Package 'dycdtools'

October 18, 2021

Title Tools for DYRESM-CAEDYM Model Development: Calibration Assistant and Post-Processing

Version 0.3.1

Description Tools for DYRESM-CAEDYM model development, including assisting with calibrating selected model parameters and visualising model output through time series plot, profile plot, contour plot, and scatter plot.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr, ncdf4, tidyr, hydroGOF, ggplot2, RColorBrewer, lubridate, R.utils, parallel

RoxygenNote 7.1.1

Depends R (>= 2.10)

Suggests testthat

NeedsCompilation no

Author Songyan Yu [aut, cre] (<https://orcid.org/0000-0001-5765-7060>), Christopher McBride [ctb], Marieke Frassl [ctb]

Maintainer Songyan Yu <sunny.yu@griffith.edu.au>

Repository CRAN

Date/Publication 2021-10-18 04:10:02 UTC

R topics documented:

calib.assist 2
change_input_file 4
delete.space 4
ext.output 5
hgt.to.dpt 6
interpol 6
objective.fun 7

obs_temp 8
output_name 9
plot_cont 9
plot_cont_comp 11
plot_prof 12
plot_scatter 14
plot_ts 16
run.iteration 17

Index **19**

`calib.assist` *Assist calibration of DYRESM-CAEDYM model. Before using this function, make sure that you have set up "Bin" and "Files" sub-folders under the DYCD model folder.*

Description

This function tries different combinations of selected parameter values and outputs corresponding values of fit-of-goodness by calculating some objective functions. Then users can choose the optimal set of parameter values to calibrate the model.

Usage

```
calib.assist(
  cal.para = "Data/Calibration parameters.csv",
  combination = "random",
  n = 1,
  model.var = c("TEMP", "DO", "TN", "TP", "NO3", "PO4", "NH4", "SALINITY"),
  phyto.group = NA,
  obs.data = "Data/Obs.csv",
  objective.function = c("NSE", "RMSE"),
  start.date = "2017-06-06",
  end.date = "2020-02-29",
  dycd.wd = "work_directory",
  dycd.output = "work_directory/DYsim.nc",
  file_name = "work_directory/Calibration.csv",
  verbose = TRUE,
  parallel = FALSE,
  n.cores = NULL,
  write.out = FALSE
)
```

Arguments

`cal.para`	a character string naming a file where parameters to be calibrated and their value ranges. this file need to have fixed colnames.

`combination` a vector of string character of how to pick up combinations of parameter values."random" or "all".

`n` the number of randomly selections. Must be provided if combination = random.

`model.var` a vector of string character of modelled variables for calibration. When multiple phytoplankton groups will be combined for calibration, use "CHLA" and the following argument of "phyto.group" to specify them. When phytoplankton groups will be calibrated separately, put their abbrev. in this argument. Currently, five abbrevs are supported: CHLOR, FDIAT, NODUL, CYANO and CRYPT.

`phyto.group` a vector of simulated phytoplankton groups, including CHLOR, FDIAT, NODUL, CYANO and CRYPT.

`obs.data` a character string naming a file of observed lake data. This file need to have fixed column names and orders.

`objective.function`
a vector of string character claiming what objective function(s) to be used for calibration. Selected from the following five functions: "NSE": Nash-Sutcliffe efficiency coefficient, "RMSE": Root Mean Square Error, "MAE": Mean Absolute Error, "RAE": Relative Absolute Error, "Pearson": Pearson's r.

`start.date, end.date`
the beginning and ending simulation dates for the intended DYRESM-CAEDYM model run. The date format must be "%Y-%m-%d".

`dycd.wd` working directory where input files (including the bat file) to DYRESM-CAEDYM are stored.

`dycd.output` a character string naming the output file from the model run.

`file_name` a character string naming a png file for writing out the auto-calibration results.

`verbose` if TRUE, the auto-calibration information is printed.

`parallel` if TRUE, the calibration process can be run on multiple cores.

`n.cores` When parallel is TRUE, n.cores is the number of cores the calibration function will be run on. If not provided, the default value is the number of available cores on the computer -1.

`write.out` if TRUE, the auto-calibration results are saved a file with a file name set by the "file_name" argument.

Value

a dataframe of trialed values of parameters and corresponding values of objective function(s).

Note

No executable examples are provided to illustrate the use of this function, as this function relies on the DYRESM-CAEDYM executables to work.

change_input_file *change parameter value of input files to DYRESM_CAEDYM model.*

Description

change parameter value of input files to DYRESM_CAEDYM model.

Usage

```
change_input_file(input_file = "par", row_no = 12, new_value)
```

Arguments

input_file	vector of input format, such as "par","cfg".
row_no	the number of row where the variable of interest is in the input file
new_value	the new value that will be assigned to the variable of interest.

delete.space *Delete all whitespace until a non-whitespace character.*

Description

Delete all whitespace until a non-whitespace character.

Usage

```
delete.space(extract_val)
```

Arguments

extract_val	a vector.

`ext.output` *Extract simulations from DYRESM-CAEDYM output.*

Description

Extract simulations from DYRESM-CAEDYM output. It is recommended to use the following example code to assign values of each simulated variable to a corresponding matrix.

Usage

```
ext.output(
  dycd.output = "data/dysim.nc",
  var.extract = c("TEMP", "DO", "TP", "TN", "NO3", "NH4", "CHLA")
)
```

Arguments

`dycd.output`	a string of characters of the output netcdf file names of DYRESM-CAEDYM model
`var.extract`	a vector of variables to be extracted from the output file. Options include TEMP, DO, TP, NO3, NH4, TN, CHLA, FDIAT, SSOL1, SSOL2, and PO4. Apart from nominated variables, simulation period and layer height data are also extracted. For a full list of variables that can be extracted, use "data("output_name")".

Value

a list of values of those variables of interest, as well as two compulsory variables (i.e. simulation period, water level)

Examples

```
# extract simulated temperature values from DYRESM-CAEDYM simulation file
var.values<-ext.output(dycd.output=system.file("extdata", "dysim.nc", package = "dycdtools"),
                       var.extract=c("TEMP"))

for(i in 1:length(var.values)){
 expres<-paste0(names(var.values)[i],"<-data.frame(var.values[[",i,"]])")
 eval(parse(text=expres))
}
```

hgt.to.dpt *convert from height to depth*

Description

convert from height to depth

Usage

```
hgt.to.dpt(height)
```

Arguments

height	a vector of height profile

interpol *Interpolation of simulation for a series of user-defined depths.*

Description

Convert simulated variable at irregular layer heights to a dataframe of the same variable at sequenced layer heights.

Usage

```
interpol(layerHeights, var, min.depth, max.depth, by.value)
```

Arguments

layerHeights	layer heights, extracted from DYCD outputs
var	simulated variable values, extracted from DYCD outputs
min.depth, max.depth, by.value	minimum and maximum depth for interpolation at the depth increment of by.value.

Value

a matrix of interpolated values of such variable.

Examples

```
# extract simulated temperature values from DYRESM-CAEDYM simulation file
 var.values<-ext.output(dycd.output=system.file("extdata", "dysim.nc", package = "dycdtools"),
                      var.extract=c("TEMP"))

 for(i in 1:length(var.values)){
  expres<-paste0(names(var.values)[i],"<-data.frame(var.values[[",i,"]])")
  eval(parse(text=expres))
 }
 # interpolate temperature for depths from 0 to 13 m at increment of 0.5 m
 temp.interpolated<-interpol(layerHeights = dyresmLAYER_HTS_Var,
                             var = dyresmTEMPTURE_Var,
                             min.dept = 0,max.dept = 13,by.value = 0.5)
```

`objective.fun` *Measure the goodness of fit for DYCD model simulations.*

Description

Five objective functions can be used to measure goodness of fit: 1) Nash-Sutcliffe efficiency coefficient (NSE), 2) Root Mean Square Error (RMSE), 3) Mean Absolute Error (MAE), 4) Relative Absolute Error (RAE), and 5) Pearson's r.

Usage

```
objective.fun(
  sim,
  obs,
  fun = "RMSE",
  start.date = "2017-06-06",
  end.date = "2020-02-29",
  min.depth = 0,
  max.depth = 33,
  by.value = 0.5
)
```

Arguments

`sim`	a matrix of bio-geochemical variable values with column of time and row of depth.
`obs`	a data frame of observed value, with three columns: Date, depth, value.
`fun`	objective function to be calculated. Selected one from "NSE", "RMSE", "MAE","RAE", and "Pearson".
`start.date, end.date`	the beginning and ending simulation dates for the intended DYRESM-CAEDYM model run. The date format must be "%Y-%m-%d".

`min.depth, max.depth`
: minimum and maximum depth to be compared between simulations and observations.

`by.value`
: the value of increment for depth.

Value

a vector of objective function values. The first is NSE and the second is RMSE.

`obs_temp` *Example observed profiling temperature data across different depths over the period of 6-11 June 2017.*

Description

A table has three columns. The first column name is Date in the form of dd-mm-YY. The second column is Depth where the temperature data was monitored. The third column is monitored temperature value.

Usage

```
data(obs_temp)
```

Format

A data frame with 77 rows and 3 variables:

Date date when the monitoring happened

Depth depth of monitoring

TEMP temperature value

Source

self-made.

output_name *Default DYCD simulation variable names with their variable name*

Description

A table has two columns. The first column name is var.name, meaning variable names that are used in the extract.output function. The second column is the default DYCD simulation variable names, such as "dyresmLAYER_HTS_Var".

Usage

```
data(output_name)
```

Format

A data frame with 65 rows and 2 variables:

var.name variable name

output.name default DYCD simulation variable name

Source

self-made.

plot_cont *Contour plot of simulation results of a bio-geochemical variable.*

Description

Contour plot a matrix of values of a bio-geochemical variable, which can be generated through "interpol" function.

Usage

```
plot_cont(
  sim = temp.interpolated,
  sim.start = "2017-06-06",
  sim.end = "2017-06-15",
  legend.title = "T °C",
  min.depth = 0,
  max.depth = 13,
  by.value = 0.5,
  nlevels = 20,
  plot.save = TRUE,
  file_name = "Contour_temp.png",
  height = 5,
```

```
  width = 8,
  ppi = 150
)
```

Arguments

sim	a matrix of simulated variables that have been interpolated
sim.start, sim.end	the start and end of the simulation period for the DYRESM-CAEDYM model run of interest. The date format must be "%Y-%m-%d".
legend.title	the legend title of the contour figure.
min.depth, max.depth, by.value	minimum and maximum depth used to be the start of y axis of the contour plot, at the increment of by.value.
nlevels	a set of levels which are used to partition the range of simulation variable.
plot.save	if TRUE, the plot is saved with the "height","width", and "ppi" parameters.
file_name	the file path to save the generated contour figure.
height, width	the relative height/width of the figure.
ppi	the ppi value of the figure.

Value

a graph file of contour plot saved in the Figure folder.

Examples

```
 # extract simulated temperature values from DYRESM-CAEDYM simulation file
 var.values<-ext.output(dycd.output=system.file("extdata", "dysim.nc", package = "dycdtools"),
                        var.extract=c("TEMP"))

 for(i in 1:length(var.values)){
  expres<-paste0(names(var.values)[i],"<-data.frame(var.values[[",i,"]])")
  eval(parse(text=expres))
 }

# interpolate temperature for depths from 0 to 13 m at increment of 0.5 m
  temp.interpolated<-interpol(layerHeights = dyresmLAYER_HTS_Var,
                              var = dyresmTEMPTURE_Var,
                              min.dept = 0,max.dept = 13,by.value = 0.5)

# contour plot of temperature simulations
  plot_cont(sim=temp.interpolated,
            sim.start="2017-06-06",
            sim.end="2017-06-15",
            legend.title="T \u00B0C",
            min.depth=0,max.depth=13,by.value=0.5,
            nlevels=20,
            plot.save=FALSE,
            file_name="Contour_temp.png",
```

```
height=5,width=8,ppi=150)
```

`plot_cont_comp` *Contour plot of a variable simulation, with observed data shown as dots in the generated contour plot.*

Description

Contour plot a matrix of a bio-geochemical variable values, which can be generated through "interpol" function.

Usage

```
plot_cont_comp(
  sim = temp.interpolated,
  obs = obs_temp,
  file_name = "Contour_temp.png",
  sim.start = "2002-01-23",
  sim.end = "2016-12-31",
  plot.start = "2017-06-06",
  plot.end = "2017-06-15",
  legend.title = "T °C",
  min.depth = 0,
  max.depth = 13,
  by.value = 0.5,
  nlevels = 20,
  plot.save = TRUE,
  height = 5,
  width = 8,
  ppi = 150
)
```

Arguments

`sim`	a matrix of simulated variables that have been interpolated
`obs`	observed values of variable.
`file_name`	the file path to save the generated contour figure.
`sim.start, sim.end`	the start and end of the simulation period for the DYRESM-CAEDYM model run of interest. The date format must be "%Y-%m-%d".
`plot.start, plot.end`	the start and end of the plot period, in the format of "%Y-%m-%d"
`legend.title`	the legend title of the contour figure.
`min.depth, max.depth, by.value`	minimum and maximum depth used to be the start of y axis of the contour plot, at the increment of by.value.

nlevels	a set of levels which are used to partition the range of simulation variable.
plot.save	if TRUE, the plot is saved with the "height","width", and "ppi" parameters.
height, width	the relative height/width of the figure.
ppi	the ppi value of the figure.

Value

a graph file of contour plot saved in the Figure folder.

Examples

```
# extract simulated temperature values from DYRESM-CAEDYM simulation file
 var.values<-ext.output(dycd.output=system.file("extdata", "dysim.nc", package = "dycdtools"),
                        var.extract=c("TEMP"))

  for(i in 1:length(var.values)){
     expres<-paste0(names(var.values)[i],"<-data.frame(var.values[[",i,"]])")
     eval(parse(text=expres))
  }

# interpolate temperature for depths from 0 to 13 m at increment of 0.5 m
  temp.interpolated<-interpol(layerHeights = dyresmLAYER_HTS_Var,
                              var = dyresmTEMPTURE_Var,
                              min.dept = 0,max.dept = 13,by.value = 0.5)

  data(obs_temp)
# contour plot of temperature simulations with observed data shown as colour-coded dots
  plot_cont_comp(sim=temp.interpolated,
                 obs=obs_temp,
                 sim.start = "2017-06-06",
                 sim.end = "2017-06-15",
                 plot.start="2017-06-06",
                 plot.end="2017-06-15",
                 legend.title="T \u00B0C",
                 min.depth=0,max.depth=13,by.value=0.5,
                 nlevels=20,
                 plot.save=FALSE,
                 file_name="Contour_temp.png",
                 height=5,width=8,ppi=150)
```

---

plot_prof *Profile plot of simulated variable values vs. depth*

Description

A post-processing function used to visualise model output in a profile graph.

Usage

```
plot_prof(
  sim = temp.interpolated,
  obs = obs.temp,
  sim.start = "2017-06-06",
  sim.end = "2020-02-29",
  plot.start = "2017-06-06",
  plot.end = "2020-02-29",
  xlabel = "Temperature °C",
  min.depth = 0,
  max.depth,
  by.value,
  plot.save = FALSE,
  file_name,
  height = 11,
  width = 18
)
```

Arguments

sim	interpolated values of variable.
obs	observed values of variable.
sim.start, sim.end	the beginning and ending simulation dates for the intended DYRESM-CAEDYM model run. The date format must be "%Y-%m-%d".
plot.start, plot.end	the beginning and ending dates for the plotting purpose. The date format must be "%Y-%m-%d".
xlabel	the x axis label of the profile figure
min.depth, max.depth, by.value	minimum and maximum depth for the profile plot at the depth increment of by.value.
plot.save	if TRUE, the plot is saved with the "height","width", and "ppi" parameters.
file_name	the file path to save the generated profile figure
height, width	the height and width of the profile figure.

Value

a profile plot of sim~depth

Examples

```
var.values<-ext.output(dycd.output=system.file("extdata", "dysim.nc", package = "dycdtools"),
                       var.extract=c("TEMP"))

for(i in 1:length(var.values)){
  expres<-paste0(names(var.values)[i],"<-data.frame(var.values[[",i,"]])")
```

```
   eval(parse(text=expres))
  }

# interpolate temperature for depths from 0 to 13 m at increment of 0.5 m
  temp.interpolated<-interpol(layerHeights = dyresmLAYER_HTS_Var,
                              var = dyresmTEMPTURE_Var,
                              min.dept = 0,max.dept = 13,by.value = 0.5)

 data(obs_temp)
# profile plot of temperature sim and obs
  plot_prof(sim=temp.interpolated,
            obs = obs_temp,
            sim.start="2017-06-06",
            sim.end="2017-06-15",
            plot.start="2017-06-06",
            plot.end="2017-06-15",
            xlabel = "Temperature \u00B0C",
            min.depth = 0,max.depth = 13,by.value = 0.5)
```

`plot_scatter` *Scatter plot of sim and obs var values*

Description

Scatter plot of sim and obs var values

Usage

```
plot_scatter(
  sim = temp.interpolated,
  obs = obs.temp,
  sim.start = "2017-06-06",
  sim.end = "2020-02-29",
  plot.start = "2017-06-06",
  plot.end = "2020-02-29",
  min.depth = 0,
  max.depth,
  by.value,
  plot.save = TRUE,
  file_name,
  height = 4,
  width = 7
)
```

Arguments

`sim`	interpolated values of variable.

obs	observed values of variable. This data need to have fixed types of colnames and orders.
sim.start, sim.end	the beginning and ending simulation dates for the intended DYRESM-CAEDYM model run. The date format must be "%Y-%m-%d".
plot.start, plot.end	the beginning and ending dates for the plotting purpose. The date format must be "%Y-%m-%d".
min.depth, max.depth, by.value	minimum and maximum depth for the profile plot at the depth increment of by.value.
plot.save	if TRUE, the plot is saved with the "height","width", and "ppi" parameters.
file_name	the file path to save the generated scatter plot.
height, width	the height and width of the scatter figure.

Value

a scatter plot of sim vs. obs

Examples

```
 var.values<-ext.output(dycd.output=system.file("extdata", "dysim.nc", package = "dycdtools"),
                        var.extract=c("TEMP"))

 for(i in 1:length(var.values)){
   expres<-paste0(names(var.values)[i],"<-data.frame(var.values[[",i,"]])")
   eval(parse(text=expres))
  }

# interpolate temperature for depths from 0 to 13 m at increment of 0.5 m
  temp.interpolated<-interpol(layerHeights = dyresmLAYER_HTS_Var,
                              var = dyresmTEMPTURE_Var,
                              min.dept = 0,max.dept = 13,by.value = 0.5)

 data(obs_temp)

# scatter plot of sim and obs temperature
 plot_scatter(sim=temp.interpolated,
              obs=obs_temp,
              sim.start="2017-06-06",
              sim.end="2017-06-15",
              plot.start="2017-06-06",
              plot.end="2017-06-15",
              plot.save=FALSE,
              min.depth = 0,max.depth = 13,by.value = 0.5)
```

plot_ts *Time series plot of simulated and observed values*

Description

Time series plot of simulated and observed values

Usage

```
plot_ts(
  sim = temp.interpolated,
  obs = obs.temp,
  target.depth = c(1, 6, 12, 30),
  sim.start = "2017-06-06",
  sim.end = "2020-02-29",
  plot.start = "2017-06-06",
  plot.end = "2020-02-29",
  min.depth = 0,
  max.depth = 33,
  by.value = 0.5,
  ylabel = "Temperature °C",
  plot.save = FALSE,
  file_name,
  height = 7,
  width = 11
)
```

Arguments

sim	interpolated values of variable
obs	observed values of variable
target.depth	a vector of depth (unit:m) to be used to extract and plot variable values.
sim.start, sim.end	the beginning and ending simulation dates for the intended DYRESM-CAEDYM model run. The date format must be "%Y-%m-%d".
plot.start, plot.end	the beginning and ending dates for the plotting purpose. The date format must be "%Y-%m-%d".
min.depth, max.depth, by.value	minimum and maximum depth for the profile plot at the depth increment of by.value.
ylabel	the y axis title.
plot.save	if TRUE, the plot is saved with the "height","width", and "ppi" parameters.
file_name	the file path to save the generated ts plot.
height, width	the height and width of the time series figure.

Value

a plot of sim and obs time series.

Examples

```
 var.values<-ext.output(dycd.output=system.file("extdata", "dysim.nc", package = "dycdtools"),
                        var.extract=c("TEMP"))

 for(i in 1:length(var.values)){
   expres<-paste0(names(var.values)[i],"<-data.frame(var.values[[",i,"]])")
   eval(parse(text=expres))
  }

# interpolate temperature for depths from 0 to 13 m at increment of 0.5 m
  temp.interpolated<-interpol(layerHeights = dyresmLAYER_HTS_Var,
                              var = dyresmTEMPTURE_Var,
                              min.dept = 0,max.dept = 13,by.value = 0.5)

 data(obs_temp)
# time series plot of temperature sim and obs
 plot_ts(sim = temp.interpolated,
         obs = obs_temp,
         target.depth=c(1,6),
         sim.start="2017-06-06",
         sim.end="2017-06-15",
         plot.start="2017-06-06",
         plot.end="2017-06-15",
         ylabel="Temperature \u00B0C",
         min.depth=0,
         max.depth=13,
         by.value=0.5)
```

---

run.iteration *Internal function to provide parallel processing support to the calibration assistant function.*

Description

Internal function to provide parallel processing support to the calibration assistant function.

Usage

```
run.iteration(this.sim, dycd.wd)
```

Arguments

`this.sim`	a numeric denoting which parameter combination to be tried.
`dycd.wd`	working directory where input files (including the bat file) to DYRESM-CAEDYM are stored.

Index

∗ **datasets**
obs_temp, 8
output_name, 9

calib.assist, 2
change_input_file, 4

delete.space, 4

ext.output, 5

hgt.to.dpt, 6

interpol, 6

objective.fun, 7
obs_temp, 8
output_name, 9

plot_cont, 9
plot_cont_comp, 11
plot_prof, 12
plot_scatter, 14
plot_ts, 16

run.iteration, 17