Package 'TTR.PGM'

April 19, 2025

Type Package

Title Thornley Transport Resistance Plant Growth Model

Version 1.0.0

Maintainer Steven Higgins <steven.higgins@uni-bayreuth.de>

Description An implementation of the Thornley transport resistance plant growth model. The package can be used to simulate plant growth as forced by climate system variables. The package provides methods for formatting forcing variables, simulating growth dynamics and calibrating model parameters. For more information see Higgins et al. (2025) TTR.PGM: An R package for modelling the distributions and dynamics of plants using the Thornley transport resistance plant growth model. Methods in Ecology and Evolution. in press.

License LGPL-3

Date 2025-04-17

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.9), checkmate, abind

LinkingTo Rcpp, RcppArmadillo

SystemRequirements CXX17

Suggests DEoptim, testthat (>= 3.0.0), LaplacesDemon, ROCR, pals, plotrix, rnaturalearth, terra, rmarkdown, knitr

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation yes

Author Steven Higgins [aut, cre, cph], Arne Burkhardt [ctb]

Repository CRAN

Date/Publication 2025-04-19 12:32:05 UTC

Contents

`calc_photo` *calc_photo*

Description

Generates estimates of photosynthetic rates using a Farquar type photosynthesis model.

Usage

```
calc_photo(leaf_temp, photo_active_rad, atmospheric_co2, ppl, pht)
```

Arguments

`leaf_temp`	matrix with ntimesteps rows, nsites columns, containing leaf temperature [°C]
`photo_active_rad`	matrix with ntimesteps rows, nsites columns, containing photosynthetically active radiation [umol*m^(-2)*s^(-1)]
`atmospheric_co2`	matrix with ntimesteps rows, nsites columns, containing atmospheric CO2 partial pressure [Pa]
`ppl`	a list of photosynthetic parameters (see `p3` and `p4`)
`pht`	a string indicating "c3" or "c4" photosynthesis

Details

This function is called internally by `get_input()`.

Value

matrix with ntimesteps rows, nsites columns, containing the calculated photosynthetic rates [umol*m^(-2)*s^(-1)]

`data_input_space_Terminalia_sericea`
TTR space example data

Description

This is an example dataset used for running the TTR.PGM space vignette. It includes both an occurrence data set and data on environmental covariates at the occurrence locations. It is setup so as to demonstrate the use of the `get_input()` function.

Usage

```
data(data_input_space_Terminalia_sericea)
```

Format

A dataframe with environmental data used to build a TTR input list

pts.lon

pts.lat

obs

tmax.CHELSA_tasmax_01_1981.2010_V.2.1

tmax.CHELSA_tasmax_02_1981.2010_V.2.1

tmax.CHELSA_tasmax_03_1981.2010_V.2.1

tmax.CHELSA_tasmax_04_1981.2010_V.2.1

tmax.CHELSA_tasmax_05_1981.2010_V.2.1

tmax.CHELSA_tasmax_06_1981.2010_V.2.1

tmax.CHELSA_tasmax_07_1981.2010_V.2.1

tmax.CHELSA_tasmax_08_1981.2010_V.2.1

tmax.CHELSA_tasmax_09_1981.2010_V.2.1

tmax.CHELSA_tasmax_10_1981.2010_V.2.1

tmax.CHELSA_tasmax_11_1981.2010_V.2.1

tmax.CHELSA_tasmax_12_1981.2010_V.2.1

tmin.CHELSA_tasmin_01_1981.2010_V.2.1

tmin.CHELSA_tasmin_02_1981.2010_V.2.1

tmin.CHELSA_tasmin_03_1981.2010_V.2.1

tmin.CHELSA_tasmin_04_1981.2010_V.2.1

tmin.CHELSA_tasmin_05_1981.2010_V.2.1

tmin.CHELSA_tasmin_06_1981.2010_V.2.1

tmin.CHELSA_tasmin_07_1981.2010_V.2.1

tmin.CHELSA_tasmin_08_1981.2010_V.2.1

tmin.CHELSA_tasmin_09_1981.2010_V.2.1

tmin.CHELSA_tasmin_10_1981.2010_V.2.1

tmin.CHELSA_tasmin_11_1981.2010_V.2.1

tmin.CHELSA_tasmin_12_1981.2010_V.2.1

tavg.CHELSA_tas_01_1981.2010_V.2.1

tavg.CHELSA_tas_02_1981.2010_V.2.1

tavg.CHELSA_tas_03_1981.2010_V.2.1

tavg.CHELSA_tas_04_1981.2010_V.2.1

tavg.CHELSA_tas_05_1981.2010_V.2.1

tavg.CHELSA_tas_06_1981.2010_V.2.1

tavg.CHELSA_tas_07_1981.2010_V.2.1

tavg.CHELSA_tas_08_1981.2010_V.2.1

tavg.CHELSA_tas_09_1981.2010_V.2.1

tavg.CHELSA_tas_10_1981.2010_V.2.1

tavg.CHELSA_tas_11_1981.2010_V.2.1

tavg.CHELSA_tas_12_1981.2010_V.2.1

rain.CHELSA_pr_01_1981.2010_V.2.1

rain.CHELSA_pr_02_1981.2010_V.2.1

rain.CHELSA_pr_03_1981.2010_V.2.1

rain.CHELSA_pr_04_1981.2010_V.2.1

rain.CHELSA_pr_05_1981.2010_V.2.1

rain.CHELSA_pr_06_1981.2010_V.2.1

rain.CHELSA_pr_07_1981.2010_V.2.1

rain.CHELSA_pr_08_1981.2010_V.2.1

rain.CHELSA_pr_09_1981.2010_V.2.1

rain.CHELSA_pr_10_1981.2010_V.2.1

rain.CHELSA_pr_11_1981.2010_V.2.1

rain.CHELSA_pr_12_1981.2010_V.2.1

rsds.CHELSA_rsds_1981.2010_01_V.2.1

rsds.CHELSA_rsds_1981.2010_02_V.2.1

rsds.CHELSA_rsds_1981.2010_03_V.2.1

rsds.CHELSA_rsds_1981.2010_04_V.2.1

rsds.CHELSA_rsds_1981.2010_05_V.2.1

rsds.CHELSA_rsds_1981.2010_06_V.2.1

rsds.CHELSA_rsds_1981.2010_07_V.2.1

rsds.CHELSA_rsds_1981.2010_08_V.2.1

rsds.CHELSA_rsds_1981.2010_09_V.2.1

rsds.CHELSA_rsds_1981.2010_10_V.2.1

rsds.CHELSA_rsds_1981.2010_11_V.2.1

rsds.CHELSA_rsds_1981.2010_12_V.2.1

pet.CHELSA_pet_penman_01_1981.2010_V.2.1

pet.CHELSA_pet_penman_02_1981.2010_V.2.1

pet.CHELSA_pet_penman_03_1981.2010_V.2.1

pet.CHELSA_pet_penman_04_1981.2010_V.2.1

pet.CHELSA_pet_penman_05_1981.2010_V.2.1

pet.CHELSA_pet_penman_06_1981.2010_V.2.1

pet.CHELSA_pet_penman_07_1981.2010_V.2.1

pet.CHELSA_pet_penman_08_1981.2010_V.2.1

pet.CHELSA_pet_penman_09_1981.2010_V.2.1

pet.CHELSA_pet_penman_10_1981.2010_V.2.1

pet.CHELSA_pet_penman_11_1981.2010_V.2.1

pet.CHELSA_pet_penman_12_1981.2010_V.2.1

fc

wp

Source

See technical description

`data_input_time_SA_BFA_BNP` *TTR time example data*

Description

This is an example dataset used for running the TTR.PGM time vignette.

Usage

```
data(data_input_time_SA_BFA_BNP)
```

Format

A dataframe with environmental data used to build a TTR input list

date
tair
tsoil1
tsoil2
tsoil3
tsoil4
srad
moist1
moist2
moist3
moist4
CA
evi
q
sif
osif
msoil
tsoil

Source

See technical description

`data_map` *TTR space example data for plotting maps*

Description

This is an example dataset used for running the TTR.PGM space vignette. It data points. It is used to demonstrate how to plot a map from a fitted TTR.PGM model.

Usage

```
data(data_map)
```

Format

A list as produced by the ‘get_input()‘ function.

Source

This dataset was generated by running ‘get_input()‘ on a global grid of environmental data at 2-degree resolution.

get_input — *Create an input data object for* make_data()

Description

This function takes the input forcing data and formats it for use by the make_data() function. The input forcing data are expected to have the units specified below. All of the forcing data should be in dataframe or matrix form. The following constant dimension sizes are expected:

nspecies : The number of species

nsites : The number of sites

nsteps : The number of timesteps

Currently only one species is supported. Depending on the model configuration different parameters need to be specified. tcur, tnur, tgrowth and tloss always need to be supplied. For the std variant of the process model, rsds has to be specified as well. For other variants photosynthesis rates can be supplied or calculated from tcur, catm and rsds. Soil water content needs to be supplied or it will be calculated from wp, fc, prec and pet. nsoil is optional. fire is currently not supported.

Usage

```
get_input(
  observations,
  tcur,
  tnur,
  tgrowth,
  tloss,
  seconds,
  p3 = TTR.PGM::p3,
  p4 = TTR.PGM::p4,
  lon = NULL,
  lat = NULL,
  rsds = NULL,
  catm = NULL,
  photoc3 = NULL,
  photoc4 = NULL,
  swc = NULL,
  pet = NULL,
  rain = NULL,
  wp = NULL,
  fc = NULL,
  fire = NULL,
  nsoil = NULL
)
```

Arguments

`observations`	The observation data that will be used in the statistical model, there are no checks performed, so make sure to format this in the way the user defined statistical model expects
`tcur`	matrix with ntimesteps rows, nsites columns, containing the temperature associated with photosynthesis [°C]
`tnur`	matrix with ntimesteps rows, nsites columns, containing the temperature associated with nitrogen uptake [°C]
`tgrowth`	matrix with ntimesteps rows, nsites columns, containing the temperature associated with growth [°C]
`tloss`	matrix with ntimesteps rows, nsites columns, containing the temperature associated with biomass loss [°C]
`seconds`	The number of seconds in a time step
`p3`	The parameters to be used to calculate C3 photosynthesis rates (see `p3`)
`p4`	The parameters to be used to calculate C4 photosynthesis rates (see `p4`)
`lon`	vector of length nsite containing the Longitudes of the sites
`lat`	vector of length nsite containing the Latitudes of the sites
`rsds`	matrix with ntimesteps rows, nsites columns, containing short wave downward solar radiation at the surface [W*m^(-2)]
`catm`	matrix with ntimesteps rows, nsites columns, containing Partial pressure of CO2 in the atmosphere [Pa]
`photoc3`	matrix with ntimesteps rows, nsites columns, containing c3 photosynthesis rates [mumol*m^(-2)*s^(-1)]
`photoc4`	matrix with ntimesteps rows, nsites columns, containing c4 photosynthesis rates [mumol*m^(-2)*s^(-1)]
`swc`	matrix with ntimesteps rows, nsites columns, containing soil water content scaled so that wilting point = 0 and field capacity = 100 [%]
`pet`	matrix with ntimesteps rows, nsites columns, containing potential evapotranspiration [kg*m^(-2)*s^(-1)]
`rain`	matrix with ntimesteps rows, nsites columns, containing precipitation [kg*m^(-2)*s^(-1)]
`wp`	vector of length nsites containing wilting points [cm^3/cm^3]
`fc`	vector of length nsites containing field capacities [cm^3/cm^3]
`fire`	matrix with ntimesteps rows, nsites columns, containing binary information on fire occurrence [no units]; currently not supported
`nsoil`	vector of length nsites containing the soil nitrogen content [kg/kg]

Value

A list containing the forcing data for the process model:

`timeseries`	All forcing variables which change with time
`timeinvariant`	All forcing variables which do not change with time
`observations`	Observations as specified above

`get_parms` *get_parms: Pack parameters in an easily accessible list*

Description

This function takes a numeric vector of all process model parameters and returns a representation that is human readable and can be used by `run_ttr()`

Usage

```
get_parms(par, Data, no.structure = FALSE)
```

Arguments

`par`	A numeric vector of parameters as supplied by an external parameter estimation algorithm, e.g. LaplacesDemon or DEoptim
`Data`	data object as returned by the `make_data()` function
`no.structure`	If 'TRUE', only run `interval_parms()` and `unname()` the result, this is convenient when using LaplacesDemon.

Value

A list containing a vector of the alpha parameters and an array of dimension (nbeta, nspecies) where nbeta is the number of beta parameters and nspecies is the number of species, containing the beta parameters per species

`interval_parms` *interval_parms: truncate parameters into bounds*

Description

This function takes a numeric vector and constrains it to specified bounds by reflecting values outside the bounds into the interval. The bounds are defined by a data object returned by `make_data()`

Usage

```
interval_parms(par, Data)
```

Arguments

`par`	A numeric vector of parameters as supplied by LaplacesDemon or DEoptim
`Data`	A data object as returned by `make_data()`

Value

A numeric vector in which each parameter is reflected into bounds.

`make_data` *Create data object for use with the process model*

Description

This function creates an object representing a configuration of the process model including forcing data. This object is used by `run_ttr()`.

Usage

```
make_data(
  input,
  options = standard_options,
  globals = standard_globals,
  bounds = list(alpha = NULL, beta = NULL),
  groups = c(),
  verbose = FALSE
)
```

Arguments

`input`	The forcing data object created by a call to `get_input()`, see specification in its help page
`options`	An options list for running the model, see `options` help page
`globals`	Optional parameter specifying global variables used in the simulation, see `standard_globals` help page
`bounds`	Optional parameter specifying the upper and lower bounds of the model's parameters, see `make_data()`. Bounds for additional parameters included in the statistical model need to be specified here. Bounds of the process model parameters can also be modified from their default values.
`groups`	Optional parameter specifying the groups of process model parameters
`verbose`	Print out a log detailing checks

Details

The data structure produced in this function can be handed to an external parameter estimation algorithm e.g. LaplacesDemon or DEoptim to fit the model.

The bounds of the process model's parameters need to be defined. The user can supply these to `make_data()` or accept the default values returned by a call to `make_data()`. Different model variants use different parameters and a call to `make_data()` returns the default bounds for the model variant specified in `options`.

bounds is a list of two lists; one for alpha (per-process) parameters and one for beta (per-species) parameters. The list bounds is formatted as: bounds = list(alpha = list(alpha_parameter = (lower, upper)), beta = list(beta_parameter = c(lower, upper))), where 'upper' specifies the maximum value a process model parameter can take on, and 'lower' is the corresponding minimum.

The external parameter estimation algorithm should then take Data$bounds[,1] as lower and Data$bounds[,2] as the upper bounds.

In case the user does not wish to estimate some of the parameters in the external optimisation process, the upper and lower bound of a parameter should be set to the same numeric value. In this case, the parameter will not be part of the Data$bounds matrix passed to the external parameter estimation algorithm, and its value is set internally in the `get_parms()` function.

For any group of parameters, e.g., CU_Ns_1 and CU_Ns_2, setting any one of the parameters to c(NA, NA) will set the whole group of parameters to a constant value that will negate the influence of this parameter group on the process model. This effectively switches off the effect of these parameters on the model's processes and removes them from Data$bounds. The constant values the parameters are set to can be observed by inspecting the output of `get_parms()` for a Data object.

The names of the parameters are written as YY_XX_i, where YY specifies the process being considered and XX the environmental driver and i the parameter number in the step or trapezoid function, e.g. CU_swc_1 and CU_swc_2 specify how CU (carbon uptake) is influenced by swc (soil water content) and 1 and 2 specify the lower and upper parameters of the step function.

An example for specifying all process bounds for a Model run with the standard options is given in the object ‘standard_bounds‘.

Value

A ttr data object, a list with the following elements:

`PGF`	Function used by LaplacesDemon for generating initial values.
`options`	Options as specified in the `options` and `standard_options` help pages
`globals`	Globals as specified in the `standard_globals` help page
`n.sites`	Number of sites
`n.species`	Number of species
`n.time`	Number of timesteps
`n.parm`	Number of parameters to be fitted
`n.parm.a`	Number of per-process parameters
`n.parm.b`	Number of per-species parameters
`parm.names`	Names of all parameters for this model configuration
`parm.names.a`	Names of per-process parameter names
`parm.names.b`	Names of per-species parameter names
`mon.names`	Required by LaplacesDemon, always set to "LP"
`out`	Names of the response variables returned by `run_ttr()`
`out.dimnames`	Dimension names for the output returned by `run_ttr()`
`out.dim`	Dimensions for the output returned by `run_ttr()`
`dimnames.beta`	Dimension names for the ttr parameters object produced by `get_parms()`
`bounds`	Bounds for all process model parameters
`timeseries`	See `get_input()`
`timeinvariant`	See `get_input()`

`y`	observations from `get_input()` cast to a matrix
`lonlat`	a matrix with lon and lat columns for the nsites rows
`pos.oe`	Indices of observation error parameters
`pos.pe`	Indices of process error parameters
`parm.trap.groups`	Representation of parameters that form a group for the trapezoid functions
`arguments`	The unprocessed arguments to this function, excluding data

Examples

```
#some dummy data as input
input_data <- get_input(
observations = c(1),
tcur = c(1),
tnur = c(1),
tgrowth = c(1),
tloss = c(1),
seconds = c(1),
lon = c(1),
lat = c(1),
rsds = c(1),
catm = c(1),
pet = c(1),
rain = c(1),
wp = c(1),
fc = c(1)
)
data <- make_data(
  input = input_data,
  options = standard_options,
  globals = standard_globals,
  bounds = list(
    alpha = list(
      #defining a new parameter, e.g. as part of a model function
      my_parameter = c(1,2),
      #overwriting a standard bound
      scale = c(0,500)
    ),
   #no more parameters than needed for the process model in beta, for these standard values.
    beta = list()
  )
)
#all standard values
data <- make_data(input_data, standard_options)
```

`make_swc` *make_swc*

Description

Creates soil water content index from input precipitation, potential evapotranspiration, field capacity and wilting point.

Usage

```
make_swc(rain, pet, fc, wp, seconds, iterations)
```

Arguments

`rain`	matrix with ntimesteps rows, nsites columns, containing precipitation [kg*m^(-2)*s^(-1)]
`pet`	matrix with ntimesteps rows, nsites columns, containing potential evapotranspiration [kg*m^(-2)*s^(-1)]
`fc`	vector of length nsites containing the field capacity [cm^3/cm^3]
`wp`	vector of length nsites containing the wilting point [cm^3/cm^3]
`seconds`	number of seconds in a time step
`iterations`	number of times to run through the data, default is 3

Details

This function is called internally by `get_input()`.

Value

matrix with ntimesteps rows, nsites columns, containing a soil water content index scaled from 0-100

`options` *Option list used in the* `make_data()` *function*

Description

The options list handed to `make_data()`. It specifies the process model options. An example is provided by `standard_options`.

Arguments

`optim`	String specifying the data calculated by the user defined Model function, e.g. one of `c("DEoptim", "LaplacesDemon")`.
`ve`	String specifying the version of the process model. Must be one of `c("std", "fqr", "red", "oak")`.
`steps`	Integer numeric vector specifying the timesteps to output. The process model will run `max(steps)` iterations and the output will contain `length(steps)` output times.

`initial_mass_par`	Boolean specifying if the initial biomass used in the process model should be estimated as a parameter.
`initial_mass`	The initial biomass to use for the process model, when `initial_mass_par` is set to `FALSE`.
`species`	A vector of names of species.
`photo`	A vector specifying the photosynthesis system used by the species. Must be either `c("c3", "c4")` for each species.
`swc_online`	Boolean specifying if soil water content should be calculated online as part of the process model (`TRUE`) or if it is given as a forcing parameter. Currently not supported.
`lc`	Boolean turning light competition simulation in the process model on/off. Currently not supported.
`fire`	Boolean turning fire in the process model on/off. Currently not supported.
`pe`	Boolean turning process error simulation on/off.
`pe_scale`	The process error scaling factor.

Details

Options to be used in the `make_data()` function

`p3` *The default c3 photosynthesis parameters*

Description

The default parameters used by `get_input()` to calculate C3 photosynthesis rates.

Usage

```
p3
```

Format

An object of class `list` of length 32.

p4 — *The default c4 photosynthesis parameters*

Description

The default parameters used by `get_input()` to calculate C4 photosynthesis rates.

Usage

```
p4
```

Format

An object of class `list` of length 43.

predict_ttr — *Make a prediction from a fitted model*

Description

Make a prediction from a fitted TTR.PGM model.

Usage

```
predict_ttr(parms, Model, Data, new.input = NULL, options = NULL, optim = NULL)
```

Arguments

`parms`	A vector of parameters compatible with the data object 'Data' and the user defined Model function
`Model`	A user defined Model function that accepts 'parms' and 'Data' as input
`Data`	A Data object as produced by `make_data()`
`new.input`	If the prediction should use different forcing data from 'Data', this can be an input object as returned by `get_input()`
`options`	If the prediction should use different options from 'Data', this can be supplied as an options list as required by `make_data()`
`optim`	If the user defined Model function tests for 'optim' in options, this can be used to set it. Defaults to the string "no"

Details

`predict_ttr()` returns the data object specified when the option 'Data$options$optim' in the user defined Model is set to "no". See the vignette for an example.

Value

The prediction as specified in the user defined Model function

`run_ttr` *run_ttr: Simulate the TTR model*

Description

run_ttr: Simulate the TTR model

Usage

```
run_ttr(parm, data)
```

Arguments

`parm`	The parameters as returned by the `get_parms()` function
`data`	The model object as defined in the `make_data()` function, see its help page for details

Value

A four-dimensional matrix-object with the dimensions (output_var,species,time,site) where:

`output_var`	refers to the array of output values produced by the process model. For identifiers, check `Data$out` as produced by `make_data()`
`species`	is the nth species
`time`	is the nth output time given in model$options$steps
`site`	is the nth site

`standard_bounds` *Standard values for bounds*

Description

This list contains the standard bounds of parameters used in the Model. See `make_data()` for more information on setting bounds.

Usage

```
standard_bounds
```

Format

An object of class `list` of length 2.

`standard_globals` *Standard values for global parameters*

Description

This vector contains the global parameter values for the process model, which must be supplied to the `make_data()` function. The time components are the units are per modelled time step and the length of these time steps are defined by the parameter seconds in `make_data()`. In the descriptions below M indicates dry mass of structural biomass (either shoot mass MS or root mass MR).

Usage

```
standard_globals
```

Format

An object of class `numeric` of length 13.

Units

mmax Loss coefficient, per timestep (kg / kg M per timestep)

gmax Growth coefficient, in (kg C kg N kg M^-2)^-1 per timestep

KM Size dependency of mass-loss (kg M)

CUmax Carbon uptake rate (kg C / kg shoot M per timestep)

NUmax Nitrogen uptake rate (kg N / kg shoot M per timestep)

KA Size dependency of uptake rates (kg M)

Jc Substrate inhibition coefficient for carbon (kg C / kg M)

Jn Substrate inhibition coefficient for nitrogen (in kg N / kg M)

q Scaling coefficient for substrate transport, no units

RHOc Specific carbon transport resistance, per timestep

RHOn Specific nitrogen transport resistance, per timestep

Fc Fraction of carbon in M (kg C / kg M)

Fn Fraction of nitrogen in M (kg N / kg M) `standard_options` *Standard options for the model*

Description

This list contains the standard options used by `make_data()`. It is provided here as an example. See `options` for more information.

Usage

```
standard_options
```

Format

An object of class `list` of length 9.

Index