Package: CroptimizR (via r-universe)

June 2, 2026

Type Package

Title A Package to Estimate Parameters of Crop Models

Version 1.0.0

Date 2026-01-31

Description The purpose of CroptimizR is to provide functions for estimating crop model parameters from observations of their simulated variables. This process is often referred to as calibration. For that, it offers a generic framework for linking crop models with up-to-date and ad-hoc algorithms, as well as a choice of goodness-of-fit criteria and additional features adapted to the problem of crop model calibration including AgMIP calibration protocol and user-defined sequential multi-step workflow. It facilitates the comparison of different types of methods on different models.

License file LICENSE

URL https://github.com/SticsRPacks/CroptimizR, https://doi.org/10.5281/zenodo.4066451

BugReports https://github.com/SticsRPacks/CroptimizR/issues

Depends R (>= 4.2.0)

Imports BayesianTools, crayon, CroPlotR, dplyr, ggplot2, lhs, lifecycle, lubridate, methods, nloptr, purrr, readxl, rlang, tibble, tictoc, tidyr

Suggests ApsimOnR, covr, devtools, doParallel, gridExtra, knitr, openxlsx, rmarkdown, spelling, SticsOnR, SticsRFiles (>= 1.1.3), testthat (>= 2.1.0)

VignetteBuilder knitr

Remotes github::cran/Matrix@1.6-5, github::hol430/ApsimOnR, github::SticsRPacks/CroPlotR@*release, github::SticsRPacks/SticsOnR@*release

ByteCompile true

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-04-03 12:22:37 UTC

RemoteUrl https://github.com/SticsRPacks/CroptimizR

RemoteRef HEAD

RemoteSha 3a10fb358d72af6ce3af9eeb79f6814e8b11813b

Contents

AIC *Computes AIC for ordinary least squares*

Description

Computes AIC for ordinary least squares

Usage

```
AIC(obs_list, crit_value, param_nb)
```

Arguments

obs_list	List of observed values to use for parameter estimation. A named list (names = situations names) of data.frame containing one column named Date with the dates (Date or POSIXct format) of the different observations and one column per observed variables with either the measured values or NA, if the variable is not observed at the given date. See details section for more information on the list of observations actually used during the parameter estimation procedure.
crit_value	Final value of the estimated criterion
param_nb	Number of estimated parameters

Value

Value of the AIC criterion for ordinary least squares. If called without arguments, returns a named list with element "name" containing the name of the function

AICc *Computes AICc for ordinary least squares*

Description

Computes AICc for ordinary least squares

Usage

```
AICc(obs_list, crit_value, param_nb)
```

Arguments

obs_list	List of observed values to use for parameter estimation. A named list (names = situations names) of data.frame containing one column named Date with the dates (Date or POSIXct format) of the different observations and one column per observed variables with either the measured values or NA, if the variable is not observed at the given date. See details section for more information on the list of observations actually used during the parameter estimation procedure.
crit_value	Final value of the estimated criterion
param_nb	Number of estimated parameters

Value

Value of the AICc criterion for ordinary least squares. If called without arguments, returns a named list with element "name" containing the name of the function and "species" containing "Information criterion"

BIC *Computes BIC for ordinary least squares*

Description

Computes BIC for ordinary least squares

Usage

```
BIC(obs_list, crit_value, param_nb)
```

Arguments

obs_list	List of observed values to use for parameter estimation. A named list (names = situations names) of data.frame containing one column named Date with the dates (Date or POSIXct format) of the different observations and one column per observed variables with either the measured values or NA, if the variable is not observed at the given date. See details section for more information on the list of observations actually used during the parameter estimation procedure.
crit_value	Final value of the estimated criterion
param_nb	Number of estimated parameters

Value

Value of the BIC criterion for ordinary least squares. If called without arguments, returns a named list with element "name" containing the name of the function and "species" containing "Information criterion"

`estim_param` *main function for parameter estimation*

Description

main function for parameter estimation

Usage

```
estim_param(
  obs_list,
  model_function,
  model_options = NULL,
  crit_function = crit_log_cwss,
  optim_method = "nloptr.simplex",
  optim_options = NULL,
  param_info,
  forced_param_values = NULL,
  candidate_param = NULL,
  transform_var = NULL,
  transform_obs = NULL,
  transform_sim = NULL,
  satisfy_par_const = NULL,
  var_to_simulate = NULL,
  info_crit_func = list(CroptimizR::AICc, CroptimizR::AIC, CroptimizR::BIC),
  weight = NULL,
  step = NULL,
  out_dir = getwd(),
  info_level = 1,
  var = lifecycle::deprecated()
)
```

Arguments

`obs_list`	List of observed values to use for parameter estimation. A `named list` (names = situations names) of data.frame containing one column named Date with the dates (Date or POSIXct format) of the different observations and one column per observed variables with either the measured values or NA, if the variable is not observed at the given date. See details section for more information on the list of observations actually used during the parameter estimation procedure.
`model_function`	Crop Model wrapper function to use.
`model_options`	List of options for the Crop Model wrapper (see help of the Crop Model wrapper function used).
`crit_function`	Function implementing the criterion to optimize (optional, see default value in the function signature). See here for more details about the list of proposed criteria.

`optim_method` Name of the parameter estimation method to use (optional, see default value in the function signature). For the moment, can be "simplex" or "dreamzs". See here for a brief description and references on the available methods.

`optim_options` List of options of the parameter estimation method, containing:

- `ranseed` Set random seed so that each execution of estim_param give the same results when using the same seed. If you want randomization, set it to NULL, otherwise set it to a number of your choice (e.g. 1234) (optional, default to NULL, which means random seed)
- specific options depending on the method used. Click on the links to see examples with the simplex and DreamZS methods.
- `out_dir` **[Deprecated]** Definition of `out_dir` in `optim_options` is no longer supported, use the new argument `out_dir` of `estim_param` instead.

`param_info` Information on the parameters to estimate. Either a list containing:

- `ub` and `lb`, named vectors of upper and lower bounds (-Inf and Inf can be used if init_values is provided),
- `default`, named vectors of default values (optional, corresponding parameters are set to these values when the parameter is part of the `candidate_param` list and when it is not estimated ; these values are also used as first initial values when the parameters are estimated)
- `init_values`, a data.frame containing initial values to test for the parameters (optional, if not provided, or if less values than number of repetitions of the minimization are provided, the, or part of the, initial values will be randomly generated using LHS sampling within parameter bounds).

or a named list containing for each parameter:

- `sit_list`, list the groups of situations for which the current estimated parameter must take different values (see here for an example),
- `ub` and `lb`, vectors of upper and lower bounds (one value per group),
- `init_values`, the list of initial values per group (data.frame, one column per group, optional).
- `default`, vector of default values per group (optional, the parameter is set to its default value when it is part of the `candidate_param` list and when it is not estimated ; the default value is also used as first initial value when the parameter is estimated)

`forced_param_values`

Named vector or list, must contain the values (or arithmetic expression, see details section) for the model parameters to force. The corresponding values will be transferred to the model wrapper through its param_values argument during the estimation procedure. Should not include values for estimated parameters (i.e. parameters defined in `param_info` argument), except if they are listed as candidate parameters (see argument `candidate_param`).

`candidate_param`

Names of the parameters, among those defined in the argument param_info, that must only be considered as candidate for parameter estimation (see details section). All parameters included in param_info that are not listed in candidate_param will be estimated.

`transform_var` Named vector of functions to apply both on simulated and observed variables. `transform_var=c(var1=log, var2=sqrt)` will for example apply log-transformation on simulated and observed values of variable var1, and square-root transformation on values of variable var2.

`transform_obs` User function for transforming observations before each criterion evaluation (optional), see details section for more information.

`transform_sim` User function for transforming simulations before each criterion evaluation (optional), see details section for more information.

`satisfy_par_const`
User function for including constraints on estimated parameters (optional), see details section for more information.

`var_to_simulate`
(optional) List of variables for which the model wrapper must return results. By default the wrapper is asked to simulate only the observed variables. However, it may be useful to simulate also other variables, typically when transform_sim and/or transform_obs functions are used. Note however that it is active only if the model_function used handles this argument.

`info_crit_func` Function (or list of functions) to compute information criteria. (optional, see default value in the function signature and here for more details about the list of proposed information criteria.). Values of the information criteria will be stored in the returned list. In case parameter selection is activated (i.e. if the argument candidate_param is defined (see details section)), the first information criterion given will be used. ONLY AVAILABLE FOR THE MOMENT FOR crit_function==crit_ols.

`weight` Weights to use in the criterion to optimize. A function that takes in input a vector of observed values and the name of the corresponding variable and that must return either a single value for the weights for the given variable or a vector of values of length the length of the vector of observed values given in input.

`step` (optional) List that describes the steps of the parameter estimation procedure (see details section). If `NULL`, a single default step will be created using the `estim_param` arguments

`out_dir` Path to the directory where the optimization results will be written. (optional, default to `getwd()`)

`info_level` (optional) Integer that controls the level of information returned and stored by estim_param (in addition to the results automatically provided that depends on the method used). Higher code give more details.

- `0` to add nothing,
- `1` to add criterion and parameters values, and constraint if satisfy_par_const is provided, for each evaluation (element params_and_crit in the returned list),
- `2` to add model results, after transformation if transform_sim is provided, and after intersection with observations, i.e. as used to compute the criterion for each evaluation (element sim_intersect in the returned list),
- `3` to add observations, after transformation if transform_obs is provided, and after intersection with simulations, i.e. as used to compute the criterion for each evaluation (element obs_intersect in the returned list),

- 4 to add all model wrapper results for each evaluation, and all transformations if transform_sim is provided. (elements sim and sim_transformed in the returned list).

`var` **[Deprecated]** `var` is no longer supported, use `var_to_simulate` instead.

Details

Observation used:

In CroptimizR, parameter estimation is based on the comparison between the values of the observed and simulated variables at corresponding dates. Only the situations, variables and dates common to both observations (provided in `obs_list` argument), and simulations returned by the wrapper used, will be taken into account in the parameter estimation procedure. In case where the value of an observed variable is NA for a given situation and date, it will not be taken into account. In case where the value of a simulated variable is NA (or Inf) for a given situation and date for which there is an observation, the optimized criterion will take the NA value, which may stop the procedure, and the user will be warned.

Parameter selection procedure (argument `candidate_param`):

If the `candidate_param` argument is given, a parameter selection procedure following Wallach et al. (2023) will be performed.

The candidate parameters are added one by one (in the given order) to the parameters that MUST be estimated (i.e. the one defined in param_info but not in `candidate_param`). Each time a new candidate is added:

- the parameter estimation is performed and an information criterion is computed (see argument `info_crit_func`)
- if the information criterion is inferior to all the ones obtained before, then the current candidate parameter is added to the list of parameters to estimate

The result includes a summary of all the steps (data.frame `param_selection_steps`).

For an example of this procedure, see the vignette Parameter selection with CroptimizR.

Transformation of simulations and observations (arguments `transform_sim` and `transform_obs`):

The optional argument `transform_sim` must be a function with 4 arguments:

- `model_results`: the list of simulated results returned by the mode_wrapper used
- `obs_list`: the list of observations as given to estim_param function
- `param_values`: a named vector containing the current parameters values proposed by the estimation algorithm
- `model_options`: the list of model options as given to estim_param function

It must return a list of simulated results (same format as this returned by the model wrapper used) that will be used to compute the criterion to optimize.

The optional argument `transform_obs` must be a function with 4 arguments:

- `model_results`: the list of simulated results returned by the mode_wrapper used
- `obs_list`: the list of observations as given to estim_param function
- `param_values`: a named vector containing the current parameters values proposed by the estimation algorithm

- model_options: the list of model options as given to estim_param function

It must return a list of observations (same format as obs_list argument) that will be used to compute the criterion to optimize.

Constraints on estimated parameters (argument satisfy_par_const):

The optional argument satisfy_par_const must be a function with 2 arguments:

- param_values: a named vector containing the current parameters values proposed by the estimation algorithm
- model_options: the list of model options as given to estim_param function

It must return a logical indicating if the parameters values satisfies the constraints (freely defined by the user in the function body) or not.

Model parameters to force (argument forced_param_values):

The optional argument forced_param_values may contain arithmetic expressions to automatically compute the values of some parameters in function of the values of parameters that are estimated (equality constraints). For that, forced_param_values must be a named list. Arithmetic expressions must be R expressions given under the shape of character strings. For example:

forced_param_values = list(p1=5, p2=7, p3="5*p5+p6")

will pass to the model wrapper the value 5 for parameter p1, 7 for parameter p2, and will dynamically compute the value of p3 in function of the values of parameters p5 and p6 iteratively provided by the parameter estimation algorithm. In this example, the parameters p5 and p6 must thus be part of the list of parameters to estimate, i.e. described in the param_info argument.

Multi-steps estimation procedure (argument step):

The argument step is a list of lists used to perform parameter estimation in multiple sequential steps. If provided, each step represents a separate stage in the estimation procedure, allowing different configurations for each step (e.g., different sets of parameters to estimate, different observed variables, different situations, etc.). When multiple steps are defined, the parameter values estimated in one step are used as fixed values in the subsequent step.

Each step is a named list that may contain **any argument** of the estim_param function (e.g. candidate_param, optim_options, ...). Only the arguments that differ from those given to estim_param need to be specified: any element not explicitly defined in a step inherits its value from the corresponding argument of estim_param.

When step **is** used, the set of parameters to estimate and observed variables to use usually differs between steps. For sake of simplicity, a **single global** param_info list can be provided to estim_param (containing bounds, etc. for all parameters that may ever be estimated), and each step specifies explicitly:

- major_param: a vector containing the name of the parameters that **must be estimated** at this step,
- candidate_param (optional): a vector containing the name of the parameters that are **candidates for estimation** at this step,
- obs_var (optional): a vector containing the name of the observed variables to use at this step,
- situation (optional): a vector containing the name of the situations to use at this step.

When step is **not** used (step = NULL), a single-step estimation is performed using the arguments of estim_param. In this case, the list of parameters to be estimated is automatically deduced from

the param_info argument: all parameters defined in param_info are considered for estimation (possibly subject to selection if candidate_param is used).

Suppose the step argument is defined as follows:

```
step <- list()
step[[1]] <- list(
  major_param = c("p1"),
  candidate_param = c("p2"),
  obs_var = c("var1")
)
step[[2]] <- list(
  major_param = c("p3"),
  obs_var = c("var2")
)
```

In this case, the parameter estimation procedure will proceed in **two steps**:

- **Step 1**: Parameter p1 is estimated, while p2 is included in a parameter selection procedure. Only observed variable var1 (from obs_list defined in argument of estim_param) is used.
- **Step 2**: Parameter p3 is estimated, and only observed variable var2 is used. Parameters p1 (and possibly p2, if selected) are fixed at the values estimated in Step 1.

Technical information about parameters (bounds, default values, ...) can be provided **once for all steps** via the global param_info argument of estim_param.

The results of the parameter estimation procedure are stored in the folder out_dir, with a separate subfolder for each step.

Value

prints, graphs and a list containing the results of the parameter estimation, which content depends on the method used and on the values of the info_level argument. All results are saved in the folder out_dir.

See Also

For more details and examples, see the different vignettes in CroptimizR website

---

filter_obs *Filter observation list to exclude situations, variables or dates*

Description

Filter observation list to exclude situations, variables or dates

Usage

```
filter_obs(
  obs_list,
  var = NULL,
  situation = NULL,
  dates = NULL,
  include = FALSE,
  var_names = lifecycle::deprecated(),
  sit_names = lifecycle::deprecated()
)
```

Arguments

obs_list	List of observed values to use for parameter estimation. A `named list` (names = situations names) of data.frame containing one column named Date with the dates (Date or POSIXct format) of the different observations and one column per observed variables with either the measured values or NA, if the variable is not observed at the given date. See details section for more information on the list of observations actually used during the parameter estimation procedure.
var	(optional, if not given all variables will be kept) Vector containing the names of the variables to include or exclude
situation	(optional, if not given all situations will be kept) Vector containing the names of the situations to include or exclude
dates	(optional, if not given all dates will be kept) Vector containing the dates (POSIXct format) to include or exclude
include	(optional, FALSE by default) Flag indicating if the variables / situations / dates listed in inputs must be included (TRUE) or not (FALSE) in the resulting list
var_names	**[Deprecated]** `var_names` is no longer supported, use `var` instead.
sit_names	**[Deprecated]** `sit_names` is no longer supported, use `situation` instead.

Value

obs_list List of filtered observed values (same format as `obs_list` input argument)

See Also

For more detail and examples, see the different vignettes in CroptimizR website

Examples

```
obs_list <- list(
  sit1 = data.frame(
    Date = as.POSIXct(c("2009-11-30", "2009-12-10")),
    var1 = c(1.1, 1.5), var2 = c(NA, 2.1)
  ),
  sit2 = data.frame(
    Date = as.POSIXct(c("2009-11-30", "2009-12-5")),
    var1 = c(1.3, 2)
```

```
  )
)

# Keep only var1
filter_obs(obs_list, var = c("var1"), include = TRUE)

# Exclude observations at date "2009-11-30"
filter_obs(obs_list, dates = as.POSIXct(c("2009-11-30")))
```

`get_agmip_protocol_example` *Get example AgMIP protocol Excel file*

Description

Returns the path to the example AgMIP calibration protocol Excel file shipped with CroptimizR, or copies it to a user-specified location.

Usage

```
get_agmip_protocol_example(path = NULL, overwrite = FALSE)
```

Arguments

`path`	Optional path where to copy the example file. If NULL (default), only returns the path to the file inside the package.
`overwrite`	Logical. Overwrite existing file?

Value

Path to the example Excel file (either inside the package or to the copied file).

`get_agmip_protocol_template` *Get AgMIP protocol Excel template*

Description

Returns the path to the AgMIP calibration protocol Excel template shipped with CroptimizR, and copies it to a user-specified location.

Usage

```
get_agmip_protocol_template(path = ".", overwrite = FALSE)
```

Arguments

path	Path where to copy the template file, or NULL to only return the path to the file inside the package. Defaults to the current working directory.
overwrite	Logical. Overwrite existing file?

Value

The path to the template file (either inside the package or to the copied file).

---

get_obs_var *Get names of observed variables*

Description

Get names of observed variables

Usage

```
get_obs_var(obs_list)
```

Arguments

obs_list	List of observed values to use for parameter estimation. A named list (names = situations names) of data.frame containing one column named Date with the dates (Date or POSIXct format) of the different observations and one column per observed variables with either the measured values or NA, if the variable is not observed at the given date. See details section for more information on the list of observations actually used during the parameter estimation procedure.

Value

Vector of names of observed variables

Likelihoods *Likelihoods*

Description

Provide several likelihoods to estimate parameters using bayesian methods.

Usage

```
likelihood_log_ciidn(sim_list, obs_list)

likelihood_log_ciidn_corr(sim_list, obs_list)
```

Arguments

`sim_list`	List of simulated variables
`obs_list`	List of observed variables

Details

The following log-likelihoods are proposed (see html version for a better rendering of equations):

- `likelihood_log_ciidn`: log transformation of concentrated version of iid normal likelihood
  The concentrated version of iid normal likelihood is:

$$\prod_j (\sum_{i,k} [Y_{ijk} - f_{jk}(X_i;\theta)]^2)^{-(n_j/2+2)}$$

  where Y_{ijk} is the observed value for the k^{th} time point of the j^{th} variable in the i^{th} situation, $f_{jk}(X_i;\theta)$ the corresponding model prediction, and n_j the number of measurements of variable j.
  `likelihood_log_ciidn` computes the log of this equation.
  Here, one assume that all errors (model and observations errors for all variables, dates and situations) are independent, and that the error variance is constant over time but may be different between variables j. These error variances are automatically estimated.

- `likelihood_log_ciidn_corr`: log transformation of concentrated version of iid normal likelihood but with hypothesis of high correlation between errors for different measurements over time
  The concentrated version of iid normal likelihood is:

$$\prod_j (\sum_i [\frac{1}{n_{ij}} \sum_k (Y_{ijk} - f_{jk}(X_i;\theta))^2])^{-(N_j/2+2)}$$

  where Y_{ijk} is the observed value for the k^{th} time point of the j^{th} variable in the i^{th} situation, $f_{jk}(X_i;\theta)$ the corresponding model prediction, N_j the number of situations including at least one observation of variable j, and n_{ij} the number of observation of variable j on situation i.
  `likelihood_log_ciidn_corr` computes the log of this equation.
  Here, one still assume that errors in different situations or for different variables in the same situation are independent. However, errors for different observations over time of the same variable in the same situation are assumed to be highly correlated. In this way, each situation contributes a single term to the overall sum of squared errors regardless of the number of observations which may be useful in case one have situations with very heterogeneous number of dates of observations.

`sim_list` and `obs_list` must have the same structure (i.e. same number of variables, dates, situations, ... use internal function intersect_sim_obs before calling the criterion functions).

Value

The value of the likelihood given the observed and simulated values of the variables.

`load_protocol_agmip` *Load an AgMIP calibration protocol Excel file*

Description

Load an AgMIP calibration protocol Excel file

Usage

```
load_protocol_agmip(protocol_file_path)
```

Arguments

`protocol_file_path`
: Character string. Path to the Excel file describing an AgMIP calibration protocol.

Details

Reads an AgMIP calibration protocol file in Excel format and verifies its structure. The Excel file must follow the AgMIP protocol template structure provided by CroptimizR.

The function loads the following sheets: variables, major_parameters, candidate_parameters, and the optional fixed_or_computed_parameters sheet. It returns them as a structured list suitable for use in `run_protocol_agmip`.

The protocol specification follows the AgMIP methodology (Wallach et al., 2024; Wallach et al., 2025).

Value

A list with the following elements:

step A list of groups, each containing major parameters, candidate parameters, and observed variables.

param_info A list containing parameter bounds (`lb`, `ub`) and default values (`default`).

forced_param_values (Optional) Named vector of parameter values or formulas to fix, if defined in the protocol file. This element is present only if the protocol defines fixed or computed parameters.

Excel template and example

CroptimizR provides helper functions to access a ready-to-use Excel template and a fully worked example:

- `get_agmip_protocol_template` to obtain the official Excel template of the AgMIP protocol. This template must be filled by the user before being used with `load_protocol_agmip()`.
- `get_agmip_protocol_example` to access a complete example used for demonstration and testing.

Both functions can either return the path to the file shipped with the package or copy it to a user-defined location for editing.

Examples

```
# Load the example protocol shipped with the package
protocol_file <- get_agmip_protocol_example()
protocol <- load_protocol_agmip(protocol_file)
```

`ls_criteria` *Criteria to optimize*

Description

Provide several least squares criteria to estimate parameters by minimizing the difference between observed and simulated values of model output variables.

Usage

```
crit_ols(sim_list, obs_list)

crit_wls(sim_list, obs_list, weight)

crit_log_cwss(sim_list, obs_list)

crit_log_cwss_corr(sim_list, obs_list)
```

Arguments

`sim_list`	List of simulated variables
`obs_list`	List of observed variables
`weight`	Weights to use in the criterion to optimize. A function that takes in input a vector of observed values and the name of the corresponding variable and that must return either a single value for the weights for the given variable or a vector of values of length the length of the vector of observed values given in input.

Details

The following criteria are proposed (see html version for a better rendering of equations):

- `crit_ols`: ordinary least squares
  The sum of squared residues for each variable:

$$\sum_{i,j,k} [Y_{ijk} - f_{jk}(X_i; \theta)]^2$$

  where Y_{ijk} is the observed value for the k^{th} time point of the j^{th} variable in the i^{th} situation, $f_{jk}(X_i; \theta)$ the corresponding model prediction.

Using this criterion, one assume that all errors (model and observations errors for all variables, dates and situations) are independent, and that the error variance is constant over time and equal for the different variables j.

- `crit_wls`: weighted least squares
  The weighted sum of squared residues for each variable:

$$\sum_{i,j,k}[\frac{Y_{ijk} - f_{jk}(X_i;\theta)}{w_{i,j,k}}]^2$$

  where Y_{ijk} is the observed value for the k^{th} time point of the j^{th} variable in the i^{th} situation, $f_{jk}(X_i;\theta)$ the corresponding model prediction,
  and $w_{i,j,k}$ a weight.
  Using this criterion, one assume that all errors (model and observations errors for all variables, dates and situations) are independent, and that the error variances are equal to $w_{i,j,k}^2$ j.
- `crit_log_cwss`: log transformation of concentrated version of weighted sum of squares
  The concentrated version of weighted sum of squares is:

$$\prod_j (\frac{1}{n_j}\sum_{i,k}[Y_{ijk} - f_{jk}(X_i;\theta)]^2)^{n_j/2}$$

  where Y_{ijk} is the observed value for the k^{th} time point of the j^{th} variable in the i^{th} situation, $f_{jk}(X_i;\theta)$ the corresponding model prediction, and n_j the number of measurements of variable j.
  `crit_log_cwss` computes the log of this equation.
  Using this criterion, one assume that all errors (model and observations errors for all variables, dates and situations) are independent, and that the error variance is constant over time but may be different between variables j. These error variances are automatically estimated. More details about this criterion are given in Wallach et al. (2011), equation 5.
- `crit_log_cwss_corr`: log transformation of concentrated version of weighted sum of squares with hypothesis of high correlation between errors for different measurements over time
  The original criterion is:

$$\prod_j (\frac{1}{N_j}\sum_i[\frac{1}{n_{ij}}\sum_k(Y_{ijk} - f_{jk}(X_i;\theta))^2])^{N_j/2}$$

  where Y_{ijk} is the observed value for the k^{th} time point of the j^{th} variable in the i^{th} situation, $f_{jk}(X_i;\theta)$ the corresponding model prediction, N_j the number of situations including at least one observation of variable j, and n_{ij} the number of observation of variable j on situation i. .
  `crit_log_cwss_corr` computes the log of this equation.
  Using this criterion, one still assume that errors in different situations or for different variables in the same situation are independent. However, errors for different observations over time of the same variable in the same situation are assumed to be highly correlated. In this way, each situation contributes a single term to the overall sum of squared errors regardless of the number of observations which may be useful in case one have situations with very heterogeneous number of dates of observations. More details about this criterion are given in Wallach et al. (2011), equation 8.

`sim_list` and `obs_list` must have the same structure (i.e. same number of variables, dates, situations, ... use internal function intersect_sim_obs before calling the criterion functions).

Value

The value of the criterion given the observed and simulated values of the variables.

`plot_estimVSinit` *Create plots of estimated versus initial values of the parameters*

Description

Create plots of estimated versus initial values of the parameters

Usage

```
plot_estimVSinit(init_values, est_values, crit, lb, ub, bubble = TRUE)
```

Arguments

`init_values`	Data.frame containing initial values of the parameters for each repetition
`est_values`	Data.frame containing estimated values of the parameters for each repetition
`crit`	Vector containing the minimum value of the criterion for each repetition of the minimization
`lb`	Vector containing the lower bounds of the estimated parameters
`ub`	Vector containing the upper bounds of the estimated parameters
`bubble`	Logical indicating if bubbles of size proportional to the minimum values of the criterion should be plot (TRUE, default value) or not (FALSE).

Details

The number of the repetition that leads to the minimal value of the criterion over all repetitions is written in white (if bubble is TRUE) or in red (if bubble is false) while the other ones are written in black.

Value

A named list containing one plot per parameter

`plot_stats_bars` *Plot bar charts of rRMSE and EF statistics by step and variable*

Description

Creates bar charts for the statistics rRMSE and EF across calibration steps, with one panel per statistic. Variables are displayed on the x-axis and bars are colored according to the step.

Usage

```
plot_stats_bars(stats_per_steps)
```

Arguments

`stats_per_steps`
: A data.frame containing at least the columns:

- `step`: the step name,
- `variable`: the variable name,
- `rRMSE`: root Relative Mean Squared Error,
- `EF`: Efficiency Factor.

Value

A ggplot object displaying the bar charts.

`plot_stats_evolution` *Plot Bias² and MSE evolution by calibration step as diagnostic of the AgMIP Protocol*

Description

Create diagnostic plots showing the evolution of Bias² and MSE statistics across calibration steps for one or several variables. For each variable, the curve is drawn with attenuated color before the calibration step where it is first used, then in normal color from this step onwards. Panels are ordered by the step of first use of each variable.

Usage

```
plot_stats_evolution(stats_per_step, steps_by_var, step_levels = NULL)
```

Arguments

stats_per_step	A data.frame containing at least the following columns: • step (character or factor): calibration step • variable (character): name of the variable • Bias2 (numeric): Bias² statistic • MSE (numeric): MSE statistic
steps_by_var	A named character vector associating each variable (names(steps_by_var)) to the name of the step in which it is (last) used (steps_by_var[variable]).
step_levels	(optional) Character vector giving the global order of steps. If not provided, the order of appearance in stats_per_step$step is used.

Value

A ggplot object with one facet per variable, ordered according to their step of use.

---

plot_valuesVSit *Create plots of parameters and criterion values per iteration or evaluation number*

Description

Create plots of parameters and criterion values per iteration or evaluation number

Usage

```
plot_valuesVSit(
  df,
  param_info,
  iter_or_eval = c("iter", "eval"),
  crit_log = TRUE,
  rep_label = c("begin_end", "begin", "end")
)
```

Arguments

df	Data.frame containing values of parameters (one column per estimated parameter), criterion (crit column), repetition number (rep), iteration number (iter) and evaluation number (eval) (similar to params_and_crit). See Details section for comments about the difference between evaluations and iterations.
param_info	Information on the parameters to estimate. Either a list containing: • ub and lb, named vectors of upper and lower bounds (-Inf and Inf can be used if init_values is provided), • default, named vectors of default values (optional, corresponding parameters are set to these values when the parameter is part of the candidate_param list and when it is not estimated ; these values are also used as first initial values when the parameters are estimated)

- `init_values`, a data.frame containing initial values to test for the parameters (optional, if not provided, or if less values than number of repetitions of the minimization are provided, the, or part of the, initial values will be randomly generated using LHS sampling within parameter bounds).

or a named list containing for each parameter:

- `sit_list`, list the groups of situations for which the current estimated parameter must take different values (see here for an example),
- `ub` and `lb`, vectors of upper and lower bounds (one value per group),
- `init_values`, the list of initial values per group (data.frame, one column per group, optional).
- `default`, vector of default values per group (optional, the parameter is set to its default value when it is part of the `candidate_param` list and when it is not estimated ; the default value is also used as first initial value when the parameter is estimated)

`iter_or_eval` Values of the x axis: "iter" for iteration number, "eval" for evaluation number

`crit_log` If TRUE, consider criterion values in log scale

`rep_label` Indicate if labels for the repetition number must be plotted at both beginning and end of lines ("begin_end"), only at the beginning ("begin") or only at the end ("end")

Details

Evaluation means evaluation of the criterion from proposed values of the parameters by the parameter estimation algorithm. An iteration is reached when an evaluation lead to a better value of the criterion than the previously obtained values. There are thus more evaluations than iterations. The criterion decreases when iteration number increases while it is not the case when evaluation number increases.

Value

A named list containing one plot per parameter and a plot for the criterion.

`plot_valuesVSit_2D` *Create 2D plots of parameters values evolution per iteration or evaluation number*

Description

Create 2D plots of parameters values evolution per iteration or evaluation number

Usage

```
plot_valuesVSit_2D(
  df,
  param_info,
  iter_or_eval = c("eval", "iter"),
  fill = c("crit", "rep"),
  crit_log = TRUE,
  lines = FALSE,
  rep_label = c("begin_end", "begin", "end")
)
```

Arguments

df	Data.frame containing values of parameters (one column per estimated parameter), criterion (crit column), repetition number (rep), iteration number (iter) and evaluation number (eval) (similar to params_and_crit). See Details section for comments about the difference between evaluations and iterations.
param_info	Information on the parameters to estimate. Either a list containing: • ub and lb, named vectors of upper and lower bounds (-Inf and Inf can be used if init_values is provided), • default, named vectors of default values (optional, corresponding parameters are set to these values when the parameter is part of the candidate_param list and when it is not estimated ; these values are also used as first initial values when the parameters are estimated) • init_values, a data.frame containing initial values to test for the parameters (optional, if not provided, or if less values than number of repetitions of the minimization are provided, the, or part of the, initial values will be randomly generated using LHS sampling within parameter bounds). or a named list containing for each parameter: • sit_list, list the groups of situations for which the current estimated parameter must take different values (see here for an example), • ub and lb, vectors of upper and lower bounds (one value per group), • init_values, the list of initial values per group (data.frame, one column per group, optional). • default, vector of default values per group (optional, the parameter is set to its default value when it is part of the candidate_param list and when it is not estimated ; the default value is also used as first initial value when the parameter is estimated)
iter_or_eval	"iter" for plotting the values for each iteration, "eval" for plotting the values for each evaluation
fill	If "crit", colours the points and lines in function of the minimized criterion value, if "rep" colours in function of the repetition number.
crit_log	If TRUE, consider criterion values in log scale
lines	If TRUE add lines between points of a same repetition

`rep_label` Indicate if labels for the repetition number must be plotted at both beginning and end of lines ("begin_end"), only at the beginning ("begin") or only at the end ("end")

Details

Evaluation means evaluation of the criterion from proposed values of the parameters by the parameter estimation algorithm. An iteration is reached when an evaluation lead to a better value of the criterion than the previously obtained values. There are thus more evaluations than iterations. The criterion decreases when iteration number increases while it is not the case when evaluation number increases.

Value

A list containing one plot per parameter pair.

`post_treat_frequentist`

Post-treat results of frequentist methods

Description

Post-treat results of frequentist methods

Usage

```
post_treat_frequentist(optim_options, param_info, optim_results, crit_options)
```

Arguments

`optim_options` List of options of the parameter estimation method, containing:

- `ranseed` Set random seed so that each execution of estim_param give the same results when using the same seed. If you want randomization, set it to NULL, otherwise set it to a number of your choice (e.g. 1234) (optional, default to NULL, which means random seed)
- specific options depending on the method used. Click on the links to see examples with the simplex and DreamZS methods.
- `out_dir` **[Deprecated]** Definition of `out_dir` in `optim_options` is no longer supported, use the new argument `out_dir` of `estim_param` instead.

`param_info` Information on the parameters to estimate. Either a list containing:

- `ub` and `lb`, named vectors of upper and lower bounds (-Inf and Inf can be used if init_values is provided),
- `default`, named vectors of default values (optional, corresponding parameters are set to these values when the parameter is part of the `candidate_param` list and when it is not estimated ; these values are also used as first initial values when the parameters are estimated)

- `init_values`, a data.frame containing initial values to test for the parameters (optional, if not provided, or if less values than number of repetitions of the minimization are provided, the, or part of the, initial values will be randomly generated using LHS sampling within parameter bounds).

or a named list containing for each parameter:

- `sit_list`, list the groups of situations for which the current estimated parameter must take different values (see here for an example),
- `ub` and `lb`, vectors of upper and lower bounds (one value per group),
- `init_values`, the list of initial values per group (data.frame, one column per group, optional).
- `default`, vector of default values per group (optional, the parameter is set to its default value when it is part of the `candidate_param` list and when it is not estimated ; the default value is also used as first initial value when the parameter is estimated)

`optim_results` Results list returned by frequentist method wrappers

`crit_options` List containing several arguments given to `estim_param` function: `param_names`, `obs_list`, `model_function`, `model_options`, `param_info`, `transform_obs`, `transform_sim` that must be passed to main_crit function by the methods wrappers.

Value

Updated results of frequentist method

`post_treat_multi_step` *Post-treat results of multi-step procedure*

Description

Post-treat results of multi-step procedure

Usage

```
post_treat_multi_step(step, optim_results_list)
```

Arguments

`step` List of steps of the multi-step procedure

`optim_results_list`
List of results returned for each step of the multi-step parameter estimation procedure

Value

List of estimated and forced parameters values

`run_protocol_agmip` *Automate the AgMIP Phase IV Calibration protocol*

Description

Automate the AgMIP Phase IV Calibration protocol

Usage

```
run_protocol_agmip(
  obs_list,
  model_function,
  model_options,
  optim_options = list(),
  param_info = NULL,
  forced_param_values = NULL,
  transform_var = NULL,
  transform_obs = NULL,
  transform_sim = NULL,
  satisfy_par_const = NULL,
  var_to_simulate = NULL,
  info_crit_func = list(CroptimizR::AICc, CroptimizR::AIC, CroptimizR::BIC),
  step,
  out_dir = getwd(),
  info_level = 0
)
```

Arguments

`obs_list`	List of observed values to use in the protocol, in `cropr` format. A named list (names = situation names) of data.frames, each containing: • one column named `Date` with the observation dates (in `Date` or `POSIXct` format), • and one column per observed variable, containing either the measured values or `NA` if the variable is not observed at the given date.
`model_function`	Crop Model wrapper function to use.
`model_options`	List of options for the Crop Model wrapper (see help of the Crop Model wrapper function used).
`optim_options`	(optional) List of options controlling the minimization method (Nelder–Mead simplex), containing: • `ranseed`: random seed used to make results reproducible. If `NULL`, the seed is not fixed and results may differ between runs. Otherwise, set it to an integer value (e.g. `1234`). Default is `NULL`.

- nb_rep: number of multi-start repetitions of the minimization. If provided, this value is used for all minimizations in both step6 and step7. The same setting is therefore applied to the whole protocol. By default, this is set to c(10, 5) for step6 (respectively for major-parameter estimation and for each candidate-parameter addition) and to c(20) for step7.
- maxeval: maximum number of evaluations of the minimized criterion per minimization (default: 50000).
- xtol_rel: relative tolerance threshold on parameter values. Minimization stops when parameter values change by less than xtol_rel times the absolute value of the parameter between two successive iterations. The default value is 1e-3, which provides a good compromise between computational cost and numerical accuracy for most applications of the AgMIP protocol.
- additional options can be provided; see ?nl.opts for a complete list.

For debugging or testing purposes (i.e. to simply check that the protocol executes correctly without aiming at meaningful results), the user can use very small values, for example nb_rep = 1 and maxeval = 2, in order to obtain a fast "dry run" of the whole protocol. Such settings must of course be removed for any real calibration experiment.

param_info Information about the parameters to estimate. A list containing:

- ub and lb: named numeric vectors of upper and lower bounds,
- default: named numeric vector of default values (optional).

The names correspond to the parameter names. Default values are used when a parameter is not estimated in the current step (e.g. major or candidate parameter estimated in a subsequent step, candidate parameter that was not selected, etc.), and also as one of the initial values when the parameter is estimated.

forced_param_values
(optional) Named vector or list specifying parameter values to force in the model. It may also contain arithmetic expressions to define equality constraints between parameters (see the Details section of estim_param). In this case, the values to force are computed before each call to the model wrapper and passed through its param_values argument during the estimation procedure. This argument should not include values for parameters that are estimated (i.e. parameters defined in param_info).

transform_var Named vector of functions to apply both on simulated and observed variables. transform_var=c(var1=log, var2=sqrt) will for example apply log-transformation on simulated and observed values of variable var1, and square-root transformation on values of variable var2.

transform_obs (optional) User-defined function to transform observations before each criterion evaluation. See the Details section of estim_param for more information.

transform_sim (optional) User-defined function to transform simulations before each criterion evaluation. See the Details section of estim_param for more information.

satisfy_par_const
(optional) User-defined function to enforce inequality constraints on estimated parameters. See the Details section of estim_param for more information.

`var_to_simulate`
: (optional) List of variables for which the model wrapper must return results. By default the wrapper is asked to simulate only the observed variables. However, it may be useful to simulate also other variables, typically when transform_sim and/or transform_obs functions are used. Note however that it is active only if the model_function used handles this argument.

`info_crit_func`
: Function or list of functions used to compute information criteria (optional; see the default value in the function signature and https://sticsrpacks.github.io/CroptimizR/reference/informat for the list of available criteria).

  The values of all provided information criteria are stored in the returned object. If parameter selection is activated (i.e. if `candidate_param` is provided in at least one step of step6), the first information criterion in the list is used for parameter selection.

`step`
: A list defining the sub-steps for step 6 of the AgMIP Calibration protocol (see Details section).

`out_dir`
: Path to the directory where the optimization results will be written. (optional, default to `getwd()`)

`info_level`
: (optional) Integer controlling how much information is stored during each call to `estim_param()` inside the AgMIP calibration protocol.

  This argument is a direct pass-through to the `info_level` argument of `estim_param()`, and therefore controls the amount of diagnostic information kept in memory during the optimization steps.

  Because the AgMIP protocol may involve a large number of successive calibrations, the default value is set to `0` in order to limit memory usage and disk storage.

  However, note that:

  - Setting `info_level = 0` disables the storage of intermediate values of the simplex and therefore prevents the generation of diagnostic plots such as `ValuesVSIt.pdf` and `ValuesVSIt_2D.pdf`.
  - A value of at least `1` is required to enable these diagnostic outputs.

  Higher values provide increasingly detailed information (simulations, observations, full model outputs) for each evaluation, but may lead to very large memory consumption and should therefore be used with caution inside the AgMIP protocol.

  See `estim_param` for a detailed description of the different levels.

Details

The AgMIP Phase IV Calibration protocol:

The AgMIP Phase IV Calibration protocol is thoroughly described in Wallach et al. (2024) and Wallach et al. (2025).

This protocol consists of two successive steps, called step6 and step7.

Step6 consists in a sequential parameter estimation by groups of variables. For each group of variables, parameters are estimated by ordinary least squares (OLS) using a multi-start Nelder–Mead optimization (i.e. several minimizations starting from different initial values). Once estimated, parameters are fixed to their estimated values for the subsequent steps.

For each group of variables, the user defines:

- a set of *major parameters*, supposed to mainly reduce bias for these variables,
- and a set of *candidate parameters*, expected to explain variability between environments.

Candidate parameters should be ordered, as far as possible, by decreasing expected importance. For each group of variables, candidate parameters are progressively added to the list of parameters to estimate, and are retained only if they improve an information criterion (corrected Akaike Information Criterion by default). If a candidate parameter is not selected, it is fixed to its default value.

By default, the estimation of the major parameters for a given step is performed using 10 multi-start repetitions. When candidate parameters are considered, 5 additional multi-start repetitions are performed each time a new candidate parameter is added to the set of parameters to estimate.

Step7 consists in re-estimating all parameters selected during step6 using all available observations, by weighted least squares (WLS). The weights are set to the estimated standard deviation of the model error for each variable, as obtained at the end of step6.

The WLS minimization is performed using a multi-start Nelder–Mead simplex algorithm with 20 repetitions by default. The first two repetitions are initialized respectively from: (i) the parameter values estimated at the end of step6, and (ii) the default parameter values. The remaining repetitions are initialized from parameter values randomly drawn within their respective bounds.

Loading protocol definitions from Excel:

Protocol definitions (`step`, `param_info`, `forced_param_values`) can either be:

- Created directly in R (as detailed in sections and examples below), or
- Loaded from an Excel file using `load_protocol_agmip`.

Helper functions `get_agmip_protocol_template` and `get_agmip_protocol_example` are provided to obtain a ready-to-use template or a fully worked example of an AgMIP calibration protocol, respectively. See the vignette *agmip_calibration_protocol* for a complete, step-by-step workflow.

Describing step6 (argument `step`):

The argument `step` is a list of lists describing the successive sub-steps to apply in step6 of the AgMIP protocol. Each sub-step corresponds to a group of variables (e.g. phenology, biomass, etc.).

Each element of `step` is a named list that must contain:

- `obs_var`: a character vector giving the names of the observed variables to use at this step,
- optionally, `major_param`: a character vector giving the names of the major parameters to estimate at this step,
- optionally, `candidate_param`: a character vector giving the names of the candidate parameters.

At least one of `major_param` or `candidate_param` must be provided. If `candidate_param` is not provided, only the major parameters are estimated for this step. If `major_param` is not provided, the step only performs candidate-parameter selection.

The name of each list element is optional, but it is recommended to use the name of the corresponding group of variables. This name is used in printed outputs and in the results.

Technical information about parameters (bounds, default values, ...), the observation list in `cropr` format, optimization options, forced parameter values, transformation functions, functions defining equality constraints, the information criterion function, etc., can be provided **once for all steps** via the corresponding arguments of `run_protocol_agmip` (`param_info`, `obs_list`, ...).

If the user wants this information to be specific to a given step, it can also be provided inside the corresponding step description, using the same argument names. Please note however, that `obs_list` and `transform_obs` **cannot** be provided inside a sub-step. They must always be passed directly as arguments to `run_protocol_agmip`.

For example:

```
param_info <- list(
  p1 = list(lb = 0, ub = 1, default = 0.1),
  p2 = list(lb = 0, ub = 1, default = 0.5),
  p3 = list(lb = 5, ub = 15, default = 15)
)

steps <- list(
  group1 = list(
    obs_var = c("var1", "var2"),
    major_param = c("p1"),
    candidate_param = c("p2")
  ),
  group2 = list(
    obs_var = c("var3"),
    major_param = c("p3")
  )
)

res <- run_protocol_agmip(
  obs_list = obs_list,
  model_function = my_model_wrapper,
  model_options = model_options,
  param_info = param_info,
  step = steps
)
```

In this example, step6 of the AgMIP protocol will be run in two successive steps called "group1" and "group2". In the first step, variables "var1" and "var2" are used to estimate parameter "p1" (major parameter), and candidate parameter "p2" is considered in the automatic parameter selection procedure. The observations for variables "var1" and "var2", as well as the information about parameters "p1" and "p2", are automatically extracted from `obs_list` and `param_info`, which contain the information for all steps.

Observations used in step7:

Note that in step7, all observations included in `obs_list` are used, regardless of the `obs_var` variables defined for step6. Thus, observations for variables not used in step6 (e.g. because no parameter is directly associated with these variables) can still be used in step7, where all parameters selected during step6 are re-estimated using all observed variables (WLS step).

Value

Prints, graphs, and a list containing the results of the AgMIP Phase IV Calibration protocol. All results are saved in the folder specified by `out_dir`.

During execution, a console display indicates the description of the step currently being run.

The generated plots include:

- diagnostics recommended in Wallach et al. (2025): MSE, bias², rRMSE, and Efficiency for each variable at each step (files `barplot_rRMSE_EF_per_step.pdf` and `plot_MSE_Bias2_per_step.pdf`),
- scatter plots of simulations versus observations before and after each step (files `scatter_plots_*.pdf`),
- diagnostic plots for each minimization performed (see subfolders `AgMIP_protocol_step6`, `AgMIP_protocol_step7`, and their contents).

The returned object is a list containing:

- `final_values`: a named vector with the values of all parameters that were finally estimated. This includes all parameters selected during step6 and re-estimated in step7.
- `forced_param_values`: a named vector with the values of all parameters that were *not* estimated in the final calibration step 7. This includes in particular:
  - candidate parameters that were tested during step6 but not selected,
  - parameters defined through equality constraints or forced by the user (see input argument `forced_param_values`).
- `obs_var_list`: a character vector with the names of observed variables used in the protocol,
- `values_per_step`: a data.frame containing the default parameter values (from `param_info$default`, or `NA` if not provided) and the estimated values after step6 and step7,
- `stats_per_step`: a data.frame containing statistics (MSE, bias², rRMSE, and Efficiency) for each variable, before and after each step,
- `step6`: a list with detailed results for step6,
- `step7`: a list with detailed results for step7.

See Also

- `load_protocol_agmip` to extract `step`, `param_info` and `forced_param_values` from a structured Excel file,
- `get_agmip_protocol_template` and `get_agmip_protocol_example` to obtain template and example protocol files,
- The vignette *agmip_calibration_protocol* for a complete example workflow,
- Wallach et al. (2024, 2025) for detailed AgMIP protocol description and examples,
- `estim_param` for basic parameter estimation using CroptimizR.

`summary_frequentist` *Summarizes results of frequentist methods*

Description

Summarizes results of frequentist methods

Usage

```
summary_frequentist(
  optim_options,
  param_info,
  optim_results,
  out_dir,
  indent = 0
)
```

Arguments

`optim_options` List of options of the parameter estimation method, containing:

- `ranseed` Set random seed so that each execution of estim_param give the same results when using the same seed. If you want randomization, set it to NULL, otherwise set it to a number of your choice (e.g. 1234) (optional, default to NULL, which means random seed)
- specific options depending on the method used. Click on the links to see examples with the simplex and DreamZS methods.
- `out_dir` **[Deprecated]** Definition of `out_dir` in `optim_options` is no longer supported, use the new argument `out_dir` of `estim_param` instead.

`param_info` Information on the parameters to estimate. Either a list containing:

- `ub` and `lb`, named vectors of upper and lower bounds (-Inf and Inf can be used if init_values is provided),
- `default`, named vectors of default values (optional, corresponding parameters are set to these values when the parameter is part of the `candidate_param` list and when it is not estimated ; these values are also used as first initial values when the parameters are estimated)
- `init_values`, a data.frame containing initial values to test for the parameters (optional, if not provided, or if less values than number of repetitions of the minimization are provided, the, or part of the, initial values will be randomly generated using LHS sampling within parameter bounds).

or a named list containing for each parameter:

- `sit_list`, list the groups of situations for which the current estimated parameter must take different values (see here for an example),
- `ub` and `lb`, vectors of upper and lower bounds (one value per group),
- `init_values`, the list of initial values per group (data.frame, one column per group, optional).

- `default`, vector of default values per group (optional, the parameter is set to its default value when it is part of the `candidate_param` list and when it is not estimated ; the default value is also used as first initial value when the parameter is estimated)

`optim_results` Results list returned by frequentist method wrappers

`out_dir` Path to the directory where the optimization results will be written. (optional, default to `getwd()`)

`indent` Integer, level of indent of the printed messages as required by make_display_prefix

Value

Prints results of frequentist methods

`summary_multi_step` *Summarizes results of multi-step procedure*

Description

Summarizes results of multi-step procedure

Usage

```
summary_multi_step(results_multi_step, path_results, indent = 0)
```

Arguments

`results_multi_step` Results of the multi_step procedure as returned by post_treat_multi_step

`path_results` Folder path where results of the multi-step optimization procedure can be found

`indent` Integer, level of indent of the printed messages as required by make_display_prefix

Value

Prints results of the multi-step procedure

`test_wrapper` *Test model wrappers*

Description

This function perform some tests of CroptimizR model wrappers. See @details for more information.

Usage

```
test_wrapper(
  model_function,
  model_options,
  param_values,
  situation,
  var = NULL,
  sit_names = lifecycle::deprecated(),
  var_names = lifecycle::deprecated()
)
```

Arguments

`model_function`	Crop Model wrapper function to use.
`model_options`	List of options for the Crop Model wrapper (see help of the Crop Model wrapper function used).
`param_values`	a named vector that contains values and names for AT LEAST TWO model parameters THAT ARE EXPECTED TO PLAY ON ITS RESULTS.
`situation`	Vector of situations names for which results must be tested.
`var`	(optional) Vector of variables names for which results must be tested.
`sit_names`	**[Deprecated]** `sit_names` is no longer supported, use `situation` instead.
`var_names`	**[Deprecated]** `var_names` is no longer supported, use `var` instead.

Details

This function runs the wrapper consecutively with different subsets of param_values. It then checks:

- the format of the returned results
- the results are different when different subsets of param_values are used,
- the results are identical when same subsets of param_values are used.

Value

A list containing:

- test_results: a vector of boolean indicating which test succeeded (TRUE) or failed (FALSE)
- param_values_1: first subset of param_values
- param_values_2: second subset of param_values
- sim_1: results obtained with param_values_1
- sim_2: results obtained with param_values_2
- sim_3: results obtained for second run with param_values_1

Index