

# Package ‘LWFBrook90R’

February 24, 2021

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

**Version** 0.4.4

**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://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), progressr (>= 0.6.0)

**Suggests** sirad, knitr, rmarkdown, roxygen2, testthat

**LazyData** true

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

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

Robert Nuske [ctb] (<<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** CRAN

**Date/Publication** 2021-02-24 22:40:29 UTC

## R topics documented:

approx_standprop . . . . .	3
calc_globrad . . . . .	5
calc_vegperiod . . . . .	6
correct_prec . . . . .	7
extract_layer_output . . . . .	8
LWFBrook90R . . . . .	9
make_rootden . . . . .	10
make_seasLAI . . . . .	11
make_standprop . . . . .	13
param_to_rlwfbrook90 . . . . .	14
plant_b90 . . . . .	15
plant_coupmode . . . . .	15
plant_linear . . . . .	16
ptfs . . . . .	17
replace_vecelements . . . . .	19
run_LWFB90 . . . . .	20
run_multisite_LWFB90 . . . . .	24
run_multi_LWFB90 . . . . .	26
r_lwfbrook90 . . . . .	29
set_optionsLWFB90 . . . . .	30
set_outputLWFB90 . . . . .	32
set_paramLWFB90 . . . . .	32
slb1_meteo . . . . .	35
slb1_soil . . . . .	36
slb1_standprop . . . . .	37
soil_to_param . . . . .	37
standprop_yearly_to_param . . . . .	38

**Index**

**39**

---

approx_standprop	<i>Create a daily sequence of stand properties from parameters using interpolation</i>
------------------	--

---

### 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 extrem 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/(m<sup>2</sup> 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_vegperiod	<i>Calculate the dates of budburst and beginning of leaf fall</i>
----------------	---

---

**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 <a href="#">vegperiod</a> .
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 <a href="#">vegperiod</a> .
dates	date vector passed to <a href="#">vegperiod</a> , ignored if both leaffall_method and budburst_method = 'fixed'
tavg	vector of daily mean air temperature (deg C) passed to <a href="#">vegperiod</a> , 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 <a href="#">vegperiod</a> .

**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	<i>Correct rain gauge precipitation data for wind and evaporation errors after Richter (1995)</i>
--------------	---

---

**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 data such as BELO and SWAT to the wide format, by casting variables with the layer number, using `data.table`'s `dcast`-function.

**Usage**

```
extract_layer_output(dat, layers = NULL, value_vars = NULL, sep = "")
```



**Arguments**

<code>dat</code>	Data.frame with layer data organized in rows and identified by a layer number column <code>nl</code> .
<code>layers</code>	Integer vector addressing the layer numbers ( <code>nl</code> ) to be extracted from <code>dat</code> . If not supplied, values from all layers will be returned.
<code>value_vars</code>	Character vector containing names of value-variables to be extracted from <code>dat</code> . If not supplied, <code>value_vars</code> will be guessed.
<code>sep</code>	Separation character for constructig names from variable name and layer index.

**Value**

A data.table with the layers' values of the variables organized in columns (wide format) with the names being made up of the variable name and layer number.

**Examples**

```
# create a data.frame with monthly values
# identifiers: layer number, yr and mo
df <- expand.grid(nl = 1:5,
                 yr = 2002:2003,
                 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 = "_")

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

---

LWFBrook90R

*LWFBrook90R: A package for simulating water fluxes, soil moisture and drought stress using the SVAT Model LWF-Brook90.*


---

**Description**

The central function `run_LWFB90()` creates model input from model control options, parameters, climate and soil data, executes the model code and returns the model results. The model control options thereby let the user select different methods for defining phenology and leaf area index seasonality, root density depth distributions, and inter-annual variation of stand properties. Additionally, a set of pedotransfer functions is provided to derive hydraulic parameters from soil physical properties. For an introduction to the basic usage see `vignette('intro_LWFB90')`.

---

make_rootden	<i>Generates a root density depth function for soil layers</i>
--------------	--

---

### 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")

```

**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",
  year,
  maxlai,
  winlaifrac = 0,
  budburst_doy = 121,
  leaffall_doy = 279,
  emerge_dur = 28,
  leaffall_dur = 58,
  shp_optdoy = 220,
  shp_budburst = 0.5,
  shp_leaffall = 10,
  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".
year	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	Integer vector of days of years.
lai_frac	Vector of values of fractional leaf area index corresponding to lai_doy (required when method = "linear").

**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 <- 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 = lai_doy,
  lai_frac = lai_frac)

lai_coupmodel <- make_seasLAI(method = "Coupmodel",
  year = 2001,
  maxlai = 5,
  winlaifrac = 0.1,
  budburst_doy = 110,
  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)

# incorporating between-year variability
years <- 2001:2003
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("2001-01-01"), as.Date("2003-12-31"), by = "day")
plot(dates,lai, col = "green", ylab = "lai [m2/m2]",
  type = "l", xlab = "", lwd = 2)

```

**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")
```

---

param\_to\_rlwfbrook90 *Create a parameter vector for the r\_lwfbrook90-function*

---

**Description**

The param vector for `r_lwfbrook90` is created from model parameters.

**Usage**

```
param_to_rlwfbrook90(param_b90, imodel)
```

**Arguments**

param_b90	A named list of model parameters.
imodel	Name of hydraulic model ('MvG' or 'CH')

**Value**

A numerical vector with the parameters in the right order for `r_lwfbrook90`.

---

 plant\_b90

*Interpolate plant properties using the 'b90' method.*


---

**Description**

Creates a daily sequence for one year from parameters

**Usage**

```
plant_b90(minval, maxval, doy.incr, incr.dur, doy.decr, decr.dur, maxdoy)
```

**Arguments**

minval	Minimum value.
maxval	Maximum value.
doy.incr	Day of year when increasing from minval to maxval begins.
incr.dur	Duration (number of days) since doy.incr until maxval is reached.
doy.decr	Day of year when decreasing to minval begins.
decr.dur	Duration (number of days) since doy.incr until minval is reached.
maxdoy	Length of the year, 366 for leap years, 365 for normal years.

**Value**

A numeric vector of length maxdoy.

**Examples**

```
plot(plant_b90(minval = 0,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,  
  maxval,  
  doy.incr,  
  doy.max,  
  doy.min,  
  shape.incr,  
  shape.decr,  
  maxdoy  
)
```

**Arguments**

minval	Minimum value.
maxval	Maximum value.
doy.incr	Day of year when increasing from minval to maxval begins.
doy.max	Day of year when maxval is reached.
doy.min	Day of year when minval is reached again.
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 Technology, Dept of Civil and Environmental Engineering Stockholm*

**Examples**

```
plot(plant_coupmodel(0,5, 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(doy, values, maxdoy)
```

**Arguments**

doy	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**

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

---

 ptfs

---

*Functions to derive soil hydraulic properties from soil 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**

```
hydp_ar_puh2(clay, silt, sand, bd, oc.pct = 0.5)
hydp_ar_hypres(clay, silt, bd, oc.pct = 0.1, topsoil = TRUE, humconv = 1.72)
hydp_ar_hypres_tab(texture, topsoil)
hydp_ar_wessolek_tab(texture)
hydp_ar_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 $\mu m$ . For <a href="#">hydp_ar_hypres</a> , the particle size limit between silt and sand should be 50 $\mu m$ .
bd	Numeric vector of bulk density in g cm <sup>-3</sup> .

oc.pct	Numeric vector of organic carbon content in mass %.
topsoil	Logical: Is the sample from the topsoil? Used in <a href="#">hyddpar_hypres_tab</a> .
humconv	Conversion factor from oc.pct to organic matter percent. Default: 1.72. Only for <a href="#">hyddpar_hypres_tab</a> .
texture	Character vector of soil texture classes. For <a href="#">hyddpar_wessolek_tab</a> classes according to KA5 (AG Boden 2005) have to be provided. When using <a href="#">hyddpar_hypres_tab</a> , texture classes according to FAO (1990) have to be 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 <a href="#">hyddpar_ff_hamken</a> ).

### Details

Function `hyddpar_puh2` derives Mualem - van Genuchten (MvG) parameters using the regression functions developed by Puhmann & von Wilpert (2011). The equations of Wösten et al. (1999) are available via [hyddpar\\_hypres](#), and their tabulated values for soil texture classes can be derived using the function [hyddpar\\_hypres\\_tab](#). The table of MvG parameters from Wesselok et al. (2009; Tab. 10) is accessible by [hyddpar\\_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 `hyddpar_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

**np** N parameter of the van Genuchten water retention function

**mp** M parameter of the van Genuchten water retention function

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

**ksat** Saturated hydraulic 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

Puhmann 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

```
hyddpar_puh2(clay = c(10,20), silt = c(40,20), sand = c(50,60), bd = c(1.6, 1.4))
```

```
hyddpar_hypres(20,20,1.5,2)
```

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

```
hyddpar_wessolek_tab(c("Us", "Ls2", "mSfS"))
```

```
hyddpar_ff_b90(n = 5)
```

---

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(th3 = rep(0.4,3), alpha = rep(23.1, 3))

varnms = c("soil_materials.th3", "soil_materials.th1", "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 = 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 <a href="#">set_optionsLWFB90</a> to generate a list with default model control options.  |
| param_b90   | Named list of model input parameters. Use <a href="#">set_paramLWFB90</a> 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	A [7,5]-matrix flagging the desired model output datasets at different time intervals. Use <a href="#">set_outputLWFB90</a> to generate and edit a default output selection matrix. Alternatively, use -1 (the default) to return the raw daily and soil layer outputs.
output_fun	A function or a list of functions of the form $f(x, \dots)$ , 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 <a href="#">run_multi_LWFB90</a> and <a href="#">run_multisite_LWFB90</a> ).
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 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/(m<sup>2</sup>d)) 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\$model = 'MvG', the hydraulic parameters are 'ths', 'thr', 'alpha' (1/m), 'npar', 'ksat' (mm/d) and 'tort'. With options\_b90\$model = '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](#).

**Daily outputs**

<b>Name</b>	<b>Description</b>	<b>Unit</b>
yr	year	-
mo	month	-
da	day of month	-
doy	day of year	-
adef	available water deficit in root zone	mm
awat	total available soil water in layers with roots between -6.18 kPa and param_b90\$psicr	mm
balerr	error in water balance	mm
byfl	total bypass flow	mm
dsfl	downslope flow	mm
evap	evapotranspiration	mm
flow	total streamflow	mm
gwat	groundwater storage below soil layers at the end of the time interval	mm
gwfl	groundwater flow	mm
intr	intercepted rain at the end of the time interval	mm
ints	intercepted snow at the end of the time interval	mm
irvp	evaporation of intercepted rain	mm
isvp	evaporation of intercepted snow	mm
nits	total number of iterations in time interval	-
pint	potential interception for a canopy always wet	mm
pslvp	potential soil evaporation	mm
ptran	potential transpiration	mm
relawat	relative available soil water in layers with roots	-
rfal	rainfall	mm
rint	rain interception	mm
rnet	rainfall to soil surface	mm
rsno	rain on snow	mm
rthr	rain throughfall rate	mm
safrac	source area fraction	-
seep	seepage loss	mm
sfal	snowfall	mm
sint	snow interception	mm
slfl	input to soil surface	mm
slvp	evaporation rate from soil	mm
smlt	snowmelt	mm
snow	snowpack water equivalent	mm
snvp	evaporation from snowpack	mm
srfl	source area flow	mm
stres	TRAN / PTRAN for time period	-
swat	total soil water in all layers at the end of the time interval	mm
tran	transpiration	mm
vrfln	vertical matrix drainage from lowest layer	mm

**Layer outputs**

<b>Name</b>	<b>Description</b>	<b>Unit</b>
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	-
wetnes	wetness of soil layer, fraction of saturation	-
psimi	matric soil water potential for soil layer	kPa
psiti	total soil water potential for a soil layer	kPa
infl	infiltration to soil water in soil layer	mm
byfl	bypass flow from soil layer	mm
tran	transpiration from soil layer	mm
slvp	soil evaporation from a soil layer	mm
vrfl	vertical matrix drainage from soil layer	mm
dsfl	downslope drainage from layer	mm
ntfl	net flow into soil layer	mm

**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, hydpars_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 = n1 <= 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),
                        rtn_output = FALSE,
                        layer = 10) # passed to output_fun
str(b90.aggswat$output_fun$swat)

```

---

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 rtn.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, hyddpar_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)

```

**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**

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

**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, hydpars_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)

# select outputs
output <- set_outputLWFB90()
```

```

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

names(b90.multi)

# extract results: EVAPDAY.ASC
evapday <- data.frame(data.table::rbindlist(lapply(b90.multi,
                                                  FUN = function(x) {x[["EVAPDAY.ASC"]]}),
                                                  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)))
}

```

---

r\_lwfbrook90

*Interface function to the LWF-Brook90 model*


---

## Description

Passes input data matrices to the Fortran model code and returns the results

## Usage

```

r_lwfbrook90(
  siteparam,
  climveg,
  param,
  pdur,
  soil_materials,
  soil_nodes,
  precdat = NULL,
  output_log = TRUE,
  timelimit = Inf
)

```

**Arguments**

siteparam	A [1,6] matrix with site level information: start year, start doy, latitude, initial snow, initial groundwater, precipitation interval.
climveg	A matrix with 15 columns of climatic and vegetation data: year, month, day, global radiation in MJ/(m <sup>2</sup> d), tmax (deg C), tmin (deg C), vappres (kPa), wind (m/s), prec (mm), mesfl (mm), densef (-), stand height (m), lai (m <sup>2</sup> /m <sup>2</sup> ), sai (m <sup>2</sup> /m <sup>2</sup> ), stand age (years).
param	A numeric vector of model input parameters (for the right order see <a href="#">param_to_rlwfbrook90</a> ).
pdur	a [1,12]-matrix of precipitation durations (hours) for each month.
soil_materials	A matrix of the 8 soil materials parameters. When imodel = 1 (Mualem-van Genuchten), these refer to: mat, ths, thr, alpha (1/m), npar, ksat (mm/d), tort (-), stonef (-). When imodel = 2 (Clapp-Hornberger): mat, thsat, thetaf, psif (kPa), bexp, kf (mm/d), wetinf (-), stonef (-).
soil_nodes	A matrix of the soil model layers with columns nl (layer number), layer midpoint (m), thickness (mm), mat, psiini (kPa), rootden (-).
precdat	A matrix of precipitation interval data with 6 columns: year, month, day, interval-number (1:precint), prec, mesflp.
output_log	Logical whether to print runtime output to console.
timelimit	Integer to set elapsed time limits for running the model.

**Value**

A list containing the daily and soil layer model outputs, along with an error code of the simulation (see [run\\_LWFB90](#)).

---

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**

<b>Name</b>	<b>Description</b>
czr	Ratio of roughness length to mean height for smooth closed canopies for heights greater than HR when LAI > LAI <sub>crit</sub> .
czs	Ratio of roughness length to mean height for smooth closed canopies for heights less than HS when LAI > LAI <sub>crit</sub> .
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
sloplen	Slope length for downslope-flow. Default: 200
gsc	Rate constant for ground water discharge (remember that a first order groundwater reservoir is placed below the canopy).
gsp	Seepage fraction of groundwater discharge. Default: 0
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. Default: 1
qffc	Quickflow fraction of infiltrating water at field capacity, for value 0 there is no quickflow (bypass or surface runoff).
qfpar	Quickflow shape parameter. Default: 1
qlayer	Number of layers which are considered for generation of surface or source area flow. Default: 0
gwatini	Initial value of groundwater storage. Default: 0
snowini	Initial value of water content of snow pack. 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 duration. Default: 0.5
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). Default: 10
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. Default: 0
prec_corr_statexp	Station exposure situation of prec measurements (passed to <code>correct_prec</code> 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 day using Menzel-model (passed to <code>vegperiod</code> ) Default: 'F'
budburstdoy	Budburst day of year - passed to <code>make_seasLAI</code> . 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 <code>options_b90\$standprop_interp = 'I'</code>
leaffalldoy	Number of days until maximum lai is reached - passed to <code>make_seasLAI</code> Default: 279
leaffalldur	Number of days until minimum lai is reached - passed to <code>make_seasLAI</code> Default: 58
sai	Stem area index. Default: 1
sai_ini	Initial stem area index at the beginning of the simulation. Used when <code>options_b90\$standprop_interp = 'I'</code>
shp_leaffall	Shape parameter for leaf fall phase - passed to <code>make_seasLAI</code> Default: 0.3
shp_budburst	Shape parameter for leaf growth phase - passed to <code>make_seasLAI</code> Default: 3
shp_optdoy	Day of year when optimum value is reached - passed to <code>make_seasLAI</code> Default: 210
lai_doy	Day of year values for lai-interpolation - passed to <code>make_seasLAI</code>
lai_frac	Fractional lai values for lai interpolation, corresponding to <code>lai_doy</code> - passed to <code>make_seasLAI</code> Default: 210
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 <code>options_b90\$standprop_interp = 'I'</code>
maxlai	Maximum projected leaf area index - passed to <code>make_seasLAI</code> Default: 5
radex	Extinction coefficient for solar radiation and net radiation in the canopy. Default: 0.5
cypd	Vapour pressure deficit at which leaf conductance is halved. Default: 2
gmax	Maximum leaf vapour conductance when stomata are fully open. Default: 0.0053
gmin	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 gmax). 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 (hydraulic 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. Default: 30
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
mxxpl	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/m<sup>2</sup>

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

**vappres** daily vapour pressure, kPa

---

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/cm<sup>3</sup>

**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	<i>Annual stand properties of the Solling Beech experimental site 'SLB1'</i>
----------------	--

---

**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, m<sup>2</sup>/m<sup>2</sup>

**sai** stem area index, m<sup>2</sup>/m<sup>2</sup>

**densef** stand density

---

soil_to_param	<i>Split up soil into materials and soil nodes.</i>
---------------	---

---

**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, [35](#)
- slb1\_soil, [36](#)
- slb1\_standprop, [37](#)

approx\_standprop, [3](#), [14](#), [31](#), [34](#)

calc\_globrad, [5](#)

calc\_vegperiod, [6](#)

correct\_prec, [7](#), [31](#), [34](#)

extract\_layer\_output, [8](#)

hydp\_ar\_ff\_b90 (ptfs), [17](#)

hydp\_ar\_hypres, [17](#)

hydp\_ar\_hypres (ptfs), [17](#)

hydp\_ar\_hypres\_tab, [18](#)

hydp\_ar\_hypres\_tab (ptfs), [17](#)

hydp\_ar\_puh2 (ptfs), [17](#)

hydp\_ar\_wessolek\_tab (ptfs), [17](#)

LWFBrook90R, [9](#)

make\_rootden, [10](#), [31](#)

make\_seasLAI, [11](#), [14](#), [31](#), [34](#)

make\_standprop, [13](#)

param\_to\_rlwfbrook90, [14](#), [30](#)

plant\_b90, [15](#)

plant\_coupmodel, [15](#)

plant\_linear, [16](#)

ptfs, [17](#)

r\_lwfbrook90, [14](#), [29](#)

replace\_vecelements, [19](#)

run\_LWFB90, [20](#), [24](#), [25](#), [27](#), [28](#), [30–33](#)

run\_multi\_LWFB90, [21](#), [26](#)

run\_multisite\_LWFB90, [21](#), [24](#)

set\_optionsLWFB90, [20](#), [30](#)

set\_outputLWFB90, [21](#), [32](#)

set\_paramLWFB90, [20](#), [32](#)

slb1\_meteo, [35](#)

slb1\_soil, [36](#)

slb1\_standprop, [37](#)

soil\_to\_param, [21](#), [37](#)

standprop\_yearly\_to\_param, [38](#)

vegperiod, [6](#), [31](#), [34](#)