Package: LWFBrook90R (via r-universe)

October 25, 2026

Title Simulate Evapotranspiration and Soil Moisture with the SVAT Model LWF-Brook90

Version 0.6.3

Description Provides a flexible and easy-to use interface for the soil vegetation atmosphere transport (SVAT) model LWF-BROOK90, written in Fortran. The model simulates daily transpiration, interception, soil and snow evaporation, streamflow and soil water fluxes through a soil profile covered with vegetation, as described in Hammel & Kennel (2001, ISBN:978-3-933506-16-0) and Federer et al. (2003) <doi:10.1175/1525-7541(2003)004%3C1276:SOAETS%3E2.0.CO;2>. A set of high-level functions for model set up, execution and parallelization provides easy access to plot-level SVAT simulations, as well as multi-run and large-scale applications.

License GPL-3

Encoding UTF-8

URL https://pschmidtwalter.github.io/LWFBrook90R/, https://github.com/pschmidtwalter/LWFBrook90R

BugReports https://github.com/pschmidtwalter/LWFBrook90R/issues

NeedsCompilation yes

Depends R (>= 3.4.0)

Imports methods, data.table (>= 1.10.4), vegperiod (>= 0.3.0), foreach (>= 1.5.0), iterators (>= 1.0.12), doFuture (>= 0.10.0), future (>= 1.19.0), parallelly (>= 1.30.0), progressr (>= 0.6.0)

Suggests knitr, rmarkdown, roxygen2, testthat

LazyData true

RoxygenNote 7.3.3

VignetteBuilder knitr

Author Paul Schmidt-Walter [aut, cre] (ORCID: <https://orcid.org/0000-0003-2699-0893>), Volodymyr Trotsiuk [aut] (ORCID: <https://orcid.org/0000-0002-8363-656X>), Klaus

Hammel [aut], Martin Kennel [aut], Anthony Federer [aut], Tobias Hohenbrink [aut] (ORCID: <https://orcid.org/0000-0002-5227-0171>), Gisbert Hetkamp [aut], Michael Köhler [aut] (ORCID: <https://orcid.org/0000-0002-7593-5471>), Robert Nuske [ctb] (ORCID: <https://orcid.org/0000-0001-9773-2061>), Bavarian State Institute of Forestry (LWF) [cph, fnd], Northwest German Forest Research Institute (NW-FVA) [cph, fnd]

Maintainer Paul Schmidt-Walter <paulsw@posteo.de>

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

Date/Publication 2025-11-18 08:30:17 UTC

RemoteUrl https://github.com/cran/LWFBrook90R

RemoteRef HEAD

RemoteSha 106ff43d57170a1e5277f707df92fe25b687d790

Contents

approx_standprop . . . 3
calc_globrad . . . 5
calc_vegperiod . . . 6
correct_prec . . . 7
extract_layer_output . . . 8
make_rootden . . . 10
make_seasLAI . . . 11
make_standprop . . . 14
parms_to_rbrook90 . . . 15
plant_b90 . . . 15
plant_coupmodel . . . 16
plant_linear . . . 17
process_outputs_LWFB90 . . . 18
ptfs . . . 19
r_lwfbrook90 . . . 21
replace_vecelements . . . 22
run_LWFB90 . . . 23
run_multi_LWFB90 . . . 27
run_multisite_LWFB90 . . . 29
set_optionsLWFB90 . . . 32
set_outputLWFB90 . . . 33
set_paramLWFB90 . . . 34
slb1_meteo . . . 37
slb1_prec2013_hh . . . 38
slb1_soil . . . 39
slb1_standprop . . . 39
soil_to_param . . . 40
standprop_yearly_to_param . . . 41

Index 42

`approx_standprop` *Create a daily sequence of stand properties from parameters using interpolation*

Description

Uses yearly values of inter-annual vegetation development values (e.g. sai, height, densef, age) and interpolates them to a daily sequence.

Usage

```
approx_standprop(
  x_yrs,
  y,
  y_ini = NULL,
  xout_yrs = x_yrs,
  use_growthperiod = FALSE,
  startdoy = 121,
  enddoy = 279,
  approx.method = "constant",
  return_xout = FALSE
)
```

Arguments

`x_yrs`	A sequence of years or a single year.
`y`	Vector of the same length as `x_yrs`. If `approx.method = 'linear'`, the values are interpreted to be valid at the end of the respective year in `x_yrs`
`y_ini`	Initial value used as a starting point for linear interpolation. Interpreted to be valid at the 1st of January of the first year in `x_yrs`. Ignored if `approx.method = 'constant'`.
`xout_yrs`	Vector of years for which output is generated. May be longer or shorter than `x_yrs`. For years outside `x_yrs`, the value of the closest data extreme is returned.
`use_growthperiod`	Logical: Use startdoy and enddoy for linear interpolation? If TRUE, yearly changes take place between startdoy and enddoy, othe wise from end of year to end of the year after.
`startdoy`	A single value or vector of the same length as `x_yrs`, with the day of year when growth begins.
`enddoy`	A single value or vector of the same length as `x_yrs`, with the day of year when growth cessates.
`approx.method`	Name of interpolation method ('constant' or 'linear').
`return_xout`	Logical: If true, daily values of `y` and a date vector are returned in a data.frame.

Details

For `approx.method = 'constant'`, the value of `y` is returned for the whole respective year in `x_yrs`, which results in a yearly changing step function. If `approx.method = 'linear'`, the values of `y` are interpolated between the years in `x_yrs`, and interpreted to be reached at the 31st of December of the respective `x_yrs`. In this case, `y_ini` is required as an initial value, from which the sequence is interpolated to the first value of `y`. The linear changes are either accomplished between 31st to 31st of December of the years in `x_yrs`, or during the growing season only (`use_growingperiod = TRUE`).

Value

A vector of interpolated daily values

Examples

```
years <- 2002:2004
height_yearly <- c(20.2,20.8,21.3)

# constant 'interpolation'
height_c <- approx_standprop(x_yrs = years,
                             y = height_yearly)

# linear interpolation
height_ini <- 19.1
height_l <- approx_standprop(x_yrs=years,
                             y = height_yearly,
                             y_ini = height_ini,
                             approx.method = 'linear')

# use growthperiod
height_l_gp <- approx_standprop(x_yrs = years,
                                y = height_yearly,
                                y_ini = height_ini,
                                use_growthperiod = TRUE,
                                startdoy = 121,
                                enddoy = 279,
                                approx.method = 'linear')

dates <- seq.Date(from = as.Date(paste0(min(years),"-01-01")),
                  to = as.Date(paste0(max(years),"-12-31")),
                  by = "day")
plot(dates, height_c,
     type = "l", lwd = 2, col = "black",
     ylim = c(19,22), ylab = "height [m]", xlab = "", xpd = TRUE)
lines(dates, height_l,
      col = "blue", lwd = 2)
lines(dates, height_l_gp,
      col = "green", lwd = 2)
legend("topleft", legend = c("'constant'", "'linear'",
                             "'linear', 'use_growthperiod'"),
       col  = c("black", "blue", "green"),  lwd = 2, pch = NULL,
```

```
        bty = "n")
```

`calc_globrad` *Calculate global solar radiation from sunshine duration hours*

Description

Uses functions taken from the 'sirad' package to determine astronomical daylength and extraterrestrial radiation, from which global radiation is calculated using the Angström-formula.

Usage

```
calc_globrad(dates, sunhours, lat, a0 = 0.25, b0 = 0.5, full_output = FALSE)
```

Arguments

`dates`	Date vector
`sunhours`	Vector of sunshine duration hours, same length as dates.
`lat`	Latitude in decimal degrees.
`a0`	Angström parameter a, defaults to 0.25.
`b0`	Angström parameter b, defaults to 0.5.
`full_output`	Return extraterrestrial radiation and daylength along with global radiation?

Value

A sequence of global radiation in MJ/(m² d) with the length of dates, or (if `full_output = TRUE`) a `data.frame` holding day of year, dates, sunhours, daylength, and extraterrestrial and calculated global solar radiation. A warning is generated if some sunshine duration hours are higher than the expected daylength at the specified latitude.

Examples

```
dates <- seq.Date(as.Date("2002-01-01"), as.Date("2003-12-31"), by = 'day')
calc_globrad(dates, sunhours = runif(365, 0, 7),lat = 52.8)
calc_globrad(dates, sunhours = runif(365, 0, 7),lat = 52.8, full_output = TRUE)
```

`calc_vegperiod` *Calculate the dates of budburst and beginning of leaf fall*

Description

Wrapper for `vegperiod`

Usage

```
calc_vegperiod(
  budburst_method,
  leaffall_method,
  dates = NULL,
  tavg = NULL,
  out_yrs = NULL,
  budburstdoy.fixed = 121,
  leaffalldoy.fixed = 279,
  ...
)
```

Arguments

`budburst_method`	name of model for estimating budburst day of year. Either 'fixed' or one of the values accepted by the 'start.method'-argument of the function `vegperiod`.
`leaffall_method`	name of model for estimating day of year when leaffall begin. Either 'fixed' or one of the values accepted by the 'end.method'-argument of the function `vegperiod`.
`dates`	date vector passed to `vegperiod`, ignored if both leaffall_method and budburst_method = 'fixed'
`tavg`	vector of daily mean air temperature (deg C) passed to `vegperiod`, ignored if `leaffall_method = 'fixed'` and `budburst_method = 'fixed'`.
`out_yrs`	integer vector of the years to be returned. If not specified, values for the years in `dates` will be returned.
`budburstdoy.fixed`	vector of values to be returned if `budburst_method = 'fixed'`.
`leaffalldoy.fixed`	vector of values to be returned if `leaffall_method = 'fixed'`.
`...`	additional argument passed to `vegperiod`.

Value

a data.frame with columns `year`, `start`, `end`. If `budburst_method = 'fixed'` or `leaffall_method = 'fixed'`, `start` and `end` contain the values specified in `budburstdoy.fixed` and `leaffalldoy.fixed` respectively.

Examples

```
# fixed budburst and leaffall doy
calc_vegperiod(out_yrs = 2001:2010,
               budburst_method = "fixed",
               leaffall_method = "fixed",
               budburstdoy.fixed = floor(runif(10, 120,130)),
               leaffalldoy.fixed = floor(runif(2, 260,280)))

# dynamic budburst and leaffall using air temperature
data(slb1_meteo)

calc_vegperiod(budburst_method = "Menzel",
               leaffall_method = "fixed",
               leaffalldoy.fixed = 280,
               dates = slb1_meteo$dates,
               tavg = slb1_meteo$tmean,
               species = "Fagus sylvatica",
               est.prev = 3)

calc_vegperiod(budburst_method = "Menzel",
               leaffall_method = "ETCCDI",
               dates = slb1_meteo$dates,
               tavg = slb1_meteo$tmean,
               species = "Quercus robur",
               est.prev = 3)
```

`correct_prec` *Correct rain gauge precipitation data for wind and evaporation errors after Richter (1995)*

Description

Correct rain gauge precipitation data for wind and evaporation errors after Richter (1995)

Usage

```
correct_prec(month, tavg, prec, station.exposure = "mg", full_output = FALSE)
```

Arguments

`month`	Vector of months.
`tavg`	Vector of air temperature values (deg C). Same length as `month`.
`prec`	Vector of measured rainfall vales (mm). Same length as `month`.
`station.exposure`	Situation of the weather station where prec was measured: one of 'frei', 'lg', 'mg', 'sg' (corresponding to full exposure, low protected, moderate protected, strong protected situation).
`full_output`	Logical wether to return the full data set additionally including input data, correction coefficients.

Value

A vector of corrected rainfall data, or (if `full_output == TRUE`) a data.table containing the input objects, the month, the precipitation type ('N4So': liquid rain, summer; 'N4Wi' liquid rain, winter; 'N8' = sleet, 'N7' = snow), correction coefficients epsilon and b, and the corrected rainfall.

References

Richter, D. (1995) Ergebnisse methodischer Untersuchungen zur Korrektur des systematischen Messfehlers des Hellmann-Niederschlagsmessers. *Berichte des Deutschen Wetterdienstes*, **194**, 93 pp, Offenbach, Germany

Examples

```
clim <- slb1_meteo[as.integer(format(slb1_meteo$dates,"%Y")) %in% 2001:2005,]
clim$month <- as.integer(format(clim$dates, "%m"))

prec_meas <- clim$prec
correct_prec_frei <- with(clim,
                      correct_prec(month, tmean, prec, station.exposure = "frei"))
correct_prec_lg <- with(clim,
                    correct_prec(month, tmean, prec, station.exposure = "lg"))
correct_prec_mg <- with(clim,
                    correct_prec(month, tmean, prec, station.exposure = "mg"))
correct_prec_sg <- with(clim,
                    correct_prec(month, tmean, prec, station.exposure = "sg"))

plot(clim$dates, cumsum(correct_prec_frei),
type = "l", col = "violet", xlab = "dates", ylab = "cum. precipitation (mm)")
lines(clim$dates, cumsum(correct_prec_lg), col = "blue")
lines(clim$dates, cumsum(correct_prec_mg), col = "green")
lines(clim$dates, cumsum(correct_prec_sg), col = "red")
lines(clim$dates, cumsum(prec_meas))
legend('bottomright', c('frei', "lg", "mg", "sg"),
       col = c("violet", "blue", "green", "red", "black"),
       lty = 1, pch = NULL )
```

---

extract_layer_output *Extracts values from layer data and organizes layer-wise variables in columns*

Description

Convenience function to reorganize soil layer time series data from `layer_output` list entry produced with `run_LWFB90`. The data is transformed to a wide format, by casting the variables with the layer number using data.table's `dcast`-function.

Usage

```
extract_layer_output(
  x,
  layers = NULL,
  value_vars = NULL,
  layer_index_name = "nl",
  sep = ""
)
```

Arguments

x	Data.frame or data.table with layer data organized in rows and identified by a layer index column named layer_index_nm.
layers	Integer vector to select a subset of layers. If not supplied, values from all layers will be returned.
value_vars	Character vector containing names of value-variables to be extracted from x. If not supplied, value_vars will be guessed.
layer_index_name	Column containing layer index. Defaults to 'nl' as in layer_output.
sep	Separation character for constructing names from variable name and layer index.

Value

A data.table with the layers' values of the variables organized in columns with the names being made up of the variable name and the layer index.

Examples

```
# create a data.frame with monthly values
# identifiers: layer number, yr and mo
df <- expand.grid(nl = 1:5,
                  yr = 2002,
                  mo = 1:12)
df

#add a value variable
df$var <- runif(nrow(df), -1,0)

extract_layer_output(df)

# add more variables
df$var1 <- runif(nrow(df), 1,2)
df$var2 <- runif(nrow(df), 2,3)

# extract specific layers
extract_layer_output(df,layers = 2:4, sep = "_layer")

#extract specific variables
extract_layer_output(df, layers = 2:4, value_vars = c("var1", "var2"), sep = "_layer")
```

make_rootden *Generates a root density depth function for soil layers*

Description

Generates a root density depth function for soil layers

Usage

```
make_rootden(
  soilnodes,
  maxrootdepth = min(soilnodes),
  method = "betamodel",
  beta = 0.97,
  rootdat = NULL
)
```

Arguments

soilnodes	Vector of soil layer depth limits (including the top and the bottom of the profile) for which the relative root distribution will be calculated (m, negative downwards).
maxrootdepth	The maximum rooting depth (m, negative downwards) below which relative root length density will be set to zero (not applying when method = 'table').
method	Method name for the root depth distribution. Possible values are 'betamodel', 'table', 'linear', 'constant'. See details.
beta	Parameter of the root distribution function.
rootdat	data.frame with a given root depth density distribution. Columns are depth limits ('upper' and 'lower' in m, negative downwards) and relative root densities of fine or absorbing roots ('rootden') per unit stonefree volume. Only used when method = 'table'.

Details

method = 'betamodel' calculates the relative root length densities of the soil layers from the cumulative proportion of roots derived by the model after Gale & Grigal (1987). method = 'table' distributes the relative root densities provided by rootdat to the soil layers, under preservation of total root mass. method = 'linear' returns linearly decreasing root densities with a value of 1 at the top of the soil profile to 0 at maxrootdepth. method = 'constant' returns a uniform root distribution with a relative root length density of 1 for all soil layers above 'maxrootdepth'.

Value

Vector of relative root length densities for the soil layers framed by soilnodes. Length is one less than length(soilnodes).

References

Gale, M.R. & Grigal D.F. (1987): "Vertical root distributions of northern tree species in relation to successional status." *Canadian Journal of Forest Research*, *17:829-834*

Examples

```
depths <- c(max(slb1_soil$upper), slb1_soil$lower)
roots_beta <- make_rootden(soilnodes = depths,
                           maxrootdepth = -1,4,
                           beta = 0.97,
                           method = "betamodel")

rootden_table <- data.frame(
  upper = c(0.03,0,-0.02, -0.15, -0.35, -0.5, -0.65,-0.9,-1.1,-1.3),
  lower = c(0,-0.02, -0.15, -0.35, -0.5, -0.65,-0.9,-1.1,-1.3,-1.6),
  rootden = c(10,15, 35, 15, 7.5, 4, 12, 2, 2, 0))

roots_table <- make_rootden(soilnodes = depths,
                            method = "table",
                            rootdat = rootden_table)

roots_linear <- make_rootden(soilnodes = depths,
                             maxrootdepth = -1.4,
                             method = 'linear')

roots_constant <- make_rootden(soilnodes = depths,
                               maxrootdepth = -1.4,
                               method = 'const')

plot(roots_constant, slb1_soil$lower +runif(n=length(slb1_soil$lower), -0.02,0.02),
     type = 's', lwd = 1.5,ylab = "soil depth [m]",xlab = "relative root density",
     xlim = c(0,1), col = "red")

lines(roots_linear, slb1_soil$lower,
      type = 's', col = "blue", lwd = 1.5)

lines(roots_beta*10, slb1_soil$lower, type = 's', col = "brown", lwd = 1.5)

lines(roots_table/100, slb1_soil$lower,
      type = 's', col = "green", lwd = 1.5)


legend("bottomright", c("'betamodel'","'table'","'linear'", "'constant'"),seg.len = 1.5,
       pch = NULL, lwd =1.5, col = c("brown", "green", "blue", "red"), bty = "n")
```

`make_seasLAI` *Construct the seasonal course of leaf area index from parameters*

Description

A daily sequence of leaf area index is derived from maximum and minimum values, dates and shape parameters using different methods.

Usage

```
make_seasLAI(
  method = "b90",
  years,
  maxlai,
  winlaifrac = 0,
  budburst_doy = 121,
  leaffall_doy = 279,
  emerge_dur = 28,
  leaffall_dur = 58,
  shp_optdoy = budburst_doy + emerge_dur,
  shp_budburst = 0.5,
  shp_leaffall = 10,
 lai_doy_table = data.frame(lai_doy = c(1, 121, 150, 280, 320, 365), lai_frac = c(0, 0,
    0.5, 1, 0.5, 0))
)
```

Arguments

method	Name of method for generating the sequence. Must be one of "b90", "linear", "Coupmodel".
years	Vector of years to be returned.
maxlai	Maximum leaf are index.
winlaifrac	Fraction of maxlai during winter (ignored when method = 'linear').
budburst_doy	Budburst day of year (ignored when method = 'linear').
leaffall_doy	Day of year when leaf fall begins (ignored when method = 'linear').
emerge_dur	Number of days from budburst until maximum leaf area index is reached.
leaffall_dur	Number of days until minimum leaf are index is reached.
shp_optdoy	Day of year when optimum value is reached (required when method = "Coupmodel").
shp_budburst	Shape parameter for the growth phase (required when method = "Coupmodel").
shp_leaffall	Shape parameter growth cessation (required when method = "Coupmodel").
lai_doy_table	Data.frame with day of year ('doy') and LAI fraction values ('lai_frac'), required when method = "linear"). Can be provided as a single data.frame or a list of data.frames with one entry for each year of the simulation.

Value

A vector of daily lai values covering the years specified.

Examples

```
# Intraannual courses of leaf area index
lai_b90 <- make_seasLAI(method = "b90",
                        year = 2001,
                        maxlai = 5,
                        winlaifrac = 0,
                        budburst_doy = 121,
                        leaffall_doy = 280,
                        emerge_dur = 15,
                        leaffall_dur = 30)

lai_doy_table <- data.frame(lai_doy = c(1,110,117,135,175,220,250,290,365),
                            lai_frac = c(0.1,0.1,0.5,0.7,1.2,1.2,1.0,0.1,0.1))
lai_linear <- make_seasLAI(method = "linear",
                           year = 2001,
                           maxlai = 5,
                           lai_doy_table = lai_doy_table)

lai_coupmodel <- make_seasLAI(method = "Coupmodel",
                              year = 2001,
                              maxlai = 5,
                              winlaifrac = 0.05,
                              budburst_doy = 100,
                              leaffall_doy = 280,
                              shp_optdoy = 180,
                              shp_budburst = 0.5,
                              shp_leaffall = 5)

plot(lai_b90, type = "n", xlab = "doy", ylab = "lai [m²/m²]", ylim = c(0,6))
lines(lai_b90, col ="green",lwd = 2,)
lines(lai_linear, col ="red",lwd = 2)
lines(lai_coupmodel, col ="blue",lwd = 2)

# incorparating between-year variability
years <- 2002:2004
lai <- make_seasLAI(method = "Coupmodel",
                    year = years,
                    maxlai = c(4,6,5),
                    budburst_doy = c(100,135,121),
                    leaffall_doy = 280,
                    shp_budburst = c(3,1,0.3),
                    shp_leaffall = 3,
                    shp_optdoy =c(210,180,240) )

dates <- seq.Date(as.Date("2002-01-01"), as.Date("2004-12-31"), by = "day")
plot(dates,lai, col = "green", ylab = "lai [m²/m²]",
     type ="l", xlab = "", lwd = 2)

# doy/value-tables
lai_linear1 <- make_seasLAI(method = "linear",
                     year = years,
                     maxlai = c(4,6,5),
```

```
                    lai_doy_table = lai_doy_table)

plot(dates,lai_linear1, col = "green", ylab = "lai [m²/m²]",
     type ="l", xlab = "", lwd = 2)

# linear interpolation for lists of annual tables
lai_doy_table_l <- list("2002" = data.frame(lai_doy = c(1,90,97,115,155,200,230,270,365),
                                            lai_frac = c(0.1,0.1,0.5,0.7,1.2,1.2,1.0,0.1,0.1)),
                        "2003" = data.frame(lai_doy = c(1,100,107,125,165,210,240,280,365),
                                            lai_frac = c(0.1,0.1,0.5,0.7,1.2,1.2,1.0,0.1,0.1)),
                        "2004"= data.frame(lai_doy = c(1,110,117,135,175,220,250,290,365),
                                            lai_frac = c(0.1,0.1,0.5,0.7,1.2,1.2,1.0,0.1,0.1)))

lai_linear2 <- make_seasLAI(method = "linear",
                    year = years,
                    maxlai = c(4,6,5),
                    lai_doy_table = lai_doy_table_l)

lines(dates,lai_linear2, col = "darkgreen", ylab = "lai [m²/m²]",
     type ="l", xlab = "", lwd = 2)
```

`make_standprop` *Create daily plant characteristics from parameters and options*

Description

Creates daily sequences of 'age', 'height', 'sai', 'densef', and 'lai' from parameters and options using `approx_standprop` and `make_seasLAI`.

Usage

```
make_standprop(options_b90, param_b90, out_yrs)
```

Arguments

`options_b90`	A list of model control options.
`param_b90`	A parameter list-object.
`out_yrs`	Years for which values are returned.

Value

A data.frame containing daily sequences of 'age', 'height', 'sai', 'densef', and 'lai'.

Examples

```
options_b90 <- set_optionsLWFB90()
param_b90 <- set_paramLWFB90()

standprop <- make_standprop(options_b90,
                            param_b90,
                            out_yrs = 2002:2004)
plot(standprop$dates, standprop$lai, type = "l")
```

`parms_to_rbrook90` *Create parameter vectors for the LWF-Brook90 Fortran library*

Description

Functions to create the `param` and `siteparam` vectors to be passed to the C-interface.

Usage

```
param_to_rlwfbrook90(b90f_input)

siteparam_to_rlwfbrook90(b90f_input)
```

Arguments

`b90f_input`	A named list of model parameters.

Value

A numeric vector with the parameters in the required order.

`plant_b90` *Interpolate plant properties using the 'b90' method*

Description

Creates a daily sequence for one year from parameters.

Usage

```
plant_b90(
  minval_before_incr,
  minval_after_decr = minval_before_incr,
  maxval,
  doy.incr,
  incr.dur,
  doy.decr,
  decr.dur,
  maxdoy
)
```

Arguments

minval_before_incr	Minimum value used for the time period before doy.incr.
minval_after_decr	Minimum value used for time period after doy.decr.
maxval	Maximum value.
doy.incr	Day of year when increasing from minval_before_incr to maxval starts.
incr.dur	Duration (number of days) since doy.incr until maxval is reached.
doy.decr	Day of year when decreasing to minval_after_decr begins.
decr.dur	Duration (number of days) since doy.incr until minval_after_decr is reached.
maxdoy	Length of the year (366 for leap years).

Value

A numeric vector of length maxdoy.

Examples

```
plot(plant_b90(minval_before_incr = 0,minval_after_decr = 0.2,
maxval=1,
doy.incr = 121,incr.dur = 28,
doy.decr = 280, decr.dur = 50,
maxdoy = 365))
```

---

plant_coupmodel *Interpolate plant properties using the 'Coupmodel' method.*

Description

Creates a daily sequence for one year from parameters

Usage

```
plant_coupmodel(
  minval_before_incr,
  minval_after_decr = minval_before_incr,
  maxval,
  doy.incr,
  doy.max,
  doy.min,
  shape.incr,
  shape.decr,
  maxdoy
)
```

Arguments

minval_before_incr	Minimum value used for the time period before doy.incr.
minval_after_decr	Minimum value used for time period after doy.decr.
maxval	Maximum value.
doy.incr	Day of year when increasing from minval_before_incr to maxval begins.
doy.max	Day of year when maxval is reached.
doy.min	Day of year when minval_after_decr is reached.
shape.incr	Shape parameter of the increasing phase.
shape.decr	Shape parameter of the decreasing phase.
maxdoy	Length of the year, 366 for leap years, 365 for normal years.

Value

A numeric vector of length maxdoy.

References

Jansson, P.-E. & Karlberg, L. (2004): "Coupled heat and mass transfer model for soil-plant-atmosphere systems." *Royal Institute of Technolgy, Dept of Civil and Environmental Engineering Stockholm*

Examples

```
plot(plant_coupmodel(0,0.2,5,121, 200, 280, 0.3, 3, 365))
```

---

plant_linear *Interpolate plant properties using the 'linear' method.*

Description

Creates a daily sequence for one year from doy/value pairs.

Usage

```
plant_linear(doys, values, maxdoy)
```

Arguments

doys	Integer vector of dates (days of year).
values	Numeric vector of values.
maxdoy	Integer length of the year, 366 for leap years, 365 for normal years.

Value

A numeric vector of length `maxdoy`.

Examples

```
doys <- c(110,200,250,280)
values <-  c(0,0.8,1,0)
maxdoy <- 365
plot(plant_linear(doys = doys, values = values, maxdoy = 365))
```

`process_outputs_LWFB90` *Aggregate and group model outputs similar to ancient LWFB90 textfile outputs (.ASC-files)*

Description

Returns selected groups of variables in the chosen temporal aggregation

Usage

```
process_outputs_LWFB90(x, selection = set_outputLWFB90(), prec_interval = NULL)
```

Arguments

`x`	Named list with items `x$output` and/or `x$layer_output` (e.g. as returned by `run_LWFB90`)
`selection`	A [7,5]-matrix with row and column names, flagging the desired groups of variables at specified time intervals (see `set_outputLWFB90`).
`prec_interval`	The precipitation interval of the simulation that produced `x`. If available, the value `x$model_input$options_b90$prec_interval` is used.

Value

A named list containing the selected groups of variables in the desired temporal resolution. The names are constructed from `selection`'s row names and column names, suffixed by '.ASC' as a reminiscence to the former text file output of LWF-Brook90.

Examples

```
data("slb1_soil")
data("slb1_meteo")
opts <- set_optionsLWFB90(startdate = as.Date("2002-06-01"), enddate = as.Date("2002-06-05"))
parms <- set_paramLWFB90()
soil <- cbind(slb1_soil, hydpar_wessolek_tab(texture = slb1_soil$texture))

outsel <- set_outputLWFB90()
outsel[,] <- 1L
```

```
res <- run_LWFB90(options_b90 = opts,
           param_b90 = parms,
           climate = slb1_meteo,
           soil = soil)

# Calculate output-aggregations using the returned object
process_outputs_LWFB90(res, selection = outsel)

# or calculate aggregations at run time by passing the function via output_fun-arg
run_LWFB90(options_b90 = opts,
           param_b90 = parms,
           climate = slb1_meteo,
           soil = soil,
           rtrn_input = FALSE,
           output_fun = process_outputs_LWFB90,
           selection = outsel)$output_fun
```

`ptfs` *Functions to derive soil hydraulic properties from soil physical properties*

Description

A set of pedotransfer functions for deriving Mualem - van Genuchten parameters from soil physical properties of soil horizons, such as soil texture, bulk density and carbon content.

Usage

```
hydpar_puh2(clay, silt, sand, bd, oc.pct = 0.5)

hydpar_hypres(clay, silt, bd, oc.pct, topsoil, humconv = 1.72)

hydpar_hypres_tab(texture, topsoil)

hydpar_wessolek_tab(texture)

hydpar_ff_b90(n = 1)
```

Arguments

`clay`, `silt`, `sand`	Numeric vectors of clay, silt, sand in mass %. Particle size ranges for clay, silt and sand correspond to <2, 2-63, and 63-2000 μm. For `hydpar_hypres`, the particle size limit between silt and sand should be 50 μm.
`bd`	Numeric vector of bulk density in g cm-3.
`oc.pct`	Numeric vector of organic carbon content in mass %.

topsoil	Logical vector stating if a sample is from the topsoil or the subsoil. Used in hydpar_hypres_tab and hydpar_hypres.
humconv	Conversion factor from oc.pct to organic matter percent. Default: 1.72. Only for hydpar_hypres_tab and hydpar_hypres.
texture	Character vector of soil texture classes. For hydpar_wessolek_tab classes according to KA5 (AG Boden 2005) have to be provided. When using hydpar_hypres_tab, texture classes according to FAO (1990) have to provided.
n	An integer value specifying the number of rows of the returned data.frame (i.e. the number of repetitions of the MvG-Parameter set, only for hydpar_ff_hamken).

Details

Function hydpar_puh2 derives Mualem - van Genuchten (MvG) parameters using the regression functions developed by Puhlmann & von Wilpert (2011). The equations of Wösten et al. (1999) are available via hydpar_hypres, and their tabulated values for soil texture classes can be derived using the function hydpar_hypres_tab. The table of MvG parameters from Wessolek et al. (2009; Tab. 10) is accessible by hydpar_wessolek_tab. For this function, soil texture classes after the German texture classification system (KA5, AG Boden 2005) have to be provided. To derive hydraulic parameters of forest floor horizons, the function hydpar_ff_b90 can be used. It returns the single MvG parameter set for forest floor horizons reported by Hammel & Kennel (2001) in their original LWF-Brook90 publication.

Value

A data.frame with the following variables:

ths Saturation water content fraction

thr Residual water content fraction

npar N parameter of the van Genuchten water retention function

mpar M parameter of the van Genuchten water retention function

alpha Alpha parameter of the van Genuchten water retention function (1/m)

ksat Saturated hyraulic conductivity parameter of Mualem hydraulic conductivity function (mm/d)

tort Tortuosity parameter of Mualem hydraulic conductivity function

References

AG Boden (2005) Bodenkundliche Kartieranleitung Schweizerbart'sche Verlagsbuchhandlung, Stuttgart

Food and Agriculture Organisation (FAO) (1990) Guidelines for soil description FAO/ISRIC, Rome, 3rd edition

Hammel K & Kennel M (2001) Charakterisierung und Analyse der Wasserverfügbarkeit und des Wasserhaushalts von Waldstandorten in Bayern mit dem Simulationsmodell BROOK90. *Forstliche Forschungsberichte München* 185

Puhlmann H, von Wilpert K (2011) Testing and development of pedotransfer functions for water retention and hydraulic conductivity of forest soils. *Waldökologie, Landschaftsforschung und Naturschutz* 12, pp. 61-71

Wessolek G, Kaupenjohann M and Renger H (2009) Bodenphysikalische Kennwerte und Berechnungsverfahren für die Praxis. *Bodenökologie und Bodengenese* 40, Berlin, Germany

Woesten JHM, Lilly A, Nemes A, Le Bas C (1999) Development and use of a database of hydraulic properties of European soils. *Geoderma* 90, pp. 169-185

Examples

```
hydpar_puh2(clay = c(10,20), silt = c(40,20), sand = c(50,60), bd = c(1.6, 1.4))

hydpar_hypres(20,20,1.5,2, topsoil = FALSE)

hydpar_hypres_tab(texture = c("C","MF"), topsoil = c(TRUE,FALSE))

hydpar_wessolek_tab(c("Us", "Ls2", "mSfS"))

hydpar_ff_b90(n = 5)
```

---

r_lwfbrook90 *Interface function to the LWF-Brook90 model*

Description

Passes input data objects (parameters and meteorology) to the Fortran model code and returns the results

Usage

```
r_lwfbrook90(
  b90f_input,
  meteo,
  precip,
  output_log = TRUE,
  chk_input = TRUE,
  timelimit = Inf
)
```

Arguments

b90f_input	Brook90 input list object containing all the parameters required.
meteo	A data.table of daily meteorological data with columns: Date (dates), global radiation (globrad, W/m²), minimum and maximum air temperature (tmin, tmax, deg C), vapour pressure (vappress, kPa), wind speed (windspeed, m/s), precipipitation (prec, mm), and measured flow (mesfl, mm).
precip	An optional data.table of precipitation interval data with 6 columns: year, month, day, interval-number (1:prec_interval), prec, mesflp.
output_log	Logical whether to print runtime output to console.
chk_input	Logical whether to check for NaNs in model inputs.

timelimit	Integer for setting a time limit (in seconds, default: Inf) for the duration of the model execution.

Value

A list containing the daily and soil layer model outputs, along with an error code of the simulation.

---

replace_vecelements *Replace elements in a data.frame or vector of length > 1 by name*

Description

Replace elements in a data.frame or vector of length > 1 by name

Usage

```
replace_vecelements(x, varnms, vals)
```

Arguments

x	A vector or data.frame.
varnms	Variable names to match: Specify position by name and index.
vals	Vector of values to insert at the specified positions.

Value

The vector or data.frame in x with the elements 'varnms' replaced by vals.

Examples

```
soil_materials <- data.frame(ths = rep(0.4,3), alpha = rep(23.1, 3))

varnms = c("soil_materials.ths3", "soil_materials.ths1", "soil_materials.alpha2")
vals = c(0.999, 0.001, 99)
soil_materials
replace_vecelements(soil_materials, varnms, vals)

x <- set_paramLWFB90()[["pdur"]]
varnms <- c("pdur2", "pdur12")
vals <- c(0,10)
x
replace_vecelements(x, varnms, vals)
```

run_LWFB90 *Run the LWF-Brook90 hydrologic model*

Description

Sets up the input objects for the LWF-Brook90 hydrologic model, starts the model, and returns the selected results.

Usage

```
run_LWFB90(
  options_b90,
  param_b90,
  climate,
  precip = NULL,
  soil = NULL,
  output_fun = NULL,
  rtrn_input = TRUE,
  rtrn_output = TRUE,
  chk_input = TRUE,
  run = TRUE,
  timelimit = Inf,
  verbose = FALSE,
  ...
)
```

Arguments

options_b90	Named list of model control options. Use set_optionsLWFB90 to generate a list with default model control options.
param_b90	Named list of model input parameters. Use set_paramLWFB90 to generate a list with default model parameters.
climate	Data.frame with daily climatic data, or a function that returns a suitable data.frame. See details for the required variables.
precip	Data.frame with columns 'dates' and 'prec' to supply precipitation data separately from climate data. Can be used to provide sub-day resolution precipitation data to LWFBrook90. For each day in dates, 1 (daily resolution) to 240 values of precipitation can be provided, with the number of values per day defined in options_b90$prec_interval.
soil	Data.frame containing the hydraulic properties of the soil layers. See section 'Soil parameters'
output_fun	A function or a list of functions of the form f(x, ...), where x is the object regularly returned by run_LWFB90. During function evaluation, x contains model input and selected output objects, irrespective of rtrn_input and rtrn_output.

	Can be used to aggregate output on-the-fly, and is especially useful if the function is evaluated within a large multi-run application, for which the output might overload the memory. (see run_multi_LWFB90 and run_multisite_LWFB90).
rtrn_input	Logical: append param_b90, options_b90, and daily plant properties (standprop_daily, as derived from parameters) to the result?
rtrn_output	Logical: return the simulation results select via output?
chk_input	Logical wether to check param_b90, options_b90, climate, precip, and soil for completeness and consistency.
run	Logical: run LWF-Brook90 or only return model input objects? Useful to inspect the effects of options and parameters on model input. Default is TRUE.
timelimit	Integer to set elapsed time limits (seconds) for running LWF-Brook90.
verbose	Logical: print messages to the console? Default is FALSE.
...	Additional arguments passed to output_fun and/or climate, if the latter is a function.

Value

A list containing the selected model output (if rtrn_output == TRUE), the model input (if rtrn_input == TRUE, except for climate), and the return values of output_fun if specified.

Climate input data

The climate data.frame (or function) must contain (return) the following variables in columns named 'dates' (Date), 'tmax' (deg C), 'tmin' (deg C), 'tmean' (deg C), 'windspeed' (m/s), 'prec' (mm) , 'vappres' (kPa), and either 'globrad' (MJ/(m²d)) or 'sunhours' (h). When using sunhours, please set options_b90$fornetrad = 'sunhours'.

Soil input parameters

Each row of soil represents one layer, containing the layers' boundaries and soil hydraulic parameters. The column names for the upper and lower layer boundaries are 'upper' and 'lower' (m, negative downwards). When using options_b90$imodel = 'MvG', the hydraulic parameters are 'ths', 'thr', 'alpha' (1/m), 'npar', 'ksat' (mm/d) and 'tort'. With options_b90$imodel = 'CH', the parameters are 'thsat', 'thetaf', 'psif' (kPa), 'bexp', 'kf' (mm/d), and 'wetinf'. For both parameterizations, the volume fraction of stones has to be named 'gravel'. If the soil argument is not provided, list items soil_nodes and soil_materials of param_b90 are used for the simulation. These have to be set up in advance, see soil_to_param.

Outputs

Name	Description	Unit
yr	year	-
mo	month	-
da	day of month	-
doy	day of year	-
aa	average available energy above canopy	W/m2

adef	available water deficit in root zone	mm
asubs	average available energy below canopy	W/m2
awat	total available soil water in layers with roots between -6.18 kPa and `param_b90$psicr`	mm
balerr	error in water balance (daily value, output at the day's last precipitation interval)	mm
byfl	total bypass flow	mm/d
cc	cold content of snowpack (positive)	MJ/m2
dsfl	downslope flow	mm/d
evap	evapotranspiration	mm/d
flow	total streamflow	mm/d
gwat	groundwater storage below soil layers	mm
gwfl	groundwater flow	mm/d
intr	intercepted rain	mm
ints	intercepted snow	mm
irvp	evaporation of intercepted rain	mm/d
isvp	evaporation of intercepted snow	mm/d
lngnet	net longwave radiation	W/m2
nits	total number of iterations	-
pint	potential interception for a canopy always wet	mm/d
pslvp	potential soil evaporation	mm/d
ptran	potential transpiration	mm/d
relawat	relative available soil water in layers with roots	-
rfal	rainfall	mm/d
rint	rain interception catch rate	mm/d
rnet	rainfall to soil surface	mm/d
rsno	rain on snow	mm/d
rthr	rain throughfall rate	mm/d
sthr	snow throughfall rate	mm/d
safrac	source area fraction	-
seep	seepage loss	mm/d
sfal	snowfall	mm/d
sint	snow interception catch rate	mm/d
slfl	input to soil surface	mm/d
slvp	evaporation rate from soil	mm/d
slrad	average solar radiation on slope over daytime	W/m2
solnet	net solar radiation on slope over daytime	W/m2
smlt	snowmelt	mm/d
snow	snowpack water equivalent	mm
snowlq	liquid water content of snow on the ground	mm
snvp	evaporation from snowpack	mm/d
srfl	source area flow	mm/d
stres	tran / ptran (daily value, output at the day's last precipitation interval)	-
swat	total soil water in all layers	mm
tran	transpiration	mm/d
vrfln	vertical matrix drainage from lowest layer	mm/d

Layer outputs

Name	Description	Unit
yr	year	-
mo	month	-
da	day of month	-
doy	day of year	-
nl	index of soil layer	
swati	soil water volume in layer	mm
theta	water content of soil layer, mm water / mm soil matrix	m3/m3
wetnes	wetness of soil layer, fraction of saturation	-
psimi	matric soil water potential for soil layer	kPa
infl	infiltration to soil water in soil layer	mm/d
byfl	bypass flow from soil layer	mm/d
tran	transpiration from soil layer	mm/d
vrfl	vertical matrix drainage from soil layer	mm/d
dsfl	downslope drainage from layer	mm/d
ntfl	net flow into soil layer	mm/d

Examples

```
# Set up lists containing model control options and model parameters:
param_b90 <- set_paramLWFB90()
options_b90 <- set_optionsLWFB90()

# Set start and end Dates for the simulation
options_b90$startdate <- as.Date("2003-06-01")
options_b90$enddate <- as.Date("2003-06-30")

# Derive soil hydraulic properties from soil physical properties
# using pedotransfer functions
soil <- cbind(slb1_soil, hydpar_wessolek_tab(slb1_soil$texture))

# Run LWF-Brook90
b90.result <- run_LWFB90(options_b90 = options_b90,
                         param_b90 = param_b90,
                         climate = slb1_meteo,
                         soil = soil)

# use a function to be performed on the output:
# aggregate soil water storage down to a specific layer
agg_swat <- function(x, layer) {
  out <- aggregate(swati~yr+doy,
                   x$SWATDAY.ASC,
                   FUN = sum,
                   subset = nl <= layer)
  out[order(out$yr, out$doy),]}
```

```
# run model without returning the selected output.
b90.aggswat <- run_LWFB90(options_b90 = options_b90,
                          param_b90 = param_b90,
                          climate = slb1_meteo,
                          soil = soil,
                          output_fun = list(swat = agg_swat),
                          rtrn_output = FALSE,
                          layer = 10)  # passed to output_fun
str(b90.aggswat$output_fun$swat)
```

`run_multi_LWFB90` *Make a multirun simulation using a set of variable input parameters.*

Description

Wrapper function for `run_LWFB90` to make multiple simulations parallel, with varying input parameters.

Usage

```
run_multi_LWFB90(
  paramvar,
  param_b90,
  paramvar_nms = names(paramvar),
  cores = 2,
  show_progress = TRUE,
  ...
)
```

Arguments

`paramvar`	Data.frame of variable input parameters. For each row, a simulation is performed, with the elements in `param_b90` being replaced by the respective column of `paramvar`. All parameter names (column names) in `paramvar` must be found in `param_b90`. See section `Parameter updating`.
`param_b90`	Named list of parameters, in which the parameters defined in `paramvar` will be replaced.
`paramvar_nms`	Names of the parameters in `paramvar` to be replaced in `param_b90`.
`cores`	Number of CPUs to use for parallel processing. Default is 2.
`show_progress`	Logical: Show progress bar? Default is TRUE. See also section `Progress bar` below.
`...`	Additional arguments passed to `run_LWFB90`: provide at least the arguments that have no defaults such as `options_b90` and `climate`).

Value

A named list with the results of the single runs as returned by `run_LWFB90`. Simulation or processing errors are passed on.

Parameter updating

The transfer of values from a row in `paramvar` to `param_b90` before each single run simulation is done by matching names from `paramvar` and `param_b90`. In order to address data.frame or vector elements in `param_b90` by a column name in `paramvar`, the respective column name has to be set up from its name and index in `param_b90`. To replace, e.g., the 2nd value of `ths` in the `soil_materials` data.frame, the respective column name in `paramvar` has to be called 'soil_materials.ths2'. In order to replace the 3rd value of `maxlai` vector in `param_b90`, the column has to be named 'maxlai3'.

Data management

The returned list of single run results can become very large, if many simulations are performed and the selected output contains daily resolution data sets, especially daily layer-wise soil moisture data. To not overload memory, it is advised to reduce the returned simulation results to a minimum, by carefully selecting the output, and make use of the option to pass a list of functions to `run_LWFB90` via argument `output_fun`. These functions perform directly on the output of a single run simulation, and can be used for aggregating model output on-the-fly, or for writing results to a file or database. The regular output of `run_LWFB90` can be suppressed by setting `rtrn.output = FALSE`, for exclusively returning the output of such functions.

Progress bar

This function provides a progress bar via the package **progressr** if `show_progress=TRUE`. The parallel computation is then wrapped with `progressr::with_progress()` to enable progress reporting from distributed calculations. The appearance of the progress bar (including audible notification) can be customized by the user for the entire session using `progressr::handlers()` (see `vignette('progressr-intro')`).

Examples

```
data("slb1_meteo")
data("slb1_soil")

# Set up lists containing model control options and model parameters:
parms <- set_paramLWFB90()
# choose the 'Coupmodel' shape option for the annual lai dynamic,
# with fixed budburst and leaf fall dates:
opts <- set_optionsLWFB90(startdate = as.Date("2003-06-01"),
                                enddate = as.Date("2003-06-30"),
                                lai_method = 'Coupmodel',
                                budburst_method = 'fixed',
                                leaffall_method = 'fixed')

# Derive soil hydraulic properties from soil physical properties using pedotransfer functions
soil <- cbind(slb1_soil, hydpar_wessolek_tab(slb1_soil$texture))

#set up data.frame with variable parameters
n <- 10
set.seed(2021)
vary_parms <- data.frame(shp_optdoy = runif(n,180,240),
```

```
                          shp_budburst = runif(n, 0.1,1),
                          winlaifrac = runif(n, 0,0.5),
                          budburstdoy = runif(n,100,150),
                          soil_materials.ths3 = runif(n, 0.3,0.5), # ths of material 3
                          maxlai = runif(n,2,7))

# add the soil as soil_nodes and soil materials to param_b90, so ths3 can be looked up
parms[c("soil_nodes", "soil_materials")] <- soil_to_param(soil)

# Make a Multirun-Simulation
b90.multi <- run_multi_LWFB90(paramvar = vary_parms,
                        param_b90 = parms,
                        options_b90 = opts,
                        climate = slb1_meteo)
names(b90.multi)

# extract results
evapday <- data.table::rbindlist(
  lapply(b90.multi, FUN = function(x) { x$output[,c("yr", "doy", "evap")] }),
  idcol = "srun")

evapday$dates <- as.Date(paste(evapday$yr, evapday$doy),"%Y %j")

srun_nms <- unique(evapday$srun)

with(evapday[evapday$srun == srun_nms[1], ],
     plot(dates, cumsum(evap), type = "n",
          ylim = c(0,100))
)
for (i in 1:length(b90.multi)){
  with(evapday[evapday$srun == srun_nms[i], ],
       lines(dates, cumsum(evap)))
}
```

`run_multisite_LWFB90` *Make a multi-site simulation using lists of climate, soil, and parameter input objects.*

Description

Wrapper function for `run_LWFB90` to make multiple parallel simulations of one or several parameter sets, for a series of sites with individual climate and soil, or individual parameter sets for each climate/soil combinations.

Usage

```
run_multisite_LWFB90(
  options_b90,
```

```
  param_b90,
  soil = NULL,
  climate,
  climate_args = NULL,
  all_combinations = FALSE,
  cores = 2,
  show_progress = TRUE,
  ...
)
```

Arguments

options_b90	Named list of model control options to be used in all simulations
param_b90	Named list of parameters to be used in all simulations, or a list of multiple parameter sets.
soil	Data.frame with soil properties to be used in all simulations, or a list of data.frames with different soil profiles.
climate	Data.frame with climate data, or a list of climate data.frames. Alternatively, a function can be supplied that returns a data.frame. Arguments to the function can be passed via climate_args.
climate_args	List of named lists of arguments passed to climate, if this is a function.
all_combinations	Logical: Set up and run all possible combinations of individual param_b90, climate and soil objects? Default is FALSE, running one object or the list of param_b90 objects for a series of climate/soil combinations.
cores	Number of cores to use for parallel processing.
show_progress	Logical: Show progress bar? Default is TRUE. See also section Progress bar below.
...	Further arguments passed to run_LWFB90.

Value

A named list with the results of the single runs as returned by run_LWFB90. Simulation or processing errors are passed on. The names of the returned list entries are concatenated from the names of the input list entries in the following form: <climate> <soil> <param_b90>. If climate is a function, the names for <climate> are taken from the names of climate_args.

Data management

The returned list of single run results can become very large, if many simulations are performed and the selected output contains daily resolution data sets, especially daily layer-wise soil moisture data. To not overload memory, it is advised to reduce the returned simulation results to a minimum, by carefully selecting the output, and make use of the option to pass a list of functions to run_LWFB90 via argument output_fun. These functions perform directly on the output of a single run simulation, and can be used for aggregating model output on-the-fly, or for writing results to a file or database. The regular output of run_LWFB90 can be suppressed by setting rtrn.output = FALSE, for exclusively returning the output of such functions. To provide full flexibility, the

names of the current soil, param_b90, and climate are automatically passed as additional arguments (soil_nm, param_nm,clim_nm) to run_LWFB90 and in this way become available to functions passed via output_fun. In order to not overload the memory with climate input data, it is advised to provide a function instead of a list of climate data.frames, and specify its arguments for individual sites in climate_args, in case many sites with individual climates will be simulated.

Progress bar

This function provides a progress bar via the package **progressr** if show_progress=TRUE. The parallel computation is then wrapped with progressr::with_progress() to enable progress reporting from distributed calculations. The appearance of the progress bar (including audible notification) can be customized by the user for the entire session using progressr::handlers() (see vignette('progressr-intro')).

Examples

```
data("slb1_meteo")
data("slb1_soil")

opts <- set_optionsLWFB90(budburst_method = "Menzel", enddate = as.Date("2002-12-31"))

# define parameter sets
param_l <- list(spruce = set_paramLWFB90(maxlai = 5,
                                         budburst_species = "Picea abies (frueh)",
                                         winlaifrac = 0.8),
                beech = set_paramLWFB90(maxlai = 6,
                                        budburst_species = "Fagus sylvatica",
                                        winlaifrac = 0))

soil <- cbind(slb1_soil, hydpar_wessolek_tab(slb1_soil$texture))

# define list of soil objects
soils <- list(soil1 = soil, soil2 = soil)

# define list of climate objects
climates <- list(clim1 = slb1_meteo, clim2 = slb1_meteo)

# run two parameter sets on a series of climate and soil-objects
res <- run_multisite_LWFB90(param_b90 = param_l,
                      options_b90 = opts,
                      soil = soils,
                      climate = climates)
names(res)

# set up and run individual parameter sets for individual locations

# set up location parameters
loc_parm <- data.frame(loc_id = names(climates),
                       coords_y = c(48.0, 54.0),
                       eslope = c(30,0),
                       aspect = c(180,0))
```

```
# create input list of multiple param_b90 list objects
param_l <- lapply(names(climates), function(x, loc_parms) {
  parms <- set_paramLWFB90()
  parms[match(names(loc_parm),names(parms), nomatch = 0)] <-
    loc_parm[loc_parm$loc_id == x, which(names(loc_parm) %in% names(parms))]
  parms
}, loc_parm = loc_parm)

names(param_l) <- c("locpar1", "locpar2")

res <- run_multisite_LWFB90(param_b90 = param_l,
                      options_b90 = opts,
                      soil = soils,
                      climate = climates)
names(res)
```

`set_optionsLWFB90` *Create a list of model control options*

Description

Create a list of model control options

Usage

```
set_optionsLWFB90(...)
```

Arguments

`...`	Named values to be included in return value.

Details

startdate start date of the simulation.

enddate end date of the simulation.

fornetrad use global solar radiation (=`'globrad'`) or sunshine duration hours (=`'sunhours'`) for net radiation calculation?

prec_interval number of precipitation intervals per day (default is 1). If `prec_interval` > 1, the `precip`-argument has to be provided to `run_LWFB90`

correct_prec correct precipitation data for wind and evaporation losses using `correct_prec`?

budburst_method name of method for budburst calculation. If `'constant'` or `'fixed'`, budburst day of year from parameters is used. All other methods calculate budburst day of year dynamically from airtemperatures, and the method name is passed to the `start.method`-argument of `vegperiod`.

leaffall_method name of method for leaffall calculation. If `'constant'` or `'fixed'`, beginning of leaffall (day of year) from parameters is used. All other methods calculate budburst day of year dynamically from temperatures, and the method name is passed to the `end.method`-argument of `vegperiod`.

standprop_input name of input for longterm (interannual) plant development. `standprop_input = 'parameters'`: yearly values of stand properties height, sai, densef, lai are taken from individual parameters, `standprop_input = 'table'`: values from `standprop_table` provided in parameters are used.

standprop_interp interpolation method for aboveground stand properties. `'linear'` or `'constant'`, see `approx.method`-argument of `approx_standprop`.

use_growthperiod Should yearly changes of stand properties (growth) only take place during the growth period? If `TRUE`, linear interpolation of height, sai, densef and age are made from budburst until leaffall. During winter values are constant. Beginning and end of the growth period are taken from parameters `budburstdoy` and `leaffalldoy`. See `use_growthperiod`-argument of `approx_standprop`.

lai_method name of method for constructing seasonal course leaf area index development from parameters. Passed to `method`-argument of `make_seasLAI`.

imodel name of retention & conductivity model: "CH" for Clapp/Hornberger, "MvG" for Mualem/van Genuchten

root_method method name of the root length density depth distribution function. Any of the names accepted by `make_rootden` are allowed. Additionally, `'soilvar'` can be used if the root length density depth distribution is specified in column 'rootden' in the `soil`-data.frame

Value

A list of model control options for use as `options_b90`-argument in `run_LWFB90`.

Examples

```
# Default options
options_b90 <- set_optionsLWFB90()
# Include specific options
options_b90_dynamic_phenology <- set_optionsLWFB90(budburst_method = 'Menzel',
leaffall_method ='vonWilpert')
```

---

set_outputLWFB90 *Select output for LWF-Brook90*

Description

Returns a `[7,5]` matrix with a default selection of LWF-Brook90 output data sets for the use as 'output'-argument `run_LWFB90`.

Usage

```
set_outputLWFB90(output = NULL, edit = FALSE)
```

Arguments

output	optional [7,5]-matrix, which is opened on R's data-editor if edit = TRUE. If no matrix is passed, a default selection of output values is returned opened in R's data-editor.
edit	open R's data-editor ?

Value

a [7,5]-matrix containing 0 and 1 for use as output-argument in run_LWFB90

Examples

```
# create matrix with default selection
output <- set_outputLWFB90()
output

# modify
output[,] <- 0L
output[,3] <- 1L
output["Evap", c("Ann","Mon")] <- 1L
output
```

---

set_paramLWFB90 *Create the list of model parameters*

Description

Create the list of model parameters

Usage

```
set_paramLWFB90(...)
```

Arguments

...	Named arguments to be included in return value.

Value

A list with model parameters for use as param_b90-argument in run_LWFB90.

List of input parameters

Name	Description
czr	Ratio of roughness length to mean height for smooth closed canopies for heights greater than HR when L
czs	Ratio of roughness length to mean height for smooth closed canopies for heights less than HS when LAI
hr	Smallest height to which CZR applies. Default: 10
hs	Largest height to which CZS applies. Default: 1
lpc	Minimum leaf area index defining a closed canopy. Default: 4
lwidth	Average leaf width. Default: 0.1
nn	Eddy diffusivity extinction coefficient within canopy. Default: 2.5
rhotp	Ratio of total leaf area to projected area. Default: 2
zminh	Reference height for weather data above the canopy top height. Default: 2
dslope	Slope for downslope-flow. Default: 0
bypar	Switch to allow (1) or prevent (0) bypass flow in deeper layers. Default: 0
drain	Switch for lower boundary condition to be free drainage (1) or no flow (0). Default: 1
slopelen	Slope length for downslope-flow. Default: 200
gsc	Rate constant for ground water discharge (remember that a first order groundwater reservoir is placed bel
gsp	Seepage fraction of groundwater discharge. Default:
ilayer	Number of layers from top to which infiltration is distributed. Default: 0
imperv	Fraction of area which has an impermeable surface (like roads). Default: 0
infexp	Shape parameter for distribution of infiltration in first ILayer, for value 0 infiltration is in top layer only. I
qffc	Quickflow fraction of infiltrating water at field capacity, for value 0 there is no quickflow (bypass or surfa
qfpar	Quickflow shape parameter. Default: 1
qlayer	Number of layers which are considered for generation of surface or source area flow. Default: 0
water_table_depth	Depth of the water table for the lower boundary condition. -9999 (the default) means unit gradient flux b
gwatini	Initial value of groundwater storage. Default: 0
snowini	Initial value of water content of snow pack. Default: 0
snowlqini	Initial value of liquid water content of snow pack. Default: 0
snowccini	Initial value of cold content of snow pack (positive). Default: 0
intrainini	Initial value of intercepted rain. Default: 0
intsnowini	Initial value of intercepted snow. Default: 0
psiini	Initial pressure head of soil layers. May have the same length as row.names(soil). Default: -6.3
cintrl	Maximum interception storage of rain per unit LAI. Default: 0.15
cintrs	Maximum interception storage of rain per unit SAI. Default: 0.15
cintsl	Maximum interception storage of snow per unit LAI. Default: 0.6
cintss	Maximum interception storage of snow per unit SAI. Default: 0.6
frintlai	Intercepted fraction of rain per unit LAI. Default: 0.06
frintsai	Intercepted fraction of rain per unit SAI. Default: 0.06
fsintlai	Intercepted fraction of snow per unit LAI. Default: 0.04
fsintsai	Intercepted fraction of snow per unit SAI. Default: 0.04
pdur	Average duration of precipitation events for each month of the year. Default: rep(4,12)
alb	Albedo of soil/vegetation surface without snow. Default: 0.2
albsn	Albedo of soil/vegetation surface with snow. Default: 0.5
c1	Intercept of relation of solar radiation to sunshine duration. Default: 0.25
c2	Intercept of relation of solar radiation to sunshine duration. Default: 0.5
c3	Constant between 0 and 1 that determines the cloud correction to net longwave radiation from sunshine d
fetch	Fetch upwind of the weather station at which wind speed was measured. Default: 5000
ksnvp	Correction factor for snow evaporation. Default: 0.3

wndrat	Average ratio of nighttime to daytime wind speed. Default: 0.3
z0s	Surface roughness of snow cover. Default: 0.001
z0w	Roughness length at the weather station at which wind speed was measured. Default: 0.005 (Grass)
coords_x	Longitude value (decimal degrees) of the simulation location (has no effect on simulation results). Defau
coords_y	Latitude value (decimal degrees) of the simulation location. Default: 51.54
zw	Height at which wind speed was measured. Default: 2
eslope	Slope for evapotranspiration and snowmelt calculation. Default: 0
aspect	Mean exposition of soil surface at soil profile (north: 0, east: 90, south: 180, west: 270). Default: 0
obsheight	Mean height of obstacles on soil surface (grass, furrows etc.), used to calculate soil surface roughness. D
prec_corr_statexp	Station exposure situation of prec measurements (passed to `correct_prec` Default: 'mg'
dpsimax	Maximum potential difference considered equal. Default: 5e-04
dswmax	Maximum change allowed in SWATI. Default: 0.05
dtimax	Maximum iteration time step. Default: 0.5
budburst_species	Name of tree species for estimating budburst doy using Menzel-model (passed to `vegperiod`) Default: 'F
budburstdoy	Budburst day of year - passed to `make_seasLAI`. Default: 121
emergedur	Leaf growth duration until maxlai is reached.. Default: 28
height	Plant height. Default: 25
height_ini	Initial plant height at the beginning of the simulation. Used when `options_b90$standprop_interp = '`
leaffalldoy	Day of year when leaf fall begins - passed to `make_seasLAI` Default: 279
leaffalldur	Number of days until minimum lai is reached - passed to `make_seasLAI` Default: 58
sai	Steam area index. Default: 1
sai_ini	Initial stem area index at the beginning of the simulation. Used when `options_b90$standprop_interp`
shp_leaffall	Shape parameter for leaf fall phase - passed to `make_seasLAI` Default: 0.3
shp_budburst	Shape parameter for leaf growth phase - passed to `make_seasLAI` Default: 3
shp_optdoy	Day of year when optimum value is reached - passed to `make_seasLAI` Default: 210
lai_doy	Day of year values for lai-interpolation - passed to `make_seasLAI`
lai_frac	Fractional lai values for lai interpolation, corresponding to lai_doy - passed to `make_seasLAI` Default: 21
lai_doy_table	data.frame with day of year values ('doy) and corresponding lai fractions ('lai_frac'), or a list of data.fram
winlaifrac	Minimum LAI as a fraction of maxlai. Default: 0
standprop_table	Data.frame with yearly values of vegetation properties with columns 'year','age', 'height', 'maxlai', 'sai'
cs	Ratio of projected stem area index to canopy height. Default: 0.035
densef	Density factor for MaxLAI, CS, RtLen, RPlant, not <.001, 1 for typical stand. Default: 1
densef_ini	Initial density factor at the beginning of the simulation. Used when `options_b90$standprop_interp =`
maxlai	Maximum projected leaf area index - passed to `make_seasLAI` Default: 5
radex	Extinction coefficient for solar radiation and net radiation in the canopy. Default: 0.5
cvpd	Vapour pressure deficit at which leaf conductance is halved. Default: 2
glmax	Maximum leaf vapour conductance when stomata are fully open. Default: 0.0053
glmin	Minimum leaf vapour conductance when stomata are closed. Default: 0.0003
r5	Solar radiation level at which leaf conductance is half of its value at RM. Default: 100
rm	Nominal maximum solar shortwave radiation possible on a leaf (to reach glmax). Default: 1000
t1	Lower suboptimal temperature threshold for stomata opening - temperature relation. Default: 10
t2	Upper suboptimal temperature threshold for stomata opening - temperature relation. Default: 30
th	Upper temperature threshold for stomata closure. Default: 40
tl	Lower temperature threshold for stomata closure. Default: 0
betaroot	Shape parameter for rootlength density depth distribution. Default: 0.97
maxrootdepth	Maximum root depth (positive downward) - passed to make_rootden. Default: -1.5
rootden_table	Data.frame of relative root density depth distribution with columns 'depth' and 'rootden'
rstemp	Base temperature for snow-rain transition. Default: -0.5

ccfac	Cold content factor. Default: 0.3
grdmlt	Rate of groundmelt of snowpack. Default: 0.35
laimlt	Parameter for snowmelt dependence on LAI. Default: 0.2
maxlqf	Maximum liquid water fraction of Snow. Default: 0.05
melfac	Degree day melt factor for open. Default: 1.5
saimlt	Parameter for snowmelt dependence on SAI. Default: 0.5
snoden	Snow density. Default: 0.3
rssa	Soil evaporation resistance at field capacity. Default: 100
rssb	Exponent in relation of RSS to water potential. Default: 1
soil_nodes	Data.frame with soil nodes discretization passed to LWF-Brook90
soil_materials	Data.frame with soil materials (hydrualic parameters) passed to LWF-Brook90
age_ini	Age of stand (for root development). Default: 100
initrdep	Initial root depth. Default: 0.25
initrlen	Initial water-absorbing root length per unit area. Default: 12
rgroper	Period of net root growth. A value of 0 prevents root growth. Default: 0
rgrorate	Vertical root growth rate. Default: 0.03
fxylem	Fraction of internal plant resistance to water flow that is in the Xylem. Default: 0.5
maxrlen	Total length of fine roots per unit ground area. Default: 3000
mxkpl	Maximum internal conductivity for water flow through the plants. Default: 8
nooutf	Switch that prevents outflow from the root to the soil when the soil is dry. Default: 1
psicr	Critical leaf water potential at which stomates close. Default: -2
rrad	Average radius of the fine or water-absorbing roots. Default: 0.35

Examples

```
# Default parameter
parms <- set_paramLWFB90()
# Include specific parameters
parms_maxlai <- set_paramLWFB90(maxlai = c(4,6,5), height =20)
```

`slb1_meteo` *Meteorological Data from the Solling Beech and Spruce experimental site*

Description

A dataset containing daily weather variables for the period 1960-2013

Usage

```
slb1_meteo
```

Format

A data.frame with 19724 rows and 9 variables

dates date

tmin daily minimum temperature, deg C

tmax daily maximum temperature, deg C

tmean daily mean temperature, deg C

prec daily sum of precipitation, mm

relhum relative Humidity, %

globrad daily sum of global radiation, MJ/m²

windspeed daily mean wind speed measured at 10 m above ground, m/s

vappres daily vapour pressure, kPa

`slb1_prec2013_hh` *Hourly precipitation data from Solling Beech experimental site 'SLB1' for year 2013*

Description

Hourly precipitation data from Solling Beech experimental site 'SLB1' for year 2013

Usage

```
slb1_prec2013_hh
```

Format

A data.frame with 8760 rows and 2 variables

dates date

prec hourly sum of precipitation, mm

slb1_soil *Soil profile data from the Solling Beech experimental site 'SLB1'*

Description

A dataset containing the soil horizons' physical properties

Usage

```
slb1_soil
```

Format

A data.frame with 21 rows and 10 variables

horizon horizon symbol

upper upper layer boundary, m

lower lower layer boundary, m

texture soil texture according to German soil texture classification system

bd bulk density of the fine earth, g/cm³

gravel fraction of coarse material

sand sand content, mass-%

silt silt content, mass-%

clay clay content, mass-%

c_org organic carbon content, mass-%

slb1_standprop *Annual stand properties of the Solling Beech experimental site 'SLB1'*

Description

A dataset containing the forests's stand properties

Usage

```
slb1_standprop
```

Format

A data.frame with 49 rows and 7 variables

year Year of observation

species Tree species

age age of the main stand

height average height of the trees, m

maxlai maximum leaf area index, m²/m²

sai stem area index, m²/m²

densef stand density

`soil_to_param` *Split up soil into materials and soil nodes.*

Description

Split up soil into materials and soil nodes.

Usage

```
soil_to_param(soil, imodel = "MvG")
```

Arguments

`soil`	Data.frame with soil layer boundaries ('upper', 'lower') and hydraulic parameters. When imodel = 'MvG', columns of soil have to be named 'ths', 'thr', 'alpha', 'npar', 'ksat', 'tort', 'gravel'. When imodel = 'CH', columns have to be named thsat , 'thetaf','psif', 'bexp','kf', 'wetinf', 'gravel'.
`imodel`	Name of the hydraulic model ('MvG' or 'CH')

Value

a list with data.frames 'soil_nodes' and 'soil_materials'

Examples

```
data(slb1_soil)
soil <- slb1_soil
soil <- cbind(soil, hydpar_wessolek_tab(soil$texture))
str(soil)

soil_layers_materials <- soil_to_param(soil)
soil_layers_materials
```

standprop_yearly_to_param *Transfer standproperties height, maxlai, sai, densef, age to parameter list object*

Description

Takes a data.frame of yearly stand properties, trims/extends the columns height, maxlai, sai, densef, and age for the years in `out_yrs`, and updates the provided parameter list.

Usage

```
standprop_yearly_to_param(standprop_yearly, param_b90, out_yrs)
```

Arguments

`standprop_yearly`	A data.frame or data.table with columns 'year', 'height', 'maxlai', 'sai', 'densef', 'age'.
`param_b90`	A list object to update.
`out_yrs`	Vector of years for which parameters should be updated.

Value

The param_b90 list-object with updated items maxlai, height, height_ini, sai, sai_ini, densef, densef_ini, age, age_ini.

Examples

```
param_b90 <- set_paramLWFB90()
dat <- slb1_standprop

years <- 2002:2005
param.new <- standprop_yearly_to_param(dat,
                                       param_b90,
                                       years)

identical(param.new$maxlai, dat$maxlai[dat$year %in% years])
identical(param.new$height, dat$height[dat$year %in% years])
```

Index

∗ **datasets**
    slb1_meteo, 37
    slb1_prec2013_hh, 38
    slb1_soil, 39
    slb1_standprop, 39

approx_standprop, 3, *14*, *33*, *36*

calc_globrad, 5
calc_vegperiod, 6
correct_prec, 7, *32*, *36*

dcast, *8*

extract_layer_output, 8

hydpar_ff_b90 *(*ptfs*)*, 19
hydpar_hypres, *19*
hydpar_hypres *(*ptfs*)*, 19
hydpar_hypres_tab, *20*
hydpar_hypres_tab *(*ptfs*)*, 19
hydpar_puh2 *(*ptfs*)*, 19
hydpar_wessolek_tab *(*ptfs*)*, 19

make_rootden, 10, *33*
make_seasLAI, 11, *14*, *33*, *36*
make_standprop, 14

param_to_rlwfbrook90
        *(*parms_to_rbrook90*)*, 15
parms_to_rbrook90, 15
plant_b90, 15
plant_coupmodel, 16
plant_linear, 17
process_outputs_LWFB90, 18
ptfs, 19

r_lwfbrook90, 21
replace_vecelements, 22
run_LWFB90, *8*, *18*, 23, *27–34*
run_multi_LWFB90, *24*, 27
run_multisite_LWFB90, *24*, 29

set_optionsLWFB90, *23*, 32
set_outputLWFB90, *18*, 33
set_paramLWFB90, *23*, 34
siteparam_to_rlwfbrook90
        *(*parms_to_rbrook90*)*, 15
slb1_meteo, 37
slb1_prec2013_hh, 38
slb1_soil, 39
slb1_standprop, 39
soil_to_param, *24*, 40
standprop_yearly_to_param, 41

vegperiod, *6*, *32*, *33*, *36*