Package ‘FAIBBase’

June 18, 2024

Title Basic functions for forest mensuration and ecology

Version 2.0.0

Description Basic R fuctions for forest mensuration and ecology.

License Apache License (== 2.0) | file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports methods,
dplyr,
data.table,
fpCompare,
rmarkdown,
sp,
bcmaps,
raster,
sf,
spatstat,
stringr

Suggests knitr

R topics documented:

annualGrowthRateCalculator 2
appendedCat 3
areaProportion 3
BEC2IC 4
biomassCalculator 5
checkLD_remeas 6
checkMissing_remeas 7
checkSize_remeas 8
convertSI2SI 9
DBHClassifier 10
DIB_ICalculator 11
getSpatial 12
HegyiCICalculator 13
heightEstimateForBTOP_D 14

heightEstimateForBTOP_H 15
lm_group 16
merge_dupUpdate 17
PHFCalculator 18
randomStemMapping 18
SIInBC 19
standardizeSpeciesName 20
stemMapping 21
stemMappingExtension 22
stemMappingExtension_square 24
taperCoeffsGenerator 26
taperImplementor 26
treeProfile 28
treeVolCalculator 29
treeVolume 31
UTM_Convertor 32

Index **34**

`annualGrowthRateCalculator`

Calculate annual growth rate

Description

This function is to calcualte annual growth rate.

Usage

```
annualGrowthRateCalculator(
  boredDiameter,
  growthIncrement,
  growthYear,
  barkThickness
)
```

Arguments

`boredDiameter`	numeric, Diameter at bored height in cm.
`growthIncrement`	numeric, Growth increment in mm over a time period.
`growthYear`	numeric, Number of years over which growth increment is measured.
`barkThickness`	numeric, Bark thickness in mm. If missing, 0.05 will be used.

Value

Calculated annual growth rate.

Author(s)

Yong Luo

`appendedCat` *Prints first text file and appends into second file*

Description

This function is a generic function to print the first text and appends into second file if it exists.

Usage

```
appendedCat(firstText, secondText = as.character(NA))
```

Arguments

`firstText`	character, First text.
`secondText`	character, Second text.

Value

Appended text file.

Author(s)

Yong Luo

`areaProportion` *This function is to derive a correction index to account for edge effect when account for competition effect*

Description

The correction index is calculated using proportion of overlapped area to full circular area.

Usage

```
areaProportion(
  bearing,
  distance,
  radius,
  baseShape = "circle",
  baseRadius = 10,
  baseCorners = list(c(-50, 50), c(50, 50), c(50, -50), c(-50, -50))
)
```

Arguments

`bearing`	numeric, The bearing of a tree from a given point of plot (centre or corner).
`distance`	numeric, The distance of a tree from a given point of plot (centre or corner).
`radius`	numeric, The radius for a focal subject, which define the circular area around a subject tree.
`baseShape`	character, The shape of the area to be overlapped. Must be either `circle` or `rectangle`. Default is `circle`.
`baseRadius`	numeric, The radius for the base area, if the shape is defined as `circle`. Default is 10.
`baseCorners`	list, If the shape is defined as `rectangle`, this argument specifies upper left, upper right, lower right and lower left corners. Default is `list(c(-50, 50), c(50, 50), c(50, -50), c(-50, -50))`, which represent a 10000 m2 base area.

Value

A ratio of overlapped area to full circular area.

Author(s)

Yong Luo

Examples

```
## Not run:
# given a tree is located with bearing of 150 degree and distance of 13 m
# and in a plot of 16.9 m radius circle, in which all the trees are measured.
# assume all the trees within 10m radius have competitive effect on this tree
# hence the trees that are in the plot and within 10m radius from focal tree
# should be corrected based on proportion
# to calculate the proportion
proportion <- areaProportion(bearing = 150,
                             distance = 13,
                             radius = 10,
                             baseShape = "circle",
                             baseRadius = 16.9)




## End(Not run)
```

`BEC2IC` *Group BEC zones into interior and coastal region*

Description

It groups the BC BEC zone into two regions: coastal region `C` and interior region `I`.

Usage

```
BEC2IC(BEC)

## S4 method for signature 'character'
BEC2IC(BEC)
```

Arguments

`BEC`	character, BC BEC zone(s)

Value

grouped region by bec zone, in which `C` stands for coastal region, `I` stands for interior region and ? stands for unknown region.

Author(s)

Yong Luo

`biomassCalculator` *This function is to calculate aboveground biomass for boreal species based on DBH or DBH/Height*

Description

This function is to calculate aboveground biomass for boreal species based on DBH or DBH/Height

Usage

```
biomassCalculator(
  species,
  DBH,
  heightIncluded = TRUE,
  height,
  paperSource = "Lambert2005"
)
```

Arguments

`species`	Character string. The species name.
`DBH`	Numeric. The tree's diameter at breast height (DBH, cm).
`heightIncluded`	Logical. Whether the biomass is calculated based on DBH and height. If TURE, height must be provided. Default `TRUE`
`height`	Numeric. The tree's height (m).
`paperSource`	Character. Determine the sources of equations. Currently, this functions has two options, i.e., "Lambert2005" and "Ung2008". Default `Lambert2005`

Value

Biomass (kg) and missedSpecies list that was not calculated.

Author(s)

Yong Luo

Examples

```
## Not run:
 DBH <- seq(1, 100, 5)
 species <- c(rep("jack pine", 10), rep("black spruce", 10))
 species[1] <- "wrongSpecies"
 height <- seq(20, 40, length = 20)
 # without height information and taking the eqations from Lambert 2005
 biomass1 <- biomassCalculator(species = species, DBH = DBH, heightIncluded = FALSE)
 # with height information and taking the eqations from Lambert 2005
 biomass2 <- biomassCalculator(species = species, DBH = DBH,
                               heightIncluded = TRUE, height = height)

## End(Not run)
```

checkLD_remeas *Check the live and dead status of a remeasured subject*

Description

This function is to check the live and deas status for a remeasured subject.

Usage

```
checkLD_remeas(subjectID, measNo, LDStatus, liveCode, deadCode)
```

Arguments

`subjectID`	character, Specifies subject ID, such as a tree id.
`measNo`	numeric, Measurement number with bigger value indicates a later measurement.
`LDStatus`	character, Live and dead status for each remeasurement.
`liveCode`	character, Code for live status.
`deadCode`	character, Code for dead status.

Value

A data table that contains pass information. TRUE indicates pass, while FALSE indicates failure.

Author(s)

Yong Luo

`checkMissing_remeas` *Check the missing measurements in repeatedly measured subject against intended measurements*

Description

This function is to check missing measurements in repeatedly measured subject against intended measurements. Note that this function allows the regeneration, which means there may be missing measurements at the begining of intended masurements. Additionally, the function may allow the missing measurements after a subject died, depending on how `deadCode` is specified.

Usage

```
checkMissing_remeas(
  subjectID,
  measNo,
  intendedMeasNo,
  deadCode = NULL,
  LDStatus
)
```

Arguments

`subjectID`	character, Specifies subject ID, such as a tree id.
`measNo`	numeric, Measurement number with bigger value indicates a later measurement.
`intendedMeasNo`	numeric, The measurement number that a subject is intended measured.
`deadCode`	character, The code indicates the subject is dead. This arguement serves two purposes: 1) switch the function whether check a subject under dead scenario (i.e., the missing measurement before last intended measurement). Setting `NULL` will turn off dead mode, arguement `LDStatus` will not be used in the function. 2) if the dead mode is turn on (i.e., anything but `NULL`), this term determines which code will be used for dead. In this case, arguement `LSStatus` must be specified. By default, this term is set as `NULL`.
`LDStatus`	character, Live or dead status. This arguement is called only when `deadCode` is not set as `NULL`.

Value

A data table that contains pass information. TRUE indicates pass, while FALSE indicates failure. The table also contains the missing measurements.

Author(s)

Yong Luo

checkSize_remeas *Check the size change of a remeasured subject*

Description

This function is to check the size change for a remeasured subject.

Usage

```
checkSize_remeas(
  subjectID,
  measTime,
  size,
  change = "increase",
  maxChangeRate = NULL,
  toleranceMethod = "both",
  toleranceAbs = 0,
  toleranceRel = 0
)
```

Arguments

`subjectID`	character, Specifies subject ID, such as a tree id.
`measTime`	numeric, Measurement number with bigger value indicates a later measurement.
`size`	numeric, Measurement of an attribute.
`change`	character, Change direction either from `increase` or `decrease`. Default is `increase`.
`maxChangeRate`	numeric, It determines the maximum change rate. If the change rate from previous to current measurement exceeds the maximum change rate, then the pass of current measurement will be flagged as `FALSE`. If missing, this term is set as `NULL`.
`toleranceMethod`	character, Method to allow acceptable measurement error in an opposite direction of change argument. It must be either `both` (break both absolute and relative tolerance), `either` (break either absolute or relative tolerance), `absolute` (break absolute tolerance only), or `relative` (break relative tolerance only). Default is `both`.
`toleranceAbs`	numeric, Absolute tolerance value (exclusive) to allow measurement error. It must be a a non-negative value. If the change is `increase`, the change from current measurement to last measurement will be compared to the negative tolerance value, and vice versa. Default is `0` for zero tolerance.
`toleranceRel`	numeric, Relative tolerance value (exclusive) to allow measurement error. It must be a a non-negative value. If the change is `increase`, the change from current measurement to last measurement will be compared to the negative tolerance value, and vice versa. Default is `0` for zero tolerance.

Value

A data table that contains pass information. TRUE indicates pass, while FALSE indicates failure.

Author(s)

Yong Luo

`convertSI2SI` *Site index conversion from one species to another species*

Description

This function takes a known species and its site index and convert the site index for the species for a given bec zone

Usage

```
convertSI2SI(
  spCode_to,
  BECZone = as.character(NA),
  BECSubZone = as.character(NA),
  availableSI
)
```

Arguments

`spCode_to`	character, The species code the site index conversion is for. This is not case sensitive.
`BECZone`	character, The BEC zone the conversion takes place.
`BECSubZone`	character, The sub BEC zone the conversion takes place.
`availableSI`	character, Contains available site index for given species. Must be provided like `"at=10, bc=25"`.

Value

either valid site index or NA

Note

The site index conversion functions were taken from 1) sindex.dll (primarily), 2) expert knowledge of Gord Nigh and Dave Waddell The original functions were written by Dave Waddell, and can be found at https://github.com/bcgov/FAIB_PSPL/blob/main/05_site_index_conversions/site_index_conversion_equations_
The function derives site index for target species based on priority orders of the available site index specifically, for at: sw>sx>se>bl>pl>fd ba: hw>ss>sx>cw>fd>bg(one_to_one)>bl(one_to_one) bl: sw>sx>pl>at>fd>lw>sb>ba(one_to_one)>bg(one_to_one) cw: hw>ba>ss>sx>fd>sw fd(coastal): hw>cw>ba>ss>sx fd(interior): pl>bl>hw>sw>sx>at>lw>sb hw(coastal): cw>ba>ss>sx>fd>hm(one_to_one) hw(interior):fd>sw>sx>pl>lw
lw: fd>bl>pl>sw>sx>hw(interior)>sb>lt(one_to_one) pl: sw>sx>hw(interior)>at>bl>fd(interior)>lw>sb>ss(interior)>pa
sb: pl>lw>fd(interior)>bl>sw>sx ss: hw(coastal)>ba>cw>fd(coastal)>sx(one_to_one)>pw(one_to_one)>fd(one_to_one
sw: pl>at>hw(interior)>bl>lw>fd(interior)>sb>se(one_to_one)>sx(one_to_one) sx(coastal): hw>ba>cw>fd sx: pl>at>hw(interior)>bl>lw>fd(interior)>sb>sw(one_to_one) se: sw(one_to_one) bg: ba pw: ss(coastal)>hw(coastal)>sw>fd(one_to_one) lt: lw(one_to_one) hm: hw pa: pl dr: fd(coastal)>hw(coastal) py: fd

Author(s)

Yong Luo

Examples

```
## Not run:
# convert sw to at
at_si <- convertSI2SI(spCode_to = "AT",
                      BECZone = "unknown", # for some species, BEC zone must be provided
                      BECSubZone = "unknown",
                      availableSI = "sw = 30, pl = 50, fd = 20")
print(at_si) # 32.8353
# convert ta to fd in CWH, for which the conversion equation can not be found
py_si <- convertSI2SI(spCode_to = "py",
                           BECZone = "CWH", # for some species, BEC zone must be provided
                              BECSubZone = "unknown",
                              availableSI = "fd = 30")
print(py_si) # 30

## End(Not run)
```

DBHClassifier *Derive DBH class from DBH*

Description

This function derives DBH classes based on DBH. This function is equivalent to dbh_cl.sas macro.

Usage

```
DBHClassifier(DBH, classInterval = 5, maxDBH = 175)
```

Arguments

`DBH`	numeric, Tree DBH.
`classInterval`	numeric, The interval that used to categorize the DBH. If missing 5 cm is used.
`maxDBH`	numeric, Upper class limit. DBH that surpasses this limit is groupped in at this limit. If missing 175 is used.

Value

Classified DBH

Author(s)

Yong Luo

`DIB_ICalculator` *Calculate the inside-bark diameter at a given height*

Description

This function uses taper equation to calculate diameter inside bark at a given height. It is equivalent to the subroutine of vol_tree_active_equation in vol_setup macro

Usage

```
DIB_ICalculator(
  taperEquationForm,
  FIZorBEC,
  species,
  height_I,
  heightTotal,
  DBH,
  volMultiplier
)

## S4 method for signature
## 'character,character,character,numeric,numeric,numeric,numeric'
DIB_ICalculator(
  taperEquationForm,
  FIZorBEC,
  species,
  height_I,
  heightTotal,
  DBH,
  volMultiplier
)
```

Arguments

`taperEquationForm`	character, Specifies which taper equations will be used, currently support `KBEC` or `KFIZ3`. `KBEC` is the Kozak's equations (2002 version) based on BEC zone, tree sizes and species. `KFIZ3` is the equations based on forest inventory zone (FIZ), tree sizes and species. Default is KBEC, if missing.
`FIZorBEC`	character, Specifies which FIZ or BEC (depends on `taperEquationForm`) zones the tree located in BC.
`species`	character, Species code.
`height_I`	numeric, Height from ground.
`heightTotal`	numeric, Total height of a tree.
`DBH`	numeric, Diameter at breast height.
`volMultiplier`	numeric, Volume adjustment multiplier.

Value

Diameter inside bark

Author(s)

Yong Luo

`getSpatial` *Generic function to derive BEC, TSA and FIZ for given locations*

Description

This function is to derive `BEC`, `TSA` or FIZ based on an UTM location and BEC map.

Usage

```
getSpatial(pointID, zone, northing, easting, spatialMap, spatialAttribute)
```

Arguments

`pointID`	character, Data point ID.
`zone`	integer, UTM zone.
`northing`	integer, UTM northing.
`easting`	integer, UTM easting.
`spatialMap`	SpatialPolygonsDataFrame or sf, Spatial map. The spatial maps are from BC Data catalogue website. You can obtain these maps using bcdata package.
`spatialAttribute`	character, Specifies which spatial attribute to be obtained. Must be one of "BEC", "TSA", "FIZ", "TFL" and "OWNERSHIP", regardless of lower or upper cases. Must be consistent with `spatialMap` arguement.

Value

Depends on what spatial attribute a function derives. For `BEC`, a table that contains:

- bec_zone bec zone.
- bec_sbz bec subzone.
- bec_var bec variant

For `TSA`, a table that contains:

- tsa tsa.
- tsa_desc tsa descriptions.

For `FIZ`, a table that contains:

- fiz fiz forest inventory zone.

For `TFL`, a table that contains:

- tfl tfl timber farm licences.
- tfl_licencee tfl timber farm licencee.

For `OWNERSHIP`, a table that contains:

- owner owner of land.
- schedule schedule.

Author(s)

Yong Luo

Examples

```
## Not run:
 ## for Prince Rupert, Fort Nelson, Prince George, Victoria, Kelowna
 citylocs <- data.frame(point_ID = c("Prince Rupert", "Prince George", "Victoria", "Kelowna"),
                        zone = c(9, 10, 10, 11),
                        northing = c(6019079.41, 5974323.27, 5361626.96, 5528467),
                        easting = c(415075.83, 516441.65, 475594.70, 321996.76))
 tsamap <- bcmaps::tsa(class = "sp")
 city_tsa <- getSpatial(pointID = citylocs$point_ID,
                        zone = citylocs$zone,
                        northing = citylocs$northing,
                        easting = citylocs$easting,
                        spatialMap = tsamap,
                        spatialAttribute = "TSA")
 print(city_tsa)
 #         pointID tsa          tsa_desc
 #   Prince Rupert  46     GBR North TSA
 #   Prince George  24 Prince George TSA
 #        Victoria  38    Arrowsmith TSA
 #         Kelowna  22      Okanagan TSA

 becmap <- bcmaps::bec(class = "sp")
 city_bec <- getSpatial(pointID = citylocs$point_ID,
                        zone = citylocs$zone,
                        northing = citylocs$northing,
                        easting = citylocs$easting,
                        spatialMap = becmap,
                        spatialAttribute = "bec")
 print(city_bec)
 #        pointID bec_zone bec_sbz bec_var
 #  Prince Rupert      CWH      vh       2
 #  Prince George      SBS      mh    <NA>
 #       Victoria      CDF      mm    <NA>
 #        Kelowna       PP      xh       1

## End(Not run)
```

HegyiCICalculator *the function to calculate intraspecific and interspecific hegyi competition index both distance and size*

Description

the function to calculate intraspecific and interspecific hegyi competition index both distance and size

Usage

```
HegyiCICalculator(
  objectID,
  species,
  coordX,
  coordY,
  size,
  maxRadius,
  distanceWeight = 1,
  sizeWeight = 0
)
```

Arguments

`objectID`	character, The unique object identifier. Must be unique.
`species`	character, Species code to identify intra and inter-specific competition.
`coordX`	numeric, The x coordinate.
`coordY`	numeric, The y coordinate.
`size`	numeric, The size that used for compute competition index.
`maxRadius`	numeric, The competition index will been calculated within this radius
`distanceWeight`	numeric, Define how the compeition sensitive to the distance of a neighbours, ie., crowdness. Default is 1, which is same as the original Hegyi index.
`sizeWeight`	numeric, Define how the compeition scales across all the plots. Default is 0, which means there is no scale.

Value

a data table that has five columns, plotNumber, treeNumber, Year, IntraH and InterH

Note

no note

Author(s)

Yong Luo

See Also

no

`heightEstimateForBTOP_D` *Estimate tree height for a broken top tree when DBH, inside bark diamater at broken top height, height at broken are available*

Description

This is the second function to estimate a tree's height for a broken top tree. A tree's height is esimated using height of the broken top (`heightBTOP`), inside bark diameter at broken height (`DIBBTOP`) and `DBH`. Specifically, this function guesses the tree height, computes inside bark diameter at broken height (`heightBTOP`) using a taper equation, compares it to an observed inside bark diameter and chooses the tree height that has closest value of inside bark diameter at broken. For the broken top trees that have field projected height, total tree height also can be estimated using `heightEstimateForBTOP_H`.

Usage

```
heightEstimateForBTOP_D(
  heightBTOP,
  DIBBTOP,
  DBH,
  taperEquationForm = "KBEC",
  FIZorBEC,
  species,
  volMultiplier = 1,
  SASOriginal = FALSE
)
```

Arguments

`heightBTOP`	numeric, Height of the broken top.
`DIBBTOP`	numeric, Diameter inside bark at the height of the broken top.
`DBH`	numeric, DBH of the tree, Must be given when BTOP is D.
`taperEquationForm`	character, Specifies which taper equaiton will be used to estimate tree height, currently supports KBEC, KBECQCI, KFIZ. If missing, the function uses KBEC as default.
`FIZorBEC`	character, Specifies which FIZ or BEC (depends on taperEquationForm) zones the tree located.
`species`	character, Tree species.
`volMultiplier`	numeric, Volume adjustment. If missing, 1 will be used.
`SASOriginal`	logical, Specifies whether the original sas algrithm will be used for guess tree height If missing, FALSE will be used.

Value

Total tree height

Author(s)

Yong Luo

See Also

`heightEstimateForBTOP_H`

`heightEstimateForBTOP_H` *Estimate tree height for a broken top tree when projected tree height is available*

Description

This function is to esimate a broken top tree's height based on projected tree height in the field (`heightProjected`). For the broken top trees that have diameter at broken and broken top trees, total tree height also can be esimated using `heightEstimateForBTOP_D`.

Usage

```
heightEstimateForBTOP_H(heightProjected)
```

Arguments

`heightProjected`	numeric, Projected tree height in the field, must be non-NA value.

Value

Total tree height

Author(s)

Yong Luo

See Also

`heightEstimateForBTOP_D`

`lm_group` *Extended lm function by adding group functionality*

Description

A generic function by adding grouping functionality in `lm` function.

Usage

```
lm_group(formula, data, groupBy, ...)

## S4 method for signature 'character,data.table,character'
lm_group(formula, data, groupBy, ...)

## S4 method for signature 'character,data.table,missing'
lm_group(formula, data, groupBy, ...)
```

Arguments

`formula`	character, Linear model formula.
`data`	data.table, The data used for the models.
`groupBy`	character, Specifies variables that used for the group.
`...`	see `lm` for the rest arguments.

Value

A list of regression analyses results

Author(s)

Yong Luo

See Also

`lm`

`merge_dupUpdate` *Merge table and update values for duplicate column*

Description

This is an extended function for `merge` function by updating values for duplicate column for the first, second or both tables.

Usage

```
merge_dupUpdate(x, y, by, updateDup, ...)

## S4 method for signature 'data.table,data.table,character,logical'
merge_dupUpdate(x, y, by, updateDup, ...)

## S4 method for signature 'data.table,data.table,character,missing'
merge_dupUpdate(x, y, by, updateDup, ...)
```

Arguments

`x`	data.table, The first table for merging.
`y`	data.table, The second table for merging.
`by`	character, The key to merge two tables.
`updateDup`	logical, Specifies whether update duplicate column in merged table when its information is available in y table, which means update from the second table. If missing, the function takes `TRUE`.
`...`	see `merge` for rest of arguments.

Value

A merged table without duplicate columes. A warning message is given if the duplicate column has different values.

Author(s)

Yong Luo

See Also

`merge`

PHFCalculator — *Calculate tree per ha factor for both fix and variable area plot*

Description

Calculates tree per ha factor for both fix and variable area plots.

Usage

```
PHFCalculator(sampleType, blowUp, treeWeight, plotWeight, treeBasalArea)
```

Arguments

`sampleType`	character, Specifies how the plot is sampled among fixed area plot or variable area plot, must be either V for variable area plot or F for fixed area plot.
`blowUp`	numeric, Specifies the blowup factor. For fixed area plot, it is calculated as 1/plotarea. For variable area plot, it is basal area factor (BAF).
`treeWeight`	numeric, Specifies whether a tree is zero counted (tree is out), one time counted (regular count) or two times counted (double counted) in the walk through sampling protocal.
`plotWeight`	numeric, Specifies how a plot is measured, i.e., full plot measured (valued as 1), half plot measured (valued as 2) or quarter plot measured (valued as 4).
`treeBasalArea`	numeric, When plot is measured using variable area plot, this value must be given, otherwise, can be missing

Value

Tree per ha factor

Author(s)

Yong Luo

randomStemMapping — *The function is to generate stem mapping using random spatial distribution of stratified size group for non-stem-mapped trees.*

Description

The function is to generate stem mapping using random spatial distribution of stratified size group for non-stem-mapped trees.

Usage

```
randomStemMapping(objectID, size, noofGroup = 5, plotSize, mapSize = 10000)
```

Arguments

`objectID`	character, The unique object identifier. Must be unique.
`size`	numeric, The size that used for compute competition index.
`noofGroup`	numeric, Defines how many groups to distribute objects. Default is 5.
`plotSize`	numeric, The plot size for the objects.
`mapSize`	numeric, The map size to distribute the objects. Default is 10000 m2.

Value

a data table that has five columns, plotNumber, treeNumber, Year, IntraH and InterH

Note

no note

Author(s)

Yong Luo

See Also

no

`SIInBC` *Derive site index for a given spatial coverage or a spatial point*

Description

This function is to derive species' site index for a given spatial coverage or spatial points based on BC provincial species productivity maps.

Usage

```
SIInBC(SIMapPath, spatialCoverage, species = "all", returnClass = "table")
```

Arguments

`SIMapPath`	character, Specifies folder location of species index maps. Please request all the maps from author and save them into your target folder. The function only supports `TIFF` format. Currently those maps were converted from BC Data catalogue.
`spatialCoverage`	spatialPolygons or spatialPoints, Specifies spatial polygons or spatial points that need to intersect.
`species`	character, Must be one or some of 22 species.
`returnClass`	character, Specifies the class you intended to return from either `sp` or `table`. If missing, `table` will be used.

Value

the returned value depends on `returnClass` arguement and class of `spatialCoverage` arguement. If `returnClass` is set as `table`, a table will be returned. If `returnClass` is set as `sp`, a raster layer will be returned for SpatialPolygons* objects, while a SpatialPointDataframe will be returned for SpatialPoints* objects.

Author(s)

Yong Luo

`standardizeSpeciesName` *Standardize species name from different forest inventory data, this function to make all the species compatible to biomassCalculation function*

Description

Standardize species name from different forest inventory data, this function to make all the species compatible to biomassCalculation function

Usage

```
standardizeSpeciesName(speciesTable, forestInventorySource)

## S4 method for signature 'data.table,character'
standardizeSpeciesName(speciesTable, forestInventorySource)
```

Arguments

`speciesTable`	data table. It must at least have one column species
`forestInventorySource,`	Character string. Give the forest inventory data source Currently support MBPSP, MBTSP, ABPSP, BCPSP, SKPSP, SKTSP and NFIPSP

Value

a data tables, the first one contains successfully standardized species. the newSpeciesName is the standardized name, unknown means the species in the original species table can not be found according to manual

Note

no note

Author(s)

Yong Luo

See Also

no

`stemMapping` *Map stems in a plot.*

Description

The function is to map all the stems in a plot based on bearing and distance.

Usage

```
stemMapping(
  objectID,
  bearing,
  distance,
  plotShape = as.character(NA),
  plotSize,
  distanceUnit = "m",
  outputFormat,
  showID = TRUE
)
```

Arguments

`objectID`	character, A object's ID, e.g., a tree's ID.
`bearing`	numeric, Azimuth of a object from the north. It should be between 0 to 360.
`distance`	numeric, Distance between a object and the centre of the circle.
`plotShape`	character, circle, Rectangle or NA. NA means no plot shape information is not available and and no plot geometry will be produced.
`plotSize`	numeric, Defines the plot geoimetry. If plotShape is circular, a number must be provided as plot radius. If plotShape is rectangle, two numbers must be provided to determine length and width of a plot.
`distanceUnit`	character, Defines the unit in the stem mapping. Default is `m`.
`outputFormat`	character, Defines whether a `table` or a `figure` will be returned from this function.
`showID`	logical, Specifies whether the objectID will be showed in outputed figure. Default is `TRUE`.

Value

1. A table contains the x and y for all the objects and the plot boundary. 2) a ggplot object will be returned for visualization.

Note

In the figure, IPC is the integrated plot center with x = 0 and y = 0.

Author(s)

Yong Luo

Examples

```
## Not run:
## randomly generate some trees
bigplottrees <- data.table(tree_id = 1:20,
                           angle = runif(20, min = 0, max = 360),
                           distance = runif(20, min = 0, max = 11.28))
## output a table
stemmapped_table <- stemMapping(objectID = bigplottrees$tree_id,
                                         bearing = bigplottrees$angle,
                                         distance = bigplottrees$distance,
                                         plotShape = "circle",
                                         plotSize = 11.28,
                                         distanceUnit = "m",
                                         outputFormat = "table")

## output a figure
stemmapped_figure <- stemMapping(objectID = bigplottrees$tree_id,
                                         bearing = bigplottrees$angle,
                                         distance = bigplottrees$distance,
                                         plotShape = "circle",
                                         plotSize = 11.28,
                                         distanceUnit = "m",
                                         outputFormat = "figure",
                                         showID = TRUE)

## output a figure without a plot
stemmapped_figure <- stemMapping(objectID = bigplottrees$tree_id,
                                         bearing = bigplottrees$angle,
                                         distance = bigplottrees$distance,
                                         outputFormat = "figure",
                                         showID = TRUE)


## End(Not run)
```

`stemMappingExtension` *Extend stem mapping to a target area and shape.*

Description

The function cuts a circle stem map into a largest hexagon in circle and extends this hexagon to a target area and shape. This is a generic function.

Usage

```
stemMappingExtension(
  objectID,
  bearing,
  distance,
  plotRadius = 11.28,
  targetArea = 1,
  targetShape = "square",
  randomRotate = FALSE,
  randomSeed = as.numeric(NA)
)
```

Arguments

`objectID`	character, A object's ID, e.g., a tree's ID.
`bearing`	numeric, Azimuth of a object from the north. It should be between 0 to 360.
`distance`	numeric, Distance between a object and the centre of the circle.
`plotRadius`	numeric, Radius of the plot circle. If missing, `11.28` will be used, as it presents the radius of a circle for big trees (i.e., trees DBH >= 9).
`targetArea`	numeric, Defines the area you may want to extend. The unit of this input is `ha`. Default is 1 ha.
`targetShape`	character, Defines the shape of the target area. It currently supports `circle` and `square`. The default is `square`.
`randomRotate`	logical, Defines whether need to random rotate the hexagon when merge into a `targetArea`. The default is `FALSE`.
`randomSeed`	numeric, Defines random seed for the random number generator. This arguement is called when `randomRotate` is TRUE. If missing, NA is used, suggesting no random seed is set.

Value

A table contains the x and y for all the objects in the extended area.

Author(s)

Yong Luo

Examples

```
## Not run:
## randomly generate some trees
library(data.table)
smallplottrees <- data.table(expand.grid(angle = seq(0, 360, 1),
                             distance = seq(0.1, 5.6, 0.1)))
smallplottrees[, tree_id := 1:nrow(smallplottrees)]
## extend it to 1 ha
treelist_smallplot <- stemMappingExtension(objectID = smallplottrees$tree_id,
                                           bearing = smallplottrees$angle,
                                           distance = smallplottrees$distance,
                                           plotRadius = 5.64,
                                           randomRotate = TRUE)
smallplottrees[tree_id %in% treelist_smallplot$objectID,
              inHexigon := "Yes"]
smallplottrees[is.na(inHexigon),
              inHexigon := "No"]
smallplottrees[,':='(x = sin(angle*pi/180)*distance,
                     y = cos(angle*pi/180)*distance)]
library(ggplot2)
trees_inplot <- ggplot(data = smallplottrees, aes(x = x, y = y))+
 geom_point(aes(col = inHexigon))
trees_all <- ggplot(data = treelist_smallplot, aes(x = x, y = y))+
 geom_point(aes(col = hexagonID))



bigplottrees <- data.table(tree_id = 1:20,
                           angle = runif(20, min = 0, max = 360),
                           distance = runif(20, min = 0, max = 11.28))
## extend it to 1 ha
treelist_bigplot <- stemMappingExtension(objectID = bigplottrees$tree_id,
                                         bearing = bigplottrees$angle,
                                         distance = bigplottrees$distance,
                                         plotRadius = 11.28)
treelist_smallplot[, source := "smallplot"]
treelist_bigplot[, source := "bigplot"]


alltreelist <- rbind(treelist_bigplot, treelist_smallplot)


alltreeplot <- ggplot(data = alltreelist, aes(x, y))+
 geom_point(aes(col = factor(source)))+
 geom_point(data = alltreelist[hexagonID == 0,],
 aes(x, y), col = "red")


## End(Not run)
```

---

stemMappingExtension_square

Extend stem mapping to a target area and shape.

Description

The function extends stem mapping in a square plot to a target area and shape.

Usage

```
stemMappingExtension_square(
  objectID,
  bearing,
  distance,
  plotLength,
  targetArea = 1,
  targetShape = "square",
  randomRotate = FALSE
)
```

Arguments

`objectID`	character, A object's ID, e.g., a tree's ID.
`bearing`	numeric, Bearing of a object from the north. It should be between 0 to 360.
`distance`	numeric, Distance between a object and the centre of the circle.

`plotLength`	numeric, Length of a square plot.
`targetArea`	numeric, Defines the area you may want to extend. The unit of this input is `ha`. Default is 1 ha.
`targetShape`	character, Defines the shape of the target area. It currently supports `circle` and `square`. The default is `square`.
`randomRotate`	logical, Defines whether need to random rotate the hexagon when merge into a `targetArea`. The default is `FALSE`.

Value

A table contains the x and y for all the objects in the extended area.

Author(s)

Yong Luo

Examples

```
## Not run:
## randomly generate some trees
library(data.table)
smallplottrees <- data.table(tree_id = 1:20,
                             angle = runif(20, min = 0, max = 360),
                             distance = runif(20, min = 0, max = 5.6))
## extend it to 1 ha
treelist_smallplot <- stemMappingExtension_square(objectID = smallplottrees$tree_id,
                                          bearing = smallplottrees$angle,
                                          distance = smallplottrees$distance,
                                          radius = 5.64,
                                          randomRotate = TRUE)


bigplottrees <- data.table(tree_id = 1:20,
                           angle = runif(20, min = 0, max = 360),
                           distance = runif(20, min = 0, max = 11.28))
## extend it to 1 ha
treelist_bigplot <- stemMappingExtension_square(objectID = bigplottrees$tree_id,
                                        bearing = bigplottrees$angle,
                                        distance = bigplottrees$distance,
                                        radius = 11.28)
treelist_smallplot[, source := "smallplot"]
treelist_bigplot[, source := "bigplot"]


alltreelist <- rbind(treelist_bigplot, treelist_smallplot)


alltreeplot <- ggplot(data = alltreelist, aes(x, y))+
 geom_point(aes(col = factor(source)))


## End(Not run)
```

`taperCoeffsGenerator` *Generate the coefficients table of taper equations*

Description

Generates the coefficients of the taper equations for based on specific taper equation form (`taperEquationForm`)

Usage

```
taperCoeffsGenerator(taperEquationForm)

## S4 method for signature 'character'
taperCoeffsGenerator(taperEquationForm)

## S4 method for signature 'missing'
taperCoeffsGenerator()
```

Arguments

`taperEquationForm`	character, Specifies a taper equation form one of KBEC, KBECQCI, KFIZ3.

Value

A coeffients table

Author(s)

Yong Luo

`taperImplementor` *Implement taper equation for a given tree*

Description

Implement taper equation for a given tree

Usage

```
taperImplementor(
  taperEquationForm,
  taperCoeffs,
  FIZorBEC,
  species,
  height_I,
  heightTotal,
  DBH,
  volMultiplier
)
```

```
## S4 method for signature
## 'character,
##   data.table,
##   character,
##   character,
##   numeric,
##   numeric,
##   numeric,
##   numeric'
taperImplementor(
  taperEquationForm,
  taperCoeffs,
  FIZorBEC,
  species,
  height_I,
  heightTotal,
  DBH,
  volMultiplier
)
```

Arguments

`taperEquationForm`	character, Specifies a taper equation form one of KBEC, KBECQCI, KFIZ3.
`taperCoeffs`	data.table, Table that stores the coefficients that match the taper equation.
`FIZorBEC`	character, FIZ or BEC.
`species`	character, Species code.
`height_I`	numeric, Height from ground.
`heightTotal`	numeric, Total height of a tree.
`DBH`	numeric, Diameter at breast height.
`volMultiplier,`	Volume multiplier adjustment.

Value

DIB_I diameter inside bark at height_I

Note

This function is inside of the VRIVolTree function

Author(s)

Yong Luo

`treeProfile` *Calculate volume for trees*

Description

This function is to produce a tree trunk profile (i.e., inside bark diameter (`DIB`)). And summarize the whole stem volume (`VOL_WSV`) and merchantable volume (`VOL_MER`).

Usage

```
treeProfile(
  taperEquationForm = "KBEC",
  FIZorBEC,
  species,
  height,
  DBH,
  stumpHeight = 0.3,
  breastHeight = 1.3,
  UTOPDIB = 10,
  BTOPHeight = NA
)
```

Arguments

`taperEquationForm`	character, Specifies which taper equations will be used, currently support `KBEC` or `KFIZ3`. `KBEC` is the Kozak's equations (2002 version) based on BEC zone, tree sizes and species. `KFIZ3` is the equations based on forest inventory zone (FIZ), tree sizes and species. Default is KBEC, if missing.
`FIZorBEC`	character, Specifies which FIZ or BEC (depends on taperEquationForm) zones the tree located in BC.
`species`	character, Tree species, must be BC species code.
`height`	numeric, Total tree height in meter.
`DBH`	numeric, DBH of the tree in cm.
`stumpHeight`	numeric, Defines stump height. If missing, 0.3 m is used.
`breastHeight`	numeric, Defines the breast height. If missing, 1.3 m is used.
`UTOPDIB`	numeric, Merchantable inside-bark diameter. If missing, UTOP is 10.
`BTOPHeight`	numeric, Height at broken top.

Value

A volume table

Note

For the volume between 0 and 0.3, also known as stump volume, the compiler calculates the volume as cylinder with the diameter of stump height. In the case of the diameter at stump height is less than diameter at breast height, the diameter at breast height is used as stump height. It calculates tree volume based on a 10 cm slices starting from 0.3 m tall using Smalian's formula.

Author(s)

Yong Luo

See Also

treeVolume

Examples

```
## Not run:
treeprofile_a <- treeProfile(FIZorBEC = "CWH",
                             species = "H",
                             height = 27.4,
                             DBH = 30.7,
                             BTOPHeight = 5.6)
treeprofile_b <- treeProfile(FIZorBEC = "CWH",
                             species = "S",
                             height = 37.3,
                             DBH = 42.3)
treeprofile_c <- treeProfile(FIZorBEC = "CWH",
                             species = "H",
                             height = 11.6,
                             DBH = 11.2)


## End(Not run)
```

`treeVolCalculator` *Calculate volume for trees in the ISMCCompiler context*

Description

This function is to calculate tree volume using taper equations on a basis of 10 cm slice. As default, the function is to calculate whole tree volume (`VOL_WSV`), total merchantable volume (`VOL_BELOW_UTOP`) and non-merchantable volume (`VOL_ABOVE_UTOP`) based on `FIZorBEC`, `species`, `height`, `DBH` using Kozak BEC taper equations. The function also handles broken top trees by specifying `BTOPEstimateType`, `BTOPHeight` and `BTOPDIB`. Accordingly, `VOL_BELOW_BTOP` and `VOL_ABOVE_BTOP` are produced. Lastly, the function derives volume (denoted as `LOG_V_X`), merchantable volume (denoted as `LOG_VM_X`) and top inside bark diameter (denoted as `LOG_D_X`) for each log when the `logLengthMatrix` is provided. For all the scenarioes, stump height (`HT_STUMP`), inside bark diameter at stump height (`DIB_STUMP`), breast height (`HT_BH`), inside bark diameter at breast height (`DIB_BH`) are generated.

Usage

```
treeVolCalculator(
  FIZorBEC,
  species,
  height,
  DBH,
  taperEquationForm = "KBEC",
  volMultiplier = 1,
  stumpHeight = 0.3,
  breastHeight = 1.3,
  UTOPDIB = 10,
  BTOPEstimateType = NA,
  BTOPHeight = NA,
  BTOPDIB = NA,
  logLengthMatrix = data.table(Log1_L = numeric()),
  logMinLength = 3
)
```

Arguments

`FIZorBEC`	character, Specifies which FIZ or BEC (depends on taperEquationForm) zones the tree located in BC.
`species`	character, Tree species, must be BC species code.
`height`	numeric, Total tree height in meter.
`DBH`	numeric, DBH of the tree in cm.
`taperEquationForm`	character, Specifies which taper equations will be used, currently support `KBEC` or `KFIZ3`. `KBEC` is the Kozak's equations (2002 version) based on BEC zone, tree sizes and species. `KFIZ3` is the equations based on forest inventory zone (FIZ), tree sizes and species. Default is KBEC, if missing.
`volMultiplier`	numeric, Volume adjustment multiplier. If missing, 1 (no adjustment) is used.
`stumpHeight`	numeric, Defines stump height. If missing, 0.3 m is used.
`breastHeight`	numeric, Defines the breast height. If missing, 1.3 m is used.
`UTOPDIB`	numeric, Merchantable inside-bark diameter. If missing, UTOP is 10.
`BTOPEstimateType`	integer, Must among NA, 1, 2, 3. Defines whether a tree has broken top and which field observation (height at broken or DIB at broken) is used to define broken point. NA means that tree is not broken top. 1 and 3 means diameter at broken top is not available, height at broken top is used to define broken point. 2 means diameter at broken top is available and is used to define broken point. Default is NA: tree does not have broken top.
`BTOPHeight`	numeric, Height at broken top.
`BTOPDIB`	numeric, Diameter inside bark at height of broken top.
`logLengthMatrix`	data.table, Log length matrix. If missing, there is no log-level volume returned.
`logMinLength`	numeric, Minimum log length. This argument is activated when logLengthMatrix is provided.

Value

A volume table

Author(s)

Yong Luo

See Also

`DIB_ICalculator`

treeVolume *Calculate volume for trees*

Description

This function is to calculate tree volume such as the whole stem volume (`VOL_WSV`), merchantable volume (`VOL_MER`) and stump volume (`STUMP`).

Usage

```
treeVolume(
  taperEquationForm = "KBEC",
  FIZorBEC,
  species,
  DBH,
  height,
  BTOPHeight = NA,
  volumeName = "WSV",
  stumpHeight = 0.3,
  UTOPDIB = 10
)
```

Arguments

`taperEquationForm`	character, Specifies which taper equations will be used, currently support `KBEC` or `KFIZ3`. `KBEC` is the Kozak's equations (2002 version) based on BEC zone, tree sizes and species. `KFIZ3` is the equations based on forest inventory zone (FIZ), tree sizes and species. Default is KBEC, if missing.
`FIZorBEC`	character, Specifies which FIZ or BEC (depends on `taperEquationForm`) zones the tree located in BC.
`species`	character, Tree species, must be BC species code.
`DBH`	numeric, DBH of the tree in cm.
`height`	numeric, Total tree height in meter.
`BTOPHeight`	numeric, Height at broken top in meter. `NA` suggests no broken top. If missing, the default is `NA`.
`volumeName`	character, Indicates which volume you want to derive. It supports the whole stem volume (`WSV`), merchantable volume (`MER`) or stump volume (`STUMP`). The merchantable volume is the whole stem volume minus volume of stump and volume less than minimum utility diameter (`UTOPDIB`). If missing, the default is `WSV`.
`stumpHeight`	numeric, Defines stump height. If missing, 0.3 m is used. It will be called to calculate the merchantable volume.
`UTOPDIB`	numeric, Merchantable inside-bark diameter. If missing, UTOP is 10. It will be called to calculate the merchantable volume.

Value

volume

Note

For the volume between 0 and 0.3, also known as stump volume, the function calculates the volume as cylinder with the diameter of stump height. In the case of the diameter at stump height is less than diameter at breast height, the diameter at breast height is used as stump height. It calculates tree volume based on a 10 cm slices starting from 0.3 m tall using Smalian's formula.

Author(s)

Yong Luo

See Also

treeVolCalculator and treeProfile

Examples

```
## Not run:
treeA_wsv <- treeVolume(FIZorBEC = "CWH",
                        species = "H",
                        DBH = 30.7,
                        height = 27.4,
                        BTOPHeight = 5.6)
trees_wsv <- treeVolume(FIZorBEC = "CWH",
                        species = c("H", "S", "H"),
                        DBH = c(30.7, 42.3, 11.2),
                        height = c(27.4, 37.3, 11.6),
                        BTOPHeight = c(5.6, NA, NA))
trees_mer <- treeVolume(FIZorBEC = "CWH",
                        volumeName = "MER",
                        species = c("H", "S", "H"),
                        DBH = c(30.7, 42.3, 11.2),
                        height = c(27.4, 37.3, 11.6),
                        BTOPHeight = c(5.6, NA, NA))

## End(Not run)
```

UTM_Convertor *Convert UTM to other coordinate reference system.*

Description

Converts UTM coordinates to the other coordinate reference system.

Usage

```
UTM_Convertor(
  point_ID,
  zone,
  northing,
  easting,
  CRS_To =
   "+proj=aea +lat_0=45 +lon_0=-126 +lat_1=50 +lat_2=58.5 +x_0=1000000 +y_0=0 +datum=NAD83 +units=m
  class = "sf"
)
```

Arguments

`point_ID`	character, Data point ID.
`zone`	integer, UTM zone.
`northing`	integer, UTM northing.
`easting`	integer, UTM easting.
`CRS_To`	character, Defines the spatial coordination reference that you wish to transform. Default is BC Albers reference system.
`class`	character, Define the class of returned objective. Currently this function supports either `table` or `sf` class. Default is `table`.

Value

Reprojected objective.

Author(s)

Yong Luo

Examples

```
## Not run:
## for Prince Rupert, Fort Nelson, Prince George, Victoria, Kelowna
citylocs <- UTM_Convertor(point_ID = c("Prince Rupert", "Prince George",
                                       "Victoria", "Kelowna"),
                          zone = c(9, 10, 10, 11),
                          northing = c(6019079.41, 5974323.27, 5361626.96, 5528467.98),
                          easting = c(415075.83, 516441.65, 475594.70, 321996.76),
                          class = "sf")
bcbdry <- bcmaps::bc_bound(class = "sp")
plot(bcbdry)
plot(citylocs, col = "red", size = 10, add = TRUE)

## End(Not run)
```

Index

annualGrowthRateCalculator, 2
appendedCat, 3
areaProportion, 3

BEC2IC, 4
BEC2IC,character-method *(BEC2IC)*, 4
biomassCalculator, 5

checkLD_remeas, 6
checkMissing_remeas, 7
checkSize_remeas, 8
convertSI2SI, 9

DBHClassifier, 10
DIB_ICalculator, 11
DIB_ICalculator,character,character,character,numeric,numeric,numeric,numeric-method
    (DIB_ICalculator), 11

getSpatial, 12

HegyiCICalculator, 13
heightEstimateForBTOP_D, 14, *15*, *16*
heightEstimateForBTOP_H, *14*, *15*, 15

lm, *16*, *17*
lm_group, 16
lm_group,character,data.table,character-method
    (lm_group), 16
lm_group,character,data.table,missing-method
    (lm_group), 16

merge, *17*
merge_dupUpdate, 17
merge_dupUpdate,data.table,data.table,character,logical-method
    (merge_dupUpdate), 17
merge_dupUpdate,data.table,data.table,character,missing-method
    (merge_dupUpdate), 17

PHFCalculator, 18

randomStemMapping, 18

SIInBC, 19
standardizeSpeciesName, 20
standardizeSpeciesName,data.table,character-method
    (standardizeSpeciesName), 20
stemMapping, 21
stemMappingExtension, 22
stemMappingExtension_square, 24

taperCoeffsGenerator, 26
taperCoeffsGenerator,character-method
    (taperCoeffsGenerator), 26
taperCoeffsGenerator,missing-method
    (taperCoeffsGenerator), 26
taperImplementor, 26
taperImplementor,character,data.table,character,charact
    (taperImplementor), 26
treeProfile, 28, *32*
treeVolCalculator, 29, *32*
treeVolume, *29*, 31

UTM_Convertor, 32