Package ‘medfate’

May 25, 2024

Type Package

Title Mediterranean Forest Simulation

Version 4.3.1

Date 2024-05-23

Description 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>; De Caceres et al. (2021) <doi:10.1016/j.agrformet.2020.108233>].

License GPL (>= 2) | LGPL (>= 3)

URL `https://emf-creaf.github.io/medfate/`

LazyLoad yes

Depends R (>= 3.5.0)

Imports ggplot2 (>= 3.0.0), meteoland (>= 2.0.0), Rcpp (>= 1.0.6), shiny

Suggests testthat (>= 3.0.0), knitr, rmarkdown

LinkingTo Rcpp, meteoland

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.3.1

BugReports `https://github.com/emf-creaf/medfate/issues`

Config/testthat/edition 3

LazyData true

Author Miquel De Cáceres [aut, cre, cph] (<`https://orcid.org/0000-0001-7132-2080`>),
Nicolas Martin-StPaul [aut] (<`https://orcid.org/0000-0001-7574-0108`>),
Víctor Granda [aut] (<`https://orcid.org/0000-0002-0469-1991`>),
Antoine Cabon [aut] (<`https://orcid.org/0000-0001-6426-1726`>),
Jordi Martínez-Vilalta [ctb] (<`https://orcid.org/0000-0002-2332-7298`>),
Maurizio Mencuccini [ctb] (<`https://orcid.org/0000-0003-0840-1477`>),
Julien Ruffault [ctb],
François Pimont [ctb] (<`https://orcid.org/0000-0002-9842-6207`>),

Hervé Cochard [ctb] (<https://orcid.org/0000-0002-2727-7072>),
Aitor Améztegui [ctb] (<https://orcid.org/0000-0003-2006-1559>),
Shengli Huang [ctb] (<https://orcid.org/0000-0003-3927-7042>),
John Burkardt [cph] (Copyright holder of C++ code in 'incgamma.cpp')

Maintainer Miquel De Cáceres <miquelcaceres@gmail.com>

Repository CRAN

Date/Publication 2024-05-25 07:50:08 UTC

R topics documented:

`aspwbInput` *Simulation in agricultural areas*

Description

Function `aspwb_day` performs water balance for a single day in an agriculture location. Function `aspwb` performs water balance for multiple days in an agriculture location.

Usage

```
aspwbInput(crop_factor, control, soil)

aspwb_day(
  x,
  date,
  meteovec,
  latitude,
  elevation,
  slope = NA_real_,
  aspect = NA_real_,
  runon = 0,
  lateralFlows = NULL,
  waterTableDepth = NA_real_,
  modifyInput = TRUE
)

aspwb(
  x,
  meteo,
  latitude,
  elevation,
  slope = NA_real_,
  aspect = NA_real_,
  waterTableDepth = NA_real_
)
```

Arguments

`crop_factor`	Agriculture crop factor.
`control`	A list with default control parameters (see `defaultControl`).
`soil`	An object of class `data.frame` or `soil`.
`x`	An object of class `aspwbInput`.
`date`	Date as string "yyyy-mm-dd".
`meteovec`	A named numerical vector with weather data. See variable names in parameter `meteo` of `spwb`.
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`runon`	Surface water amount running on the target area from upslope (in mm).
`lateralFlows`	Lateral source/sink terms for each soil layer (interflow/to from adjacent locations) as mm/day.
`waterTableDepth`	Water table depth (in mm). When not missing, capillarity rise will be allowed if lower than total soil depth.

modifyInput	Boolean flag to indicate that the input x object is allowed to be modified during the simulation.
meteo	A data frame with daily meteorological data series (see spwb).

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

spwbInput, spwb

Examples

```
control <- defaultControl()
examplesoil <- defaultSoilParams(4)

x <- aspwbInput(0.75, control, examplesoil)

# Day to be simulated
d <- 100
meteovec <- unlist(examplemeteo[d,-1])
date <- as.character(examplemeteo$dates[d])

#Call simulation function for a single days
sd <- aspwb_day(x, date, meteovec,
               latitude = 41.82592, elevation = 100)

#Call simulation function for multiple days
S <- aspwb(x, examplemeteo, latitude = 41.82592, elevation = 100)
```

biophysics *Physical and biophysical utility functions*

Description

Utility functions for the calculation of biophysical variables.

Usage

```
biophysics_radiationDiurnalPattern(t, daylength)

biophysics_temperatureDiurnalPattern(
  t,
  tmin,
  tmax,
  tminPrev,
  tmaxPrev,
  tminNext,
  daylength
)

biophysics_leafTemperature(absRad, airTemperature, u, E, leafWidth = 1)

biophysics_leafTemperature2(
  SWRabs,
  LWRnet,
  airTemperature,
  u,
  E,
  leafWidth = 1
)

biophysics_leafVapourPressure(leafTemp, leafPsi)

biophysics_irradianceToPhotonFlux(I, lambda = 546.6507)

biophysics_waterDynamicViscosity(temp)
```

Arguments

`t`	Time of the day (in seconds).
`daylength`	Day length (in seconds).
`tmin`, `tmax`	Minimum and maximum daily temperature (ºC).
`tminPrev`, `tmaxPrev`, `tminNext`	Maximum and minimum daily temperatures of the previous and following day (ºC).
`absRad`	Absorbed long- and short-wave radiation (in W·m-2).
`airTemperature`	Air temperature (in ºC).
`u`	Wind speed above the leaf boundary layer (in m/s).
`E`	Transpiration flow (in mmol H20·m-2·s-1) per one sided leaf area basis.
`leafWidth`	Leaf width (in cm).
`SWRabs`	Absorbed short-wave radiation (in W·m-2).
`LWRnet`	Net long-wave radiation balance (in W·m-2).
`leafTemp`	Leaf temperature (ºC).
`leafPsi`	Leaf water potential (MPa).
`I`	Irradiance (in W*m-2).
`lambda`	Wavelength (in nm).
`temp`	Temperature (ºC).

Details

Functions `biophysics_leafTemperature` and `biophysics_leafTemperature2` calculate leaf temperature according to energy balance equation given in Campbell and Norman (1988).

Function `biophysics_radiationDiurnalPattern` follows the equations given in Liu and Jordan (1960).

Function `biophysics_temperatureDiurnalPattern` determines diurnal temperature pattern assuming a sinusoidal pattern with T = Tmin at sunrise and T = (Tmin+Tmax)/2 at sunset and a linear change in temperature between sunset and Tmin of the day after (McMurtrie et al. 1990).

Function `biophysics_waterDynamicViscosity` calculates water dynamic viscosity following the Vogel (1921) equation.

Value

Values returned for each function are:

- `biophysics_leafTemperature` and `biophysics_leafTemperature2`: leaf temperature (in ºC)
- `biophysics_leafVapourPressure`: leaf vapour pressure (in kPa)
- `biophysics_radiationDiurnalPattern`: the proportion of daily radiation corresponding to the input time in seconds after sunrise.
- `biophysics_temperatureDiurnalPattern`: diurnal pattern of temperature.
- `biophysics_waterDynamicViscosity`: Water dynamic viscosity relative to 20ºC.

Author(s)

Miquel De Cáceres Ainsa, CREAF

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.

H. Vogel, "Das Temperaturabhangigkeitsgesetz der Viskositat von Flussigkeiten", Physikalische Zeitschrift, vol. 22, pp. 645–646, 1921.

See Also

`spwb`

carbon *Carbon-related functions*

Description

Low-level functions used in the calculation of carbon balance.

Usage

```
carbon_sugarStarchDynamicsLeaf(sugarConc, starchConc, eqSugarConc)

carbon_sugarStarchDynamicsStem(sugarConc, starchConc, eqSugarConc)

carbon_osmoticWaterPotential(sugarConc, temp, nonSugarConc)

carbon_sugarConcentration(osmoticWP, temp, nonSugarConc)

carbon_relativeSapViscosity(sugarConc, temp)

carbon_leafStructuralBiomass(LAI, N, SLA)

carbon_leafStarchCapacity(LAI, N, SLA, leafDensity)

carbon_sapwoodStructuralBiomass(SA, H, L, V, woodDensity)

carbon_sapwoodStructuralLivingBiomass(
  SA,
  H,
  L,
  V,
  woodDensity,
  conduit2sapwood
)

carbon_sapwoodStarchCapacity(SA, H, L, V, woodDensity, conduit2sapwood)

carbon_carbonCompartments(x, biomassUnits = "g_m2")
```

Arguments

`sugarConc`	Concentration of soluble sugars (mol/l).
`starchConc`	Concentration of starch (mol/l)
`eqSugarConc`	Equilibrium concentration of soluble sugars (mol/l).
`temp`	Temperature (degrees Celsius).
`nonSugarConc`	Concentration of inorganic solutes (mol/l).

`osmoticWP`	Osmotic water potential (MPa).
`LAI`	Leaf area index.
`N`	Density (ind·ha-1).
`SLA`	Specific leaf area (mm2/mg = m2/kg).
`leafDensity`	Density of leaf tissue (dry weight over volume).
`SA`	Sapwood area (cm2).
`H`	Plant height (cm).
`L`	Coarse root length (mm) for each soil layer.
`V`	Proportion of fine roots in each soil layer.
`woodDensity`	Wood density (dry weight over volume).
`conduit2sapwood`	Proportion of sapwood corresponding to conducive elements (vessels or tracheids) as opposed to parenchymatic tissue.
`x`	An object of class `growthInput`.
`biomassUnits`	A string for output biomass units, either "g_ind" (g per individual) or "g_m2" (g per square meter).

Value

Values returned for each function are:

- `carbon_leafStarchCapacity`: Capacity of storing starch in the leaf compartment (mol gluc/ind.).
- `carbon_leafStructuralBiomass`: Leaf structural biomass (g dry/ind.)
- `carbon_sapwoodStarchCapacity`: Capacity of storing starch in the sapwood compartment (mol gluc/ind.).
- `carbon_sapwoodStructuralBiomass`: Sapwood structural biomass (g dry/ind.)
- `carbon_sapwoodStructuralLivingBiomass`: Living sapwood (parenchyma) structural biomass (g dry/ind.)
- `carbon_sugarConcentration`: Sugar concentration (mol gluc/l)
- `carbon_osmoticWaterPotential`: Osmotic component of water potential (MPa)
- `carbon_relativeSapViscosity`: Relative viscosity of sapwood with respect to pure water (according to Forst et al. (2002)).
- `carbon_sugarStarchDynamicsLeaf`: Rate of conversion from sugar to starch in leaf (mol gluc/l/s).
- `carbon_sugarStarchDynamicsStem`: Rate of conversion from sugar to starch in leaf (mol gluc/l/s).
- `carbon_carbonCompartments`: A data frame with the size of compartments for each plant cohort, in the specified units.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Forst P, Wermer F, Delgado A (2002). On the pressure dependence of the viscosity of aqueous sugar solutions. Rheol Acta 41: 369–374 DOI 10.1007/s00397-002-0238-y

See Also

`growth`

`defaultControl` *Control parameters for simulation models*

Description

Creates a list control parameters default values for simulations

Usage

```
defaultControl(transpirationMode = "Granier")
```

Arguments

`transpirationMode`
: Transpiration model (either 'Granier', 'Sperry' or 'Sureau'). See `spwbInput`.

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 (default values in brackets):

General:

- `verbose [= TRUE]`: Boolean flag to indicate console output during calculations. In function `fordyn` `verbose` is always set to FALSE.
- `fillMissingRootParams [= TRUE]`: Boolean flag to indicate that initializing functions should provide estimates for Z50 and Z95 if these are missing in the forest data. Note that if `fillMissingRootParams` is set to `FALSE` then simulations may fail if the user does not provide values for Z50 and Z95 in tree or shrub data.
- `fillMissingSpParams [= TRUE]`: Boolean flag to indicate that initializing functions should provide estimates for functional parameters if these are missing in the species parameter table `SpParams`. Note that if `fillMissingSpParams` is set to `FALSE` then simulations may fail if the user does not provide values for required parameters.
- `fillMissingWithGenusParams [=TRUE]`: Boolean flag to indicate that initializing functions should provide estimates from genus value, if species-level values are missing in the species parameter table `SpParams` but genus-level ones are not.

- standResults [= TRUE]: Boolean flag to keep stand-level results (in a data frame called 'Stand').
- soilResults [= TRUE]: Boolean flag to keep soil-level results (in a list called 'Soil').
- snowResults [= TRUE]: Boolean flag to keep snow results (in a data frame called 'Snow').
- plantResults [= TRUE]: Boolean flag to keep plant-level results (in a list called 'Plants').
- leafResults [= TRUE]: Boolean flag to keep leaf-level results (in elements called 'SunlitLeaves' and 'ShadeLeaves').
- temperatureResults [= TRUE]: Boolean flag to keep temperature results (in elements called 'Temperature' and 'TemperatureLayers').
- 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. In function fordyn subdailyResults is always set to FALSE.
- fireHazardResults [= FALSE]: Boolean flag to force calculation of daily fire hazard.
- fireHazardStandardWind [= NA]: Wind speed (in m/s) for fire-hazard estimation. If missing, actual wind-speed is used.
- fireHazardStandardDFMC [= NA]: Dead fuel moisture content for fire-hazard estimation. If missing, estimation from current weather is used.

Water balance (functions spwb, pwb or spwb_day):

- transpirationMode [= "Granier"]: Transpiration model (either 'Granier', 'Sperry' or 'Sureau'). See spwbInput.
- soilFunctions [= "VG"]: Soil water retention curve and conductivity functions, either 'SX' (for Saxton) or 'VG' (for Van Genuchten). If transpirationMode is 'Sperry' or 'Sureau' then soilFunctions is forced to 'VG'. Only simulations with 'Granier' are allowed to use Saxton functions.
- VG_PTF: String indicating the pedotransfer functions for van Genuchten parameters (either 'Toth' or 'Carsel').
- ndailysteps [= 24]: Number of steps into which each day is divided for determination of soil water balance, stomatal conductance, transpiration and photosynthesis (24 equals 1-hour intervals).
- max_nsubsteps_soil [= 300]: Maximum number of substeps for soil water balance solving.
- defaultWindSpeed [= 2.5]: Default wind speed value (in m/s) to be used when missing from data.
- defaultCO2 [= 386]: Default atmospheric (abovecanopy) CO2 concentration (in micromol·mol-1 = ppm). This value will be used whenever CO2 concentration is not specified in the weather input.
- defaultRainfallIntensityPerMonth [= c(1.5, 1.5, 1.5, 1.5, 1.5, 1.5, 5.6, 5.6, 5.6, 5.6, 5.6, 1.5)]: A vector of twelve values indicating the rainfall intensity (mm/h) per month. By default synoptic storms (1.5 mm/h) are assumed between December and June, and convective storms (5.6 mm/h) are assumed between July and November.
- leafPhenology [= TRUE]: Boolean flag to indicate the simulation of leaf phenology for winter-deciduous species.
- bareSoilEvaporation [= TRUE]: Boolean flag to indicate the simulation of evaporation from bare soil.
- unlimitedSoilWater [= FALSE]: Boolean flag to indicate the simulation of plant transpiration assuming that soil water is always at field capacity.

- `unfoldingDD` [= `300`]: Degree-days for complete leaf unfolding after budburst has occurred.
- `interceptionMode` [= `"Gash1995"`]: Infiltration model, either "Gash1995" or "Liu2001".
- `infiltrationMode` [= `"GreenAmpt1911"`]: Infiltration model, either "GreenAmpt1911" or "Boughton1989".
- `infiltrationCorrection` [= `5.0`]: Factor to correct infiltration amount in the GreenAmpt1911 model in single-domain simulations.
- `soilDomains` [= `"single"`]: Either 'single' (for single-domain) or 'dual' (for dual-permeability).
- `rhizosphereOverlap` [= `"total"`]: A string indicating the degree of rhizosphere spatial overlap between plant cohorts:
  - "none" - no overlap (independent water pools).
  - "partial" - partial overlap determined by coarse root volume.
  - "total" - total overlap (plants extract from common soil pools).
- `verticalLayerSize` [= `100`]: Size of vertical layers (in cm) for the calculation of light extinction (and photosynthesis).
- `windMeasurementHeight` [= `200`]: Height (in cm) over the canopy corresponding to wind measurements.
- `segmentedXylemVulnerability` [= `TRUE/FALSE`]: If `FALSE` leaf and root vulnerability curves will be equal to those of stem. By default, `segmentedXylemVulnerability = TRUE` for `transpirationMode = "Sperry"` and `segmentedXylemVulnerability = FALSE` for `transpirationMode = "Sureau"`.
- `leafCavitationEffects`, `stemCavitationEffects` [= `FALSE/TRUE`]: A flag indicating whether cavitation effects on conductance of leaves and stem are applied. Only relevant for `transpirationMode = "Sperry"`.
- `leafCavitationRecovery`, `stemCavitationRecovery` [= `"rate"`]: A string indicating how recovery of previous cavitation leaf/stem xylem is done (only relevant for functions `spwb` and `spwb_day`):
  - "none" - no recovery.
  - "annual" - every first day of the year.
  - "rate" - following a rate of new leaf or sapwood formation.
  - "total" - instantaneous complete recovery.
- `cavitationRecoveryMaximumRate` [= `0.05`]: Maximum rate of daily refilling of embolized conduits as sapwood area per leaf area (in cm2·m-2·day-1).

Water balance (functions `spwb`, `pwb` or `spwb_day` when `traspirationMode = "Granier"` only):

- `hydraulicRedistributionFraction` [= `0.1`]: Fraction of plant transpiration corresponding to hydraulic redistribution.

Water balance (functions `spwb`, `pwb` or `spwb_day` when `traspirationMode = "Sperry"` or `traspirationMode = "Sureau"`):

- `nsubsteps_canopy` [= `3600`]: Number of substeps into which each step is divided for multi-layer canopy energy balance solving.
- `multiLayerBalance` [= `FALSE`]: Flag to indicate multiple canopy energy balance. If `FALSE`, canopy is considered a single layer for energy balance.
- `sapFluidityVariation` [= `TRUE`]: Flag to indicate that temperature affects sap fluidity (and indirectly plant conductance).

- `TPhase_gmin` [= `37.5`]: Temperature for transition phase of gmin.
- `Q10_1_gmin` [= `1.2`]: Temperature dependance of gmin when T less than or equal to TPhase.
- `Q10_2_gmin` [= `4.8`]: Temperature dependance of gmin when T greater than TPhase.
- `taper` [= `TRUE`]: Whether taper of xylem conduits is accounted for when calculating aboveground stem conductance from xylem conductivity.
- `thermalCapacityLAI` [= `1000000`]: Thermal canopy capacitance per LAI unit.
- `rootRadialConductance` [= `4`]: Radial conductance in roots (mmol·s-1·m-2·MPa-1).
- `averageFracRhizosphereResistance` [= `0.15`]: Fraction to total continuum (leaf+stem+root+rhizosphere) resistance that corresponds to rhizosphere (averaged across soil water potential values).
- `boundaryLayerSize` [= `2000`]: Size of the boundary layer (in cm) over the canopy (relevant for multi-layer canopy energy balance).

Water balance (functions `spwb`, `pwb` or `spwb_day` when `traspirationMode = "Sperry"` only):

- `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.

Water balance (functions `spwb`, `pwb` or `spwb_day` when `traspirationMode = "Sureau"` only):

- `plantCapacitance` [= `TRUE`]: Whether the effect of (symplasmic) plant water compartments is considered in simulations.
- `cavitationFlux` [= `TRUE`]: Whether the effect of water flux generated by cavitation of apoplasmic tissues is considered in simulations.
- `soilDisconnection` [= `FALSE`]: Whether the ability of the plants to physically disconnect their root system from the soil is considered in simulations.
- `leafCuticularTranspiration` [= `TRUE`]: Whether the effect of leaf cuticular transpiration is considered in simulations.
- `stemCuticularTranspiration` [= `TRUE`]: Whether the effect of stem cuticular transpiration is considered in simulations.
- `C_SApoInit` [= `2.0e-5`]: Maximum capacitance of the stem apoplasm (mmol·m-2).
- `C_LApoInit` [= `1.0e-5`]: Maximum capacitance of the leaf apoplasm (mmol·m-2).
- `k_SSym` [= `0.26`]: Conductance from stem apoplasm to stem symplasm (mmol·s-1·m-2·MPa-1).
- `fractionLeafSymplasm` [= `0.5`]: Fraction of the leaf resistance from leaf apoplasm to leaf symplasm ([0-1]).
- `gs_NightFrac` [= `0.05`]: Stomatal conductance at night as fraction of maximum stomatal conductance ([0-1]).
- `stomatalSubmodel` [= `"Baldocchi"`]: Stomatal regulation sub-model, either "Jarvis" or "Baldocchi".
- `JarvisPAR` [= `0.003`]: Parameter regulating the response of stomatal conductance to light (PAR) in the Jarvis model.
- `fTRBToLeaf` [= `0.8`]: Fraction of surface of bark exposed to air per leaf area.

Forest growth (functions `growth` or `growth_day`):

- subdailyCarbonBalance [= FALSE]: Boolean flag to indicate that labile carbon balance should be conducted at sub-daily steps (applies only to transpirationMode = "Sperry").
- allowDessication [= TRUE]: Boolean flag to indicate that mortality by dessication is allowed.
- allowStarvation [= TRUE]: Boolean flag to indicate that mortality by starvation is allowed.
- sinkLimitation [= TRUE]: Boolean flag to indicate that temperature and turgor limitations to growth are applied.
- shrubDynamics [= TRUE]: Boolean flag to allow the application of demographic processes to shrubs.
- herbDynamics [= TRUE]: Boolean flag to allow dynamic herb leaf area as a function of shading due to leaf area of woody cohorts.
- allocationStrategy [= "Al2As"]: Strategy for allocation (either "Plant_kmax", for constant maximum plant conductance, or "Al2As" for constant Huber value).
- phloemConductanceFactor [= 0.2]): Factor to transform stem xylem conductance to stem phloem conductance (only for transpirationMode = "Sperry").
- nonSugarConcentration [= 0.25]: Non-sugar (inorganic) solute concentration (mol·l-1) in cells.
- equilibriumOsmoticConcentration [= c(leaf = 0.8, sapwood = 0.6)]: Equilibrium osmotic concentrations (mol·l-1) for leaf and sapwood cells. The difference between leaf and sapwood values helps maintaining phloem transport. The equilibrium sugar concentration is equilibriumOsmoticConcentration - nonSugarConcentration defaults to [= c(leaf = 0.55, sapwood = 0.35)].
- minimumRelativeStarchForGrowth [= 0.50]: Default minimum concentration of storage carbon (starch), relative to the maximum storage capacity, for sapwood growth to occur, when not specified via SpParams (RSSG).
- constructionCosts [= c(leaf = 1.5, sapwood = 1.47, fineroot = 1.30)]: Default construction costs, including respiration and structural carbon, per dry weight of new tissue (g gluc · g dry -1) when not specified via SpParams (CCleaf, CCsapwood and CCfineroot).
- senescenceRates [= c(sapwood = 0.0001261398, fineroot = 0.001897231)]: Default senescence rates (day-1) for sapwood and fineroots when not specified via SpParams (SRsapwood and SRfineroot). Defaults are equivalent to 9%, 5% and 50% annual turnover for gymnosperm sapwood, angiosperm sapwood and fine roots, respectively.
- maximumRelativeGrowthRates [= c(leaf = 0.09, cambium = 0.005, sapwood = 0.002, fineroot = 0.1)]: Default maximum relative growth rates for leaves (m2 leaf ·cm-2 sapwood· day-1), tree sapwood (cm2 sapwood· cm-1 cambium · day-1), shrub sapwood (cm2 sapwood ·cm-2 sapwood· day-1) and fine roots (g dw · g dw -1 · day -1) when not specified via SpParams (RGRleafmax, RGRcambiummax , RGRsapwoodmax and RGRfinerootmax, respectively).
- mortalityMode [= "density/deterministic"]: String describing how mortality is applied. Current accepted values are combinations of "cohort" vs "density" (for whole-cohort mortality vs reduction of stem density) and "deterministic" vs. "stochastic".
- mortalityBaselineRate [= 0.0015]: Default deterministic proportion or probability specifying the baseline reduction of cohort's density occurring in a year (for mortalityMode = "density/deterministic" or "density/stochastic").

- mortalityRelativeSugarThreshold [= 0.4]: Threshold of stem sugar concentration relative to the equilibrium sugar concentration, resulting in an increased starvation mortality rate/probability whenever levels are below.
- mortalityRWCThreshold [= 0.4]: Threshold of stem relative water content resulting in increased mortality rate/probability whenever levels are below.
- recrTreeDBH [= 1]: Default DBH (cm) for recruited trees (when species parameter RecrTreeDBH is missing).
- recrTreeDensity [= 3000]: Default density (ind·ha-1) for recruited trees (when species parameter RecrTreeDensity is missing).
- ingrowthTreeDBH [= 7.5]: Default DBH (cm) for ingrowth trees (when species parameter RecrTreeDBH is missing).
- ingrowthTreeDensity [= 127]: Default density (ind·ha-1) for ingrowth trees (when species parameter RecrTreeDensity is missing).

Forest dynamics (function fordyn):

- allowSeedBankDynamics [= TRUE]: Boolean flag to indicate that seed production and seed bank dynamics is simulated.
- allowRecruitment [= TRUE]: Boolean flag to indicate that recruitment from seeds is allowed.
- allowResprouting [= TRUE]: Boolean flag to indicate that resprouting is allowed.
- recruitmentMode [= "stochastic"]: String describing how recruitment from seeds is applied. Current accepted values are "deterministic" or "stochastic".
- removeEmptyCohorts [= TRUE]: Boolean flag to indicate the removal of cohorts whose density is too low.
- minimumTreeCohortDensity [= 1]: Threshold of tree density resulting in cohort removal.
- minimumShrubCohortCover [= 0.01]: Threshold of shrub cover resulting in cohort removal.
- dynamicallyMergeCohorts [= TRUE]: Boolean flag to indicate that cohorts should be merged when possible. This option speeds up calculations but results in a loss of cohort identity and reinitialization of many state variables.
- seedRain [= NULL]: Vector of species names whose seed rain is to be added to seed bank, regardless of local seed production.
- seedProductionTreeHeight [= 300]: Default minimum tree height for producing seeds (when species parameter SeedProductionHeight is missing).
- seedProductionShrubHeight [= 30]: Default minimum shrub height for producing seeds (when species parameter SeedProductionHeight is missing).
- probRecr [= 0.05]: Default annual probability of seed-recruitment (when species parameter ProbRecr is missing).
- minTempRecr [= 0]: Default threshold of minimum average temperature of the coldest month necessary for recruiting from seeds (when species parameter MinTempRecr is missing).
- minMoistureRecr [= 0.3]: Default threshold of minimum moisture index (annual precipitation over annual ETP) necessary for seed-recruiting (when species parameter MinMoistureRecr is missing).
- minFPARRecr [= 10]: Default threshold of minimum fraction of PAR (in %) reaching the ground necessary for recruiting (when species parameter MinFPARRecr is missing).

- recrTreeHeight [= 620]: Default height (cm) for recruited trees (when species parameter RecrTreeHeight is missing).
- recrShrubCover [= 1]: Default cover (%) for shrubs recruited from seed (when species parameter RecrShrubCover is missing).
- recrShrubHeight [= 25]: Default height (cm) for recruited shrubs (when species parameter RecrShrubHeight is missing).
- recrTreeZ50 [= 100]: Default value for Z50 (mm) in seed-recruited trees (when species parameter RecrZ50 is missing).
- recrShrubZ50 [= 50]: Default value for Z50 (mm) in seed-recruited shrubs (when species parameter RecrZ50 is missing).
- recrTreeZ95 [= 1000]: Default value for Z95 (mm) in seed-recruited trees (when species parameter RecrZ50 is missing).
- recrShrubZ50 [= 500]: Default value for Z95 (mm) in seed-recruited shrubs (when species parameter RecrZ50 is missing).

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

spwbInput, spwb, growth, fordyn

---

defaultManagementFunction
Default forest management actions

Description

Function defaultManagementFunction implements actions for 'regular' and 'irregular' management models of monospecific or mixed stands, whereas function defaultManagementArguments returns a list with default values for the parameters regulating management. Both functions are meant to be used in simulations with fordyn.

Usage

```
defaultManagementFunction(x, args, verbose = FALSE)

defaultManagementArguments()
```

Arguments

x	An object of class forest
args	A list of arguments regulating management actions, e.g. the list returned by defaultManagementArguments
verbose	A logical flag enabling console printing

Details

This function implements silvicultural actions following either 'regular' or 'irregular' management models. Irregular models are implemented by executing thinning operations only, whereas regular models include both thinning and a set of final cuts. Thinning occurs anytime a stand-level metric (e.g. basal area) crosses a given threshold, and different kinds of thinning operations are allowed. Unrealistic high frequency thinning can be avoided by setting a minimum number of years to happen between thinning operations. Final cuts start whenever mean DBH exceeds a given threshold, and may include different cuts separated a number of years. The function can be applied to target management of specific taxa (instead of assuming a monospecific stand), but the thresholds that determine thinning operations apply to stand-level metrics. Mean DBH will be calculated for the target species only. Planting is only allowed under regular management models, and is applied after the last final cut. Understory clearings are assumed to occur anytime there is an intervention on trees, an only a residual shrub cover is left.

Thinning types:

- `above`: Extract largest trees (according to DBH) until thinning objective is met.
- `below`: Extract smallest trees (according to DBH) until thinning objective is met.
- `systematic`: Extract equally from all size classes until thinning objective is met.
- `above-systematic`: Extract half the objective as systematic thinning and the other hald as above thinning.
- `below-systematic`: Extract half the objective as systematic thinning and the other hald as below thinning.
- `free string`: A string specifying the proportion of tree cuts from size classes, with size classes separated by "/" and each one composed of a number specifying the upper limit and a number indicating its proportion, separated by "-" (e.g. "10-50/40-30/60-20").

Value

Function `defaultManagementFunction` returns a list with the following items:

- `"action"`: A string identifying the action performed (e.g. "thinning").
- `"N_tree_cut"`: A vector with the density of trees removed.
- `"Cover_shrub_cut"`: A vector with the cover of shrubs removed.
- `"planted_forest"`: An object of class `forest` with the new plant cohorts resulting from tree/shrub planting.
- `"management_args"`: A list of management arguments to be used in the next call to the management function.

Function `defaultManagementArguments` returns a list with default arguments:

- `"type"`: Management model, either 'regular' or 'irregular'.
- `"targetTreeSpecies"`: Either `"all"` for unspecific cuttings or a numeric vector of target tree species to be selected for cutting operations.
- `"thinning"`: Kind of thinning to be applied in irregular models or in regular models before the final cuts. Options are 'below', 'above', 'systematic', 'below-systematic', 'above-systematic' or a string with the proportion of cuts to be applied to different diameter sizes (see details).

- "thinningMetric": The stand-level metric used to decide whether thinning is applied, either 'BA' (basal area), 'N' (density) or 'HB' (Hart-Becking index).
- "thinningThreshold": The threshold value of the stand-level metric causing the thinning decision.
- "thinningPerc": Percentage of stand's basal area to be removed in thinning operations.
- "minThinningInterval": Minimum number of years between thinning operations.
- "yearsSinceThinning": State variable to count the years since the last thinning ocurred.
- "finalMeanDBH": Mean DBH threshold to start final cuts.
- "finalPerc": String with percentages of basal area to be removed in final cuts, separated by '-' (e.g. "40-60-100").
- "finalPreviousStage": Integer state variable to store the stage of final cuts ('0' before starting final cuts).
- "finalYearsBetweenCuts": Number of years separating final cuts.
- "finalYearsToCut": State variable to count the years to be passed before new final cut is applied.
- "plantingSpecies": Species code to be planted. If missing, planting does not occur and only natural regeneration is allowed.
- "plantingDBH": Initial DBH (cm) of planted species.
- "plantingHeight": Initial height (cm) of planted species.
- "plantingDensity": Initial density (ind./ha) of the planted species.
- "understoryMaximumCover": Percentage of overall shrub cover to be left after any silvicultural intervention. If missing, shrub cover will not be left unmodified.

Author(s)

Miquel De Cáceres Ainsa, CREAF

Aitor Améztegui, UdL

Jose-Ramon Gonzalez Olabarria, CTFC

See Also

fordyn

Examples

```
# Load example forest object
data(exampleforest)

# Define arguments
args = defaultManagementArguments()

# Call management function
f = defaultManagementFunction(exampleforest, args)

#list names
```

```
names(f)

# Action performed
f$action

# Number of trees cut for each cohort
f$N_tree_cut

# Percent cover of shrubs removed
f$Cover_shrub_cut
```

`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 (between two and five).

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 %) (optional).
- `nitrogen` (= `NA`): Sum of total nitrogen (ammonia, organic and reduced nitrogen) for each layer (in g/kg) (optional).
- `bd` (= `1.5`): Bulk density for each layer (in g/cm3).
- `rfc` (= `c(20,40,60,85)`): Percentage of rock fragment content (volume basis) for each layer.

Note

While this function is limited to five soil layers, user defined data frames can discretize soils using an unlimited number of soil layers.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`soil`, `soil_redefineLayers`, `defaultControl`, `SpParamsMED`

Examples

```
defaultSoilParams(4)
```

`droughtStress` *Drought stress indicators*

Description

Calculates plant drought stress indices, at different temporal scales, from simulation results.

Usage

```
droughtStress(x, index = "NDD", freq = "years", bySpecies = FALSE, draw = TRUE)
```

Arguments

`x`	An object of class `spwb`, `pwb`, `growth` or `fordyn`.
`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.
`draw`	A boolean flag to indicate that a plot should be returned.

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. If `draw=TRUE` a ggplot is returned instead.

Author(s)

Miquel De Cáceres Ainsa, CREAF

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

`summary.spwb`, `waterUseEfficiency`

`evaluation` *Evaluation of simulations results*

Description

Functions to compare model predictions against observed values.

Usage

```
evaluation_table(
  out,
  measuredData,
  type = "SWC",
  cohort = NULL,
  temporalResolution = "day"
)

evaluation_stats(
  out,
  measuredData,
  type = "SWC",
  cohort = NULL,
  temporalResolution = "day"
)

evaluation_plot(
  out,
  measuredData,
  type = "SWC",
  cohort = NULL,
  temporalResolution = "day",
  plotType = "dynamics"
)

evaluation_metric(
  out,
  measuredData,
  type = "SWC",
  cohort = NULL,
  temporalResolution = "day",
  metric = "loglikelihood"
)
```

Arguments

`out`	An object of class `spwb`, `growth` or `pwb`.
`measuredData`	A data frame with observed/measured values. Dates should be in row names, whereas columns should be named according to the type of output to be evaluated (see details).
`type`	A string with the kind of model output to be evaluated. Accepted values are: • `"SWC"`: Soil water content (percent volume). See details for specific soil layers. • `"RWC"`: Relative water content (relative to field capacity). See details for specific soil layers. • `"REW"`: Relative extractable water. See details for specific soil layers. • `"ETR"`: Total evapotranspiration. • `"SE+TR"`: Modelled soil evaporation + plant transpiration against observed total evapotranspiration • `"E"`: Transpiration per leaf area • `"LE"`: Latent heat (vaporisation) turbulent flux • `"H"`: Canopy sensible heat turbulent flux • `"GPP"`: Stand-level gross primary productivity • `"LFMC"`: Live fuel moisture content • `"WP"`: Plant water potentials • `"BAI"`: Basal area increment • `"DI"`: Diameter increment • `"DBH"`: Diameter at breast height • `"Height"`: Plant height
`cohort`	A string of the cohort to be compared (e.g. "T1_68"). If `NULL` results for the first cohort will be evaluated.

`temporalResolution`
: A string to indicate the temporal resolution of the model evaluation, which can be "day", "week", "month" or "year". Observed and modelled values are aggregated temporally (using either means or sums) before comparison.

`plotType`
: Plot type to draw, either `"dynamics"` or `"scatter"`.

`metric`
: An evaluation metric:

  - `"MAE"`: Mean absolute error.
  - `"MAE.rel"`: Mean absolute error in relative terms.
  - `"r"`: Pearson's linear correlation coefficient.
  - `"NSE"`: Nash-Sutcliffe model efficiency coefficient.
  - `"NSE.abs"`: Modified Nash-Sutcliffe model efficiency coefficient (L1 norm) (Legates & McCabe 1999).
  - `"loglikelihood"`: Logarithm of the likelihood of observing the data given the model predictions, assuming independent Gaussian errors.

Details

Users should provide the appropriate columns in `measuredData`, depending on the type of output to be evaluated:

- `"SWC", "RWC", "REW"`: A column with the same name should be present. By default, the first soil layer is compared. Evaluation can be done for specific soil layers, for example using "RWC.2" for the relative water content of the second layer.
- `"ETR"` or `"SE+TR"`: A column named `"ETR"` should be present, containing stand's evapotranspiration in mm/day (or mm/week, mm/month, etc, depending on the temporal resolution). If `type="ETR"` observed values will be compared against modelled evapotranspiration (i.e. sum of transpiration, soil evaporation and interception loss), whereas if `type= "SE+TR"` observed values will be compared against the sum of transpiration and soil evaporation only.
- `"LE"`: A column named `"LE"` should be present containing daily latent heat turbulent flux in MJ/m2.
- `"H"`: A column named `"H"` should be present containing daily sensible heat turbulent flux in MJ/m2.
- `"E"`: For each plant cohort whose transpiration is to be evaluated, a column starting with `"E_"` and continuing with a cohort name (e.g. `"E_T1_68"`) with transpiration in L/m2/day on a leaf area basis (or L/m2/week, L/m2/month, etc, depending on the temporal resolution).
- `"GPP"`: A column named `"GPP"` should be present containing daily gross primary productivity in gC/m2.
- `"LFMC"`: For each plant cohort whose transpiration is to be evaluated, a column starting with `"FCM_"` and continuing with a cohort name (e.g. `"FMC_T1_68"`) with fuel moisture content as percent of dry weight.
- `"WP"`: For each plant cohort whose transpiration is to be evaluated, two columns, one starting with `"PD_"` (for pre-dawn) and the other with `"MD_"` (for midday), and continuing with a cohort name (e.g. `"PD_T1_68"`). They should contain leaf water potential values in MPa. These are compared against sunlit water potentials.

- `"BAI"`: For each plant cohort whose growth is to be evaluated, a column starting with `"BAI_"` and continuing with a cohort name (e.g. `"BAI_T1_68"`) with basal area increment in cm2/day, cm2/week, cm2/month or cm2/year, depending on the temporal resolution.
- `"DI"`: For each plant cohort whose growth is to be evaluated, a column starting with `"DI_"` and continuing with a cohort name (e.g. `"DI_T1_68"`) with basal area increment in cm/day, cm/week, cm/month or cm/year, depending on the temporal resolution.
- `"DBH"`: For each plant cohort whose growth is to be evaluated, a column starting with `"DBH_"` and continuing with a cohort name (e.g. `"DBH_T1_68"`) with DBH values in cm.
- `"Height"`: For each plant cohort whose growth is to be evaluated, a column starting with `"Height_"` and continuing with a cohort name (e.g. `"Height_T1_68"`) with Height values in cm.

Additional columns may exist with the standard error of measured quantities. These should be named as the referred quantity, followed by `"_err"` (e.g. `"PD_T1_68_err"`), and are used to draw confidence intervals around observations.

Row names in `measuredData` indicate the date of measurement (in the case of days). Alternatively, a column called `"dates"` can contain the measurement dates. If measurements refer to months or years, row names should also be in a "year-month-day" format, although with "01" for days and/or months (e.g. "2001-02-01" for february 2001, or "2001-01-01" for year 2001).

Value

- Function `evaluation_table` returns a data frame with dates, observed and predicted values.
- Function `evaluation_stats` returns evaluation statistics (a vector or a data frame depending on `type`):
  - `Bias`: Mean deviation (positive values correspond to model overestimations).
  - `Bias.rel`: Bias in relative terms (%).
  - `MAE`: Mean absolute error.
  - `MAE.rel`: Mean absolute error in relative terms (%).
  - `r`: Pearson's linear correlation coefficient.
  - `NSE`: Nash-Sutcliffe model efficiency coefficient.
  - `NSE.abs`: Modified Nash-Sutcliffe model efficiency coefficient (L1 norm) (Legates & McCabe 1999).
- Function `evaluation_plot` returns a ggplot object.
- Function `evaluation_metric` returns a scalar with the desired metric.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Legates, D.R., McCabe, G.J., 1999. Evaluating the use of "goodness-of-fit" measures in hydrologic and hydroclimatic model validation. Water Resour. Res. 35, 233–241.

See Also

spwb, growth, optimization, exampleobs

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x1 <- spwbInput(exampleforest, examplesoil, SpParamsMED, control)

#Call simulation function
S1 <- spwb(x1, examplemeteo, latitude = 41.82592, elevation = 100)

#Load observed data (in this case the same simulation results with some added error)
data(exampleobs)

#Evaluation statistics for soil water content
evaluation_stats(S1, exampleobs)

#NSE only
evaluation_metric(S1, exampleobs, metric="NSE")

#Comparison of temporal dynamics
evaluation_plot(S1, exampleobs)

#Loglikelihood value
evaluation_metric(S1, exampleobs)
```

---

exampleforest *Example forest stands*

Description

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

Usage

```
data(exampleforest)
data(exampleforest2)
```

Format

An object of class `forest` containing the description of the woody (tree or shrub) cohorts and herb layer of a forest patch. `exampleforest` represents the standard forest inventory description, whereas `exampleforest2` is an alternative forest description where leaf area index and crown ratio are supplied instead of structural (density, DBH and cover) parameters.

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`, `spwbInput`

Examples

```
data(exampleforest)
data(exampleforest2)
```

`examplemeteo` *Example daily meteorology data*

Description

Example data set of meteorological input.

Format

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

`dates` Vector of `Date` objects.

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

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

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

`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).

Source

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

See Also

`spwb`

Examples

```
data(examplemeteo)
```

`exampleobs` *Example observed data*

Description

Example (fake) data set of variables measured in a plot.

Format

A data frame containing daily 'observed' values for year 2001:

`dates` Measurement dates.

`SWC` Soil moisture content (in m3/m3).

`ETR` Total evapotranspiration (mm).

`E_T1_148` Transpiration of Pinus halepensis cohort 'T1_148' (L/m2 of leaf area).

`E_T2_168` Transpiration of Quercus ilex cohort 'T2_168' (L/m2 of leaf area).

`FMC_T1_148` Fuel moisture content of Pinus halepensis cohort 'T1_148' (in percent).

`FMC_T2_168` Fuel moisture content of Quercus ilex cohort 'T2_168' (in percent).

`BAI_T1_148` Basal area increment for Pinus halepensis cohort 'T1_148' (in cm2).

`BAI_T2_168` Basal area increment for Quercus ilex cohort 'T2_168' (in cm2).

`DI_T1_148` Diameter increment for Pinus halepensis cohort 'T1_148' (in cm).

`DI_T2_168` Diameter increment for Quercus ilex cohort 'T2_168' (in cm).

Source

This data set was actually created by running a simulation and adding some gaussian error to the outputs.

See Also

`evaluation`

Examples

```
data(exampleobs)
```

extract *Extracts model outputs*

Description

Function `extract()` extracts daily or subdaily output and returns it as a tidy data frame.

Usage

```
extract(
  x,
  level = "forest",
  output = NULL,
  vars = NULL,
  dates = NULL,
  subdaily = FALSE
)
```

Arguments

`x`	An object returned by simulation functions `spwb`, `pwb` or `growth`.
`level`	Level of simulation output, either "forest" (stand-level results), "soillayer" (soil layer-level results), "cohort" (cohort-level results), "sunlitleaf" or "shadeleaf" (leaf-level results)
`output`	Section of the model output to be explored. See details.
`vars`	Variables to be extracted (by default, all of them).
`dates`	A date vector indicating the subset of simulated days for which output is desired.
`subdaily`	A flag to indicate that subdaily values are desired (see details).

Details

When `subdaily = FALSE`, parameter `output` is used to restrict the section in `x` where variables are located. For example `output = "Plants"` will correspond to variables "LAI", "LAIlive", "Transpiration", "StemPLC",... as returned by a call `names(x$Plants)`.

Option `subdaily = TRUE` only works when simulations have been carried using control option 'subdailyResults = TRUE' (see `defaultControl`). When using `subdaily = TRUE`, parameter `output` is not taken into account, and options for parameter `vars` are the following:

- Variables for `level = "forest"` or `level = "soillayer"`: Not allowed. An error is raised.
- Variables for `level = "cohort"`: "E","Ag","An","dEdP","RootPsi","StemPsi","LeafPsi","StemPLC","StemRWC","Le
- Variables for `level = "shadeleaf"` and `level="sunlitleaf"`: "Abs_SWR","Abs_PAR","Net_LWR","E","Ag","An",

Value

Function `extract()` returns a data frame:

- If `level` = `"forest"`, columns are "date" and variable names.
- If `level` = `"soillayer"`, columns are "date", "soillayer" and variable names.
- If `level` = `"cohort"`, `level` = `"sunlitleaf"` or `level` = `"shadeleaf"`, columns are "date", "cohorts", "species" and variable names.
- If `subdaily` = `TRUE`, columns are "datetime", "cohorts", "species" and variable names.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`summary.spwb`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function (ten days)
S1<-spwb(x, examplemeteo[1:10, ], latitude = 41.82592, elevation = 100)

#Extracts daily forest-level output as a data frame
extract(S1, level = "forest")

#Extracts daily soil layer-level output as a data frame
extract(S1, level = "soillayer")

#Extracts daily cohort-level output as a data frame
extract(S1, level = "cohort")

#Select the output tables/variables to be extracted
extract(S1, level ="cohort", output="Plants", vars = c("PlantStress", "StemPLC"))
```

fireHazard *Fire hazard*

Description

Estimates potential fire behaviour at each daily step of a simulation

Usage

```
fireHazard(
  x,
  SpParams,
  forest = NULL,
  standardConditions = FALSE,
  freq = "days",
  fun = "max"
)
```

Arguments

`x`	An object of class `spwb`, `spwb_day`, `pwb`, `growth`, `growth_day` or `fordyn`.
`SpParams`	A data frame with species parameters (see `SpParamsDefinition` and `SpParamsMED`).
`forest`	An object of class `forest` (needed if `x` is not of class `fordyn`).
`standardConditions`	A logical flag to indicate that standard fire weather conditions are to be used (instead of deriving fuel moisture and windspeed from `x`).
`freq`	Frequency of summary statistics (see `cut.Date`).
`fun`	Summary function (by default, maximum values).

Details

Live fuel moisture of shrub and canopy layers is estimated from plant water status. Dead fuel moisture is estimated following Resco-de-Dios et al. (2015).

Value

A matrix with fire behaviour variables (columns) for each simulated day (rows) or coarser time steps if summaries are requested.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Resco de Dios, V., A. W. Fellows, R. H. Nolan, M. M. Boer, R. A. Bradstock, F. Domingo, and M. L. Goulden. 2015. A semi-mechanistic model for predicting the moisture content of fine litter. Agricultural and Forest Meteorology 203:64–73.

Ruffault J, Limousin JM, Pimont F, Dupuy JL, De Cáceres M, Cochard H, Mouillot F, Blackman C, Torres-Ruiz JM, Parsons R, Moreno M, Delzon S, Jansen S, Olioso A, Choat B, Martin-StPaul N. 2023. Plant hydraulic modelling of leaf and canopy fuel moisture content reveals increasing vulnerability of a Mediterranean forest to wildfires under extreme drought. New Phytologist. (10.1111/nph.18614).

See Also

`spwb`, `fuel_FCCS`, `fire_FCCS`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x1 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S1 <- spwb(x1, examplemeteo, latitude = 41.82592, elevation = 100)

#Evaluate fire hazard
F1 <- fireHazard(S1, SpParamsMED, exampleforest)
```

`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. Live moisture values are drawn from column `ActFCM` in `FCCSpropsSI` if available (see `fuel_FCCS`). Otherwise, moisture values supplied for `MliveSI` are used.
`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, CREAF

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, 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)
```

fire_severity *Fire severity functions*

Description

Functions to estimate fire effects on foliage, buds and cambium, based on the model by Michaletz & Johnson (2008)

Usage

```
fire_plumeTemperature(Ib_surf, z, T_air = 25, rho_air = 1.169)

fire_barkThermalDiffusivity(fmc_bark, rho_bark = 500, T_air = 25)
```

```
fire_radialBoleNecrosis(
  Ib_surf,
  t_res,
  bark_diffusivity,
  T_air = 25,
  rho_air = 1.169,
  T_necrosis = 60
)

fire_leafThermalFactor(SLA, h = 130, c = 2500)

fire_necrosisCriticalTemperature(
  t_res,
  thermal_factor,
  T_air = 25,
  T_necrosis = 60
)

fire_necrosisHeight(
  Ib_surf,
  t_res,
  thermal_factor,
  T_air = 25,
  rho_air = 1.169,
  T_necrosis = 60
)
```

Arguments

`Ib_surf`	Surface fireline intensity (kW/m).
`z`	height (m).
`T_air`	Air temperature (degrees Celsius).
`rho_air`	Air density (kg/m3).
`fmc_bark`	Bark moisture content (% dry weight).
`rho_bark`	Bark density (kg/m3).
`t_res`	fire residence time (seconds).
`bark_diffusivity`	Bark thermal diffusivity (m2/s).
`T_necrosis`	Temperature of tissue necrosis (degrees Celsius).
`SLA`	Specific leaf area (m2/kg).
`h`	Heat transfer coefficient
`c`	Specific heat capacity
`thermal_factor`	Tissue thermal factor.

Value

- Function `fire_plumeTemperature` returns the plume temperature at a given height.
- Function `fire_barkThermalDiffusivity` returns the bark thermal diffusivity given a bark moisture value.
- Function `fire_radialBoleNecrosis` returns the depth of radial bole necrosis in cm.
- Function `fire_leafThermalFactor` returns the thermal factor of leaves as a function of specific leaf area.
- Function `fire_necrosisCriticalTemperature` returns the (plume) temperature yielding necrosis for a given residence time and tissue thermal factor.
- Function `fire_necrosisHeight` returns the height (in m) of necrosis for tissues with given thermal factor.

References

Michaletz, S.T., and Johnson, E.A. 2006. A heat transfer model of crown scorch in forest fires. Can. J. For. Res. 36: 2839–2851. doi:10.1139/X06-158.

Michaletz ST, Johnson EA. 2008. A biophysical process model of tree mortality in surface fires. Canadian Journal of Forest Research 38: 2013–2029.

`fordyn` *Forest dynamics*

Description

Function `fordyn` implements a forest dynamics model that simulates growth, mortality, recruitment and (optionally) management actions in a given forest stand during a period specified in the input climatic data.

Usage

```
fordyn(
  forest,
  soil,
  SpParams,
  meteo,
  control,
  latitude,
  elevation = NA,
  slope = NA,
  aspect = NA,
  CO2ByYear = numeric(0),
  management_function = NULL,
  management_args = NULL
)
```

Arguments

`forest`	An object of class `forest`. Alternatively, the output of a previous run, if continuing a previous simulation.
`soil`	An object of class `data.frame` or `soil`.
`SpParams`	A data frame with species parameters (see `SpParamsMED` and `SpParamsDefinition`).
`meteo`	A data frame with daily weather data series (see `spwb`).
`control`	A list with default control parameters (see `defaultControl`).
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`CO2ByYear`	A named numeric vector with years as names and atmospheric CO2 concentration (in ppm) as values. Used to specify annual changes in CO2 concentration along the simulation (as an alternative to specifying daily values in `meteo`).
`management_function`	A function that implements forest management actions (see details).
`management_args`	A list of additional arguments to be passed to the `management_function`.

Details

Function `fordyn` simulates forest dynamics for annual time steps, building on other simulation functions. For each simulated year, the function performs the following steps:

1. Calls function `growth` to simulate daily water/carbon balance, growth and mortality processes and update the forest object.
2. If required, calls function `management_function`, using as parameters the forest object and `management_args`, which may result in a density reduction for existing plant cohorts and/or a set of new planted cohorts.
3. Simulate natural recruitment (for species present in the stand or given in a seed rain input).
4. Prepares the input of function `growth` for the next annual time step.
5. Store forest status, management arguments, and summaries.

To enable forest management, the user needs to provide a function that implements it, which is passed to `fordyn` via its argument `management_function`. Such function should have the following arguments:

- `"x"`: the `forest` object representing the stand to be managed.
- `"args"`: a list of parameters regulating the behavior of the management function.
- `"verbose"`: a logical flag to enable console output during the execution of the management function.

and return a list with the following elements:

- `"action"`: A string identifying the action performed (e.g. "thinning").

- `"N_tree_cut"`: A vector with the density of trees removed.
- `"Cover_shrub_cut"`: A vector with the cover of shrubs removed.
- `"planted_forest"`: An object of class `forest` with the new plant cohorts resulting from tree/shrub planting.
- `"management_args"`: A list of management arguments to be used in the next call to the management function.

An example of management function is provided in `defaultManagementFunction`.

Value

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

- `"StandSummary"`: A data frame with stand-level summaries (tree basal area, tree density, shrub cover, etc.) at the beginning of the simulation and after each simulated year.
- `"SpeciesSummary"`: A data frame with species-level summaries (tree basal area, tree density, shrub cover, etc.) at the beginning of the simulation and after each simulated year.
- `"CohortSummary"`: A data frame with cohort-level summaries (tree basal area, tree density, shrub cover, etc.) at the beginning of the simulation and after each simulated year.
- `"TreeTable"`: A data frame with tree-cohort data (species, density, diameter, height, etc.) at the beginning of the simulation (if any) and after each simulated year.
- `"DeadTreeTable"`: A data frame with dead tree-cohort data (species, density, diameter, height, etc.) at the beginning of the simulation and after each simulated year.
- `"CutTreeTable"`: A data frame with cut tree data (species, density, diameter, height, etc.) after each simulated year.
- `"ShrubTable"`: A data frame with shrub-cohort data (species, density, cover, height, etc.) at the beginning of the simulation and after each simulated year.
- `"DeadShrubTable"`: A data frame with dead shrub-cohort data (species, density, cover, height, etc.) at the beginning of the simulation (if any) and after each simulated year.
- `"CutShrubTable"`: A data frame with cut shrub data (species, density, cover, height, etc.) after each simulated year.
- `"ForestStructures"`: A list with the `forest` object of the stand at the beginning of the simulation and after each simulated year.
- `"GrowthResults"`: A list with the results of calling function `growth` for each simulated year.
- `"ManagementArgs"`: A list of management arguments to be used in another call to `fordyn`.
- `"NextInputObject"`: An object of class `growthInput` to be used in a subsequent simulation.
- `"NextForestObject"`: An object of class `forest` to be used in a subsequent simulation.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

De Cáceres M, Molowny-Horas R, Cabon A, Martínez-Vilalta J, Mencuccini M, García-Valdés R, Nadal-Sala D, Sabaté S, Martin-StPaul N, Morin X, D'Adamo F, Batllori E, Améztegui A (2023) MEDFATE 2.9.3: A trait-enabled model to simulate Mediterranean forest function and dynamics at regional scales. Geoscientific Model Development 16: 3165-3201 (https://doi.org/10.5194/gmd-16-3165-2023).

See Also

`growth`, `regeneration`, `plot.growth`, `defaultManagementFunction`

Examples

```
#Load example daily meteorological data
data(examplemeteo)
#Prepare a two-year meteorological data with half precipitation during
#the second year
meteo2001 <- examplemeteo
meteo2002 <- examplemeteo
meteo2002$Precipitation <- meteo2002$Precipitation/2
meteo2002$dates <- seq(as.Date("2002-01-01"),
                           as.Date("2002-12-31"), by="day")
meteo_01_02 <- rbind(meteo2001, meteo2002)

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Initialize control parameters
control <- defaultControl("Granier")

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Call simulation function
fd<-fordyn(exampleforest, examplesoil,
           SpParamsMED, meteo_01_02, control,
           latitude = 41.82592, elevation = 100)

#Stand-level summaries
fd$StandSummary

#Tree table by annual steps
fd$TreeTable

#Dead tree table by annual steps
fd$DeadTreeTable
```

`forest` *Forest description*

Description

Description of a forest stand

Usage

```
emptyforest(ntree = 0, nshrub = 0, nseed = 0)

## S3 method for class 'forest'
summary(object, SpParams, ...)

## S3 method for class 'summary.forest'
print(x, digits = getOption("digits"), ...)
```

Arguments

`ntree, nshrub`	Number of tree and shrub cohorts, respectively.
`nseed`	Number of species in the seed bank.
`object`	An object of class `forest` has the following structure (see details):

- `treeData`: A data frame of tree cohorts (in rows) and the following columns:
  - `Species`: String with species (taxon) name or a non-negative integer for tree species identity (i.e., 0,1,2,...) matching SpParams.
  - `Height`: Total tree height (in cm).
  - `DBH`: Tree diameter at breast height (in cm).
  - `N`: Density (number of individuals/hectare) that the measured tree represents.
  - `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`: String with species (taxon) name or a non-negative integer for shrub species identity (i.e., 0,1,2,...) matching SpParams.
  - `Height`: Average total height of plants (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.
- `herbCover`: Percent cover of the herb layer (optional).
- `herbHeight`: Mean height (in cm) of the herb layer (optional).
- `seedBank`: A data frame containing seed bank information with the following columns:

- `Species`: String with species (taxon) name or a non-negative integer for tree species identity (i.e., 0,1,2,...) matching SpParams.
- `Percent`: Amount of seeds in relation to full seed bank (in %).

`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`...`	Additional parameters for functions `summary` and `print`.
`x`	The object returned by `summary.forest`.
`digits`	Minimal number of significant digits.

Details

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

The structure presented above for `forest` objects corresponds to the required data elements. A `forest` object can contain additional information when this is available. Data frames `treeData` and `shrubData` can contain additional columns:

- `LAI`: Leaf area index (m2/m2)
- `FoliarBiomass`: Standing dry biomass of leaves (kg/m2)
- `FuelLoading`: Fine fuel loading (kg/m2)
- `CrownRatio`: The ratio between crown length and total height (between 0 and 1)

Similarly, one can define `forest` list elements `herbLAI`, `herbFoliarBiomass` or `herbFuelLoading`. All these values are used to override allometry-based estimates of those variables when initializing inputs for functions `spwb` or `spwb_day`. Note that leaf area index, foliar biomass and fuel loading are related entities, and they are treated as such in medfate. Therefore, users are expected to supply one or the other, and not all of them at the same time.

Value

Function `summary.forest` returns a list with several structural attributes, such as the basal area and LAI of the forest. Function `emptyforest` returns an empty `forest` object.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`exampleforest`, `forest_mapWoodyTables`, `forest_mergeTrees`, `plot.forest`, `tree2forest`

Examples

```
data(exampleforest)
data(SpParamsMED)

# Prints forest as a list of data items
exampleforest
```

```
# Summary of example forest
summary(exampleforest, SpParamsMED)
```

forest2aboveground *Input for simulation models (deprecated)*

Description

Functions `forest2spwbInput()` and `forest2growthInput()` take an object of class `forest` and a soil data input to create input objects for simulation functions `spwb` (or `pwb`) and `growth`, respectively. Function `forest2aboveground()` calculates aboveground variables such as leaf area index. Function `forest2belowground()` calculates belowground variables such as fine root distribution.

Usage

```
forest2aboveground(x, SpParams, gdd = NA_real_, loading = FALSE)

forest2belowground(x, soil, SpParams)

forest2spwbInput(x, soil, SpParams, control)

forest2growthInput(x, soil, SpParams, control)
```

Arguments

`x`	An object of class `forest`.
`SpParams`	A data frame with species parameters (see `SpParamsDefinition` and `SpParamsMED`).
`gdd`	Growth degree days to account for leaf phenology effects (in Celsius). This should be left `NA` in most applications.
`loading`	A logical flag to indicate that fuel loading should be included (for fire hazard calculations).
`soil`	An object of class `data.frame` or `soil`, containing soil parameters per soil layer.
`control`	A list with default control parameters (see `defaultControl`).

Details

Function `forest2aboveground()` extracts height and species identity from plant cohorts of `x`, and calculate leaf area index and crown ratio.

IMPORTANT NOTE: Function names `forest2spwbInput()` and `forest2growthInput()` are now deprecated, but they can still be used for back-compatibility. They correspond to functions `spwbInput` and `growthInput`

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), includes leaves in winter dormant buds.
- `LAI_expanded`: Leaf area index of expanded leaves (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).
- `Loading`: Fine fuel loading (kg/m2), only if `loading = TRUE`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`spwbInput`, `soil`, `forest`, `SpParamsMED`, `defaultSoilParams`, `plant_ID`

Examples

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

#Default species parameterization
data(SpParamsMED)

# Aboveground parameters
forest2aboveground(exampleforest, SpParamsMED)

# Example of aboveground parameters taken from a forest
# described using LAI and crown ratio
data(exampleforest2)
forest2aboveground(exampleforest2, SpParamsMED)

# Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

# Bewowground parameters (distribution of fine roots)
forest2belowground(exampleforest, examplesoil, SpParamsMED)
```

`forest_mapWoodyTables` *Map forest plot data*

Description

Mapping functions to facilitate building forest objects from forest plot data

Usage

```
forest_mapTreeTable(x, mapping_x, SpParams, plot_size_x = NULL)

forest_mapShrubTable(y, mapping_y, SpParams, plot_size_y = NULL)

forest_mapWoodyTables(
  x = NULL,
  y = NULL,
  mapping_x = NULL,
  mapping_y = NULL,
  SpParams,
  plot_size_x = NULL,
  plot_size_y = NULL
)
```

Arguments

`x`	A data frame with tree records in rows and attributes in columns. Tree records can correspond to individual trees or groups of trees with an associated density.
`mapping_x`	A named character vector to specify mappings of columns in `x` into attributes of `treeData` data frames. Accepted names (and the corresponding specifications for the columns in `x` are:
`SpParams`	A data frame with species parameters (see `SpParamsMED`) from which valid species names are drawn.
`plot_size_x`	The size of tree plot sampled area (in m2). Alternatively, 'plot_size_x' can be a column in `x` and specified in `mapping_x` to indicate that trees have been measured in different subplots and, therefore, they represent different densities per hectare.
`y`	A data frame with shrub records in rows and attributes in columns. Records can correspond to individual shrubs (with crown dimensions and height) or groups of shrubs with an associated cover estimate.
`mapping_y`	A named character vector to specify mappings of columns in `y` into attributes of `shrubData` data frames. Accepted names (and the corresponding specifications for the columns in `y`) are: • "Species": Species code (should follow codes in `SpParams`). • "Species.name": Species name. In this case, the species code will be drawn by matching names with species names in `SpParams`.

- "N": Tree density (in ind./ha).
- "Cover": Shrub cover (in %).
- "D1": Shrub largest crown diameter (in cm).
- "D2": Shrub crown diameter orthogonal to the largest one (in cm).
- "plot.size": Plot size (in m2) to which each record refers to. This is used to calculate tree density (stems per hectare) when not supplied or shrub cover when shrub data is given at the individual level.
- "DBH": Diameter at breast height (in cm).
- "Height": Tree or shrub height (in cm).
- "Z50": Depth (in mm) corresponding to 50% of fine roots.
- "Z95": Depth (in mm) corresponding to 95% of fine roots.

`plot_size_y`	The size of shrub plot sampled area (in m2). Alternatively, 'plot_size_y' can be a column in `y` and specified in `mapping_y` to indicate that shrubs have been measured in different subplots and, therefore, they represent different cover values.

Value

Functions `forest_mapTreeTable` and `forest_mapShrubTable` return a data frame with the structure of `treeData` and `shrubData` from `forest` objects. Function `forest_mapWoodyTable` returns directly a `forest` object.

Author(s)

Miquel De Cáceres Ainsa, EMF-CREAF

See Also

`forest`, `poblet_trees`, `forest_mergeTrees`, `tree2forest`

Examples

```
# Load species parameters
data(SpParamsMED)

# Create an empty forest object
f <- emptyforest()

# (1) Mapping tree data
# Load Poblet tree data
data(poblet_trees)

# Subset control plot
x <- subset(poblet_trees, Plot.Code=="POBL_CTL")

# Estimate sampled area (15-m radius plot)
sampled_area <- pi*15^2

# Define mapping
```

```
mapping_x <- c("Species.name" = "Species", "DBH" = "Diameter.cm")

# Map tree data for plot 'POBL_CTL'
f$treeData <- forest_mapTreeTable(x,
                    mapping_x = mapping_x, SpParams = SpParamsMED,
                    plot_size_x = sampled_area)

# (2) Mapping shrub individual data
#
# Create the individual shrub data frame
species <- c("Erica arborea","Cistus albidus", "Erica arborea", "Cistus albidus", "Cistus albidus")
H <- c(200,50,100,40,30)
D1 <- c(140,40,100, 35,30)
D2 <- D1
y <- data.frame(species, H, D1, D2)

# Define mapping (D1 and D2 map to variables with the same name)
mapping_y <- c("Species.name"= "species", "Height" ="H", "D1", "D2")

# Map individual shrub data to cover data (here each individual becomes a cohort)
# assuming that the sampled area was 4 m2
f$shrubData <- forest_mapShrubTable(y,
                     mapping_y = mapping_y, SpParams = SpParamsMED,
                     plot_size_y = 4)

# (3) Print forest attributes
summary(f, SpParamsMED)

# (4) Forest initialization in a single step
f <- forest_mapWoodyTables(x, y,
                           mapping_x = mapping_x, mapping_y = mapping_y,
                           SpParams = SpParamsMED,
                           plot_size_x = sampled_area, plot_size_y = 4)
summary(f, SpParamsMED)
```

`forest_simplification` *Forest complexity reduction*

Description

Functions `forest_mergeTrees` and `forest_mergeShrubs` merge cohorts of a `forest` object. Function `forest_reduceToDominant` performs a strongest simplification of plant cohorts (see details).

Usage

```
forest_mergeTrees(x, byDBHclass = TRUE)

forest_mergeShrubs(x, byHeightclass = TRUE)

forest_reduceToDominant(x, SpParams)
```

Arguments

x	An object of class forest.
byDBHclass	Logical flag to indicate that 5-cm tree DBH classes should be kept separated.
byHeightclass	Boolean flag to indicate that 10-cm shrub height classes should be kept separated.
SpParams	A data frame with species parameters (see SpParamsDefinition and SpParamsMED).

Details

Tree DBH classes are defined in 5-cm intervals, whereas shrub height classes are defined in 10-cm intervals. Tree DBH and shrub height classes are defined up to a specific size (i.e. larger plants are not merged) corresponding to 52.5 cm and 90 cm, respectively.

Function `forest_reduceToDominant` simplifies the input forest to the tree cohort of highest LAI, among those of the tree species with highest LAI. The leaf area index of the whole tree layer will be attributed to the chosen cohort. The same is performed for the shrub layer.

Value

Another forest object with simplified structure/composition, depending on the function.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

spwb, forest, forest_mapWoodyTables, fordyn, summary.forest

Examples

```
# Example forest data
data("exampleforest")

# Reduce to dominant tree and dominant shrub
reduced <- forest_reduceToDominant(exampleforest, SpParamsMED)

# Check that overall LAI does not change
stand_LAI(exampleforest, SpParamsMED)
stand_LAI(reduced, SpParamsMED)
```

`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).

Usage

```
fuel_stratification(
  object,
  SpParams,
  gdd = NA_real_,
  heightProfileStep = 10,
  maxHeightProfile = 5000,
  bulkDensityThreshold = 0.05
)

fuel_FCCS(
  object,
  SpParams,
  cohortFMC = as.numeric(c()),
  gdd = NA_real_,
  heightProfileStep = 10,
  maxHeightProfile = 5000,
  bulkDensityThreshold = 0.05,
  depthMode = "crownaverage"
)

fuel_windAdjustmentFactor(
  topShrubHeight,
  bottomCanopyHeight,
  topCanopyHeight,
  canopyCover
)
```

Arguments

`object`	An object of class `forest`
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`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.
`cohortFMC`	A numeric vector of (actual) fuel moisture content by cohort.
`depthMode`	Specifies how fuel depth (and therefore canopy and understory bulk density) should be estimated: • `"crownaverage"`: As weighed average of crown lengths using loadings as weights. • `"profile"`: As the difference of base and top heights in bulk density profiles. • `"absoluteprofile"`: As the difference of absolute base and absolute top heights in bulk density profiles.
`topShrubHeight`	Shrub stratum top height (in m).
`bottomCanopyHeight`	Canopy base height (in m).
`topCanopyHeight`	Canopy top height (in m).
`canopyCover`	Canopy percent cover.

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).

- 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).
- ActFMC: Actual fuel moisture content (as percent over dry weight). These are set to NA if parameter cohortFMC is empty.

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, CREAF

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, SpParamsMED)
fccs

fuel_windAdjustmentFactor(fccs$htc[2], fccs$hbc[1], fccs$htc[1], fccs$cover[1])
```

---

growth *Forest growth*

Description

Function `growth` is a process-based model that performs energy, water and carbon balances; and determines changes in water/carbon pools, functional variables (leaf area, sapwood area, root area) and structural ones (tree diameter, tree height, shrub cover) for woody plant cohorts in a given forest stand during a period specified in the input climatic data.

Usage

```
growth(
  x,
  meteo,
  latitude,
  elevation,
  slope = NA_real_,
  aspect = NA_real_,
  CO2ByYear = numeric(0),
  waterTableDepth = NA_real_
)
```

Arguments

`x`	An object of class `growthInput`.
`meteo`	A data frame with daily meteorological data series (see `spwb`).
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`CO2ByYear`	A named numeric vector with years as names and atmospheric CO2 concentration (in ppm) as values. Used to specify annual changes in CO2 concentration along the simulation (as an alternative to specifying daily values in `meteo`).
`waterTableDepth`	Water table depth (in mm). When not missing, capillarity rise will be allowed if lower than total soil depth.

Details

Detailed model description is available in the medfate book. Simulations using the 'Sperry' or 'Sureau' transpiration modes are computationally much more expensive than those using the 'Granier' transpiration mode.

Value

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

- `"latitude"`: Latitude (in degrees) given as input.
- `"topography"`: Vector with elevation, slope and aspect given as input.
- `"weather"`: A copy of the input weather data frame.
- `"growthInput"`: A copy of the object `x` of class `growthInput` given as input.
- `"growthOutput"`: An copy of the final state of the object `x` of class `growthInput`.
- `"WaterBalance"`: A data frame where different water balance variables (see `spwb`).
- `"EnergyBalance"`: A data frame with the daily values of energy balance components for the soil and the canopy (only for `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`; see `spwb`).
- `"CarbonBalance"`: A data frame where different stand-level carbon balance components (gross primary production, maintenance respiration, synthesis respiration and net primary production), all in g C · m-2.
- `"BiomassBalance"`: A data frame with the daily values of stand biomass balance components (in g dry · m-2.
- `"Temperature"`: A data frame with the daily values of minimum/mean/maximum temperatures for the atmosphere (input), canopy and soil (only for `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`; see `spwb`).
- `"Soil"`: A data frame where different soil variables (see `spwb`).
- `"Stand"`: A data frame where different stand-level variables (see `spwb`).
- `"Plants"`: A list of daily results for plant cohorts (see `spwb`).
- `"SunlitLeaves"` and `"ShadeLeaves"`: A list with daily results for sunlit and shade leaves (only for `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`; see `spwb`).
- `"LabileCarbonBalance"`: A list of daily labile carbon balance results for plant cohorts, with elements:
  - `"GrossPhotosynthesis"`: Daily gross photosynthesis per dry weight of living biomass (g gluc · g dry-1).
  - `"MaintentanceRespiration"`: Daily maintenance respiration per dry weight of living biomass (g gluc · g dry-1).
  - `"GrowthCosts"`: Daily growth costs per dry weight of living biomass (g gluc · g dry-1).
  - `"RootExudation"`: Root exudation per dry weight of living biomass (g gluc · g dry-1).
  - `"LabileCarbonBalance"`: Daily labile carbon balance (photosynthesis - maintenance respiration - growth costs - root exudation) per dry weight of living biomass (g gluc · g dry-1).
  - `"SugarLeaf"`: Sugar concentration (mol·l-1) in leaves.

  - `"StarchLeaf"`: Starch concentration (mol·l-1) in leaves.
  - `"SugarSapwood"`: Sugar concentration (mol·l-1) in sapwood.
  - `"StarchSapwood"`: Starch concentration (mol·l-1) in sapwood.
  - `"SugarTransport"`: Average instantaneous rate of carbon transferred between leaves and stem compartments via floem (mol gluc·s-1).
- `"PlantBiomassBalance"`: A list of daily plant biomass balance results for plant cohorts, with elements:
  - `"StructuralBiomassBalance"`: Daily structural biomass balance (g dry · m-2).
  - `"LabileBiomassBalance"`: Daily labile biomass balance (g dry · m-2).
  - `"PlantBiomassBalance"`: Daily plant biomass balance, i.e. labile change + structural change (g dry · m-2).
  - `"MortalityBiomassLoss"`: Biomass loss due to mortality (g dry · m-2).
  - `"CohortBiomassBalance"`: Daily cohort biomass balance (including mortality) (g dry · m-2).
- `"PlantStructure"`: A list of daily area and biomass values for compartments of plant cohorts, with elements:
  - `"LeafBiomass"`: Daily amount of leaf structural biomass (in g dry) for an average individual of each plant cohort.
  - `"SapwoodBiomass"`: Daily amount of sapwood structural biomass (in g dry) for an average individual of each plant cohort.
  - `"FineRootBiomass"`: Daily amount of fine root biomass (in g dry) for an average individual of each plant cohort.
  - `"LeafArea"`: Daily amount of leaf area (in m2) for an average individual of each plant cohort.
  - `"SapwoodArea"`: Daily amount of sapwood area (in cm2) for an average individual of each plant cohort.
  - `"FineRootArea"`: Daily amount of fine root area (in m2) for an average individual of each plant cohort.
  - `"HuberValue"`: The ratio of sapwood area to (target) leaf area (in cm2/m2).
  - `"RootAreaLeafArea"`: The ratio of fine root area to (target) leaf area (in m2/m2).
  - `"DBH"`: Diameter at breast height (in cm) for an average individual of each plant cohort.
  - `"Height"`: Height (in cm) for an average individual of each plant cohort.
- `"GrowthMortality"`: A list of daily growth and mortality rates for plant cohorts, with elements:
  - `"LAgrowth"`: Leaf area growth (in m2·day-1) for an average individual of each plant cohort.
  - `"SAgrowth"`: Sapwood area growth rate (in cm2·day-1) for an average individual of each plant cohort.
  - `"FRAgrowth"`: Fine root area growth (in m2·day-1) for an average individual of each plant cohort.
  - `"StarvationRate"`: Daily mortality rate from starvation (ind/d-1).
  - `"DessicationRate"`: Daily mortality rate from dessication (ind/d-1).
  - `"MortalityRate"`: Daily mortality rate (any cause) (ind/d-1).
- `"subdaily"`: A list of objects of class `growth_day`, one per day simulated (only if required in `control` parameters, see `defaultControl`).

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

De Cáceres M, Molowny-Horas R, Cabon A, Martínez-Vilalta J, Mencuccini M, García-Valdés R, Nadal-Sala D, Sabaté S, Martin-StPaul N, Morin X, D'Adamo F, Batllori E, Améztegui A (2023) MEDFATE 2.9.3: A trait-enabled model to simulate Mediterranean forest function and dynamics at regional scales. Geoscientific Model Development 16: 3165-3201 (https://doi.org/10.5194/gmd-16-3165-2023).

See Also

`growthInput`, `growth_day`, `plot.growth`

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("Granier")

#Initialize soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize model input
x1 <- growthInput(exampleforest, examplesoil, SpParamsMED, control)

#Call simulation function
G1 <- growth(x1, examplemeteo, latitude = 41.82592, elevation = 100)


#Switch to 'Sperry' transpiration mode
control <- defaultControl("Sperry")

#Initialize model input
x2 <- growthInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
G2 <-growth(x2, examplemeteo, latitude = 41.82592, elevation = 100)

#Switch to 'Sureau' transpiration mode
control <- defaultControl("Sureau")
```

```
#Initialize model input
x3 <- growthInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
G3 <-growth(x3, examplemeteo, latitude = 41.82592, elevation = 100)
```

growth_day *Single-day simulation*

Description

Function spwb_day performs water balance for a single day and growth_day performs water and carbon balance for a single day.

Usage

```
growth_day(
  x,
  date,
  meteovec,
  latitude,
  elevation,
  slope = NA_real_,
  aspect = NA_real_,
  runon = 0,
  lateralFlows = NULL,
  waterTableDepth = NA_real_,
  modifyInput = TRUE
)

spwb_day(
  x,
  date,
  meteovec,
  latitude,
  elevation,
  slope = NA_real_,
  aspect = NA_real_,
  runon = 0,
  lateralFlows = NULL,
  waterTableDepth = NA_real_,
  modifyInput = TRUE
)
```

Arguments

`x`	An object of class `spwbInput` or `growthInput`.
`date`	Date as string "yyyy-mm-dd".
`meteovec`	A named numerical vector with weather data. See variable names in parameter `meteo` of `spwb`.
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`runon`	Surface water amount running on the target area from upslope (in mm).
`lateralFlows`	Lateral source/sink terms for each soil layer (interflow/to from adjacent locations) as mm/day.
`waterTableDepth`	Water table depth (in mm). When not missing, capillarity rise will be allowed if lower than total soil depth.
`modifyInput`	Boolean flag to indicate that the input `x` object is allowed to be modified during the simulation.

Details

The simulation functions allow using three different sub-models of transpiration and photosynthesis:

- The sub-model corresponding to 'Granier' transpiration mode is illustrated by function `transp_transpirationGranie` and was described in De Caceres et al. (2015), and implements an approach originally described in Granier et al. (1999).
- The sub-model corresponding to 'Sperry' transpiration mode is illustrated by function `transp_transpirationSperry` and was described in De Caceres et al. (2021), and implements a modelling approach originally described in Sperry et al. (2017).
- The sub-model corresponding to 'Sureau' transpiration mode is illustrated by function `transp_transpirationSureau` and was described for model SurEau-Ecos v2.0 in Ruffault et al. (2022).

Simulations using the 'Sperry' or 'Sureau' transpiration mode are computationally much more expensive than 'Granier'.

Value

Function `spwb_day()` returns a list of class `spwb_day` with the following elements:

- `"cohorts"`: A data frame with cohort information, copied from `spwbInput`.
- `"topography"`: Vector with elevation, slope and aspect given as input.
- `"weather"`: A vector with the input weather.
- `"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:

- `"Psi"`: Soil water potential (in MPa) at the end of the day.
- `"HerbTranspiration"`: Water extracted by herbaceous plants from each soil layer (in mm).
- `"HydraulicInput"`: Water entering each soil layer from other layers, transported via plant roots (in mm).
- `"HydraulicOutput"`: Water leaving each soil layer (going to other layers or the transpiration stream) (in mm).
- `"PlantExtraction"`: Water extracted by woody plants from each soil layer (in mm).

- `"Stand"`: A named vector with with stand values for the simulated day, equivalent to one row of 'Stand' object returned by `spwb`.
- `"Plants"`: A data frame of results for each plant cohort (see `transp_transpirationGranier` or `transp_transpirationSperry`).

The following items are only returned when `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`:

- `"EnergyBalance"`: Energy balance of the stand (see `transp_transpirationSperry`).
- `"RhizoPsi"`: Minimum water potential (in MPa) inside roots, after crossing rhizosphere, per cohort and soil layer.
- `"SunlitLeaves"` and `"ShadeLeaves"`: For each leaf type, a data frame with values of LAI, Vmax298 and Jmax298 for leaves of this type in each plant cohort.
- `"ExtractionInst"`: Water extracted by each plant cohort during each time step.
- `"PlantsInst"`: A list with instantaneous (per time step) results for each plant cohort (see `transp_transpirationSperry`).
- `"LightExtinction"`: A list of information regarding radiation balance through the canopy, as returned by function `light_instantaneousLightExtinctionAbsortion`.
- `"CanopyTurbulence"`: Canopy turbulence (see `wind_canopyTurbulence`).

Author(s)

- Miquel De Cáceres Ainsa, CREAF
- Nicolas Martin-StPaul, URFM-INRAE

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).

De Cáceres M, Mencuccini M, Martin-StPaul N, Limousin JM, Coll L, Poyatos R, Cabon A, Granda V, Forner A, Valladares F, Martínez-Vilalta J (2021) Unravelling the effect of species mixing on water use and drought stress in holm oak forests: a modelling approach. Agricultural and Forest Meteorology 296 (doi:10.1016/j.agrformet.2020.108233).

Granier A, Bréda N, Biron P, Villette S (1999) A lumped water balance model to evaluate duration and intensity of drought constraints in forest stands. Ecol Modell 116:269–283. https://doi.org/10.1016/S0304-3800(98)00205-1.

Ruffault J, Pimont F, Cochard H, Dupuy JL, Martin-StPaul N (2022) SurEau-Ecos v2.0: a trait-based plant hydraulics model for simulations of plant water status and drought-induced mortality at the ecosystem level. Geoscientific Model Development 15, 5593-5626 (doi:10.5194/gmd-15-5593-2022).

Sperry, J. S., M. D. Venturas, W. R. L. Anderegg, M. Mencuccini, D. S. Mackay, Y. Wang, and D. M. Love. 2017. Predicting stomatal responses to the environment from the optimization of photosynthetic gain and hydraulic cost. Plant Cell and Environment 40, 816-830 (doi: 10.1111/pce.12852).

See Also

`spwbInput`, `spwb`, `plot.spwb_day`, `growthInput`, `growth`, `plot.growth_day`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil parameters
examplesoil <- defaultSoilParams(4)

# Day to be simulated
d <- 100
meteovec <- unlist(examplemeteo[d,-1])
date <- as.character(examplemeteo$dates[d])

#Simulate water balance one day only (Granier mode)
control <- defaultControl("Granier")
x1 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)
sd1 <- spwb_day(x1, date, meteovec,
                latitude = 41.82592, elevation = 100, slope=0, aspect=0)

#Simulate water balance for one day only (Sperry mode)
control <- defaultControl("Sperry")
x2 <- spwbInput(exampleforest, examplesoil, SpParamsMED, control)
sd2 <-spwb_day(x2, date, meteovec,
              latitude = 41.82592, elevation = 100, slope=0, aspect=0)

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

#Simulate water balance for one day only (Sureau mode)
control <- defaultControl("Sureau")
x3 <- spwbInput(exampleforest, examplesoil, SpParamsMED, control)
sd3 <-spwb_day(x3, date, meteovec,
              latitude = 41.82592, elevation = 100, slope=0, aspect=0)
```

```
#Simulate water and carbon balance for one day only (Granier mode)
control <- defaultControl("Granier")
x4  <- growthInput(exampleforest,examplesoil, SpParamsMED, control)
sd4 <- growth_day(x4, date, meteovec,
                latitude = 41.82592, elevation = 100, slope=0, aspect=0)
```

herb_values *Herbaceous description functions*

Description

Functions to calculate attributes of the herbaceous component of a `forest` object

Usage

```
herb_foliarBiomass(x, SpParams)

herb_fuelLoading(x, SpParams)

herb_LAI(x, SpParams)
```

Arguments

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

Value

A single scalar:

- `herb_foliarBiomass`: Herbaceous biomass of leaves (in kg/m2).
- `herb_fuelLoading`: Herbaceous fine fuel loading (in kg/m2).
- `herb_LAI`: Herbaceous leaf area index (m2/m2).

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

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

hydraulics_conductancefunctions

Hydraulic confuctance functions

Description

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

Usage

```
hydraulics_psi2K(psi, psi_extract, exp_extract = 3)

hydraulics_K2Psi(K, psi_extract, exp_extract = 3)

hydraulics_averagePsi(psi, v, exp_extract, psi_extract)

hydraulics_xylemConductance(psi, kxylemmax, c, d)

hydraulics_xylemPsi(kxylem, kxylemmax, c, d)

hydraulics_psiCrit(c, d, pCrit = 0.001)

hydraulics_vanGenuchtenConductance(psi, krhizomax, n, alpha)

hydraulics_correctConductanceForViscosity(kxylem, temp)

hydraulics_psi2Weibull(psi50, psi88 = NA_real_, psi12 = NA_real_)

hydraulics_vulnerabilityCurvePlot(
  x,
  soil = NULL,
  type = "leaf",
  vulnerabilityFunction = "Weibull",
  psiVec = seq(-0.1, -8, by = -0.01),
  relative = FALSE,
  speciesNames = FALSE,
  draw = TRUE,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`psi`	A scalar (or a vector, depending on the function) with water potential (in MPa).
`psi_extract`	Soil water potential (in MPa) corresponding to 50% whole-plant relative transpiration.

`exp_extract`	Exponent of the whole-plant relative transpiration Weibull function.
`K`	Whole-plant relative conductance (0-1).
`v`	Proportion of fine roots within each soil layer.
`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).
`kxylem`	Xylem hydraulic conductance (defined as flow per surface unit and per pressure drop).
`pCrit`	Proportion of maximum conductance considered critical for hydraulic functioning.
`krhizomax`	Maximum rhizosphere hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
`n, alpha`	Parameters of the Van Genuchten function (rhizosphere vulnerability curve).
`temp`	Temperature (in degrees Celsius).
`psi50, psi88, psi12`	Water potentials (in MPa) corresponding to 50%, 88% and 12% percent conductance loss.
`x`	An object of class `spwbInput`.
`soil`	A list containing the description of the soil (see `soil`).
`type`	Plot type for `hydraulics_vulnerabilityCurvePlot`, either `"leaf"`, `"stem"`, `"root"` or `"rhizosphere"`).
`vulnerabilityFunction`	String indicating the function used to represent vulnerability in plant segments, either "Weibull" or "Sigmoid".
`psiVec`	Vector of water potential values to evaluate for the vulnerability curve.
`relative`	A flag to relativize vulnerability curves to the [0-1] interval.
`speciesNames`	A flag to indicate the use of species names instead of cohort names in plots.
`draw`	A flag to indicate whether the vulnerability curve should be drawn or just returned.
`ylim, xlab, ylab`	Graphical parameters to override function defaults.

Details

Details of plant hydraulic models are given the medfate book. 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, CREAF

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 and 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

```
#Manual display of vulnerability curve
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)
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))

#Load example dataset
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Initialize soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)
```

```
#Initialize control parameters
control <- defaultControl("Granier")

#Switch to 'Sperry' transpiration mode
control <- defaultControl("Sperry")

#Initialize input
x <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Leaf vulnerability curves
hydraulics_vulnerabilityCurvePlot(x, type="leaf")

#Stem vulnerability curves
hydraulics_vulnerabilityCurvePlot(x, type="stem")
```

`hydraulics_defoliation` *Hydraulic-related defoliation*

Description

Functions to calculate the proportion of crown defoliation due to hydraulic disconnection.

Usage

```
hydraulics_proportionDefoliationSigmoid(
  psiLeaf,
  P50,
  slope,
  PLC_crit = 0.88,
  P50_cv = 10
)

hydraulics_proportionDefoliationWeibull(
  psiLeaf,
  c,
  d,
  PLC_crit = 0.88,
  P50_cv = 10
)
```

Arguments

`psiLeaf`	Leaf water potential (in MPa).
`P50, slope`	Parameters of the Sigmoid function.
`PLC_crit`	Critical leaf PLC corresponding to defoliation

`P50_cv`	Coefficient of variation (in percent) of leaf P50, to describe the variability in hydraulic vulnerability across crown leaves.
`c, d`	Parameters of the Weibull function.

Details

The functions assume that crowns are made of a population of leaves whose hydraulic vulnerability (i.e. the water potential corresponding to 50 follows a Gaussian distribution centered on the input P50 and with a known coefficient of variation (`P50_cv`). The slope parameter (or the c exponent in the case of a Weibull function) is considered constant. Leaves are hydraulically disconnected, and shedded, when their embolism rate exceeds a critical value (`PLC_crit`).

Value

The proportion of crown defoliation.

Author(s)

Hervé Cochard, INRAE

Miquel De Cáceres Ainsa, CREAF

See Also

`hydraulics_conductancefunctions`

`hydraulics_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,
  initialValue = 0
)

hydraulics_taperFactorSavage(height)

hydraulics_terminalConduitRadius(height)

hydraulics_referenceConductivityHeightFactor(refheight, height)
```

```
hydraulics_maximumStemHydraulicConductance(
  xylemConductivity,
  refheight,
  Al2As,
  height,
  taper = FALSE
)

hydraulics_rootxylemConductanceProportions(L, V)
```

Arguments

`krhizomax`	Maximum rhizosphere 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).
`kstemmax`	Maximum stem 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).
`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.
`PLCstem`	Percent loss of conductance (in %) in the stem.
`psiLeaf`	Water potential (in MPa) in the leaf.
`n, alpha`	Parameters of the Van Genuchten function (rhizosphere 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).
`psiStep`	Water potential precision (in MPa).
`averageResistancePercent`	Average (across water potential values) resistance percent of the rhizosphere, with respect to total resistance (rhizosphere + root xylem + stem xylem).
`initialValue`	Initial value of rhizosphere conductance.
`height`	Plant height (in cm).
`refheight`	Reference plant height of measurement of xylem conductivity (in cm).
`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).
`taper`	A boolean flag to indicate correction by taper of xylem conduits (Christoffersen et al. 2017).
`L`	Vector with the length of coarse roots (mm) for each soil layer.
`V`	Vector with the proportion [0-1] of fine roots within each soil layer.

Details

Details of the hydraulic model are given in the medfate book

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, CREAF

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., and 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` `hydraulics_supplyfunctions` *Hydraulic supply functions*

Description

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

Usage

```
hydraulics_EXylem(
  psiPlant,
  psiUpstream,
  kxylemmax,
  c,
  d,
  allowNegativeFlux = TRUE,
  psiCav = 0
)

hydraulics_E2psiXylem(E, psiUpstream, kxylemmax, c, d, psiCav = 0)

hydraulics_E2psiXylemUp(E, psiDownstream, kxylemmax, c, d, psiCav = 0)

hydraulics_EVanGenuchten(psiRhizo, psiSoil, krhizomax, n, alpha, l = 0.5)

hydraulics_ECrit(psiUpstream, kxylemmax, c, d, pCrit = 0.001)

hydraulics_E2psiVanGenuchten(
  E,
  psiSoil,
  krhizomax,
  n,
  alpha,
  psiStep = -1e-04,
  psiMax = -10
)

hydraulics_E2psiTwoElements(
  E,
  psiSoil,
  krhizomax,
  kxylemmax,
  n,
  alpha,
  c,
  d,
  psiCav = 0,
  psiStep = -1e-04,
  psiMax = -10
)

hydraulics_E2psiBelowground(E, hydraulicNetwork, psiIni = as.numeric(c(0)))

hydraulics_E2psiAboveground(E, psiRootCrown, hydraulicNetwork)

hydraulics_E2psiNetwork(E, hydraulicNetwork, psiIni = as.numeric(c(0)))

hydraulics_supplyFunctionOneXylem(
  psiSoil,
  v,
  kstemmax,
  stemc,
  stemd,
  psiCav = 0,
  maxNsteps = 200L,
  dE = 0.01
)

hydraulics_supplyFunctionTwoElements(
  Emax,
  psiSoil,
  krhizomax,
  kxylemmax,
  n,
  alpha,
  c,
  d,
  psiCav = 0,
  dE = 0.1,
  psiMax = -10
)

hydraulics_supplyFunctionThreeElements(
  Emax,
  psiSoil,
  krhizomax,
  kxylemmax,
  kleafmax,
  n,
  alpha,
  stemc,
  stemd,
  leafc,
  leafd,
  psiCav = 0,
  dE = 0.1,
  psiMax = -10
)

hydraulics_supplyFunctionBelowground(
  hydraulicNetwork,
  minFlow = 0,
  pCrit = 0.001
)

hydraulics_supplyFunctionAboveground(
  Erootcrown,
  psiRootCrown,
  hydraulicNetwork
)

hydraulics_supplyFunctionNetwork(hydraulicNetwork, minFlow = 0, pCrit = 0.001)

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
)

hydraulics_initSperryNetworks(x)

hydraulics_supplyFunctionPlot(
  x,
  draw = TRUE,
  type = "E",
  speciesNames = FALSE,
  ylim = NULL
)
```

Arguments

`psiPlant`	Plant water potential (in MPa).
`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.
`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).
`allowNegativeFlux`	A boolean to indicate whether negative flux (i.e. from plant to soil) is allowed.
`psiCav`	Minimum water potential (in MPa) experienced (for irreversible cavitation).
`E`	Flow per surface unit.
`psiDownstream`	Water potential upstream (in MPa).
`psiRhizo`	Soil water potential (in MPa) in the rhizosphere (root surface).
`psiSoil`	Soil water potential (in MPa). A scalar or a vector depending on the function.
`krhizomax`	Maximum rhizosphere hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
`n, alpha, l`	Parameters of the Van Genuchten function (rhizosphere vulnerability curve).
`pCrit`	Critical water potential (in MPa).
`psiStep`	Water potential precision (in MPa).
`psiMax`	Minimum (maximum in absolute value) water potential to be considered (in MPa).
`hydraulicNetwork`	List with the hydraulic characteristics of nodes in the hydraulic network.
`psiIni`	Vector of initial water potential values (in MPa).
`psiRootCrown`	Soil water potential (in MPa) at the root crown.
`v`	Proportion of fine roots within each soil layer.
`kstemmax`	Maximum stem xylem hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
`stemc, stemd`	Parameters of the Weibull function for stems (stem xylem vulnerability curve).
`maxNsteps`	Maximum number of steps in the construction of supply functions.
`dE`	Increment of flow per surface unit.
`Emax`	Maximum flow per surface unit.
`kleafmax`	Maximum leaf hydraulic conductance (defined as flow per leaf surface unit and per pressure drop).
`leafc, leafd`	Parameters of the Weibull function for leaves (leaf vulnerability curve).
`minFlow`	Minimum flow in supply function.
`Erootcrown`	Flow per surface unit at the root crown.
`x`	An object of class `spwbInput`.
`draw`	A flag to indicate whether the supply function should be drawn or just returned.
`type`	Plot type for `hydraulics_supplyFunctionPlot`, either `"E"`, `"ERhizo"`, `"StemPsi"`, `"RootPsi"` or `"dEdP"`).
`speciesNames`	A flag to indicate the use of species names instead of cohort names in plots.
`ylim`	Graphical parameter to override function defaults.

Details

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`: If `draw = FALSE` a list with the result of calling `hydraulics_supplyFunctionNet` 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, CREAF

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 and 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))
```

`hydrology_infiltration` *Soil infiltration*

Description

Soil infiltration functions:

- Function `hydrology_infiltrationBoughton` calculates the amount of water that infiltrates into the topsoil, according to the USDA SCS curve number method (Boughton 1989).
- Function `hydrology_infiltrationGreenAmpt` calculates the amount of water that infiltrates into the topsoil, according to the model by Green & Ampt (1911).
- Function `hydrology_infiltrationAmount` uses either Green & Ampt (1911) or Boughton (1989) to estimate infiltration.
- Function `hydrology_infiltrationRepartition` distributes infiltration among soil layers depending on macroporosity.

Usage

```
hydrology_infiltrationBoughton(input, Ssoil)

hydrology_infiltrationGreenAmpt(t, psi_w, Ksat, theta_sat, theta_dry)

hydrology_infiltrationRepartition(I, widths, macro, a = -0.005, b = 3)

hydrology_infiltrationAmount(
  rainfallInput,
  rainfallIntensity,
  soil,
  soilFunctions,
  model = "GreenAmpt1911",
  K_correction = 1
)
```

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).
`t`	Time of the infiltration event
`psi_w`	Matric potential at the wetting front
`Ksat`	hydraulic conductivity at saturation
`theta_sat`	volumetric content at saturation
`theta_dry`	volumetric content at the dry side of the wetting front
`I`	Soil infiltration (in mm of water).
`widths`	Width of soil layers (in mm).
`macro`	Macroporosity of soil layers (in %).
`a, b`	Parameters of the extinction function used for water infiltration.
`rainfallInput`	Water from the rainfall event reaching the soil surface (mm)
`rainfallIntensity`	rainfall intensity rate (mm/h)
`soil`	A list containing the description of the soil (see `soil`).
`soilFunctions`	Soil water retention curve and conductivity functions, either 'SX' (for Saxton) or 'VG' (for Van Genuchten).
`model`	Infiltration model, either "GreenAmpt1911" or "Boughton1989"
`K_correction`	Correction for saturated conductivity, to account for increased infiltration due to macropore presence

Details

When using function `hydrology_infiltrationGreenAmpt`, the units of `Ksat`, `t` and `psi_wat` have to be in the same system (e.g. cm/h, h and cm).

Value

Functions `hydrology_infiltrationBoughton`, `hydrology_infiltrationGreenAmpt` and `hydrology_infiltrationAmou`
return the daily amount of water that infiltrates into the soil (in mm of water).

Function `hydrology_infiltrationRepartition` returns the amount of infiltrated water that reaches each soil layer.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

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

Green, W.H. and Ampt, G.A. (1911) Studies on Soil Physics, 1: The Flow of Air and Water through Soils. The Journal of Agricultural Science, 4, 1-24.

See Also

`spwb`, `hydrology_waterInputs`

`hydrology_rainfallIntensity`
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_rainfallIntensity(month, prec, rainfallIntensityPerMonth)

hydrology_rainInterception(Rainfall, Cm, p, ER = 0.05, model = "Gash1995")

hydrology_interceptionPlot(
  x,
  SpParams,
  ER = 0.05,
  gdd = NA,
  throughfall = FALSE,
  model = "Gash1995"
)
```

Arguments

`month`	Month of the year (from 1 to 12).
`prec`	Precipitation for a given day (mm).
`rainfallIntensityPerMonth`	A vector with twelve positions with average intensity of rainfall (in mm/h) for each month.
`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.
`model`	Rainfall interception model (either `"Gash1995"` or `"Liu2001"`).
`x`	An object of class `spwbInput`.
`SpParams`	A data frame with species parameters (see `SpParamsMED` and `SpParamsMED`).
`gdd`	Growth degree days (in Celsius).
`throughfall`	Boolean flag to plot relative throughfall instead of percentage of intercepted rainfall.

Details

Function `hydrology_rainInterception` can accept either vectors or scalars as parameters `Cm`, `p` and `ER`. If they are supplied as vectors they should be of the same length as `Rainfall`.

Function `hydrology_rainfallIntensity` estimates the rainfall intensity (mm/h) for input values of rainfall and seasonal variation in rainfall intensity (mm/h).

Value

Function `hydrology_rainInterception` returns a vector of the same length as `Rainfall` containing intercepted rain values.

Function `hydrology_rainfallIntensity` returns a scalar with the rainfall intensity.

Author(s)

Miquel De Cáceres Ainsa, CREAF

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

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

#Default species parameterization
data(SpParamsMED)

#Draw rainfall interception for two values of the E/R ratio
hydrology_interceptionPlot(exampleforest, SpParamsMED, ER = c(0.05, 0.2))
```

---

hydrology_snowMelt *Water vertical inputs*

Description

High-level functions to define water inputs into the soil of a stand:

- Function hydrology_waterInputs performs canopy water interception and snow accumulation/melt.
- Function hydrology_snowMelt estimates snow melt using a simple energy balance, according to Kergoat (1998).

Usage

```
hydrology_snowMelt(tday, rad, LgroundSWR, elevation)

hydrology_waterInputs(
  x,
  prec,
  rainfallIntensity,
  pet,
  tday,
  rad,
  elevation,
  Cm,
  LgroundPAR,
  LgroundSWR,
  modifyInput = TRUE
)
```

Arguments

`tday`	Average day temperature (ºC).
`rad`	Solar radiation (in MJ/m2/day).
`LgroundSWR`	Percentage of short-wave radiation (SWR) reaching the ground.
`elevation`	Altitude above sea level (m).
`x`	An object of class `spwbInput` or `growthInput`.
`prec`	Precipitation for the given day (mm)
`rainfallIntensity`	Rainfall intensity rate (mm/h).
`pet`	Potential evapotranspiration for the given day (mm)
`Cm`	Canopy water storage capacity.
`LgroundPAR`	Percentage of photosynthetically-active radiation (PAR) reaching the ground.
`modifyInput`	Boolean flag to indicate that the input `x` object should be modified during the simulation.

Details

The function simulates different vertical hydrological processes, which are described separately in other functions. If `modifyInput = TRUE` the function will modify the `x` object (including both soil moisture and the snowpack on its surface) as a result of simulating hydrological processes.

Value

Function `hydrology_waterInputs` returns a named vector with the following elements, all in mm:

`Rain`	Precipitation as rainfall.
`Snow`	Precipitation as snow.
`Interception`	Rainfall water intercepted by the canopy and evaporated.
`Snowmelt`	Snow melted during the day, and added to the water infiltrated.
`NetRain`	Rainfall reaching the ground.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Kergoat L. (1998). A model for hydrological equilibrium of leaf area index on a global scale. Journal of Hydrology 212–213: 268–286.

See Also

`spwb_day`, `hydrology_rainInterception`, `hydrology_soilEvaporation`

hydrology_soilEvaporationAmount
Bare soil evaporation and herbaceous transpiration

Description

Functions:

- Function `hydrology_soilEvaporationAmount` calculates the amount of evaporation from bare soil, following Ritchie (1972).
- Function `hydrology_soilEvaporation` calculates the amount of evaporation from bare soil and distributes it among soil layers.
- Function `hydrology_herbaceousTranspiration` calculates the amount of transpiration due to herbaceous plants.

Usage

```
hydrology_soilEvaporationAmount(DEF, PETs, Gsoil)

hydrology_soilEvaporation(
  soil,
  snowpack,
  soilFunctions,
  pet,
  LgroundSWR,
  modifySoil = TRUE
)

hydrology_herbaceousTranspiration(
  pet,
  LherbSWR,
  herbLAI,
  soil,
  soilFunctions,
  modifySoil = TRUE
)
```

Arguments

`DEF`	Water deficit in the (topsoil) layer.
`PETs`	Potential evapotranspiration at the soil surface.
`Gsoil`	Gamma parameter (maximum daily evaporation).
`soil`	An object of class `soil`.
`snowpack`	The amount of snow (in water equivalents, mm) in the snow pack.

`soilFunctions`	Soil water retention curve and conductivity functions, either 'SX' (for Saxton) or 'VG' (for Van Genuchten).
`pet`	Potential evapotranspiration for a given day (mm)
`LgroundSWR`	Percentage of short-wave radiation (SWR) reaching the ground.
`modifySoil`	Boolean flag to indicate that the input `soil` object should be modified during the simulation.
`LherbSWR`	Percentage of short-wave radiation (SWR) reaching the herbaceous layer.
`herbLAI`	Leaf area index of the herbaceous layer.

Value

Function `hydrology_soilEvaporationAmount` returns the amount of water evaporated from the soil.

Function `hydrology_soilEvaporation` returns a vector of water evaporated from each soil layer.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

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

See Also

`spwb`, `hydrology_waterInputs`, `hydrology_infiltration`

`hydrology_soilWaterBalance` *Soil water balance*

Description

Function `hydrology_soilWaterBalance` estimates water balance of soil layers given water inputs/outputs, including the simulation of water movement within the soil.

Usage

```
hydrology_soilWaterBalance(
  soil,
  soilFunctions,
  rainfallInput,
  rainfallIntensity,
  snowmelt,
  sourceSink,
  runon = 0,
  lateralFlows = NULL,
  waterTableDepth = NA_real_,
  infiltrationMode = "GreenAmpt1911",
  infiltrationCorrection = 5,
  soilDomains = "single",
  nsteps = 24L,
  max_nsubsteps = 3600L,
  modifySoil = TRUE
)
```

Arguments

`soil`	Object of class `soil`.
`soilFunctions`	Soil water retention curve and conductivity functions, either 'SX' (for Saxton) or 'VG' (for Van Genuchten).
`rainfallInput`	Amount of water from rainfall event (after excluding interception), in mm.
`rainfallIntensity`	Rainfall intensity, in mm/h.
`snowmelt`	Amount of water originated from snow melt, in mm.
`sourceSink`	Local source/sink term for each soil layer (from soil evaporation or plant transpiration/redistribution) as mm/day.
`runon`	Surface water amount running on the target area from upslope (in mm).
`lateralFlows`	Lateral source/sink terms for each soil layer (interflow/to from adjacent locations) as mm/day.
`waterTableDepth`	Water table depth (in mm). When not missing, capillarity rise will be allowed if lower than total soil depth.
`infiltrationMode`	Infiltration model, either "GreenAmpt1911" or "Boughton1989"
`infiltrationCorrection`	Correction for saturated conductivity, to account for increased infiltration due to macropore presence
`soilDomains`	Either "single" (for single-domain) or "dual" (for dual-permeability).
`nsteps`	Number of time steps per day
`max_nsubsteps`	Maximum number of substeps per time step
`modifySoil`	Boolean flag to indicate that the input `soil` object should be modified during the simulation.

Details

The single-domain model simulates water flows by solving Richards's equation using the predictor-corrector method, as described in Bonan et al. (2019).

The dual-permeability model is an implementation of the model MACRO 5.0 (Jarvis et al. 1991; Larsbo et al. 2005).

Value

Returns a named vector with different elements, depending on `soilDomains`. If `soilDomains == "single"`:

- `Snowmelt`: Snowmelt input (mm).
- `Source/sinks`: Sum of source/sink input across layers (mm).
- `Infiltration`: Water infiltrated into the soil (mm).
- `InfiltrationExcess`: Excess infiltration in the topmost layer (mm) leading to an increase in runoff.
- `SaturationExcess`: Excess saturation in the topmost layer (mm) leading to an increase in runoff.
- `Runoff`: Surface runoff generated by saturation excess or infiltration excess (mm).
- `DeepDrainage`: Water draining from the bottom layer (mm). This quantity is corrected to close the water balance.
- `CapillarityRise`: Water entering the soil via capillarity rise (mm) from the water table, if `waterTableDepth` is supplied.
- `Correction`: Amount of water (mm) added to deep drainage to correct the water balance.
- `VolumeChange`: Change in soil water volume (mm).
- `Substep`: Time step of the moisture solving (seconds).

If `soilDomains == "dual"` the named vector contains the following additional elements:

- `Lateral flows`: Sum of water circulating between micropores and macropores, positive when filling micropores (mm).
- `InfiltrationMatrix`: Water infiltrated into the soil matrix (mm).
- `InfiltrationMacropores`: Water infiltrated into the soil macropore domain (mm).
- `InfiltrationExcessMatrix/InfiltrationExcessMacropores`: Excess infiltration in the topmost layer (mm) leading to an increase in runoff.
- `SaturationExcessMatrix/SaturationExcessMacropores`: Excess saturation in the topmost layer (mm) leading to an increase in runoff.
- `DrainageMatrix`: Water draining from the bottom layer of the matrix domain (mm). This quantity is corrected to close water balance in the micropore domain.
- `DrainageMacropores`: Water draining from the bottom layer of the macropore domain (mm). This quantity is corrected to close the water balance in the macropore domain.
- `CorrectionMatrix`: Amount of water (mm) added to deep drainage of soil matrix to correct the water balance.
- `CorrectionMacropores`: Amount of water (mm) added to deep drainage of macropores to correct the water balance.
- `MatrixVolumeChange`: Change in soil water volume in the soil matrix domain (mm).
- `MacroporeVolumeChange`: Change in soil water volume in the macropore domain (mm).

Author(s)

Miquel De Cáceres Ainsa, CREAF

María González Sanchís, UPV-CTFC

References

Bonan, G. (2019). Climate change and terrestrial ecosystem modeling. Cambridge University Press, Cambridge, UK.

Jarvis, N.J., Jansson, P-E., Dik, P.E. & Messing, I. (1991). Modelling water and solute transport in macroporous soil. I. Model description and sensitivity analysis. Journal of Soil Science, 42, 59–70.

Larsbo, M., Roulier, S., Stenemo, F., Kasteel, R. & Jarvis, N. (2005). An Improved Dual-Permeability Model of Water Flow and Solute Transport in the Vadose Zone. Vadose Zone Journal, 4, 398–406.

See Also

`spwb`, `hydrology_waterInputs`, `hydrology_infiltration`

Examples

```
# Define soil parameters
spar <- defaultSoilParams(4)

# Initializes soil hydraulic parameters
examplesoil <- soil(spar)

# Water balance in a single-domain simulation (Richards equation)
hydrology_soilWaterBalance(examplesoil, "VG", 10, 5, 0, c(-1,-1,-1,-1),
                           soilDomains = "single", modifySoil = FALSE)

# Water balance in a dual-permeability model (MACRO)
hydrology_soilWaterBalance(examplesoil, "VG", 10, 5, 0, c(-1,-1,-1,-1),
                           soilDomains = "dual", modifySoil = FALSE)
```

`light_advanced` *Advanced radiation transfer functions*

Description

Functions `light_layerDirectIrradianceFraction` and `light_layerDiffuseIrradianceFraction` calculate the fraction of above-canopy direct and diffuse radiation 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 absorbed by cohort and vegetation layers, while differentiating between sunlit and shade leaves.

Usage

```
light_leafAngleCDF(leafAngle, p, q)

light_leafAngleBetaParameters(leafAngle, leafAngleSD)

light_directionalExtinctionCoefficient(p, q, solarElevation)

light_layerDirectIrradianceFraction(
  LAIme,
  LAImd,
  LAImx,
  kb,
  ClumpingIndex,
  alpha,
  gamma,
  trunkExtinctionFraction = 0.1
)

light_layerDiffuseIrradianceFraction(
  LAIme,
  LAImd,
  LAImx,
  K,
  ClumpingIndex,
  ZF,
  alpha,
  gamma,
  trunkExtinctionFraction = 0.1
)

light_cohortSunlitShadeAbsorbedRadiation(
  Ib0,
  Id0,
  LAIme,
  LAImd,
  LAImx,
  kb,
  K,
  ClumpingIndex,
  ZF,
  alpha,
  gamma,
  trunkExtinctionFraction = 0.1
)

light_layerSunlitFraction(LAIme, LAImd, kb, ClumpingIndex)

light_instantaneousLightExtinctionAbsortion(
  LAIme,
  LAImd,
  LAImx,
  p,
  q,
  ClumpingIndex,
  alphaSWR,
  gammaSWR,
  ddd,
  ntimesteps = 24L,
  trunkExtinctionFraction = 0.1
)

light_longwaveRadiationSHAW(
  LAIme,
  LAImd,
  LAImx,
  LWRatm,
  Tsoil,
  Tair,
  trunkExtinctionFraction = 0.1
)
```

Arguments

`leafAngle`	Average leaf inclination angle (in radians).
`p, q`	Parameters of the beta distribution for leaf angles
`leafAngleSD`	Standard deviation of leaf inclination angle (in radians).
`solarElevation`	Solar elevation (in radians).
`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).
`kb`	A vector of direct light extinction coefficients.
`ClumpingIndex`	The extent to which foliage has a nonrandom spatial distribution.
`alpha`	A vector of leaf absorbance by species.
`gamma`	A vector of leaf reflectance values.
`trunkExtinctionFraction`	Fraction of extinction due to trunks (for winter deciduous forests).
`K`	A vector of light extinction coefficients.
`ZF`	Fraction of sky angles.
`Ib0`	Above-canopy direct incident radiation.

`Id0`	Above-canopy diffuse incident radiation.
`alphaSWR`	A vecfor of hort-wave absorbance 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).
`ntimesteps`	Number of subdaily time steps.
`LWRatm`	Atmospheric downward long-wave radiation (W/m2).
`Tsoil`	Soil temperature (Celsius).
`Tair`	Canopy layer air temperature vector (Celsius).

Details

Functions for short-wave radiation are adapted from Anten & Bastiaans (2016), whereas long-wave radiation balance follows Flerchinger et al. (2009). Vegetation layers are assumed to be ordered from bottom to top.

Value

Functions `light_layerDirectIrradianceFraction`, `light_layerDiffuseIrradianceFraction` 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, CREAF

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.

Flerchinger, G. N., Xiao, W., Sauer, T. J., Yu, Q. 2009. Simulation of within-canopy radiation exchange. NJAS - Wageningen Journal of Life Sciences 57 (1): 5–15. https://doi.org/10.1016/j.njas.2009.07.004.

See Also

`spwb`, `light_basic`

Examples

```
solarElevation <- 0.67 # in radians
SWR_direct <- 1100
SWR_diffuse <- 300
PAR_direct <- 550
PAR_diffuse <- 150
```

```
LAI <- 2
nlayer <- 10
LAIlayerlive <- matrix(rep(LAI/nlayer,nlayer),nlayer,1)
LAIlayerdead <- matrix(0,nlayer,1)
meanLeafAngle <- 60 # in degrees
sdLeafAngle <- 20

beta <- light_leafAngleBetaParameters(meanLeafAngle*(pi/180), sdLeafAngle*(pi/180))

## Extinction coefficients
kb <- light_directionalExtinctionCoefficient(beta["p"], beta["q"], solarElevation)
kd_PAR <- 0.5
kd_SWR <- kd_PAR/1.35
```

light_basic *Radiation extinction functions used in basic transpiration sub-model*

Description

Radiation extinction functions used in basic transpiration sub-model

Usage

```
light_PARcohort(x, SpParams, gdd = NA_real_)

light_PARground(x, SpParams, gdd = NA_real_)

light_SWRground(x, SpParams, gdd = NA_real_)

light_cohortAbsorbedSWRFraction(z, x, SpParams, gdd = NA_real_)
```

Arguments

x	An object of class forest
SpParams	A data frame with species parameters (see SpParamsMED).
gdd	Growth degree days.
z	A numeric vector with height values.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

spwb, light_advanced

`modelInput` *Input for simulation models*

Description

Functions `spwbInput()` and `growthInput()` take an object of class `forest` and a soil data input to create input objects for simulation functions `spwb` (or `pwb`) and `growth`, respectively.

Usage

```
spwbInput(x, soil, SpParams, control)

growthInput(x, soil, SpParams, control)
```

Arguments

`x`	An object of class `forest`.
`soil`	An object of class `data.frame` or `soil`, containing soil parameters per soil layer.
`SpParams`	A data frame with species parameters (see `SpParamsDefinition` and `SpParamsMED`).
`control`	A list with default control parameters (see `defaultControl`).

Details

Functions `spwbInput()` and `growthInput()` initialize inputs differently depending on control parameters.

IMPORTANT NOTE: Older function names `forest2spwbInput` and `forest2growthInput` are now deprecated, but they can still be used for back-compatibility.

Value

Function `spwbInput()` returns 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`).
- `soil`: A data frame with initialized soil parameters (see `soil`).
- `snowpack`: The amount of snow (in mm) in the snow pack over the soil.
- `canopy`: A list of stand-level state variables.
- `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 data frame with columns `Z50`, `Z95`. If `control$transpirationMode = "Sperry"` additional columns are `fineRootBiomass` and `coarseRootSoilVolume`.
- `belowLayers`: A list. If `control$transpirationMode = "Granier"` it contains elements:
  - `V`: A matrix with the proportion of fine roots of each cohort (in rows) in each soil layer (in columns).

  - `L`: A matrix with the length of coarse roots of each cohort (in rows) in each soil layer (in columns).
  - `Wpool`: A matrix with the soil moisture relative to field capacity around the rhizosphere of each cohort (in rows) in each soil layer (in columns).

  If `control$transpirationMode = "Sperry"` or `control$transpirationMode = "Sureau"` 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).
  - `RhizoPsi`: A matrix with the water potential around the rhizosphere of each cohort (in rows) in each soil layer (in columns).

- `paramsPhenology`: A data frame with leaf phenology parameters:
  - `PhenologyType`: Leaf phenology type.
  - `LeafDuration`: Leaf duration (in years).
  - `Sgdd`: Degree days needed for leaf budburst (for winter decideous species).
  - `Tbgdd`: Base temperature for the calculation of degree days to leaf budburst.
  - `Ssen`: Degree days corresponding to leaf senescence.
  - `Phsen`: Photoperiod corresponding to start counting senescence degree-days.
  - `Tbsen`: Base temperature for the calculation of degree days to leaf senescence.
- `paramsAnatomy`: A data frame with plant anatomy parameters for each cohort:
  - `Hmax`: Maximum plant height (cm).
  - `Hmed`: Median plant height (cm).
  - `Al2As`: Leaf area to sapwood area ratio (in m2·m-2).
  - `Ar2Al`: Fine root area to leaf area ratio (in m2·m-2).
  - `SLA`: Specific leaf area (mm2/mg = m2/kg).
  - `LeafWidth`: Leaf width (in cm).
  - `LeafDensity`: Density of leaf tissue (dry weight over volume).
  - `WoodDensity`: Density of wood tissue (dry weight over volume).
  - `FineRootDensity`: Density of fine root tissue (dry weight over volume).
  - `SRL`: Specific Root length (cm·g-1).
  - `RLD`: Root length density (cm·cm-3).
  - `r635`: Ratio between the weight of leaves plus branches and the weight of leaves alone for branches of 6.35 mm.
- `paramsInterception`: A data frame with rain interception and light extinction parameters for each cohort:
  - `kPAR`: PAR extinction coefficient.
  - `g`: Canopy water retention capacity per LAI unit (mm/LAI).

  If `control$transpirationMode = "Sperry"` or `control$transpirationMode = "Sureau"` additional columns are:

  - `gammaSWR`: Reflectance (albedo) coefficient for SWR .
  - `alphaSWR`: Absorbance coefficient for SWR .

- `paramsTranspiration`: A data frame with parameters for transpiration and photosynthesis. If `control$transpirationMode = "Granier"`, columns are:
  - `Gswmin`: Minimum stomatal conductance to water vapor (in mol H2O·m-2·s-1).
  - `Tmax_LAI`: Coefficient relating LAI with the ratio of maximum transpiration over potential evapotranspiration.
  - `Tmax_LAIsq`: Coefficient relating squared LAI with the ratio of maximum transpiration over potential evapotranspiration.
  - `Psi_Extract`: Water potential corresponding to 50% relative transpiration (in MPa).
  - `Exp_Extract`: Parameter of the Weibull function regulating transpiration reduction.
  - `VCstem_c`, `VCstem_d`: Parameters of the stem xylem vulnerability curve.
  - `WUE`: Daily water use efficiency (gross photosynthesis over transpiration) under no light, water or CO2 limitations and VPD = 1kPa (g C/mm water).
  - `WUE_par`: Coefficient regulating the influence of % PAR on gross photosynthesis.
  - `WUE_par`: Coefficient regulating the influence of atmospheric CO2 concentration on gross photosynthesis.
  - `WUE_par`: Coefficient regulating the influence of vapor pressure deficit (VPD) on gross photosynthesis.

  If `control$transpirationMode = "Sperry"` columns are:
  - `Gswmin`: Minimum stomatal conductance to water vapor (in mol H2O·m-2·s-1).
  - `Gswmax`: 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).
  - `Kmax_stemxylem`: Sapwood-specific hydraulic conductivity of stem xylem (in kg H2O·s-1·m-2).
  - `Kmax_rootxylem`: Sapwood-specific hydraulic conductivity of root xylem (in kg H2O·s-1·m-2).
  - `VCleaf_kmax`: Maximum leaf hydraulic conductance.
  - `VCleaf_c`, `VCleaf_d`: Parameters of the leaf vulnerability curve.
  - `VCstem_kmax`: Maximum stem xylem conductance.
  - `VCstem_c`, `VCstem_d`: Parameters of the stem xylem vulnerability curve.
  - `VCroot_c`, `VCroot_d`: Parameters of the root xylem vulnerability curve.
  - `Plant_kmax`: Maximum whole-plant conductance.

  If `control$transpirationMode = "Sureau"` columns are:
  - `Gswmin`: Minimum stomatal conductance to water vapor (in mol H2O·m-2·s-1).
  - `Gswmax`: 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).
  - `Kmax_stemxylem`: Sapwood-specific hydraulic conductivity of stem xylem (in kg H2O·s-1·m-2).
  - `Kmax_rootxylem`: Sapwood-specific hydraulic conductivity of root xylem (in kg H2O·s-1·m-2).
  - `VCleaf_kmax`: Maximum leaf hydraulic conductance.
  - `VCleaf_c`, `VCleaf_d`: Parameters of the leaf vulnerability curve.

  - VCstem_kmax: Maximum stem xylem conductance.
  - VCstem_c, VCstem_d: Parameters of the stem xylem vulnerability curve.
  - VCroot_c, VCroot_d: Parameters of the root xylem vulnerability curve.
  - Plant_kmax: Maximum whole-plant conductance.
- paramsWaterStorage: A data frame with plant water storage parameters for each cohort:
  - 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.
  - Vleaf: Storage water capacity in leaves, per leaf area (L/m2).
  - 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.
  - Vstem: Storage water capacity in sapwood, per leaf area (L/m2).
- internalPhenology and internalWater: data frames to store internal state variables.
- internalFCCS: A data frame with fuel characteristics, according to fuel_FCCS (only if fireHazardResults = TRUE, in the control list).

Function growthInput() returns 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 growth parameters for each cohort:
  - RERleaf: Maintenance respiration rates (at 20ºC) for leaves (in g gluc·g dry-1·day-1).
  - RERsapwood: Maintenance respiration rates (at 20ºC) for sapwood (in g gluc·g dry-1·day-1).
  - RERfineroot: Maintenance respiration rates (at 20ºC) for fine roots (in g gluc·g dry-1·day-1).
  - CCleaf: Leaf construction costs (in g gluc·g dry-1).
  - CCsapwood: Sapwood construction costs (in g gluc·g dry-1).
  - CCfineroot: Fine root construction costs (in g gluc·g dry-1).
  - RGRleafmax: Maximum leaf relative growth rate (in m2·cm-2·day-1).
  - RGRsapwoodmax: Maximum sapwood relative growth rate (in cm2·cm-2·day-1).
  - RGRfinerootmax: Maximum fine root relative growth rate (in g dry·g dry-1·day-1).
  - SRsapwood: Sapwood daily senescence rate (in day-1).
  - SRfineroot: Fine root daily senescence rate (in day-1).
  - RSSG: Minimum relative starch for sapwood growth (proportion).
  - fHDmin: Minimum value of the height-to-diameter ratio (dimensionless).
  - fHDmax: Maximum value of the height-to-diameter ratio (dimensionless).

  - `WoodC`: Wood carbon content per dry weight (g C /g dry).
- `paramsMortalityRegeneration`: A data frame with mortality/regeneration parameters for each cohort:
  - `MortalityBaselineRate`: Deterministic proportion or probability specifying the baseline reduction of cohort's density occurring in a year.
  - `SurvivalModelStep`: Time step in years of the empirical survival model depending on stand basal area (e.g. 10).
  - `SurvivalB0`: Intercept of the logistic baseline survival model depending on stand basal area.
  - `SurvivalB1`: Slope of the logistic baseline survival model depending on stand basal area.
  - `RecrTreeDensity`: Density of tree recruits from seeds.
  - `IngrowthTreeDensity`: Density of trees reaching ingrowth DBH.
  - `RecrTreeDBH`: DBH for tree recruits from seeds or resprouting (e.g. 1 cm).
  - `IngrowthTreeDBH`: Ingrowth DBH for trees (e.g. 7.5 cm).
- `paramsAllometry`: A data frame with allometric parameters for each cohort:
  - `Aash`: Regression coefficient relating the square of shrub height with shrub area.
  - `Absh`, `Bbsh`: Allometric coefficients relating phytovolume with dry weight of shrub individuals.
  - `Acr`, `B1cr`, `B2cr`, `B3cr`, `C1cr`, `C2cr`: Regression coefficients used to calculate crown ratio of trees.
  - `Acw`, `Bcw`: Regression coefficients used to calculated crown width of trees.
- `internalAllocation`: A data frame with internal allocation variables for each cohort:
  - `allocationTarget`: Value of the allocation target variable.
  - `leafAreaTarget`: Target leaf area (m2) per individual.
  - `sapwoodAreaTarget`: Target sapwood area (cm2) per individual.
  - `fineRootBiomassTarget`: Target fine root biomass (g dry) per individual.
  - `crownBudPercent`: Percentage of the crown with buds.
- `internalCarbon`: A data frame with the concentration (mol·gluc·l-1) of metabolic and storage carbon compartments for leaves and sapwood.
- `internalMortality`: A data frame to store the cumulative mortality (density for trees and cover for shrubs) predicted during the simulation, also distinguishing mortality due to starvation or dessication.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`resetInputs`, `spwb`, `soil`, `forest`, `SpParamsMED`, `defaultSoilParams`, `plant_ID`

Examples

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

# Example of aboveground parameters taken from a forest
# described using LAI and crown ratio
data(exampleforest2)

#Default species parameterization
data(SpParamsMED)


# Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

# Initialize control parameters using 'Granier' transpiration mode
control <- defaultControl("Granier")

# Prepare spwb input
spwbInput(exampleforest, examplesoil, SpParamsMED, control)

# Prepare input for 'Sperry' transpiration mode
control <- defaultControl("Sperry")
spwbInput(exampleforest,examplesoil,SpParamsMED, control)

# Prepare input for 'Sureau' transpiration mode
control <- defaultControl("Sureau")
spwbInput(exampleforest,examplesoil,SpParamsMED, control)

# Example of initialization from a forest
# described using LAI and crown ratio
control <- defaultControl("Granier")
spwbInput(exampleforest2, examplesoil, SpParamsMED, control)
```

`modifyParams` *Modify parameters*

Description

Routines to modify species parameter table or model input objects

Usage

```
modifySpParams(SpParams, customParams, subsetSpecies = TRUE)

modifyCohortParams(x, customParams, verbose = TRUE)

modifyInputParams(x, customParams, verbose = TRUE)
```

Arguments

`SpParams`	A species parameter data frame, typically `SpParamsMED`.
`customParams`	A data frame or a named vector with new parameter values (see details).
`subsetSpecies`	A logical flag to indicate that the output data frame should include only those species mentioned in `customParams`.
`x`	A model input object of class `spwbInput` or `growthInput`.
`verbose`	A logical flag to indicate that messages should be printed on the console.

Details

When calling function `modifySpParams`, `customParams` should be a data frame with as many rows as species and as many columns as parameters to modify, plus a column called 'SpIndex' or 'Species' to match species between the two tables.

When calling `modifyCohortParams`, `customParams` can be a data frame with as many rows as cohorts and as many columns as parameters to modify, plus a column called 'Cohort' which will be matched with the cohort names given by `spwbInput` or `growthInput`. Alternatively, `customParams` can be a named list or named numeric vector as for `modifyInputParams`.

When calling `modifyInputParams`, `customParams` must be either a named list or a named numeric vector. Cohort parameters are specified using the syntax "<cohortName>/<paramName>" for names (e.g. "T2_176/Z50" to modify parameter 'Z50' of cohort 'T2_176'). Soil layer parameters are specified using the syntax "<paramName>@#layer" for names, where #layer is the layer index (e.g. "rfc@1" will modify the rock fragment content of soil layer 1). Control parameters are specified using either "<paramName>" (e.g "phloemConductanceFactor") or "<paramName>$<subParamName>" (e.g "maximumRelativeGrowthRates$leaf"). It may seem unnecessary to modify soil or control parameters via a function, but `modifyInputParams` is called from optimization functions (see `optimization`).

Value

Function `modifySpParams` returns a modified species parameter data frame.

Functions `modifyCohortParams` and `modifyInputParams` return a modified `spwbInput` or `growthInput` object. Note that modifications may affect other parameters beyond those indicated in `customParams`, as a result of parameter dependencies (see examples).

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`spwbInput`, `SpParamsMED`, `optimization`

Examples

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

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

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x1 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

# Cohort name for Pinus halepensis
PH_coh <- paste0("T1_", SpParamsMED$SpIndex[SpParamsMED$Name=="Pinus halepensis"])
PH_coh

# Modify Z50 and Z95 of Pinus halepensis cohort
customParams <- c(200,2000)
names(customParams) <- paste0(PH_coh,c("/Z50", "/Z95"))
x1m <- modifyInputParams(x1, customParams)

# Inspect original and modified objects
x1$below
x1m$below

# Inspect dependencies: fine root distribution across soil layers
x1$belowLayers$V
x1m$belowLayers$V

# Modify rock fragment content and sand proportion of soil layer 1
x1s <- modifyInputParams(x1, c("rfc@1" = 5, "sand@1" = 10))

# Inspect original and modified soils
x1$soil
x1s$soil

# When modifying growth input objects dependencies increase
x1 <- growthInput(exampleforest,examplesoil, SpParamsMED, control)
customParams <- c(2000,2)
names(customParams) <- paste0(PH_coh,c("/Al2As", "/LAI_live"))
x1m <- modifyInputParams(x1, customParams)
```

moisture *Tissue moisture functions*

Description

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

Usage

```
moisture_sapwoodWaterCapacity(Al2As, height, V, L, wd)

moisture_leafWaterCapacity(SLA, ld)

moisture_turgorLossPoint(pi0, epsilon)

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)

plant_water(x)

moisture_pressureVolumeCurvePlot(
  x,
  segment = "stem",
  fraction = "all",
  psiVec = seq(-0.1, -8, by = -0.01),
  speciesNames = FALSE
)
```

Arguments

`Al2As`	Leaf area to sapwood area (in m2·m-2).
`height`	Plant height (in cm).
`V`	Vector with the proportion [0-1] of fine roots within each soil layer.
`L`	Vector with the length of coarse roots (mm) for each soil layer.
`wd`	Wood density (g·cm-3).
`SLA`	Specific leaf area (mm2·mg-1).
`ld`	Leaf tissue density (g·cm-3).
`pi0`	Full turgor osmotic potential (MPa).
`epsilon`	Bulk modulus of elasticity (MPa).
`psiSym, psiApo`	Symplastic or apoplastic water potential (MPa).
`RWC`	Relative water content [0-1].
`c, d`	Parameters of the xylem vulnerability curve.
`af`	Apoplastic fraction (proportion) in the segment (e.g. leaf or stem).
`x`	An object of class `spwbInput` or `growthInput`.

`segment`	Segment whose relative water content curve to plot, either `"stem"` or `"leaf"` (the latter only available if `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`).
`fraction`	Tissue fraction, either `"symplastic"`, `"apoplastic"` or `"all"`.
`psiVec`	Vector of water potential values to evaluate for the pressure-volume curve.
`speciesNames`	A flag to indicate the use of species names instead of cohort names in plots.

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_turgorLossPoint`: Water potential (in MPa) corresponding to turgor loss point.
- `moisture_segmentRWC`: Segment relative water content [0-1].
- `water_plant`: A vector of water content (mm) per plant cohort.

Author(s)

Miquel De Cáceres Ainsa, CREAF

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")
```

mortality_dailyProbability *Mortality*

Description

A simple sigmoid function to determine a daily mortality likelihood according to the value of a stress variable.

Usage

```
mortality_dailyProbability(stressValue, stressThreshold)
```

Arguments

`stressValue`	Current value of the stress variable (0 to 1, with higher values indicate stronger stress).
`stressThreshold`	Threshold to indicate 50% annual mortality probability.

Value

Returns a probability (between 0 and 1)

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`growth`, `regeneration`

optimization *Multiple model runs and function factories for optimization routines*

Description

Function factories to generate functions to be used in model calibration, uncertainty or sensitivity analysis.

Usage

```
multiple_runs(
  parMatrix,
  x,
  meteo,
  latitude,
  elevation = NA,
  slope = NA,
  aspect = NA,
  summary_function = NULL,
  args = NULL,
  verbose = TRUE
)

optimization_function(
  parNames,
  x,
  meteo,
  latitude,
  elevation = NA,
  slope = NA,
  aspect = NA,
  summary_function,
  args = NULL
)

optimization_evaluation_function(
  parNames,
  x,
  meteo,
  latitude,
  elevation = NA,
  slope = NA,
  aspect = NA,
  measuredData,
  type = "SWC",
  cohorts = NULL,
  temporalResolution = "day",
  metric = "loglikelihood"
)

optimization_multicohort_function(
  cohortParNames,
  cohortNames,
  x,
  meteo,
  latitude,
  otherParNames = NULL,
  elevation = NA,
  slope = NA,
  aspect = NA,
  summary_function,
  args = NULL
)

optimization_evaluation_multicohort_function(
  cohortParNames,
  cohortNames,
  x,
  meteo,
  latitude,
  otherParNames = NULL,
  elevation = NA,
  slope = NA,
  aspect = NA,
  measuredData,
  type = "SWC",
  cohorts = cohortNames,
  temporalResolution = "day",
  metric = "loglikelihood"
)
```

Arguments

`parMatrix`	A matrix of parameter values with runs in rows and parameters in columns. Column names should follow parameter modification naming rules (see examples and naming rules in `modifyInputParams`).
`x`	An object of class `spwbInput` or `growthInput`.
`meteo, latitude, elevation, slope, aspect`	Additional parameters to simulation functions `spwb` or `growth`.
`summary_function`	A function whose input is the result of `spwb` or `growth`. The function must return a numeric scalar in the case of `optimization_function`, but is not restricted in the case of `multiple_runs`.
`args`	A list of additional arguments of `optimization_function`.
`verbose`	A flag to indicate extra console output.
`parNames`	A string vector of parameter names (see examples and naming rules in `modifyInputParams`).
`measuredData`	A data frame with observed/measured values. Dates should be in row names, whereas columns should be named according to the type of output to be evaluated (see details).
`type`	A string with the kind of model output to be evaluated. Accepted values are `"SWC"` (soil moisture content), `"REW"` relative extractable water, `"ETR"` (total evapotranspiration), `"E"` (transpiration per leaf area), `"LFMC"` (live fuel moisture content) and `"WP"` (plant water potentials).

`cohorts`	A string or a vector of strings with the cohorts to be compared (e.g. "T1_68"). If several cohort names are provided, the function `optimization_cohorts_function` evaluates the performance for each one and provides the mean value. If `NULL` results for the first cohort will be evaluated.
`temporalResolution`	A string to indicate the temporal resolution of the model evaluation, which can be "day", "week", "month" or "year". Observed and modelled values are aggregated temporally (using either means or sums) before comparison.
`metric`	An evaluation metric (see `evaluation_metric`).
`cohortParNames`	A string vector of vegetation parameter names for cohorts (e.g. 'Z95' or 'psiExtract').
`cohortNames`	A string vector of cohort names. All cohorts will be given the same parameter values for each parameter in 'cohortParNames'.
`otherParNames`	A string vector of parameter names (see examples and naming rules in `modifyInputParams`) for non-vegetation parameters (i.e. control parameters and soil parameters).

Details

See `evaluation` for details regarding how to specify measured data.

Functions produced by these function factories should be useful for sensitivity analyses using package 'sensitivity'.

Parameter naming (i.e. `parNames`) should follow the rules specified in section details of `modifyInputParams`. The exception to the naming rules applies when multiple cohorts are to be modified to the same values with functions `optimization_multicohort_function` and `optimization_evaluation_multicohort_function`. Then, only a vector of parameter names is supplied for `cohortParNames`.

Value

Function `multiple_runs` returns a list, whose elements are either the result of calling simulation models or the result of calling `summary_function` afterwards.

Function `optimization_function` returns a function whose parameters are parameter values and whose return is a prediction scalar (e.g. total transpiration).

Function `optimization_evaluation_function` returns a function whose parameters are parameter values and whose return is an evaluation metric (e.g. loglikelihood of the data observations given model predictions). If evaluation data contains information for different cohorts (e.g. plant water potentials or transpiration rates) then the evaluation is performed for each cohort and the metrics are averaged.

Function `optimization_multicohorts_function` returns a function whose parameters are parameter values and whose return is a prediction scalar (e.g. total transpiration). The difference with `optimization_function` is that multiple cohorts are set to the same parameter values.

Function `optimization_evaluation_multicohort_function` returns a function whose parameters are parameter values and whose return is an evaluation metric (e.g. loglikelihood of the data observations given model predictions). If evaluation data contains information for different cohorts (e.g. plant water potentials or transpiration rates) then the evaluation is performed for each cohort and the metrics are averaged. The difference with `optimization_evaluation_function` is that multiple cohorts are set to the same parameter values.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`evaluation_metric`, `modifyInputParams`, `spwb`, `growth`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x1 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

# Cohort name for Pinus halepensis
PH_coh <- paste0("T1_", SpParamsMED$SpIndex[SpParamsMED$Name=="Pinus halepensis"])
PH_coh

#Parameter names of interest
parNames <- c(paste0(PH_coh,"/Z50"), paste0(PH_coh,"/Z95"))

#Specify parameter matrix
parMatrix <- cbind(c(200,300), c(500,1000))
colnames(parMatrix) <- parNames

#Define a summary function as the total transpiration over the simulated period
sf<-function(x) {sum(x$WaterBalance$Transpiration, na.rm=TRUE)}

#Perform two runs and evaluate the summary function
multiple_runs(parMatrix,
              x1, examplemeteo, latitude = 42, elevation = 100,
              summary_function = sf)

#Load observed data (in this case the same simulation results with some added error)
# Generate a prediction function for total transpiration over the simulated period
# as a function of parameters "Z50" and "Z95" for Pinus halepensis cohort
of<-optimization_function(parNames = parNames,
                          x = x1,
                              meteo = examplemeteo,
                              latitude = 41.82592, elevation = 100,
                              summary_function = sf)

# Evaluate for the values of the parameter matrix
of(parMatrix[1, ])
of(parMatrix)


# Generate a loglikelihood function for soil water content
# as a function of parameters "Z50" and "Z95" for Pinus halepensis cohort
data(exampleobs)
oef<-optimization_evaluation_function(parNames = parNames,
                                      x = x1,
                              meteo = examplemeteo, latitude = 41.82592, elevation = 100,
                                      measuredData = exampleobs, type = "SWC",
                                      metric = "loglikelihood")

# Loglikelihood for the values of the parameter matrix
oef(parMatrix[1, ])
oef(parMatrix)
```

`Parameter means` *Parameter average values*

Description

Internal data set with parameter averages for taxonomic families. This is used by input initialization functions to provide suitable parameter values when missing from species parameter tables.

Format

Data frame `trait_family_means` has taxonomic families in rows and parameter names as columns.

Source

Same sources as `SpParamsMED`

See Also

`SpParamsMED`, `spwbInput`

Examples

```
medfate::trait_family_means
```

pheno_updateLeaves *Leaf phenology*

Description

Function pheno_leafDevelopmentStatus returns the expanded status (0 to 1) of leaves according to the growth degree days required to start bud burst and leaf unfolding, as dictated by a simple ecodormancy (one-phase) model (Chuine et al. 2013). Function pheno_leafSenescenceStatus returns the 0/1 senescence status of leaves according to the one-phase senescence model of Delpierre et al. (2009) on the basis of photoperiod and temperature. Function pheno_updateLeaves updates the status of expanded leaves and dead leaves of object x given the photoperiod, temperature and wind of a given day. It applies the development model for 1 < doy < 180 and the senescence model for 181 > doy > 365.

Usage

```
pheno_leafDevelopmentStatus(Sgdd, gdd, unfoldingDD = 300)

pheno_leafSenescenceStatus(Ssen, sen)

pheno_updatePhenology(x, doy, photoperiod, tmean)

pheno_updateLeaves(x, wind, fromGrowthModel)
```

Arguments

Sgdd	Degree days required for leaf budburst (in Celsius).
gdd	Cumulative degree days (in Celsius)
unfoldingDD	Degree-days for complete leaf unfolding after budburst has occurred.
Ssen	Threshold to start leaf senescence.
sen	Cumulative senescence variable.
x	An object of class spwbInput.
doy	Day of the year.
photoperiod	Day length (in hours).
tmean	Average day temperature (in Celsius).
wind	Average day wind speed (in m/s).
fromGrowthModel	Boolean flag to indicate that routine is called from growth simulation function.

Value

Function pheno_leafDevelopmentStatus returns a vector of values between 0 and 1, whereas function pheno_leafSenescenceStatus returns a vector of 0 (senescent) and 1 (expanded) values. The other two functions do not return any value (see note).

Note

Functions `pheno_updatePhenology` and `pheno_updateLeaves` modify the input object `x`. The first modifies phenological state and the second modifies the leaf area accordingly.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Chuine, I., De Cortazar-Atauri, I.G., Kramer, K., Hänninen, H., 2013. Plant development models. Phenology: An Integrative Environmental Science. Springer, pp. 275–293.

Delpierre N, Dufrêne E, Soudani K et al (2009) Modelling interannual and spatial variability of leaf senescence for three deciduous tree species in France. Agric For Meteorol 149:938–948. doi:10.1016/j.agrformet.2008.11.014

See Also

spwb, spwbInput

---

photo *Photosynthesis submodel functions*

Description

Set of functions used in the calculation of photosynthesis

Usage

```
photo_GammaTemp(Tleaf)

photo_KmTemp(Tleaf, Oi = 209)

photo_VmaxTemp(Vmax298, Tleaf)

photo_JmaxTemp(Jmax298, Tleaf)

photo_electronLimitedPhotosynthesis(Q, Ci, GT, Jmax)

photo_rubiscoLimitedPhotosynthesis(Ci, GT, Km, Vmax)

photo_photosynthesis(Q, Catm, Gc, Tleaf, Vmax298, Jmax298, verbose = FALSE)

photo_photosynthesisBaldocchi(
  Q,
  Catm,
  Tleaf,
  u,
  Vmax298,
  Jmax298,
  leafWidth,
  Gsw_AC_slope,
  Gsw_AC_intercept
)

photo_leafPhotosynthesisFunction(
  E,
  psiLeaf,
  Catm,
  Patm,
  Tair,
  vpa,
  u,
  absRad,
  Q,
  Vmax298,
  Jmax298,
  leafWidth = 1,
  refLeafArea = 1,
  verbose = FALSE
)

photo_leafPhotosynthesisFunction2(
  E,
  psiLeaf,
  Catm,
  Patm,
  Tair,
  vpa,
  u,
  SWRabs,
  LWRnet,
  Q,
  Vmax298,
  Jmax298,
  leafWidth = 1,
  refLeafArea = 1,
  verbose = FALSE
)

photo_sunshadePhotosynthesisFunction(
  E,
  psiLeaf,
  Catm,
  Patm,
  Tair,
  vpa,
  SLarea,
  SHarea,
  u,
  absRadSL,
  absRadSH,
  QSL,
  QSH,
  Vmax298SL,
  Vmax298SH,
  Jmax298SL,
  Jmax298SH,
  leafWidth = 1,
  verbose = FALSE
)

photo_multilayerPhotosynthesisFunction(
  E,
  psiLeaf,
  Catm,
  Patm,
  Tair,
  vpa,
  SLarea,
  SHarea,
  u,
  absRadSL,
  absRadSH,
  QSL,
  QSH,
  Vmax298,
  Jmax298,
  leafWidth = 1,
  verbose = FALSE
)
```

Arguments

`Tleaf`	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).
`verbose`	Boolean flag to indicate console output.
`u`	Wind speed above the leaf boundary (in m/s) (for each canopy layer in the case of `photo_multilayerPhotosynthesisFunction`).
`leafWidth`	Leaf width (in cm).
`Gsw_AC_slope`	Slope of the An/C vs Gsw relationship
`Gsw_AC_intercept`	Intercept of the An/C vs Gsw relationship
`E`	Transpiration flow rate per leaf area (mmol*s-1*m-2).
`psiLeaf`	Leaf water potential (MPa).
`Patm`	Atmospheric air pressure (in kPa).
`Tair`	Air temperature (in ºC).
`vpa`	Vapour pressure deficit (in kPa).
`absRad`	Absorbed long- and short-wave radiation (in W*m^-2).
`refLeafArea`	Leaf reference area.
`SWRabs`	Absorbed short-wave radiation (in W·m-2).
`LWRnet`	Net long-wave radiation balance (in W·m-2).
`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 the medfate book

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).
  - `LeafCi`: Internal CO2 concentration (micromol * mol-1).
  - `Gsw`: Leaf stomatal conductance to water vapor (mol * s-1 * m-2).
  - `GrossPhotosynthesis`: Gross photosynthesis (micromol*s-1*m-2).
  - `NetPhotosynthesis`: Net photosynthesis, after discounting autotrophic respiration (micromol*s-1*m-2).
- `photo_sunshadePhotosynthesisFunction`: Returns a data frame with the following columns:
  - `GrossPhotosynthesis`: Gross photosynthesis (micromol*s-1*m-2).
  - `NetPhotosynthesis`: Net photosynthesis, after discounting autotrophic respiration (micromol*s-1*m-2).
  - `LeafCiSL`: Sunlit leaf internal CO2 concentration (micromol * mol-1).
  - `LeafCiSH`: Shade leaf internal CO2 concentration (micromol * mol-1).
  - `LeafTempSL`: Sunlit leaf temperature (ºC).
  - `LeafTempSH`: Shade leaf temperature (ºC).
  - `LeafVPDSL`: Sunlit leaf vapor pressure deficit (kPa).
  - `LeafVPDSH`: Shade leaf vapor pressure deficit (kPa).
- `photo_multilayerPhotosynthesisFunction`: Return a data frame with the following columns:
  - `GrossPhotosynthesis`: 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, CREAF

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` *Woody plant cohort description functions*

Description

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

Usage

```
plant_ID(x, SpParams, treeOffset = 0L, shrubOffset = 0L)

plant_basalArea(x, SpParams)

plant_largerTreeBasalArea(x, SpParams, self_proportion = 0.5)

plant_cover(x, SpParams)

plant_species(x, SpParams)

plant_speciesName(x, SpParams)

plant_density(x, SpParams)

plant_height(x, SpParams)

plant_individualArea(x, SpParams)
```

```
plant_crownRatio(x, SpParams)

plant_crownBaseHeight(x, SpParams)

plant_crownLength(x, SpParams)

plant_foliarBiomass(x, SpParams, gdd = NA_real_)

plant_fuelLoading(x, SpParams, gdd = NA_real_, includeDead = TRUE)

plant_equilibriumLeafLitter(x, SpParams, AET = 800)

plant_equilibriumSmallBranchLitter(
  x,
  SpParams,
  smallBranchDecompositionRate = 0.81
)

plant_phytovolume(x, SpParams)

plant_LAI(x, SpParams, gdd = NA_real_, bounded = TRUE)

plant_characterParameter(x, SpParams, parName)

plant_parameter(x, SpParams, parName, fillMissing = TRUE, fillWithGenus = TRUE)
```

Arguments

`x`	An object of class `forest`.
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`treeOffset, shrubOffset`	Integers to offset cohort IDs.
`self_proportion`	Proportion of the target cohort included in the assessment
`gdd`	Growth degree days (to account for leaf phenology effects).
`includeDead`	A flag to indicate that standing dead fuels (dead branches) are included.
`AET`	Actual annual evapotranspiration (in mm).
`smallBranchDecompositionRate`	Decomposition rate of small branches.
`bounded`	A boolean flag to indicate that extreme values should be prevented (maximum tree LAI = 7 and maximum shrub LAI = 3)
`parName`	A string with a parameter name.
`fillMissing`	A boolean flag to try imputation on missing values.
`fillWithGenus`	A boolean flag to try imputation of missing values using genus values.

Value

A vector with values for each woody plant cohort 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_fuelLoading`: 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_individualArea`: Area (m2) occupied by a shrub individual.
- `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, CREAF

See Also

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

Examples

```
#Default species parameterization
data(SpParamsMED)

#Load example plot
data(exampleforest)

#A plant-level way to obtain stand basal area
sum(plant_basalArea(exampleforest, SpParamsMED), na.rm=TRUE)

#The analogous plant-level function for LAI
sum(plant_LAI(exampleforest, SpParamsMED))

#The analogous plant-level function for fuel loading
sum(plant_fuelLoading(exampleforest, SpParamsMED))

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

#Cohort IDs in the models
plant_ID(exampleforest, SpParamsMED)
```

`plot.forest` *Plot forest attributes*

Description

Convenient wrappers for vertical forest profiles (see `vprofile_leafAreaDensity`).

Usage

```
## S3 method for class 'forest'
plot(
  x,
  SpParams,
  type = "LeafAreaDensity",
  byCohorts = FALSE,
  bySpecies = FALSE,
  includeHerbs = FALSE,
  ...
)

## S3 method for class 'forest'
shinyplot(x, SpParams, ...)
```

Arguments

`x`	An object of class `forest`.
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`type`	A string of the plot type: "LeafAreaDensity", "RootDistribution", "FuelBulkDensity", "PARExtinction", "SWRExtinction" or "WindExtinction".
`byCohorts`	A logical flag to separate profiles for each cohort.
`bySpecies`	A logical flag to aggregate results by species.
`includeHerbs`	A logical flag to include herbaceous layer in the profile.
`...`	Additional parameters to vertical profiles

Value

A ggplot or a shiny application, depending on the function.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`vprofile_leafAreaDensity`

Examples

```
data(exampleforest)
data(SpParamsMED)
plot(exampleforest, SpParamsMED)
```

`plot.spwb` *Plots simulation results*

Description

Function `plot` plots time series of the results of the soil plant water balance model (see `spwb`), plant water balance model (see `pwb`), the forest growth model (see `growth`) or the forest dynamics model (see `fordyn`).

Usage

```
## S3 method for class 'spwb'
plot(
  x,
  type = "PET_Precipitation",
  cohorts = NULL,
  bySpecies = FALSE,
  dates = NULL,
  subdaily = FALSE,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  summary.freq = NULL,
  ...
)

## S3 method for class 'pwb'
plot(
  x,
  type = "PlantTranspiration",
  cohorts = NULL,
  bySpecies = FALSE,
  dates = NULL,
  subdaily = FALSE,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  summary.freq = NULL,
  ...
)

## S3 method for class 'growth'
plot(
  x,
  type = "PET_Precipitation",
  cohorts = NULL,
  bySpecies = FALSE,
  dates = NULL,
  subdaily = FALSE,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  summary.freq = NULL,
  ...
)
```

```
## S3 method for class 'fordyn'
plot(
  x,
  type = "StandBasalArea",
  cohorts = NULL,
  bySpecies = FALSE,
  dates = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  summary.freq = NULL,
  ...
)
```

Arguments

`x`	An object of class `spwb`, `pwb`, `growth` or `fordyn`.
`type`	The information to be plotted (see details)
`cohorts`	An integer, boolean or character vector to select the plant cohorts to be plotted. If `cohorts = "T"` (resp. `cohorts = "S"`) then all tree (resp. shrub) cohorts will be displayed.
`bySpecies`	Allows aggregating output by species, before 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), where LAI values are those of `LAIlive`.
`dates`	A Date vector with a subset of dates to be plotted.
`subdaily`	Whether subdaily results should be shown, only for simulations using `transpirationMode = "Sperry"` and having set `subdailyResults = TRUE` in the simulation control object. If `subdaily = TRUE`, then the valid strings for `type` are listed in `plot.spwb_day`.
`xlim`	Range of values for x.
`ylim`	Range of values for y.
`xlab`	x-axis label.
`ylab`	y-axis label.
`summary.freq`	Frequency of summary statistics (see `cut.Date`).
`...`	Additional parameters for function `plot` (not used).

Details

The following plots are currently available for `spwb` (most of them also for `pwb`):

- `"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.
- `"SoilRWC"`: Soil relative water content (in percent of field capacity).
- `"SoilTheta"`: Soil moisture water content (in percent volume).
- `"SoilVol"`: Soil water volumetric content (in mm).
- `"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.
- `"LAI"`: Expanded and dead leaf area index of the whole stand.
- `"PlantLAI"`: Plant cohort leaf area index (expanded leaves).
- `"PlantLAIlive"`: Plant cohort leaf area index ("live" leaves).
- `"PlantStress"`: Plant cohort average daily drought stress.
- `"PlantTranspiration"`: Plant cohort transpiration.
- `"TranspirationPerLeaf"`: Plant cohort transpiration per leaf area.
- `"PlantGrossPhotosynthesis"`: Plant cohort photosynthesis.
- `"GrossPhotosynthesisPerLeaf"`: Plant cohort photosynthesis per leaf area.
- `"StemRWC"`: Average daily stem relative water content.
- `"LeafRWC"`: Average daily leaf relative water content.
- `"LFMC"`: Live fuel moisture content.

The following plots are available for `spwb` and `pwb` *only if* `transpirationMode = "Granier"`:

- `"PlantPsi"`: Plant cohort water potential.
- `"FPAR"`: Fraction of PAR at the canopy level of each plant cohort.
- `"AbsorbedSWRFraction"`: Fraction of SWR absorbed by each plant cohort.

The following plots are available for `spwb` and `pwb` *only if* `transpirationMode = "Sperry"`:

- `"SoilPlantConductance"`: Average instantaneous overall soil plant conductance (calculated as the derivative of the supply function).
- `"LeafPsiMin"`: Midday leaf water potential.
- `"LeafPsiMax"`: Pre-dawn leaf water potential.
- `"LeafPsiRange"`: Range of leaf water potential.
- `"LeafPsiMin_SL"`: Minimum water potential of sunlit leaves.
- `"LeafPsiMax_SL"`: Maximum water potential of sunlit leaves.
- `"LeafPsiMin_SH"`: Minimum water potential of shade leaves.
- `"LeafPsiMax_SH"`: Maximum water potential of shade leaves.
- `"TempMin_SL"`: Minimum temperature of sunlit leaves.
- `"TempMax_SL"`: Maximum temperature of sunlit leaves.

- "TempMin_SH": Minimum temperature of shade leaves.
- "TempMax_SH": Maximum temperature of shade leaves.
- "GSWMin_SL": Minimum stomatal conductance of sunlit leaves.
- "GSWMax_SL": Maximum stomatal conductance of sunlit leaves.
- "GSWMin_SH": Minimum stomatal conductance of shade leaves.
- "GSWMax_SH": Maximum stomatal conductance of shade leaves.
- "StemPsi": Midday (upper) stem water potential.
- "RootPsi": Midday root crown water potential.
- "PlantNetPhotosynthesis": Plant cohort net photosynthesis.
- "NetPhotosynthesisPerLeaf": Plant cohort net photosynthesis per leaf area.
- "PlantWUE": Plant cohort daily water use efficiency.
- "PlantAbsorbedSWR": Plant cohort absorbed short wave radiation.
- "AbsorbedSWRPerLeaf": Plant cohort absorbed short wave radiation per leaf area.
- "PlantNetLWR": Plant cohort net long wave radiation.
- "NetLWRPerLeaf": Plant cohort net long wave radiation per leaf area.
- "AirTemperature": Minimum/maximum/mean daily temperatures above canopy.
- "CanopyTemperature": Minimum/maximum/mean daily temperatures inside canopy.
- "SoilTemperature": Minimum/maximum/mean daily temperatures inside the first soil layer.
- "CanopyEnergyBalance": Canopy energy balance components.
- "SoilEnergyBalance": Soil energy balance components.

In addition to the former, the following plots are available for objects growth or fordyn:

- "CarbonBalance": Stand-level carbon balance components.
- "BiomassBalance": Stand-level biomass balance components.
- "GrossPhotosynthesis": Gross photosynthesis rate per dry weight.
- "MaintenanceRespiration": Maintenance respiration cost per dry weight.
- "PhotosynthesisMaintenanceRatio": The ratio of gross photosynthesis over maintenance respiration.
- "RootExudation": Root exudation rate per dry weight.
- "LabileCarbonBalance": Labile carbon balance per dry weight.
- "SugarLeaf": Sugar concentration in leaves.
- "StarchLeaf": Starch concentration in leaves.
- "SugarSapwood": Sugar concentration in sapwood.
- "StarchSapwood": Starch concentration in sapwood.
- "SugarTransport": Phloem sugar transport rate.
- "StructuralBiomassBalance": Daily structural biomass balance (g dry · ind-2).
- "LabileBiomassBalance": Daily labile biomass balance (g dry · ind-2).

- `"PlantBiomassBalance"`: Daily plant biomass balance, i.e. labile change + structural change (g dry · ind-2).
- `"MortalityBiomassLoss"`: Biomass loss due to mortality (g dry · m-2).
- `"PlantBiomassBalance"`: Daily cohort biomass balance (including mortality) (g dry · m-2).
- `"LeafBiomass"`: Leaf structural dry biomass per individual.
- `"SapwoodBiomass"`: Sapwood dry biomass per individual.
- `"FineRootBiomass"`: Fine root dry biomass per individual.
- `"SapwoodArea"`: Sapwood area per individual.
- `"LeafArea"`: Leaf area per individual.
- `"FineRootArea"`: Fine root area per individual (only for `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`).
- `"DBH"`: Diameter at breast height (in cm) for an average individual of each plant cohort.
- `"Height"`: Height (in cm) for an average individual of each plant cohort.
- `"SAgrowth"`: Sapwood area growth rate.
- `"LAgrowth"`: Leaf area growth rate.
- `"FRAgrowth"`: Fine root area growth rate (only for `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`).
- `"HuberValue"`: Ratio of leaf area to sapwood area.
- `"RootAreaLeafArea"`: Ratio of fine root area to leaf area (only for `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`).

Finally, the following plots are only available for `fordyn` simulation results:

- `"StandBasalArea"`: Stand basal area of living trees.
- `"StandDensity"`: Stand density of living trees.
- `"SpeciesBasalArea"`: Basal area of living trees by species.
- `"SpeciesDensity"`: Density of living trees by species.
- `"CohortBasalArea"`: Basal area of living trees by plant cohort.
- `"CohortDensity"`: Density of living trees by plant cohort.

Value

An ggplot object

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`spwb`, `pwb`, `growth`, `fordyn`, `summary.spwb`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S1 <- spwb(x, examplemeteo, latitude = 41.82592, elevation = 100)

#Plot results
plot(S1)
```

`plot.spwb_day` *Plots simulation results for one day*

Description

Functions to plot the sub-daily simulation results of `spwb_day`, `growth_day` or the transpiration calculations of `transp_transpirationSperry` or `transp_transpirationSureau`.

Usage

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

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

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

Arguments

`x`	An object of class `spwb_day`, `growth_day` or `pwb_day`.
`type`	The information to be plotted (see details).
`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).
`xlim`	Range of values for x.
`ylim`	Range of values for y.
`xlab`	x-axis label.
`ylab`	y-axis label.
`...`	Additional parameters for function `plot`.

Details

The following plots are available for `spwb_day` and `pwb_day`:

- `"LeafPsi"`: Leaf water potential (for shade and sunlit leaves).
- `"LeafPsiAverage"`: Average leaf water potential.
- `"RootPsi"`: Root crown water potential.
- `"StemPsi"`: Stem water potential.
- `"StemPLC"`: (Average) percentage of loss conductance in the stem conduits.

- `"StemRWC"`: (Average) relative water content in the stem.
- `"LeafRWC"`: Relative water content in the leaf.
- `"StemSympRWC"`: (Average) relative water content in the stem symplasm.
- `"LeafSympRWC"`: 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.
- `"TranspirationPerLeaf"`: Plant cohort transpiration per leaf area.
- `"PlantGrossPhotosynthesis"`: Plant cohort gross photosynthesis per ground area.
- `"GrossPhotosynthesisPerLeaf"`: Plant cohort gross photosynthesis per leaf area.
- `"PlantNetPhotosynthesis"`: Plant cohort net photosynthesis per ground area.
- `"NetPhotosynthesisPerLeaf"`: Plant cohort net photosynthesis per leaf area.
- `"LeafTranspiration"`: Instantaneous transpiration per leaf area (differentiates sunlit and shade leaves).
- `"LeafGrossPhotosynthesis"`: Instantaneous gross photosynthesis per leaf area (differentiates sunlit and shade leaves).
- `"LeafNetPhotosynthesis"`: Instantaneous net photosynthesis per leaf area (differentiates sunlit and shade leaves).
- `"LeafAbsorbedSWR"`: Absorbed short wave radiation per leaf area (differentiates sunlit and shade leaves).
- `"LeafAbsorbedPAR"`: Absorbed photosynthetically-active radiation per leaf area (differentiates sunlit and shade leaves).
- `"LeafNetLWR"`: Net long wave radiation per leaf area (differentiates sunlit and shade leaves).
- `"LeafCi"`: Leaf intercellular CO2 concentration (differentiates sunlit and shade leaves).
- `"LeafIntrinsicWUE"`: Leaf intrinsic water use efficiency, i.e. the ratio between instantaneous photosynthesis and stomatal conductance (differentiates sunlit and shade leaves).
- `"LeafVPD"`: Leaf vapour pressure deficit (differentiates sunlit and shade leaves).
- `"LeafStomatalConductance"`: Leaf stomatal conductance to water vapour (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.
- `"PlantWaterBalance"`: Difference between water extraction from the soil and transpired water per ground area.
- `"WaterBalancePerLeaf"`: Difference between water extraction from the soil and transpired water per leaf area.

And the following plots are additionally available for `growth_day`:

- "GrossPhotosynthesis": Gross photosynthesis rate per dry weight.
- "MaintenanceRespiration": Maintenance respiration cost per dry weight.
- "RootExudation": Root exudation rate per dry weight.
- "LabileCarbonBalance": Labile carbon balance per dry weight.
- "SugarLeaf": Sugar concentration in leaves.
- "StarchLeaf": Starch concentration in leaves.
- "SugarSapwood": Sugar concentration in sapwood.
- "StarchSapwood": Starch concentration in sapwood.
- "SugarTransport": Phloem sugar transport rate.

Value

An ggplot object

Note

Only for soil plant water balance simulations using `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`. This function can be used to display subdaily dynamics of corresponding to single days on `spwb` runs, if control option `subdailyResults` is set to `TRUE`. See also option `subdaily` in `plot.spwb`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`spwb_day`, `growth_day`, `plot.spwb`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (2 layers)
examplesoil <- defaultSoilParams(4)

#Switch to 'Sperry' transpiration mode
control <- defaultControl("Sperry")

#Simulate one day only
x2 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)
d <- 100
```

```
date <- examplemeteo$dates[d]
meteovec <- unlist(examplemeteo[d,])
sd2 <- spwb_day(x2, date, meteovec,
              latitude = 41.82592, elevation = 100, slope= 0, aspect = 0)

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

poblet_trees *Example forest inventory data*

Description

Example data to illustrate the creation of forest objects from inventory data, coming from a forest inventory survey, used to illustrate the general function forest_mapTreeTable:

- poblet_trees - Data frame with example tree plot data from Poblet, Catalonia (717 observations and 4 variables).
    - Plot.Code - Plot ID (character)
    - Indv.Ref - Tree individual (integer)
    - Species - Species name (character)
    - Diameter.cm - Tree diameter at breast height (cm)

Source

- Data table poblet_trees corresponds to field data sampled by the Catalan Forest Ownership Center (Centre de la Propietat Forestal; CPF).

See Also

forest_mapTreeTable

regeneration *Plant regeneration*

Description

Functions to simulate annual plant regeneration from seed recruitment or from resprouting

Usage

```
regeneration_seedproduction(forest, SpParams, control)

regeneration_seedrefill(seedBank, refillSpecies, refillPercent = NULL)

regeneration_seedmortality(seedBank, SpParams, minPercent = 1)

regeneration_recruitment(
  forest,
  SpParams,
  control,
  minMonthTemp,
  moistureIndex,
  verbose = FALSE
)

regeneration_resprouting(
  forest,
  internalMortality,
  SpParams,
  control,
  management_results = NULL
)
```

Arguments

`forest`	An object of class `forest`.
`SpParams`	A data frame with species parameters (see `SpParamsMED` and `SpParamsDefinition`).
`control`	A list with default control parameters (see `defaultControl`).
`seedBank`	A data frame with columns 'Species' and 'Percent', describing a seed bank.
`refillSpecies`	A string vector of species names corresponding to seed rain to refill seed bank.
`refillPercent`	A numeric vector of indicating the percentage of seed bank refilling (if missing then seed bank is set to 100%).
`minPercent`	A minimum percent of seed bank to retain entry in `seedBank` element of `forest`.
`minMonthTemp`	Minimum month temperature.
`moistureIndex`	Moisture index (annual precipitation over annual potential evapotranspiration).
`verbose`	Boolean flag to indicate console output during calculations.
`internalMortality`	A data frame with mortality occurred in the last year of simulation.
`management_results`	The result of calling a management function (see `defaultManagementFunction`).

Details

- `regeneration_seedproduction` evaluates if reproductive individuals (i.e. sufficiently tall individuals) are present.

- `regeneration_seedrefill` fills seed bank of input `forest` object with seed rain.
- `regeneration_seedmortality` updates seed bank of input `forest` object according to annual seed mortality.
- `regeneration_recruitment` evaluates recruitment from the seed bank (or local seed production if seed bank is missing). Minimum month temperature and moisture index values are used to determine if recruitment was successful. Species also require a minimum amount of light at the ground level.
- `regeneration_resprouting` evaluates resprouting occurs after "mortality" from die-back (including drought- or pathogen-induced dessication), cutting or burning of the aerial part in a species with resprouting ability, but not after carbon starvation or baseline mortality (unspecific mortality causes).

Value

- `regeneration_seedproduction` returns a list of species names
- `regeneration_seedrefill` and `regeneration_seedmortality` return a copy of the input `data.frame` object with an update seed bank.
- `regeneration_resprouting` and `regeneration_recruitment` return a new object of class `forest` with the new plant cohorts.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`fordyn`

Examples

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

#Default species parameterization
data(SpParamsMED)

#Initialize control parameters
control <- defaultControl("Granier")
control$recruitmentMode = "deterministic"

#Recruitment limits
plant_parameter(exampleforest, SpParamsMED, "MinTempRecr")
plant_parameter(exampleforest, SpParamsMED, "MinMoistureRecr")

#Compare seed recruitment outcomes
regeneration_recruitment(exampleforest, SpParamsMED, control, 0, 0.25)
regeneration_recruitment(exampleforest, SpParamsMED, control, 3, 0.25)
```

resetInputs *Reset simulation inputs*

Description

Function `resetInputs()` allows resetting state variables in `x` to their defaults.

Usage

```
resetInputs(x)
```

Arguments

`x`	An object of class `spwbInput` or `growthInput`.

Value

Does not return any value. Instead, it modifies input object `x`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`spwbInput`, `growthInput`, `spwb`

resistances *Soil-plant resistances*

Description

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

Usage

```
resistances(
  x,
  cohort,
  relative = FALSE,
  draw = FALSE,
  cumulative = FALSE,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

`x`	An object of class `spwb`, `pwb`, `growth` or `fordyn`. The function only works with the result of simulations with `transpirationMode = "Sperry"`.
`cohort`	An string indicating the cohort for which resistances are desired.
`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.
`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` then a plot object is returned.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`waterUseEfficiency`, `droughtStress`

`root` *Root functions*

Description

Functions to calculate properties of fine/coarse roots within the soil, given root system parameters and soil layer definition.

Usage

```
root_conicDistribution(Zcone, d)

root_ldrDistribution(Z50, Z95, d)

root_individualRootedGroundArea(VolInd, V, d, rfc)

root_specificRootSurfaceArea(specificRootLength, rootTissueDensity)
```

```
root_fineRootRadius(specificRootLength, rootTissueDensity)

root_fineRootHalfDistance(rootLengthDensity)

root_fineRootAreaIndex(
  Ksoil,
  krhizo,
  lai,
  specificRootLength,
  rootTissueDensity,
  rootLengthDensity
)

root_fineRootBiomass(
  Ksoil,
  krhizo,
  lai,
  N,
  specificRootLength,
  rootTissueDensity,
  rootLengthDensity
)

root_rhizosphereMaximumConductance(
  Ksoil,
  fineRootBiomass,
  lai,
  N,
  specificRootLength,
  rootTissueDensity,
  rootLengthDensity
)

root_fineRootSoilVolume(fineRootBiomass, specificRootLength, rootLengthDensity)

root_coarseRootSoilVolumeFromConductance(
  Kmax_rootxylem,
  VCroot_kmax,
  Al2As,
  v,
  d,
  rfc
)

root_coarseRootLengthsFromVolume(VolInd, v, d, rfc)

root_coarseRootLengths(v, d, depthWidthRatio = 1)
```

```
root_coarseRootSoilVolume(v, d, depthWidthRatio = 1)

root_horizontalProportions(poolProportions, VolInd, N, V, d, rfc)
```

Arguments

`Zcone`	A vector of depths (in mm) corresponding to the root cone tip.
`d`	The width (in mm) corresponding to each soil layer.
`Z50`	A vector of depths (in mm) corresponding to 50% of roots.
`Z95`	A vector of depths (in mm) corresponding to 95% of roots.
`VolInd`	Volume of soil (in m3) occupied by coarse roots per individual.
`V`	Matrix of proportions of fine roots (cohorts x soil layers).
`rfc`	Percentage of rock fragment content (volume basis) for each layer.
`specificRootLength`	Specific fine root length (length of fine roots over weight).
`rootTissueDensity`	Fine root tissue density (weight over volume at turgidity).
`rootLengthDensity`	Fine root length density (length of fine roots over soil volume; cm/cm3)
`Ksoil`	Soil saturated conductivity (mmol·m-1·s-1·MPa-1).
`krhizo`	Rhizosphere maximum conductance per leaf area (mmol·m-2·s-1·MPa-1).
`lai`	Leaf area index.
`N`	Density of individuals per hectare.
`fineRootBiomass`	Biomass of fine roots (g).
`Kmax_rootxylem`	Sapwood-specific hydraulic conductivity of root xylem (in kg H2O·s-1·m-1·MPa-1).
`VCroot_kmax`	Root xylem maximum conductance per leaf area (mmol·m-2·s-1·MPa-1).
`Al2As`	Leaf area to sapwood area ratio (in m2·m-2).
`v`	Vector of proportions of fine roots in each soil layer.
`depthWidthRatio`	Ratio between radius of the soil layer with the largest radius and maximum rooting depth.
`poolProportions`	Division of the stand area among plant cohorts (proportions).

Details

- `root_conicDistribution()` assumes a (vertical) conic distribution of fine roots, whereas `root_ldrDistribution()` distributes fine roots according to the linear dose response model of Schenck & Jackson (2002). Return a matrix of fine root proportions in each layer with as many rows as elements in `Z` (or `Z50`) and as many columns as soil layers.

- `root_coarseRootLengths()` and `root_coarseRootLengthsFromVolume()` estimate the length of coarse roots (mm) for each soil layer, including axial and radial lengths.
- `root_coarseRootSoilVolume` estimates the soil volume (m3) occupied by coarse roots of an individual.
- `root_coarseRootSoilVolumeFromConductance` estimates the soil volume (m3) occupied by coarse roots of an individual from root xylem conductance.
- `root_fineRootHalfDistance()` calculates the half distance (cm) between neighbouring fine roots.
- `root_fineRootRadius()` calculates the radius of fine roots (cm).
- `root_fineRootAreaIndex()` estimates the fine root area index for a given soil conductivity and maximum rhizosphere conductance.
- `root_fineRootBiomass()` estimates the biomass of fine roots (g dry/individual) for a given soil conductivity and maximum rhizosphere conductance.
- `root_rhizosphereMaximumConductance()` is the inverse of the preceeding function, i.e. it estimates rhizosphere conductance from soil conductivity and fine root biomass.
- `root_fineRootSoilVolume()` calculates the soil volume (m3) occupied with fine roots.
- `root_specificRootSurfaceArea()` returns the specific fine root area (cm2/g).
- `root_individualRootedGroundArea()` calculates the area (m2) covered by roots of an individual, for each soil layer.
- `root_horizontalProportions()` calculates the (horizontal) proportion of roots of each cohort in the water pool corresponding to itself and that of other cohorts, for each soil layer. Returns a list (with as many elements as cohorts) with each element being a matrix.

Value

See details.

Author(s)

Miquel De Cáceres Ainsa, CREAF

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`, `spwbInput`, `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 <- defaultSoilParams(4)

#Calculate conic root system for trees
V1 <- root_conicDistribution(Z=rep(2000,ntree), s$widths)
print(V1)

#Calculate LDR root system for trees (Schenck & Jackson 2002)
V2 <- root_ldrDistribution(Z50 = rep(200,ntree),
                           Z95 = rep(1000,ntree), s$widths)
print(V2)
```

`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).

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., and 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)
```

---

shinyplot *Shiny app with interactive plots*

Description

Creates a shiny app with interactive plots for simulation results and evaluation

Usage

```
shinyplot(x, ...)

## S3 method for class 'growth'
shinyplot(x, measuredData = NULL, ...)

## S3 method for class 'spwb'
shinyplot(x, measuredData = NULL, ...)

## S3 method for class 'pwb'
shinyplot(x, measuredData = NULL, ...)
```

```
## S3 method for class 'fordyn'
shinyplot(x, measuredData = NULL, ...)

## S3 method for class 'growth_day'
shinyplot(x, ...)

## S3 method for class 'spwb_day'
shinyplot(x, ...)

## S3 method for class 'pwb_day'
shinyplot(x, ...)
```

Arguments

`x`	An object of the right class.
`...`	Additional parameters.
`measuredData`	A data frame with observed/measured values (see `evaluation_plot`).

Details

Only run this function in interactive mode. When `measuredData` is not `NULL`, an additional panel is shown for evaluation plots.

Value

An object that represents the shiny app

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`plot.spwb`, `evaluation_plot`

`soil` *Soil initialization*

Description

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

Usage

```
soil(x, VG_PTF = "Toth")

## S3 method for class 'soil'
summary(object, model = "SX", ...)
```

Arguments

`x`	A data frame of soil parameters (see an example in `defaultSoilParams`).
`VG_PTF`	Pedotransfer functions to obtain parameters for the van Genuchten-Mualem equations. Either `"Carsel"` (Carsel and Parrish 1988) or `"Toth"` (Toth et al. 2015).
`object`	An object of class `soil`.
`model`	Either 'SX' or 'VG' for Saxton or Van Genuchten pedotransfer models.
`...`	Additional parameters to `summary`.

Details

Function `summary` 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.

If available, the user can specify columns `VG_alpha`, `VG_n`, `VG_theta_res`, `VG_theta_sat` and `K_sat`, to override Van Genuchten parameters an saturated conductivity estimated from pedotransfer functions when calling function `soil`.

Value

Function `soil` returns a data frame of class `soil` with the following columns:

- `widths`: 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).
- `nitrogen`: Sum of total nitrogen (ammonia, organic and reduced nitrogen) for each layer (in g/kg).
- `rfc`: Percentage of rock fragment content for each layer.
- `macro`: Macroporosity for each layer (estimated using Stolf et al. 2011).
- `Ksat`: Saturated soil conductivity for each layer (estimated using function `soil_saturatedConductivitySX`.
- `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.
- `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.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Carsel, R.F., and 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., and 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_redefineLayers`, `soil_psi2thetaSX`, `soil_psi2thetaVG`, `spwb`, `defaultSoilParams`

Examples

```
# Default parameters
df_soil <- defaultSoilParams()

# Initializes soil
s = soil(df_soil)
s

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

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

# Add columns 'VG_theta_sat' and 'VG_theta_res' with custom values
df_soil$VG_theta_sat <- 0.400
df_soil$VG_theta_res <- 0.040

# Reinitialize soil (should override estimations)
s2 = soil(df_soil)
s2
summary(s2, model="VG")
```

`soil_redefineLayers` *Redefine soil layer widths*

Description

Allows redefining soil layer widths of an input data frame of soil parameters.

Usage

```
soil_redefineLayers(x, widths = c(300, 700, 1000, 2000))
```

Arguments

x	A data frame of soil parameters (see an example in defaultSoilParams) or an object of class soil.
widths	A numeric vector indicating the desired layer widths, in mm.

Details

If an initialized soil is supplied, its hydraulic parameters will be recalculated and the value of state variables will be lost.

Value

A data frame or soil object with soil parameters, depending on the class of x.

Author(s)

Víctor Granda, EMF-CREAF

Miquel De Cáceres Ainsa, EMF-CREAF

See Also

soil, defaultSoilParams

Examples

```
# Define initial soil with 5 layers
spar <- defaultSoilParams(5)
spar

# Redefine to four layers
soil_redefineLayers(spar)

# Same but after soil parameter initialization
examplesoil <- soil(spar)
examplesoil

soil_redefineLayers(examplesoil)
```

---

soil_texture *Soil texture and hydraulics*

Description

Low-level functions relating soil texture with soil hydraulics and soil water content.

Usage

```
soil_saturatedConductivitySX(clay, sand, bd, om = NA_real_, mmol = TRUE)

soil_unsaturatedConductivitySX(
  theta,
  clay,
  sand,
  bd,
  om = NA_real_,
  mmol = TRUE
)

soil_thetaSATSX(clay, sand, om = NA_real_)

soil_theta2psiSX(clay, sand, theta, om = NA_real_)

soil_psi2thetaSX(clay, sand, psi, om = NA_real_)

soil_psi2kVG(ksat, n, alpha, theta_res, theta_sat, psi)

soil_psi2cVG(n, alpha, theta_res, theta_sat, psi)

soil_psi2thetaVG(n, alpha, theta_res, theta_sat, psi)

soil_theta2psiVG(n, alpha, theta_res, theta_sat, theta)

soil_USDAType(clay, sand)

soil_thetaFC(soil, model = "SX")

soil_thetaWP(soil, model = "SX")

soil_thetaSAT(soil, model = "SX")

soil_waterFC(soil, model = "SX")

soil_waterSAT(soil, model = "SX")

soil_waterWP(soil, model = "SX")

soil_waterPsi(soil, psi, model = "SX")

soil_waterExtractable(soil, model = "SX", minPsi = -5)

soil_theta(soil, model = "SX")

soil_water(soil, model = "SX")
```

```
soil_rockWeight2Volume(pWeight, bulkDensity, rockDensity = 2.3)

soil_psi(soil, model = "SX")

soil_conductivity(soil, model = "SX")

soil_capacitance(soil, model = "SX")

soil_saturatedWaterDepth(soil, model = "SX")

soil_vanGenuchtenParamsCarsel(soilType)

soil_campbellParamsClappHornberger(soilType)

soil_vanGenuchtenParamsToth(clay, sand, om, bd, topsoil)

soil_retentionCurvePlot(
  soil,
  model = "SX",
  layer = 1,
  psi = seq(0, -6, by = -0.01),
  relative = TRUE,
  to = "SAT"
)
```

Arguments

`clay`	Percentage of clay (in percent weight).
`sand`	Percentage of sand (in percent weight).
`bd`	Bulk density (in g/cm3).
`om`	Percentage of organic matter (optional, in percent weight).
`mmol`	Boolean flag to indicate that saturated conductivity units should be returned in mmol/m/s/MPa. If `mmol = FALSE` then units are cm/day.
`theta`	Relative water content (in percent volume).
`psi`	Water potential (in MPa).
`ksat`	saturated hydraulic conductance
`n, alpha, theta_res, theta_sat`	Parameters of the Van Genuchten-Mualem model (m = 1 - 1/n).
`soil`	Initialized soil object (returned by function `soil`).
`model`	Either 'SX' or 'VG' for Saxton's or Van Genuchten's water retention models; or 'both' to plot both retention models.
`minPsi`	Minimum water potential (in MPa) to calculate the amount of extractable water.
`pWeight`	Percentage of corresponding to rocks, in weight.
`bulkDensity`	Bulk density of the soil fraction (g/cm3).
`rockDensity`	Rock density (g/cm3).

`soilType`	A string indicating the soil type.
`topsoil`	A boolean flag to indicate topsoil layer.
`layer`	Soil layer to be plotted.
`relative`	Boolean flag to indicate that retention curve should be relative to field capacity or saturation.
`to`	Either 'SAT' (saturation) or 'FC' (field capacity).

Details

- `soil_psi2thetaSX()` and `soil_theta2psiSX()` calculate water potentials (MPa) 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.
- `soil_psi2thetaVG()` and `soil_theta2psiVG()` to the same calculations as before, but using the Van Genuchten - Mualem equations (Wösten & van Genuchten 1988).
- `soil_saturatedConductivitySX()` returns the saturated conductivity of the soil (in cm/day or mmol/m/s/MPa), estimated from formulae of Saxton et al. (1986) or Saxton & Rawls (2006) depending on whether organic matter is available.
- `soil_unsaturatedConductivitySX()` returns the unsaturated conductivity of the soil (in cm/day or mmol/m/s/MPa), estimated from formulae of Saxton et al. (1986) or Saxton & Rawls (2006) depending on whether organic matter is available.
- `soil_USDAType()` returns the USDA type (a string) for a given texture.
- `soil_vanGenuchtenParamsCarsel()` gives parameters for van Genuchten-Mualem equations (alpha, n, theta_res and theta_sat, where alpha is in MPa-1) for a given texture type (Leij et al. 1996)
- `soil_vanGenuchtenParamsToth()` gives parameters for van Genuchten-Mualem equations (alpha, n, theta_res and theta_sat, where alpha is in MPa-1) for a given texture, organic matter and bulk density (Toth et al. 2015).
- `soil_psi()` returns the water potential (MPa) of each soil layer, according to its water retention model.
- `soil_theta()` returns the moisture content (as percent of soil volume) of each soil layer, according to its water retention model.
- `soil_water()` returns the water volume (mm) of each soil layer, according to its water retention model.
- `soil_conductivity()` returns the conductivity of each soil layer (mmol/m/s/MPa), according the Saxton model.
- `soil_waterExtractable()` returns the water volume (mm) extractable from the soil according to its water retention curves and up to a given soil water potential.
- `soil_waterFC()` and `soil_thetaFC()` calculate the water volume (in mm) and moisture content (as percent of soil volume) of each soil layer at field capacity, respectively.
- `soil_waterWP()` and `soil_thetaWP()` calculate the water volume (in mm) and moisture content (as percent of soil volume) of each soil layer at wilting point (-1.5 MPa), respectively.
- `soil_waterSAT()`, `soil_thetaSATSX()` and `soil_thetaSAT()` calculate the saturated water volume (in mm) and moisture content (as percent of soil volume) of each soil layer.

- `soil_saturatedWaterDepth()` returns the depth to saturation in mm from surface.
- `soil_rockWeight2Volume()` transforms rock percentage from weight to volume basis.
- `soil_retentionCurvePlot()` allows ploting the water retention curve of a given soil layer.

Value

Depends on the function (see details).

Author(s)

Miquel De Cáceres Ainsa, CREAF

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., and Tóth, G. 2015. New generation of hydraulic pedotransfer functions for Europe. European Journal of Soil Science 66: 226–238.

See Also

`soil`

Examples

```
#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

# Define soil with default params
soil_df <- defaultSoilParams(4)
soil_df

# Initialize soil parameters and state variables
s = soil(soil_df)

# Plot Saxton's and Van Genuchten's water retention curves
soil_retentionCurvePlot(s, model="both")
```

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_thermalCapacity(soil, model = "SX")

soil_thermalConductivity(soil, model = "SX")

soil_temperatureGradient(widths, Temp)

soil_temperatureChange(
  widths,
  Temp,
  sand,
  clay,
  W,
  Theta_SAT,
  Theta_FC,
  Gdown,
  tstep
)
```

Arguments

soil	Soil object (returned by function soil).
model	Either 'SX' or 'VG' for Saxton's or Van Genuchten's pedotransfer models.
widths	Width of soil layers (in mm).
Temp	Temperature (in ºC) for each soil layer.
sand	Percentage of sand (in percent weight) for each layer.
clay	Percentage of clay (in percent weight) for each layer.
W	Soil moisture (in percent of field capacity) for each layer.
Theta_SAT	Relative water content (in percent volume) at saturation 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).
tstep	Time step (interval) in seconds.

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, CREAF

References

Cox, P.M., Betts, R.A., Bunton, C.B., Essery, R.L.H., Rowntree, P.R., and 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., and Best, M. 2009. New soil physical properties implemented in the Unified Model at PS18. 9–12.

See Also

`soil`

Examples

```
#Define soil and complete parameters
examplesoil = soil(defaultSoilParams(4))

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)
```

`species_values` *Species description functions*

Description

Functions to calculate attributes of a `forest` object by species or to extract species parameters from a species parameter table (`SpParamsMED`).

Usage

```
species_basalArea(x, SpParams)

species_cover(x, SpParams)

species_density(x, SpParams)

species_foliarBiomass(x, SpParams, gdd = NA_real_)

species_fuelLoading(x, SpParams, gdd = NA_real_, includeDead = TRUE)

species_LAI(x, SpParams, gdd = NA_real_, bounded = TRUE)

species_characterParameter(species, SpParams, parName)

species_parameter(
  species,
  SpParams,
  parName,
  fillMissing = TRUE,
  fillWithGenus = TRUE
)
```

Arguments

`x`	An object of class `forest`.
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`gdd`	Growth degree days (to account for leaf phenology effects).
`includeDead`	A flag to indicate that standing dead fuels (dead branches) are included.
`bounded`	A boolean flag to indicate that extreme values should be prevented (maximum tree LAI = 7 and maximum shrub LAI = 3)
`species`	A character vector of species names.
`parName`	A string with a parameter name.
`fillMissing`	A boolean flag to try imputation on missing values.
`fillWithGenus`	A boolean flag to try imputation of missing values using genus values.

Value

A vector with values for each species in `SpParams`:

- `species_basalArea`: Species basal area (m2/ha).
- `species_cover`: Shrub cover (in percent).
- `species_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.

- `species_foliarBiomass`: Standing biomass of leaves (in kg/m2).
- `species_fuel`: Fine fuel load (in kg/m2).
- `species_LAI`: Leaf area index (m2/m2).
- `species_phytovolume`: Shrub phytovolume (m3/m2).
- `species_parameter`: A numeric vector with the parameter values of each input species.
- `species_characterParameter`: A character vector with the parameter values of each input species.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

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

Examples

```
# Default species parameterization
data(SpParamsMED)

# Load example plot
data(exampleforest)

# Species basal area in the forest plot
species_basalArea(exampleforest, SpParamsMED)

# Value of parameter "Psi_Extract" for two species
species_parameter(c("Pinus halepensis", "Quercus ilex"), SpParamsMED, "Psi_Extract")
```

`SpParams` *Data tables with species parameter definition and values for different countries*

Description

A data sets of species parameter definition and values, the latter resulting from existing databases, fit to empirical data or expert-based guesses.

Format

- Data frame `SpParamsDefinition` has parameters in rows and columns 'ParameterName', 'ParameterGroup', 'Definition', 'Type' and 'Units'.
- Data frames `SpParamsMED` has species or genus as rows and column names equal to parameter names in `SpParamsDefinition`.

Details

SpParamsMED was the official species parameter for package versions up to v.4.0.0, but will not be maintained in the future. Additional species parameter tables for different countries are distributed via package medfatetraits, available from GitHub (https://emf-creaf.github.io/medfatetraits/).

Examples

```
data(SpParamsDefinition)
data(SpParamsMED)
```

---

spwb *Soil-plant water balance*

Description

Function spwb() is a water balance model that determines changes in soil moisture, soil water potentials, plant transpiration and drought stress at daily steps for a given forest stand during a period specified in the input climatic data. Function pwb() performs plant water balance only (i.e. soil moisture dynamics is an input) at daily steps for a given forest stand during a period specified in the input climatic data. On both simulation functions plant transpiration and photosynthesis processes are conducted with different level of detail depending on the transpiration mode.

Usage

```
spwb(
  x,
  meteo,
  latitude,
  elevation,
  slope = NA_real_,
  aspect = NA_real_,
  CO2ByYear = numeric(0),
  waterTableDepth = NA_real_
)

pwb(
  x,
  meteo,
  W,
  latitude,
  elevation,
  slope = NA_real_,
  aspect = NA_real_,
  canopyEvaporation = numeric(0),
  snowMelt = numeric(0),
  soilEvaporation = numeric(0),
  herbTranspiration = numeric(0),
  CO2ByYear = numeric(0)
)
```

Arguments

`x`	An object of class `spwbInput`.
`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`). Alternatively, a column called `"dates"` or `"Dates"` can contain `Date` or `POSIXct` classes. The following columns are required and cannot have missing values: • `MinTemperature`: Minimum temperature (in degrees Celsius). • `MaxTemperature`: Maximum temperature (in degrees Celsius). • `Precipitation`: Precipitation (in mm). The following columns are required but can contain missing values (NOTE: missing values will raise warnings): • `MinRelativeHumidity`: Minimum relative humidity (in percent). • `MaxRelativeHumidity`: Maximum relative humidity (in percent). • `Radiation`: Solar radiation (in MJ/m2/day). The following columns are optional: • `WindSpeed`: Above-canopy wind speed (in m/s). This column may not exist, or can be left with `NA` values. In both cases simulations will assume a constant value specified in `defaultControl`. • `CO2`: Atmospheric (above-canopy) CO2 concentration (in ppm). This column may not exist, or can be left with `NA` values. In both cases simulations will assume a constant value specified in `defaultControl`. • `Patm`: Atmospheric pressure (in kPa). This column may not exist, or can be left with `NA` values. In both cases, a value is estimated from elevation.
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`CO2ByYear`	A named numeric vector with years as names and atmospheric CO2 concentration (in ppm) as values. Used to specify annual changes in CO2 concentration along the simulation (as an alternative to specifying daily values in `meteo`).
`waterTableDepth`	Water table depth (in mm). When not missing, capillarity rise will be allowed if lower than total soil depth.
`W`	A matrix with the same number of rows as `meteo` and as many columns as soil layers, containing the soil moisture of each layer as proportion of field capacity.
`canopyEvaporation`	A vector of daily canopy evaporation (from interception) values (mm). The length should match the number of rows in `meteo`.
`snowMelt`	A vector of daily snow melt values (mm). The length should match the number of rows in `meteo`.

`soilEvaporation`
: A vector of daily bare soil evaporation values (mm). The length should match the number of rows in `meteo`.

`herbTranspiration`
: A vector of daily herbaceous transpiration values (mm). The length should match the number of rows in `meteo`.

Details

The simulation functions allow using three different sub-models of transpiration and photosynthesis:

- The sub-model corresponding to 'Granier' transpiration mode is illustrated by function `transp_transpirationGranie` and was described in De Caceres et al. (2015), and implements an approach originally described in Granier et al. (1999).
- The sub-model corresponding to 'Sperry' transpiration mode is illustrated by function `transp_transpirationSperry` and was described in De Caceres et al. (2021), and implements a modelling approach originally described in Sperry et al. (2017).
- The sub-model corresponding to 'Sureau' transpiration mode is illustrated by function `transp_transpirationSureau` and was described for model SurEau-Ecos v2.0 in Ruffault et al. (2022).

Simulations using the 'Sperry' or 'Sureau' transpiration mode are computationally much more expensive than 'Granier'.

Value

Function spwb returns a list of class 'spwb' whereas function pwb returns a list of class 'pwb'. There are many elements in common in these lists, so they are listed here together:

- `"latitude"`: Latitude (in degrees) given as input.
- `"topography"`: Vector with elevation, slope and aspect given as input.
- `"weather"`: A copy of the input weather data frame.
- `"spwbInput"`: An copy of the object `x` of class `spwbInput` given as input.
- `"spwbOutput"`: An copy of the final state of the object `x` of class `spwbInput`.
- `"WaterBalance"`: A data frame where different variables (in columns) are given for each simulated day (in rows):
  - `"PET"`: Potential evapotranspiration (in mm).
  - `"Precipitation"`: Input precipitation (in mm).
  - `"Rain"`: Precipitation as rainfall (in mm).
  - `"Snow"`: Precipitation as snowfall (in mm).
  - `"NetRain"`: Net rain, after accounting for interception (in mm).
  - `"Infiltration"`: The amount of water infiltrating into the soil (in mm).
  - `"InfiltrationExcess"`: Excess infiltration in the topmost layer leading to an increase in runoff (in mm).
  - `"SaturationExcess"`: Excess saturation in the topmost layer leading to an increase in runoff (in mm).

  - "CapillarityRise": Water entering the soil via capillarity rise (mm) from the water table, if waterTableDepth is supplied.
  - "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).
  - "HerbTranspiration": Transpiration due to the herbaceous layer (in mm).
  - "PlantExtraction": Amount of water extracted from soil by woody plants (in mm).
  - "Transpiration": Woody plant transpiration (in mm).
  - "HydraulicRedistribution": Water redistributed among soil layers, transported through the plant hydraulic network.
- "EnergyBalance": A data frame with the daily values of energy balance components for the soil and the canopy (only for transpirationMode = "Sperry" or transpirationMode = "Sureau").
- "Temperature": A data frame with the daily values of minimum/mean/maximum temperatures for the atmosphere (input), canopy and soil (only for transpirationMode = "Sperry" or transpirationMode = "Sureau").
- "Soil": A list with the following subelements:
  - "SWC": Soil water content (percent of soil volume) in each soil layer (and overall).
  - "RWC": Relative soil moisture content (relative to field capacity) in each soil layer (and overall).
  - "REW": Relative extractable water (min. psi = -5 MPa) in each soil layer (and overall).
  - "ML": Soil water volume in each soil layer (in L/m2) (and overall).
  - "Psi": Soil water potential in each soil layer (in MPa) (and overall).
  - "PlantExt": Plant extraction from each soil layer (in mm) (and overall).
  - "HydraulicInput": Water that entered the layer coming from other layers and transported via the plant hydraulic network (in mm) (and overall).
- "Snow": A data frame where the following variable (in columns) is given for each simulated day (in rows):
  - "SWE": Snow water equivalent (mm) of the snow pack.
- "Stand": A data frame where different variables (in columns) are given for each simulated day (in rows):
  - "LAI": LAI of the stand (including the herbaceous layer and live + dead leaves of woody plants) (in m2/m2).
  - "LAIherb": LAI of the herbaceous layer (in m2/m2).
  - "LAIlive": LAI of the woody plants assuming all leaves are unfolded (in m2/m2).
  - "LAIexpanded": LAI of the woody plants with leaves actually unfolded (in m2/m2).
  - "LAIdead": LAI of the woody plants corresponding to dead leaves (in m2/m2).
  - "Cm": Water retention capacity of the canopy (in mm) (accounting for leaf phenology).
  - "LgroundPAR": The percentage of PAR that reaches the ground (accounting for leaf phenology).
  - "LgroundSWR": The percentage of SWR that reaches the ground (accounting for leaf phenology).

- `"Plants"`: A list of daily results for plant cohorts (see below).
- `"subdaily"`: A list of objects of class `spwb_day`, one per day simulated (only if required in `control` parameters, see `defaultControl`).

When `transpirationMode = "Granier"`, element `"Plants"` is a list with the following subelements:

- `"LAI"`: A data frame with the daily leaf area index for each plant cohort.
- `"LAIlive"`: A data frame with the daily leaf area index for each plant cohort, assuming all leaves are unfolded (in m2/m2).
- `"FPAR"`: A data frame with the fraction of PAR at the canopy level of each plant cohort.
- `"AbsorbedSWRFraction"`: A data frame with the fraction of SWR absorbed by each plant cohort.
- `"Transpiration"`: A data frame with the amount of daily transpiration (in mm) for each plant cohort.
- `"GrossPhotosynthesis"`: A data frame with the amount of daily gross photosynthesis (in g C·m-2) for each plant cohort.
- `"PlantPsi"`: A data frame with the average daily water potential of each plant (in MPa).
- `"LeafPLC"`: A data frame with the average daily proportion of leaf conductance loss of each plant ([0-1]).
- `"StemPLC"`: A data frame with the average daily proportion of stem conductance loss of each plant ([0-1]).
- `"PlantWaterBalance"`: A data frame with the daily balance between transpiration and soil water extraction for each plant cohort.
- `"LeafRWC"`: A data frame with the average daily leaf relative water content of each plant (in percent).
- `"StemRWC"`: A data frame with the average daily stem relative water content of each plant (in percent).
- `"LFMC"`: A data frame with the daily live fuel moisture content (in percent of dry weight).
- `"PlantStress"`: A data frame with the amount of daily stress [0-1] suffered by each plant cohort (relative whole-plant conductance).

If `transpirationMode="Sperry"` or `transpirationMode="Sureau"`, element `"Plants"` is a list with the following subelements:

- `"LAI"`: A data frame with the daily leaf area index for each plant cohort.
- `"AbsorbedSWR"`: A data frame with the daily SWR absorbed by each plant cohort.
- `"NetLWR"`: A data frame with the daily net LWR by each plant cohort.
- `"Transpiration"`: A data frame with the amount of daily transpiration (in mm) for each plant cohorts.
- `"GrossPhotosynthesis"`: A data frame with the amount of daily gross photosynthesis (in g C·m-2) for each plant cohort.
- `"NetPhotosynthesis"`: A data frame with the amount of daily net photosynthesis (in g C·m-2) for each plant cohort.

- `"dEdP"`: A data frame with mean daily values of soil-plant conductance (derivative of the supply function) for each plant cohort.
- `"PlantWaterBalance"`: A data frame with the daily balance between transpiration and soil water extraction for each plant cohort.
- `"SunlitLeaves"` and `"ShadeLeaves"`: A list with daily results for sunlit and shade leaves:
  - `"PsiMin"`: A data frame with the minimum (midday) daily sunlit or shade leaf water potential (in MPa).
  - `"PsiMax"`: A data frame with the maximum (predawn) daily sunlit or shade leaf water potential (in MPa).
- `"LeafPsiMin"`: A data frame with the minimum (midday) daily (average) leaf water potential of each plant (in MPa).
- `"LeafPsiMax"`: A data frame with the maximum (predawn) daily (average) leaf water potential of each plant (in MPa).
- `"LeafRWC"`: A data frame with the average daily leaf relative water content of each plant (in percent).
- `"StemRWC"`: A data frame with the average daily stem relative water content of each plant (in percent).
- `"LFMC"`: A data frame with the daily live fuel moisture content (in percent of dry weight).
- `"StemPsi"`: A data frame with the minimum daily stem water potential of each plant (in MPa).
- `"LeafPLC"`: A data frame with the average daily proportion of leaf conductance loss of each plant ([0-1]).
- `"StemPLC"`: A data frame with the average daily proportion of stem conductance loss of each plant ([0-1]).
- `"RootPsi"`: A data frame with the minimum daily root water potential of each plant (in MPa).
- `"RhizoPsi"`: A list of data frames (one per plant cohort) with the minimum daily root water potential of each plant (in MPa).
- `"PlantStress"`: A data frame with the amount of daily stress [0-1] suffered by each plant cohort (relative whole-plant conductance).

Author(s)

- Miquel De Cáceres Ainsa, CREAF
- Nicolas Martin-StPaul, URFM-INRAE

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).

De Cáceres M, Mencuccini M, Martin-StPaul N, Limousin JM, Coll L, Poyatos R, Cabon A, Granda V, Forner A, Valladares F, Martínez-Vilalta J (2021) Unravelling the effect of species mixing on water use and drought stress in holm oak forests: a modelling approach. Agricultural and Forest Meteorology 296 (doi:10.1016/j.agrformet.2020.108233).

Granier A, Bréda N, Biron P, Villette S (1999) A lumped water balance model to evaluate duration and intensity of drought constraints in forest stands. Ecol Modell 116:269–283. https://doi.org/10.1016/S0304-3800(98)00205-1.

Ruffault J, Pimont F, Cochard H, Dupuy JL, Martin-StPaul N (2022) SurEau-Ecos v2.0: a trait-based plant hydraulics model for simulations of plant water status and drought-induced mortality at the ecosystem level. Geoscientific Model Development 15, 5593-5626 (doi:10.5194/gmd-15-5593-2022).

Sperry, J. S., M. D. Venturas, W. R. L. Anderegg, M. Mencuccini, D. S. Mackay, Y. Wang, and D. M. Love. 2017. Predicting stomatal responses to the environment from the optimization of photosynthetic gain and hydraulic cost. Plant Cell and Environment 40, 816-830 (doi: 10.1111/pce.12852).

See Also

`spwbInput`, `spwb_day`, `plot.spwb`, `extract`, `summary.spwb`, `forest`, `aspwb`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x1 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S1 <- spwb(x1, examplemeteo, latitude = 41.82592, elevation = 100)


#Switch to 'Sperry' transpiration mode
control <- defaultControl("Sperry")

#Initialize input
x2 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S2 <- spwb(x2, examplemeteo, latitude = 41.82592, elevation = 100)

#Switch to 'Sureau' transpiration mode
control <- defaultControl("Sureau")
```

```
#Initialize input
x3 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S3 <- spwb(x3, examplemeteo, latitude = 41.82592, elevation = 100)
```

stand_basalArea *Stand values*

Description

Functions to calculate stand attributes of a `forest` object.

Usage

```
stand_basalArea(x, minDBH = 7.5)

stand_foliarBiomass(x, SpParams, gdd = NA_real_)

stand_fuelLoading(x, SpParams, gdd = NA_real_, includeDead = TRUE)

stand_shrubVolume(x, SpParams)

stand_LAI(x, SpParams, gdd = NA_real_, bounded = TRUE)

stand_dominantTreeDiameter(x, minDBH = 7.5)

stand_treeDensity(x, minDBH = 7.5)

stand_meanTreeHeight(x, minDBH = 7.5)

stand_dominantTreeHeight(x, minDBH = 7.5)

stand_hartBeckingIndex(x, minDBH = 7.5)

stand_quadraticMeanTreeDiameter(x, minDBH = 7.5)

stand_dominantTreeSpecies(x, SpParams)
```

Arguments

`x`	An object of class `forest`.
`minDBH`	Minimum diameter at breast height (in cm) to include in estimation.
`SpParams`	A data frame with species parameters (see `SpParamsMED`).
`gdd`	Growth degree days (to account for leaf phenology effects).

`includeDead`	A flag to indicate that standing dead fuels (dead branches) are included.
`bounded`	A boolean flag to indicate that extreme values should be prevented (maximum tree LAI = 7 and maximum shrub LAI = 3)

Value

- `stand_basalArea`: Stand basal area (m2/ha).
- `stand_treeDensity`: Stand tree density (in ind/ha).
- `stand_dominantTreeDiameter`: Dominant tree diameter, i.e the average diameter of the 100 widest trees (in cm).
- `stand_quadraticMeanTreeDiameter`: Quadratic mean tree diameter, i.e. the diameter value corresponding to the current basal area and density.
- `stand_meanTreeHeight`: Mean tree height (in cm).
- `stand_dominantTreeHeight`: Dominant tree height, i.e the average height of the 100 tallest trees (in cm).
- `stand_hartBeckingIndex`: Hart-Becking index.
- `stand_foliarBiomass`: Standing biomass of leaves (in kg/m2).
- `stand_fuel`: Stand fine fuel load (in kg/m2).
- `stand_LAI`: Stand leaf area index (m2/m2).
- `stand_shrubVolume`: Stand shrub phytovolume (m3/m2).

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`forest`, `plant_basalArea`, `summary.forest`

Examples

```
#Default species parameterization
data(SpParamsMED)

#Load example plot
data(exampleforest)

#A short way to obtain total basal area
stand_basalArea(exampleforest)
```

summary.spwb *Summarize simulation results*

Description

Function `summary` summarizes the model's output in different temporal steps (i.e. weekly, annual, ...).

Usage

```
## S3 method for class 'spwb'
summary(
  object,
  freq = "years",
  output = "WaterBalance",
  FUN = sum,
  bySpecies = FALSE,
  months = NULL,
  ...
)

## S3 method for class 'pwb'
summary(
  object,
  freq = "years",
  output = "WaterBalance",
  FUN = sum,
  bySpecies = FALSE,
  months = NULL,
  ...
)

## S3 method for class 'growth'
summary(
  object,
  freq = "years",
  output = "WaterBalance",
  FUN = sum,
  bySpecies = FALSE,
  months = NULL,
  ...
)

## S3 method for class 'fordyn'
summary(
  object,
  freq = "years",
  output = "WaterBalance",
  FUN = sum,
  bySpecies = FALSE,
  months = NULL,
  ...
)
```

Arguments

`object`	An object of class `spwb`, `pwb`, `growth` or `fordyn`.
`freq`	Frequency of summary statistics (see `cut.Date`).
`output`	The data table to be summarized. Accepted values are the path to data tables in `object`, such as 'WaterBalance', 'Soil', 'Stand' or 'Plants$LAI'. It is also possible to use strings like 'Transpiration' and the function will interpret it as 'Plants$Transpiration'.
`FUN`	The function to summarize results (e.g., `sum`, `mean`, ...)
`bySpecies`	Allows aggregating output by species before calculating summaries (only has an effect with some values of `output`). Aggregation can involve a sum (as for plant lai or transpiration) or a LAI-weighted mean (as for plant stress or plant water potential).
`months`	A vector of month numbers (1 to 12) to subset the season where summaries apply.
`...`	Additional parameters for function `summary`.

Value

A matrix with dates as row names and the desired summaries in columns

Note

When applied to `fordyn` objects, the summary function can be used to gather the results of different yearly steps into a single table while keeping a daily resolution (i.e. using `freq = "days"`.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`spwb`, `pwb`, `growth`, `fordyn`, `plot.spwb`, `extract`

Examples

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

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

```
#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

#Call simulation function
S1<-spwb(x, examplemeteo, latitude = 41.82592, elevation = 100)

#Queries the tables in 'Soil'
names(S1$Soil)

#Monthly summary (averages) of soil relative water content
summary(S1, freq="months",FUN=mean, output="RWC")

#Queries the tables in 'Plants'
names(S1$Plants)

#Monthly summary (averages) of plant stress
summary(S1, freq="months",FUN=mean, output="PlantStress",
        bySpecies = TRUE)
```

sureau_ecos — *Sureau-ECOS inner functions for testing only*

Description

Function `initSureauNetworks` initializes hydraulic networks for all plant cohorts in x Function `semi_implicit_integration` updates water potentials and cavitation across the hydraulic network

Usage

```
initSureauNetworks(x)

semi_implicit_integration(
  network,
  dt,
  opt,
  stemCavitationRecovery = "annual",
  leafCavitationRecovery = "total"
)
```

Arguments

x	An object of class `spwbInput` or `growthInput` created using `transpirationMode = "Sureau"`.
network	A hydraulic network element of the list returned by `initSureauNetworks`
dt	Smallest time step (seconds)
opt	Option flag vector
stemCavitationRecovery, leafCavitationRecovery	A string indicating how refilling of embolized conduits is done: • "none" - no refilling. • "annual" - every first day of the year. • "rate" - following a rate of new sapwood formation. • "total" - instantaneous complete refilling.

Value

Function `initSureauNetworks` returns a vector of length equal to the number of cohorts. Each element is a list with Sureau-ECOS parameters. Function `semi_implicit_integration` does not return anything, but modifies input parameter `network`.

Author(s)

- Miquel De Cáceres Ainsa, CREAF
- Nicolas Martin-StPaul, URFM-INRAE

References

Ruffault J, Pimont F, Cochard H, Dupuy JL, Martin-StPaul N (2022) SurEau-Ecos v2.0: a trait-based plant hydraulics model for simulations of plant water status and drought-induced mortality at the ecosystem level. Geoscientific Model Development 15, 5593-5626 (doi:10.5194/gmd-15-5593-2022).

See Also

`spwb`

`transp_maximumTranspirationModel`
Maximum transpiration vs. LAI

Description

Builds a model of maximum transpiration (Tmax) over potential evapotranspiration (PET) for increasing leaf area index (LAI) values for each plant cohort.

Usage

```
transp_maximumTranspirationModel(
  x,
  meteo,
  latitude,
  elevation,
  slope,
  aspect,
  LAI_seq = c(0.1, 0.25, seq(0.5, 10, by = 0.5)),
  draw = TRUE
)
```

Arguments

`x`	An object of class `spwbInput`, built using the 'Sperry' transpiration mode.
`meteo`	A data frame with daily meteorological data series.
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`LAI_seq`	Sequence of stand LAI values to be tested.
`draw`	Logical flag to indicate plotting of results.

Details

This function performs a meta-modelling exercise using the Sperry transpiration model, with the aim to estimate coefficients for the equation used in the Granier transpiration model (Granier et al. 1999). The model to be fitted is: `y ~ a*LAI + b*LAI^2`, where `y` is the ratio between maximum transpiration (Tmax) and Penman's potential evapotranspiration (PET) and `LAI` is the stand LAI. Unlike the original equation of Granier et al. (1999), we fit a zero intercept model so that LAI = 0 translates into zero plant transpiration.

The function fits the model for each cohort separately, assuming it represents the whole stand. For each stand LAI value in the input sequence, the function uses simulations with Sperry transpiration and the input weather to estimate `y = Tmax/PET` as a function of stand's LAI (deciduous stands include leaf phenology). Once simulations have been conducted for each stand LAI value, the function fits a Generalized Linear Model with the above equation, assuming a Gamma distribution of residuals and an identity link.

The coefficients of the model can be used to parametrize Granier's transpiration, since coefficients `a` and `b` in the equation above correspond to parameters `Tmax_LAI` and `Tmax_LAIsq`, respectively (see `SpParamsMED`).

Value

Returns a list with as many elements as plant cohorts, each element being a `glm` model.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Granier A, Bréda N, Biron P, Villette S (1999) A lumped water balance model to evaluate duration and intensity of drought constraints in forest stands. Ecol Modell 116:269–283. https://doi.org/10.1016/S0304-3800(98)00205-1.

See Also

`spwb`, `transp_transpirationGranier`, `transp_transpirationSperry`, `SpParamsMED`

Examples

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

# Load example plot plant data
data(exampleforest)

# Load default species parameters
data(SpParamsMED)

# Define soil with default soil params
examplesoil <- defaultSoilParams(4)

# Initialize control parameters for 'Sperry' transpiration mode
control <- defaultControl(transpirationMode="Sperry")

# Initialize input
x2 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

# Estimate maximum transpiration ratio models for each cohort
# Weather is subset to speed-up results
m <- transp_maximumTranspirationModel(x2, examplemeteo[1:10,],
                                      41.82592, elevation = 100,
                                      slope = 0, aspect = 0)

# Inspect the model for first cohort
m[[1]]
```

`transp_stomatalregulation` *Stomatal regulation*

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_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,
  Gswmin,
  Gswmax
)

transp_stomatalRegulationPlot(
  x,
  meteo,
  day,
  timestep,
  latitude,
  elevation,
  slope = NA,
  aspect = NA,
  type = "E"
)
```

Arguments

`supplyFunction`	Water supply function (see `hydraulics_supplyFunctionNetwork`).
`photosynthesisFunction`	Function returned by `photo_photosynthesisFunction()`.
`Gswmin, Gswmax`	Minimum and maximum stomatal conductance to water vapour (mol·m-2·s-1).
`x`	An object of class `spwbInput` built using the 'Sperry' transpiration mode.
`meteo`	A data frame with daily meteorological data series (see `spwb`).
`day`	An integer to identify a day (a row) within `meteo`.
`timestep`	An integer between 1 and `ndailysteps` specified in `x` (see `defaultControl`).
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`type`	A string with plot type, either `"E"` (transpiration flow), `"Ag"` (gross photosynthesis), `"An"` (net photosynthesis), `"Gsw"` (stomatal conductance to water vapour), `"T"` (temperature) or `"VPD"` (leaf vapour pressure deficit).

Value

Function `transp_profitMaximization` returns 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).

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Sperry, J. S., M. D. Venturas, W. R. L. Anderegg, M. Mencuccini, D. S. Mackay, Y. Wang, and D. M. Love. 2017. Predicting stomatal responses to the environment from the optimization of photosynthetic gain and hydraulic cost. Plant Cell and Environment 40, 816-830 (doi: 10.1111/pce.12852).

See Also

`transp_transpirationSperry`, `hydraulics_supplyFunctionNetwork`, `biophysics_leafTemperature`, `photo_photosynthesis`, `spwb_day`, `plot.spwb_day`

Examples

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

#Load example plot plant data
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl(transpirationMode="Sperry")

#Initialize input
x2 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

# Stomatal VPD curve and chosen value for the 12th time step at day 100
transp_stomatalRegulationPlot(x2, examplemeteo, day=100, timestep = 12,
                              latitude = 41.82592, elevation = 100, type="VPD")
```

`transp_transpirationSperry` *Transpiration modes*

Description

High-level sub-models representing transpiration, plant hydraulics, photosynthesis and water relations within plants.

Usage

```
transp_transpirationSperry(
  x,
  meteo,
  day,
  latitude,
  elevation,
  slope,
  aspect,
  canopyEvaporation = 0,
  snowMelt = 0,
  soilEvaporation = 0,
  herbTranspiration = 0,
  stepFunctions = NA_integer_,
  modifyInput = TRUE
)

transp_transpirationSureau(
  x,
  meteo,
  day,
  latitude,
  elevation,
  slope,
  aspect,
  canopyEvaporation = 0,
  snowMelt = 0,
  soilEvaporation = 0,
  herbTranspiration = 0,
  modifyInput = TRUE
)

transp_transpirationGranier(
  x,
  meteo,
  day,
  latitude,
  elevation,
  slope,
  aspect,
  modifyInput = TRUE
)
```

Arguments

`x`	An object of class `spwbInput` or `growthInput`, built using the 'Granier', 'Sperry' or 'Sureau' transpiration modes.
`meteo`	A data frame with daily meteorological data series (see `spwb`).
`day`	An integer to identify a day (row) within the `meteo` data frame.
`latitude`	Latitude (in degrees).
`elevation, slope, aspect`	Elevation above sea level (in m), slope (in degrees) and aspect (in degrees from North).
`canopyEvaporation`	Canopy evaporation (from interception) for `day` (mm).
`snowMelt`	Snow melt values for `day` (mm).
`soilEvaporation`	Bare soil evaporation for `day` (mm).
`herbTranspiration`	Transpiration of herbaceous plants for `day` (mm).
`stepFunctions`	An integer to indicate a simulation step for which photosynthesis and profit maximization functions are desired.
`modifyInput`	Boolean flag to indicate that the input `x` object is allowed to be modified during the simulation.

Details

Three sub-models are available:

- Sub-model in function `transp_transpirationGranier` was described in De Cáceres et al. (2015), and implements an approach originally described in Granier et al. (1999).
- Sub-model in function `transp_transpirationSperry` was described in De Cáceres et al. (2021), and implements a modelling approach originally described in Sperry et al. (2017).
- Sub-model in function `transp_transpirationSureau` was described for SurEau-Ecos v2.0 model in Ruffault et al. (2022).

Value

A list with the following elements:

- `"cohorts"`: A data frame with cohort information, copied from `spwbInput`.
- `"Stand"`: A vector of stand-level variables.

- "Plants": A data frame of results for each plant cohort. When using transp_transpirationGranier, element "Plants" includes:
  - "LAI": Leaf area index of the plant cohort.
  - "LAIlive": Leaf area index of the plant cohort, assuming all leaves are unfolded.
  - "AbsorbedSWRFraction": Fraction of SWR absorbed by each cohort.
  - "Transpiration": Transpirated water (in mm) corresponding to each cohort.
  - "GrossPhotosynthesis": Gross photosynthesis (in gC/m2) 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 transp_transpirationSperry or transp_transpirationSureau, element "Plants" includes:
  - "LAI": Leaf area index of the plant cohort.
  - "LAIlive": Leaf area index of the plant cohort, assuming all leaves are unfolded.
  - "Extraction": Water extracted from the soil (in mm) for each cohort.
  - "Transpiration": Transpirated water (in mm) corresponding to each cohort.
  - "GrossPhotosynthesis": Gross photosynthesis (in gC/m2) corresponding to each cohort.
  - "NetPhotosynthesis": Net photosynthesis (in gC/m2) corresponding to each cohort.
  - "RootPsi": Minimum water potential (in MPa) at the root collar.
  - "StemPsi": Minimum water potential (in MPa) at the stem.
  - "StemPLC": Proportion of conductance loss in stem.
  - "LeafPsiMin": Minimum (predawn) water potential (in MPa) at the leaf (representing an average leaf).
  - "LeafPsiMax": Maximum (midday) water potential (in MPa) at the leaf (representing an average leaf).
  - "LeafPsiMin_SL": Minimum (predawn) water potential (in MPa) at sunlit leaves.
  - "LeafPsiMax_SL": Maximum (midday) water potential (in MPa) at sunlit leaves.
  - "LeafPsiMin_SH": Minimum (predawn) water potential (in MPa) at shade leaves.
  - "LeafPsiMax_SH": Maximum (midday) water potential (in MPa) at shade leaves.
  - "dEdP": Overall soil-plant conductance (derivative of the supply function).
  - "DDS": Daily drought stress [0-1] (relative whole-plant conductance).
  - "StemRWC": Relative water content of stem tissue (including symplasm and apoplasm).
  - "LeafRWC": Relative water content of leaf tissue (including symplasm and apoplasm).
  - "LFMC": Live fuel moisture content (in percent of dry weight).
  - "WaterBalance": Plant water balance (extraction - transpiration).
- "Extraction": A data frame with mm of water extracted from each soil layer (in columns) by each cohort (in rows).

  The remaining items are only given by transp_transpirationSperry or transp_transpirationSureau:
- "EnergyBalance": A list with the following elements:
  - "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.
- "RhizoPsi": Minimum water potential (in MPa) inside roots, after crossing rhizosphere, per cohort and soil layer.
- "Sunlitleaves" and "ShadeLeaves": Data frames for sunlit leaves and shade leaves and the following columns per cohort:
  - "LAI": Cumulative leaf area index of sunlit/shade leaves.
  - "Vmax298": Average maximum carboxilation rate for sunlit/shade leaves.
  - "Jmax298": Average maximum electron transport rate for sunlit/shade leaves.
- "ExtractionInst": Water extracted by each plant cohort during each time step.
- "PlantsInst": A list with instantaneous (per time step) results for each plant cohort:
  - "E": A data frame with the cumulative transpiration (mm) for each plant cohort during each time step.
  - "Ag": A data frame with the cumulative gross photosynthesis (gC/m2) for each plant cohort during each time step.
  - "An": A data frame with the cumulative net photosynthesis (gC/m2) for each plant cohort during each time step.
  - "Sunlitleaves" and "ShadeLeaves": Lists with instantaneous (for each time step) results for sunlit leaves and shade leaves and the following items:
    * "Abs_SWR": A data frame with instantaneous absorbed short-wave radiation (SWR).
    * "Net_LWR": A data frame with instantaneous net long-wave radiation (LWR).
    * "An": A data frame with instantaneous net photosynthesis (in micromol/m2/s).
    * "Ci": A data frame with instantaneous intercellular CO2 concentration (in ppm).
    * "GW": A data frame with instantaneous stomatal conductance (in mol/m2/s).
    * "VPD": A data frame with instantaneous vapour pressure deficit (in kPa).
    * "Temp": A data frame with leaf temperature (in degrees Celsius).
    * "Psi": A data frame with leaf water potential (in MPa).
  - "dEdP": A data frame with the slope of the plant supply function (an estimation of whole-plant conductance).
  - "RootPsi": A data frame with root crown water potential (in MPa) for each plant cohort during each time step.
  - "StemPsi": A data frame with stem water potential (in MPa) for each plant cohort during each time step.
  - "LeafPsi": A data frame with leaf (average) water potential (in MPa) for each plant cohort during each time step.
  - "StemPLC": A data frame with the proportion loss of conductance [0-1] for each plant cohort during each time step.
  - "StemRWC": A data frame with the (average) relative water content of stem tissue [0-1] for each plant cohort during each time step.
  - "LeafRWC": A data frame with the relative water content of leaf tissue [0-1] for each plant cohort during each time step.

  - "StemSympRWC": A data frame with the (average) relative water content of symplastic stem tissue [0-1] for each plant cohort during each time step.
  - "LeafSympRWC": A data frame with the relative water content of symplastic leaf tissue [0-1] for each plant cohort during each time step.
  - "PWB": A data frame with plant water balance (extraction - transpiration).
- "LightExtinction": A list of information regarding radiation balance through the canopy, as returned by function `light_instantaneousLightExtinctionAbsortion`.
- "CanopyTurbulence": Canopy turbulence (see `wind_canopyTurbulence`).
- "SupplyFunctions": If `stepFunctions` is not missing, a list of supply functions, photosynthesis functions and profit maximization functions.

Author(s)

- Miquel De Cáceres Ainsa, CREAF
- Nicolas Martin-StPaul, URFM-INRAE

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).

De Cáceres M, Mencuccini M, Martin-StPaul N, Limousin JM, Coll L, Poyatos R, Cabon A, Granda V, Forner A, Valladares F, Martínez-Vilalta J (2021) Unravelling the effect of species mixing on water use and drought stress in holm oak forests: a modelling approach. Agricultural and Forest Meteorology 296 (doi:10.1016/j.agrformet.2020.108233).

Granier A, Bréda N, Biron P, Villette S (1999) A lumped water balance model to evaluate duration and intensity of drought constraints in forest stands. Ecol Modell 116:269–283. https://doi.org/10.1016/S0304-3800(98)00205-1.

Ruffault J, Pimont F, Cochard H, Dupuy JL, Martin-StPaul N (2022) SurEau-Ecos v2.0: a trait-based plant hydraulics model for simulations of plant water status and drought-induced mortality at the ecosystem level. Geoscientific Model Development 15, 5593-5626 (doi:10.5194/gmd-15-5593-2022).

Sperry, J. S., M. D. Venturas, W. R. L. Anderegg, M. Mencuccini, D. S. Mackay, Y. Wang, and D. M. Love. 2017. Predicting stomatal responses to the environment from the optimization of photosynthetic gain and hydraulic cost. Plant Cell and Environment 40, 816-830 (doi: 10.1111/pce.12852).

See Also

`spwb_day`, `plot.spwb_day`

Examples

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

#Load example plot plant data
```

```
data(exampleforest)

#Default species parameterization
data(SpParamsMED)

#Define soil with default soil params (4 layers)
examplesoil <- defaultSoilParams(4)

#Initialize control parameters
control <- defaultControl("Granier")

#Initialize input
x1 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

# Transpiration according to Granier's model, plant water potential
# and plant stress for a given day
t1 <- transp_transpirationGranier(x1, examplemeteo, 1,
                              latitude = 41.82592, elevation = 100, slope = 0, aspect = 0,
                                  modifyInput = FALSE)

#Switch to 'Sperry' transpiration mode
control <- defaultControl("Sperry")

#Initialize input
x2 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

# Transpiration according to Sperry's model
t2 <- transp_transpirationSperry(x2, examplemeteo, 1,
                             latitude = 41.82592, elevation = 100, slope = 0, aspect = 0,
                                modifyInput = FALSE)

#Switch to 'Sureau' transpiration mode
control <- defaultControl("Sureau")

#Initialize input
x3 <- spwbInput(exampleforest,examplesoil, SpParamsMED, control)

# Transpiration according to Sureau model
t3 <- transp_transpirationSureau(x3, examplemeteo, 1,
                              latitude = 41.82592, elevation = 100, slope = 0, aspect = 0,
                                   modifyInput = FALSE)
```

tree2forest *Single-cohort forests*

Description

Creates a forest object with a single plant cohort

Usage

```
tree2forest(
  Species,
  Height,
  LAI = NA,
  N = NA,
  DBH = NA,
  Z50 = NA,
  Z95 = NA,
  CrownRatio = NA,
  FoliarBiomass = NA,
  FuelLoading = NA
)

shrub2forest(
  Species,
  Height,
  LAI = NA,
  Cover = NA,
  Z50 = NA,
  Z95 = NA,
  CrownRatio = NA,
  FoliarBiomass = NA,
  FuelLoading = NA
)
```

Arguments

`Species`	String with species (taxon) name or a non-negative integer for species identity (i.e., 0,1,2,...) matching SpParams.
`Height`	Plant height (cm).
`LAI`	Leaf area index (m2/m2)
`N`	Tree density (ind/ha)
`DBH`	Tree DBH (cm).
`Z50`	Depth (in mm) corresponding to 50% of fine roots.
`Z95`	Depth (in mm) corresponding to 95% of fine roots.
`CrownRatio`	Crown ratio (fraction of total height)
`FoliarBiomass`	Standing dry biomass of leaves (kg/m2)
`FuelLoading`	Fine fuel loading (kg/m2)
`Cover`	Percent cover

Value

An object of class `forest`

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`forest`, `forest_mergeTrees`, `plot.forest`

Examples

```
oak_forest <-tree2forest("Quercus ilex", Height= 200, LAI = 2)
```

---

vprofile_leafAreaDensity *Vertical profiles*

Description

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

Usage

```
vprofile_leafAreaDensity(
  x,
  SpParams = NULL,
  z = NULL,
  gdd = NA,
  byCohorts = FALSE,
  bySpecies = FALSE,
  includeHerbs = FALSE,
  draw = TRUE,
  xlim = NULL
)

vprofile_rootDistribution(
  x,
  SpParams,
  d = NULL,
  bySpecies = FALSE,
  draw = TRUE,
  xlim = NULL
)

vprofile_fuelBulkDensity(
  x,
  SpParams,
  z = NULL,
  gdd = NA,
  draw = TRUE,
  xlim = NULL
)

vprofile_PARExtinction(
  x,
  SpParams,
  z = NULL,
  gdd = NA,
  includeHerbs = FALSE,
  draw = TRUE,
  xlim = c(0, 100)
)

vprofile_SWRExtinction(
  x,
  SpParams,
  z = NULL,
  gdd = NA,
  includeHerbs = FALSE,
  draw = TRUE,
  xlim = c(0, 100)
)

vprofile_windExtinction(
  x,
  SpParams,
  u = 1,
  windMeasurementHeight = 200,
  boundaryLayerSize = 2000,
  target = "windspeed",
  z = NULL,
  gdd = NA,
  includeHerbs = FALSE,
  draw = TRUE,
  xlim = NULL
)
```

Arguments

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

`bySpecies`	Aggregate cohort profiles by species.
`includeHerbs`	Include herbaceous layer in the profile.
`draw`	Logical flag to indicate that a plot is desired.
`xlim`	Limits of the x-axis.
`d`	A numeric vector with soil layer widths.
`u`	The value of measured wind speed (in m/s).
`windMeasurementHeight`	Height corresponding to wind measurement (in cm over the canopy).
`boundaryLayerSize`	Size of the boundary layer (in cm) over the canopy.
`target`	Wind property to draw, either "windspeed", "kineticenergy" (turbulent kinetic energy) or "stress" (Reynold's stress).

Value

If `draw = FALSE`, the functions return 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.

If `draw = TRUE` the functions return a ggplot object, instead.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`forest`, `plot.forest`, `wind_canopyTurbulence`

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)
```

waterUseEfficiency *Water use efficiency*

Description

Calculates plant water use efficiency (WUE), at different temporal scales, from simulation results.

Usage

```
waterUseEfficiency(
  x,
  type = "Plant Ag/E",
  leaves = "average",
  freq = "days",
  draw = TRUE,
  ylim = NULL
)
```

Arguments

`x`	An object of class `spwb`, `pwb`, `growth` or `fordyn`.
`type`	A string to indicate the scale of WUE calculation. Either: • `"Leaf iWUE"`: Leaf intrinsic WUE, i.e. instantaneous ratio between photosynthesis and stomatal conductance (only for simulations with `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"` and `subdailyResults = TRUE`). • `"Leaf Ci"`: Leaf intercellular CO2 concentration (only for simulations with `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"` and `subdailyResults = TRUE`). • `"Plant An/E"`: Plant (cohort) net photosynthesis over plant transpiration (only for simulations with `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`) • `"Stand An/E"`: Stand net photosynthesis over stand transpiration (only for simulations with `transpirationMode = "Sperry"` or `transpirationMode = "Sureau"`) • `"Plant Ag/E"`: Plant (cohort) gross photosynthesis over plant transpiration • `"Stand Ag/E"`: Stand gross photosynthesis over stand transpiration
`leaves`	Either `"sunlit"`, `"shade"` or `"average"`. Refers to the WUE of different leaf types or the average (with weights according to the LAI of sunlit and shade leaves). Only relevant for `type = "iWUE"`.

`freq`	Frequency of summary statistics (see `cut.Date`).
`draw`	A boolean flag to indicate that a plot should be returned.
`ylim`	Range of values for y.

Details

Temporal aggregation of WUE values is done differently depending on the value of `type`. For `type = "Plant Ag/E"`, `type = "Stand Ag/E"`, `type = "Plant An/E"` and `type = "Stand An/E"` sums or daily photosynthesis and transpiration are first calculated at the desired temporal scale and the ratio is calculated afterwards. For `type = "Leaf iWUE"` intrinsic WUE values are first calculated at the daily scale (as averages of instantaneous An/gs ratios weighted by An) and then they are aggregated to the desired scale by calculating weighted averages, where weights are given by daily photosynthesis.

Value

If `draw=TRUE` a plot is returned. Otherwise, the function returns a matrix with WUE values, where rows are dates (at the desired temporal scale), and columns are plant cohorts. In the case of `type = "Plant Ag/E"`, `type = "Stand Ag/E"`, `type = "Plant An/E"` and `type = "Stand An/E"` values are in gC/L. In the case of `type = "Leaf iWUE"` values are in micromol of carbon per mmol of water.

Author(s)

Miquel De Cáceres Ainsa, CREAF

See Also

`droughtStress`

`wind` *Models for canopy turbulence*

Description

Models for canopy turbulence by Katul et al (2004).

Usage

```
wind_canopyTurbulenceModel(zm, Cx, hm, d0, z0, model = "k-epsilon")

wind_canopyTurbulence(
  zmid,
  LAD,
  canopyHeight,
  u,
  windMeasurementHeight = 200,
  model = "k-epsilon"
)
```

Arguments

zm	A numeric vector with height values (m).
Cx	Effective drag = Cd x leaf area density.
hm	Canopy height (m).
d0	Zero displacement height (m).
z0	Momentum roughness height (m).
model	Closure model.
zmid	A numeric vector of mid-point heights (in cm) for canopy layers.
LAD	A numeric vector of leaf area density values (m3/m2).
canopyHeight	Canopy height (in cm).
u	Measured wind speed (m/s).
windMeasurementHeight	Height of wind speed measurement with respect to canopy height (cm).

Details

Implementation in Rcpp of the K-epsilon canopy turbulence models by Katul et al (2004) originally in Matlab code (https://nicholas.duke.edu/people/faculty/katul/k_epsilon_model.htm).

Value

Function `wind_canopyTurbulenceModel` returns a data frame of vertical profiles for variables:

- `z1`: Height values.
- `U1`: U/u*, where U is mean velocity and u* is friction velocity.
- `dU1`: dUdz/u*, where dUdz is mean velocity gradient and u* is friction velocity.
- `epsilon1`: epsilon/(u^3/h) where epsilon is the turbulent kinetic dissipation rate, u* is friction velocity and h is canopy height.
- `k1`: k/(u*^2), where k is the turbulent kinetic energy and u* is friction velocity.
- `uw1`: <uw>/(u*^2), where <uw> is the Reynolds stress and u* is friction velocity.
- `Lmix1`: Mixing length.

Function `wind_canopyTurbulence` returns a data frame of vertical profiles for transformed variables:

- `zmid`: Input mid-point heights (in cm) for canopy layers.
- `u`: Wind speed (m/s).
- `du`: Mean velocity gradient (1/s).
- `epsilon`: Turbulent kinetic dissipation rate.
- `k`: Turbulent kinetic energy.
- `uw`: Reynolds stress.

Author(s)

Miquel De Cáceres Ainsa, CREAF

References

Katul GG, Mahrt L, Poggi D, Sanz C (2004) One- and two-equation models for canopy turbulence. Boundary-Layer Meteorol 113:81–109. https://doi.org/10.1023/B:BOUN.0000037333.48760.e5

See Also

vprofile_windExtinction

Examples

```
#Default species parameterization
data(SpParamsMED)

#Load example plot plant data
data(exampleforest)

#Canopy height (in m)
h= max(exampleforest$treeData$Height/100)
d0 = 0.67*h
z0 = 0.08*h

#Height values (cm)
z = seq(50,1000, by=50)
zm = z/100 # (in m)

# Leaf area density
lad = vprofile_leafAreaDensity(exampleforest, SpParamsMED, draw = FALSE,
                               z = c(0,z))

# Effective drag
Cd = 0.2
Cx = Cd*lad

# canopy turbulence model
wind_canopyTurbulenceModel(zm, Cx,h,d0,z0)
```

`woodformation` *Wood formation*

Description

Functions to initialize and expand a ring of tracheids to simulate secondary growth.

Usage

```
woodformation_initRing()

woodformation_temperatureEffect(
  Tc,
  Y_T = 5,
  DHa = 87500,
  DSd = 1090,
  DHd = 333000
)

woodformation_relativeExpansionRate(psi, Tc, pi, phi, Y_P, Y_T)

woodformation_growRing(
  ring,
  psi,
  Tc,
  Nc = 8.85,
  phi0 = 0.13,
  pi0 = -0.8,
  CRD0 = 8.3,
  Y_P = 0.05,
  Y_T = 5,
  h = 0.043 * 1.8,
  s = 1.8
)

woodformation_relativeGrowthRate(dbh1, dbh2, yeardiff, lower = -2, upper = 8)
```

Arguments

`Tc`	Temperature in Celsius.
`Y_T`	Temperature yield threshold (in Celsius)
`DHa, DSd, DHd`	Enthalpy of activation, enthalpy difference and entropy difference between the catalytically active and inactive states of the enzymatic system (Parent et al. 2010).
`psi`	Water potential (in MPa).
`pi`	Osmotic potential (in MPa)
`phi`	Cell extensibility (in MPa-1 day-1)
`Y_P`	Turgor pressure yield threshold (in MPa)
`ring`	An object of class `ring` returned by function `woodformation_initRing`.
`Nc`	Number of active cells in the cambium.
`phi0`	Initial value of cell extensibility (in MPa-1 day-1)
`pi0`	Initial value of cell osmotic potential (in MPa)
`CRD0`	Initial value of cell radial diameter

`h`	Cell wall hardening coefficient (in day-1)
`s`	Cell wall softening coefficient (unitless)
`dbh1, dbh2`	Initial and final diameter at breast height.
`yeardiff`	Interval between dbh measurements, in years.
`lower, upper`	Lower and upper bounds for root finding.

Value

Function `woodformation_initRing()` returns a list of class 'ring', that is a list containing a data frame `cells` and two vectors: `P` and `SA`. Dataframe `cells` contains the columns "formation_date", "phi", "pi" and "CRD" and as many rows as dates processed. Vectors `P` and `SA` contain, respectively, the number of cells produced and the sapwood area corresponding to the ring of cells (assuming a tangencial radius of 20 micrometers).

Function `woodformation_growRing()` modifies the input 'ring' object according to the environmental conditions given as input.

Function `woodformation_relativeExpansionRate()` returns a numeric scalar with the relative expansion rate.

Function `woodformation_temperatureEffect()` returns a scalar between 0 and 1 reflecting the temperature effect on tissue formation rate.

Function `woodformation_relativeGrowthRate` returns the annual growth rate, relative to cambium perimeter, estimated from initial and final diameter values.

Note

Code modified from package xylomod by Antoine Cabon, available at GitHub

Author(s)

Antoine Cabon, CTFC

Miquel De Cáceres Ainsa, CREAF

References

Cabon A, Fernández-de-Uña L, Gea-Izquierdo G, Meinzer FC, Woodruff DR, Martínez-Vilalta J, De Cáceres M. 2020a. Water potential control of turgor-driven tracheid enlargement in Scots pine at its xeric distribution edge. New Phytologist 225: 209–221.

Cabon A, Peters RL, Fonti P, Martínez-Vilalta J, De Cáceres M. 2020b. Temperature and water potential co-limit stem cambial activity along a steep elevational gradient. New Phytologist: nph.16456.

Parent, B., O. Turc, Y. Gibon, M. Stitt, and F. Tardieu. 2010. Modelling temperature-compensated physiological rates, based on the co-ordination of responses to temperature of developmental processes. Journal of Experimental Botany 61:2057–2069.

See Also

`growth`

Index