Package: plant (via r-universe)

March 13, 2025

Title A Package for Modelling Forest Trait Ecology and Evolution

Version 2.0.0.9000

Maintainer Daniel Falster <daniel.falster@unsw.edu.au>

Description Solves trait, size and patch structured model from (Falster et al. 2016) using either method of characteristics or as stochastic, finite-sized population.

Depends R (>= 4.1.0)

Encoding UTF-8

License GPL-2

LinkingTo Rcpp, BH

Imports Rcpp, R6, loggr, crayon, magrittr, tibble, rlang, dplyr, tidyselect (>= 1.2.0), tidyr, purrr

Suggests testthat, rprojroot, pkgdown, here, deSolve, numDeriv, RcppR6 (>= 0.2.3), knitr, rmarkdown, markdown, bench, ggplot2, ggridges, patchwork, kableExtra

Remotes smbache/loggr, richfitz/RcppR6

VignetteBuilder knitr

URL https://github.com/traitecoevo/plant

BugReports https://github.com/traitecoevo/plant/issues

RoxygenNote 7.3.1

Config/pak/sysreqs libicu-dev

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

RemoteUrl https://github.com/traitecoevo/plant

RemoteRef HEAD

RemoteSha 6dd5cae58f74ce729ac098f14aef741cb489accb

Contents

`build_schedule` *Build Node Schedule*

Description

Build an appropriately refined schedule for node introduction.

Usage

```
build_schedule(
  p,
  env = make_environment(parameters = p),
  ctrl = scm_base_control()
)
```

Arguments

`p`	Parameters object
`env`	Environment object
`ctrl`	Control object

Details

There are control options (within the `Parameters` object) that affect how this function runs, in particular `schedule_nsteps` and `schedule_eps` control how refined the schedule will end up, and `schedule_verbose` controls if details are printed to the screen during construction.

Value

A Parameters object, with schedule components set. The output offspring produced is also available as an attribute `birth_rate`.

Author(s)

Rich FitzJohn

`Control` *Control parameters*

Description

Control parameters that tune various aspects of the numerical solvers.

Usage

```
Control(..., values = list(...))
```

Arguments

`..., values`	Values to initialise the struct with (either as variadic arguments, or as a list, but not both).

`Disturbance_Regime` *Disturbance regime*

Description

Base class of representing patch disturbance

Usage

```
Disturbance_Regime()
```

`environment_type` *Make environment objects for a strategy*

Description

Make environment objects for a strategy

Usage

```
environment_type(type)

make_environment(type = NULL, parameters = NULL, ...)
```

Arguments

`type`	Any strategy name as a string, e.g.: `"FF16"`.
`parameters`	a object
`...`	other arguments passed through

`expand_parameters` *Setup parameters to run resindets or mutants*

Description

The functions expand_parameters and mutant_parameters convert trait values into parametr objects for the model. By default, expand_parameters adds an extra strategy to existing.

Usage

```
expand_parameters(
  trait_matrix,
  p,
  hyperpar = param_hyperpar(p),
  birth_rate_list = 1,
  keep_existing_strategies = TRUE
)

mutant_parameters(..., keep_existing_strategies = FALSE)
```

Arguments

`trait_matrix`	A matrix of traits corresponding to the new types to introduce.
`p`	A `Parameters` object.
`hyperpar`	Hyperparameter function to use. By default links to standard function for this strategy type.
`birth_rate_list`	List object with birth rates for each species in
`keep_existing_strategies`	Should existing residents be retained x. Birth rates can take the form of a scalar (constant) or a vector. In either case birth rates are set as `strategy$birth_rate_y`, however varying birth rates will also have `strategy$birth_rate_x` and
`...`	Arguments to `expand_parameters`

Author(s)

Rich FitzJohn

`fast_control` *Fast Control Defaults*

Description

Sets reasonable defaults for fast numerical calculations

Usage

```
fast_control(base = Control())
```

Arguments

`base`	An optional `Control` object. If omitted, the defaults are used.

Value

A Control object with parameters set.

Author(s)

Rich FitzJohn

`FF16r_hyperpar` *Hyperparameter function for FF16r physiological model*

Description

Hyperparameter function for FF16r physiological model

Usage

```
FF16r_hyperpar(m, s, filter = TRUE)
```

Arguments

`m`	A matrix of trait values, as returned by `trait_matrix`
`s`	A strategy object
`filter`	A flag indicating whether to filter columns. If TRUE, any numbers that are within eps of the default strategy are not replaced.

`FF16r_Individual` *Create a FF16r Plant or Node*

Description

Create a FF16r Plant or Node

Usage

```
FF16r_Individual(s = FF16r_Strategy())
```

Arguments

`s`	A `FF16r_Strategy` object

Examples

```
pl <- FF16r_Individual()
pl$height
```

FF16r_Strategy *Strategy parameters*

Description

Strategy parameters that tune various aspects of the biological model.

Usage

```
FF16r_Strategy(..., values = list(...))
```

Arguments

`..., values`	Values to initialise the struct with (either as variadic arguments, or as a list, but not both).

FF16w_Individual *Create a FF16w Plant or Node*

Description

Create a FF16w Plant or Node

Usage

```
FF16w_Individual(s = FF16w_Strategy())

FF16w_Parameters()
```

Arguments

`s`	A `FF16w_Strategy` object

Examples

```
pl <- FF16w_Individual()
pl$height
```

`FF16w_Strategy` *Strategy parameters*

Description

Strategy parameters that tune various aspects of the biological model.

Usage

```
FF16w_Strategy(..., values = list(...))
```

Arguments

`..., values`	Values to initialise the struct with (either as variadic arguments, or as a list, but not both).

`FF16_Environment` *Create an FF16_Environment object.*

Description

Create an FF16_Environment object.

Usage

```
FF16_Environment(
  light_availability_spline_rescale_usually,
  soil_number_of_depths
)

K93_Environment()

TF24_Environment(
  light_availability_spline_rescale_usually,
  soil_number_of_depths
)
```

Arguments

`light_availability_spline_rescale_usually`	whether to rescale intervals when estimating light environment
`soil_number_of_depths`	Number of soil layers to include

FF16_expand_state — *Add additional state variables to the species component in output of FF16 model.*

Description

Add additional state variables to the species component in output of FF16 model.

Usage

```
FF16_expand_state(tidy_patch_results)
```

Arguments

`tidy_patch_results`	from ‘tidy_patch‘

Value

similar format to input, but with additional columns for additional state variables

FF16_fixed_environment — *Construct a fixed environment for a model*

Description

Construct a fixed environment for a model

Construct a fixed environment for FF16w strategy

Usage

```
FF16_fixed_environment(e = 1, height_max = 150, ...)

FF16w_fixed_environment(e = 1, height_max = 150)

K93_fixed_environment(e = 1, height_max = 300, ...)

TF24_fixed_environment(e = 1, height_max = 150, ...)
```

Arguments

`e`	Value of environment (deafult = 1.0)
`height_max`	= 150.0 maximum possible height in environment
`...`	Additional parameters to be passed to `XXXX_make_environment`, where XXXX referes to model name.

FF16_generate_stand_report

Generates a report on stand grown with FF16 strategy

Description

Builds a detailed report on stand grown with FF16 strategy, based on the template Rmd file provided. The reports are rendered as html files and saved in the specified output folder.

Usage

```
FF16_generate_stand_report(
  results,
  output_file = "FF16_report.html",
  overwrite = FALSE,
  target_ages = NA,
  input_file = system.file("reports", "FF16_report.Rmd", package = "plant"),
  quiet = TRUE
)
```

Arguments

`results`	results of runnning `run_scm_collect`
`output_file`	name of output file
`overwrite`	logical value to determine whether to overwrite existing report
`target_ages`	Patches ages at which to make plots
`input_file`	report script (.Rmd) file to build study report
`quiet`	An option to suppress printing during rendering from knitr, pandoc command line and others.

Value

html file of the rendered report located in the specified output folder.

FF16_hyperpar

Hyperparameter function for FF16 physiological model

Description

Hyperparameter function for FF16 physiological model

Usage

```
FF16_hyperpar(m, s, filter = TRUE)
```

Arguments

`m`	A matrix of trait values, as returned by `trait_matrix`
`s`	A strategy object
`filter`	A flag indicating whether to filter columns. If TRUE, any numbers that are within eps of the default strategy are not replaced.

`FF16_Individual` *Create a FF16 Individual*

Description

Create a FF16 Individual

Usage

```
FF16_Individual(s = FF16_Strategy())
```

Arguments

`s`	A `FF16_Strategy` object

Examples

```
pl <- FF16_Individual()
pl$height
```

`FF16_make_environment` *create a model Environment object*

Description

create a model Environment object

Usage

```
FF16_make_environment(
  light_availability_spline_tol = 1e-04,
  light_availability_spline_nbase = 17,
  light_availability_spline_max_depth = 16,
  light_availability_spline_rescale_usually = TRUE
)

FF16r_make_environment(
  light_availability_spline_tol = 1e-04,
  light_availability_spline_nbase = 17,
  light_availability_spline_max_depth = 16,
  light_availability_spline_rescale_usually = TRUE
)

FF16w_make_environment(
  light_availability_spline_tol = 1e-04,
  light_availability_spline_nbase = 17,
  light_availability_spline_max_depth = 16,
  light_availability_spline_rescale_usually = TRUE,
  soil_number_of_depths = 1,
  soil_initial_state = 0,
  rainfall = 1
)

K93_make_environment(
  light_availability_spline_tol = 1e-04,
  light_availability_spline_nbase = 17,
  light_availability_spline_max_depth = 16,
  light_availability_spline_rescale_usually = TRUE
)

TF24_make_environment(
  light_availability_spline_tol = 1e-04,
  light_availability_spline_nbase = 17,
  light_availability_spline_max_depth = 16,
  light_availability_spline_rescale_usually = TRUE
)
```

Arguments

`light_availability_spline_tol`	Error tolerance of adpative spline method. Deafult is 1e-4.
`light_availability_spline_nbase`	Parameter used in adaptive spline method. Default is 17.
`light_availability_spline_max_depth`	Parameter used in adaptive spline method. Default is 16.
`light_availability_spline_rescale_usually`	whether to rescale intervals when estimating light environment
`soil_number_of_depths`	the number of soil layers
`soil_initial_state`	the initial state of the soil layers
`rainfall`	constant function value for rainfall driver, y = rainfall

FF16_Parameters *Setup an a model system with default or specified parameters*

Description

Setup an a model system with default or specified parameters. This function enables you initialize a model system. Use the model name to start different models.

Usage

```
FF16_Parameters(...)

FF16r_Parameters()

K93_Parameters()

TF24_Parameters(...)
```

Arguments

`...`	Arguments to be passed to the model constructor. These include *‘patch_area‘: Area of idnividfual patch. Only relevant for stochastic model. Default is 1.0m2. *‘max_patch_lifetime‘: The maximum time in years we want to simulate *‘strategies‘: A list of stratgies to simulate. The default is an empty list. *‘strategy_default‘: Values for the default startegy. The default values are those specified in the C++ code for the model. *‘node_schedule_times_default‘: Default vector of times at which to introduce nodes. The default is chosen to have close spacing at the start of the simulation. *‘node_schedule_times‘: A list with each element containing the vector of times we want to introduce nodes for each strategy. The default is an empty list. *‘ode_times‘: A vector of patch ages we want the ode solver to stop at

Examples

```
p1 <- FF16_Parameters()
p2 <- FF16_Parameters(max_patch_lifetime = 10.0, patch_area = 1.0, strategies = list(FF16_Strategy()), strategy_de
p1 <- TF24_Parameters()
p2 <- TF24_Parameters(max_patch_lifetime = 10.0)
```

FF16_Strategy *Strategy parameters*

Description

Strategy parameters that tune various aspects of the biological model.

Usage

```
FF16_Strategy(..., values = list(...))
```

Arguments

`..., values`	Values to initialise the struct with (either as variadic arguments, or as a list, but not both).

`FF16_test_environment` *Create a test environment for FF16 startegy*

Description

This makes a pretend light environment over the plant height, slightly concave up, whatever.

Usage

```
FF16_test_environment(height, n = 101, light_env = NULL, n_strategies = 1)

FF16r_test_environment(height, n = 101, light_env = NULL, n_strategies = 1)

FF16w_test_environment(height, n = 101, light_env = NULL, n_strategies = 1)

K93_test_environment(height, n = 101, light_env = NULL, n_strategies = 1)

TF24_test_environment(height, n = 101, light_env = NULL, n_strategies = 1)
```

Arguments

`height`	top height of environment object
`n`	number of points
`light_env`	function for light environment in test object
`n_strategies`	number of strategies for test environment

Examples

```
environment <- plant:::FF16_test_environment(10)
environment <- plant:::FF16w_test_environment(10)
environment <- plant:::K93_test_environment(10)
environment <- plant:::TF24_test_environment(10)
```

grow_individual_to_size

Grow individual to given size

Description

Grow an individual up to particular sizes.

Usage

```
grow_individual_to_size(
  individual,
  sizes,
  size_name,
  env,
  time_max = Inf,
  warn = TRUE,
  filter = FALSE
)

grow_individual_to_height(individual, heights, env, ...)
```

Arguments

`individual`	An `Individual` object.
`sizes`	A vector of sizes to grow the plant to (increasing in size).
`size_name`	The name of the size variable within `individual$rates` (e.g., height).
`env`	An `Environment` object.
`time_max`	Time to run the ODE out for – only exists to prevent an infinite loop (say, on an unreachable size).
`warn`	Warn if requesting a plant that is too large?
`filter`	Filter individuals that are too large?
`heights`	Heights (when using `grow_individual_to_height`)
`...`	Additional parameters passed to `grow_individual_to_size`.

Value

A list with elements `time` (the time that a given size was reached), `state` (the *ode state* at these times, as a matrix) and `plant` a list of individuals grown to the appropriate size. Note that if only a single size is given, a list of length 1 is returned.

`grow_individual_to_time` *Grow a plant*

Description

Grow a plant up for particular time lengths

Usage

```
grow_individual_to_time(individual, times, env)
```

Arguments

`individual`	An `Individual` object
`times`	A vector of times
`env`	An `Environment` object

`Individual` *Individual object*

Description

Individual object

Usage

```
Individual(T, E)
```

Arguments

`T`	String containing class of plant strategy to create;
`E`	String containing class of environmnet to create;

`integrate_over_size_distribution` *Integrate over the size distribution for each species at each time point, to give totals of each variable Integrations are performed using trapezium integration*

Description

Integrate over the size distribution for each species at each time point, to give totals of each variable Integrations are performed using trapezium integration

Usage

```
integrate_over_size_distribution(tidy_species_data)
```

Arguments

`tidy_species_data`	output of either ‘tidy_patch‘ or ‘tidy_species‘

Value

a tibble whose columns provide metrics on integrated totals for each variable for each species at each time

`Internals` *Extract Internals from plant object*

Description

Internals class holding vectors of states, thier associated rates and auxiliary (aux) state which is calculated from the state through running compute_rates

Usage

```
Internals(s_size, a_size)
```

Arguments

`s_size`	???
`a_size`	???

`interpolate_to_heights` *Interpolate data on size distributions for each species to specific heights at every time point*

Description

Interpolate data on size distributions for each species to specific heights at every time point

Usage

```
interpolate_to_heights(tidy_species_data, heights, method = "natural")
```

Arguments

`tidy_species_data`	output of either ‘tidy_patch‘ or ‘tidy_species‘
`heights`	heights to interpolate to
`method`	Method for interpolation. For more info see help on stats::spline

Value

Returns a tibble of similar format to input, but with all outputs interpolated to specified hieghts.

`interpolate_to_times` *Interpolate data on size distributions for each species to specific timer points, using specified interpolation method*

Description

Interpolate data on size distributions for each species to specific timer points, using specified interpolation method

Usage

```
interpolate_to_times(tidy_species_data, times, method = "natural")
```

Arguments

`tidy_species_data`	output of either ‘tidy_patch‘ or ‘tidy_species‘
`times`	times to interpolate to
`method`	Method for interpolation. For more info see help on stats::spline

Value

Returns a tibble of similar format to input, but with all outputs interpolated to specified hieghts.

`Interpolator` *Spline interpolation*

Description

Spline interpolation

Usage

```
Interpolator()
```

`K93_hyperpar` *Hyperparameter function for K93 physiological model*

Description

Hyperparameter function for K93 physiological model

Usage

```
K93_hyperpar(m, s, filter = TRUE)
```

Arguments

`m`	A matrix of trait values, as returned by `trait_matrix`
`s`	A strategy object
`filter`	A flag indicating whether to filter columns. If TRUE, any numbers that are within eps of the default strategy are not replaced.

`K93_Individual` *Create a K93 Individual or Node*

Description

Create a K93 Individual or Node

Usage

```
K93_Individual(s = K93_Strategy())
```

Arguments

`s`	A `K93_Strategy` object

Examples

```
pl <- K93_Individual()
pl$height
```

`K93_Strategy` *Strategy parameters*

Description

Strategy parameters that tune various aspects of the biological model.

Usage

```
K93_Strategy(..., values = list(...))
```

Arguments

`..., values`	Values to initialise the struct with (either as variadic arguments, or as a list, but not both).

`make_FF16r_hyperpar` *Hyperparameters for FF16r physiological model*

Description

Hyperparameters for FF16r physiological model

Usage

```
make_FF16r_hyperpar(
  lma_0 = 0.1978791,
  B_kl1 = 0.4565855,
  B_kl2 = 1.71,
  rho_0 = 608,
  B_dI1 = 0.01,
  B_dI2 = 0,
  B_ks1 = 0.2,
  B_ks2 = 0,
  B_rs1 = 4012,
  B_rb1 = 2 * 4012,
  B_f1 = 3,
  narea = 0.00187,
  narea_0 = 0.00187,
  B_lf1 = 5120.738 * 0.00187 * 24 * 3600/1e+06,
  B_lf2 = 0.5,
  B_lf3 = 0.04,
  B_lf4 = 21000,
  B_lf5 = 1,
  k_I = 0.5,
  latitude = 0
)
```

Arguments

`lma_0`	Central (mean) value for leaf mass per area [kg /m2]
`B_kl1`	Rate of leaf turnover at lma_0 [/yr]
`B_kl2`	Scaling slope for phi in leaf turnover [dimensionless]
`rho_0`	Central (mean) value for wood density [kg /m3]
`B_dI1`	Rate of instantaneous mortality at rho_0 [/yr]
`B_dI2`	Scaling slope for wood density in intrinsic mortality [dimensionless]
`B_ks1`	Rate of sapwood turnover at rho_0 [/yr]
`B_ks2`	Scaling slope for rho in sapwood turnover [dimensionless]
`B_rs1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]
`B_rb1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]
`B_f1`	Cost of seed accessories per unit seed mass [dimensionless]
`narea`	nitrogen per leaf area [kg / m2]
`narea_0`	central (mean) value for nitrogen per leaf area [kg / m2]
`B_lf1`	Potential CO_2 photosynthesis at average leaf nitrogen [mol / d / m2]
`B_lf2`	Curvature of leaf photosynthetic light response curve [dimensionless]
`B_lf3`	Quantum yield of leaf photosynthetic light response curve [dimensionless]
`B_lf4`	CO_2 respiration per unit leaf nitrogen [mol / yr / kg]
`B_lf5`	Scaling exponent for leaf nitrogen in maximum leaf photosynthesis [dimensionless]
`k_I`	light extinction coefficient [dimensionless]
`latitude`	degrees from equator (0-90), used in solar model [deg]

`make_FF16w_hyperpar` *Hyperparameters for FF16w physiological model*

Description

Hyperparameters for FF16w physiological model

Hyperparameter function for FF16w physiological model

Usage

```
make_FF16w_hyperpar(
  lma_0 = 0.1978791,
  B_kl1 = 0.4565855,
  B_kl2 = 1.71,
  rho_0 = 608,
  B_dI1 = 0.01,
  B_dI2 = 0,
  B_ks1 = 0.2,
  B_ks2 = 0,
  B_rs1 = 4012,
  B_rb1 = 2 * 4012,
  B_f1 = 3,
  narea = 0.00187,
  narea_0 = 0.00187,
  B_lf1 = 5120.738 * 0.00187 * 24 * 3600/1e+06,
  B_lf2 = 0.5,
  B_lf3 = 0.04,
  B_lf4 = 21000,
  B_lf5 = 1,
  k_I = 0.5,
  latitude = 0
)

FF16w_hyperpar(m, s, filter = TRUE)
```

Arguments

`lma_0`	Central (mean) value for leaf mass per area [kg /m2]
`B_kl1`	Rate of leaf turnover at lma_0 [/yr]
`B_kl2`	Scaling slope for phi in leaf turnover [dimensionless]
`rho_0`	Central (mean) value for wood density [kg /m3]
`B_dI1`	Rate of instantaneous mortality at rho_0 [/yr]
`B_dI2`	Scaling slope for wood density in intrinsic mortality [dimensionless]
`B_ks1`	Rate of sapwood turnover at rho_0 [/yr]
`B_ks2`	Scaling slope for rho in sapwood turnover [dimensionless]
`B_rs1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]
`B_rb1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]
`B_f1`	Cost of seed accessories per unit seed mass [dimensionless]
`narea`	nitrogen per leaf area [kg / m2]
`narea_0`	central (mean) value for nitrogen per leaf area [kg / m2]
`B_lf1`	Potential CO_2 photosynthesis at average leaf nitrogen [mol / d / m2]
`B_lf2`	Curvature of leaf photosynthetic light response curve [dimensionless]
`B_lf3`	Quantum yield of leaf photosynthetic light response curve [dimensionless]

`B_lf4`	CO_2 respiration per unit leaf nitrogen [mol / yr / kg]
`B_lf5`	Scaling exponent for leaf nitrogen in maximum leaf photosynthesis [dimensionless]
`k_I`	light extinction coefficient [dimensionless]
`latitude`	degrees from equator (0-90), used in solar model [deg]
`m`	A matrix of trait values, as returned by `trait_matrix`
`s`	A strategy object
`filter`	A flag indicating whether to filter columns. If TRUE, any numbers that are within eps of the default strategy are not replaced.

`make_FF16_hyperpar` *Hyperparameters for FF16 physiological model*

Description

Hyperparameters for FF16 physiological model

Usage

```
make_FF16_hyperpar(
  lma_0 = 0.1978791,
  B_kl1 = 0.4565855,
  B_kl2 = 1.71,
  rho_0 = 608,
  B_dI1 = 0.01,
  B_dI2 = 0,
  B_ks1 = 0.2,
  B_ks2 = 0,
  B_rs1 = 4012,
  B_rb1 = 2 * 4012,
  B_f1 = 3,
  narea = 0.00187,
  narea_0 = 0.00187,
  B_lf1 = 5120.738 * 0.00187 * 24 * 3600/1e+06,
  B_lf2 = 0.5,
  B_lf3 = 0.04,
  B_lf4 = 21000,
  B_lf5 = 1,
  k_I = 0.5,
  latitude = 0
)
```

Arguments

`lma_0`	Central (mean) value for leaf mass per area [kg /m2]
`B_kl1`	Rate of leaf turnover at lma_0 [/yr]
`B_kl2`	Scaling slope for phi in leaf turnover [dimensionless]
`rho_0`	Central (mean) value for wood density [kg /m3]
`B_dI1`	Rate of instantaneous mortality at rho_0 [/yr]
`B_dI2`	Scaling slope for wood density in intrinsic mortality [dimensionless]
`B_ks1`	Rate of sapwood turnover at rho_0 [/yr]
`B_ks2`	Scaling slope for rho in sapwood turnover [dimensionless]
`B_rs1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]
`B_rb1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]
`B_f1`	Cost of seed accessories per unit seed mass [dimensionless]
`narea`	nitrogen per leaf area [kg / m2]
`narea_0`	central (mean) value for nitrogen per leaf area [kg / m2]
`B_lf1`	Potential CO_2 photosynthesis at average leaf nitrogen [mol / d / m2]
`B_lf2`	Curvature of leaf photosynthetic light response curve [dimensionless]
`B_lf3`	Quantum yield of leaf photosynthetic light response curve [dimensionless]
`B_lf4`	CO_2 respiration per unit leaf nitrogen [mol / yr / kg]
`B_lf5`	Scaling exponent for leaf nitrogen in maximum leaf photosynthesis [dimensionless]
`k_I`	light extinction coefficient [dimensionless]
`latitude`	degrees from equator (0-90), used in solar model [deg]

`make_hyperpar` *Hyperparameters for FF16 physiological model*

Description

Set a suitable hyperparameter function for chosen physiological model

Usage

```
make_hyperpar(type)

param_hyperpar(parameters)

hyperpar(type)
```

Arguments

`type`	Any strategy name as a string, e.g.: `"FF16"`.
`parameters`	A parameters object

make_K93_hyperpar *Hyperparameters for K93 physiological model*

Description

Construct hyperparameter object for K93 physiological model

Usage

```
make_K93_hyperpar(
  b_0 = 0.059,
  b_1 = 0.012,
  b_2 = 0.00041,
  c_0 = 0.008,
  c_1 = 0.00044,
  d_0 = 0.00073,
  d_1 = 0.044,
  eta = 12,
  k_I = 0.01
)
```

Arguments

`b_0`	Growth intercept year-1
`b_1`	Growth asymptote year-1.(ln cm)-1
`b_2`	Growth suppression rate m2.cm-2.year-1
`c_0`	Mortality intercept year-1
`c_1`	Mortality suppression rate m2.cm-2.year-1
`d_0`	Recruitment rate (cm2.year-1)
`d_1`	Recruitment suppression rate (m2.cm-2)
`eta`	Crown shape parameter
`k_I`	Extinction coefficient used when estimating competitive effect

make_patch *Reconstruct a patch*

Description

Functions for reconstructing a Patch from an SCM

Usage

```
make_patch(
  state,
  p,
  env = make_environment(parameters = p),
  ctrl = scm_base_control()
)

scm_state(i, x)

scm_patch(i, x)
```

Arguments

`state`	State object created by `scm_state`
`p`	Parameters object
`env`	Environment object (defaults to FF16_Environment)
`ctrl`	Control object
`i`	Index to extract from `x`
`x`	Result of running `run_scm_collect`

`make_scm_integrate` *Integrate SCM variables*

Description

Create a function that allows integrating aggregate properties of the SCM system.

Usage

```
make_scm_integrate(obj)
```

Arguments

`obj`	An object from `run_scm` or `run_scm_collect`

Details

The workflow here is to run an SCM to create an SCM by running `run_scm`, or a set of data from `run_scm_collect` and then reconstitute all the intermediate bits of data so that an any variable that `PlantPlus` tracks can be integrated out. Because the pre-processing step is reasonably slow, this function returns a function that takes a variable name and integrates it.

make_TF24_hyperpar *Hyperparameters for TF24 physiological model*

Description

Hyperparameters for TF24 physiological model

Usage

```
make_TF24_hyperpar(
  lma_0 = 0.1978791,
  B_kl1 = 0.4565855,
  B_kl2 = 1.71,
  rho_0 = 608,
  B_dI1 = 0.01,
  B_dI2 = 0,
  B_ks1 = 0.2,
  B_ks2 = 0,
  B_rs1 = 4012,
  B_rb1 = 2 * 4012,
  B_f1 = 3,
  narea = 0.00187,
  narea_0 = 0.00187,
  B_lf1 = 5120.738 * 0.00187 * 24 * 3600/1e+06,
  B_lf2 = 0.5,
  B_lf3 = 0.04,
  B_lf4 = 21000,
  B_lf5 = 1,
  k_I = 0.5,
  latitude = 0
)
```

Arguments

`lma_0`	Central (mean) value for leaf mass per area [kg /m2]
`B_kl1`	Rate of leaf turnover at lma_0 [/yr]
`B_kl2`	Scaling slope for phi in leaf turnover [dimensionless]
`rho_0`	Central (mean) value for wood density [kg /m3]
`B_dI1`	Rate of instantaneous mortality at rho_0 [/yr]
`B_dI2`	Scaling slope for wood density in intrinsic mortality [dimensionless]
`B_ks1`	Rate of sapwood turnover at rho_0 [/yr]
`B_ks2`	Scaling slope for rho in sapwood turnover [dimensionless]
`B_rs1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]
`B_rb1`	CO_2 respiration per unit sapwood volume [mol / yr / m3]

`B_f1`	Cost of seed accessories per unit seed mass [dimensionless]
`narea`	nitrogen per leaf area [kg / m2]
`narea_0`	central (mean) value for nitrogen per leaf area [kg / m2]
`B_lf1`	Potential CO_2 photosynthesis at average leaf nitrogen [mol / d / m2]
`B_lf2`	Curvature of leaf photosynthetic light response curve [dimensionless]
`B_lf3`	Quantum yield of leaf photosynthetic light response curve [dimensionless]
`B_lf4`	CO_2 respiration per unit leaf nitrogen [mol / yr / kg]
`B_lf5`	Scaling exponent for leaf nitrogen in maximum leaf photosynthesis [dimensionless]
`k_I`	light extinction coefficient [dimensionless]
`latitude`	degrees from equator (0-90), used in solar model [deg]

`NodeSchedule` *Schedule of node introduction times*

Description

Schedule of node introduction times

Usage

```
NodeSchedule(n_species)
```

Arguments

`n_species`	number of species

`node_schedule_times_default` *Generate Default Node Introduction Times*

Description

Generate a suitable set of default node introduction times, biased so that introductions are more closely packed at the beginning of time, become increasingly spread out.

Usage

```
node_schedule_times_default(max_time)
```

Arguments

`max_time`	Time to generate introduction times up to (the last introduction time will be at least `max_time`).

Details

The reason for the stepped distribution is to keep step sizes as series of doublings. Doing this limits the range of possible introduction times from an infinite set of possible values to a very limited subset of values (based on combinations of 1, 0.5, 0.25, 0.125 etc). The reason for doing this is to minimise the number of unique introduction times across all species. The ODE stepper needs to stop at each point where a node is introduced. If each species was selecting a bunch of points that was essentially unique (compared to those selected for all other species), the number of unique node introductions times could get very large, requiring more ODE steps.

Value

Vector of introduction times.

Author(s)

Rich FitzJohn, adapted from original C++ code by Daniel S. Falster.

`No_Disturbance` *No disturbance regime No_Disturbance_Regime control object*

Description

A disturbance-free regime for running individual patches

Usage

```
No_Disturbance()
```

`OdeControl` *ODE Control parameters*

Description

Control parameters for the ode system

Usage

```
OdeControl(..., values = list(...))
```

Arguments

`..., values`	Values to initialise the struct with (either as variadic arguments, or as a list, but not both).

optimise_individual_rate_at_size_by_trait

The function ‘optimise_individual_rate_at_height_by_trait‘ and ‘optimise_individual_rate_at_size_by_trait‘ solve for the maximum of some rate (e.g. growth rate) at a specified height within the interval of the bounds of a given trait

Description

The function ‘optimise_individual_rate_at_height_by_trait‘ and ‘optimise_individual_rate_at_size_by_trait‘ solve for the maximum of some rate (e.g. growth rate) at a specified height within the interval of the bounds of a given trait

Usage

```
optimise_individual_rate_at_size_by_trait(
  type = "FF16",
  bounds,
  log_scale = TRUE,
  tol = 0.001,
  size = 1,
  size_name = "height",
  rate = size_name,
  params = scm_base_parameters(type),
  env = make_environment(type),
  hyperpars = hyperpar(type),
  set_state_directly = FALSE
)

optimise_individual_rate_at_height_by_trait(..., height = 1)
```

Arguments

`type`	The type of model to use (e.g. "FF16"). Defaults to "FF16"
`bounds`	A vector giving the lower and upper bounds of the trait
`log_scale`	Should the trait be optimised on a log scale? Defaults to TRUE
`tol`	The tolerance for the optimisation
`size`	The size of the individual to optimise the rate at
`size_name`	The name of the size variable specified by `size`
`rate`	The name of the rate to optimise. Defaults to `size_name`
`params`	The parameters of the model
`env`	The environment of the model
`hyperpars`	The hyperparameter function of the model

`set_state_directly`	If TRUE, set the state directly to the size, otherwise grows the plant to that size. Defaults to FALSE
`...`	Additional parameters passed to `optimise_individual_rate_at_size_by_trait`
`height`	Heigh at which grow is optimsied. Defaults to 1

Author(s)

Isaac Towers, Daniel Falster and Andrew O'Reilly-Nugent

`plant_log_console` *Activate logging with loggr*

Description

Activate logging with loggr

Usage

```
plant_log_console(
  file_name = "console",
  .message = FALSE,
  .warning = FALSE,
  .error = FALSE,
  ...
)
```

Arguments

`file_name`	File to save output (default = console)
`.message`, `.warning`, `.error`	Include messages, warnings or errors? By default (and in contrast to `loggr::log_file` these are disabled here.
`...`	Additional parameters passed to `loggr::log_file`, but *not* `file_name` which is hard coded here to `"console"`.

Details

By default plant prints little information about its progress. This can be modified by enabling logging. A formatter that is different to the default `loggr::log_file` formatter is selected here; it will print additional information that plant's internal logging functions record.

"Schedule" events (splitting) are sent to the DEBUG stream, everything else is sent to INFO. All events have a "routine" field added to them, which is useful if sent to a Redis server (using `loggr.redis`).

plot_size_distribution *Title*

Description

Title

Usage

```
plot_size_distribution(data_species)
```

Arguments

data_species	??

QK *Gauss-Kronrod Quadrature*

Description

Gauss-Kronrod Quadrature

Usage

```
QK(rule)
```

Arguments

rule	Degree of the rule; valid sizes are 15, 21, 31, 41, 51, 61.

resource_compensation_point *Whole plant light compensation point*

Description

Compute the whole plant light compensation point for a single plant.

Usage

```
resource_compensation_point(p, ...)
```

Arguments

`p`	A `PlantPlus`, with strategy, height, etc set.
`...`	Additional arguments that are ignored

Author(s)

Rich FitzJohn

`run_plant_benchmarks` *Check performance on current system using package bench*

Description

Check performance on current system using package bench

Usage

```
run_plant_benchmarks(
  strategy_types = list(FF16 = FF16_Strategy, FF16w = FF16w_Strategy, K93 = K93_Strategy),
   iterations = 1
)
```

Arguments

`strategy_types`	A list of name strategy types to be tests
`iterations`	The number of iterations to be run

Value

A dataframe of results

`run_scm` *Run SCM*

Description

Run the SCM, returning the SCM object for interrogation

Usage

```
run_scm(
  p,
  env = make_environment(parameters = p),
  ctrl = scm_base_control(),
  use_ode_times = FALSE
)
```

Arguments

`p`	Parameters object
`env`	Environment object (defaults to FF16_Environment)
`ctrl`	Control object
`use_ode_times`	Should ODE times be used?

Details

This is the simplest way of using the SCM, probably.

Value

A `SCM` object.

Author(s)

Rich FitzJohn

`run_scm_collect` *Run the SCM, Collecting Output*

Description

Run the SCM model, given a Parameters and NodeSchedule

Usage

```
run_scm_collect(
  p,
  env = make_environment(parameters = p),
  ctrl = scm_base_control(),
  collect_auxiliary_variables = FALSE
)
```

Arguments

`p`	A `Parameters` object
`env`	Environment object (defaults to FF16_Environment)
`ctrl`	Control object
`collect_auxiliary_variables`	Return additional strategy variables (eg competition_effect)

Details

This is mostly a simple wrapper around some of the SCM functions. Not sure if this is how we will generally want to do this. Consider this function liable to change.

Author(s)

Rich FitzJohn

`run_stochastic_collect` *Run a stochastic patch, Collecting Output*

Description

Run a stochastic simulation of a patch, given a Parameters

Usage

```
run_stochastic_collect(
  p,
  env = make_environment(parameters = p),
  ctrl = scm_base_control(),
  random_schedule = TRUE
)
```

Arguments

`p`	A `FF16_Parameters` object
`env`	Environment object
`ctrl`	Control object
`random_schedule`	setting to TRUE causes algorithm to generate a random schedule based on offspring arrival and area.

Details

This one might need to be made differently so that different schedules can be added easily. Not sure if this is how we will generally want to do this. Consider this function liable to change.

Author(s)

Rich FitzJohn

`scm_base_control` *Sensible, fast (ish) SCM control settings*

Description

Hopefully sensible set of parameters for use with the SCM. Turns accuracy down a bunch, makes it noisy, sets up the hyperparameterisation that we most often use.

Usage

```
scm_base_control()
```

Author(s)

Rich FitzJohn

`scm_base_parameters` *Basic default parameters for a given strategy*

Description

Basic default settings for a given strategy, environment only really used for templating initially and will be overloaded later by passing an environment to the SCM API (suggesting perhaps the template could be removed).

Usage

```
scm_base_parameters(type = NA, env = environment_type(type))
```

Arguments

`type`	Any strategy name as a string, e.g.: `"FF16"`.
`env`	And environment object

Author(s)

Rich FitzJohn

seq_log *Sequence in log space*

Description

Sequence in log space

Usage

```
seq_log(from, to, length.out)

seq_log_range(r, length.out)

seq_range(r, length.out)
```

Arguments

`from`	Starting point
`to`	Ending point
`length.out`	Number of points to generate
`r`	range (i.e., c(from, to))

Details

Unlike the billions of options for `seq`, only `length.out` is supported here, and both `from` and `to` must be provided. For completeness, `seq_range` generates a range in non-log space.

Author(s)

Rich FitzJohn

strategy_list *Create a list of Strategies*

Description

Create a list of Strategies or Plants by varying a single trait.

Usage

```
strategy_list(
  x,
  parameters,
  hyperpar = param_hyperpar(parameters),
  birth_rate_list
)
```

Arguments

`x`	Values for the trait. This must be a *matrix*, with column names corresponding to entries in `Strategy` and rows representing different values.
`parameters`	`Parameters` object containing a default strategy to modify. Any hyperparameterisation included will be applied.
`hyperpar`	Hyperparameter function to use. By default links to standard function for this strategy type.
`birth_rate_list`	List object with birth rates for each species in x. Birth rates can take the form of a scalar (constant) or a vector. In either case birth rates are set as `strategy$birth_rate_y`, however varying birth rates will also have `strategy$birth_rate_x` and

---

`TF24_generate_stand_report`
Generates a report on stand grown with TF24 strategy

Description

Builds a detailed report on stand grown with TF24 strategy, based on the template Rmd file provided. The reports are rendered as html files and saved in the specified output folder.

Usage

```
TF24_generate_stand_report(
  results,
  output_file = "TF24_report.html",
  overwrite = FALSE,
  target_ages = NA,
  input_file = system.file("reports", "TF24_report.Rmd", package = "plant"),
  quiet = TRUE
)
```

Arguments

`results`	results of runnning `run_scm_collect`
`output_file`	name of output file
`overwrite`	logical value to determine whether to overwrite existing report
`target_ages`	Patches ages at which to make plots
`input_file`	report script (.Rmd) file to build study report
`quiet`	An option to suppress printing during rendering from knitr, pandoc command line and others.

Value

html file of the rendered report located in the specified output folder.

`TF24_hyperpar` *Hyperparameter function for TF24 physiological model*

Description

Hyperparameter function for TF24 physiological model

Usage

```
TF24_hyperpar(m, s, filter = TRUE)
```

Arguments

`m`	A matrix of trait values, as returned by `trait_matrix`
`s`	A strategy object
`filter`	A flag indicating whether to filter columns. If TRUE, any numbers that are within eps of the default strategy are not replaced.

`TF24_Individual` *Create a TF24 Plant or Node*

Description

Create a TF24 Plant or Node

Usage

```
TF24_Individual(s = TF24_Strategy())
```

Arguments

`s`	A `TF24_Strategy` object

Examples

```
pl <- TF24_Individual()
pl$height
```

`TF24_Strategy` *Strategy parameters*

Description

Strategy parameters that tune various aspects of the biological model.

Usage

```
TF24_Strategy(..., values = list(...))
```

Arguments

`..., values`	Values to initialise the struct with (either as variadic arguments, or as a list, but not both).

`tidy_env` *Turn ‘env‘ component of solver output into a tidy data object*

Description

Turn ‘env‘ component of solver output into a tidy data object

Usage

```
tidy_env(env)
```

Arguments

`env`	a list, the ‘env‘ component of solver output.

Value

a tibble describing the environment in a patch

`tidy_individual` *Turn ‘results‘ of plant solver, when solving individuals into a tidy data object*

Description

Turn ‘results‘ of plant solver, when solving individuals into a tidy data object

Usage

```
tidy_individual(results)
```

Arguments

`results`	plant solver output.

Value

a tibble whose columns provide metrics on each individual over time

`tidy_patch` *Turns output of plant solver into a tidy data object*

Description

Turns output of plant solver into a tidy data object

Usage

```
tidy_patch(results)
```

Arguments

`results`	output of run_scm_collect

Value

a list, containing outputs of plant solver in tidy format

`tidy_species` *Turn ‘species‘ component of plant solver output into a tidy data object*

Description

Turn ‘species‘ component of plant solver output into a tidy data object

Usage

```
tidy_species(data)
```

Arguments

`data`	a list, the ‘species‘ component of plant solver output.

Value

a tibble whose columns provide metrics on each breakpoint in species size distribution

`trait_matrix` *Create trait matrix*

Description

Helper function to create trait matrices suitable for `strategy_list`.

Usage

```
trait_matrix(x, trait_name)
```

Arguments

`x`	Values
`trait_name`	Name of a single trait

Author(s)

Rich FitzJohn

`util_colour_set_opacity` *Make colours transparent*

Description

Make colours transparent

Usage

```
util_colour_set_opacity(col, opacity = 0.5)
```

Arguments

`col`	Vector of colours
`opacity`	Vector of opacities

Examples

```
util_colour_set_opacity("red", seq(0, 1, length.out=6))
util_colour_set_opacity(c("red", "blue"), .5)
```

`Weibull_Disturbance_Regime` *Weibull disturbance regime The frequency of disturbance in a meta-population of patches follows a Weibull distribution*

Description

Weibull_Disturbance_Regime control object.

Usage

```
Weibull_Disturbance_Regime(max_patch_lifetime)
```

Arguments

`max_patch_lifetime`	Maximum lifetime of a patch, determines length of a simulation

Index