Package: rBDAT (via r-universe)

October 17, 2024

Type Package

Title Implementation of BDAT Tree Taper Fortran Functions

Version 1.0.1.9000

Description Implementing the BDAT tree taper Fortran routines, which were developed for the German National Forest Inventory (NFI), to calculate diameters, volume, assortments, double bark thickness and biomass for different tree species based on tree characteristics and sorting information. See Kublin (2003) <doi:10.1046/j.1439-0337.2003.00183.x> for details.

License BSD_2_clause + file LICENSE

URL `https://gitlab.com/vochr/rbdat`

Imports utils, graphics

Suggests knitr, pkgload, rmarkdown, RUnit, testthat

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

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

RemoteUrl https://gitlab.com/vochr/rbdat

RemoteRef HEAD

RemoteSha d2395e10c159cb3ef2216f61fedd02bca5fe8e55

Contents

`rBDAT-package` *Implementation of BDAT Tree Taper Fortran Functions*

Description

Implementing the BDAT tree taper Fortran routines, which were developed for the German National Forest Inventory (NFI), to calculate diameters, volume, assortments, double bark thickness and biomass for different tree species based on tree characteristics and sorting information.

References

Kublin E. (2003): Einheitliche Beschreibung der Schaftform – Methoden und Programme – BDAT-Pro. Forstw. Cbl. 122, 183–200, `https://link.springer.com/article/10.1046/j.1439-0337.2003.00183.x`

Kublin E., Scharnagl. G. (1988): Verfahrens- und Programmbeschreibung zum BWI-Unterprogramm BDAT: Abschlußbericht zum Forschungsauftrag: "Biometrische Lösungen für die Berechnung des Volumens, der Sortierung, der Rindenabzüge und der Ernteverluste im Rahmen der Bundeswald-inventur". ISSN: 0178-3165. Available at `https://gitlab.com/vochr/rbdat/-/blob/master/bdatdocs/`

`BDAT20` *BDAT 2.0 assortment function*

Description

Calculates volumes and assortments for given tree/s.

Usage

```
BDAT20(
  BDATArtNr,
  D1,
  H1 = 0,
  D2 = 0,
  H2 = 0,
  H,
  lX = 0,
  Hkz = 0,
  Skz = 0,
  Az = 0,
  Hsh = 0,
  Zsh = 0,
  Zab = 0,
  Sokz = 1,
  NMaxFixLng = 0,
  FixLngDef = matrix(rep(0, length(BDATArtNr) * 4), ncol = 4),
  result = "raw"
)
```

Arguments

`BDATArtNr`	numeric, 1 <= spp <= 36, see `getSpeciesCode`
`D1`	numeric, first measured diameter [cm], usually in 1.3m
`H1`	numeric, height of first measured diameter [m]
`D2`	numeric, second measured diameter [cm], or form parameter, see `buildTree`.
`H2`	H2: numeric, height of second measured diameter [m], or form parameter, see `buildTree`.
`H`	numeric, tree height [m]
`lX`	length of unusable wood at stem foot [m], defaults to 0
`Hkz`	indicator for tree top, 0 - normal (default), 1 - Wipfelbruch, 2 - Gipfelbruch
`Skz`	indicator for stem type, defaults to 0, see `buildTree`
`Az`	minimum cutting diameter over bark [cm], defaults to 0, using tabulated data depending on DBH (not documented)
`Hsh`	usable stem height, defaults to 0, i.e. 0.7*H
`Zsh`	minimum cutting diameter under bark for stem wood [cm], defaults to 0, using tabulated data depending on DBH (not documented)
`Zab`	minimum cutting diameter under bark for top segment [cm], defaults to 0, i.e. 14cm under bark.
`Sokz`	type assortment calculation, 0 - no assortment, 1 - mid diameter (Mittenstärke), 2 - Heilbronner Sortierung, defaults to 1
`NMaxFixLng`	number of fixed length assortments at stem foot, defaults to 0 (no fixed length assortments, irrespective of `FixLngDef`)

`FixLngDef`	matrix of 4 * length(spp), having minimum cutting diameter, required assortment length, absolute and relative add-on
`result`	indicator about what information should be returned

Value

Using default value of `result`, which is 'raw', a list is returned keeping information about the input value and produced assortments - the unprocessed returns from the Fortran code.

See `getAssortment` for more details, as this function is internally called.

See Also

`getAssortment` for a more flexible function with a more convenient english name.

Examples

```
BDAT20(BDATArtNr = c(1, 1), D1 = c(30, 25), H = c(25, 20)) # returns long list
BDAT20(BDATArtNr = c(1, 1), D1 = c(30, 25), H = c(25, 20), result = "Vol")
# size class
BDAT20(BDATArtNr = c(1, 1), D1 = c(30, 25), H = c(25, 20), result = "Skl")
BDAT20(
  BDATArtNr = c(1, 1), D1 = c(30, 25), H = c(25, 20), NMaxFixLng = 1,
  result = "Fix"
)
```

BDATBIOMASSE *Get total aboveground biomasse*

Description

BDAT-Function to get total aboveground biomass.

Usage

```
BDATBIOMASSE(BDATArtNr, D1, H1 = 0, D2 = 0, H2 = 0, H)
```

Arguments

`BDATArtNr`	numeric vector of species code, see `getSpeciesCode`
`D1`	first measured diameter of tree [cm], e.g. diameter in breast height
`H1`	measurement height of `D1` [m]
`D2`	second measured diameter of tree, see `buildTree` for details on how to specify different taper forms
`H2`	measurement height of D2, see `buildTree` for details on how to specify different taper forms
`H`	total tree height [m]

Details

This function returns total aboveground biomass for given tree/s, based on the biomass functions developed for the german NFI 3. See the additional material for some german reference.

Value

vector of biomass for given trees

See Also

`getBiomass` for a function with more convenient english name

Examples

```
## simple call of function, with all parameters given
BDATBIOMASSE(1, 30, 1.3, 0, 0, 25)

## same with variables
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
H <- 25
BDATBIOMASSE(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H)

## calling with a subset of tree characteristics
## german species names, abbreviated
BDATBIOMASSE(getSpeciesCode(c("Fi", "Bu")), 30, H = 25)
## english abbreviated
BDATBIOMASSE(getSpeciesCode(c("NS", "BE")), 30, H = 25)
```

`BDATDMRHX` *Calculate diameter in height Hx over bark.*

Description

Function to call BDAT Fortran subroutine to calculate diameter over bark in height Hx for specified tree/s.

Usage

```
BDATDMRHX(BDATArtNr, D1, H1, D2, H2, H, Hx)
```

Arguments

`BDATArtNr`	numeric vector of species code; see `getSpeciesCode`.
`D1`	first measured diameter of tree [cm], e.g. diameter in breast height.
`H1`	measurement height of `D1` [m]
`D2`	second measured diameter of tree, see `buildTree` for details on how to specify different taper forms
`H2`	measurement height of D2, see `buildTree` for details on how to specify different taper forms
`H`	total tree height [m]
`Hx`	height in tree for which diameter over bark is required

Details

conventional function interface for Fortran function BDATDMRHX. See `getDiameter` for more details.

Value

vector of diameters over bark

See Also

`BDATDORHX` for BDAT routine calculating diameter under bark and `getDiameter` for a function with a more convenient english name, more options and including a bark switch.

Examples

```
# simple call of function, with all parameters
BDATDMRHX(1, 30, 1.3, 0, 0, 25, Hx = 1.3)
# same with variables
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
H <- 25
Hx <- 1.3
BDATDMRHX(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H, Hx = Hx)
## calling with a subset of tree characteristics
## german species names, abbreviated
BDATDMRHX(getSpeciesCode(c("Fi", "Bu")), 30, 0, 0, 0, H = 25, Hx = 1.3)
## english species names abbreviated
BDATDMRHX(getSpeciesCode(c("NS", "BE")), 30, 0, 0, 0, H = 25, Hx = 1.3)
```

`BDATDORHX` *Calculate diameter in height Hx under bark.*

Description

Function to call BDAT Fortran subroutine to calculate diameter under bark in height Hx for specified tree/s.

Usage

```
BDATDORHX(BDATArtNr, D1, H1, D2, H2, H, Hx)
```

Arguments

`BDATArtNr`	numeric vector of species code; see `getSpeciesCode`.
`D1`	first measured diameter of tree [cm], e.g. diameter in breast height
`H1`	measurement height of `D1` [m]
`D2`	second measured diameter of tree, see `buildTree` for details on how to specify different taper forms
`H2`	measurement height of D2, see `buildTree` for details on how to specify different taper forms
`H`	total tree height [m]
`Hx`	height in tree for which diameter under bark is required

Details

conventional function interface for BDATDORHX. See `getDiameter` for more details.

Value

vector of diameters under bark

See Also

`BDATDMRHX` for BDAT routine calculating diameter over bark, `getDiameter` for a function with a more convenient english name, more options and a bark switch.

Examples

```
# simple call of function, with all parameters
BDATDORHX(1, 30, 1.3, 0, 0, 25, Hx = 1.3)
# same with variables
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
```

```
H <- 25
Hx <- 1.3
BDATDORHX(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H, Hx = Hx)
## calling with a subset of tree characteristics
## german species names, abbreviated
BDATDORHX(getSpeciesCode(c("Fi", "Bu")), 30, 0, 0, 0, H = 25, Hx = 1.3)
## english species names abbreviated
BDATDORHX(getSpeciesCode(c("NS", "BE")), 30, 0, 0, 0, H = 25, Hx = 1.3)
```

BDATRINDE2HX *Calculate double bark thickness*

Description

BDAT-Function to get double bark thickness at given height Hx

Usage

```
BDATRINDE2HX(BDATArtNr, D1, H1 = 1.3, D2 = 0, H2 = 0, H, Hx)
```

Arguments

`BDATArtNr`	numeric vector of species code; see `getSpeciesCode`.
`D1`	first measured diameter of tree [cm] e.g. diameter in breast height
`H1`	measurement height of `D1` [m]
`D2`	second measured diameter of tree, see `buildTree` for details on how to specify different taper forms
`H2`	measurement height of D2, see `buildTree` for details on how to specify different taper forms
`H`	total tree height [m]
`Hx`	height for which double bark thickness is required [m]

Details

This function returns double bark thickness in given height `Hx` in stem taper (hence, it depends on the diameter in given height). This can be added onto a diameter under bark to receive diameter over bark.

Value

vector of double bark thickness given height `Hx` inside stem taper.

See Also

`getBark` for a function with a convenient english name

Examples

```
## simple call of function, with all parameters given
BDATRINDE2HX(1, 30, 1.3, 0, 0, 25, Hx = 1.3)

## same with variables
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
H <- 25
Hx <- 1.3
BDATRINDE2HX(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H, Hx = Hx)
## calling with a subset of tree characteristics
## german species names, abbreviated
BDATRINDE2HX(getSpeciesCode(c("Fi", "Bu")), 30, 0, 0, 0, H = 25, Hx = 1.3)
## english abbreviated
BDATRINDE2HX(getSpeciesCode(c("NS", "BE")), 30, 0, 0, 0, H = 25, Hx = 1.3)
```

BDATVOLABMR *Calculate wood volume over bark of a tree between height A and B*

Description

BDAT-Function to get wood volume over bark of one or many trees of a section between height A and height B

Usage

```
BDATVOLABMR(BDATArtNr, D1, H1 = 1.3, D2 = 0, H2 = 0, H, A, B, SekLng = 2)
```

Arguments

`BDATArtNr`	numeric vector of species code; see `getSpeciesCode`.
`D1`	first measured diameter of tree [cm], e.g. diameter in breast height.
`H1`	measurement height of `D1` [m]
`D2`	second measured diameter of tree, see `buildTree` for details on how to specify different taper forms
`H2`	measurement height of D2, see `buildTree` for details on how to specify different taper forms
`H`	total tree height [m]
`A`	lower height of section for which volume is required [m]
`B`	upper height of section for which volume is required [m]
`SekLng`	length of section over which the integral of taper form should be applied, defaults to 2.0m

Details

wood volume is calculated using BDAT Fortran routines.

Value

vector of same length as input variables transformed into a data.frame, returning the required wood volume in cubic meter.

See Also

`BDATVOLABOR` for BDAT routine calculating volume under bark, `getVolume` for a function with a convenient english name, more options and a bark switch.

Examples

```
## simple call of function, with all parameters
BDATVOLABMR(1, 30, 1.3, 0, 0, 25, .25, 5.25, 2.0)
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
H <- 25
A <- 1
B <- 10

## same given variables
BDATVOLABMR(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H, A = A, B = B)

## calling with a subset of tree characteristics
## german species names, abbreviated
BDATVOLABMR(getSpeciesCode(c("Fi", "Bu")), 30, 0, 0, 0, H = 25, A = 0, B = 25)
## english abbreviated
BDATVOLABMR(getSpeciesCode(c("NS", "BE")), 30, 0, 0, 0, H = 25, A = 0, B = 25)
```

BDATVOLABOR *Calculate volume under bark of a tree between height A and B*

Description

BDAT-Function to get wood volume under bark of one or many trees of a section between height A and height B

Usage

```
BDATVOLABOR(BDATArtNr, D1, H1 = 1.3, D2 = 0, H2 = 0, H, A, B, SekLng = 2)
```

Arguments

`BDATArtNr`	numeric vector of species code; see `getSpeciesCode`
`D1`	first measured diameter of tree [cm], e.g. diameter in breast height
`H1`	measurement height of `D1` [m]
`D2`	second measured diameter of tree, see `buildTree` for details on how to specify different taper forms
`H2`	measurement height of D2, see `buildTree` for details on how to specify different taper forms
`H`	total tree height [m]
`A`	lower height of section for which volume is required [m]
`B`	upper height of section for which volume is required [m]
`SekLng`	length of section over which the integral of taper form should be applied, defaults to 2.0m

Details

wood volume is calculated using BDAT Fortran routines.

Value

vector of same length as input variables transformed into a data.frame, returning the required wood volume in cubic meter.

See Also

`BDATVOLABMR` for BDAT routine calculating volume over bark, `getVolume` for a function with a convenient english name, more options and including a bark switch.

Examples

```
## simple call of function, with all parameters
BDATVOLABOR(1, 30, 1.3, 0, 0, 25, .25, 5.25, 2.0)
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
H <- 25
A <- 1
B <- 10

## same with variables
BDATVOLABOR(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H, A = A, B = B)

## calling with a subset of tree characteristics
## german species names, abbreviated
BDATVOLABOR(getSpeciesCode(c("Fi", "Bu")), 30, 0, 0, 0, H = 25, A = 0, B = 25)
## english abbreviation
BDATVOLABOR(getSpeciesCode(c("NS", "BE")), 30, 0, 0, 0, H = 25, A = 0, B = 25)
```

`BDATVOLDHMR` *Calculate (coarse) wood volume over bark of a tree up to given diameter*

Description

BDAT-Function to get (coarse) wood volume over bark up to a given diameter of one or many trees.

Usage

```
BDATVOLDHMR(BDATArtNr, D1, H1 = 1.3, D2 = 0, H2 = 0, H, DHGrz = 7, SekLng = 2)
```

Arguments

`BDATArtNr`	numeric vector of species code; see `getSpeciesCode`.
`D1`	first measured diameter of tree [cm], e.g. diameter in breast height.
`H1`	measurement height of `D1` [m]
`D2`	second measured diameter of tree, see `buildTree` for details on how to specify different taper forms
`H2`	measurement height of D2, see `buildTree` for details on how to specify different taper forms
`H`	total tree height [m]
`DHGrz`	diameter over bark up to which volume should be calculated
`SekLng`	length of section over which the integral of taper form should be applied, defaults to 2.0m.

Details

Volume is calculated using BDAT Fortran routines. In particular, `BDATVOLDHMR` internally calls `BDATVOLABMR` with parameter `A = 0`, i.e. volume is calculated from forest floor up to given diameter, which itself is transformed into height `B`.

Value

vector of same length as input variables transformed into a data.frame, returning the required volume in cubic meter.

See Also

`BDATVOLDHOR` for BDAT routine calculating volume under bark, `getVolume` for a function with a convenient english name, more options and including a bark switch.

Examples

```
## simple call of function, with all parameters given
BDATVOLDHMR(1, 30, 1.3, 0, 0, 25, 7, 2.0)
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
H <- 25
DHGrz <- 7

## same using variables
BDATVOLDHMR(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H, DHGrz = DHGrz)

## calling with a subset of tree characteristics
## german species names, abbreviated
BDATVOLDHMR(getSpeciesCode(c("Fi", "Bu")), 30, 0, 0, 0, H = 25, DHGrz = 7)
## english abbreviation
BDATVOLDHMR(getSpeciesCode(c("NS", "BE")), 30, 0, 0, 0, H = 25, DHGrz = 7)
```

BDATVOLDHOR *Calculate (coarse) wood volume under bark of a tree up to given diameter*

Description

BDAT-Function to get (coarse) wood volume under bark up to a given diameter of one or many trees.

Usage

```
BDATVOLDHOR(BDATArtNr, D1, H1 = 1.3, D2 = 0, H2 = 0, H, DHGrz = 7, SekLng = 2)
```

Arguments

BDATArtNr	numeric vector of species code; see getSpeciesCode.
D1	first measured diameter of tree [cm], e.g. diameter in breast height.
H1	measurement height of D1, [m].
D2	second measured diameter of tree, see buildTree for details on how to specify different taper forms
H2	measurement height of D2, see buildTree for details on how to specify different taper forms
H	total tree height [m]
DHGrz	diameter inside bark up to which volume should be calculated
SekLng	length of section over which the integral of taper form should be applied, defaults to 2.0m.

Details

Volume is calculated using BDAT Fortran routines. In particular, `BDATVOLDHOR` internally calls `BDATVOLABOR` with parameter `A = 0`, i.e. volume is calculated from forest floor up to given diameter, which itself is transformed into height `B`.

Value

vector of same length as input variables transformed into a data.frame, returning the required volume in cubic meter.

See Also

`BDATVOLDHMR` for BDAT routine calculating volume over bark, `getVolume` for a function with a convenient english name, more options and including a bark switch.

Examples

```
## simple call of function, with all parameters
BDATVOLDHOR(1, 30, 1.3, 0, 0, 25, 7, 2.0)

## same using variables
BDATArtNr <- 1
D1 <- 30
H1 <- 1.3
D2 <- 0
H2 <- 0
H <- 25
DHGrz <- 7
BDATVOLDHOR(BDATArtNr = BDATArtNr, D1 = D1, H1 = H1, D2 = D2, H2 = H2, H = H, DHGrz = DHGrz)

## calling with a subset of tree characteristics
## german species names, abbreviated
BDATVOLDHOR(getSpeciesCode(c("Fi", "Bu")), 30, 0, 0, 0, H = 25, DHGrz = 7)
## english abbreviation
BDATVOLDHOR(getSpeciesCode(c("NS", "BE")), 30, 0, 0, 0, H = 25, DHGrz = 7)
```

`buildTree` *Build and check tree data for subsequent use in BDAT Fortran subroutines*

Description

this functions takes the data provided and builds a data.frame to be used in other BDAT get*-functions. It discriminates between different type of required output via the check-parameter. Checks are done on the type and range of the variables given to make sure calls to the Fortran routines do not suffer from type-mismatch (with potential freezing of R).

Usage

```
buildTree(tree, check = NULL, vars = NULL, mapping = NULL)
```

Arguments

`tree`	either a data.frame or a list containing the variables needed, i.e. spp, D1, H and optionally H1, D2, H2. See details for more information and parameter `mapping` for mapping of variable names.
`check`	character vector which indicates the type of required output and determines the checks to be done
`vars`	named list with additional variables for the specific BDAT-functions; see `getDiameter`, `getHeight`, `getVolume`, `getBiomass`, `getBark`, `getForm` and `getAssortment`. These variables might be included to `tree` as well, see details.
`mapping`	mapping of variable names in case a data.frame is given into parameter `tree` and `vars` between final `colnames(tree)` and required parameter names. See details.

Details

Parameter `tree` is able to take either a data.frame with correct variables names or arbitrary names if `mapping` is provided to map the data.frame names to the required names by `c("df-colname" = "var-name")` or to take a named list. If same-named variables are present in both `tree` and `vars`, priority is put on the ones in `vars` since explicitly given.

Possible variables are (*=required, depending on function):

- spp*: numeric, 1 <= spp <= 36, see `getSpeciesCode`
- D1*: numeric, first measured diameter [cm], usually at 1.3m
- H1: numeric, height of first measured diameter [m], if zero, internally transformed to 1.3m
- D2: numeric, second measured diameter [cm], or form parameter: latter is defined in conjunction with `H2`:
  - D2=0 and H2=0 => taper form of volume tables according to Grundner & Schwappach (1906-1938), the default
  - D2=0 and 0 < H2 < 100 => german NFI1-taper form, with H2 given as percentile of the NFI1-$q_{0.30}$-distribution; H2=50 corresponds to mean NFI1 taper form, H2<50 to slenderly and H2>50 to thicker trees; see `getForm` for more information about $q_{0.30}$
  - D2=0 and H2>100 => mean NFI1 taper form
  - D2>0 and H2=0 => D2 is a diameter and H2 is assumed to be 7m
  - D2>0 and H2>0 => D2 and H2 are given as diameter and height
  - -1<D2<0 => abs(D2) is interpreted as $q_{0.30}$
  - -1>D2 => mean NFI1 taper form
- H2: numeric, height of second measured diameter [m], or in conjunction with `D2`, see there.
- H*: numeric, tree height [m]
- A*: numeric, lower diameter [cm] or height [m] of section for which volume should be calculated, interpretation depends on `iAB`, see `getVolume`

- B*: numeric, upper diameter [cm] or height [m] of section for which volume should be calculated, interpretation depends on `iAB`, see `getVolume`
- sl: numeric, length of section over which should be integrated, defaults to 2.0m
- Dx*: diameter for which height or bark thickness is required
- Hx*: height for which diameter is required
- inv: inventory for which mean q03 is required, defaults to 1, see `getForm`

For deriving assortments, the following variables are optional (if not given, default values are used):

- lX: length of unusable wood at stem foot [m], defaults to 0 (X-Holz)
- Hkz: indicator for tree top, 0 - normal (default), 1 - Wipfelbruch, 2 - Gipfelbruch
  - 0 => H=H
  - 1 => H=H+2
  - 2 => DBH < 30 => H=DBH; dbh > 30 => H = 30 + (DBH-30) * 0.3
- Skz: indicator for stem type, defaults to 0
  - 0 => conifer trees => no restriction; deciduous trees => no assortments
  - 1 => monopodial deciduous trees => Hsh = 0.7*H
  - 2 => branching between dbh and 7m => Hsh = 5m
  - 3 => crown base < 3m => Hsh=0.1
  - 4 => dead or broken stem => Az = H*0.7
  - 5 => dead tree => non-usable wood
- Hsh: usable stem height, defaults to 0, i.e. 0.7*H
- Az: minimum cutting diameter over bark [cm], defaults to 0, using an exponential function given DBH
- Zsh: minimum cutting diameter under bark for stem wood [cm], defaults to 0, using parameter `Az` if estimated length < maximum length (i.e. 20m)
- Zab: minimum cutting diameter under bark for top segment [cm], defaults to 0, i.e. 14cm under bark
- Sokz: type assortment calculation, 0 - no assortment, 1 - Mid diameter (Mittenstärke), 2 - Heilbronner Sortierung, defaults to 1
- fixN: number of fixed length assortments at stem foot, defaults to 0 (no fixed length assortments, irrespective of other fix* parameters)
- fixZ: mininum diameter under bark for fixed length assortment at stem foot, defaults to 0
- fixL: length of fixed length assortment at stem foot, defaults to 0
- fixA: fixed length assortement add-on in [cm], defaults to 0
- fixR: fixed length assortement add-on in [%], defaults to 0

If parameter `tree` is used to hand over all tree data in form of a data.frame, at least the parameter spp, D1, H must be provided, eventually mapped via `mapping`. Parameter `Hx` and `Dx`, which specify height and diameter for which a diameter or height is requested, respectively, can either be included to the definition of the tree data or alternatively given separately using the vars parameter. In that case, vars is used in priority to a identically named variable in `tree`. Additionally, `tree` and `vars` are merged via a full outer join. The add-on in fixed length assortments can be given in absolute and relative units at the same time, but the higher value will be used.

Value

a data.frame of class datBDAT.<check> having all variables needed in specific functions. If check is NULL, only a basic tree-data.frame of class "datBDAT" is returned.

Examples

```
## example for only tree data
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
res <- buildTree(tree = tree)
head(res)
class(res)

tree <- list(species = c(1, 1), dbh = c(30, 25), h = c(25, 30))
mapping <- c("species" = "spp", "dbh" = "D1", "h" = "H")
res <- buildTree(tree = tree, mapping = mapping)
head(res)
class(res)

## example for diameter calculation
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
vars <- list(Hx = c(1.3, 1.3))
mapping <- NULL
res <- buildTree(tree = tree, check = "diameter", vars = vars)
head(res)
class(res)
tree <- list(Art = c(1, 1), Bhd = c(30, 25), H = c(25, 30))
vars <- list(X = c(1.3, 1.3))
mapping <- c("Art" = "spp", "Bhd" = "D1", "X" = "Hx")
res <- buildTree(tree = tree, check = "diameter", vars = vars, mapping = mapping)
head(res)
class(res)

## example with many diameters for one tree
tree <- list(spp = c(1), D1 = c(30), H = c(25))
vars <- list(Hx = seq(0, 25, 0.1))
mapping <- NULL
res <- buildTree(tree = tree, check = "diameter", vars = vars)

tree <- data.frame(s = 1, d = 30, h = 25, hx = 1.3)
mapping <- c("s" = "spp", "d" = "D1", "h" = "H", "hx" = "Hx")
res <- buildTree(tree, check = "diameter", mapping = mapping)
head(res)
class(res)

## example for height calculation
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
vars <- list(Dx = c(30, 25))
res <- buildTree(tree = tree, check = "height", vars = vars)
head(res)
class(res)

## example for volume calculation
```

```
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
check <- "volume"
vars <- list(A = c(30, 25), B = c(7, 7), sl = 0.1)
mapping <- NULL
res <- buildTree(tree = tree, check = "volume", vars = vars)
head(res)
class(res)

## example for bark calculation
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
vars <- list(Hx = c(1.3, 1.3))
res <- buildTree(tree = tree, check = "bark", vars = vars)
head(res)
class(res)

## example for assortment calculation
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
vars <- list(fixN = 1, fixZ = 10, fixL = 5, fixA = 10, fixR = 0.1)
res <- buildTree(tree = tree, check = "assortment", vars = vars)
head(res)
class(res)

## for cases where 'vars' could be a vector (i.e. getBark, getDiameter and
## getHeight), the following is also possible
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
vars <- c(1.3, 1.3)
res <- buildTree(tree = tree, check = "bark", vars = vars)
head(res)
class(res)

res <- buildTree(tree = tree, check = "height", vars = vars)
head(res)
class(res)

## but it is not possible in case of getVolume or getAssortment
## instead, use a named list to achieve a cross join / cartesian product
vars <- list(A = rep(1, 3), B = 5:7)
res <- buildTree(tree = tree, check = "volume", vars = vars)
head(res)
class(res)

## example for 'biomass' calculation
tree <- list(spp = c(1, 1), D1 = c(30, 25), H = c(25, 30))
res <- buildTree(tree = tree, check = "biomass")
head(res)
class(res)

## example with H1 != 1.3m
tree <- list(
  spp = c(1, 1), D1 = c(30, 25), H1 = c(2, 2), H = c(25, 30)
)
res <- buildTree(tree = tree, check = "biomass")
head(res)
```

```
class(res)
getBiomass(res)
```

---

getAssortment — *Get assortments for one or many trees*

Description

Function to get assortments given harvest specifications for a tree of dimension D1, possible D2, and H. Assortments could be diameter or length constrained.

Usage

```
getAssortment(tree, ...)

## S3 method for class 'data.frame'
getAssortment(tree, sort = NULL, mapping = NULL, value = "merge", ...)

## S3 method for class 'list'
getAssortment(tree, sort = NULL, mapping = NULL, value = "merge", ...)

## S3 method for class 'datBDAT'
getAssortment(tree, sort = NULL, mapping = NULL, value = "merge", ...)
```

Arguments

`tree`	either an object of class 'datBDAT.assortment' or a data.frame or list containing the variables needed, i.e. spp, D1, H, and optionally H1, D2, H2 for specifying the tree. See `buildTree` for more information and parameter `mapping` for mapping of variable names. Indeed, `tree` could additionally take the variables specified in `sort`; otherwise a full outer join is produced between `tree` and `sort`
`...`	passing arguments to methods.
`sort`	named list with variables specifying assortments, see `buildTree`.
`mapping`	mapping of variable names in case a data.frame is given into parameter `tree` between colnames(`tree`) and required parameter names. See details.
`value`	character vector indicating return type: either "Vol", "Skl", "Fix", "LDSort", "merge" (default) or "raw". See section Value.

Details

Parameter 'tree' is able to take either a data.frame with correct variables names or arbitrary names if `mapping` is provided to map the data.frame names to the required names by `c("df-colname" = "var-name")` or to take a named list.

Assortments are calculated using BDAT2.0 Fortran routines. Slightly extended, it now is possible to return length and diameter information also about standard assortments (which are held in "Vol"-element of return list, if value="raw" or value="Vol").

The standard assortment names are:

- X = non-usable wood at stem foot (X-Holz)
- Sth = stem wood
- Ab = upper part of stem wood, second length after transport cut
- Ind = industrial wood
- nvDh = non-usable coarse wood

Value

depending on `value` either information about 'Skl,' 'Vol' or 'Fix' (these are elements of standard output of Fortran BDAT20 subroutine) and, additionally, information about the 'Vol' elements is retrieved by `value="LDSort"` or if `value` = 'merge' than a combined information of all produced assortments with name, base position, assortment length, mid-diameter, top-diameter and volume is produced. This is most likely what you want, hence the default. Standard output of BDAT is provided by `value="raw"`. Since v0.4.0 a vectorized BDAT-fortran-function is implemented, so each element of the returned list in case of value="raw" keeps information of all trees given. Make sure to appropriately split and manipulate this data.

Methods (by class)

- `getAssortment(data.frame)`: transforming `data.frame` before calling `getAssortment` using `buildTree`
- `getAssortment(list)`: transforming `list` before calling `getAssortment` using `buildTree`
- `getAssortment(datBDAT)`: class method for class `datBDAT`

See Also

`BDATVOLABMR` and `BDATVOLABOR` for a BDAT-type wrapper and function to keep back-compatability. To get coarse wood volume over bark (>=7m diameter over bark) BDAT-type functions are `BDATVOLDHMR` and `BDATVOLDHOR`.

Examples

```
tree <- data.frame(spp = 1, D1 = 30, H = 25)
sort <- list(Az = 7, Sokz = 1)
getAssortment(tree, sort, value = "Vol")
getAssortment(tree, sort, value = "Skl")
sort <- list(Az = 7, Sokz = 1, fixN = 1, fixZ = 10, fixL = 5,
             fixA = 10, fixR = 1)
getAssortment(tree, sort, value = "Vol")
getAssortment(tree, sort, value = "Skl")
getAssortment(tree, sort, value = "Fix")
getAssortment(tree, sort, value = "LDSort")
getAssortment(tree, sort, value = "merge")

## prepare data for repeated sorting
## (get rid of preparating data handling)
n <- 3
tree <- data.frame(
  spp = rep(1, n), D1 = seq(20, 50, length.out = n),
  H = seq(15, 40, length.out = n)
)
sort <- list(lX = 0, Sokz = 1, Az = 7,
             fixN = 2, fixZ = 10, fixL = 5,
             fixA = 10, fixR = 1)
tree <- buildTree(tree = tree, check = "assortment", vars = sort)
getAssortment(tree, value = "Vol")
getAssortment(tree, value = "Skl")
getAssortment(tree, value = "Fix")
getAssortment(tree, value = "LDSort")
getAssortment(tree, value = "merge")

## to get bare BDAT-Output, use value='raw'
# very long list, each element keeping all trees
getAssortment(tree, value="raw")
# bonus: it returns the calling parameters as well
```

getBark *Get double bark thickness of tree at given height Hx*

Description

this function calculates double bark thickness in given height for a given tree

Usage

```
getBark(tree, ...)

## S3 method for class 'data.frame'
getBark(tree, Hx = NULL, mapping = NULL, ...)

## S3 method for class 'list'
getBark(tree, Hx = NULL, mapping = NULL, ...)

## S3 method for class 'datBDAT'
getBark(tree, Hx = NULL, mapping = NULL, ...)
```

Arguments

`tree`	either a data.frame or a list containing the variables needed to decribe a tree, i.e. spp, D1, H, and optionally H1, D2, H2. Additionally, parameter `Hx` might be directly given via `tree`. See `buildTree` for more details and parameter `mapping` for mapping of variable names.
`...`	passing arguments to methods.
`Hx`	height in tree for which double bark thickness is required
`mapping`	mapping of variable names in case a data.frame is given into parameter `tree` between colnames(`tree`) and required parameter names. See details.

Details

This function returns double bark thickness in given height `Hx` in stem taper (hence, it depends on the diameter in given height). This can be added to an diameter under bark to get the diameter over bark. Parameter `tree` is able to take either a data.frame with correct variables names or arbitrary names if `mapping` is provided to map the data.frame names to the required names by `c("df-colname" = "var-name")` or to take a named list.

Value

vector of double bark thickness given height `Hx` inside stem taper

Methods (by class)

- `getBark(data.frame)`: transforming `data.frame` before calling `getBark` using `buildTree`
- `getBark(list)`: transforming `list` before calling `getBark` using `buildTree`
- `getBark(datBDAT)`: class method for class `datBDAT`

Examples

```
tree <- data.frame(spp = c(1, 1), D1 = c(30, 25), H = c(25, 25), Hx = c(1.3, 22.248))
getBark(tree)
tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 25), h = c(25, 25), Hx = c(1.3, 22.248))
getBark(tree, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))
tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 25), h = c(25, 25))
Hx <- list(Hx = c(1.3, 22.248))
getBark(tree = tree, Hx = Hx, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))
```

`getBiomass` *Get total aboveground biomass of tree*

Description

this function calculates total aboveground biomass for a given tree

Usage

```
getBiomass(tree, ...)

## S3 method for class 'data.frame'
getBiomass(tree, mapping = NULL, ...)

## S3 method for class 'list'
getBiomass(tree, mapping = NULL, ...)

## S3 method for class 'datBDAT'
getBiomass(tree, mapping = NULL, ...)
```

Arguments

`tree`	either a data.frame or a list containing the variables needed to decribe a tree, i.e. spp, D1, H, and optionally H1, D2, H2. See `buildTree` for details and parameter `mapping` for mapping of variable names
`...`	passing arguments to methods.
`mapping`	mapping of variable names in case a data.frame is given into parameter `tree` between colnames(`tree`) and required parameter names. See details.

Details

This function returns total aboveground biomass according to the biomass functions developed for the german NFI3 (BWI3) on the basis of a field survey covering whole Germany from 2007 to 2010. Hence, the base data differs from those used to fit the taper functions. Neverteless and although the original fortran code does not provide an interface to pass H2 or negative D2 values to the functions (they were fitted with abolute D03-values, i.e. diameter in 30 for reasons of consistency.

The functions themselves are fitted for four species (Norway spruce, Scots pine, European beech and Oak spp.) directly, and for another fourteen species by generating pseudo-observations and fitting the functions to these. See the enclosed report for details.

The biomass functions integrate four different ranges of validity: (i) tree height below 1.3m, (ii) dbh below 10cm, (iii) dbh between 10cm and the 99th-quantile of the data and (iv) dbh above.

Value

vector of total aboveground biomass

Methods (by class)

- `getBiomass(data.frame)`: transforming `data.frame` before calling `getBiomass` using `buildTree`
- `getBiomass(list)`: transforming `list` before calling `getBiomass` using `buildTree`
- `getBiomass(datBDAT)`: class method for class `datBDAT`

References

Riedel, T. and G. Kaendler (2017). "Nationale Treibhausgasberichterstattung: Neue Funktionen zur Schätzung der oberirdischen Biomasse am Einzelbaum." Forstarchiv 88(2): 31-38.

Examples

```
tree <- data.frame(spp = c(1, 1), D1 = c(30, 25), H = c(25, 25))
getBiomass(tree)

tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 25), h = c(25, 25))
getBiomass(tree, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))

tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 25), h = c(25, 25))
getBiomass(tree = tree, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))

## this is standard usage of the fortran code
## now changing the taper form via second diameter D2
tree <- data.frame(spp = 1, D1 = 30, D2 = c(26, 24), H = 25)
getBiomass(tree)

## the following usage of D2 and H2 w.r.t. the biomass functions are new
## and unique to R
## now changing the taper form via quantile of second diameter in H2
tree <- data.frame(spp = 1, D1 = 30, H2 = c(25, 50, 75), H = 25)
getBiomass(tree)

## now changing the taper form via form quotient (i.e. negative D2)
fq <- getForm(list(spp = 1, D1 = 30, H = 25), inv = 1:4)
tree <- data.frame(spp = 1, D1 = 30, D2 = -fq, H = 25)
getBiomass(tree) # biomass of an average tree according to different inventories
```

getDiameter *Get diameter in given height inside tree taper*

Description

this function calculates the diameter inside or outside bark of in given height for a given tree

Usage

```
getDiameter(tree, ...)

## S3 method for class 'data.frame'
getDiameter(tree, Hx = NULL, bark = TRUE, mapping = NULL, ...)

## S3 method for class 'list'
getDiameter(tree, Hx = NULL, bark = TRUE, mapping = NULL, ...)

## S3 method for class 'datBDAT'
getDiameter(tree, Hx = NULL, bark = TRUE, mapping = NULL, ...)
```

Arguments

`tree`	either a data.frame, a list or an object of class `datBDAT` containing the variables needed to decribe a tree, i.e. spp, D1, H, and optionally H1, D2, H2. See `buildTree` for details and parameter `mapping` for mapping of variable names
`...`	passing arguments to methods.
`Hx`	height in tree for which diameter over or under bark is required; defaults to NULL
`bark`	logical, if TRUE returned diameter `Dx` is over bark, if FALSE returned diameter is under bark. Coerced to logical by `as.logical(bark[1])`.
`mapping`	mapping of variable names in case a data.frame is given into parameter `tree` between colnames(`tree`) and required parameter names. See details.

Details

if tree does not includes variable Hx, a full outer join is generated between both

Value

a matrix with one row for each tree and one column for each `Hx` given, holding the diameter over or under bark of provided height `Hx` inside stem taper. The matrix is simplified by `[,,drop=TRUE]`, especially if `Hx=NULL`.

Methods (by class)

- `getDiameter(data.frame)`: transforming `data.frame` before calling `getDiameter` using `buildTree`
- `getDiameter(list)`: transforming `list` before calling `getDiameter` using `buildTree`
- `getDiameter(datBDAT)`: class method for class `datBDAT`

Examples

```
tree <- data.frame(spp = c(1, 1), D1 = c(30, 30), H = c(25, 25), Hx = c(1.3, 22.248))
getDiameter(tree, bark = TRUE)
getDiameter(tree, bark = FALSE)

tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 30), h = c(25, 25), Hx = c(1.3, 22.248))
getDiameter(tree, bark = TRUE, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))

tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 30), h = c(25, 25))
Hx <- c(1.3, 22.248)
getDiameter(tree, Hx = Hx, bark = TRUE, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))
```

`getForm` *Get estimated mean of form factor q03*

Description

this function returns for a given dbh-height-class and species the estimated mean of the q03-distribution, which can be considered as a form factor for different inventories (i.e. NFI 1, NSoG, IS08, NFI3).

Usage

```
getForm(tree, ...)

## S3 method for class 'data.frame'
getForm(tree, inv = NULL, mapping = NULL, ...)

## S3 method for class 'list'
getForm(tree, inv = NULL, mapping = NULL, ...)
```

```
## S3 method for class 'datBDAT'
getForm(tree, inv = NULL, mapping = NULL, ...)
```

Arguments

`tree`	either a data.frame or a list containing the variables needed to decribe a tree, i.e. spp, D1, H, and optionally H1, D2, H2. See `buildTree` for details and parameter `mapping` for mapping of variable names
`...`	passing arguments to methods.
`inv`	integer, indicator for which inventory the form factor should be returned; defaults to 1 (0=volume tables of Grundner-Schwappach 1921, 1=NFI1, 2=NSoG, 3=IS08, 4=NFI3). Any other value is silently set back to `inv=1`.
`mapping`	mapping of variable names in case a data.frame is given into parameter `tree` between colnames(`tree`) and required parameter names.

Details

This function returns the estimated mean of the q03-distribution for a given dbh-height-class and species. The q03-distribution is the ratio between the diameter in 30% of tree height and 5% of tree height and describes the form of the taper curve. This value can be returned for different inventories, during which upper stem diameters have been sampled.

Only spp, D1 and H are used to estimate mean q03, if provided and H1 equals 1.3m, otherwise dbh is estimated using BDAT functions (i.e. getDiameter(tree, Hx=1.3)).

Value

vector of form factor q03.

Methods (by class)

- `getForm(data.frame)`: transforming `data.frame` before calling `getForm` using `buildTree`
- `getForm(list)`: transforming `list` before calling `getForm` using `buildTree`
- `getForm(datBDAT)`: class method for class `datBDAT`

Examples

```
tree <- data.frame(spp = c(1, 1), D1 = c(30, 25), H = c(25, 25))
tree <- buildTree(tree, check = "form", vars = list(inv = 1))
str(tree)
getForm(tree) # default is 'inv=1'
getForm(tree, inv = 0) # taper form of BDAT from model fit
getForm(tree, inv = 1) # NFI 1 (without NSoG)
getForm(tree, inv = 2) # NSoG = New States of Germany
getForm(tree, inv = 3) # carbon inventory 2008
getForm(tree, inv = 4) # NFI 3 (reunificated Germany)
```

getHeight — *Get height of given diameter inside tree taper*

Description

this function calculates the height of a given diameter inside or outside bark for a given tree

Usage

```
getHeight(tree, ...)

## S3 method for class 'data.frame'
getHeight(tree, Dx = NULL, bark = TRUE, mapping = NULL, ...)

## S3 method for class 'list'
getHeight(tree, Dx = NULL, bark = TRUE, mapping = NULL, ...)

## S3 method for class 'datBDAT'
getHeight(tree, Dx = NULL, bark = TRUE, mapping = NULL, ...)
```

Arguments

`tree`	either a data.frame or a list containing the variables needed to decribe a tree, i.e. spp, D1, H, and optionally H1, D2, H2. See `buildTree` for details and parameter `mapping` for mapping of variable names
`...`	passing arguments to methods.
`Dx`	diameter of tree for which height is required; defaults to NULL
`bark`	logical, if TRUE given diameter `Dx` is considered over bark, if FALSE diameter is considered to be under bark. Coerced to logical by `as.logical(bark[1])`.
`mapping`	mapping of variable names in case a data.frame is given into parameter `tree` between colnames(`tree`) and required parameter names.

Details

see `buildTree` for how to specify a tree object

Value

a matrix with one row for each tree and one column for each `Dx` given, holding the height of provided diameter `Dx` inside stem taper. The matrix is simplified by `[,,drop=TRUE]`, especially if `Dx=NULL`.

Methods (by class)

- `getHeight(data.frame)`: transforming `data.frame` before calling `getHeight` using `buildTree`
- `getHeight(list)`: transforming `list` before calling `getHeight` using `buildTree`
- `getHeight(datBDAT)`: class method for class `datBDAT`

Examples

```
tree <- data.frame(spp = c(1, 1), D1 = c(30, 25), H = c(25, 20), Dx = c(7, 7))
getHeight(tree, bark = TRUE)
getHeight(tree, bark = FALSE)

tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 25), h = c(25, 20), Dx = c(7, 7))
getHeight(tree, bark = TRUE, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))

tree <- data.frame(BDATCode = c(1, 1), dbh = c(30, 25), h = c(25, 20))
Dx <- c(7, 5)
getHeight(tree, Dx = Dx, bark = TRUE, mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H"))

tree <- data.frame(spp = c(1, 1), D1 = c(30, 25), H = c(25, 20), Dx = c(7, 7))
getHeight(tree, Dx = c(1:5), bark = TRUE)
```

---

getSpeciesCode — *Get BDAT species code or transform it to a name.*

Description

Function to get BDAT species code, or transform it to a german or english name, possibly an abbreviated version or even a scientific name

Usage

```
getSpeciesCode(inSp = NULL, outSp = NULL)
```

Arguments

`inSp`	species information given, either numeric or character
`outSp`	character vector of names, for which information should be returned

Details

The function matches inSp to outSp. Depending on inSp, being either a numeric vector of values between 1 and 36 or a character vector of species names. Possible names are those which could be return values. One can get all names and the respective species code by calling the function with inSP=NULL and outSP=NULL (the default).

English species names and codes are taken from https://www.forestry.gov.uk/pdf/PF2011_Tree_Species.pdf/$FILE/PF2011_ while slightly adjusting the codes to be unique compared to the german codes (e.g. European larch is now ELA instead of EL).

Any given species code outside the interval [1, 36] is given the code 1 (i.e. Norway spruce), while throwing a warning. If any inSp - name is invalid, i.e. not in species list, this throws an error.

All elements of outSp, which are not colnames of the default returned data.frame, are silently dropped.

Value

vector or data.frame, depending on length of 'outSp'.

Examples

```
getSpeciesCode(inSp = NULL, outSp = NULL) ## the default
getSpeciesCode() ## the same
getSpeciesCode(outSp = "scientific")
getSpeciesCode(inSp = c(1, 2)) ## giving codes
getSpeciesCode(inSp = c(1, 2, -1, 37)) ## values outside [1, 36] are given code 1
getSpeciesCode(inSp = c(1, 2), outSp = c("scientific")) ## output a vector
getSpeciesCode(inSp = c("Bu", "Fi")) ## asking for codes of abbreviated german names
getSpeciesCode(inSp = c("Bu", "Fi", "Bu")) ## order is preserved
getSpeciesCode(inSp = c("Buche", "Fichte")) ## asking for codes of german names
getSpeciesCode(inSp = c("BE", "NS")) ## ... abbreviated english names
getSpeciesCode(inSp = c("beech", "Norway spruce")) ## ... english names
getSpeciesCode(inSp = c("Fagus sylvatica", "Picea abies")) ### ... scientific names
```

`getVolume` *Get segment volume for one or many trees*

Description

Function to get segment volume over or under bark for a tree of dimension D1, possible D2, and H. Volume is calculated for a given segment defined by A and B, which might be specified as heights or diameters.

Usage

```
getVolume(tree, ...)

## S3 method for class 'data.frame'
getVolume(tree, AB = NULL, iAB = "H", bark = TRUE, mapping = NULL, ...)

## S3 method for class 'list'
getVolume(tree, AB = NULL, iAB = "H", bark = TRUE, mapping = NULL, ...)

## S3 method for class 'datBDAT'
getVolume(tree, AB = NULL, iAB = "H", bark = TRUE, mapping = NULL, ...)
```

Arguments

`tree`	either a data.frame or a list containing the variables needed to decribe a tree, i.e. spp, D1, H, and optionally H1, D2, H2. See `buildTree` for details and parameter `mapping` for mapping of variable names
`...`	passing arguments to methods.

AB	list with heights or diameters A and B of section for which volume over or under bark should be calculated. Additionally, add in sl for the segment length over which the integral should be calculated. See details.
iAB	character indicating how to interpret given A and B values. Either "H" (the default), "Dob" (diameter over bark) or "Dub" (diameter under bark). Could be of length one or two, depending on whether A and B are both height or diameter variables or not. See examples.
bark	boolean of length one indicator for whether required volume should include bark volume or not. Defaults to TRUE. Coerced to logical by as.logical(bark[1]).
mapping	mapping of variable names in case a data.frame is given into parameter tree between colnames(tree) and required parameter names. See details.

Details

iAB can be a vector of length two, indicating how to interpret A and B. Hence, one can calculate volume between a given height and a given diameter, either over or under bark. If of length one, it is assumed the indicator applies to both A and B.

Internally, provided diameters in A or B are tranformed to heights by BDATHXDX if iAB="dob" or by BDATHXDXOR if iAB="dub".

Value

vector of same length as tree, returning the required volume in cubic meter.

Methods (by class)

- getVolume(data.frame): transforming data.frame before calling getVolume using buildTree
- getVolume(list): transforming data.frame before calling getVolume using buildTree
- getVolume(datBDAT): class method for class datBDAT

See Also

BDATVOLABMR and BDATVOLABOR for a BDAT-type wrapper and function to keep back-compatability. To get total coarse wood volume (>=7m diameter over bark) BDAT-type functions are BDATVOLDHMR and BDATVOLDHOR.

Examples

```
## default return with just one or several trees given is total coarse wood
## volume over bark, i.e. stock volume (in german: Vfm m.R.)
getVolume(list(spp = 1, D1 = 30, H = 25))

## first using a data.frame and height information
tree <- data.frame(spp = 1, D1 = 30, H = 25, A = 0.1, B = 10)
getVolume(tree) # iAB = "H", bark = TRUE, mapping = NULL as default values
getVolume(tree, iAB = "H", bark = FALSE, mapping = NULL)

## now, use diameter information
tree <- data.frame(spp = 1, D1 = 30, H = 25, A = 30, B = 7)
```

```
getVolume(tree, iAB = "Dob", bark = TRUE, mapping = NULL)

## now use both diameter and height information
tree <- data.frame(spp = 1, D1 = 30, H = 25, A = 0, B = 7)
getVolume(tree, iAB = c("H", "Dob"), bark = TRUE, mapping = NULL)
## is equivalent to the original BDAT-function:
BDATVOLDHMR(1, 30, 0, 0, 0, H = 25)
## and
getVolume(tree, iAB = c("H", "Dob"), bark = FALSE, mapping = NULL)
## is equivalent to:
BDATVOLDHOR(1, 30, 0, 0, 0, H = 25)

## use a misnamed data.frame and mapping argument
tree <- data.frame(BDATCode = 1, dbh = 30, h = 25, l = 30, u = 7)
getVolume(tree,
  iAB = "Dob", bark = TRUE,
  mapping = c("BDATCode" = "spp", "dbh" = "D1", "h" = "H", "l" = "A", "u" = "B")
)

## using a list to provide the data
tree <- list(
  spp = c(1, 1), D1 = c(30, 25), H = c(25, 30), A = c(30, 25),
  B = c(7, 7)
)
getVolume(tree, iAB = "Dob") #' defaults: bark = TRUE, mapping = NULL
getVolume(tree, iAB = "Dob", bark = FALSE) #' defaults: mapping = NULL

## using a misnamed list and mapping argument
tree <- list(
  BDATCode = c(1, 1), dbh = c(30, 25), H = c(25, 30),
  A = c(0.1, 0.1), B = c(1, 1)
)
getVolume(tree, mapping = c("BDATCode" = "spp", "dbh" = "D1"))

## using parameter AB to provide the segment data
## in this case a cross join between tree and AB is produced and evaluated
tree <- list(BDATCode = c(1, 1), dbh = c(30, 25), H = c(25, 30))
AB <- list(A = c(0.1, 1), B = c(1, 2))
getVolume(tree, AB, iAB = "H", bark = TRUE, mapping = c("BDATCode" = "spp", "dbh" = "D1"))

## effect of adjusting 'sl'
tree <- list(spp = 1, D1 = 30, H = 25)
getVolume(tree = tree, AB = list(A = 0.1, B = 5.1, sl = 2)) # default
getVolume(tree = tree, AB = list(A = 0.1, B = 5.1, sl = 1))
getVolume(tree = tree, AB = list(A = 0.1, B = 5.1, sl = 0.1))
getVolume(tree = tree, AB = list(A = 0.1, B = 5.1, sl = 0.01))

## compare Smalian formula to mid-diameter volume for sl=5
R1 <- getDiameter(tree, Hx = 1.3) / 100 / 2 # radius in m
R2 <- getDiameter(tree, Hx = 6.3) / 100 / 2 # radius in m
h <- 5
pi * ((R1 + R2) / 2)^2 * h # average of bottom and top diameter
(h * pi) / 3 * (R1^2 + R1 * R2 + R2^2) # truncated-cone volume
```

```
getVolume(tree, AB = list(A = 1.3, B = 6.3, sl = 5)) # mid-diameter volume
getVolume(tree, AB = list(A = 1.3, B = 6.3, sl = 0.01)) # physical volume
```

plot.datBDAT *Plot taper curve of a tree*

Description

creating a plot of the taper curve of a tree, over or under bark

Usage

```
## S3 method for class 'datBDAT'
plot(x, bark = NULL, col.bark = NULL, legend = FALSE, assort = NULL, ...)
```

Arguments

`x`	an object of class 'datBDAT'
`bark`	either NULL or logical; if TRUE taper curve over bark is plotted, if FALSE taper curve under bark is plotted; if NULL, both are plotted
`col.bark`	color to be used for plot of bark, if plot of taper curve over and under bark is requested
`legend`	logical, if legend should be added
`assort`	assortments produced by `getAssortment(, value="merge")`
`...`	further arguments for `plot` and `points`

Details

Creates graphics of the taper curve of trees. Either over bark or under bark, or both. Elements design can partly be chosen. If assortments are given, these are added to the plot. Doing that, the assortment bottom and top position is indicated by a vertical line and mid-diameter is shown as a point with vertical dashed line. N.B. the mid-diameter shown is under bark and rounded downwards for 0.5 cm if mid-diameter < 20 and for 0.75 cm if bigger. Assortment volume is calculated using this diameter according to the legal rules for roundwood assortments (formerly german Forst-HKS and now RVR). Additionally, assortment names are indicated. One can provide assortment names in a column of `assort` named 'assortname', which will be used if available, otherwise the 'Sort'-column will be used. See Examples.

Value

No return value, called for side effects

Examples

```
## plotting the taper curve of a tree
oldpar <- par(no.readonly = TRUE)
par(mfrow = c(1, 1))
t <- data.frame(spp = 1, D1 = 40, H = 35)
tree <- buildTree(tree = t)
plot(tree, type = "l", las = 1, legend = TRUE)
plot(tree, bark = TRUE, las = 1)
plot(tree, bark = FALSE, las = 1)
t <- data.frame(spp = c(1, 1), D1 = c(40, 35), H = c(35, 30))
tree <- buildTree(tree = t)
plot(tree, bark = FALSE, las = 1, legend = TRUE)
plot(tree, bark = TRUE, las = 1, legend = TRUE)

t <- data.frame(spp = c(1, 8), D1 = 40, H = 35)
tree <- buildTree(tree = t)
plot(tree, bark = NULL, las = 1, col.bark = "blue", legend = TRUE)
plot(tree[1, ], main = getSpeciesCode(tree[1, ]$spp, out = "long"))
plot(tree[2, ], main = getSpeciesCode(tree[2, ]$spp, out = "scientific"))
par(mfrow = c(2, 1))
plot(tree, bark = TRUE, las = 1)

## now add assortments into taper curve
par(mfrow = c(1, 1))
ass <- getAssortment(tree, sort = list(lX = 1, fixN = 2, fixL = 4, fixA = 10))
plot(tree, assort = ass)
plot(tree, bark = FALSE, assort = ass)
plot(tree, bark = FALSE, assort = ass, legend = TRUE)
plot(tree[1, ], assort = ass[ass$tree == 1, ], main = "first tree in subset")
plot(tree[2, ], assort = ass[ass$tree == 2, ], main = "second tree in subset")

## adding own assortment labels using column 'assortname'
ass$assortname <- ifelse(grepl("Fix", ass$Sort), paste0("Fix:", ass$length), ass$Sort)
plot(tree, assort = ass)
par(oldpar)
```

---

updateBdatNamespace *replace NAMESPACE from 'rBDATPRO' to 'rBDAT'.*

Description

function replaces calls to package 'rBDATPRO' by calls to package 'rBDAT', simply by replacing the package name.

Usage

```
updateBdatNamespace(inpath = file.choose(), outpath = NULL)
```

Arguments

`inpath`	path of file to process
`outpath`	either a file name or a directory, both or NULL, see details.

Details

This function merely exists to account for the renaming of the package from `rBDATPRO` (which was internally used for some period of time) to `rBDAT`. Its sole purpose is to update R-scripts which use `rBDATPRO` and now should be updated to use `rBDAT`. Internally, `gsub` is used.

`outpath` can be (i) a filename, then the newly generated file is stored under that name in the `inpath` directory, (ii) a directory, then the `inpath`-filename is used with prefixed `rbdat_`, (iii) a complete path (directory name and file name), then this is used to store the file, (iv) NULL, then the inpath is used with prefixed `rbdat_` to the inpath filename.

Value

a character holding path and filename of the updated file

Examples

```
## Not run:
p <- tempdir()
f <- "rbdatpro.r"
tx <- c("require(rBDATPRO)", "library(rBDATPRO)",
        "rBDATPRO::getDiameter(list(spp=1, D1=30, H=27))")
pf <- file.path(p, f)
writeLines(tx, con=pf)
file.exists(pf)
list.files(p)
# file.show(pf)

## define different output specs
outpath1 <- file.path(tempdir(), "devel/rbdatScript.r")
outpath2 <- p
outpath3 <- "rbdatScript.r"

(updated_file <- updateBdatNamespace(pf, outpath = NULL))
list.files(p)
# file.show(updated_file)

(updated_file <- updateBdatNamespace(pf, outpath1))
list.files(file.path(p, "devel"))
# file.show(updated_file)

(updated_file <- updateBdatNamespace(pf, outpath2))
list.files(p)
# file.show(updated_file)

(updated_file <- updateBdatNamespace(pf, outpath3))
list.files(p)
# file.show(updated_file)
```

```
## End(Not run)
```

Index