Package 'medfate'

March 22, 2019

Type Package

Title Mediterranean Forest Simulation

Version 0.7.4

Date 2019-03-19

Description
Functions to simulate Mediterranean forest functioning and dynamics using cohort-based description of vegetation [De Caceres et al. (2015) <doi:10.1016/j.agrformet.2015.06.012>].

License GPL (>= 2)

URL http://vegmod.ctfc.cat/medfateweb

LazyLoad yes

Depends R (>= 3.4.0), sp

Imports spdep, GSIF, methods, meteoland (>= 0.7.1), Rcpp (>= 0.12.12)

Suggests knitr, rmarkdown, rjson

LinkingTo Rcpp, meteoland

Encoding UTF-8

NeedsCompilation yes

VignetteBuilder utils, knitr

Author Miquel De Cáceres [aut, cre],
Víctor Granda [aut],
Antoine Cabon [aut]

Maintainer Miquel De Cáceres <miquelcaceres@gmail.com>

Repository CRAN

Date/Publication 2019-03-22 14:00:06 UTC

R topics documented:

biophysics *Physical and biophysical utility functions*

Description

Set of functions used in the calculation of biophysical variables.

Usage

```
biophysics.leafTemperature(absRad, airTemperature, u, E, leafWidth = 1.0)
biophysics.radiationDiurnalPattern(t, daylength)
biophysics.temperatureDiurnalPattern(t, tmin, tmax, daylength)
```

Arguments

u	Wind speed above the leaf boundary layer (in m/s).
airTemperature	Air temperature (in ºC).
tmin, tmax	Minimum and maximum daily temperature (ºC).
absRad	Absorbed long- and short-wave radiation (in W·m-2).
E	Transpiration flow (in mmol H20·m-2·s-1) per one sided leaf area basis.
leafWidth	Leaf width (in cm).
t	Time of the day (in seconds).
daylength	Day length (in seconds).

Value

Values returned for each function are:

- biophysics.leafTemperature: leaf temperature (in ºC)
- biophysics.radiationDiurnalPattern: the proportion of daily radiation corresponding to the input time in seconds after sunrise.
- biophysics.temperatureDiurnalPattern: diurnal pattern of temperature assuming a sinusoidal pattern with T = Tmin at sunrise and T = (Tmin+Tmax)/2 at sunset.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Campbell, G. S., and J. M. Norman. 1998. AN INTRODUCTION TO ENVIRONMENTAL BIOPHYSICS.: 2nd edition. (eqns. 14.1 & 14.3)

B. Y. H. Liu and R. C. Jordan, "The interrelationship and characteristic distribution of direct, diffuse and total solar radiation," Solar Energy, vol. 4, no. 3, pp. 1–19, 1960.

McMurtrie, R. E., D. A. Rook, and F. M. Kelliher. 1990. Modelling the yield of Pinus radiata on a site limited by water and nitrogen. Forest Ecology and Management 30:381–413.

McMahon, T. A., M. C. Peel, L. Lowe, R. Srikanthan, and T. R. McVicar. 2013. Estimating actual, potential, reference crop and pan evaporation using standard meteorological data: a pragmatic synthesis. Hydrology & Earth System Sciences 17:1331–1363. See also: http://www.fao.org/docrep/x0490e/x0490e06.htm

See Also

spwb

---

conductancefunctions *Hydraulic confuctance functions*

Description

Set of functions used in the calculation of soil and plant hydraulic conductance.

Usage

```
hydraulics.psi2K(psi, Psi_extract, ws = 3.0)
hydraulics.K2Psi(K, Psi_extract, ws = 3.0)
hydraulics.averagePsi(psi, v, c, d)
hydraulics.vulnerabilityCurvePlot(x, soil = NULL, type="leaf",
                                  psiVec =  seq(-0.1, -8.0, by =-0.01),
                                  relative = FALSE, draw = TRUE)
hydraulics.psiCrit(c, d, pCrit = 0.001)
hydraulics.vanGenuchtenConductance(psi, krhizomax, n, alpha)
hydraulics.xylemConductance(psi, kxylemmax, c, d)
hydraulics.xylemPsi(kxylem, kxylemmax, c, d)
hydraulics.psi2Weibull(psi50, psi88)
```

Arguments

psi	A scalar (or a vector, depending on the function) with water potential (in MPa).
K	Whole-plant relative conductance (0-1).
Psi_extract	Soil water potential (in MPa) corresponding to 50% whole-plant relative conductance.
ws	Exponent of the whole-plant relative conductance Weibull function.
v	Proportion of fine roots within each soil layer.

krhizomax	Maximum rhizosphere hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
kxylemmax	Maximum xylem hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
c, d	Parameters of the Weibull function (generic xylem vulnerability curve).
n, alpha	Parameters of the Van Genuchten function (rhizosphere vulnerability curve).
kxylem	Xylem hydraulic conductance (defined as flow per surface unit and per pressure drop).
x	An object of class spwbInput.
soil	A list containing the description of the soil (see soil).
type	Plot type. For hydraulics.supplyFunctionPlot, either "E", "Elayers", "PsiStem", "PsiRoot", "PsiRhizo" or "dEdP"). For hydraulics.vulnerabilityCurvePlot, either "leaf", "stem", "root", or "rhizosphere").
psiVec	Vector of water potential values to evaluate for the vulnerability curve.
relative	A flag to relativize vulnerability curves to the [0-1] interval.
draw	A flag to indicate whether the vulnerability curve should be drawn or just returned.
pCrit	Proportion of maximum conductance considered critical for hydraulic functioning.
psi50, psi88	Water potentials (in MPa) corresponding to 50% and 88% of percent loss of conductance.

Details

Details of the hydraulic model are given in a vignette. Function hydraulics.vulnerabilityCurvePlot draws a plot of the vulnerability curves for the given soil object and network properties of each plant cohort in x.

Value

Values returned for each function are:

- hydraulics.psi2K: Whole-plant relative conductance (0-1).
- hydraulics.K2Psi: Soil water potential (in MPa) corresponding to the given whole-plant relative conductance value (inverse of hydraulics.psi2K()).
- hydraulics.averagePsi: The average water potential (in MPa) across soil layers.
- hydraulics.vanGenuchtenConductance: Rhizosphere conductance corresponding to an input water potential (soil vulnerability curve).
- hydraulics.xylemConductance: Xylem conductance (flow rate per pressure drop) corresponding to an input water potential (plant vulnerability curve).
- hydraulics.xylemPsi: Xylem water potential (in MPa) corresponding to an input xylem conductance (flow rate per pressure drop).
- hydraulics.psi2Weibull: Parameters of the Weibull vulnerability curve that goes through the supplied psi50 and psi88 values.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Sperry, J. S., F. R. Adler, G. S. Campbell, and J. P. Comstock. 1998. Limitation of plant water use by rhizosphere and xylem conductance: results from a model. Plant, Cell \& Environment 21:347–359.

Sperry, J. S., and D. M. Love. 2015. What plant hydraulics can tell us about responses to climate-change droughts. New Phytologist 207:14–27.

See Also

hydraulics.supplyFunctionPlot, hydraulics.maximumStemHydraulicConductance, spwb, soil

Examples

```
kstemmax = 4 # in mmol·m-2·s-1·MPa-1
stemc = 3
stemd = -4 # in MPa

psiVec = seq(-0.1, -7.0, by =-0.01)

#Vulnerability curve
kstem = unlist(lapply(psiVec, hydraulics.xylemConductance, kstemmax, stemc, stemd))
plot(-psiVec, kstem, type="l",ylab="Xylem conductance (mmol·m-2·s-1·MPa-1)",
xlab="Canopy pressure (-MPa)", lwd=1.5,ylim=c(0,kstemmax))
```

---

defaultControl *Default control parameters for models*

Description

Creates a list with global default parameters for simulation models.

Usage

```
defaultControl()
```

Details

The function returns a list with default parameters. Users can change those defaults that need to be set to other values and use the list as input for model functions. The relevant parameters are different for each model function.

Value

A list, with the following options:

- `verbose (=TRUE)`: Boolean flag to indicate console output during calculations.
- `subdailyResults (=FALSE)`: Boolean flag to force subdaily results to be stored (as a list called 'subdaily' of `spwb.day` objects, one by simulated date) in calls to `spwb`.
- `soilFunctions (="SX")`: Soil water retention curve and conductivity functions, either 'SX' (for Saxton) or 'VG' (for Van Genuchten).
- `snowpack (=TRUE)`: Boolean flag to indicate the simulation of snow accumulation and melting.
- `drainage (=TRUE)`: Boolean flag to indicate the simulation of deep drainage (not used in `spwbgrid`).
- `transpirationMode (="Simple")`: Transpiration model. See `spwbInput`.
- `hydraulicCostFunction (= 1)`: Variant of the hydraulic cost function used in the stomatal regulation model of Sperry & Love (2016). Values accepted are 1 (original cost function based on the derivative of supply function), 2 (leaf vulnerability curve).
- `ndailysteps (= 24)`: Number of steps into which each day is divided for determination of stomatal conductance, transpiration and photosynthesis (24 equals 1-hour intervals).
- `canopyMode (="sunshade")`: Indicates how crowns should be described to calculate photosynthesis. Accepted values are `"sunshade"` (distinguishes photosynthesis in sun leaves from shade leaves) and `"multilayer"` (distinguishes photosynthesis of sun leaves and shade leaves in each canopy layer).
- `verticalLayerSize (= 100)`: The size of vertical layers (in cm) for photosynthesis calculation.
- `nStemSegments (= 1)`: Number of segments within the stem.
- `capacitance (=FALSE)`: Whether capacitance is considered in simulations.
- `cavitationRefill (= TRUE)`: Whether refilling of embolized conduits is activated.
- `klat (= 0.1)`: Symplastic-apoplastic lateral conductance.
- `taper (= TRUE)`: Whether taper of xylem conduits is accounted for when calculating aboveground stem conductance from xylem conductivity.
- `numericParams`: A list with the following elements:
  - `maxNsteps (= 400)`: Maximum number of steps in supply function.
  - `ntrial (= 200)`: Number of iteration trials when finding root of equation system.
  - `psiTol (= 0.0001)`: Tolerance value for water potential.
  - `ETol (= 0.0001)`: Tolerance value for flow.
- `thermalCapacityLAI (=1000000)`: Thermal canopy capacitance per LAI unit.
- `defaultWindSpeed (=5)`: Default wind speed value (in m/s) to be used when missing from data.
- `Catm (=386)`: Atmospheric CO2 concentration (in micromol·mol-1 = ppm).
- `averageFracRhizosphereResistance (=0.15)`: Fraction to total continuum (stem+root+rhizosphere) resistance that corresponds to rhizosphere (averaged across soil water potential values).
- `storagePool (="none")`: Whether carbon storage pools are considered (either "none", "one" or "two").

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

spwbInput, spwb, spwbpoints

---

defaultSoilParams *Default soil parameters*

Description

Creates a data frame with default soil physical description for model functions.

Usage

```
defaultSoilParams(n = 4)
```

Arguments

n	An integer with the number of soil layers.

Details

The function returns a data frame with default physical soil description, with soil layers in rows. Users can change those that need to be set to other values and use the list as input for function soil.

Value

A data frame with layers in rows and the following columns (and default values):

- widths (= c(300,700,1000,2000): Width of soil layers (in mm).
- clay (= 25): Clay percentage for each layer (in %).
- sand (= 25): Sand percentage for each layer (in %).
- om (= NA): Organic matter percentage for each layer (in %).
- bd (= 1.5): Bulk density for each layer (in g/cm3).
- rfc (= c(20,40,60,85)): Percentage of rock fragment content for each layer.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

soil, soilgridsParams, defaultControl, SpParamsMED

Examples

```
defaultSoilParams(4)
```

exampleforest *Example forest stand*

Description

Data set for illustration of model behaviour. Includes a description of the plant cohorts of a forest stand.

Usage

```
data(exampleforest)
```

Format

An object of class `forest` containing the description of the tree, sapling and shrub cohorts of a forest patch as well as the seed bank and the size of the patch.

Source

DGCN (2005). Tercer Inventario Forestal Nacional (1997-2007): Catalunya. Dirección General de Conservación de la Naturaleza, Ministerio de Medio Ambiente, Madrid.

See Also

`forest`, `spwb`, `forest2spwbInput`

Examples

```
data(exampleforest)
```

examplemeteo *Example daily meteorology data*

Description

Example data set of meteorological input.

Usage

```
data(examplemeteo)
```

Format

A data frame containing daily meteorology of a location in Catalonia (Spain) for year 2001.

`MeanTemperature` Mean daily temperature (in degrees Celsius).

`MinTemperature` Minimum daily temperature (in degrees Celsius).

`MaxTemperature` Maximum daily temperature (in degrees Celsius).

`Precipitation` Daily precipitation (in mm of water).

`MeanRelativeHumidity` Mean daily relative humidity (in percent).

`MinRelativeHumidity` Minimum daily relative humidity (in percent).

`MaxRelativeHumidity` Maximum daily relative humidity (in percent).

`Radiation` Incoming radiation (in MJ/m2).

`WindSpeed` Wind speed (in m/s).

`WindDirection` Wind direction (in degrees from North).

`PET` Potential evapo-transpiration (in mm of water).

Source

Interpolated from weather station data (Spanish and Catalan meteorology agencies) using package 'meteoland'.

See Also

spwb

Examples

```
data(examplemeteo)
```

---

exampleSGL *Example of spatial grid with forest data*

Description

An example of an object of SpatialGridLandscape-class, with data taken from the Spanish Forest Inventory (DGCN 2005).

Usage

```
data("exampleSGL")
```

Format

The data format is that of an object SpatialGridLandscape-class

Source

DGCN (2005). Tercer Inventario Forestal Nacional (1997-2007): Catalunya. Dirección General de Conservación de la Naturaleza, Ministerio de Medio Ambiente, Madrid.

See Also

forest, exampleforest, SpatialGridLandscape-class

Examples

```
data(exampleSGL)

#Inspect forest object corresponding to the first pixel
exampleSGL@forestlist[[1]]
```

---

exampleSPL *Example of spatial points with forest data*

Description

An example of an object of SpatialPointsLandscape-class with data for 30 plots, taken from the Spanish Forest Inventory (DGCN 2005).

Usage

```
data("exampleSPL")
```

Format

The data format is that of an object SpatialPointsLandscape-class

Source

DGCN (2005). Tercer Inventario Forestal Nacional (1997-2007): Catalunya. Dirección General de Conservación de la Naturaleza, Ministerio de Medio Ambiente, Madrid.

See Also

forest, exampleforest, SpatialPointsLandscape-class

Examples

```
data(exampleSPL)

#Plot forest coordinates
plot(exampleSPL)

#Inspect forest object corresponding to the first point
exampleSPL@forestlist[[1]]
```

`extractSFIforest` *Extract forest from SFI data*

Description

Creates a `forest` object from Spanish Forest Inventory (SFI) data (DGCN 2005).

Usage

```
extractSFIforest(SFItreeData, SFIshrubData, ID, SpParams,
                 SFIherbData = NULL, SFIcodes=NULL,
                 patchsize= 10000, setDefaults=TRUE)
translateSpeciesCodes(x, SFIcodes)
```

Arguments

`SFItreeData`	A data frame with measured tree data.
`SFIshrubData`	A data frame with measured shrub data.
`ID`	A string with the ID of the plot to be extracted.
`SpParams`	A data frame with species parameters (see details).
`SFIcodes`	A string vector (of length equal to the number of rows in `SpParams` of the SFI species codes that correspond to the model species codification. Each string may contain different coma-separated codes in order to merge SFI species into a single model species.
`SFIherbData`	A data frame with cover and mean height of the herb layer.
`patchsize`	The area of the forest stand, in square meters.
`setDefaults`	Initializes default values for missing fields in SFI data.
`x`	A data frame with a column called 'Especie'.

Details

SFI data needs to be in a specific format. Function `extractSFIforest` calls `translateSpeciesCodes` internally.

Value

An object of class forest.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

DGCN (2005). Tercer Inventario Forestal Nacional (1997-2007): Catalunya. Dirección General de Conservación de la Naturaleza, Ministerio de Medio Ambiente, Madrid.

See Also

SFI2SPL, forest

---

fire.behaviour *Fire behaviour functions*

Description

Function `fire.FCCS()` implements a modification of the fire behavior models described for the Fuel Characteristics Classification System (FCCS) in Prichard et al. (2013). Function `fire.Rothermel()` implements Rothermel's (1972) fire behaviour model (modified from package 'Rothermel' (Giorgio Vacchiano, Davide Ascoli)).

Usage

```
fire.FCCS(FCCSpropsSI, MliveSI = as.numeric(c(90, 90, 60)),
          MdeadSI = as.numeric(c(6, 6, 6, 6, 6)),
          slope = 0, windSpeedSI = 11)
fire.Rothermel(modeltype, wSI, sSI, delta, mx_dead,
               hSI, mSI, u, windDir, slope, aspect)
```

Arguments

FCCSpropsSI	A data frame describing the properties of five fuel strata (canopy, shrub, herbs, dead woody and litter) returned by fuel.FCCS.
MliveSI	Moisture of live fuels (in percent of dry weight) for canopy, shrub, and herb strata.
MdeadSI	Moisture of dead fuels (in percent of dry weight) for canopy, shrub, herb, woody and litter strata.
slope	Slope (in degrees).
windSpeedSI	Wind speed (in m/s) at 20 ft (6 m) over vegetation (default 11 m/s = 40 km/h)
modeltype	'S'(tatic) or 'D'(ynamic)

wSI	A vector of fuel load (t/ha) for five fuel classes.
sSI	A vector of surface-to-volume ratio (m2/m3) for five fuel classes.
delta	A value of fuel bed depth (cm).
mx_dead	A value of dead fuel moisture of extinction (percent).
hSI	A vector of heat content (kJ/kg) for five fuel classes.
mSI	A vector of percent moisture on a dry weight basis (percent) for five fuel classes.
u	A value of windspeed (m/s) at midflame height.
windDir	Wind direction (in degrees from north). North means blowing from north to south.
aspect	Aspect (in degrees from north).

Details

Default moisture, slope and windspeed values are benchmark conditions used to calculate fire potentials (Sandberg et al. 2007) and map vulnerability to fire.

Value

Both functions return list with fire behavior variables. In the case of `fire.FCCS`, the function returns the variables in three blocks (lists `SurfaceFire`, `CrownFire` and `FirePotentials`), and the values are:

- `SurfaceFire$\`midflame_WindSpeed [m/s]\``: Midflame wind speed in the surface fire.
- `SurfaceFire$phi_wind`: Spread rate modifier due to wind.
- `SurfaceFire$phi_slope`: Spread rate modifier due to slope.
- `SurfaceFire$\`I_R_surf [kJ/m2/min]\``: Intensity of the surface fire reaction.
- `SurfaceFire$\`I_R_litter [kJ/m2/min]\``: Intensity of the litter fire reaction.
- `SurfaceFire$\`q_surf [kJ/m2]\``: Heat sink of the surface fire.
- `SurfaceFire$\`q_litter [kJ/m2]\``: Heat sink of the litter fire.
- `SurfaceFire$xi_surf`: Propagating flux ratio of the surface fire.
- `SurfaceFire$xi_litter`: Propagating flux ratio of the litter fire.
- `SurfaceFire$\`ROS_surf [m/min]\``: Spread rate of the surface fire(without accounting for faster spread in the litter layer).
- `SurfaceFire$\`ROS_litter [m/min]\``: Spread rate of the litter fire.
- `SurfaceFire$\`ROS_windslopecap [m/min]\``: Maximum surface fire spread rate according to wind speed.
- `SurfaceFire$\`ROS [m/min]\``: Final spread rate of the surface fire.
- `SurfaceFire$\`I_b [kW/m]\``: Fireline intensity of the surface fire.
- `SurfaceFire$\`FL [m]\``: Flame length of the surface fire.
- `CrownFire$\`I_R_canopy [kJ/m2/min]\``: Intensity of the canopy fire reaction.
- `CrownFire$\`I_R_crown [kJ/m2/min]\``: Intensity of the crown fire reaction (adding surface and canopy reactions).

- `CrownFire$`q_canopy [kJ/m2]``: Heat sink of the canopy fire.
- `CrownFire$`q_crown [kJ/m2]``: Heat sink of the crown fire (adding surface and canopy heat sinks).
- `CrownFire$xi_surf`: Propagating flux ratio of the crown fire.
- `CrownFire$`canopy_WindSpeed [m/s]``: Wind speed in the canopy fire (canopy top wind speed).
- `CrownFire$WAF`: Wind speed adjustment factor for crown fires.
- `CrownFire$`ROS [m/min]``: Spread rate of the crown fire.
- `CrownFire$Ic_ratio`: Crown initiation ratio.
- `CrownFire$`I_b [kW/m]``: Fireline intensity of the crown fire.
- `CrownFire$`FL [m]``: Flame length of the crown fire.
- `FirePotentials$RP`: Surface fire reaction potential ([0-9]).
- `FirePotentials$SP`: Surface fire spread rate potential ([0-9]).
- `FirePotentials$FP`: Surface fire flame length potential ([0-9]).
- `FirePotentials$SFP`: Surface fire potential ([0-9]).
- `FirePotentials$IC`: Crown initiation potential ([0-9]).
- `FirePotentials$TC`: Crown-to-crown transmission potential ([0-9]).
- `FirePotentials$RC`: Crown fire spread rate potential ([0-9]).
- `FirePotentials$CFC`: Crown fire potential ([0-9]).

Note

Default moisture, slope and windspeed values are benchmark conditions used to calculate fire potentials (Sandberg et al. 2007) and map vulnerability to fire.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Albini, F. A. (1976). Computer-based models of wildland fire behavior: A users' manual. Ogden, UT: US Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station.

Rothermel, R. C. 1972. A mathematical model for predicting fire spread in wildland fuels. USDA Forest Service Research Paper INT USA.

Prichard, S. J., D. V Sandberg, R. D. Ottmar, E. Eberhardt, A. Andreu, P. Eagle, and K. Swedin. 2013. Classification System Version 3.0: Technical Documentation.

See Also

`fuel.FCCS`

Examples

```
#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Calculate fuel properties according to FCCS
fccs = fuel.FCCS(exampleforest, 50,100, SpParamsMED)

#Calculate fire behavior according to FCCS
fire.FCCS(fccs)

#Load fuel model parameter data
data(SFM_metric)

#Fuel stratification (returns heights in cm)
fs = fuel.Stratification(exampleforest, SpParamsMED)

#Correct windspeed (transform heights to m)
u = 11 #m/s
umf = u*fuel.WindAdjustmentFactor(fs$surfaceLayerTopHeight/100, fs$canopyBaseHeight/100,
                                  fs$canopyTopHeight/100, 60)

#Call Rothermel function using fuel model 'A6'
fire.Rothermel(modeltype="D", wSI = as.numeric(SFM_metric["A6",2:6]),
                    sSI = as.numeric(SFM_metric["A6",7:11]),
                    delta = as.numeric(SFM_metric["A6",12]),
                    mx_dead = as.numeric(SFM_metric["A6",13]),
                    hSI = as.numeric(SFM_metric["A6",14:18]),
                    mSI = c(10,10,10,30,60),
                    u=umf, windDir=0, slope=0, aspect=0)
```

---

forest *Forest description*

Description

Description of a forest patch.

Usage

```
## S3 method for class 'forest'
summary(object, SpParams, detailed=FALSE, ...)
## S3 method for class 'summary.forest'
print(x, digits = getOption("digits"), ...)
emptyforest(ID="", patchsize=10000, ntree = 0, nshrub = 0)
```

Arguments

object	An object of class forest has the following structure: • ID: An identifier of the forest stand (a string). • patchsize: The area of the forest stand, in square meters. • treeData: A data frame of tree cohorts (in rows) and the following columns: – Species: Non-negative integer for tree species identity (i.e., 0,1,2,...). – Height: Total height (in cm). – DBH: Diameter at breast height (in cm). – N: Density (number of individuals/cell). – Z50: Depth (in mm) corresponding to 50% of fine roots. – Z95: Depth (in mm) corresponding to 95% of fine roots. • shrubData: A data frame of shrub cohorts (in rows) and the following columns: – Species: Non-negative integer for shrub species identity (i.e., 0,1,2,...). – Height: Total height (in cm). – Cover: Percent cover. – Z50: Depth (in mm) corresponding to 50% of fine roots. – Z95: Depth (in mm) corresponding to 95% of fine roots. • seedBank: A data frame containing the abundance of seeds for each species (in rows) and the following columns: – Species: Non-negative integer for shrub species identity (i.e., 0,1,2,...). – Abundance: Abundance class (0 - none; 1 - low; 2 - medium; 3 - high; 4 - very high). • herbCover: Percent cover of the herb layer. • herbHeight: Mean height (in cm) of the herb layer.
SpParams	A data frame with species parameters (see SpParamsMED).
detailed	A boolean flag to indicate that a detailed summary is desired.
x	The object returned by summary.forest.
digits	Minimal number of significant digits.
...	Additional parameters for functions summary and print.
ID	An identifier of the forest stand (a string).
patchsize	The area of the forest stand, in square meters.
ntree, nshrub	Number of tree and shrub cohorts, respectively.

Details

Function summary.forest can be used to summarize a forest object in the console. Function emptyforest creates an empty forest object.

Value

Function summary.forest returns a data frame with the basal area and LAI of the forest, either expressed as totals or divided among life stages and species. Function emptyforest returns an empty forest object.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

exampleforest, extractSFIforest, soil, SFI2SPL

Examples

```
data(exampleforest)
data(SpParamsMED)

summary(exampleforest, SpParamsMED)
```

Forest values *Forest description functions*

Description

Functions to calculate overall attributes of a forest object.

Usage

```
forest.BasalArea(x)
```

Arguments

x	An object of class forest.

Value

A vector with values for each cohort of the input forest object:

- forest.BasalArea: Total basal area (m2/ha).

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

spwb, forest, plant.BasalArea, summary.forest

Examples

```
#Default species parameterization
data(SpParamsMED)

#Load example plot
data(exampleforest)

#A short way to obtain total basal area
forest.BasalArea(exampleforest)
```

`fuel.properties` *Fuel stratification and fuel characteristics*

Description

Function `fuel.Stratification` provides a stratification of the stand into understory and canopy strata. Function `fuel.FCCS` calculates fuel characteristics from a `forest` object following an adaptation of the protocols described for the Fuel Characteristics Classification System (Prichard et al. 2013). Function `fuel.WindAdjustmentFactor` determines the adjustment factor of wind for surface fires, according to Andrews (2012). Function `fuel.cohortFineFMC` calculates the fuel moisture content of leaves and twigs of each cohort, from the results of soil water balance.

Usage

```
fuel.Stratification(object, SpParams, gdd = NA,
                    heightProfileStep = 10.0, maxHeightProfile = 5000.0,
                    bulkDensityThreshold = 0.05)
fuel.FCCS(object, ShrubCover, CanopyCover, SpParams, cohortFMC = as.numeric(c()),
          gdd = NA, heightProfileStep = 10, maxHeightProfile = 5000,
          bulkDensityThreshold = 0.05)
fuel.cohortFineFMC(spwb, x)
fuel.WindAdjustmentFactor(topShrubHeight, bottomCanopyHeight, topCanopyHeight,
                          canopyCover)
```

Arguments

`object`	An object of class `forest`
`ShrubCover`	Total shrub cover (in percent) of the stand.
`CanopyCover`	Total canopy cover (in percent) of the stand.
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`cohortFMC`	A numeric vector of (actual) fuel moisture content by cohort (e.g. taken from the result of `fuel.cohortFineFMC`).
`gdd`	Growth degree-days.
`heightProfileStep`	Precision for the fuel bulk density profile.

`maxHeightProfile`
: Maximum height for the fuel bulk density profile.

`bulkDensityThreshold`
: Minimum fuel bulk density to delimit fuel strata.

`spwb`
: Object returned by function `spwb`.

`topShrubHeight`
: Shrub stratum top height (in m).

`bottomCanopyHeight`
: Canopy base height (in m).

`topCanopyHeight`
: Canopy top height (in m).

`canopyCover`
: Canopy percent cover.

`x`
: An object of class `spwbInput`.

Details

Details are described in a vignette.

Value

Function `fuel.FCCS` returns a data frame with five rows corresponding to fuel layers: `canopy`, `shrub`, `herb`, `woody` and `litter`. Columns correspond fuel properties:

- `w`: Fine fuel loading (in kg/m2).
- `cover`: Percent cover.
- `hbc`: Height to base of crowns (in m).
- `htc`: Height to top of crowns (in m).
- `delta`: Fuel depth (in m).
- `rhob`: Fuel bulk density (in kg/m3).
- `rhop`: Fuel particle density (in kg/m3).
- `PV`: Particle volume (in m3/m2).
- `beta`: Packing ratio (unitless).
- `betarel`: Relative packing ratio (unitless).
- `etabetarel`: Reaction efficiency (unitless).
- `sigma`: Surface area-to-volume ratio (m2/m3).
- `pDead`: Proportion of dead fuels.
- `FAI`: Fuel area index (unitless).
- `h`: High heat content (in kJ/kg).
- `etaF`: Flammability modifier (between 1 and 2).
- `RV`: Reactive volume (in m3/m2).
- `MinFMC`: Minimum fuel moisture content (as percent over dry weight).
- `MaxFMC`: Maximum fuel moisture content (as percent over dry weight).

Function `fuel.Stratification` returns a list with the following items:

- `surfaceLayerBaseHeight`: Base height of crowns of shrubs in the surface layer (in cm).
- `surfaceLayerTopHeight`: Top height of crowns of shrubs in the surface layer (in cm).
- `understoryLAI`: Cumulated LAI of the understory layer (i.e. leaf area comprised between surface layer base and top heights).
- `canopyBaseHeight`: Base height of tree crowns in the canopy (in cm).
- `canopyTopHeight`: Top height of tree crowns in the canopy (in cm).
- `canopyLAI`: Cumulated LAI of the canopy (i.e. leaf area comprised between canopy base and top heights).

Function `fuel.cohortFineFMC` returns a list with three matrices (for leaves, twigs and fine fuels). Each of them contains live moisture content values for each day (in rows) and plant cohort (in columns).

Function `fuel.WindAdjustmentFactor` returns a value between 0 and 1.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Andrews, P. L. 2012. Modeling wind adjustment factor and midflame wind speed for Rothermel's surface fire spread model. USDA Forest Service - General Technical Report RMRS-GTR:1–39.

Prichard, S. J., D. V Sandberg, R. D. Ottmar, E. Eberhardt, A. Andreu, P. Eagle, and K. Swedin. 2013. Classification System Version 3.0: Technical Documentation.

Reinhardt, E., D. Lutes, and J. Scott. 2006. FuelCalc: A method for estimating fuel characteristics. Pages 273–282.

See Also

`fire.FCCS`, `spwb`

Examples

```
#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Show stratification of fuels
fuel.Stratification(exampleforest, SpParamsMED)

#Calculate fuel properties according to FCCS
fccs = fuel.FCCS(exampleforest, 50,100, SpParamsMED)
fccs

fuel.WindAdjustmentFactor(fccs$htc[2], fccs$hbc[1], fccs$htc[1], fccs$cover[1])
```

growth *Forest growth*

Description

Function growth is a forest growth model that calculates changes in leaf area, sapwood area and structural variables for all plant cohorts in a given forest stand during a period specified in the input climatic data.

Usage

```
growth(x, soil, meteo,  latitude = NA, elevation = NA, slope = NA, aspect = NA)
```

Arguments

x	An object of class growthInput.
soil	A list containing the description of the soil (see soil).
meteo	A data frame with daily meteorological data series. Row names of the data frame should correspond to date strings with format "yyyy-mm-dd" (see Date). When x$TranspirationMode = "Simple" the following columns are required: • MeanTemperature: Mean temperature (in degrees Celsius). • Precipitation: Precipitation (in mm). • Radiation: Solar radiation (in MJ/m2/day), required only if snowpack = TRUE. • PET: Potential evapotranspiration (in mm). • WindSpeed: Wind speed (in m/s). If not available, this column can be left with NA values. When x$TranspirationMode = "Complex" the following columns are required: • MeanTemperature: Mean temperature(in degrees Celsius). • MinTemperature: Minimum temperature (in degrees Celsius). • MaxTemperature: Maximum temperature (in degrees Celsius). • MinRelativeHumidity: Minimum relative humidity (in percent). • MaxRelativeHumidity: Maximum relative humidity (in percent). • Precipitation: Precipitation (in mm). • Radiation: Solar radiation (in MJ/m2/day). • WindSpeed: Wind speed (in m/s). If not available, this column can be left with NA values.
latitude	Latitude (in degrees). Required when x$TranspirationMode = "Complex".
elevation, slope, aspect	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North). Required when x$TranspirationMode = "Complex". Elevation is also required for 'Simple' if snowpack dynamics are simulated.

Details

Detailed model description is available in the vignettes section. Simulations using the 'Complex' transpiration mode are computationally much more expensive than those using the simple transpiration mode.

Value

A list of class 'growth' with the following elements:

- `"spwbInput"`: A copy of the object `x` of class `spwbInput` given as input.
- `"soilInput"`: A copy of the object `soil` of class `soil` given as input.
- `"WaterBalance"`: A data frame where different variables (in columns) are given for each simulated day (in rows):
  - `"LAIcell"`: The LAI of the stand (accounting for leaf phenology) (in m2/m2).
  - `"Cm"`: The water retention capacity of the stand (in mm) (accounting for leaf phenology).
  - `"Lground"`: The proportion of PAR that reaches the ground (accounting for leaf phenology).
  - `"PET"`: Potential evapotranspiration (in mm).
  - `"Rainfall"`: Input precipitation (in mm).
  - `"NetPrec"`: Net precipitation, after accounting for interception (in mm).
  - `"Infiltration"`: The amount of water infiltrating into the soil (in mm).
  - `"Runoff"`: The amount of water exported via surface runoff (in mm).
  - `"DeepDrainage"`: The amount of water exported via deep drainage (in mm).
  - `"Etot"`: Evapotranspiration (in mm).
  - `"Esoil"`: Bare soil evaporation (in mm).
  - `"Eplanttot"`: Plant transpiration (considering all soil layers) (in mm).
  - `"Eplant.1"`, ..., `"Eplant.k"`: Plant transpiration from each soil layer (in mm).
- `"Soil"`: A data frame where different variables (in columns) are given for each simulated day (in rows):
  - `"W.1"`, ..., `"W.k"`: Relative soil moisture content (relative to field capacity) in each soil layer.
  - `"ML.1"`, ..., `"ML.k"`: Soil water volume in each soil layer (in L/m2).
  - `"MLTot"`: Total soil water volume (in L/m2).
  - `"psi.1"`, ..., `"psi.k"`: Soil water potential in each soil layer (in MPa).
- `"PlantTranspiration"`: A data frame with the amount of daily transpiration (in mm) for each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- `"PlantPhotosynthesis"`: A data frame with the amount of daily photosynthesis (in g C · m-2) for each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- `"PlantRespiration"`: A data frame with the amount of daily maintenance respiration (in g C · m-2) for each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.

- "PlantCStorageFast": A data frame with the daily amount of fast-dynamics carbon reserves (as proportion of total storage capacity) for an average individual of each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantCStorageSlow": A data frame with the daily amount of slow-dynamics carbon reserves (as proportion of total storage capacity) for an average individual of each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantSAgrowth": A data frame with the daily amount of newly created sapwood area (in cm2) for an average individual of each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantSA": A data frame with the daily amount of sapwood area (in cm2) for an average individual of each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantStress": A data frame with the amount of daily stress suffered by each plant cohort (relative whole-plant conductance). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantPsi": A data frame with the average daily water potential of each plant (in MPa). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantLAIdead": A data frame with the dead leaf area index (in m2 · m-2) for each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantLAIlive": A data frame with the live leaf area index (in m2 · m-2) for each plant cohort. Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.

Note

Objects x and soil are modified during the simulation.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

growthInput, growthpoints, forest

Examples

```
#Load example daily meteorological data
data(examplemeteo)

#Load example plot plant data
data(exampleforest)

#Default species parameterization
```

```
data(SpParamsMED)

#Initialize soil with default soil params (2 layers)
examplesoil = soil(defaultSoilParams(2))

#Initialize control parameters
control = defaultControl()

#Initialize input
x = forest2growthInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
G1<-growth(x, examplesoil, examplemeteo, elevation = 100)
```

`hydrology.rainInterception` *Rainfall interception*

Description

Function `hydrology.rainInterception` calculates the amount of rainfall intercepted daily by the canopy, given a rainfall and canopy characteristics. Two canopy interception models are currently available: the sparse Gash (1995) model and the Liu (2001) model. In both cases the current implementation assumes no trunk interception.

Usage

```
hydrology.rainInterception(Rainfall, Cm, p, ER=0.05, method="Gash1995")
```

Arguments

`Rainfall`	A numeric vector of (daily) rainfall.
`Cm`	Canopy water storage capacity.
`p`	Proportion of throughfall (normally 1 - c, where c is the canopy cover).
`ER`	The ratio of evaporation rate to rainfall rate.
`method`	Rainfall interception method (either `"Gash1995"` or `Liu2001`).

Details

The function is prepared to accept both vectors or scalars for parameters `Cm`, `p` and `ER`. If they are supplied as vectors they should be of the same length as `Rainfall`.

Value

Returns a vector of the same length as `Rainfall` containing intercepted rain values.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Liu (2001). Evaluation of the Liu model for predicting rainfall interception in forests world-wide. - Hydrol. Process. 15: 2341-2360.

Gash (1979). An analytical model of rainfall interception by forests. - Quarterly Journal of the Royal Meteorological Society

Gash et al. (1995). Estimating sparse forest rainfall interception with an analytical model. - Journal of Hydrology

See Also

spwb

Examples

```
throughfallMatrixGash<-function(P = seq(1,50, by=1), Cm = seq(1,5, by=1),
                                ER = 0.08,p=0.8) {
  m2<- P - hydrology.rainInterception(P,Cm[1],p,ER=ER)
  for(i in 2:length(Cm)) {
    m2<-rbind(m2,P-hydrology.rainInterception(P,Cm[i],p,ER=ER))
  }
  colnames(m2)<-P
  rownames(m2)<-Cm
  return(m2)
}

Cm = c(0.5,seq(1,4, by=1))
P = seq(1,50, by=1)

m2 = throughfallMatrixGash(P=P, p=0.2, Cm=Cm,ER = 0.05)
rt = sweep(m2,2,P,"/")*100
matplot(t(rt), type="l", axes=TRUE, ylab="Relative throughfall (%)",
        xlab="Gross rainfall (mm)", xlim=c(0,length(P)),
        lty=1:length(Cm), col="black", ylim=c(0,100))
title(main="p = 0.2 E/R = 0.05")
legend("bottomright",lty=1:length(Cm), legend=paste("Cm =",Cm), bty="n")
```

---

hydrology.soilInfiltration

Soil infiltration and bare soil evaporation

Description

Function `hydrology.soilInfiltration` calculates the amount of water that infiltrates into the topsoil, according to the USDA SCS curve number method (Boughton 1989). The remaining is assumed to be lost as surface runoff. Function `hydrology.soilEvaporation` calculates the amount of evaporation from bare soil, following Ritchie (1972).

Usage

```
hydrology.soilInfiltration(input, Ssoil)
hydrology.soilEvaporation(DEF,PETs, Gsoil)
hydrology.infiltrationRepartition(I, dVec, macro)
```

Arguments

`input`	A numeric vector of (daily) water input (in mm of water).
`Ssoil`	Soil water storage capacity (can be referred to topsoil) (in mm of water).
`DEF`	Water deficit in the (topsoil) layer.
`PETs`	Potential evapotranspiration at the soil surface.
`Gsoil`	Gamma parameter (maximum daily evaporation).
`I`	Soil infiltration (in mm of water).
`dVec`	Width of soil layers (in mm).
`macro`	Macroporosity of soil layers (in %).

Details

See description of infiltration and soil evaporation processes in De Caceres et al. (submitted).

Value

Function `hydrology.soilInfiltration` a vector of the same length as `input` containing the daily amount of water that infiltrates into the soil (in mm of water). Function `hydrology.soilEvaporation` returns the amount of water evaporated from the soil. Function `hydrology.infiltrationRepartition` estimates the amount of infiltrated water that reaches each soil layer.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Boughton (1989). A review of the USDA SCS curve number method. - Australian Journal of Soil Research 27: 511-523.

De Cáceres M, Martínez-Vilalta J, Coll L, Llorens P, Casals P, Poyatos R, Pausas JG, Brotons L. (submitted) Coupling a water balance model with forest inventory data to evaluate plant drought stress at the regional level. Agricultural and Forest Meteorology.

Ritchie (1972). Model for predicting evaporation from a row crop with incomplete cover. - Water resources research.

See Also

spwb

Examples

```
SoilDepth = c(200,400,800,1200,1500)

#TOPSOIL LAYERS
d1 = pmin(SoilDepth, 300) #<300
#SUBSOIL LAYERS
d2 = pmax(0, pmin(SoilDepth-300,1200)) #300-1500 mm
#ROCK LAYER
d3 = 4000-(d1+d2) #From SoilDepth down to 4.0 m

TS_clay = 15
TS_sand = 25
SS_clay = 15
SS_sand = 25
RL_clay = 15
RL_sand = 25
TS_gravel = 20
SS_gravel = 40
RL_gravel = 95

Theta_FC1=soil.psi2thetaSX(TS_clay, TS_sand, -33) #in m3/m3
Theta_FC2=soil.psi2thetaSX(SS_clay, SS_sand, -33) #in m3/m3
Theta_FC3=soil.psi2thetaSX(RL_clay, RL_sand, -33) #in m3/m3
pcTS_gravel = 1-(TS_gravel/100)
pcSS_gravel = 1-(SS_gravel/100)
pcRL_gravel = 1-(RL_gravel/100)
MaxVol1 = (d1*Theta_FC1*pcTS_gravel)
MaxVol2 = (d2*Theta_FC2*pcSS_gravel)
MaxVol3 = (d3*Theta_FC3*pcRL_gravel)
V = MaxVol1+MaxVol2+MaxVol3

par(mar=c(5,5,1,1), mfrow=c(1,2))
NP = seq(0,60, by=1)
plot(NP,hydrology.soilInfiltration(NP, V[1]), type="l", xlim=c(0,60), ylim=c(0,60),
     ylab="Infiltration (mm)", xlab="Net rainfall (mm)", frame=FALSE)
lines(NP,hydrology.soilInfiltration(NP, V[2]), lty=2)
lines(NP,hydrology.soilInfiltration(NP, V[3]), lty=3)
lines(NP,hydrology.soilInfiltration(NP, V[4]), lty=4)
lines(NP,hydrology.soilInfiltration(NP, V[5]), lty=5)
legend("topleft", bty="n", lty=1:5,
       legend=c(paste("d =", SoilDepth, "Vsoil =",round(V),"mm")))
plot(NP,NP-hydrology.soilInfiltration(NP, V[1]), type="l", xlim=c(0,60), ylim=c(0,60),
     ylab="Runoff (mm)", xlab="Net rainfall (mm)", frame=FALSE)
lines(NP,NP-hydrology.soilInfiltration(NP, V[2]), lty=2)
lines(NP,NP-hydrology.soilInfiltration(NP, V[3]), lty=3)
lines(NP,NP-hydrology.soilInfiltration(NP, V[4]), lty=4)
lines(NP,NP-hydrology.soilInfiltration(NP, V[5]), lty=5)
legend("topleft", bty="n", lty=1:5,
```

```
          legend=c(paste("d =", SoilDepth,"Vsoil =",round(V),"mm")))
```

---

light *Light extinction and absortion functions*

Description

Functions `light.layerIrradianceFraction` and `light.layerIrradianceFractionBottomUp` calculate the fraction of above-canopy irradiance (and the soil irradiance, respectively) reaching each vegetation layer. Function `light.layerSunlitFraction` calculates the proportion of sunlit leaves in each vegetation layer. Function `light.cohortSunlitShadeAbsorbedRadiation` calculates the amount of radiation absorved by cohort and vegetation layers, while differentiating between sunlit and shade leaves.

Usage

```
light.layerIrradianceFraction(LAIme, LAImd, LAImx, k, alpha,
                              trunkExtinctionFraction = 0.1)
light.layerIrradianceFractionBottomUp(LAIme, LAImd, LAImx, k, alpha,
                              trunkExtinctionFraction = 0.1)
light.layerSunlitFraction(LAIme, LAImd, kb)
light.cohortSunlitShadeAbsorbedRadiation(Ib0, Id0, Ibf, Idf, beta,
                                         LAIme, LAImd,
                                         kb, kd, alpha, gamma)
light.instantaneousLightExtinctionAbsortion(LAIme, LAImd, LAImx,
                                            kPAR, gammaSWR,
                                            ddd, LWR_diffuse,
                                        ntimesteps = 24, canopyMode= "sunshade",
                                            trunkExtinctionFraction = 0.1)
```

Arguments

`LAIme`	A numeric matrix of live expanded LAI values per vegetation layer (row) and cohort (column).
`LAImd`	A numeric matrix of dead LAI values per vegetation layer (row) and cohort (column).
`LAImx`	A numeric matrix of maximum LAI values per vegetation layer (row) and cohort (column).
`k`	A vector of light extinction coefficients.
`kb`	A vector of direct light extinction coefficients.
`kd`	A vector of diffuse light extinction coefficients.
`Ib0`	Above-canopy direct incident radiation.
`Id0`	Above-canopy diffuse incident radiation.

Ibf	Fraction of above-canopy direct radiation reaching each vegetation layer.
Idf	Fraction of above-canopy diffuse radiation reaching each vegetation layer.
alpha	A vecfor of leaf absorvance by species.
beta	Solar elevation (in radians).
gamma	Vector of canopy reflectance values.
kPAR	A vector of visible light extinction coefficients for each cohort.
gammaSWR	A vector of short-wave reflectance coefficients (albedo) for each cohort.
ddd	A dataframe with direct and diffuse radiation for different subdaily time steps (see function radiation_directDiffuseDay in package meteoland).
LWR_diffuse	A vector with diffuse longwave radiation for different subdaily time steps.
ntimesteps	Number of subdaily time steps.
canopyMode	Indicates how crowns should be described to calculate photosynthesis/extinction. Accepted values are "sunshade" (distinguishes photosynthesis in sun leaves from shade leaves) and "multilayer" (distinguishes photosynthesis of sun leaves and shade leaves in each canopy layer).
trunkExtinctionFraction	Fraction of extinction due to trunks (for winter deciduous forests).

Details

Function codification adapted from Anten & Bastiaans (2016). Vegetation layers are assumed to be ordered from bottom to top.

Value

Functions light.layerIrradianceFraction, light.layerIrradianceFractionBottomUp and light.layerSunlitFraction return a numeric vector of length equal to the number of vegetation layers. Function light.cohortSunlitShadeAbsorbedRadiation returns a list with two elements (matrices): I_sunlit and I_shade.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Anten, N.P.R., Bastiaans, L., 2016. The use of canopy models to analyze light competition among plants, in: Hikosaka, K., Niinemets, U., Anten, N.P.R. (Eds.), Canopy Photosynthesis: From Basics to Application. Springer, pp. 379–398.

See Also

spwb

Examples

```
LAI = 2
nlayer = 10
LAIlayerlive = matrix(rep(LAI/nlayer,nlayer),nlayer,1)
LAIlayerdead = matrix(0,nlayer,1)
kb = 0.8
kd_PAR = 0.5
kd_SWR = kd_PAR/1.35
alpha_PAR = 0.9
gamma_PAR = 0.04
gamma_SWR = 0.05
alpha_SWR = 0.7

Ibfpar = light.layerIrradianceFraction(LAIlayerlive,LAIlayerdead,LAIlayerlive,kb, alpha_PAR)
Idfpar = light.layerIrradianceFraction(LAIlayerlive,LAIlayerdead,LAIlayerlive,kd_PAR, alpha_PAR)
Ibfswr = light.layerIrradianceFraction(LAIlayerlive,LAIlayerdead,LAIlayerlive,kb, alpha_SWR)
Idfswr = light.layerIrradianceFraction(LAIlayerlive,LAIlayerdead,LAIlayerlive,kd_SWR, alpha_SWR)
fsunlit = light.layerSunlitFraction(LAIlayerlive, LAIlayerdead, kb)
SHarea = (1-fsunlit)*LAIlayerlive[,1]
SLarea = fsunlit*LAIlayerlive[,1]

par(mar=c(4,4,1,1), mfrow=c(1,2))
plot(Ibfpar*100, 1:nlayer,type="l", ylab="Layer",
     xlab="Percentage of irradiance", xlim=c(0,100), ylim=c(1,nlayer), col="dark green")
lines(Idfpar*100, 1:nlayer, col="dark green", lty=2)
lines(Ibfswr*100, 1:nlayer, col="red")
lines(Idfswr*100, 1:nlayer, col="red", lty=2)

plot(fsunlit*100, 1:nlayer,type="l", ylab="Layer",
     xlab="Percentage of leaves", xlim=c(0,100), ylim=c(1,nlayer))
lines((1-fsunlit)*100, 1:nlayer, lty=2)


solarElevation = 0.67
SWR_direct = 1100
SWR_diffuse = 300
PAR_direct = 550
PAR_diffuse = 150


abs_PAR = light.cohortSunlitShadeAbsorbedRadiation(PAR_direct, PAR_diffuse,
                        Ibfpar, Idfpar, beta = solarElevation,
                        LAIlayerlive, LAIlayerdead, kb, kd_PAR, alpha_PAR, gamma_PAR)
abs_SWR = light.cohortSunlitShadeAbsorbedRadiation(SWR_direct, SWR_diffuse,
                        Ibfswr, Idfswr, beta = solarElevation,
                        LAIlayerlive, LAIlayerdead, kb, kd_SWR, alpha_SWR, gamma_SWR)
par(mar=c(4,4,1,1), mfrow=c(1,2))
absRadSL = abs_SWR$I_sunlit[,1]
absRadSH = abs_SWR$I_shade[,1]
lambda = 546.6507
QSL = abs_PAR$I_sunlit[,1]*lambda*0.836*0.01
QSH = abs_PAR$I_shade[,1]*lambda*0.836*0.01
```

```
plot(QSL, 1:nlayer,type="l", ylab="Layer",
     xlab="Absorbed PAR quantum flux per leaf area", ylim=c(1,nlayer), col="dark green",
     xlim=c(0,max(QSL)))
lines(QSH, 1:nlayer, col="dark green", lty=2)
plot(absRadSL, 1:nlayer,type="l", ylab="Layer",
     xlab="Absorbed SWR per leaf area (W/m2)", ylim=c(1,nlayer), col="red",
     xlim=c(0,max(absRadSL)))
lines(absRadSH, 1:nlayer, col="red", lty=2)
```

---

photo *Photosynthesis submodel functions*

Description

Set of functions used in the calculation of photosynthesis.

Usage

```
photo.GammaTemp(leaf_temp)
photo.KmTemp(leaf_temp, Oi = 209)
photo.VmaxTemp(Vmax298, leaf_temp)
photo.JmaxTemp(Jmax298, leaf_temp)
photo.electronLimitedPhotosynthesis(Q, Ci, GT, Jmax)
photo.rubiscoLimitedPhotosynthesis(Ci, GT, Km, Vmax)
photo.photosynthesis(Q, Catm, Gc, leaf_temp, Vmax298, Jmax298, verbose = FALSE)
photo.leafPhotosynthesisFunction(E, Catm, Patm, Tair, vpa, u,
                                 absRad, Q, Vmax298, Jmax298, Gwmin, Gwmax,
                                 leafWidth = 1.0, refLeafArea = 1, verbose = FALSE)
photo.sunshadePhotosynthesisFunction(E, Catm, Patm, Tair, vpa,
                                     SLarea, SHarea, u,
                                     absRadSL, absRadSH, QSL, QSH,
                                    Vmax298SL, Vmax298SH, Jmax298SL, Jmax298SH,
                                  Gwmin, Gwmax, leafWidth = 1.0, verbose = FALSE)
photo.multilayerPhotosynthesisFunction(E, Catm, Patm, Tair, vpa,
                                       SLarea, SHarea, u,
                                       absRadSL, absRadSH, QSL, QSH,
                                 Vmax298, Jmax298, Gwmin, Gwmax, leafWidth = 1.0,
                                       verbose = FALSE)
```

Arguments

`leaf_temp`	Leaf temperature (in ºC).
`Oi`	Oxigen concentration (mmol*mol-1).
`Vmax298, Vmax298SL, Vmax298SH`	Maximum Rubisco carboxylation rate per leaf area at 298ºK (i.e. 25 ºC) (micromol*s-1*m-2) (for each canopy layer in the case of `photo.multilayerPhotosynthesisFunction`). 'SH' stands for shade leaves, whereas 'SL' stands for sunlit leaves.

`Jmax298, Jmax298SL, Jmax298SH`
: Maximum electron transport rate per leaf area at 298ºK (i.e. 25 ºC) (micromol*s-1*m-2) (for each canopy layer in the case of `photo.multilayerPhotosynthesisFunction`). 'SH' stands for shade leaves, whereas 'SL' stands for sunlit leaves.

`Q`
: Active photon flux density (micromol * s-1 * m-2).

`Ci`
: CO2 internal concentration (micromol * mol-1).

`GT`
: CO2 saturation point corrected by temperature (micromol * mol-1).

`Jmax`
: Maximum electron transport rate per leaf area (micromol*s-1*m-2).

`Km`
: Km = Kc*(1.0+(Oi/Ko)) - Michaelis-Menten term corrected by temperature (in micromol * mol-1).

`Vmax`
: Maximum Rubisco carboxylation rate per leaf area (micromol*s-1*m-2).

`Catm`
: CO2 air concentration (micromol * mol-1).

`Gc`
: CO2 leaf (stomatal) conductance (mol * s-1 * m-2).

`E`
: Transpiration flow rate per leaf area (mmol*s-1*m-2).

`Patm`
: Atmospheric air pressure (in kPa).

`Tair`
: Air temperature (in ºC).

`vpa`
: Vapour pressure deficit (in kPa).

`u`
: Wind speed above the leaf boundary (in m/s) (for each canopy layer in the case of `photo.multilayerPhotosynthesisFunction`).

`absRad`
: Absorbed long- and short-wave radiation (in W*m^-2).

`leafWidth`
: Leaf width (in cm).

`refLeafArea`
: Leaf reference area.

`Gwmin, Gwmax`
: Minimum and maximum leaf water conductance (i.e. cuticular and maximum stomatal conductance) (mol * s-1 * m-2).

`verbose`
: Boolean flag to indicate console output.

`SLarea, SHarea`
: Leaf area index of sunlit/shade leaves (for each canopy layer in the case of `photo.multilayerPhotosynthesisFunction`).

`absRadSL, absRadSH`
: Instantaneous absorbed radiation (W·m-2) per unit of sunlit/shade leaf area (for each canopy layer in the case of `photo.multilayerPhotosynthesisFunction`).

`QSL, QSH`
: Active photon flux density (micromol * s-1 * m-2) per unit of sunlit/shade leaf area (for each canopy layer in the case of `photo.multilayerPhotosynthesisFunction`).

Details

Details of the photosynthesis submodel are given in a vignette.

Value

Values returned for each function are:

- `photo.GammaTemp`: CO2 compensation concentration (micromol * mol-1).

- `photo.KmTemp`: Michaelis-Menten coefficients of Rubisco for Carbon (micromol * mol-1) and Oxigen (mmol * mol-1).
- `photo.VmaxTemp`: Temperature correction of Vmax298.
- `photo.JmaxTemp`: Temperature correction of Jmax298.
- `photo.electronLimitedPhotosynthesis`: Electron-limited photosynthesis (micromol*s-1*m-2) following Farquhar et al. (1980).
- `photo.rubiscoLimitedPhotosynthesis`: Rubisco-limited photosynthesis (micromol*s-1*m-2) following Farquhar et al. (1980).
- `photo.photosynthesis`: Calculates gross photosynthesis (micromol*s-1*m-2) following (Farquhar et al. (1980) and Collatz et al (1991).
- `photo.leafPhotosynthesisFunction`: Returns a data frame with the following columns:
  - `LeafTemperature`: Leaf temperature (ºC).
  - `LeafVPD`: Leaf vapor pressure deficit (kPa).
  - `WaterVaporConductance`: Leaf vapor conductance (mol * s-1 * m-2).
  - `Photosynthesis`: Gross photosynthesis (micromol*s-1*m-2).
  - `NetPhotosynthesis`: Net photosynthesis, after discounting autotrophic respiration (micromol*s-1*m-2).
- `photo.sunshadePhotosynthesisFunction` and `photo.multilayerPhotosynthesisFunction`: Return a data frame with the following columns:
  - `Photosynthesis`: Gross photosynthesis (micromol*s-1*m-2).
  - `NetPhotosynthesis`: Net photosynthesis, after discounting autotrophic respiration (micromol*s-1*m-2).

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Bernacchi, C. J., E. L. Singsaas, C. Pimentel, A. R. Portis, and S. P. Long. 2001. Improved temperature response functions for models of Rubisco-limited photosynthesis. Plant, Cell and Environment 24:253–259.

Collatz, G. J., J. T. Ball, C. Grivet, and J. A. Berry. 1991. Physiological and environmental regulation of stomatal conductance, photosynthesis and transpiration: a model that includes a laminar boundary layer. Agricultural and Forest Meteorology 54:107–136.

Farquhar, G. D., S. von Caemmerer, and J. A. Berry. 1980. A biochemical model of photosynthetic CO2 assimilation in leaves of C3 species. Planta 149:78–90.

Leuning, R. 2002. Temperature dependence of two parameters in a photosynthesis model. Plant, Cell and Environment 25:1205–1210.

Sperry, J. S., M. D. Venturas, W. R. L. Anderegg, M. Mencuccini, D. S. Mackay, Y. Wang, and D. M. Love. 2016. Predicting stomatal responses to the environment from the optimization of photosynthetic gain and hydraulic cost. Plant Cell and Environment.

See Also

`hydraulics.supplyFunctionNetwork`, `biophysics.leafTemperature`, `spwb`

`Plant values` *Plant description functions*

Description

Functions to calculate attributes of plants in a `forest` object.

Usage

```
plant.BasalArea(x)
plant.LargerTreeBasalArea(x)
plant.CharacterParameter(x, SpParams, parName)
plant.Cover(x)
plant.CrownBaseHeight(x, SpParams)
plant.CrownLength(x, SpParams)
plant.CrownRatio(x, SpParams)
plant.Density(x, SpParams)
plant.EquilibriumLeafLitter(x, SpParams, AET = 800)
plant.EquilibriumSmallBranchLitter(x, SpParams,
                                   smallBranchDecompositionRate = 0.81)
plant.FoliarBiomass(x, SpParams, gdd = NA)
plant.Fuel(x, SpParams, gdd = NA, includeDead = TRUE)
plant.Height(x)
plant.ID(x)
plant.LAI(x, SpParams, gdd = NA)
plant.Parameter(x, SpParams, parName)
plant.Phytovolume(x, SpParams)
plant.Species(x)
plant.SpeciesName(x, SpParams)
```

Arguments

`x`	An object of class `forest`.
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`parName`	A string with a parameter name.
`gdd`	Growth degree days (to account for leaf phenology effects).
`AET`	Actual annual evapotranspiration (in mm).
`smallBranchDecompositionRate`	Decomposition rate of small branches.
`includeDead`	A flag to indicate that standing dead fuels (dead branches) are included.

Value

A vector with values for each plant of the input `forest` object:

- `plant.BasalArea`: Tree basal area (m2/ha).

- `plant.LargerTreeBasalArea`: Basal area (m2/ha) of trees larger (in diameter) than the tree. Half of the trees of the same record are included.
- `plant.CharacterParameter`: The parameter values of each plant, as strings.
- `plant.Cover`: Shrub cover (in percent).
- `plant.CrownBaseHeight`: The height corresponding to the start of the crown (in cm).
- `plant.CrownLength`: The difference between crown base height and total height (in cm).
- `plant.CrownRatio`: The ratio between crown length and total height (between 0 and 1).
- `plant.Density`: Plant density (ind/ha). Tree density is directly taken from the forest object, while the shrub density is estimated from cover and height by calculating the area of a single individual.
- `plant.EquilibriumLeafLitter`: Litter biomass of leaves at equilibrium (in kg/m2).
- `plant.EquilibriumSmallBranchLitter`: Litter biomass of small branches (< 6.35 mm diameter) at equilibrium (in kg/m2).
- `plant.FoliarBiomass`: Standing biomass of leaves (in kg/m2).
- `plant.Fuel`: Fine fuel load (in kg/m2).
- `plant.Height`: Total height (in cm).
- `plant.ID`: Cohort coding for simulation functions (concatenation of 'T' (Trees) or 'S' (Shrub), cohort index and species index).
- `plant.LAI`: Leaf area index (m2/m2).
- `plant.Parameter`: The parameter values of each plant, as numeric.
- `plant.Phytovolume`: Shrub phytovolume (m3/m2).
- `plant.Species`: Species identity integer (indices start with 0).
- `plant.SpeciesName`: String with species taxonomic name (or a functional group).

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

`spwb`, `forest`, `summary.forest`

Examples

```
#Default species parameterization
data(SpParamsMED)

#Load example plot
data(exampleforest)

#A short way to obtain total basal area
sum(plant.BasalArea(exampleforest), na.rm=TRUE)

#The same forest level function for LAI
sum(plant.LAI(exampleforest, SpParamsMED))
```

```
#The same forest level function for fuel loading
sum(plant.Fuel(exampleforest, SpParamsMED))

#Summary function for 'forest' objects can be also used
summary(exampleforest, SpParamsMED)

plant.SpeciesName(exampleforest, SpParamsMED)

plant.ID(exampleforest)
```

`plot.spwb` *Displays simulation results*

Description

Function `plot` plots the results of the soil water balance model (see `spwb`) or the forest growth model (see `growth`), whereas function `summary` summarizes the model's output in different temporal steps (i.e. weekly, annual, ...).

Usage

```
## S3 method for class 'spwb'
plot(x, type="PET_Precipitation", bySpecies = FALSE,
                  yearAxis=FALSE, xlim = NULL, ylim=NULL, xlab=NULL, ylab=NULL,
                   add = FALSE, ...)
## S3 method for class 'growth'
plot(x, type="PET_Precipitation", bySpecies = FALSE, yearAxis=FALSE,
    xlim = NULL, ylim=NULL, xlab=NULL, ylab=NULL,...)
## S3 method for class 'spwb'
summary(object, freq="years", output="WaterBalance", FUN=sum, bySpecies = FALSE, ...)
## S3 method for class 'growth'
summary(object, freq="years", output="WaterBalance", FUN=sum, bySpecies = FALSE, ...)
```

Arguments

`x, object`	An object of class `spwb`.
`type`	The information to be plotted: • `"PET_Precipitation"`: Potential evapotranspiration and Precipitation. • `"PET_NetRain"`: Potential evapotranspiration and Net rainfall. • `"Snow"`: Snow precipitation and snowpack dynamics. • `"Export"`: Water exported through deep drainage and surface runoff. • `"Evapotranspiration"`: Plant transpiration and soil evaporation. • `"SoilPsi"`: Soil water potential. • `"SoilTheta"`: Soil relative water content. • `"SoilVol"`: Soil water volumetric content.

- "PlantExtraction": Water extracted by plants from each soil layer.
- "HydraulicRedistribution":Water added to each soil layer coming from other soil layers, transported through the plant hydraulic network (only for transpirationMode = "Complex").
- "WTD": Water table depth.
- "LAI": Expanded and dead leaf area index of the whole stand.
- "PlantLAI": Plant cohort leaf area index (expanded leaves).
- "SoilPlantConductance":Average instantaneous overall soil plant conductance (calculated as the derivative of the supply function).
- "PlantStress": Plant cohort average daily drought stress.
- "PlantPsi": Plant cohort water potential (only for transpirationMode = "Simple").
- "LeafPsi": Midday leaf water potential (only for transpirationMode = "Complex").
- "StemPsi": Midday (upper) stem water potential (only for transpirationMode = "Complex").
- "RootPsi": Midday root crown water potential (only for transpirationMode = "Complex").
- "PlantTranspiration": Plant cohort transpiration.
- "PlantTranspirationLeaf": Plant cohort transpiration per leaf area.
- "PlantPhotosynthesis": Plant cohort photosynthesis.
- "PlantPhotosynthesisLeaf": Plant cohort photosynthesis per leaf area.
- "PlantWUE": Plant cohort daily water use efficiency (photosynthesis over transpiration).
- "PlantAbsorbedSWR": Plant cohort absorbed short wave radiation (only for transpirationMode = "Complex").
- "PlantAbsorbedSWRLeaf": Plant cohort absorbed short wave radiation per leaf area (only for transpirationMode = "Complex").
- "PlantAbsorbedLWR": Plant cohort absorbed long wave radiation (only for transpirationMode = "Complex").
- "PlantAbsorbedLWRLeaf": Plant cohort absorbed long wave radiation per leaf area (only for transpirationMode = "Complex").
- "AirTemperature": Minimum/maximum/mean daily temperatures above canopy (only for transpirationMode = "Complex").
- "CanopyTemperature": Minimum/maximum/mean daily temperatures inside canopy (only for transpirationMode = "Complex").
- "SoilTemperature": Minimum/maximum/mean daily temperatures inside the first soil layer (only for transpirationMode = "Complex").
- "CanopyEnergyBalance": Canopy energy balance components (only for transpirationMode = "Complex").
- "SoilEnergyBalance": Soil energy balance components (only for transpirationMode = "Comple
- "PlantRespiration": Plant cohort respiration (only for plot.growth).
- "PlantRespirationLeaf": Plant cohort respiration per leaf area (only for plot.growth).
- "PlantCBalance": Plant cohort carbon balance (only for plot.growth).
- "PlantCBalanceLeaf": Plant cohort carbon balance per leaf area (only for plot.growth).
- "PlantCstorageFast": Amount of fast-dynamics carbon reserves (only for plot.growth).

- "PlantCstorageSlow": Amount of slow-dynamics carbon reserves (only for plot.growth).
- "PlantSA": Amount of sapwood area in an individual (only for plot.growth).
- "PlantSAgrowth": Amount of newly-created sapwood area (only for plot.growth).
- "PlantRelativeSAgrowth": Amount of newly-created sapwood area per sapwood area (only for plot.growth).
- "PlantLAIlive": Plant cohort leaf area index of live leaves (only for plot.growth).
- "PlantLAIdead": Plant cohort leaf area index of dead leaves (only for plot.growth).

bySpecies — Allows aggregating output by species, before calculating summaries or drawing plots (only has an effect with some values of type). Aggregation can involve a sum (as for plant lai or transpiration) or a LAI-weighted mean (as for plant stress or plant water potential).

yearAxis — A boolean to indicate whether the units of the x-axis are years (by default they are dates).

xlim — Range of values for x.

ylim — Range of values for y.

xlab — x-axis label.

ylab — y-axis label.

add — Boolean flag to add new elements to the current plot.

freq — Frequency of summary statistics (see cut.Date).

output — The data table to be summarized. Accepted values are "DailyBalance", "PlantStress", "PlantPsi", "PlantTranspiration", "PlantPhotosynthesis" and "SoilWaterBalance", "Temperature" and "EnergyBalance".

FUN — The function to summarize results (e.g., sum, mean, ...)

... — Additional parameters for functions plot or summary.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

De Cáceres M, Martínez-Vilalta J, Coll L, Llorens P, Casals P, Poyatos R, Pausas JG, Brotons L. (2015) Coupling a water balance model with forest inventory data to predict drought stress: the role of forest structural changes vs. climate changes. Agricultural and Forest Meteorology 213: 77-90 (doi:10.1016/j.agrformet.2015.06.012).

See Also

spwb, spwbpoints

Examples

```
#Load example daily meteorological data
data(examplemeteo)

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Initialize soil with default soil params (2 layers)
examplesoil = soil(defaultSoilParams(2))

#Initialize control parameters
control = defaultControl()

#Initialize input
x = forest2spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S1<-spwb(x, examplesoil, examplemeteo, elevation = 100)

#Plot results
plot(S1)

#Monthly summary (averages) of soil status
summary(S1, freq="months",FUN=mean, output="Soil")
```

---

plot.spwb.day *Displays simulation results for one day*

Description

Function `plot` plots the results of the soil water balance model for one day (see `spwb.day`).

Usage

```
## S3 method for class 'spwb.day'
plot(x, type="PlantTranspiration", bySpecies = FALSE, xlab = NULL, ylab = NULL, ...)
```

Arguments

`x`	An object of class `spwb`
`type`	The information to be plotted: • `"LeafPsi"`:Leaf water potential. • `"StemPsi"`:(Upper) stem water potential. • `"RootPsi"`:Root crown water potential.

- "StemPLC":(Average) percentage of loss conductance in the stem conduits.
- "StemRWC":(Average) relative water content in the stem symplasm.
- "LeafRWC":Relative water content in the leaf symplasm.
- "SoilPlantConductance":Overall soil plant conductance (calculated as the derivative of the supply function).
- "PlantExtraction": Water extracted from each soil layer.
- "PlantTranspiration": Plant cohort transpiration per ground area.
- "LeafTranspiration": Plant cohort transpiration per leaf area.
- "PlantWaterBalance": Difference between water extraction from the soil and transpired water per ground area.
- "PlantPhotosynthesis": Plant cohort net photosynthesis per ground area.
- "LeafPhotosynthesis":Net photosynthesis per leaf area.
- "PlantAbsorbedSWR":Cohort's absorbed short wave radiation per ground area (differentiates sunlit and shade leaves).
- "LeafAbsorbedSWR":Cohort's absorbed short wave radiation per leaf area (differentiates sunlit and shade leaves).
- "LeafVPD":Leaf vapour pressure deficit (differentiates sunlit and shade leaves).
- "LeafStomatalConductance":Leaf stomatal conductance (differentiates sunlit and shade leaves).
- "LeafTemperature":Leaf temperature (differentiates sunlit and shade leaves).
- "Temperature":Above-canopy, inside-canopy and soil temperature.
- "CanopyEnergyBalance": Canopy energy balance components.
- "SoilEnergyBalance": Soil energy balance components.

bySpecies — Allows aggregating output by species, before drawing plots. Aggregation can involve a sum (as for plant lai or transpiration) or a LAI-weighted mean (as for plant stress or plant water potential).

xlab — x-axis label.

ylab — y-axis label.

... — Additional parameters for function plot.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

spwb.day, plot.spwb

Examples

```
#Load example daily meteorological data
data(examplemeteo)

#Load example plot plant data
data(exampleforest)
```

```
#Default species parameterization
data(SpParamsMED)

#Initialize control parameters
control = defaultControl()
control$ndailysteps = 24

#Initialize soil with default soil params (2 layers)
examplesoil = soil(defaultSoilParams(2), W=c(0.5,0.5))


#Switch to 'Complex' transpiration mode
control$transpirationMode="Complex"

#Simulate one day only
x2 = forest2spwbInput(exampleforest,examplesoil, SpParamsMED, control)
d = 100
sd2<-spwb.day(x2, examplesoil, rownames(examplemeteo)[d],
              examplemeteo$MinTemperature[d], examplemeteo$MaxTemperature[d],
              examplemeteo$MinRelativeHumidity[d], examplemeteo$MaxRelativeHumidity[d],
              examplemeteo$Radiation[d], examplemeteo$WindSpeed[d],
              latitude = 41.82592, elevation = 100,
              slope= 0, aspect = 0, prec = examplemeteo$Precipitation[d])

#Display transpiration for subdaily steps
plot(sd2, "PlantTranspiration")
```

`root` *Distribution of fine roots*

Description

Functions to calculate the distribution of fine roots within the soil, given root system parameters and soil layer definition (layer widths).

Usage

```
root.conicDistribution(Zcone, d)
root.ldrDistribution(Z50, Z95, d)
root.xylemConductanceProportions(v, d, depthWidthRatio = 1)
root.rootLengths(v, d, depthWidthRatio = 1.0)
```

Arguments

`Z50`	A vector of depths (in mm) corresponding to 50% of roots.
`Z95`	A vector of depths (in mm) corresponding to 95% of roots.
`Zcone`	A vector of depths (in mm) corresponding to the root cone tip.
`d`	The width (in mm) corresponding to each soil layer.

v — Proportions of fine roots, as returned by functions `root.conicDistribution` or `root.ldrDistribution`.

`depthWidthRatio` — Ratio between radius of the soil layer with the largest radius and maximum rooting depth.

Details

Function `root.conicDistribution` assumes a conic distribution of fine roots, whereas function `root.ldrDistribution` distributes fine roots according to the linear dose response model of Schenck & Jackson (2002). Function `root.xylemConductanceProportions` calculates the proportion of total root xylem conductance that can be attributed to each layer, according to layer widths and the proportion of fine roots (Sperry et al. 2016).

Value

Functions `root.conicDistribution` and `root.ldrDistribution` return a matrix with as many rows as elements in `Z` (or `Z50`) and as many columns as soil layers. Values in all cases correspond to the proportion of fine roots in each soil layer. Function `root.xylemConductanceProportions` returns a vector of proportions of the same length as the number of layers.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Schenk, H., Jackson, R., 2002. The global biogeography of roots. Ecol. Monogr. 72, 311–328.

Sperry, J. S., Y. Wang, B. T. Wolfe, D. S. Mackay, W. R. L. Anderegg, N. G. Mcdowell, and W. T. Pockman. 2016. Pragmatic hydraulic theory predicts stomatal responses to climatic water deficits. New Phytologist 212, 577–589.

See Also

`spwb`, `spwb.ldrOptimization`, `forest2spwbInput`, `soil`

Examples

```
#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

ntree = nrow(exampleforest$treeData)

#Initialize soil with default soil params
S = soil(defaultSoilParams())

#Calculate conic root system for trees
V1 = root.conicDistribution(Z=rep(2000,ntree), S$dVec)
```

```
print(V1)

#Calculate LDR root system for trees (Schenck & Jackson 2002)
V2 = root.ldrDistribution(Z50 = rep(200,ntree),
          Z95 = rep(1000,ntree), S$dVec)
print(V2)

#Equal xylem conductance proportions for a cone distribution
#(assuming depth-width ratio 1)
root.xylemConductanceProportions(V1[1,], S$dVec)

#Xylem conductance proportions for LDR distribution
root.xylemConductanceProportions(V2[1,], S$dVec)
```

---

scalingconductance *Scaling from conductivity to conductance*

Description

Functions used to scale from tissue conductivity to conductance of different elements of the continuum.

Usage

```
hydraulics.maximumSoilPlantConductance(krhizomax, krootmax,
                                       kstemmax, kleafmax)
hydraulics.soilPlantResistances(psiSoil, psiRhizo,
                                psiStem, PLCstem, psiLeaf,
                                krhizomax, n, alpha,
                                krootmax, rootc, rootd,
                                kstemmax, stemc, stemd,
                                kleafmax, leafc, leafd)
hydraulics.averageRhizosphereResistancePercent(krhizomax, n, alpha,
                                               krootmax, rootc, rootd,
                                               kstemmax, stemc, stemd,
                                              kleafmax, leafc, leafd, psiStep = -0.01)
hydraulics.findRhizosphereMaximumConductance(averageResistancePercent, n, alpha,
                                             krootmax, rootc, rootd,
                                             kstemmax, stemc, stemd,
                                             kleafmax, leafc, leafd)
hydraulics.maximumRootHydraulicConductance(xylemConductivity, Al2As,
                                           v, widths, depthWidthRatio = 1.0)
hydraulics.maximumStemHydraulicConductance(xylemConductivity, refheight, Al2As, height,
                                           angiosperm = TRUE, taper = FALSE)
hydraulics.referenceConductivityHeightFactor(refheight, height)
hydraulics.terminalConduitRadius(height)
hydraulics.taperFactorSavage(height)
hydraulics.stemWaterCapacity(Al2As, height, wd)
hydraulics.leafWaterCapacity(SLA, ld)
```

Arguments

psiSoil	Soil water potential (in MPa). A scalar or a vector depending on the function.
psiRhizo	Water potential (in MPa) in the rhizosphere (root surface).
psiStem	Water potential (in MPa) in the stem.
psiLeaf	Water potential (in MPa) in the leaf.
PLCstem	Percent loss of conductance (in %) in the stem.
v	Proportion of fine roots within each soil layer.
krhizomax	Maximum rhizosphere hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
kleafmax	Maximum leaf hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
kstemmax	Maximum stem xylem hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
krootmax	Maximum root xylem hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
psiStep	Water potential precision (in MPa).
rootc, rootd	Parameters of the Weibull function for roots (root xylem vulnerability curve).
stemc, stemd	Parameters of the Weibull function for stems (stem xylem vulnerability curve).
leafc, leafd	Parameters of the Weibull function for leaves (leaf vulnerability curve).
n, alpha	Parameters of the Van Genuchten function (rhizosphere vulnerability curve).
averageResistancePercent	Average (across water potential values) resistance percent of the rhizosphere, with respect to total resistance (rhizosphere + root xylem + stem xylem).
xylemConductivity	Xylem conductivity as flow per length of conduit and pressure drop (in kg·m-1·s-1·MPa-1).
Al2As	Leaf area to sapwood area (in m2·m-2).
height	Plant height (in cm).
refheight	Reference plant height (in cm).
angiosperm	A boolean flag to indicate an angiosperm species.
taper	A boolean flag to indicate correction by taper of xylem conduits (Christoffersen et al. 2017).
widths	Soil layer depths (in mm).
depthWidthRatio	Ratio between radius of the soil layer with the largest radius and maximum rooting depth.
SLA	Specific leaf area (mm2·mg-1).
wd	Wood density (g·cm-3).
ld	Leaf tissue density (g·cm-3).

Details

Details of the hydraulic model are given in a vignette.

Value

Values returned for each function are:

- `hydraulics.maximumSoilPlantConductance`: The maximum soil-plant conductance, in the same units as the input segment conductances.
- `hydraulics.averageRhizosphereResistancePercent`: The average percentage of resistance due to the rhizosphere, calculated across water potential values.
- `hydraulics.findRhizosphereMaximumConductance`: The maximum rhizosphere conductance value given an average rhizosphere resistance and the vulnerability curves of rhizosphere, root and stem elements.
- `hydraulics.taperFactorSavage`: Taper factor according to Savage et al. (2010).

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Christoffersen, B. O., M. Gloor, S. Fauset, N. M. Fyllas, D. R. Galbraith, T. R. Baker, L. Rowland, R. A. Fisher, O. J. Binks, S. A. Sevanto, C. Xu, S. Jansen, B. Choat, M. Mencuccini, N. G. McDowell, and P. Meir. 2016. Linking hydraulic traits to tropical forest function in a size-structured and trait-driven model (TFS v.1-Hydro). Geoscientific Model Development Discussions 9: 4227–4255.

Savage, V. M., L. P. Bentley, B. J. Enquist, J. S. Sperry, D. D. Smith, P. B. Reich, and E. I. von Allmen. 2010. Hydraulic trade-offs and space filling enable better predictions of vascular structure and function in plants. Proceedings of the National Academy of Sciences of the United States of America 107:22722–7.

Olson, M.E., Anfodillo, T., Rosell, J.A., Petit, G., Crivellaro, A., Isnard, S., León-Gómez, C., Alvarado-Cárdenas, L.O., & Castorena, M. 2014. Universal hydraulics of the flowering plants: Vessel diameter scales with stem length across angiosperm lineages, habits and climates. Ecology Letters 17: 988–997.

See Also

`hydraulics.psi2K`, `hydraulics.supplyFunctionPlot`, `spwb`, `soil`

Examples

```
kstemmax = 4 # in mmol·m-2·s-1·MPa-1
stemc = 3
stemd = -4 # in MPa
```

SFI2SPL *Create Landscape Classes from SFI*

Description

Functions to create spatial landscape classes from Spanish Forest Inventory (SFI) data (see details). Internally, these functions make calls to `extractSFIforest`.

Usage

```
SFI2SPL(SFItreeData, SFIshrubData, SFIherbData = NULL,
                    SpatialPointsIDs, elevation, slope, aspect, SpParams,
                 SoilParamData = NULL, SFIcodes=NULL, control = defaultControl())
SFI2SGL(landTopo, SFItreeData, SFIshrubData, SpatialPointsIDs,
                 SpParams, SoilParamData = NULL, lctInput,
              forestLCTs, shrublandLCTs, grasslandLCTs, agricultureLCTs= numeric(0),
                 rockLCTs= numeric(0), staticLCTs= numeric(0),
                 SFIcodes=NULL, FMcodes = NULL, control = defaultControl())
```

Arguments

`landTopo`	An object of class `SpatialGridTopography`.
`SFItreeData`	A data frame with measured tree data.
`SFIshrubData`	A data frame with measured shrub data.
`SFIherbData`	A data frame with percent cover and mean height of the herb layer.
`SpatialPointsIDs`	An object of class `SpatialPoints-class` containing the coordinates of the forest plots. Coordinates must include row names corresponding to plot IDs.
`elevation`	A numeric vector with elevation values (in m).
`slope`	A numeric vector with slope values (in degrees).
`aspect`	A numeric vector with aspect values (in degrees from North).
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`SoilParamData`	A data frame with soil parameters for each forest stand. The rows must match `SpatialPointsIDs` or `SpatialGridIDs`, depending on the function called. If `NULL`, then the parameters given by `defaultSoilParams` are used.
`SFIcodes`	A string vector (of length equal to the number of rows in `SpParams` of the SFI species codes that correspond to the model species codification. Each string may contain different coma-separated codes in order to merge SFI species into a single model species.
`FMcodes`	A string vector with the fuel model code corresponding to each forest plot. Names of this string vector should be plot IDs.
`control`	A list of control parameters (see `defaultControl`).

`lctInput` A character vector with the land cover type input for initialization of unsurveyed cells.

`forestLCTs, shrublandLCTs, grasslandLCTs`
Integer vectors with the land cover types that are to be considered 'wildland' (i.e. forests, shrublands or grasslands). Initialization of forest and shrublands cells will make use of SFI plot data, but in shrublands only shrubs are used. Initialization of grasslands specifies an `emptyforest`.

`agricultureLCTs, rockLCTs, staticLCTs`
Integer vectors with the land cover types that are to be considered 'agriculture', 'rock' (i.e. outcrops) or 'static' (i.e. urban, water masses). Agricultural areas are like grasslands, but they can never be vegetated with wildland. Rock cells allow surface water flow over them, but do not contain vegetation nor soil and cannot be burned. Static cells do not burn and water entering them does not flow to any other cell.

Value

An object of class `SpatialPointsLandscape-class` or `SpatialGridLandscape-class` depending on the function.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

`forest`, `extractSFIforest`

SFM_metric *Standard fuel models (Albini 1976, Scott & Burgan 2005)*

Description

Standard fuel models converted to metric system. Copied from package 'Rothermel' (Giorgio Vacchiano, Davide Ascoli).

Usage

```
data("SFM_metric")
```

Format

A data frame including standard fuel models as in Albini (1976) and Scott and Burgan (2005), to be used as input of `fire.Rothermel` function. All values converted to metric format.

`Fuel_Model_Type` A factor with levels `D` (for dynamic) or `S` (for static).

`Load_1h` Loading of 1h fuel class [t/ha].

`Load_10h` Loading of 10h fuel class [t/ha].

`Load_100h` Loading of 100h fuel class [t/ha]

`Load_Live_Herb` Loading of herbaceous fuels [t/ha]

`Load_Live_Woody` Loading of woody fuels [t/ha]

'`SA/V_1h`' Surface area to volume ratio of 1h fuel class [m2/m3]

'`SA/V_10h`' Surface area to volume ratio of 10h fuel class [m2/m3]

'`SA/V_100h`' Surface area to volume ratio of 100h fuel class [m2/m3]

'`SA/V_Live_Herb`' Surface area to volume ratio of herbaceous fuels [m2/m3]

'`SA/V_Live_Woody`' Surface area to volume ratio of woody fuels [m2/m3]

`Fuel_Bed_Depth` Fuel bed depth [cm]

`Mx_dead` Dead fuel moisture of extinction [percent]

`Heat_1h` Heat content of 1h fuel class [kJ/kg]

`Heat_10h` Heat content of 10h fuel class [kJ/kg]

`Heat_100h` Heat content of 100h fuel class [kJ/kg]

`Heat_Live_Herb` Heat content of herbaceous fuels [kJ/kg]

`Heat_Live_Woody` Heat content of woody fuels [kJ/kg]

Source

Albini, F. A. (1976). Computer-based models of wildland fire behavior: A users' manual. Ogden, UT: US Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station.

Scott, J., & Burgan, R. E. (2005). A new set of standard fire behavior fuel models for use with Rothermel's surface fire spread model. Gen. Tech. Rep. RMRSGTR-153. Fort Collins, CO: US Department of Agriculture, Forest Service, Rocky Mountain Research Station.

See Also

`fire.Rothermel`

Examples

```
data(SFM_metric)
```

soil *Soil initialization*

Description

Initializes soil parameters and state variables for its use in simulations.

Usage

```
soil(SoilParams, VG_PTF = "Carsel", W = as.numeric(c(1)), SWE = 0)
## S3 method for class 'soil'
print(x, model="SX", ...)
```

Arguments

SoilParams	A list of soil parameters (see defaultSoilParams).
VG_PTF	Pedotransfer functions to obtain parameters for the van Genuchten-Mualem equations. Either "Carsel" (Carsel & Parrish 1988) or "Toth" (Toth et al. 2015).
W	A numerical vector with the initial relative water content of each soil layer.
SWE	Initial snow water equivalent of the snow pack on the soil surface (mm).
x	An object of class soil.
model	Either 'SX' or 'VG' for Saxton or Van Genuchten pedotransfer models.
...	Additional parameters to print.

Details

Function print prompts a description of soil characteristics and state variables (water content and temperature) according to a water retention curve (either Saxton's or Van Genuchten's). Volume at field capacity is calculated assuming a soil water potential equal to -0.033 MPa. Parameter Temp is initialized as missing for all soil layers.

Value

An list of class soil with the following elements:

- SoilDepth: Soil depth (in mm).
- W: State variable with relative water content of each layer (in as proportion relative to FC).
- Temp: State variable with temperature (in ºC) of each layer.
- Ksoil: Kappa parameter for infiltration.
- Gsoil: Gamma parameter for infiltration.
- dVec: Width of soil layers (in mm).
- sand: Sand percentage for each layer (in percent volume).
- clay: Clay percentage for each layer (in percent volume).

- `om`: Organic matter percentage for each layer (in percent volume).
- `usda_Type`: USDA texture type.
- `VG_alpha`, `VG_n`, `VG_theta_res`, `VG_theta_sat`: Parameters for van Genuchten's pedotransfer functions, for each layer, corresponding to the USDA texture type.
- `macro`: Macroporosity for each layer (estimated using Stolf et al. 2011).
- `rfc`: Percentage of rock fragment content for each layer.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Carsel, R.F., & Parrish, R.S. 1988. Developing joint probability distributions of soil water retention characteristics. Water Resources Research 24: 755–769.

Tóth, B., Weynants, M., Nemes, A., Makó, A., Bilas, G., & Tóth, G. 2015. New generation of hydraulic pedotransfer functions for Europe. European Journal of Soil Science 66: 226–238.

Stolf, R., Thurler, A., Oliveira, O., Bacchi, S., Reichardt, K., 2011. Method to estimate soil macroporosity and microporosity based on sand content and bulk density. Rev. Bras. Ciencias do Solo 35, 447–459.

See Also

`soil.psi2thetaSX`, `soil.psi2thetaVG`, `spwb`, `defaultSoilParams`

Examples

```
# Initializes soil
s = soil(defaultSoilParams())

# Prints soil characteristics according to Saxton's water retention curve
print(s, model="SX")

# Prints soil characteristics according to Van Genuchten's water retention curve
print(s, model="VG")
```

`soil texture and hydraulics`

Soil texture and hydraulics

Description

Functions soil.psi2thetaSX and soil.theta2psiSX calculate water potentials (psi) and water contents (theta) using texture data the formulae of Saxton et al. (1986) or Saxton & Rawls (2006) depending on whether organic matter is available. Functions codesoil.psi2thetaVG and soil.theta2psiVG to the same calculations as before, but using the Van Genuchten - Mualem equations (Wösten & van Genuchten 1988). Function soil.USDAType returns the USDA type for a given texture. Function soil.vanGenuchtenParamsCarsel gives parameters for van Genuchten-Mualem equations for a given texture type (Leij et al. 1996), whereas function soil.vanGenuchtenParamsToth gives parameters for van Genuchten-Mualem equations for a given texture, organic matter and bulk density (Toth et al. 2015). Correspondingly, functions soil.waterFC and soil.thetaFC calculate the water volume (in mm or as percent of soil volume) of each soil layer at field capacity, according to a given water retention model. Functions soil.waterWP and soil.thetaWP calculate the water volume (in mm or as percent of soil volume) of each soil layer at wilting point (-1.5 MPa), and functions soil.waterSAT, soil.thetaSATSX and soil.thetaSAT calculate the saturated water volume (in mm or as percent of soil volume) of each soil layer. Functions soil.psi and soil.theta return the current water potential and water content of the soil object, according to a given water retention model.

Usage

```
soil.psi2thetaSX(clay, sand, psi, om = NA)
soil.psi2thetaVG(n, alpha, theta_res, theta_sat, psi)
soil.theta2psiSX(clay, sand, theta, om = NA)
soil.theta2psiVG(n, alpha, theta_res, theta_sat, theta)
soil.USDAType(clay, sand)
soil.vanGenuchtenParamsCarsel(soilType)
soil.vanGenuchtenParamsToth(clay, sand, om, bd, topsoil)
soil.psi(soil, model="SX")
soil.theta(soil, model="SX")
soil.waterFC(soil, model="SX")
soil.waterWP(soil, model="SX")
soil.waterSAT(soil, model="SX")
soil.thetaFC(soil, model="SX")
soil.thetaWP(soil, model="SX")
soil.thetaSAT(soil, model="SX")
soil.thetaSATSX(clay, sand, om = NA)
soil.waterTableDepth(soil, model="SX")
```

Arguments

clay	Percentage of clay (in percent weight).
sand	Percentage of sand (in percent weight).
n, alpha, theta_res, theta_sat	Parameters of the Van Genuchten-Mualem model (m = 1 - 1/n).
psi	Water potential (in MPa).
theta	Relative water content (in percent volume).
om	Percentage of organic matter (optional, in percent weight).

`bd`	Bulk density (in g/cm3).
`topsoil`	A boolean flag to indicate topsoil layer.
`soilType`	A string indicating the soil type.
`soil`	Soil object (returned by function `soil`).
`model`	Either 'SX' or 'VG' for Saxton's or Van Genuchten's water retention models.

Value

Functions `soil.psi2thetaSX` and `soil.psi2thetaVG` return the soil water potential (in MPa) from soil volumetric water content, and functions `soil.theta2psiSX` and `soil.theta2psiVG` do the reverse calculation returning water potential in MPa. Function `soil.USDAType` returns a string. Function `soil.vanGenuchtenParamsToth` and `soil.vanGenuchtenParamsCarsel` return a vector with four parameter values (alpha, n, theta_res and theta_sat, where alpha is in MPa-1). Function `soil.waterTableDepth` returns water table depth in mm from surface.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Leij, F.J., Alves, W.J., Genuchten, M.T. Van, Williams, J.R., 1996. The UNSODA Unsaturated Soil Hydraulic Database User's Manual Version 1.0.

Saxton, K.E., Rawls, W.J., Romberger, J.S., Papendick, R.I., 1986. Estimating generalized soil-water characteristics from texture. Soil Sci. Soc. Am. J. 50, 1031–1036.

Saxton, K.E., Rawls, W.J., 2006. Soil water characteristic estimates by texture and organic matter for hydrologic solutions. Soil Sci. Soc. Am. J. 70, 1569. doi:10.2136/sssaj2005.0117

Wösten, J.H.M., & van Genuchten, M.T. 1988. Using texture and other soil properties to predict the unsaturated soil hydraulic functions. Soil Science Society of America Journal 52: 1762–1770.

Tóth, B., Weynants, M., Nemes, A., Makó, A., Bilas, G., & Tóth, G. 2015. New generation of hydraulic pedotransfer functions for Europe. European Journal of Soil Science 66: 226–238.

See Also

`soil`

Examples

```
# Plot Saxton's water retention curve
psi = seq(0, -6.0, by=-0.01)
plot(-psi, lapply(as.list(psi), FUN=soil.psi2thetaSX, clay=40, sand=10, om = 1),
     type="l", ylim=c(0,0.6),ylab="Water content (prop. volume)",
     xlab = "Soil water potential (-MPa)")

#Determine USDA soil texture type
type = soil.USDAType(clay=40, sand=10)
type
```

```
#Van Genuchten's params (bulk density = 1.3 g/cm)
vg = soil.vanGenuchtenParamsToth(40,10,1,1.3,TRUE)
vg

#Add Van Genuchten water retention curve
lines(-psi, lapply(as.list(psi), FUN=soil.psi2thetaVG, alpha=vg[1], n = vg[2],
      theta_res = vg[3], theta_sat = vg[4]), lty=2)

legend("topright", legend=c("Saxton", "Van Genuchten"), lty=c(1,2), bty="n")
```

`soil thermodynamics` *Soil thermodynamic functions*

Description

Functions `soil.thermalconductivity` and `soil.thermalcapacity` calculate thermal conductivity and thermal capacity for each soil layer, given its texture and water content. Functions `soil.temperaturegradient` and `soil.temperaturechange` are used to calculate soil temperature gradients (in ºC/m) and temporal temperature change (in ºC/s) given soil layer texture and water content (and possibly including heat flux from above).

Usage

```
soil.thermalconductivity(soil, model = "SX")
soil.thermalcapacity(soil, model = "SX")
soil.temperaturechange(dVec, Temp, sand, clay, W, Theta_FC, Gdown)
soil.temperaturegradient(dVec, Temp)
```

Arguments

`soil`	Soil object (returned by function `soil`).
`model`	Either 'SX' or 'VG' for Saxton's or Van Genuchten's pedotransfer models.
`dVec`	Width of soil layers (in mm).
`Temp`	Temperature (in ºC) for each soil layer.
`clay`	Percentage of clay (in percent weight) for each layer.
`sand`	Percentage of sand (in percent weight) for each layer.
`W`	Soil moisture (in percent of field capacity) for each layer.
`Theta_FC`	Relative water content (in percent volume) at field capacity for each layer.
`Gdown`	Downward heat flux from canopy to soil (in W·m-2).

Value

Function `soil.thermalconductivity` returns a vector with values of thermal conductivity (W/m/ºK) for each soil layer. Function `soil.thermalcapacity` returns a vector with values of heat storage capacity (J/m3/ºK) for each soil layer. Function `soil.temperaturegradient` returns a vector with values of temperature gradient between consecutive soil layers. Function `soil.temperaturechange` returns a vector with values of instantaneous temperature change (ºC/s) for each soil layer.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Cox, P.M., Betts, R.A., Bunton, C.B., Essery, R.L.H., Rowntree, P.R., & Smith, J. 1999. The impact of new land surface physics on the GCM simulation of climate and climate sensitivity. Climate Dynamics 15: 183–203.

Dharssi, I., Vidale, P.L., Verhoef, A., MacPherson, B., Jones, C., & Best, M. 2009. New soil physical properties implemented in the Unified Model at PS18. 9–12.

See Also

soil

Examples

```
examplesoil = soil(defaultSoilParams())
soil.thermalconductivity(examplesoil)
soil.thermalcapacity(examplesoil)

#Values change when altering water content (drier layers have lower conductivity and capacity)
examplesoil$W = c(0.1, 0.4, 0.7, 1.0)
soil.thermalconductivity(examplesoil)
soil.thermalcapacity(examplesoil)
```

---

soilgridsParams *SoilGrids soil description fetcher*

Description

soilgridsParams takes a vector of depths and returns a list of soil characteristics ready to use with soil function.

Usage

```
soilgridsParams(lat, long, depths = c(300, 500, 1200))
```

Arguments

lat	Latitude in decimal format
long	Longitude in decimal format
depths	A numeric vector indicating the desired depths, in *mm*

Details

This function connects with the SoilGrids REST API (https://rest.soilgrids.org) to retrieve the soil physical and chemical characteristics for a site, selected by its coordinates. Also, in case the depths are not the default ones in the SoilGrids API, the function uses the trapezoidal rule to calculate the values for the desired depths (as described in Hengl *et al*. 2007). To do this, it uses internally the `rjson` and `GSIF` R packages. Although soil layer definition is taken from the input, the function includes absolute depth to bedrock and depth to the R horizon from SoilGrids, in case the user wants to redefine soil definition.

Value

A list with the following elements:

- `soilparams`: A data frame containing the soil characteristics ready to use with the `soil` function.
- `soilgrids_Rhorizondepth`: Depth to the R horizon (in mm).
- `soilgrids_Rhorizonprob`: Probability of the R horizon.
- `soilgrids_absolutesoildepth`: Absolute depth to bedrock (in mm).

Author(s)

Víctor Granda, Centre Tecnologic Forestal de Catalunya

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Hengl T, Mendes de Jesus J, Heuvelink GBM, Ruiperez Gonzalez M, Kilibarda M, Blagotić A, et al. (2017) SoilGrids250m: Global gridded soil information based on machine learning. PLoS ONE 12(2): e0169748. doi:10.1371/journal.pone.0169748.

See Also

`soil`, `defaultSoilParams`

Examples

```
  ## Not run:
  foo <- soilgridsParams(lat = 42.6667, long = -5.6333, depths = c(300, 600, 1200))
  foo_soil <- soil(foo$soilparams)
  foo_soil

## End(Not run)
```

spatialForestSummary *Forest and soil summaries over space*

Description

Functions to calculates a summary function for the forest or soil of all cells in a SpatialPointsLandscape-class or SpatialGridLandscape-class.

Usage

```
spatialForestSummary(object, summaryFunction, ...)
spatialSoilSummary(object, summaryFunction, ...)
```

Arguments

object	An object of class SpatialPointsLandscape-class, SpatialPixelsLandscape-class or SpatialGridLandscape-class.
summaryFunction	A function that accepts objects of class forest or soil, respectively.
...	Additional arguments to the summary function.

Value

An object of class SpatialPointsDataFrame, SpatialPixelsDataFrame or SpatialGridDataFrame, depending on the input, containing the calculated statistics.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya.

See Also

forest, soil, summary.forest

Examples

```
#Load plot data
data(exampleSPL)

#Load species parameters
data(SpParamsMED)

#Apply summary function
y <- spatialForestSummary(exampleSPL,summary.forest, SpParamsMED)
head(y@data)

#Plot basal area
spplot(y["BA"])
```

SpatialGridLandscape-class
Class "SpatialGridLandscape"

Description

An S4 class that represents a landscape configuration over a grid of coordinates.

Objects from the Class

Objects can be created by calls of the form `new("SpatialGridLandscape", ...)`, or by calls to the function `SpatialGridLandscape`.

Slots

grid: Object of class "GridTopology".

bbox: Object of class "matrix" with the boundary box.

proj4string: Object of class "CRS" with the projection string.

data: Object of class "data.frame" containing the elevation (in m), slope (in degrees) and aspect (in degrees from North) of every cell.

lct: A character vector of land cover type for each cell. Values allowed are: 'wildland', 'agriculture', 'rock' and 'static'.

forestlist: Object of class "list" containing a set of `forest` objects.

soillist: Object of class "list" containing a set of `soil` objects.

waterOrder: A numeric vector of cell processing order.

waterQ: A list of water discharge values to neighbours.

queenNeigh: A list of neighbours for each cell.

Extends

Class "SpatialGridTopography", directly. Class "SpatialGridDataFrame", distance 2. Class "SpatialGrid", distance 3. Class "Spatial", by class "SpatialGrid", distance 4.

Methods

getLCTs `signature(object = "SpatialGridLandscape")`: returns a SpatialGridDataFrame with the land cover types of the landscape cells.

spatialForestSummary `signature(object = "SpatialGridLandscape")`: calculates a summary function for all forest stands and returns an object of class `SpatialGridDataFrame-class`.

spatialSoilSummary `signature(object = "SpatialGridLandscape")`: calculates a summary function for the soil of all stands and returns an object of class `SpatialGridDataFrame-class`.

spplot `signature(object = "SpatialGridLandscape")`: allows plotting maps of the landscape state.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

SpatialGridTopography-class, SpatialPointsLandscape-class, spatialForestSummary, forest, soil

Examples

```
#Structure of the S4 object
showClass("SpatialGridLandscape")
```

SpatialPixelsLandscape-class

Class "SpatialPixelsLandscape"

Description

An S4 class that represents a landscape configuration over an (incomplete) grid of coordinates.

Objects from the Class

Objects can be created by calls of the form `new("SpatialPixelsLandscape", ...)`.

Slots

data: Object of class "data.frame" containing the elevation (in m), slope (in degrees) and aspect (in degrees from North) of every cell.

coords.nrs: Inherited from SpatialPointsDataFrame but not used.

grid: Object of class "GridTopology".

grid.index: Index of points in full grid.

bbox: Object of class "matrix" with the boundary box.

proj4string: Object of class "CRS" with the projection string.

lct: A character vector of land cover type for each cell. Values allowed are: 'wildland', 'agriculture', 'rock' and 'static'.

forestlist: Object of class "list" containing a set of forest objects.

soillist: Object of class "list" containing a set of soil objects.

waterOrder: A numeric vector of cell processing order.

waterQ: A list of water discharge values to neighbours.

queenNeigh: A list of (queen) neighbours for each cell.

Extends

Class "SpatialPixelsTopography", directly. Class "SpatialPixelsDataFrame", distance 2. Class "SpatialPixels", distance 3. Class "SpatialPointsDataFrame", distance 3. Class "SpatialPoints", distance 4. Class "Spatial", by class "SpatialPixelsTopography", distance 5.

Methods

getLCTs `signature(object = "SpatialPixelsLandscape")`: returns a SpatialPixelsDataFrame with the land cover types of the landscape cells.

spatialForestSummary `signature(object = "SpatialPixelsLandscape")`: calculates a summary function for all forest stands and returns an object of class SpatialPixelsDataFrame-class.

spatialSoilSummary `signature(object = "SpatialPixelsLandscape")`: calculates a summary function for the soil of all stands and returns an object of class SpatialPixelsDataFrame-class.

spplot `signature(object = "SpatialPixelsLandscape")`: allows plotting maps of the landscape state.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

SpatialPixelsTopography-class, SpatialPointsLandscape-class, spatialForestSummary, forest, soil

Examples

```
#Structure of the S4 object
showClass("SpatialPixelsLandscape")
```

SpatialPointsLandscape

Creates Spatial Landscape Classes

Description

Functions to initialize spatial landscape classes.

Usage

```
SpatialPointsLandscape(spt, forestlist, soillist)
SpatialPixelsLandscape(spxt, lct, forestlist, soillist, verbose=TRUE)
SpatialGridLandscape(sgt, lct, forestlist, soillist, verbose=TRUE)
```

Arguments

spt	An object of class SpatialPointsTopography.
spxt	An object of class SpatialPixelsTopography.
sgt	An object of class SpatialGridTopography.
lct	A character vector with the land cover type of each grid cell (values should be 'wildland', 'agriculture', 'rock' or 'static').
forestlist	A list of objects of class 'forest' with the same number of elements as spatial points/pixels.
soillist	A list of objects of class 'forest' with the same number of elements as spatial points/pixels.
verbose	A flag to specify console output during spatial object initialization.

Value

An object of class SpatialPointsLandscape-class, SpatialPixelsLandscape-class or SpatialGridLandscape-class depending on the function.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

forest, soil

SpatialPointsLandscape-class

Class "SpatialPointsLandscape"

Description

An S4 class that represents a set of forest stands along with their spatial location

Objects from the Class

Objects can be created by calls of the form `new("SpatialPointsLandscape", ...)` or by calls to function SpatialPointsLandscape.

Slots

forestlist: Object of class "list" containing a set of forest objects.

soillist: Object of class "list" containing a set of soil objects.

data: Object of class "data.frame" containing the elevation (in m), slope (in degrees) and aspect (in degrees from North) of every cell.

coords: Object of class "matrix" with spatial coordinates.

bbox: Object of class "matrix" with the boundary box.

proj4string: Object of class "CRS" with the projection string.

Extends

Class "SpatialPointsTopography", directly. Class "SpatialPointsDataFrame", distance 2. Class "SpatialPoints", distance 3. Class "Spatial", by class "SpatialPoints", distance 4.

Methods

[`signature(x = "SpatialPointsLandscape", i = "ANY", j = "ANY", drop = "ANY")`: subsets the points, the corresponding topography data, and the forest and soil lists; only rows (points can be subsetted.

head `signature(x = "SpatialPointsLandscape", n = "numeric", ... = "ANY")`: returns the first n points of object x.

tail `signature(x = "SpatialPointsLandscape", n = "numeric", ... = "ANY")`: returns the last n points of object x.

print `signature(x = "SpatialPointsLandscape", ...= "ANY", digits = "numeric")`: prints object x.

show `signature(object = "SpatialPointsLandscape")`: prints object x.

spatialForestSummary `signature(object = "SpatialPointsLandscape")`: calculates a summary function for all forest stands and returns an object of class SpatialPointsDataFrame-class.

spatialSoilSummary `signature(object = "SpatialPointsLandscape")`: calculates a summary function for the soil of all stands and returns an object of class SpatialPointsDataFrame-class.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

SpatialPointsTopography, SpatialPointsLandscape, SpatialGridLandscape, spatialForestSummary, forest

Examples

```
showClass("SpatialPointsLandscape")
```

Species values *Species description functions*

Description

Functions to calculate attributes of a forest object by species.

Usage

```
species.BasalArea(x, SpParams)
```

Arguments

x	An object of class forest.
SpParams	A data frame with species parameters (see SpParamsMED).

Value

A vector with values for each species in SpParams:

- species.BasalArea: Species basal area (m2/ha).

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

spwb, forest, plant.BasalArea, summary.forest

Examples

```
#Default species parameterization
data(SpParamsMED)

#Load example plot
data(exampleforest)

#Species basal area
species.BasalArea(exampleforest, SpParamsMED)
```

SpParamsMED *Parameter values for Mediterranean species*

Description

A data set of parameter values for Mediterranean species, resulting from bibliographic search, fit to empirical data or expert-based guesses.

Usage

```
data("SpParamsMED")
```

Format

A data frame with 89 observations (species) on the following 71 variables.

`Name` A factor with taxon names (mostly species names).

`IFNcodes` A factor with levels corresponding to species codes in the Third Spanish forest inventory (DGCN 2005.)

`SpIndex` A numeric vector of the species index.

`Group` Either "Gymnosperm" or "Angiosperm".

`Order` Taxonomical order.

`Family` Taxonomical family.

`GrowthForm` A factor with levels `Shrub`, `Tree` and `Tree/Shrub`.

`TreeType` A factor with levels `Conifer`, `Deciduous`, `Evergreen` or `Shrub`

`Hmed` Median plant height (in cm).

`Hmax` Maximum plant height (in cm).

`Z50` Rooting depth (in mm) corresponding to 50% of fine roots.

`Z95` Rooting depth (in mm) corresponding to 95% of fine roots.

`Zmax` Maximum rooting depth (in mm).

`a_ash` Regression coefficient relating the square of shrub height with shrub area.

`a_bsh`, `b_bsh` Allometric coefficients relating phytovolume with dry weight of shrub individuals.

`cr` Ratio between crown length and total height (for shrubs).

`a_fbt`, `b_fbt`, `c_fbt`, `d_fbt` Regression coefficients used to calculate foliar biomass of an individual tree from its dbh and the cummulative basal area of larger trees.

`a_cr`, `b_1cr`, `b_2cr`, `b_3cr`, `c_1cr`, `c_2cr` Regression coefficients used to calculate crown ratio of trees.

`a_cw`, `b_cw` Regression coefficients used to calculated crown width of trees.

`SLA` Specific leaf area (mm2/mg = m2/kg).

`LeafDensity` Density of leaf tissue (dry weight over volume).

`r635` Ratio between the weight of leaves plus branches and the weight of leaves alone for branches of 6.35 mm.

`pDead` Proportion of total fine fuels that are dead

`maxFMC` Maximum fuel moisture (in percent of dry weight)

`minFMC` Minimum fuel moisture (in percent of dry weight)

`LeafPI0` Osmotic potential at full turgor of leaves (MPa).

`LeafEPS` Modulus of elasticity (capacity of the cell wall to resist changes in volume in response to changes in turgor) of leaves (MPa).

`LeafAF` Apoplastic fraction (proportion of water outside the living cells) in leaves.

`StemPI0` Osmotic potential at full turgor of symplastic xylem tissue (MPa).

`StemEPS` Modulus of elasticity (capacity of the cell wall to resist changes in volume in response to changes in turgor) of symplastic xylem tissue (Mpa).

StemAF Apoplastic fraction (proportion of water outside the living cells) in stem xylem.

Cstoragepmax Maximum storage capacity, expressed as the fraction of C per total respiratory of C.

LeafDuration Leaf duration (in years).

LigninPercent Percent of lignin+cutin over dry weight in leaves.

ParticleDensity Particle density (kg/m3).

LeafLitterFuelType Fuel type for leaf litter, with levels Broadleaved, LongLinear, Scale and ShortLinear.

Flammability Flammability modifier (either 1 or 2 for normal or high, respectively).

SAV Surface-area-to-volume ratio of the small fuel (1h) fraction (leaves and branches < 6.35mm) (m2/m3).

HeatContent High fuel heat content (kJ/kg).

fHDmin Minimum value of the height-to-diameter ratio (dimensionless).

fHDmax Maximum value of the height-to-diameter ratio (dimensionless).

albedo Leaf reflectance for short wave radiation.

k Light extinction coeficient for PAR.

g Canopy water storage capacity per LAI unit (in mm/LAI).

Sgdd Growth degree days for leaf budburst (in Celsius)

Al2As Leaf area to sapwood area ratio (in m2·m-2).

WUE Water use efficiency for carbon assimilation (g C /mm water).

WoodC Wood carbon content per dry weight (g C /g dry).

WoodDensity Wood density (at 0 percent humidity!).

RGRmax Maximum relative growth rate (in basal area or sapwood area) (in cm2·cm-2).

Psi_Extract Water potential corresponding to 50% relative conductance (in MPa).

pRootDisc Relative root conductance leading to hydraulic disconnection from a soil layer.

Gwmin Minimum stomatal conductance to water vapor per leaf area unit (in mol·s-1·m-2).

Gwmax Maximum stomatal conductance to water vapor per leaf area unit (in mol·s-1·m-2).

VCleaf_kmax Leaf hydraulic conductance (in mmol H2O·s-1·m-2·MPa-1).

VCleaf_c, VCleaf_d Parameters of the leaf vulnerability curve (VCleaf_d in MPa).

Kmax_stemxylem Sapwood-specific hydraulic conductivity of stem xylem (in kg H2O·s-1·m-1·MPa-1).

VCstem_c, VCstem_d Parameters of the stem xylem vulnerability curve (VCstem_d in MPa).

Kmax_rootxylem Sapwood-specific hydraulic conductivity of root xylem (in kg H2O·s-1·m-1·MPa-1).

VCroot_c, VCroot_d Parameters of the root xylem vulnerability curve (VCroot_d in MPa).

LeafWidth Leaf width (in cm).

Vmax298 Maximum Rubisco carboxilation rate at 25ºC (in micromol CO2·s-1·m-2).

Jmax298 Maximum electron transport rate at 25ºC (in micromol electrons·s-1·m-2).

Details

See details of parameterization in De Caceres et al. (2015) and De Caceres et al. (submitted).

Source

De Cáceres M, Martínez-Vilalta J, Coll L, Llorens P, Casals P, Poyatos R, Pausas JG, Brotons L. (2015) Coupling a water balance model with forest inventory data to predict drought stress: the role of forest structural changes vs. climate changes. Agricultural and Forest Meteorology (doi:10.1016/j.agrformet.2015.06.012).

DGCN (2005). Tercer Inventario Forestal Nacional (1997-2007): Catalunya. Dirección General de Conservación de la Naturaleza, Ministerio de Medio Ambiente, Madrid.

See Also

spwb

Examples

```
data(SpParamsMED)
```

---

spwb *Soil-plant water balance*

Description

Function spwb() is a water balance model that determines changes in soil moisture, soil water potentials and plant transpiration and drought stress at daily steps for a given forest stand during a period specified in the input climatic data. Additionally, the function also calculates plant net photosynthesis. Transpiration and photosynthesis processes are conducted with different level of detail depending on the transpiration mode.

Usage

```
spwb(x, soil, meteo,  latitude = NA, elevation = NA, slope = NA, aspect = NA)
spwb.resetInputs(x, soil, from = NULL, day = NA)
```

Arguments

x	An object of class spwbInput.
soil	A list containing the description of the soil (see soil).
meteo	A data frame with daily meteorological data series. Row names of the data frame should correspond to date strings with format "yyyy-mm-dd" (see Date). When using the 'Simple' transpiration mode the following columns are required: • MeanTemperature: Mean temperature (in degrees Celsius). • Precipitation: Precipitation (in mm). • Radiation: Solar radiation (in MJ/m2/day), required only if snowpack = TRUE.

- PET: Potential evapotranspiration (in mm).

When using the 'Complex' transpiration mode the following columns are required:

- MeanTemperature: Mean temperature (in degrees Celsius).
- MinTemperature: Minimum temperature (in degrees Celsius).
- MaxTemperature: Maximum temperature (in degrees Celsius).
- MinRelativeHumidity: Minimum relative humidity (in percent).
- MaxRelativeHumidity: Maximum relative humidity (in percent).
- Precipitation: Precipitation (in mm).
- Radiation: Solar radiation (in MJ/m2/day).
- WindSpeed: Wind speed (in m/s). If not available, this column can be left with NA values.

`latitude` Latitude (in degrees). Required when using the 'Complex' transpiration mode.

`elevation, slope, aspect`
Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North). Required when using the 'Complex' transpiration mode. Elevation is also required for 'Simple' if snowpack dynamics are simulated.

`from` An object of class spwb storing the results of a previous simulation, including values of state variables. If `from = NULL`, state variables are set to their defaults (i.e. soil moisture set to field capacity and cumulative growth degree days set to zero).

`day` An integer with the day from which state variable values stored in `from` should be copied. If missing, values are copied from the first day of stored values.

Details

Detailed model description is available in the vignettes section. The model using 'Simple' transpiration mode is described in De Caceres et al. (2015). Simulations using the 'Complex' transpiration mode are computationally much more expensive. Function `spwb.resetInputs()` allows resetting state variables in `x` and `soil` to their defaults, or to copy values of state variables from a previous `spwb()` simulation results stored in `from`.

Value

Function spwb returns a list of class 'spwb' with the following elements:

- `"spwbInput"`: A copy of the object `x` of class `spwbInput` given as input.
- `"soilInput"`: A copy of the object `soil` of class `soil` given as input.
- `"WaterBalance"`: A data frame where different variables (in columns) are given for each simulated day (in rows):
  - `"GDD"`: Cumulative growth degree days.
  - `"LAIcell"`: The LAI of the stand (accounting for leaf phenology) (in m2/m2).
  - `"Cm"`: The water retention capacity of the stand (in mm) (accounting for leaf phenology).
  - `"Lground"`: The proportion of light that reaches the ground (accounting for leaf phenology).

  - "PET": Potential evapotranspiration (in mm).
  - "Precipitation": Input precipitation (in mm).
  - "Rain": Precipitation as rain (in mm).
  - "Snow": Precipitation as snow (in mm).
  - "NetRain": Net rain, after accounting for interception (in mm).
  - "Infiltration": The amount of water infiltrating into the soil (in mm).
  - "Runoff": The amount of water exported via surface runoff (in mm).
  - "DeepDrainage": The amount of water exported via deep drainage (in mm).
  - "Evapotranspiration": Evapotranspiration (in mm).
  - "SoilEvaporation": Bare soil evaporation (in mm).
  - "PlantExtraction": Amount of water extracted from soil by plants (in mm) (can only be diferent from transpiration for transpirationMode = "Complex" when capacitance is considered).
  - "Transpiration": Plant transpiration (considering all soil layers) (in mm).
  - "HydraulicRedistribution": Water redistributed among soil layers, transported through the plant hydraulic network (only for transpirationMode = "Complex").

- "Soil": A data frame where different variables (in columns) are given for each simulated day (in rows):
  - "W.1", ..., "W.k": Relative soil moisture content (relative to field capacity) in each soil layer.
  - "ML.1", ..., "ML.k": Soil water volume in each soil layer (in L/m2).
  - "MLTot": Total soil water volume (in L/m2).
  - "SWE": Snow water equivalent (mm) of the snow pack.
  - "PlantExt.1", ..., "PlantExt.k": Plant extraction from each soil layer (in mm).
  - "HydraulicInput.1", ..., "HydraulicInput.k": Water that entered the layer coming from other layers and transported via the plant hydraulic network (in mm) (only for transpirationMode = "Complex").
  - "psi.1", ..., "psi.k": Soil water potential in each soil layer (in MPa).
- "PlantLAI": A data frame with the daily leaf area index for each plant cohort. Days are in rows and plant cohorts are in columns.
- "PlantTranspiration": A data frame with the amount of daily transpiration (in mm) for each plant cohort. Days are in rows and plant cohorts are in columns.
- "PlantPhotosynthesis": A data frame with the amount of daily net photosynthesis (in g C·m-2) for each plant cohort. Days are in rows and plant cohorts are in columns.
- "PlantPsi": A data frame with the average daily water potential of each plant (in MPa). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "PlantStress": A data frame with the amount of daily stress [0-1] suffered by each plant cohort (relative whole-plant conductance). Days are in rows and plant cohorts are in columns.
- "subdaily": A list of objects of class spwb.day, one per day simulated (only if required in control parameters, see defaultControl).

If transpirationMode=="Complex" the list also includes the following elements:

- "PlantAbsorbedSWR": A data frame with the daily SWR absorbed by each plant cohort. Days are in rows and plant cohorts are in columns.
- "PlantAbsorbedLWR": A data frame with the daily LWR absorbed by each plant cohort. Days are in rows and plant cohorts are in columns.
- "dEdP": A data frame with mean daily values of soil-plant conductance (derivative of the supply function) for each plant cohort. Days are in rows and plant cohorts in columns.
- "EnergyBalance": A data frame with the daily values of energy balance components for the soil and the canopy.
- "Temperature": A data frame with the daily values of minimum/mean/maximum temperatures for the atmosphere (input), canopy and soil.
- "LeafPsi": A data frame with the minimum daily leaf water potential of each plant (in MPa). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "LeafRWC": A data frame with the average daily leaf relative water content of each plant (in percent). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "StemPsi": A data frame with the minimum daily stem water potential of each plant (in MPa). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "StemPLC": A data frame with the average daily proportion of stem conductance loss of each plant ([0-1]). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "StemRWC": A data frame with the average daily stem relative water content of each plant (in percent). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "RootPsi": A data frame with the minimum daily root water potential of each plant (in MPa). Days are in rows and plant cohorts are in columns. Columns in this data frame correspond to the elements in 'SP'.
- "RhizoPsi": A list of data frames (one per plant cohort) with the minimum daily root water potential of each plant (in MPa). In each data frame, days are in rows and layers are in columns.

Note

State variables stored in objects `x` and `soil` are modified during the simulation. Use function `spwb.resetInputs()` to reset state variables to defaults. Daily transpiration and photosynthesis values are stored in columns `Transpiration` and `Photosynthesis` of object `x`. Water content relative to field capacity (vector `W`) of `soil` is also modified.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

De Cáceres M, Martínez-Vilalta J, Coll L, Llorens P, Casals P, Poyatos R, Pausas JG, Brotons L. (2015) Coupling a water balance model with forest inventory data to predict drought stress: the role of forest structural changes vs. climate changes. Agricultural and Forest Meteorology 213: 77-90 (doi:10.1016/j.agrformet.2015.06.012).

See Also

spwbInput, spwb.day, plot.spwb, spwbpoints, spwbgrid, spwb.ldrOptimization, forest

Examples

```
#Load example daily meteorological data
data(examplemeteo)

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Initialize soil with default soil params (2 layers)
examplesoil = soil(defaultSoilParams(2))

#Initialize control parameters
control = defaultControl()

#Initialize input
x1 = forest2spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S1<-spwb(x1, examplesoil, examplemeteo, elevation = 100)

#Plot results
plot(S1)

#Monthly summary (averages) of soil water balance
summary(S1, freq="months",FUN=mean, output="Soil")



#Initialize soil with default soil params (2 layers)
examplesoil2 = soil(defaultSoilParams(2))

#Switch to 'Complex' transpiration mode
control$transpirationMode="Complex"

#Initialize input
x2 = forest2spwbInput(exampleforest,examplesoil2, SpParamsMED, control)

#Call simulation function (5 days)
S2<-spwb(x2, examplesoil2, examplemeteo[100:105,], latitude = 41.82592, elevation = 100)
```

spwb.day *Soil-plant water balance for a single day*

Description

Function spwb.day performs water balance for a single day.

Usage

```
spwb.day(x, soil, date, tmin, tmax, rhmin, rhmax, rad, wind,
        latitude, elevation, slope, aspect, prec, runon = 0.0)
```

Arguments

x	An object of class spwbInput.
soil	A list containing the description of the soil (see soil).
date	Date as string "yyyy-mm-dd".
tmin, tmax	Minimum and maximum temperature (in degrees Celsius).
rhmin, rhmax	Minimum and maximum relative humidity (in percent).
rad	Solar radiation (in MJ/m2/day).
wind	Wind speed (in m/s).
prec	Precipitation (in mm).
latitude	Latitude (in degrees). Required when using the 'Complex' transpiration mode.
elevation, slope, aspect	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North). Required when using the 'Complex' transpiration mode.
runon	Surface water amount running on the target area from upslope (in mm).

Details

Detailed model description is available in the vignettes section. The model using 'Simple' transpiration mode is described in De Caceres et al. (2015). Simulations using the 'Complex' transpiration mode are computationally much more expensive.

Value

An object (a list) of class spwb.day with the following elements:

- "cohorts": A data frame with cohort information, copied from spwbInput.
- "WaterBalance": A vector of water balance components (rain, snow, net rain, infiltration, ...) for the simulated day, equivalent to one row of 'WaterBalance' object given in spwb.
- "Soil": A data frame with results for each soil layer:

  - "SoilEvaporation": Water evaporated from the soil surface (in mm).
  - "HydraulicInput": Water entering each soil layer from other layers, transported via plant hydraulic network (in mm) (only for transpirationMode = "Complex").
  - "HydraulicOutput": Water leaving each soil layer (going to other layers or the transpiration stream) (in mm) (only for transpirationMode = "Complex").
  - "PlantExtraction": Water extracted by plants from each soil layer (in mm).
  - "psi": Soil water potential (in MPa).
- "EnergyBalance": When using the 'Complex' transpiration mode, the model performs energy balance of the stand and 'EnergyBalance' is a list with the following:
  - "Temperature": A data frame with the temperature of the atmosphere ('Tatm'), canopy ('Tcan') and soil ('Tsoil.1', 'Tsoil.2', ...) for each time step.
  - "CanopyEnergyBalance": A data frame with the components of the canopy energy balance (in W/m2) for each time step.
  - "SoilEnergyBalance": A data frame with the components of the soil energy balance (in W/m2) for each time step.
- "Plants": A data frame of results for each plant cohort. When using the 'Simple' transpiration mode this includes:
  - "LAI": Leaf area index of the plant cohort.
  - "Transpiration": Transpirated water (in mm) corresponding to each cohort.
  - "psi": Water potential (in MPa) of the plant cohort (average over soil layers).
  - "DDS": Daily drought stress [0-1] (relative whole-plant conductance).

  When using the 'Complex' transpiration mode the data frame "Plants" also includes columns:
  - "Extraction": Water extracted from the soil (in mm) for each cohort.
  - "RootPsi": Minimum water potential (in MPa) at the root collar.
  - "StemPsi": Minimum water potential (in MPa) at the stem.
  - "LeafPsi": Minimum water potential (in MPa) at the leaf.
  - "StemPLC": Proportion of conductance loss in stem.
  - "StemRWC": Relative water content of symplastic stem tissue.
  - "LeafRWC": Relative water content of symplastic leaf tissue.
  - "dEdP": Overall soil-plant conductance (derivative of the supply function).
- "RhizoPsi": Minimum water potential (in MPa) inside roots, after crossing rhizosphere, per cohort and soil layer.
- "PlantsInst": A list with instantaneous (per time step) results for each plant cohort:
  - "LAIsunlit": Leaf area index of sunlit leaves of the plant cohort.
  - "LAIshade": Leaf area index of shade leaves of the plant cohort.
  - "AbsRad": A list with four data frames containing the instantaneous absorved radiation for each plant cohort during each time step. The data frames are combinations of short-wave radiation (SWR) vs long-wave radiation (LWR) and sunlit leaves ('SL') vs. shade leaves ('SH').
  - "E": A data frame containing the cumulative transpiration (mm) for each plant cohort during each time step.
  - "An": A data frame containing the cumulative net photosynthesis (gC/m2) for each plant cohort during each time step.

- "GWsunlit": A data frame containing instantaneous stomatal conductance (in mmol/m2/s) for sunlit leaves each plant cohort during each time step.
- "GWshade": A data frame containing instantaneous stomatal conductance (in mmol/m2/s) for shade leaves each plant cohort during each time step.
- "VPDsunlit": A data frame containing vapour pressure deficit (in kPa) for sunlit leaves each plant cohort during each time step.
- "VPDshade": A data frame containing vapour pressure deficit (in kPa) for shade leaves each plant cohort during each time step.
- "Tempsunlit": A data frame containing temperature (in degrees Celsius) for sunlit leaves each plant cohort during each time step.
- "Tempshade": A data frame containing temperature (in degrees Celsius) for shade leaves each plant cohort during each time step.
- "PsiRoot": A data frame containing root crown water potential (in MPa) for each plant cohort during each time step.
- "PsiPlant": A data frame containing leaf water potential (in MPa) for each plant cohort during each time step.
- "PLCstem": A data frame containing the proportion loss of conductance [0-1] for each plant cohort during each time step.
- "RWCstem": A data frame containing the (average) relative water content of symplastic stem tissue [0-1] for each plant cohort during each time step.
- "RWCstem": A data frame containing the relative water content of symplastic leaf tissue [0-1] for each plant cohort during each time step.

Note

Objects x and soil are modified during the simulation. Daily transpiration and photosynthesis values are stored in columns Transpiration and Photosynthesis of object x. Water content relative to field capacity (vector W) of soil is also modified.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

De Cáceres M, Martínez-Vilalta J, Coll L, Llorens P, Casals P, Poyatos R, Pausas JG, Brotons L. (2015) Coupling a water balance model with forest inventory data to predict drought stress: the role of forest structural changes vs. climate changes. Agricultural and Forest Meteorology (doi:10.1016/j.agrformet.2015.06.012).

See Also

spwbInput, spwb, spwbpoints, spwbgrid, spwb.ldrOptimization, forest

Examples

```
#Load example daily meteorological data
data(examplemeteo)
```

```
#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Initialize control parameters
control = defaultControl()

#Initialize soil with default soil params (2 layers)
examplesoil = soil(defaultSoilParams(2), W=c(0.5,0.5))


#Simulate one day only
x1 = forest2spwbInput(exampleforest,examplesoil, SpParamsMED, control)
d = 100
sd1<-spwb.day(x1, examplesoil, rownames(examplemeteo)[d],
              examplemeteo$MinTemperature[d], examplemeteo$MaxTemperature[d],
              examplemeteo$MinRelativeHumidity[d], examplemeteo$MaxRelativeHumidity[d],
              examplemeteo$Radiation[d], examplemeteo$WindSpeed[d],
              latitude = 41.82592, elevation = 100, slope=0, aspect=0,
              prec = examplemeteo$Precipitation[d])

#Initialize soil with default soil params (2 layers)
examplesoil2 = soil(defaultSoilParams(2))

#Switch to 'Complex' transpiration mode
control$transpirationMode="Complex"


#Simulate one day only
x2 = forest2spwbInput(exampleforest,examplesoil2, SpParamsMED, control)
d = 100
sd2<-spwb.day(x2, examplesoil2, rownames(examplemeteo)[d],
              examplemeteo$MinTemperature[d], examplemeteo$MaxTemperature[d],
              examplemeteo$MinRelativeHumidity[d], examplemeteo$MaxRelativeHumidity[d],
              examplemeteo$Radiation[d], examplemeteo$WindSpeed[d],
              latitude = 41.82592, elevation = 100, slope=0, aspect=0,
              prec = examplemeteo$Precipitation[d])

#Plot plant transpiration (see function 'plot.swb.day()')
plot(sd2)
```

---

spwb.ldrCalibration *Calibration of root distribution*

Description

The function spwb.ldrCalibration calibrates the species root distribution within spwb, given the arguments x, meteo, soil, psi_crit, obs and calibVar. This calibration is based on reference measured values. These reference measured values can be Soil water content, Total tranpiration or Transpiration by cohort. Return the calibrated root distribution for each tree species (no shrub calibration is done), expressed as parameters of the function root.ldrDistribution.

Usage

```
spwb.ldrCalibration(x, soil, meteo, calibVar, obs,
                    RZmin = 301, RZmax = 4000,
                    V1min = 0.01, V1max = 0.94, resolution = 20, heat_stop = 0,
                    transformation = "identity", verbose = FALSE)
```

Arguments

x	An object of class spwbInput.
soil	A list containing the description of the soil (see soil).
meteo	A data frame with daily meteorological data series. When using the 'Simple' transpiration mode the following columns are required: • DOY: Day of the year (Julian day). • Precipitation: Precipitation (in mm). • MeanTemperature: Mean temperature (in degrees Celsius). • PET: Potential evapotranspiration (in mm).
calibVar	A character string indicating the calibration variable to be used. It can be one of the following: SWC, Eplanttot or Cohorts.
obs	Measured calibration variable. Depending on the value of calibVar it can be a numeric vector with the measured SWC values (if calibVar = "SWC"), or a data frame with the first column containing the measured total transpiration (named Eplanttot) and the following columns containing the cohorts transpiration.
RZmin	The minimum value of RZ (the rooting depth) to be explored (in mm)
RZmax	The maximum value of RZ (the rooting depth) to be explored (in mm)
V1min	The minimum value of V1 (the root proportion in the first soil layer) to be explored
V1max	The maximum value of V1 (the root proportion in the first soil layer) to be explored
resolution	An integer defining the number of values to obtain by discretization of the root parameters RZ and V1. The number of parameter combinations and therefore the computation cost increases increase with the square of resolution
transformation	Function to modify the size of Z intervals to be explored (by default, bins are equal).
heat_stop	An integer defining the number of days during to discard from the calculation of the optimal root distribution. Usefull if the soil water content initialization is not certain
verbose	A logical value. Print the internal messages of the function?

Details

This function performs three different kinds of calibration, selecting those root distribution parameters that minimize the MAE between the predicted values and the measured values provided in obs argument. If calibVar = "SWC" different V1 values are tested running spwb maintaining the total soil depth provided in x and assuming that value is also the depth containing 95 percent of the roots. If calibVar = "Eplanttot" or calibVar = 'Cohorts' different combinations of RZ and V1 values are tested for each tree cohort and the root paramters are selected based on the MAE between the total transpiration or the cohort transpiration.

Value

THe function returns a data frame containing the species index used in medfate, calibrated values for Z50, Z95 and V1 and the MAE value for that combination.

Author(s)

Víctor Granda, Centre Tecnologic Forestal de Catalunya

Antoine Cabon, Centre Tecnologic Forestal de Catalunya

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

spwb.ldrOptimization for when no measured data is available, spwb, soil, root.ldrDistribution

---

spwb.ldrOptimization *Optimization of root distribution*

Description

The function spwb.ldrOptimization optimizes the species root distribution within spwb, given the arguments x, meteo, soil and psi_crit. The optimization is based on the eco-hydrological equilibrium hypothesis (Eagleson, 1982), which is formulated here as the root distribution for which plant transpiration is maximized while the plant water potential is close to the species-defined critical value psi_crit (Cabon et al.,2018). Returns the optimized root distribution for each species, expressed as parameters of the function root.ldrDistribution.

Usage

```
spwb.ldrOptimization(x, soil, meteo, psi_crit, opt_mode = 1, RZmin = 301, RZmax = 4000,
                     V1min = 0.01, V1max = 0.94, resolution = 20, heat_stop = 0,
                     transformation = "identity", explore_out = FALSE,
                     verbose = FALSE, ...)
```

Arguments

x	An object of class spwbInput.
soil	A list containing the description of the soil (see soil).
meteo	A data frame with daily meteorological data series. When using the 'Simple' transpiration mode the following columns are required: • DOY: Day of the year (Julian day). • Precipitation: Precipitation (in mm). • MeanTemperature: Mean temperature (in degrees Celsius). • PET: Potential evapotranspiration (in mm).
psi_crit	A numerical vector of length iqual to the number of species in the plot containing the species values of water potential inducing hydraulic failure (in MPa). Use NA values to skip optimization for particular plant cohorts.
opt_mode	Optimization mode: • opt_mode = 1 maximizes transpiration along the line of stress equal to psi_crit (Cabon et al. 2018). • opt_mode = 2 maximizes transpiration, subject to root construction constrains, among combinations with stress according to psi_crit).
RZmin	The minimum value of RZ (the rooting depth) to be explored (in mm)
RZmax	The maximum value of RZ (the rooting depth) to be explored (in mm)
V1min	The minimum value of V1 (the root proportion in the first soil layer) to be explored
V1max	The maximum value of V1 (the root proportion in the first soil layer) to be explored
resolution	An integer defining the number of values to obtain by discretization of the root parameters RZ and V1. The number of parameter combinations and therefore the computation cost increases increase with the square of resolution
transformation	Function to modify the size of Z intervals to be explored (by default, bins are equal).
heat_stop	An integer defining the number of days during to discard from the calculation of the optimal root distribution. Usefull if the soil water content initialization is not certain
explore_out	A logical value. Are the values of average daily plant transpiration and minimum plant water potential to be returned?
verbose	A logical value. Print the internal messages of the function?
...	Additional parameters to function spwb.

Details

For each combination of the parameters RZ and V1 the function runs spwb, setting the total soil depth equal to RZ. The root proportion in each soil layer is derived from V1, the depth of the first soil layer (as defined in the argument soil) and RZ using the LDR root distribution model (Schenk and Jackson, 2002) and assuming that the depth containing 95 percent of the roots is equal to RZ. psi_crit, the species specific water potential inducing hydraulic failure, can be approached by

the water potential inducing 50 percent of loss of conductance for the and gymnosperms and 88 percent for the angiosperms (Urli et al., 2013, Brodribb et al., 2010). Details of the hypothesis and limitations of the optimization method are given in Cabon et al. (under review).

Value

If `explore_out = FALSE` returns a data frame with containing the species index used in medfate, `psi_crit` and the otpimized values of V1 and the LDR parameters Z50 and Z95 (see `root.ldrDistribution`) and as many rows as the number of species. If `explore_out = TRUE` returns a list containing `optim`, the aforementioned data frame, and `explore_out`, a list containing two array with the values of average daily plant transpiration and the minimum plant water potential for each species and parameter combination.

Author(s)

Antoine Cabon, Centre Tecnologic Forestal de Catalunya

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Brodribb, T.J., Bowman, D.J.M.S., Nichols, S., Delzon, S., Burlett, R., 2010. Xylem function and growth rate interact to determine recovery rates after exposure to extreme water deficit. New Phytol. 188, 533–542. doi:10.1111/j.1469-8137.2010.03393.x

Cabon, A., Martínez-Vilalta, J., Poyatos, R., Martínez de Aragón, J., De Cáceres, M. (2018) Applying the eco-hydrological equilibrium hypothesis to estimate root ditribution in water-limited forests. Ecohydrology 11: e2015.

Eagleson, P.S., 1982. Ecological optimality in water-limited natural soil-vegetation systems: 1. Theory and hypothesis. Water Resour. Res. 18, 325–340. doi:10.1029/WR018i002p00325

Schenk, H.J., Jackson, R.B., 2002. The Global Biogeography of Roots. Ecol. Monogr. 72, 311. doi:10.2307/3100092

Urli, M., Porte, A.J., Cochard, H., Guengant, Y., Burlett, R., Delzon, S., 2013. Xylem embolism threshold for catastrophic hydraulic failure in angiosperm trees. Tree Physiol. 33, 672–683. doi:10.1093/treephys/tpt030

See Also

`spwb`, `soil`, `root.ldrDistribution`

Examples

```
## Not run:
#Load example daily meteorological data
data(examplemeteo)

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)
```

```
#Initialize soil with default soil params
examplesoil = soil(defaultSoilParams(2))

#Initialize control parameters
control = defaultControl()

#Initialize input
x = forest2spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Run optimization
spwb.ldrOptimization(x = x, soil = examplesoil, meteo = examplemeteo,
                     psi_crit = c(-2,-3,-4), elevation = 100)

## End(Not run)
```

`spwb.resistances` *Soil-plant resistances*

Description

Calculates and draws rhizosphere, root, stem and leaf resistances for simulation time steps

Usage

```
spwb.resistances(x, cohort = 1, relative = FALSE, draw = FALSE,
                 cumulative = FALSE, yearAxis = FALSE, xlab = NULL, ylab = NULL)
```

Arguments

`x`	An object of class `spwb`. The function only works with the result of simulations with `transpirationMode = "Complex"`.
`cohort`	An integer index indicating the cohort for which resistances are desired (by default the first cohort).
`relative`	A boolean flag to indicate that relative percentages are desired as output
`draw`	A boolean flag to indicate that a plot should be drawn.
`cumulative`	A flag to indicate that drawn series should be cumulative.
`yearAxis`	A boolean to indicate whether the units of the x-axis are years (by default they are dates).
`xlab`	x-axis label.
`ylab`	y-axis label.

Details

The function makes internal calls to `hydraulics.soilPlantResistances`.

Value

A data frame with dates in rows and resistance segments in columns (Rhizosphere, Root, Stem and Leaf). Values depend on whether `relative = TRUE` (percentages) or `relative = FALSE` (absolute resistance values). If `draw = TRUE` the calculated resistances are returned as an invisible object.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

`spwb`, `plot.spwb`

`spwb.stress` *Drought stress indices*

Description

Allows calculating annual-based or monthly-based drought stress indices from `spwb` objects.

Usage

```
spwb.stress(x, index = "NDD", freq = "years", bySpecies = FALSE)
```

Arguments

`x`	An object of class `spwb`.
`index`	A string with the index to be calculated, either `"DI"`, `"NDD"`, `"ADS"`, `"MDS"` or `"WSI"` (see details).
`freq`	Frequency of stress statistics (see `cut.Date`). Normally, either `"years"` or `"months"` for yearly-based or monthly-based indices.
`bySpecies`	Allows aggregating output by species.

Details

The currently available drought stress indices are:

- `"ADS"`: Average of daily drought stress values for the period considered.
- `"MDS"`: Maximum daily drought stress during the period considered.
- `"DI"`: Drought intensity, as defined in De Cáceres et al. (2015).
- `"NDD"`: Number of drought days, as defined in De Cáceres et al. (2015).
- `"WSI"`: Water stress integral, as defined in Myers (1988).

Value

A data frame with periods (e.g., years or months) in rows and plant cohorts (or species) in columns. Values are the calculated stress index.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

De Cáceres M, Martínez-Vilalta J, Coll L, Llorens P, Casals P, Poyatos R, Pausas JG, Brotons L. (2015) Coupling a water balance model with forest inventory data to predict drought stress: the role of forest structural changes vs. climate changes. Agricultural and Forest Meteorology 213: 77-90 (doi:10.1016/j.agrformet.2015.06.012).

Myers BJ (1988) Water stress integral - a link between short-term stress and long-term growth. Tree Physiology 4: 315–323 (doi: 10.1093/treephys/4.4.315)

See Also

spwb, summary.spwb

---

spwbgrid *Soil-plant water balance and lateral water discharge*

Description

Function spwbgrid conducts daily soil and plant water balance over a grid of cells while incorporating water runoff from upperslope cells into the current cell.

Usage

```
spwbgrid(y, SpParams, meteo, dates,
           summaryFreq = "years", trackSpecies = numeric(),
           control = defaultControl())
## S3 method for class 'spwbgrid'
plot(x, type = "Runon", summaryIndex = 1, spIndex = NULL, ...)
```

Arguments

y	An object of class SpatialGridLandscape-class.
SpParams	A data frame with species parameters (see SpParamsMED).
meteo	A SpatialGridMeteorology-class object, a data frame with two columns: 'dir' and 'filename, to indicate the path to the meteorological data, or a data frame with meteorological data (the same for all cells).
dates	A Date object describing the days of the period to be modeled.
summaryFreq	Frequency in which summary layers will be produced (e.g. "years", "months", ...) (see cut.Date).
trackSpecies	An integer vector containing the indices of species for which transpiration and drought stress is to be tracked.
control	A list of control parameters (see defaultControl).

x	An object of class spwbgrid.
type	Type of information to be drawn.
summaryIndex	The index of the summary to be ploted.
spIndex	The index of the species to be ploted (for some types).
...	Additional parameters to function spplot.

Details

Function spwbgrid requires daily meteorological data over a grid. The user may supply:

1. an object of class SpatialGridMeteorology.
2. a data frame with information regarding where to read meteorological data.
3. a data frame with meteorological data common for all cells of the grid.

Value

A list of class 'spwbgrid' with the following elements:

- grid: The GridTopology object corresponding to the simulated area.
- LandscapeBalance: A data frame with as many rows as summary points and where columns are components of the water balance at the landscape level (i.e., rain, snow, interception, infiltration, soil evaporation, plant transpiration, ...).

Then, the following matrices are included (each with as many rows as cells and as many columns as summary points):

- Rain: Rainfall (in mm).
- Snow: Snowfall (in mm).
- Interception: Rainfall interception (in mm).
- Infiltration: The amount of water infiltrating into the soil (in mm).
- Runon: The amount of water imported from other cells via surface runoff (in mm).
- Runoff: The amount of water exported via surface runoff (in mm).
- DeepDrainage: The amount of water exported via deep drainage (in mm).
- Esoil: Bare soil evaporation (in mm).
- Eplant: Plant transpiration (in mm).

The same list contains two three-dimensional arrays (each with dimensions number of cells, number of summary layers and number of tracked species):

- Transpiration: Total transpiration (in mm) of the tracked species for each summary period.
- DI: Drought intensity (from 0 to 1) of the tracked species for each summary period.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya.

See Also

spwb, spwbpoints, SpatialGridLandscape-class

Examples

```
## Not run:
# Load spatial grid landscape object
data("exampleSGL")

#Load meteorological data (the same will be used for all points)
data("examplemeteo")

#Load species parameters
data("SpParamsMED")

# Deactivate extra console output
control = defaultControl()
control$verbose = FALSE

# Run water balance simulation over the grid for five days
# (you can test longer periods)
res <- spwbgrid(exampleSGL, SpParamsMED, examplemeteo[1:5,],
                control = control, summaryFreq = "months")

## End(Not run)
```

---

spwbInput *Input for simulation models*

Description

Functions forest2spwbInput and forest2growthInput take an object of class forest and calculate input data for functions spwb and growth, respectively. Functions spwbInput and growthInput does the same from input data. Function forest2aboveground calculates aboveground variables that may be used in spwbInput and growthInput functions. Similarly, function forest2belowground calculates belowground root distribution that may be used in spwbInput and growthInput functions.

Usage

```
forest2aboveground(x, SpParams, gdd = NA)
forest2belowground(x, soil, SpParams)
forest2growthInput(x, soil, SpParams, control)
forest2spwbInput(x, soil, SpParams, control)
growthInput(above,  Z, V,  soil, SpParams, control)
spwbInput(above,  V,  soil, SpParams, control)
```

Arguments

x	An object of class forest.
SpParams	A data frame with species parameters (see SpParamsMED and SpParamsMED).
gdd	Growth degree days to account for leaf phenology effects (in Celsius). This should be left NA in most applications.
soil	An object of class soil.
control	A list with default control parameters (see defaultControl).
above	A data frame with aboveground plant information (see the return value of forest2aboveground below). In the case of spwbInput the variables should include SP, LAI_live, LAI_dead, H and CR.
Z	A numeric vector with cohort rooting depths (in cms).
V	A numeric matrix (with as many columns as soil layers and as many rows as the length as SP) containing the proportion of roots of each plant in each soil layer.

Details

Functions forest2spwbInput and forest2abovegroundInput extracts height and species identity from plant cohorts of x, and calculate leaf area index and crown ratio.forest2spwbInput also calculates the distribution of fine roots across soil. Both forest2spwbInput and spwbInput find parameter values for each plant cohort according to the parameters of its species as specified in SpParams. If control$transpirationMode = "Complex" the functions also estimate the maximum conductance of rhizosphere, root xylem and stem xylem elements.

Value

Function forest2aboveground returns a data frame with the following columns (rows are identified as specified by function plant.ID):

- SP: Species identity (an integer) (first species is 0).
- N: Cohort density (ind/ha) (see function plant.Density).
- DBH: Tree diameter at breast height (cm).
- H: Plant total height (cm).
- CR: Crown ratio (crown length to total height) (between 0 and 1).
- LAI_live: Live leaf area index (m2/m2) (one-side leaf area relative to plot area).
- LAI_dead: Dead leaf area index (m2/m2) (one-side leaf area relative to plot area).

Functions forest2spwbInput and spwbInput return a list of class spwbInput with the following elements (rows of data frames are identified as specified by function plant.ID):

- control: List with control parameters (see defaultControl).
- cohorts: A data frame with cohort information, with columns SP and Name.
- above: A data frame with columns H, CR and LAI (see function forest2aboveground).
- below: A list. If control$transpirationMode = "Simple" it contains a single element:
  - V: A matrix with the proportion of fine roots of each cohort (in rows) in each soil layer (in columns).

If `control$transpirationMode = "Complex"` there are the following additional elements:

- `VGrhizo_kmax`: A matrix with maximum rhizosphere conductance values of each cohort (in rows) in each soil layer (in columns).
- `VGroot_kmax`: A matrix with maximum root xylem conductance values of each cohort (in rows) in each soil layer (in columns).

- `paramsBase`: A data frame with columns:
  - `k`: PAR extinction coefficient.
  - `g`: Canopy water retention capacity per LAI unit (mm/LAI).
  - `Sgdd`: Growth degree days needed for leaf budburst (for winter decideous species).
- `paramsTransp`: A data frame with transpiration parameters. If `control$transpirationMode = "Simple"`, columns are:
  - `Psi_Extract`: Water potential corresponding to 50% relative conductance (in MPa).
  - `WUE`: Water use efficiency for carbon assimilation (g C/mm water).

  If `control$transpirationMode = "Complex"` columns are:
  - `Gwmax`: Maximum stomatal conductance to water vapor (in mol H2O·m-2·s-1).
  - `Vmax298`: Maximum Rubisco carboxilation rate at 25ºC (in micromol CO2·s-1·m-2).
  - `Jmax298`: Maximum rate of electron transport at 25ºC (in micromol photons·s-1·m-2).
  - `VCroot_c`, `VCroot_d`: Parameters of the root xylem vulnerability curve.
  - `xylem_kmax`: Sapwood-specific hydraulic conductivity of stem xylem (in kg H2O·s-1·m-2).
  - `VCstem_kmax`: Maximum stem xylem conductance values of each cohort.
  - `VCstem_c`, `VCstem_d`: Parameters of the stem xylem vulnerability curve.
- `Transpiration`: Plant cohort transpiration of the current day (mm of water = L/m2; filled with zeroes before simulations).
- `Photosynthesis`: Plant cohort photosynthesis of the current day (g C/m2; filled with zeroes before simulations).

Functions `forest2growthInput` and `growthInput` return a list of class `growthInput` with the same elements as `spwbInput`, but with additional information.

- Element `above` includes the following additional columns:
  - `LA_live`: Live leaf area per individual (m2/ind).
  - `LA_dead`: Dead leaf area per individual (m2/ind).
  - `SA`: Live sapwood area per individual (cm2/ind).
- `paramsGrowth`: A data frame with columns:
  - `SLA`:Specific leaf area (mm2/mg = m2/kg).
  - `Al2As`: Leaf area to sapwood area ratio (in m2·m-2).

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

`spwb`, `soil`, `forest`, `SpParamsMED`, `defaultSoilParams`, `plant.ID`

Examples

```
#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

# Aboveground parameters
above = forest2aboveground(exampleforest, SpParamsMED)
above

# Initialize soil with default soil params
examplesoil = soil(defaultSoilParams())

# Belowground parameters
below = forest2belowground(exampleforest, examplesoil, SpParamsMED)
below

# Initialize control parameters
control = defaultControl()

# Prepare spwb input
spwbInput(above, below, examplesoil,SpParamsMED, control)

# When starting from an object of class 'forest' the whole process
# can be simplified:
forest2spwbInput(exampleforest, examplesoil, SpParamsMED, control)


# Prepare input for complex transpiration mode
control$transpirationMode = "Complex"
forest2spwbInput(exampleforest,examplesoil,SpParamsMED, control)
```

---

spwbpoints *Model simulations for spatially-distributed forest stands*

Description

Functions `spwbpoints` and `growthpoints` allow calling local models `spwb` and `growth`, respectively, for a set of forest stands distributed in specific locations. No spatial processes are simulated.

Usage

```
spwbpoints(y, SpParams, meteo, control = defaultControl(),
       dates = NULL, summaryFunction = NULL, args=NULL)
growthpoints(y, SpParams, meteo, control = defaultControl(),
       dates = NULL, summaryFunction = NULL, args=NULL)
```

Arguments

y	An object of class SpatialPointsLandscape-class.
SpParams	A data frame with species parameters (see SpParamsMED).
meteo	Meteorology data (see details).
control	A list of control parameters (see defaultControl).
dates	A Date object with the days of the period to be modeled. If NULL, then the whole period of meteo is used.
summaryFunction	An appropriate function to calculate summaries (e.g., summary.spwb).
args	List with additional arguments for the summary function.

Details

Functions spwbpoints and growthpoints accept different formats for meteorological input (parameter meteo). If a data.frame is supplied (as in spwb or growth) then the same meteorology is used for all points (not recommended). To specify different meteorology for different points, the user can use an object of SpatialPointsMeteorology-class. Alternatively, the user can supply an object of class SpatialPointsDataFrame-class containing the meta data (columns dir and filename) of meteorological files that will be read from the disk.

Value

Functions spwbpoints and growthpoints return a list with the following elements:

- sp: An object of class SpatialPoints with the spatial coordinates of forest stands.
- input: A list of objects of class spwbInput or growthInput (one per forest stand). In the case of growthpoints, the input variables will have been modified by the call to the growth function.
- result: A list the result of calling spwb or growth on each forest stand. If summaryFunction is not null, then each element of the list will contain the result of the summary function.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

spwb, growth, SpatialPointsLandscape-class

Examples

```
## Not run:
# Load spatial points forest object
data("exampleSPL")

#Load meteorological data (the same will be used for all points)
data("examplemeteo")
```

```
#Load species parameters
data("SpParamsMED")

# Deactivate extra console output
control = defaultControl()
control$verbose = FALSE

# Run simulation for the first two forest plots
# (you can run the code without subsetting 'exampleSPL')
res <- spwbpoints(exampleSPL, SpParamsMED, examplemeteo, control = control)

#Extract summaries for a given forest plot
summary(res$result$`80001`, freq="months",FUN=sum,
        output="PlantStress", bySpecies = TRUE)

## End(Not run)
```

`supplyfunctions` *Hydraulic supply functions*

Description

Set of functions used in the implementation of hydraulic supply functions (Sperry \& Love 2015).

Usage

```
hydraulics.EXylem(psiPlant, psiUpstream,
                  kxylemmax, c, d, allowNegativeFlux = TRUE,
                  psiCav = 0)
hydraulics.EVanGenuchten(psiRhizo, psiSoil, krhizomax,
                      n, alpha, l = 0.5)
hydraulics.ECapacitance(psi, psiPrev, PLCprev,
                      V, fapo, c,  d,
                      pi0, eps,timestep)
hydraulics.ECrit(psiUpstream, kxylemmax, c, d, pCrit = 0.001)
hydraulics.E2psiXylem(E, psiUpstream,
                      kxylemmax, c, d, psiCav = 0)
hydraulics.E2psiVanGenuchten(E, psiSoil, krhizomax, n, alpha,
                      psiStep = -0.0001, psiMax = -10.0)
hydraulics.E2psiTwoElements(E, psiSoil, krhizomax, kxylemmax, n, alpha, c, d,
                      psiCav = 0, psiStep = -1e-04, psiMax = -10.0)
hydraulics.E2psiBelowground(E, psiSoil,
                      krhizomax, nsoil, alphasoil,
                      krootmax, rootc, rootd,
                      psiIni = as.numeric(c(0)),
                      ntrial = 10, psiTol = 0.0001, ETol = 0.0001)
hydraulics.E2psiAboveground(E, psiRootCrown,
                      kstemmax, stemc, stemd,
```

```
                      kleafmax, leafc, leafd,
                      PLCstem)
  hydraulics.E2psiAbovegroundCapacitance(E, psiRootCrown,
                      psiStemPrev, PLCstem,
                      psiLeafPrev,
                      kstemmax, stemc, stemd,
                      kleafmax, leafc, leafd,
                      Vsapwood, stemfapo, stempi0, stemeps,
                      Vleaf, leaffapo, leafpi0, leafeps,
                      tstep)
  hydraulics.E2psiAbovegroundCapacitanceDisconnected(E,
                      psiStemPrev, PLCstem, RWCsympstemPrev,
                      psiLeafPrev, RWCsympleafPrev,
                      kstemmax, stemc, stemd,
                      kleafmax, leafc, leafd,
                      Vsapwood, stemfapo, stempi0, stemeps,
                      Vleaf, leaffapo, leafpi0, leafeps,
                      klat,
                      tstep = 3600.0)
  hydraulics.E2psiNetwork(E, psiSoil,
                    krhizomax, nsoil, alphasoil,
                    krootmax, rootc, rootd,
                    kstemmax, stemc, stemd,
                    kleafmax, leafc, leafd,
                    PLCstem,
                    psiIni = as.numeric(c(0)),
                    ntrial = 10,
                    psiTol = 0.0001, ETol = 0.0001)
  hydraulics.E2psiNetworkCapacitance(E, psiSoil,
                    psiStemPrev, PLCstem,
                    psiLeafPrev,
                    krhizomax, nsoil, alphasoil,
                    krootmax, rootc, rootd,
                    kstemmax, stemc, stemd,
                    kleafmax, leafc, leafd,
                    Vsapwood, stemfapo, stempi0, stemeps,
                    Vleaf, leaffapo, leafpi0, leafeps,
                    tstep = 3600.0,
                    psiIni = as.numeric(c(0)),
                    ntrial = 10, psiTol = 0.0001, ETol = 0.0001)

  hydraulics.supplyFunctionOneXylem(psiSoil, v,
                    kstemmax, stemc, stemd, psiCav = 0,
                    maxNsteps=200, dE=0.01)
  hydraulics.supplyFunctionTwoElements(Emax, psiSoil,
                    krhizomax, kxylemmax, n, alpha, c, d,
                    psiCav = 0, dE = 0.1, psiMax = -10.0)
  hydraulics.supplyFunctionThreeElements(Emax, psiSoil,
```

```
                  krhizomax, kxylemmax, kleafmax,
                  n, alpha, stemc, stemd, leafc, leafd,
                  psiCav = 0, dE = 0.1, psiMax = -10.0)


hydraulics.supplyFunctionBelowground(psiSoil,
                  krhizomax, nsoil, alphasoil,
                  krootmax, rootc, rootd,
                  minFlow = 0.0, maxNsteps=400,
                  ntrial = 10, psiTol = 0.0001, ETol = 0.0001, pCrit = 0.001)

hydraulics.supplyFunctionAboveground(Erootcrown, psiRootCrown,
                  kstemmax, stemc, stemd,
                  kleafmax, leafc, leafd,
                  PLCstem)

hydraulics.supplyFunctionAbovegroundCapacitance(Erootcrown, psiRootCrown,
                  psiStemPrev, PLCstemPrev,
                  psiLeafPrev,
                  kstemmax, stemc, stemd,
                  kleafmax, leafc, leafd,
                  Vsapwood, stemfapo, stempi0, stemeps,
                  Vleaf, leaffapo, leafpi0, leafeps,
                  tstep = 3600.0)

hydraulics.supplyFunctionNetwork(psiSoil,
                  krhizomax, nsoil, alphasoil,
                  krootmax, rootc, rootd,
                  kstemmax, stemc, stemd,
                  kleafmax, leafc, leafd,
                  PLCstem, minFlow = 0.0, maxNsteps=400,
                  ntrial = 200, psiTol = 0.0001, ETol = 0.0001, pCrit = 0.001)

hydraulics.supplyFunctionPlot(x, soil, draw = TRUE, type = "E")
hydraulics.regulatedPsiXylem(E, psiUpstream, kxylemmax, c, d, psiStep = -0.01)
hydraulics.regulatedPsiTwoElements(Emax, psiSoil, krhizomax, kxylemmax, n, alpha,
                                   c, d, dE = 0.1, psiMax = -10.0)
```

Arguments

v	Proportion of fine roots within each soil layer.
krhizomax	Maximum rhizosphere hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
kxylemmax	Maximum xylem hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
kleafmax	Maximum leaf hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).

`kstemmax` Maximum stem xylem hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).

`krootmax` Maximum root xylem hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).

`klat` Lateral hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).

`E` Flow per surface unit.

`Emax` Maximum flow per surface unit.

`Erootcrown` Flow per surface unit at the root crown.

`psi` Water potential (in MPa).

`psiPrev` Water potential (in MPa) in the previous time step.

`psiStemPrev, psiLeafPrev` Stem or leaf water potential (in MPa) in the previous time step.

`psiUpstream` Water potential upstream (in MPa). In a one-component model corresponds to soil potential. In a two-component model corresponds to the potential inside the roots.

`psiCav` Minimum water potential (in MPa) experienced (for irreversible cavitation).

`minFlow` Minimum flow in supply function.

`psiPlant` Plant water potential (in MPa).

`psiSoil` Soil water potential (in MPa). A scalar or a vector depending on the function.

`psiRhizo` Soil water potential (in MPa) in the rhizosphere (root surface).

`psiRootCrown` Soil water potential (in MPa) at the root crown.

`psiStep` Water potential precision (in MPa).

`psiTol` Precision for water potential estimates (in MPa).

`psiIni` Vector of initial water potential values (in MPa).

`psiMax` Minimum (maximum in absolute value) water potential to be considered (in MPa).

`pCrit` Critical water potential (in MPa).

`PLCstem` Proportion of loss conductance in the stem [0-1].

`PLCprev` Previous proportion of loss conductance [0-1].

`PLCstemPrev` Previous proportion of loss conductance [0-1] in the stem.

`RWCsympstemPrev, RWCsympleafPrev` Previous relative water content in the stem or leaf [0-1].

`V` Capacity of the compartment per leaf area (in L/m2).

`Vleaf, Vsapwood` Capacity of the compartment per leaf area (in L/m2) of leaves and sapwood.

`fapo` Apoplastic fraction (proportion) in the segment.

`stemfapo, leaffapo` Apoplastic fraction (proportion) in the leaf or stem.

`pi0` Full turgor osmotic potential (MPa).

`stempi0,leafpi0`
: Full turgor osmotic potential in the stem or the leaf (MPa).

`eps`
: Bulk modulus of elasticity (MPa).

`stemeps,leafeps`
: Bulk modulus of elasticity (MPa) in the stem or the leaf.

`dE`
: Increment of flow per surface unit.

`ETol`
: Precision for water flow per surface unit.

`c, d`
: Parameters of the Weibull function (generic xylem vulnerability curve).

`rootc, rootd`
: Parameters of the Weibull function for roots (root xylem vulnerability curve).

`stemc, stemd`
: Parameters of the Weibull function for stems (stem xylem vulnerability curve).

`leafc, leafd`
: Parameters of the Weibull function for leaves (leaf vulnerability curve).

`n, alpha, l`
: Parameters of the Van Genuchten function (rhizosphere vulnerability curve).

`nsoil, alphasoil`
: Parameter vectors of the Van Genuchten function (rhizosphere vulnerability curve) with one value for each soil layer.

`allowNegativeFlux`
: A boolean to indicate wether negative flux (i.e. from plant to soil) is allowed.

`maxNsteps`
: Maximum number of steps in the construction of supply functions.

`ntrial`
: Maximum number of steps in Newton-Raphson optimization.

`x`
: An object of class `spwbInput`.

`soil`
: A list containing the description of the soil (see `soil`).

`type`
: Plot type. For `hydraulics.supplyFunctionPlot`, either `"E"`, `"Elayers"`, `"PsiStem"`, `"PsiRoot"`, `"PsiRhizo"` or `"dEdP"`). For `hydraulics.vulnerabilityCurvePlot`, either `"leaf"`, `"stem"`, `"root"`, or `"rhizosphere"`).

`draw`
: A flag to indicate whether the supply function should be drawn or just returned.

`tstep, timestep`
: Time step in seconds.

Details

Details of the hydraulic model are given in a vignette. Function `hydraulics.supplyFunctionPlot` draws a plot of the supply function for the given `soil` object and network properties of each plant cohort in `x`. Function `hydraulics.vulnerabilityCurvePlot` draws a plot of the vulnerability curves for the given `soil` object and network properties of each plant cohort in `x`.

Value

Values returned for each function are:

- `hydraulics.E2psiXylem`: The plant (leaf) water potential (in MPa) corresponding to the input flow, according to the xylem supply function and given an upstream (soil or root) water potential.
- `hydraulics.E2psiVanGenuchten`: The root water potential (in MPa) corresponding to the input flow, according to the rhizosphere supply function and given a soil water potential.

- `hydraulics.E2psiTwoElements`: The plant (leaf) water potential (in MPa) corresponding to the input flow, according to the rhizosphere and plant supply functions and given an input soil water potential.
- `hydraulics.E2psiNetwork`: The rhizosphere, root crown and plant (leaf water potential (in MPa) corresponding to the input flow, according to the vulnerability curves of rhizosphere, root and stem elements in a network.
- `hydraulics.Ecrit`: The critical flow according to the xylem supply function and given an input soil water potential.
- `hydraulics.EVanGenuchten`: The flow (integral of the vulnerability curve) according to the rhizosphere supply function and given an input drop in water potential (soil and rhizosphere).
- `hydraulics.EXylem`: The flow (integral of the vulnerability curve) according to the xylem supply function and given an input drop in water potential (rhizosphere and plant).
- `hydraulics.supplyFunctionOneXylem`, `hydraulics.supplyFunctionTwoElements` and `hydraulics.supplyFunc` A list with different numeric vectors with information of the two-element supply function:
  - `E`: Flow values (supply values).
  - `FittedE`: Fitted flow values (for `hydraulics.supplyFunctionTwoElements`).
  - `Elayers`: Flow values across the roots of each soil layer (only for `hydraulics.supplyFunctionNetwork`).
  - `PsiRhizo`: Water potential values at the root surface (only for `hydraulics.supplyFunctionNetwork`).
  - `PsiRoot`: Water potential values inside the root crown (not for `hydraulics.supplyFunctionOneXylem`).
  - `PsiPlant`: Water potential values at the canopy (leaf).
  - `dEdP`: Derivatives of the supply function.
- `hydraulics.supplyFunctionPlot`: A (hidden) list with the result of calling `hydraulics.supplyFunctionNetwork` for each cohort.
- `hydraulics.regulatedPsiXylem`: Plant water potential after regulation (one-element loss function) given an input water potential.
- `hydraulics.regulatedPsiTwoElements`: Plant water potential after regulation (two-element loss function) given an input soil water potential.

Author(s)

Miquel De Cáceres Ainsa, CTFC, Catalonia, Spain

References

Sperry, J. S., F. R. Adler, G. S. Campbell, and J. P. Comstock. 1998. Limitation of plant water use by rhizosphere and xylem conductance: results from a model. Plant, Cell \& Environment 21:347–359.

Sperry, J. S., and D. M. Love. 2015. What plant hydraulics can tell us about responses to climate-change droughts. New Phytologist 207:14–27.

See Also

`hydraulics.psi2K`, `hydraulics.maximumStemHydraulicConductance`, `spwb`, `soil`

Examples

```
kstemmax = 4 # in mmol·m-2·s-1·MPa-1
stemc = 3
stemd = -4 # in MPa

psiVec = seq(-0.1, -7.0, by =-0.01)

#Vulnerability curve
kstem = unlist(lapply(psiVec, hydraulics.xylemConductance, kstemmax, stemc, stemd))
plot(-psiVec, kstem, type="l",ylab="Xylem conductance (mmol·m-2·s-1·MPa-1)",
xlab="Canopy pressure (-MPa)", lwd=1.5,ylim=c(0,kstemmax))
```

---

tissuemoisture *Tissue moisture functions*

Description

Set of functions used to calculate tissue moisture from water potential and viceversa.

Usage

```
moisture.symplasticRWC(psiSym, pi0, epsilon)
moisture.symplasticPsi(RWC, pi0, epsilon)
moisture.apoplasticRWC(psiApo, c, d)
moisture.apoplasticPsi(RWC, c, d)
moisture.tissueRWC(psiSym, pi0, epsilon,
                   psiApo, c, d,
                   af, femb = 0)
moisture.tissueFMC(RWC, density, d0 = 1.54)
moisture.pressureVolumeCurvePlot(x, segment="leaf",
                    fraction = "all",
                    psiVec =  seq(-0.1, -8.0, by =-0.01))
```

Arguments

psiSym, psiApo	Symplastic or apoplastic water potential (MPa).
RWC	Relative water content [0-1].
pi0	Full turgor osmotic potential (MPa).
epsilon	Bulk modulus of elasticity (MPa).
c,d	Parameters of the xylem vulnerability curve.
af	Apoplastic fraction (proportion) in the segment (e.g. leaf or stem).
femb	Fraction of embolized conduits.
x	An object of class spwbInput.
segment	Curve to plot, either "stem", "leaf" or "both".
fraction	Tissue fraction, either "symplastic", "apoplastic" or "all".

psiVec	Vector of water potential values to evaluate for the pressure-volume curve.
density	Tissue density (g·cm-1).
d0	Matric density (g·cm-1).

Details

Details of the tissue moisture calculations are given in a vignette.

Value

Values returned for each function are:

- moisture.symplasticRWC: Relative water content [0-1] of the symplastic fraction.
- moisture.apoplasticRWC: Relative water content [0-1] of the apoplastic fraction.
- moisture.symplasticWaterPotential: Water potential (in MPa) of the symplastic fraction.
- moisture.apoplasticWaterPotential: Water potential (in MPa) of the apoplastic fraction.
- moisture.segmentRWC: Segment relative water content [0-1].

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Bartlett, M.K., Scoffoni, C., Sack, L. 2012. The determinants of leaf turgor loss point and prediction of drought tolerance of species and biomes: a global meta-analysis. Ecology Letters 15: 393–405.

Hölttä, T., Cochard, H., Nikinmaa, E., Mencuccini, M. 2009. Capacitive effect of cavitation in xylem conduits: Results from a dynamic model. Plant, Cell and Environment 32: 10–21.

Martin-StPaul, N., Delzon, S., Cochard, H. 2017. Plant resistance to drought depends on timely stomatal closure. Ecology Letters 20: 1437–1447.

See Also

hydraulics.psi2K, hydraulics.supplyFunctionPlot, spwb, soil

Examples

```
psi = seq(-10,0, by=0.1)
rwc_s = rep(NA, length(psi))
for(i in 1:length(psi)) rwc_s[i] = moisture.symplasticRWC(psi[i],-3,12)
plot(psi, rwc_s, type="l", xlab="Water potential (MPa)", ylab = "Symplasmic RWC")
```

transp *Transpiration submodel functions*

Description

Set of high-level functions used in the calculation of stomatal conductance and transpiration. Function `transp.profitMaximization` calculates gain and cost functions, as well as profit maximization from supply and photosynthesis input functions. Function `transp.stomatalRegulation` calculates profit maximization for all time steps in a day, starting from stand description, soil description and meteorological input. Function `transp.stomatalRegulationPlot` produces a plot with the cohort supply functions against water potential and a plot with the cohort photosynthesis functions against water potential, both with the maximum profit values indicated.

Usage

```
transp.profitMaximization(supplyFunction, photosynthesisFunction, type,
                          Gwmin, Gwmax, kleafmax = NA)
transp.stomatalRegulation(x, soil, meteo, day,
                             latitude, elevation)
transp.stomatalRegulationPlot(x, soil, meteo, day, timestep,
                              latitude, elevation)
```

Arguments

`supplyFunction`	Water supply function (see `hydraulics.supplyFunctionNetwork`).
`photosynthesisFunction`	Function returned by `photo.photosynthesisFunction()`.
`type`	Optimization type (1,2 or 3).
`Gwmin, Gwmax`	Minimum and maximum stomatal conductance (mol·m-2·s-1).
`kleafmax`	Maximum leaf hydraulic conductance (flow per pressure drop).
`x`	An object of class `spwbInput` built using the 'Complex' transpiration mode.
`soil`	An object of class `soil`.
`meteo`	A data frame with daily meteorological data series: • `DOY`: Day of the year (Julian day). • `Precipitation`: Precipitation (in mm). • `MeanTemperature`: Mean temperature (in degrees Celsius). • `MinTemperature`: Minimum temperature (in degrees Celsius). • `MaxTemperature`: Maximum temperature (in degrees Celsius). • `MinRelativeHumidity`: Minimum relative humidity (in percent). • `MaxRelativeHumidity`: Maximum relative humidity (in percent). • `Radiation`: Solar radiation (in MJ/m2/day). • `WindSpeed`: Wind speed (in m/s). If not available, this column can be left with `NA` values.

day	An integer to identify a day within meteo.
timestep	An integer between 1 and ndailysteps specified in x (see defaultControl).
latitude	Latitude (in degrees).
elevation	Elevation above sea level (in m).

Details

Details of the transpiration submodel are given in a vignette.

Value

Values returned for each function are:

- transp.profitMaximization: A list with the following elements:
  - Cost: Cost function [0-1].
  - Gain: Gain function [0-1].
  - Profit: Profit function [0-1].
  - iMaxProfit: Index corresponding to maximum profit (starting from 0).
- transp.stomatalRegulation: A list whose elements correspond to cohorts. Each element has the following subelements:
  - supply: Supply function (see hydraulics.supplyFunctionNetwork).
  - photoSunlit, photoShade: Photosynthesis function for sunlit and shade leaves and for each time step (see photo.leafPhotosynthesisFunction).
  - PMSunlit, PMShade: Profit maximization for sunlit and shade leaves and each time step (see transp.profitMaximization).

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

References

Sperry, J. S., M. D. Venturas, W. R. L. Anderegg, M. Mencuccini, D. S. Mackay, Y. Wang, and D. M. Love. 2016. Predicting stomatal responses to the environment from the optimization of photosynthetic gain and hydraulic cost. Plant Cell and Environment.

See Also

hydraulics.supplyFunctionNetwork, biophysics.leafTemperature, photo.photosynthesis, spwb

Examples

```
#Load example daily meteorological data
data(examplemeteo)

#Load example plot plant data
data(exampleforest)
```

```
#Default species parameterization
data(SpParamsMED)

#Initialize soil with default soil params (2 layers)
examplesoil2 = soil(defaultSoilParams(2))

#Initialize control parameters
control = defaultControl()
#Switch to 'Complex' transpiration mode
control$transpirationMode="Complex"

#Initialize input
x2 = forest2spwbInput(exampleforest,examplesoil2, SpParamsMED, control)

transp.stomatalRegulationPlot(x2, examplesoil2, examplemeteo, day=100, timestep = 12,
                              latitude = 41.82592, elevation = 100)
```

`Vertical profiles` *Vertical profiles*

Description

Functions to generate vertical profiles generated by an input `forest` object.

Usage

```
vprofile.LeafAreaDensity(x, SpParams, z = NULL, gdd = NA,
                         byCohorts = FALSE, bySpecies = FALSE, draw = TRUE,
                         legend = TRUE, xlim = NULL)
vprofile.RootDistribution(x, SpParams, d = NULL,  bySpecies = FALSE, draw = TRUE,
                          legend = TRUE, xlim = NULL)
vprofile.FuelBulkDensity(x, SpParams, z = NULL, gdd = NA, draw = TRUE)
vprofile.PARExtinction(x, SpParams, z = NULL, gdd = NA, draw = TRUE)
vprofile.SWRExtinction(x, SpParams, z = NULL, gdd = NA, draw = TRUE)
vprofile.WindExtinction(x, SpParams, wind20H, z = NULL,  gdd = NA, draw = TRUE)
```

Arguments

x	An object of class `forest`
SpParams	A data frame with species parameters (see `SpParamsMED`).
z	A numeric vector with height values.
d	A numeric vector with soil layer widths.
gdd	Growth degree days.
byCohorts	Separate profiles for each cohort.
bySpecies	Aggregate cohort profiles by species.

`wind20H`	The value of measured wind speed at 6 m = 20ft (in m/s).
`draw`	Logical flag to indicate that a plot is desired.
`legend`	Logical flag to indicate that legend should be included.
`xlim`	Limits of the x-axis.

Value

A numeric vector with values measured at each height. Units depend on the profile function:

- `vprofile.LeafAreaDensity`: Cumulative LAI (m2/m2) per height bin.
- `vprofile.FuelBulkDensity`: Fuel bulk density (kg/m3) per height bin.
- `vprofile.PARExtinction`: Percent of photosynthetically active radiation (%) corresponding to each height.
- `vprofile.SWRExtinction`: Percent of shortwave radiation (%) corresponding to each height.
- `vprofile.WindExtinction`: Wind speed (m/s) corresponding to each height.

Author(s)

Miquel De Cáceres Ainsa, Centre Tecnologic Forestal de Catalunya

See Also

`forest`

Examples

```
#Default species parameterization
data(SpParamsMED)

#Load example plot plant data
data(exampleforest)

vprofile.LeafAreaDensity(exampleforest, SpParamsMED)
vprofile.FuelBulkDensity(exampleforest, SpParamsMED)

vprofile.PARExtinction(exampleforest, SpParamsMED)
vprofile.SWRExtinction(exampleforest, SpParamsMED)

vprofile.WindExtinction(exampleforest, SpParamsMED, 20)
```

Index