many meaningful filterings are conceivable, distinctions between total stand, removal stand, and remaining stand are the most probable applications. Defaults to TRUE, i.e. all trees are included. See examples.
method	Character string defining the calculation method to be applied. Must be one of "ba_wd" (default), "ba", and "n" (see Details).
include_ingrowth	If TRUE (default), newly ingrown trees will be included in the calculation.

Details

The calculation uses the trees data frame of the input fe_stand object. The small tree cohort is not taken into account. Three different methods are available to choose from (parameter method:

Basal area shares, weighted with wood density (method "ba_wd"): The species shares are based on basal areas which are weighted with the species specific wood densities raised to the power of 2/3. The exponent of 2/3 takes into account that wood density is a three-dimensional quantity, while basal area is two-dimensional. This is the default method. It works, however, only with species codings that can be converted into the fe_species_tum_wwk_short coding (default), or with fe_species_bavrn_state_short. The latter is used if this species coding is directly provided or if the species coding is fe_species_bavrn_state. The reason for that restriction is that wood densities are currently only provided for the two species codings fe_species_tum_wwk_short, and fe_species_bavrn_state_short. The resulting shares, however, will always relate to the original coding.

Unweighted basal area shares (method "ba"): Species shares are calculated as shares of the unweighted basal areas.

Stem number shares (method "n"): Species shares are calculated as stem number shares, i.e. tree size does not matter for that calculation.

Value

A data frame (tibble) with the three columns species_id, time_yr, and species_share. If no tree passes the user-defined tree_filter, the tibble will have no lines.

Examples

```
species_shares(selection_forest_1_fe_stand) # default method ("ba_wd")
species_shares(selection_forest_1_fe_stand, method = "ba")
species_shares(selection_forest_1_fe_stand, method = "n")

# Same stand, different cohorts
mm_forest_1_fe_stand_spatial |> species_shares() # all trees
mm_forest_1_fe_stand_spatial |> species_shares(!removal) # remaining only
mm_forest_1_fe_stand_spatial |> species_shares(removal) # removal only
```

---

standing_area_gnfi3 *Estimate the Standing Area of Single Trees*

Description

An implementation of the standing area estimation of the third German National Forest Inventory (Riedel et al. 2017). Its main intended use is the calculation of virtual species areas in mixed stands. According to (Riedel et al. 2017), it is recommended only to include the main stand in such calculations, neither understorey, nor any layers above the main stand.

Usage

```
standing_area_gnfi3(species_id, dbh_cm)
```

Arguments

species_id	Vector of species id's preferably following the *ger_nfi_2012* species coding. Ideally, these species_id's are provided as a fe_species_ger_nfi_2012 object. See Details for how other species codings are handled.
dbh_cm	Vector of tree dbh values in cm (dbh = stem diameter at breast height, i.e. 1.3 m)

Details

Originally, the function was parameterized for species and species groups corresponding to the national forest inventory's species coding (fe_species_ger_nfi_2012). We have attributed in addition these the original parameters also to the species codings fe_species_tum_wwk_short, and fe_species_bavrn_state_short. When called with a given species coding, the function will try to use the "nearest" of these three alternatives. Fallback option is the attempt to use fe_species_tum_wwk_short.

Value

A vector of the estimated standing areas in m²

Examples

```
# Three spruces, two pines, two beech
species_id <- fe_species_ger_nfi_2012(c(10, 10, 10, 20, 20, 100, 100))
dbh_cm     <- c(10.1, 27.4, 31.4, 35.5, 39.8, 45.2, 47.2)

standing_area_gnfi3(species_id, dbh_cm)
```

---

stand_level_increment *stand_level_increment*

Description

Calculate periodic annual stand level increments from time-ordered vectors of appropriate stand sum variables. Typically the variable of interest is a stand's wood volume per unit area, but it works equally for stand basal area and biomass.

Usage

```
stand_level_increment(time, x_remain, x_remove)
```

Arguments

time	Vector of points in time. Must be unique and in ascending order
x_remain	Vector of the variable of interest (typically wood volume) for the remaining stand. "Remaining stand" means the amount which is actually there at the corresponding point in time. The vector x_remaing must be arranged corresponding to the input vector time.
x_remove	Vector of the variable of interest (typically wood volume) for the removal stand. Each entry in this vector represents the amount removed up to and including the corresponding point of time, but after the previous point in time.

Details

The input vector x_remove is to be understood as the amount removed up to (or at) the corresponding points in time and after the preceding points in time in the time input vector. The resulting increments are to be understood in a similar way: The entries in the increment vector relate to the period between the entry at the same position in the time input vector and the time entry at the position before. Therefore, the first element of the resulting increment vector is always NA.

Value

A vector of the annual increments for the stand level variable of interest corresponding to the input vector time. The increments always relate to the period from (and including) the corresponding point in time back to (and excluding) the previous point in time. Thus, the first element of the output vector is always NA. If the input vectors are of length 1 only, the function consequently returns NA.

Examples

```
# Stand age, remaining and removal volume (m³/ha)
age        <- seq(20, 70, 5)
vol_remain <- c(65, 118, 175, 233, 293, 355, 416, 476, 534, 589, 642)
vol_remove <- c(16,  29,  35,  39,  39,  39,  38,  37,  36,  35,  34)

stand_level_increment(age, vol_remain, vol_remove) # m³/ha/yr

# Works also with basal area (m²/ha)
age        <- seq(20, 60, 5)
ba_remain  <- c(26.0, 30.1, 32.5, 34.2, 35.5, 37.1, 38.7, 40.3, 41.9)
ba_remove  <- c( 0.0,  5.0,  5.9,  5.4,  5.1,  4.2,  3.3,  3.0,  2.7)

stand_level_increment(age, ba_remain, ba_remove) # m²/ha/yr
```

`stand_sums_dynamic` *stand_sums_dynamic*

Description

Calculate periodic annual volume and basal area increments for `fe_stand` objects with repeated surveys

Usage

```
stand_sums_dynamic(x, tree_filter = TRUE)
```

Arguments

`x`	An `fe_stand` object
`tree_filter`	A `data-masking` expression that applies to the data.frame `x$trees`. It must return a logical value, and is defined in terms of the variables in `x$trees`. In this function, it is used internally in order to define the cohort of trees which is to be evaluated by this function (within a call to `dplyr::filter()`). In order to obtain meaningful increments, you should be very careful when changing the default value (`TRUE`) of `tree_filter`.

Details

For the sake of robustness, `stand_sums_dynamic` does not perform a plausibility check on single tree level before calculating increments. Internally, the function `stand_sums_static` is called separately for the remaining and the removal stand. Both are required for calculating meaningful increments. Basal area increments are always calculated, because valid `fe_stand` objects always contain the required information (i.e. all trees' dbh); volume increments are calculated if also height values (either measured or estimated) are available for all trees.

Value

A data frame (tibble) that contains the periodic annual basal area and volume growth (the latter if enough information is available, see Details) per ha for each species and each period. The year where the increment entry is, means the end point of the period of interest. Therefore, the first increment value will always be NA. If the input `fe_stand` object `x` does only comprise one survey, the increment values will be NA, because it takes at least two subsequent surveys to calculate meaningful increments. In case an object of class `fe_ccircle_spatial_notrees` (which is a special child of `fe_stand`) is provided as input `x`, the function returns an empty data frame.

Examples

```
oo <- options(fe_spec_lang = "eng") # Display species names in English

# Mixed mountain forest with several surveys
stand_inc <- stand_sums_dynamic(mm_forest_1_fe_stand_spatial)
stand_inc

# Combine to species overarching increments. Zero in the first year results
# as there cannot be increments available at the first survey
stand_inc |>
  dplyr::group_by(time_yr) |>
    dplyr::summarise(
      iba_m2_ha_yr = sum(iba_m2_ha_yr, na.rm = TRUE),
      iv_m3_ha_yr  = sum(iv_m3_ha_yr,  na.rm = TRUE)
    )

# When there is only one single survey, all increments must be NA
stand_sums_dynamic(spruce_beech_1_fe_stand)

options(oo) # Set options to previous values
```

---

stand_sums_static *Static Stand Sum and Mean Values for an fe_stand Object*

Description

Calculate ha-wise static stand sum and mean values for a fe_stand object. The term 'static' means that no growth and increment variables are calculated, only descriptive variables for each point in time.

Usage

```
stand_sums_static(x, tree_filter = TRUE, hd_dom_method = c("Weise", "Assmann"))
```

Arguments

x	An fe_stand object
tree_filter	A data-masking expression that applies to the data.frame x$trees. It must return a logical value, and is defined in terms of the variables in x$trees. In this function, it is used internally in order to define the cohort of trees which is to be evaluated by this function (within a call to dplyr::filter()). While many meaningful filterings are conceivable, distinctions between total stand, removal stand, and remaining stand are the most probable applications. Defaults to TRUE, i.e. all trees are included. See examples.
hd_dom_method	Method for calculationg the dominant diameter and dominant height. The default choice is "Weise", using the functions d_dom_weise, and h_dom_weise. Alternatively, the option "Assmann" uses the functions d_100, and h_100.

Details

Default setting for the dominant heights and diameters is the method by Weise, i.e. quadratic mean diameter and height for the 20% biggest trees. Alternatively, the d100, h100 method by Assmann can be selected. This should be, however, done with care, because d100 and h100 are only well defined in monospecific stands. Note, that this function does not take into account species shares in mixed stands when calculating d100 and h100. If the tree heights in the input object contain missing values, the output variables requiring height information will be NA.

Value

A data frame (tibble) with the ha-related static sum values stem number, basal area, volume, quadratic mean diameter, dominant diameter, quadratic mean height, dominant height. If no tree in x$trees passes tree_filter, as defined above, an empty data frame is returned. In case an object of class fe_ccircle_spatial_notrees (which is a special child of fe_stand) is provided as input x, the function also returns an empty data frame. In case the tree heights in the input object contain missing values, the output variables requiring height information will be NA.

Examples

```
# Evaluation for all trees
mm_forest_1_fe_stand_spatial |> stand_sums_static()

# Exclude removal trees
mm_forest_1_fe_stand_spatial |> stand_sums_static(!removal)

# Exclude removal trees and include only trees with dbh > 30 cm
mm_forest_1_fe_stand_spatial |> stand_sums_static(!removal & dbh_cm > 30)

# Exclude removal trees, use Assmann's d100 h100 for dominant height
# and diameter
mm_forest_1_fe_stand_spatial |>
  stand_sums_static(!removal, hd_dom_method = "Assmann")

# Include all trees, use Assmann's d100 h100 for dominant height
# and diameter
```

```
mm_forest_1_fe_stand_spatial |>
  stand_sums_static(hd_dom_method = "Assmann")

# Incomplete height information leads to missing values in all variables
# that require height as an input
demo_stand <- spruce_beech_1_fe_stand     # Copy an existing fe_stand object
index <- seq(2:nrow(demo_stand$trees))    # remove every 2nd height
demo_stand$trees[index, ]$height_m <- NA
demo_stand |> stand_sums_static()
```

`stocking_level` *Calculate the Stocking Level ("Bestockungsgrad") of a Stand*

Description

The stocking level (German "Bestockungsgrad") is an important measure for stand density in practice. It is the ratio of a stand's actual basal area and its expected basal area due to a yield table.

Usage

```
stocking_level(ba, age, si, ytable)
```

Arguments

`ba`	The stand's basal area in m²/ha
`age`	The stand's age in years
`si`	The stand's site index according to the yield table `yt`
`ytable`	The yield table to be used as reference. Must be an object of class `fe_yield_table`

Value

The stocking level of the stand based on the yield table of interest

See Also

Other yield table functions: `fe_yield_table()`, `plot.fe_yield_table()`, `si_to_mai_age()`, `si_to_mai_max()`, `site_index()`, `yield_tables_for_species`, `ytable_age_slice()`, `ytable_lookup()`, `ytable_max_slice()`

Examples

```
# Scots pine stand, 72 years old, site index 1.2, basal area 41.3 m²/ha

# 1. Reference: Yield table for pine by Wiedemann
stocking_level(
  ba = 41.3, age = 72, si = 1.2,
  ytable = fe_ytable_pine_wiedemann_moderate_1943
```

```
)

# 2. Reference: Yield table for pine by Wiedemann
stocking_level(
  ba = 41.3, age = 72, si = 1.2,
  ytable = fe_ytable_pine_gehrhardt_moderate_1921
)

# Norway spruce stand, 72 years old, site index 38, basal area 41.3 m²/ha
# 1. Reference Yield Table by Assmann-Franz
stocking_level(
  ba = 41.3, age = 72, si = 38,
  ytable = fe_ytable_spruce_assmann_franz_mean_yield_level_1963
)

# 2. Reference Yield Table for spruce by Wiedemann, moderate thinning,
# site index 1.0
stocking_level(
  ba = 41.3, age = 72, si = 1.0,
  ytable = fe_ytable_spruce_wiedemann_moderate_1936_42
)
```

`summary.fe_ccircle_spatial_notrees` *summary For a fe_ccircle_spatial_notrees object*

Description

For the time being, this is function serves only to obtain a controled output when `summary` is called for an object of class `fe_ccircle_spatial_notrees`. This output is just an empty data data frame.

Usage

```
## S3 method for class 'fe_ccircle_spatial_notrees'
summary(object, ...)
```

Arguments

`object`	An `fe_ccircle_spatial_notrees` object
`...`	Additional arguments, not used in `plot.fe_ccircle_spatial_notrees`

Value

An empty data frame

Examples

```
# Abuse input data, that would actually allow for a full fe_ccircle_spatial
# object to construct a fe_ccircle_spatial_notrees object.
x <- spruce_pine_ccircle_raw
x <- x |> fe_ccircle_spatial_notrees()

# Make the dummy summary
summary(x)
```

`summary.fe_species_bavrn_state`
Summary of an **fe_species_bavrn_state** *Vector*

Description

Produces a summary for a fe_species_bavrn_state object in the same style as R does for factors. Actually, after some conversions `summary.factor` *is* called by this function. The species naming in the summary depends on the parameter `spec_lang`.

Usage

```
## S3 method for class 'fe_species_bavrn_state'
summary(
  object,
  spec_lang = options("fe_spec_lang")$fe_spec_lang,
  maxsum = 100L,
  ...
)
```

Arguments

`object`	Object of class `fe_species_bavrn_state`
`spec_lang`	Choice of how species (group) names or id's are displayed in the summary. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.
`maxsum`	Same as parameter `maxsum` in `summary.factor`
`...`	Other parameters (not used)

Value

A named vector in the same style as returned by `summary.factor`

Examples

```
# Construct some species id vector
spec_ids <- c(
  rep(fe_species_bavrn_state(c("10", "20", "21", "89")),
    times = c(15, 31, 70, 12)
  ),
  NA, NA
)

summary(spec_ids)
spec_ids |> summary()
spec_ids |> summary(spec_lang = "eng")

# Usual application: Set option for species code output
# Any summary of an fe_species object will use the last setting of the
# option
options(fe_spec_lang = "sci")
spec_ids |> summary()
```

`summary.fe_species_bavrn_state_short`
Summary of an **fe_species_bavrn_state_short** *Vector*

Description

Produces a summary for a fe_species_bavrn_state_short object in the same style as R does for factors. Actually, after some conversions `summary.factor` *is* called by this function. The species naming in the summary depends on the parameter `spec_lang`.

Usage

```
## S3 method for class 'fe_species_bavrn_state_short'
summary(
  object,
  spec_lang = options("fe_spec_lang")$fe_spec_lang,
  maxsum = 100L,
  ...
)
```

Arguments

`object`	Object of class `fe_species_bavrn_state_short`
`spec_lang`	Choice of how species (group) names or id's are displayed in the summary. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.

maxsum	Same as parameter maxsum in summary.factor
...	Other parameters (not used)

Value

A named vector in the same style as returned by summary.factor

Examples

```
# Construct some species id vector
spec_ids <- c(
  rep(fe_species_bavrn_state_short(c("1", "2", "6", "9")),
    times = c(15, 31, 70, 12)
  ),
  NA, NA
)

summary(spec_ids)
spec_ids |> summary()
spec_ids |> summary(spec_lang = "eng")

# Usual application: Set option for species code output
# Any summary of an fe_species object will use the last setting of the
# option
options(fe_spec_lang = "sci")
spec_ids |> summary()
```

---

summary.fe_species_ger_nfi_2012

Summary of an **fe_species_ger_nfi_2012** *Vector*

Description

Produces a summary for a fe_species_ger_nfi_2012 object in the same style as R does for factors. Actually, after some conversions summary.factor *is* called by this function. The species naming in the summary depends on the parameter spec_lang.

Usage

```
## S3 method for class 'fe_species_ger_nfi_2012'
summary(
  object,
  spec_lang = options("fe_spec_lang")$fe_spec_lang,
  maxsum = 100L,
  ...
)
```

Arguments

object	Object of class fe_species_ger_nfi_2012
spec_lang	Choice of how species (group) names or id's are displayed in the summary. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute species_coding. The default is to request the choice with options("fe_spec_lang"). If this option is not set, the choice "code" is used.
maxsum	Same as parameter maxsum in summary.factor
...	Other parameters (not used)

Value

A named vector in the same style as returned by summary.factor

Examples

```
# Construct some species id vector
spec_ids <- c(
  rep(fe_species_ger_nfi_2012(c("170", "140", "120", "150")),
    times = c(151, 231, 70, 122)
  ),
  NA, NA, NA, NA
)

summary(spec_ids)
spec_ids |> summary()
spec_ids |> summary(spec_lang = "eng")

# Usual application: Set option for species code output
# Any summary of an fe_species object will use the last setting of the
# option
options(fe_spec_lang = "sci")
spec_ids |> summary()
```

---

summary.fe_species_master

Summary of an **fe_species_master** *Vector*

Description

Produces a summary for a fe_species_master object in the same style as R does for factors. Actually, after some conversions summary.factor *is* called by this function. The species naming in the summary depends on the parameter spec_lang.

Usage

```
## S3 method for class 'fe_species_master'
summary(
  object,
  spec_lang = options("fe_spec_lang")$fe_spec_lang,
  maxsum = 100L,
  ...
)
```

Arguments

object	Object of class fe_species_master
spec_lang	Choice of how species (group) names or id's are displayed in the summary. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute species_coding. The default is to request the choice with options("fe_spec_lang"). If this option is not set, the choice "code" is used.
maxsum	Same as parameter maxsum in summary.factor
...	Other parameters (not used)

Value

A named vector in the same style as returned by summary.factor

Examples

```
# Construct some species id vector
spec_ids <- c(
  rep(
    fe_species_master(c(
      "pinus_001", "quercus_003", "tilia_002", "carpinus_001", "sorbus_002"
    )),
    times = c(12, 7, 24, 16, 32)
  ),
  NA, NA, NA, NA
)

summary(spec_ids)
spec_ids |> summary()
spec_ids |> summary(spec_lang = "eng")

# Usual application: Set option for species code output
# Any summary of an fe_species object will use the last setting of the
# option
options(fe_spec_lang = "sci")
spec_ids |> summary()
```

summary.fe_species_tum_wwk_long

Summary of an **fe_species_tum_wwk_long** *Vector*

Description

Produces a summary for a fe_species_tum_wwk_long object in the same style as R does for factors. Actually, after some conversions `summary.factor` *is* called by this function. The species naming in the summary depends on the parameter `spec_lang`.

Usage

```
## S3 method for class 'fe_species_tum_wwk_long'
summary(
  object,
  spec_lang = options("fe_spec_lang")$fe_spec_lang,
  maxsum = 100L,
  ...
)
```

Arguments

`object`	Object of class `fe_species_tum_wwk_long`
`spec_lang`	Choice of how species (group) names or id's are displayed in the summary. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.
`maxsum`	Same as parameter `maxsum` in `summary.factor`
`...`	Other parameters (not used)

Value

A named vector in the same style as returned by `summary.factor`

Examples

```
# Construct some species id vector
spec_ids <- c(
  rep(fe_species_tum_wwk_long(c("10", "60", "87", "811")), each = 15),
  NA, NA
)

summary(spec_ids)
spec_ids |> summary()
spec_ids |> summary(spec_lang = "eng")
```

```
# Usual application: Set option for species code output
# Any summary of an fe_species object will use the last setting of the
# option
options(fe_spec_lang = "sci")
spec_ids |> summary()
```

`summary.fe_species_tum_wwk_short`
Summary of an **fe_species_tum_wwk_short** *Vector*

Description

Produces a summary for a fe_species_tum_wwk_short object in the same style as R does for factors. Actually, after some conversions `summary.factor` *is* called by this function. The species naming in the summary depends on the parameter `spec_lang`.

Usage

```
## S3 method for class 'fe_species_tum_wwk_short'
summary(
  object,
  spec_lang = options("fe_spec_lang")$fe_spec_lang,
  maxsum = 100L,
  ...
)
```

Arguments

`object`	Object of class `fe_species_tum_wwk_short`
`spec_lang`	Choice of how species (group) names or id's are displayed in the summary. Supported choices are "code" (displays the species codes as they are), "eng" (English species names), "ger" (German species names), and "sci" (scientific species names). The names and the codes refer to the species coding given in the object's attribute `species_coding`. The default is to request the choice with `options("fe_spec_lang")`. If this option is not set, the choice "code" is used.
`maxsum`	Same as parameter `maxsum` in `summary.factor`
`...`	Other parameters (not used)

Value

A named vector in the same style as returned by `summary.factor`

Examples

```
# Construct some species id vector
spec_ids <- c(
  rep(fe_species_tum_wwk_short(as.character(1:10)), each = 5), NA
)

summary(spec_ids)
spec_ids |> summary()
spec_ids |> summary(spec_lang = "eng")

# Usual application: Set option for species code output
# Any summary of an fe_species object will use the last setting of the
# option
options(fe_spec_lang = "sci")
spec_ids |> summary()

# The summary is also used in the summary of a data frame which contains
# an fe_species object, but displayed differently
options(fe_spec_lang = "eng")
selection_forest_1_fe_stand$trees |> summary()
```

`summary.fe_stand` *Summary for an* **fe_stand** *Object*

Description

Compute an overview of basic stand level variables for an `fe_stand` object

Usage

```
## S3 method for class 'fe_stand'
summary(object, ...)
```

Arguments

`object`	an `fe_stand` object
`...`	additional arguments, passed to `stand_sums_static`

Details

The summary calls the function `stand_sums_static` with its default settings. The result is a data.frame (tibble) with species and year wise stand sum values per ha and mean values. Stem numbers, basal areas, quadratic mean diameters and dominant diameters (d 100) are always calculated. Quadratic mean heights, dominant heights (h 100), and wood volumes are only calculated if the heights of all trees are given in `object` (i.e. no NA). The summary contains a column `species_id`. Depending on the setting of `options("fe_spec_lang")`, the species ids will be printed as the species code (settings NULL or "ger"), scientific, English, or German species names (settings "sci", "eng", or "ger", respectively).

Value

data.frame (tibble) resulting from applying the function stand_sums_static to object

See Also

stand_sums_static

Examples

```
# Make a stand data.frame (or nicer, a tibble) that meets the minimum
# requirements for setting up a fe_Stand object
some_stand <- tibble::tibble(
  tree_id = as.character(c(1:100)),
  species_id = as_fe_species_tum_wwk_short(
    as.character(c(rep(1, 40), rep(5, 60)))
  ),
  time_yr = rep(2022, 100),
  dbh_cm = c(rnorm(40, 40.1, 7.3), rnorm(60, 32.8, 8.4)),
)

# Make the object
some_fe_stand <- fe_stand(
  some_stand,
  tree_id_col = "tree_id",
  species_id_col = "species_id",
  time_yr_col = "time_yr",
  dbh_cm_col = "dbh_cm",
  area_ha = 0.25
)

# The summary with different language choices
options(fe_spec_lang = "code")
summary(some_fe_stand)
options(fe_spec_lang = "sci")
summary(some_fe_stand)
options(fe_spec_lang = "eng")
summary(some_fe_stand)
options(fe_spec_lang = "ger")
summary(some_fe_stand)


# Use example stands
options(fe_spec_lang = "eng")
norway_spruce_1_fe_stand |> summary()
summary(european_beech_1_fe_stand)
options(fe_spec_lang = "sci")
summary(selection_forest_1_fe_stand)
spruce_beech_1_fe_stand |> summary()
options(fe_spec_lang = "code")
summary(selection_forest_1_fe_stand)
spruce_beech_1_fe_stand |> summary()
```

`survey_overview` *Generate an Overview of the Surveys of an fe_stand Object*

Description

The tree data frame of an `fe_stand` object is evaluated in order to get survey and species specific meta information about these data.

Usage

```
survey_overview(x, tree_filter = TRUE)
```

Arguments

`x`	An object of class `fe_stand`
`tree_filter`	A `data-masking` expression that applies to the data.frame `x$trees`. It must return a logical value, and is defined in terms of the variables in `x$trees`. In this function, it is used internally in order to define the cohort of trees which is to be evaluated by this function (within a call to `dplyr::filter()`). For this function, `tree_filter` should almost *never* be something else than `TRUE` (default)

Details

This function provides meta information that is useful for evaluations that can be complex on the detail level, e.g. increment calculations from subsequent surveys. In such contexts, the column `n_species_occurence` of the output data frame can be of special interest. If a species is present for a number of consecutive surveys, all these surveys get the same integer number in this column. If the same species vanishes, but occurs again later, the next block of surveys gets the subsequent number, and so on. So, all consecutive blocks of a species' occurrence are numbered as 1, 2, 3, etc. At surveys where the species is not present `n_species_occurrence` has the value 1.

Value

A data frame (tibble) that gives an overview of the surveys represented in the input object `x`. It is basically an evaluation of the data frame `x$trees`. It provides information about how many trees were present in each survey, how many dbh and heights were measured, and if the dbh and height measurements cover all trees, For dbh this must be always true, because this is a requirement for a valid `fe_stand` object. In addition, we are informed whether a species that has been documented in the object is represented in a specific survey or not. As a basis for advanced evaluations, species occurences are numbered in the columen `n_species_occurrence` (see Details). In case an object of class `fe_ccircle_spatial_notrees` (which is a special child of `fe_stand`) is provided as input `x`, the function returns an empty data frame.

Examples

```
# Example data: Mixed mountain forest plot with several surveys
mm_forest_1_fe_stand_spatial |> survey_overview()
```

`validate_fe_ccircle_spatial`
Validate an **fe_ccircle_spatial** *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor `new_fe_ccircle_spatial`. Regular users, please construct `fe_ccircle_spatial` objects with `fe_ccircle_spatial`.

Usage

```
validate_fe_ccircle_spatial(x, method = c("strict", "flexible"))
```

Arguments

`x`	An object that is expected to be a correct `fe_ccircle_spatial` object
`method`	Character string that specifies whether tree_positions is allowed to contain less tree_ids than trees (i.e. in this case, not all trees have coordinates). Possible choices are "strict" (default) and "flexible". If method == "flexible", a warning is issued if not all trees have coordinates. If method == "strict", the validation terminates with an error.

Value

Returns `x`, but this function is mainly called for its side effect which is pointing out any violations of the `fe_ccircle_spatial` object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Validate the example fe_ccircle_spatial object
spruce_pine_ccircle_spatial |>
  validate_fe_ccircle_spatial(method = "flexible")
```

---

validate_fe_ccircle_spatial_notrees
Validate an **fe_ccircle_spatial_notrees** *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor new_fe_ccircle_spatial_notrees. Regular users, please construct fe_ccircle_spatial_notrees objects with fe_ccircle_spatial_notrees.

Usage

```
validate_fe_ccircle_spatial_notrees(x)
```

Arguments

x	An object that is expected to be a correct fe_ccircle_spatial_notrees object

Value

Returns x, but this function is mainly called for its side effect which is pointing out any violations of the fe_ccircle_spatial_notrees object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Validate the example fe_ccircle_spatial object
spruce_pine_ccircle_spatial_notrees |>
  validate_fe_ccircle_spatial_notrees()
```

---

validate_fe_species_bavrn_state
Validate an fe_species_bavrn_state *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor fe_species_bavrn_state. Regular users, please construct fe_species_bavrn_state objects with fe_species_bavrn_state.

Usage

```
validate_fe_species_bavrn_state(x = character())
```

Arguments

x	An object that is expected to be a correct fe_species_bavrn_state object

Value

Returns x, but this function is mainly called for its side effect which is pointing out any violations of the fe_species_bavrn_state object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Passes validation
spec_ids <- as.character(c(30, 30, 30, 10, 10, 10, 10, 20, 20, 80))
spec_ids <- new_fe_species_bavrn_state(spec_ids)
validate_fe_species_bavrn_state(spec_ids)

# Validating the following spec_ids throws an error due to
# non-supported species codes
spec_ids <- as.character(c(30, 30, 8712, 10, 10, 10, 349, 20, 20, 80))
spec_ids <- new_fe_species_bavrn_state(spec_ids)
try(
  validate_fe_species_bavrn_state(spec_ids)
)
```

---

validate_fe_species_bavrn_state_short

Validate an fe_species_bavrn_state_short *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor fe_species_bavrn_state_short. Regular users, please construct fe_species_bavrn_state_short objects with fe_species_bavrn_state_short.

Usage

```
validate_fe_species_bavrn_state_short(x = character())
```

Arguments

x	An object that is expected to be a valid fe_species_bavrn_state_short object

Value

Returns x, but this function is mainly called for its side effect which is pointing out any violations of the fe_species_bavrn_state_short object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Passes validation
spec_ids <- as.character(c(3, 3, 3, 1, 1, 1, 1, 2, 2, 8))
spec_ids <- new_fe_species_bavrn_state_short(spec_ids)
validate_fe_species_bavrn_state_short(spec_ids)

# Validating the following spec_ids throws an error due to
# non-supported species codes
spec_ids <- as.character(c(3, 3, 8712, 1, 1, 1, 349, 2, 2, 8))
spec_ids <- new_fe_species_bavrn_state_short(spec_ids)
try(
  validate_fe_species_bavrn_state_short(spec_ids)
)
```

---

validate_fe_species_ger_nfi_2012

Validate an fe_species_ger_nfi_2012 *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor new_fe_species_ger_nfi_2012. Regular users, please construct fe_species_ger_nfi_2012 objects with fe_species_ger_nfi_2012.

Usage

```
validate_fe_species_ger_nfi_2012(x = character())
```

Arguments

x	An object that is expected to be a correct fe_species_ger_nfi_2012 object

Value

Returns x, but this function is mainly called for its side effect which is pointing out any violations of the fe_species_ger_nfi_2012 object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Passes validation
spec_ids <- as.character(c(30, 30, 30, 10, 10, 10, 10, 20, 20, 290))
spec_ids <- new_fe_species_ger_nfi_2012(spec_ids)
validate_fe_species_ger_nfi_2012(spec_ids)

# Validating the following spec_ids throws an error due to
# non-supported species codes
spec_ids <- as.character(c(30, 30, 542, 10, 10, 10, 1234, 20, 20, 290))
```

```
spec_ids <- new_fe_species_ger_nfi_2012(spec_ids)
try(
  validate_fe_species_ger_nfi_2012(spec_ids)
)
```

`validate_fe_species_master` *Validate an* fe_species_master *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor `new_fe_species_master`. Regular users, please construct `fe_species_master` objects with `fe_species_master`.

Usage

```
validate_fe_species_master(x = character())
```

Arguments

`x`	An object that is expected to be a correct `fe_species_master` object

Value

Returns `x`, but this function is mainly called for its side effect which is pointing out any violations of the `fe_species_master` object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Passes validation
spec_ids <- c("pinus_001", "quercus_002", "pinus_001", "fagus_001")
spec_ids <- new_fe_species_master(spec_ids)
validate_fe_species_master(spec_ids)

# Validating the following spec_ids throws an error due to
# non-supported species codes
spec_ids <- c("pinus_001", "my_awesome_species_003", "wonder_tree_3012")
spec_ids <- new_fe_species_master(spec_ids)
try(
  validate_fe_species_master(spec_ids)
)
```

validate_fe_species_tum_wwk_long
Validate an fe_species_tum_wwk_long *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor new_fe_species_tum_wwk_long. Regular users, please construct fe_species_tum_wwk_long objects with fe_species_tum_wwk_long.

Usage

```
validate_fe_species_tum_wwk_long(x = character())
```

Arguments

x	An object that is expected to be a correct fe_species_tum_wwk_long object

Value

Returns x, but this function is mainly called for its side effect which is pointing out any violations of the fe_species_tum_wwk_long object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Passes validation
spec_ids <- as.character(c(70, 61, 88, 88, 10, 971, 32))
spec_ids <- new_fe_species_tum_wwk_long(spec_ids)
validate_fe_species_tum_wwk_long(spec_ids)

# Validating the following spec_ids throws an error due to
# non-supported species codes
spec_ids <- as.character(c(70, 61, 1221, 88, 88, 10, 971, 32, 4031))
spec_ids <- new_fe_species_tum_wwk_long(spec_ids)
try(
  validate_fe_species_tum_wwk_long(spec_ids)
)
```

validate_fe_species_tum_wwk_short
Validate an fe_species_tum_wwk_short *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor `new_fe_species_tum_wwk_short`. Regular users, please construct `fe_species_tum_wwk_short` objects with `fe_species_tum_wwk_short`.

Usage

```
validate_fe_species_tum_wwk_short(x = character())
```

Arguments

x	An object that is expected to be a correct `fe_species_tum_wwk_short` object

Value

Returns `x`, but this function is mainly called for its side effect which is pointing out any violations of the `fe_species_tum_wwk_short` object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Passes validation
spec_ids <- as.character(c(2, 2, 2, 1, 1, 1, 1, 3, 3, 9))
spec_ids <- new_fe_species_tum_wwk_short(spec_ids)
validate_fe_species_tum_wwk_short(spec_ids)

# Validating the following spec_ids throws an error due to
# non-supported species codes
spec_ids <- as.character(c(2, 2, 52, 1, 1, 1, 123, 3, 3, 9))
spec_ids <- new_fe_species_tum_wwk_short(spec_ids)
try(
  validate_fe_species_tum_wwk_short(spec_ids)
)
```

validate_fe_stand *Validate an* **fe_stand** *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor new_fe_stand. Regular users, please construct fe_stand objects with fe_stand.

Usage

```
validate_fe_stand(x)
```

Arguments

x	An object that is expected to be a correct fe_stand object

Value

Returns x, but this function is mainly called for its side effect which is pointing out any violations of the fe_stand object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Validate the example stands
validate_fe_stand(norway_spruce_1_fe_stand)
validate_fe_stand(european_beech_1_fe_stand)
validate_fe_stand(spruce_beech_1_fe_stand)
selection_forest_1_fe_stand |> validate_fe_stand()
```

validate_fe_stand_spatial *Validate an* **fe_stand_spatial** *Object*

Description

Regular users will not require this function. Expert users will want to use it in combination with the constructor new_fe_stand_spatial. Regular users, please construct fe_stand_spatial objects with fe_stand_spatial.

Usage

```
validate_fe_stand_spatial(x)
```

Arguments

x	An object that is expected to be a correct `fe_stand_spatial` object

Value

Returns `x`, but this function is mainly called for its side effect which is pointing out any violations of the `fe_stand_spatial` object specifications. In case of such violations, the function will terminate with an error.

Examples

```
# Validate the example fe_stand_spatial object
mm_forest_1_fe_stand_spatial |> validate_fe_stand_spatial()
```

`validate_fe_yield_table` *Validate a Candidate For an fe_yield_table Object*

Description

Validate a Candidate For an fe_yield_table Object

Usage

```
validate_fe_yield_table(x)
```

Arguments

x	The candidate object to be validated

Value

If `x` is not a valid `fe_yield_table` object, the function will terminate with an error. Otherwise, `x` will be returned.

Examples

```
validate_fe_yield_table(fe_ytable_beech_wiedemann_moderate_1931)
validate_fe_yield_table(fe_ytable_larch_schober_moderate_1946)
validate_fe_yield_table(fe_ytable_pine_wiedemann_moderate_1943)
validate_fe_yield_table(fe_ytable_spruce_gehrhardt_moderate_1921)
validate_fe_yield_table(fe_ytable_douglas_schober_moderate_1956)
validate_fe_yield_table(fe_ytable_spruce_wiedemann_moderate_1936_42)
validate_fe_yield_table(fe_ytable_silver_fir_hausser_moderate_1956)
```

vec_ptype_abbr.fe_species_bavrn_state

Abbreviation for the fe_species_bavrn_state *Type*

Description

Provide an abbreviated name for the class `fe_species_bavrn_state` to be displayed in tibbles and `str()`

Usage

```
## S3 method for class 'fe_species_bavrn_state'
vec_ptype_abbr(x, ...)
```

Arguments

`x`	An object of type `fe_species_bavrn_state`
`...`	Other parameters (not used)

Value

The abbreviation to be displayed for the species coding (`character`) in tibbles and in `str()`

Examples

```
spec_ids <- fe_species_bavrn_state(as.character(c(10, 30, 60)))
vctrs::vec_ptype_abbr(spec_ids)
str(spec_ids)
```

vec_ptype_abbr.fe_species_bavrn_state_short

Abbreviation for the fe_species_bavrn_state_short *Type*

Description

Provide an abbreviated name for the class `fe_species_bavrn_state_short` to be displayed in tibbles and `str()`

Usage

```
## S3 method for class 'fe_species_bavrn_state_short'
vec_ptype_abbr(x, ...)
```

Arguments

x	An object of type fe_species_bavrn_state_short
...	Other parameters (not used)

Value

The abbreviation to be displayed for the species coding (character) in tibbles and in str()

Examples

```
spec_ids <- fe_species_bavrn_state_short(as.character(c(2, 4, 6)))
vctrs::vec_ptype_abbr(spec_ids)
str(spec_ids)
```

---

vec_ptype_abbr.fe_species_ger_nfi_2012

Abbreviation for the fe_species_ger_nfi_2012 *Type*

Description

Provide an abbreviated name for the class fe_species_ger_nfi_2012 to be displayed in tibbles and str()

Usage

```
## S3 method for class 'fe_species_ger_nfi_2012'
vec_ptype_abbr(x, ...)
```

Arguments

x	An object of type fe_species_ger_nfi_2012
...	Other parameters (not used)

Value

The abbreviation to be displayed for the species coding (character) in tibbles and in str()

Examples

```
spec_ids <- fe_species_ger_nfi_2012(as.character(c(10, 20, 50)))
vctrs::vec_ptype_abbr(spec_ids)
str(spec_ids)
```

vec_ptype_abbr.fe_species_master

Abbreviation for the fe_species_master *Type*

Description

Provide an abbreviated name for the class `fe_species_master` to be displayed in tibbles and `str()`

Usage

```
## S3 method for class 'fe_species_master'
vec_ptype_abbr(x, ...)
```

Arguments

`x`	An object of type `fe_species_master`
`...`	Other parameters (not used)

Value

The abbreviation to be displayed for the species coding (`character`) in tibbles and in `str()`

Examples

```
spec_ids <- fe_species_master(c("pinus_001", "quercus_002", "pinus_001"))
vctrs::vec_ptype_abbr(spec_ids)
str(spec_ids)
```

vec_ptype_abbr.fe_species_tum_wwk_long

Abbreviation for the fe_species_tum_wwk_long *Type*

Description

Provide an abbreviated name for the class `fe_species_tum_wwk_long` to be displayed in tibbles and `str()`

Usage

```
## S3 method for class 'fe_species_tum_wwk_long'
vec_ptype_abbr(x, ...)
```

Arguments

`x`	An object of type `fe_species_tum_wwk_long`
`...`	Other parameters (not used)

Value

The abbreviation to be displayed for the species coding (`character`) in tibbles and in `str()`

Examples

```
spec_ids <- fe_species_tum_wwk_long(as.character(c(10, 50, 87, 813)))
vctrs::vec_ptype_abbr(spec_ids)
str(spec_ids)
```

---

vec_ptype_abbr.fe_species_tum_wwk_short

Abbreviation for the fe_species_tum_wwk_short *Type*

Description

Provide an abbreviated name for the class `fe_species_tum_wwk_short` to be displayed in tibbles and `str()`

Usage

```
## S3 method for class 'fe_species_tum_wwk_short'
vec_ptype_abbr(x, ...)
```

Arguments

`x`	An object of type `fe_species_tum_wwk_short`
`...`	Other parameters (not used)

Value

The abbreviation to be displayed for the species coding (`character`) in tibbles and in `str()`

Examples

```
spec_ids <- fe_species_tum_wwk_short(as.character(c(1, 3, 4)))
vctrs::vec_ptype_abbr(spec_ids)
str(spec_ids)
```

v_gri *Calculate Tree Volumes With the GRI Volume Equations (Franz et al. 1973)*

Description

Merchantable standing tree volumes over bark calculated with the GRI volume equations developed by Friedrich Franz in 1971. These volume equations are standard in the German Federal State of Bavaria

Usage

```
v_gri(species_id, dbh_cm, height_m)
```

Arguments

species_id	Vector of species id's following the *tum_wwk_short* species coding. Ideally, these species_id's are provided as an fe_species_tum_wwk_short or an fe_species_bavrn_state_sho object. If they are provided as another fe_species object, v_gri will make an attempt to convert them into fe_species_tum_wwk_short. The exception is the coding fe_species_bavrn_state which will be converted into fe_species_bavrn_state_short. If all conversion attempts fail, the function will terminate with an error. The species id's can also be provided as numeric values (double or integer) or character. These will be internally converted to fe_species_tum_wwk_short. If this fails (i.e. the user provided species codes are not defined in the *tum_wwk_short* coding), an error is thrown and the function terminates.
dbh_cm	Vector of tree dbh values in cm (dbh = stem diameter at breast height, i.e. 1.3 m)
height_m	Vector of tree height values in m. While missing values in species_id and dbh_cm are not allowed, they are accepted in height_m but v_gri will return NA and trigger a warning.

Details

The abbreviation *GRI* stands for the German word "Großrauminventur" (large area inventory). This forest inventory was conducted in 1971 by Friedrich Franz and his team of the Chair for Forest Growth and Yield Science at the Munich Ludwig-Maximilians-University (Franz et al. 1973). The inventory covered the whole federal state of Bavaria (~ 70,600 km²). The volume equations implemented in this function were calibrated with the data of several ten thousands of trees which were felled for that purpose during the inventory. The volume equations are available for exactly the species (groups) defined in the coding *tum_wwk_short*. If they are called with another species coding supported by the package **ForestElementsR**, v_gri will attempt to convert them accordingly.

Value

A vector of merchantable standing tree wood volumes over bark in m³. "Merchantable" means only wood with a minimum diameter of 7 cm over bark is taken into account. Therefore, for small trees without any merchandable wood, the function will return 0 m³.

References

Franz F, Bachler B, Deckelmann E, Kennel R, Schmidt A, Wotschikowski U (1973). *Bayerische Waldinventur 1970/71, Inventurabschnitt I: Großrauminventur Aufnahme- und Auswertungsverfahren*, volume 11 of *Forstliche Forschungsberichte München*. Forstwissenschaftliche Fakultät der Universität München und Bayerische Forstliche Versuchs- und Forschungsanstalt.

Examples

```
# Find out the species codes that work with v_gri
fe_species_get_coding_table("tum_wwk_short") |>
  dplyr::select(-genus, -species_no) |>
  dplyr::distinct()

# Merchantable volume of a European beech with dbh = 30 cm,
# and height = 29 m
v_gri("5", 30, 29)
v_gri(5, 30, 29)
v_gri(as_fe_species_tum_wwk_short(5), 30, 29)

# Several trees (three species, three sizes)
species_id <- fe_species_tum_wwk_short(c(1, 1, 1, 3, 3, 3, 5, 5, 5))
dbh_cm <- c(12, 30, 55, 12, 30, 55, 12, 30, 55)
height_m <- c(14, 33, 39, 14, 33, 39, 14, 33, 39)
v_gri(species_id, dbh_cm, height_m)

# The same, but the species id's are now originally defined in the
# coding of the 2012 German national forest inventory
species_id <- fe_species_ger_nfi_2012(
  c(10, 10, 10, 20, 20, 20, 100, 100, 100)
)
v_gri(species_id, dbh_cm, height_m)
```

---

v_red_harvest_ubark *Reduce a Given standing Volume Over Bark to Harvested Volume Under Bark*

Description

Many tree volume functions (like `v_gri`) calculate wood volumes defined as standing and over bark. Practictioners often prefer to work with volumes where the harvest losses and the bark volume have been substracted. Given an over bark standing volume, this function uses species specific reduction factors in order to obtain harvested volume under bark. The reduction factors are taken from (BayMinELF 1990); they relate to the species coding `fe_species_tum_wwk_short` or, alternatively `fe_species_bavrn_state_short`.

Usage

```
v_red_harvest_ubark(species_id, v_orig_m3)
```

Arguments

species_id	Vector of species id's. Ideally, these species_id's are provided as a fe_species_tum_wwk_short or a fe_species_bavrn_state_short object. If they are provided as another fe_species object, v_red_harvest_ubark will make an attempt to convert them into fe_species_tum_wwk_short. The exception is the coding fe_species_bavrn_state which will be converted into fe_species_bavrn_state_short. If all conversion attempts fail, the function will terminate with an error. The species id's can also be provided as numeric values (double or integer) or character. These will be internally converted to fe_species_tum_wwk_short. If this fails (i.e. the user provided species codes are not defined in the *tum_wwk_short* coding), an error is thrown and the function terminates.
v_orig_m3	Vector of wood volumes (m³) defined as standing over bark. If species_id and v_orig_m3 do not have the same length, an attempt is made to recyle them according to the tibble rules.

Value

A vector of the reduced volumes defined as harveest

References

BayMinELF (1990). *Hilfstafeln für die Forsteinrichtung. Zusammengestellt für den Gebrauch in der Bayerischen Staatsforstverwaltung*. Bayerisches Staatsministerium für Ernährung Landwirtschaft und Forsten.

Examples

```
# Take all species groups of tum_wwk_short and a standing volume of 1 m³
# over bark
species_id <- fe_species_tum_wwk_short(1:10)
v_red_harvest_ubark(species_id, 1)
```

---

yield_tables *Yield Tables*

Description

The yield table system of **ForestElementsR** allows easy use of yield tables that are available as an fe_yield_table object. Here, we list all yield tables that come with the current version of the package. For test purposes and as an example, the Scots Pine table by Wiedemann (1943) is provided as a raw data frame (prefix "ytable_") and an fe_yield_table object (prefix "fe_ytable_"). The raw data frame can be transformed into a fe_yield_table object with the function fe_yield_table (see the example there). When we refer to *Schober's yield table collection* below, we mean Schober (1975). Many of the yield tables listed were implemented in the version published in what we refer to as the *Hilfstafeln edited by the Bavarian Forest Administration*. Hereby, we mean, more precisely, BayMinELF (1990), and BayMinELF (2018). The yield table collection in both editions is

almost identical, and represented in **ForestElementsR** (including error corrections beyond the 2018 edition). Only the yield table by Kenk and Hradetzky for Douglas fir that is new in the 2018 hasn't been imported here, yet.

The following yield tables are currently implemented (alphabetically ordered by author names):

- *Assmann-Franz 1963, Norway Spruce, Mean Yield Level*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. The total volume production (tvp), mean annual increment (mai), and the periodic annual increment were given as reduced values (under bark, harvest losses substracted) only. Thus, they were converted into standing m³ over bark by dividing them by 0.81 (standard factor for Norway spruce in Bavaria). The yield table allows site indexing based on the dominant height h100, and the quadratic mean height hq, whereby the former is the way Ernst Assmann and Friedrich Franz had in mind. The site index of this table is, unusual for German yield tables, given as a stand's expected dominant height at an age of 100 years. Unlike in most yield table, the number of removal trees (n_rmv_ha), the volume of the removal stand (v_rmv_ha), and the periodic annual increment (pai_m3_ha_yr, red_m3_ha_yr) relate to the subsequent, not the previous time span. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume.
- *Bauer, 1955, Red Oak*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. In the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site indexes I.5 and II.5 were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 21 % which takes into account both, harvest losses (mainly stump) and bark volume.
- *Blume, 1949, Poplar*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. Originally, this table was designed for site indexing by quadratic mean diameter (d_q_cm), not by height. Therefore, this diameter was given in the table as lower threshold, here listed as "d_q_cm_si_plus_025". As site indexing by height is always preferable, options for site indexing by both, diameter and height, were included in the fe_yield_table representation of this table. The actual quadratic mean diameter (d_q_cm) which is used for site indexing if the user decides so was calculated from dividing the given basal area by the stem number, multiplying it with 4/pi, and taking the square root of the result. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. Despite the lower diameter threshold for site indexing, the table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume. Overall, this table seems not really well constructed, the time-curves of its variables are often not intuitively plausible. Some inconsistencies (e.g. decreasing tvp resulting from a to low mean annual increment value) were corrected.

- *Gehrhardt, 1908, European Beech, Moderate Thinning*:Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 15.4 % which takes into account both, harvest losses (mainly stump) and bark volume. The quadratic mean diameter (d_q_cm) for site index 2.0 was obviously wrong in the source data. It was corrected by interpolation.
- *Gehrhardt 1921, Norway Spruce, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume.
- *Gehrhardt, 1921, Scots Pine, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 21 % which takes into account both, harvest losses (mainly stump) and bark volume.
- *Guttenberg, 1915, Norway Spruce, High Mountains*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume. The interpolated site indexes contained many mistakes (highlighted in the source data), mostly in the mai which could, however, be easily corrected by interpolation between the adjacent site index tables.

- *Hausser 1956, Silver Fir, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. In the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site indexes I.5, II.5, etc. were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume.
- *Juettner 1955, Oak, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. In the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site indexes I.5, II.5, etc. were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection.
- *Mitscherlich, 1945, Black Alder*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume.
- *Schober 1946, European Larch, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site indexes I.5, II.5, etc. were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 28 % which takes into account both, harvest losses (mainly stump) and bark volume. There were some mistakes in the mean diameters and basal areas of the interpolated site index tables I.5 (most) and II.5 (less). These have been corrected by interpolating between the original values of the integer site index tables (I.0, II.0, III.0).
- *Schober, 1953, Japanese Larch, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. In the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original

heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site index I.5 were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 28 % which takes into account both, harvest losses (mainly stump) and bark volume.

- *Schober 1956, Douglas Fir, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 21 % which takes into account both, harvest losses (mainly stump) and bark volume.
- *Schwappach, 1903/29, Birch*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. In the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site indexes I.5 and II.5 were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume. An obvious interpoliation error was corrected (site index 1.5, basal area at age = 40 years).
- *Vanselow, 1951, Norway Spruce, Southern Bavaria*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume. Two values, basal area at age 20 and standing volume at age 40 for site index I.5 were obviously wrong, probably due to interpolation errors. Both values were replaced by those obtained from correct interpolation between site indexes I and II.
- *Wiedemann 1931, European Beech, Moderate Thinning*: Imported from the "Hilfstafeln"

edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights, as required for automated use were generated by inter- and extrapolation. The total volume production (tvp) was not contained in the Bavarian edition of the table. Therefore, it was re-calculated by multiplying the table's mean annual increment (mai) with the stand age. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 15.4 % which takes into account both, harvest losses (mainly stump) and bark volume. Site index I.5, age 130: Mistake in n_ha, value 465 was wrong. Linearly interpolated from the neighboring values to 238.5.

- *Wiedemann 1936/42, Norway Spruce, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site indexes I.5, II.5, etc. were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume. Several mistakes were found in the source files, especially consistently wrong stem numbers and mean diameters in site indexes 4 and 5.

- *Wiedemann 1943, Scots Pine, Moderate Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. in the source files, the quadratic mean height of the site index classes was only available as lower threshold (i.e 1/4 site index lower). So the original heights were taken from the version published in Schober's (1975) yield table collection for the whole site indexes (I.0, II.0, etc.). The values for the site indexes I.5, II.5, etc. were generated by linear inter- and extrapolation. The total volume production was not contained in the Bavarian edition of the table; so, it was added from the version published in Schober's (1975) collection. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume means is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 21 % which takes into account both, harvest losses (mainly stump) and bark volume. The Bavarian original values had a mistake in N/ha, site index I.0, age 95: 343 (wrong) instead of 373 (right), correct value was taken from the version in Schober's (1975) yield table collection.

- *Wimmenauer-Schwappach, 1919/29, Ash, Weak Thinning*: Imported from the "Hilfstafeln" edited by the Bavarian Forest Administration. The table follows the standard structure of all tables that are part of the Bavarian "Hilfstafeln": Stem number, mean diameter, standing volume, and basal area describe the remaining stand. Wood volume is defined as standing coarse wood over bark. Variables whose names begin with "red_" relate to harvested volume under bark; i.e. reduced by 19 % which takes into account both, harvest losses (mainly stump) and bark volume.

Usage

```
fe_ytable_spruce_assmann_franz_mean_yield_level_1963

fe_ytable_poplar_blume_1949

fe_ytable_redoak_bauer_1955

fe_ytable_beech_gehrhardt_moderate_1908

fe_ytable_spruce_gehrhardt_moderate_1921

fe_ytable_pine_gehrhardt_moderate_1921

fe_ytable_spruce_guttenberg_1915

fe_ytable_silver_fir_hausser_moderate_1956

fe_ytable_oak_juettner_moderate_1955

fe_ytable_blackalder_mitscherlich_heavy_1945

fe_ytable_larch_schober_moderate_1946

fe_ytable_japanlarch_schober_moderate_1953

fe_ytable_douglas_schober_moderate_1956

fe_ytable_birch_schwappach_1903_29

fe_ytable_beech_wiedemann_moderate_1931

fe_ytable_spruce_wiedemann_moderate_1936_42

fe_ytable_pine_wiedemann_moderate_1943

ytable_pine_wiedemann_moderate_1943_raw

fe_ytable_ash_wimmenauer_1919_29

fe_ytable_spruce_vanselow_1951
```

Format

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 237 rows and 16 columns.

An object of class `fe_yield_table` of length 7.

An object of class `fe_yield_table` of length 7.

References

BayMinELF (1990). *Hilfstafeln für die Forsteinrichtung. Zusammengestellt für den Gebrauch in der Bayerischen Staatsforstverwaltung*. Bayerisches Staatsministerium für Ernährung Landwirtschaft und Forsten.

BayMinELF (2018). *Hilfstafeln für die Forsteinrichtung*. Bayerisches Staatsministerium für Ernährung Landwirtschaft und Forsten.

Schober R (1975). *Ertragstafeln wichtiger Baumarten bei verschiedener Durchforstung*. J. D. Sauerländer's Verlag, Frankfurt a. M.

`yield_tables_for_species`

Yield Tables To Species Assignments

Description

In order to facilitate the application of yield tables, we provide data frames that link the names of implemented `fe_yield_table` objects to species codings. Currently, there are three such data frames: `fe_species_tum_wwk_short`, `fe_species_bavrn_state_short`, `fe_species_bavrn_state` Note, that different yield table assignemnts for the same coding can be defined and coexist. In future, such tables will be added also for less aggregated species codings.

Usage

```
ytables_bavrn_state_short_var_1

ytables_tum_wwk_short_var_1

ytables_bavrn_state_var_1
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 9 rows and 2 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 9 rows and 2 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 45 rows and 2 columns.

See Also

Other yield table functions: `fe_yield_table()`, `plot.fe_yield_table()`, `si_to_mai_age()`, `si_to_mai_max()`, `site_index()`, `stocking_level()`, `ytable_age_slice()`, `ytable_lookup()`, `ytable_max_slice()`

---

ytable_age_slice *Take an Age Slice out of an fe_yield_table_object*

Description

Age slices out of yield tables are typically required for finding out the site index for a given age-height pair, or for extracting a yield table value when age and site index are given.

Usage

```
ytable_age_slice(age, variable, ytable)
```

Arguments

`age`	The age (in years) for which the time slice has to be drawn
`variable`	Name of the yield table variable for which the slice is to be taken
`ytable`	An object of class `fe_yield_table`

Details

If the age provided by the user is not directly contained in the table, linear interpolation and also extrapolation is used for obtaining the age slice. Currently, this is only done inside the general age span covered by the table (slot `$age_coverage` of the yield table object). For ages outside this range, the slice is given for the nearest covered age extreme, and a warning is issued.

Value

A (named) vector representing the vertical slice of the desired yield table variable. The names are the site indexes as defined in the yield table's element `$site_index` (in the same order) with the prefix "si_".

See Also

Other yield table functions: `fe_yield_table()`, `plot.fe_yield_table()`, `si_to_mai_age()`, `si_to_mai_max()`, `site_index()`, `stocking_level()`, `yield_tables_for_species`, `ytable_lookup()`, `ytable_max_slice()`

Examples

```
# Get the yield table heights of the Wiedemann 1943 Scots pine table at age
# 73
ytable_age_slice(
  fe_ytable_pine_wiedemann_moderate_1943,
  age = 73,
  variable = "h_q_m"
)
```

`ytable_lookup` *Look Up Values From Yield Tables*

Description

Provide yield table values for a given age and site index. If necessary, values are linearly inter- and extrapolated.

Usage

```
ytable_lookup(age, si, variable, ytable)
```

Arguments

`age`	Stand age (years)
`si`	Site index (according to the yield table of interest's (`ytable`) site index definition). If si is outside the yield table's coverage, an extrapolated value is returned, and a warning is raised.
`variable`	Name of the variable to be looked up. If the name is not one of the variable names available in the `fe_yield_table` object, the function will terminate with an error.
`ytable`	A yield table, must be an `fe_yield_table` object

Value

The requested yield table value

See Also

Other yield table functions: `fe_yield_table()`, `plot.fe_yield_table()`, `si_to_mai_age()`, `si_to_mai_max()`, `site_index()`, `stocking_level()`, `yield_tables_for_species`, `ytable_age_slice()`, `ytable_max_slice()`

Examples

```
age <- 72
si  <- 3.2

ytable_lookup(
  age, si, "h_q_m", fe_ytable_spruce_gehrhardt_moderate_1921
)

ytable_lookup(
  age, si, "v_m3_ha", fe_ytable_spruce_gehrhardt_moderate_1921
)

ytable_lookup(
  age, si, "mai_m3_ha_yr", fe_ytable_spruce_gehrhardt_moderate_1921
)
```

---

ytable_max_slice *Take a Max Slice out of an fe_yield_table_object*

Description

Max slices out of yield tables are typically required for finding out the mai max site index for a given height based standard site index.

Usage

```
ytable_max_slice(variable, ytable)
```

Arguments

`variable`	Name of the yield table variable for which the slice is to be taken
`ytable`	Name of the yield table variable for which the slice is to be taken

Details

A max slice in the sense of this function means a vector that, for each of the yield table's standard site indexes, contains the table's max value of the variable of interest.

Value

A (named) vector representing the max slice of the desired yield table variable. The names are the site indexes as defined in the yield table's element `$site_index` (in the same order) with the prefix "si_".

See Also

Other yield table functions: `fe_yield_table()`, `plot.fe_yield_table()`, `si_to_mai_age()`, `si_to_mai_max()`, `site_index()`, `stocking_level()`, `yield_tables_for_species`, `ytable_age_slice()`, `ytable_lookup()`

Examples

```
ytable_max_slice("mai_m3_ha_yr", fe_ytable_beech_wiedemann_moderate_1931)
```

Index