Package: PEcAnAssimSequential (via r-universe)

November 4, 2024

Type Package

Title PEcAn Functions Used for Ecological Forecasts and Reanalysis

Version 1.8.0.9000

Author Mike Dietze

Maintainer Mike Dietze <dietze@bu.edu>

Description The Predictive Ecosystem Carbon Analyzer (PEcAn) is a scientific workflow management tool that is designed to simplify the management of model parameterization, execution, and analysis. The goal of PECAn is to streamline the interaction between data and models, and to improve the efficacy of scientific investigation.

Imports coda, dplyr, furrr, future, ggplot2, lubridate (>= 1.6.0), magrittr, Matrix, mvtnorm, ncdf4, nimble, PEcAn.DB, PEcAn.logger, PEcAn.remote, PEcAn.settings, PEcAn.uncertainty, PEcAn.workflow, purrr, rlang, stringr

Suggests corrplot, exactextractr, ggrepel, emdbook, glue, ggpubr, gridExtra, magic (>= 1.5.0), methods, PEcAn.benchmark, PEcAn.data.land, PEcAn.data.remote, PEcAn.utils, PEcAn.visualization, plotrix, plyr (>= 1.8.4), randomForest, keras3 (>= 1.0.0), raster, readr, reshape2 (>= 1.4.2), rlist, sf, stats, terra, testthat, tictoc, tidyr, sp, utils, XML

License BSD_3_clause + file LICENSE

Copyright Authors

Encoding UTF-8

RoxygenNote 7.3.2

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

RemoteUrl https://github.com/PecanProject/pecan

RemoteRef HEAD

RemoteSha caad7b3f8386df43eaf44f9316459f66ffc69b0b

Contents

adj.ens . . . 3
aggregate . . . 4
alltocs . . . 5
alr . . . 6
Analysis.sda . . . 6
analysis_sda_block . . . 7
assess.params . . . 9
block.2.vector . . . 9
block_matrix . . . 10
build.block.xy . . . 11
build_X . . . 12
conj_wt_wishart_sampler . . . 13
Construct.H.multisite . . . 13
Construct.R . . . 14
construct_nimble_H . . . 14
Construc_H . . . 15
Contruct.Pf . . . 16
Create_Site_PFT_CSV . . . 17
dwtmnorm . . . 18
EnKF . . . 18
EnKF.MultiSite . . . 19
GEF . . . 20
GEF.MultiSite.Nimble . . . 21
get_ensemble_weights . . . 21
GrabFillMatrix . . . 22
hop_test . . . 22
interactive.plotting.sda . . . 23
inv.alr . . . 26
load_data_paleon_sda . . . 26
load_nimble . . . 27
Local.support . . . 27
matrix_network . . . 28
MCMC_block_function . . . 28
MCMC_function . . . 29
MCMC_Init . . . 29
metSplit . . . 30
Obs.data.prepare.MultiSite . . . 31
obs_timestep2timepoint . . . 31
outlier.detector.boxplot . . . 32
piecew.poly.local . . . 32
Prep_OBS_SDA . . . 33
Remote_Sync_launcher . . . 33
rescaling_stateVars . . . 34
rwtmnorm . . . 34
sample.parameters . . . 35
sampler_toggle . . . 35
sample_met . . . 36

sda.enkf . . . 36
sda.enkf.multisite . . . 37
sda.enkf.original . . . 39
SDA_control . . . 40
SDA_downscale . . . 41
SDA_downscale_hrly . . . 42
SDA_downscale_metrics . . . 43
SDA_downscale_preprocess . . . 43
sda_matchparam . . . 44
SDA_OBS_Assembler . . . 45
SDA_remote_launcher . . . 46
sda_weights_site . . . 47
simple.local . . . 47
tobit.model . . . 48
tobit2space.model . . . 48
tobit_model_censored . . . 49
update_q . . . 49

Index **51**

`adj.ens` *adj.ens*

Description

This functions gives weights to different ensemble members based on their likelihood during the analysis step. Then it adjusts the analysis mean estimates of state variables based on the estimated weights.

Usage

```
adj.ens(Pf, X, mu.f, mu.a, Pa)
```

Arguments

`Pf`	A cov matrix of forecast state variables.
`X`	Dataframe or matrix of forecast state variables for different ensembles.
`mu.f`	A vector with forecast mean estimates of state variables.
`mu.a`	A vector with analysis mean estimates of state variables.
`Pa`	The state estimate cov matrix of analysis.

Value

Returns a vector of adjusted analysis mean estimates of state variables.

Author(s)

Michael Dietze `<dietze@bu.edu>`, Ann Raiho and Hamze Dokoohaki

`aggregate` *Aggregation Function*

Description

Aggregation Function

Usage

```
aggregate(downscale_output, polygon_data, func = "mean")
```

Arguments

`downscale_output`	Raster file output from downscale_function.R. Read file in this way if stored locally: `downscale_output <- readRDS("xxx.rds")`
`polygon_data`	A spatial polygon object (e.g., an 'sf' object) that defines the spatial units for aggregation. This data should be in a coordinate reference system compatible with the raster data (e.g., "EPSG:4326").
`func`	A character string specifying the aggregation function to use (e.g., 'mean', 'sum').

Details

This function will aggregate previously downscaled carbon flux amount to a spatial unit of choice

Value

It returns the 'polygon_data' with added columns for mean and sum values of the aggregated raster data for each ensemble member.

Author(s)

Harunobu Ishii

Examples

```
## Not run:
# Download a shapefile of U.S. (polygon data)
url <- "https://www2.census.gov/geo/tiger/GENZ2020/shp/cb_2020_us_state_20m.zip"
download.file(url, destfile = "polygon/us_states.zip")

# Unzip the downloaded file and save locally
unzip("polygon/us_states.zip", exdir = "polygon/us_states")
us_states <- st_read("polygon/us_states/cb_2020_us_state_20m.shp")
saveRDS(us_states, "polygon/us_states.rds")

# Load the saved polygon data with Massachusetts as an example
```

```
us_states <- readRDS("polygon/us_states.rds")
state <- "MA"
polygon_data <- st_transform(us_states[us_states$STUSPS == state, ], crs = "EPSG:4326")

# Load the downscaled raster output
downscale_output <- readRDS("path/to/downscale_output.rds")

# Slot in as argument to the aggregate function
result <- aggregate(downscale_output, polygon_data)
print(result)

## End(Not run)
```

`alltocs` *alltocs*

Description

This function finds all the tic functions called before and estimates the time elapsed for each one saves/appends it to a csv file.

Usage

```
alltocs(fname = "tocs.csv")
```

Arguments

`fname`	string path to where the output needs to be saved as a csv file.

Value

This function writes down a csv file with three columns: 1- message sepecified in the ‘tic‘ 2- Total elapsed time and 3- the execution time

Examples

```
## Not run:
 library(tictoc)
 tic("Analysis")
 Sys.sleep(5)
 testfunc()
 tic("Adjustment")
 Sys.sleep(4)
 alltocs("timing.csv")

## End(Not run)
```

`alr` *Additive Log Ratio transform*

Description

Additive Log Ratio transform

Usage

```
alr(y)
```

Arguments

`y`	state var

`Analysis.sda` *Analysis.sda*

Description

This functions uses the FUN to perform the analysis. EnKF function is developed inside the PEcAnAssimSequential package which can be sent to this function to perform the Ensemble Kalman Filter. The other option is GEF function inside the same package allowing to perform Generalized Ensemble kalman Filter.

If you're using an arbitrary function you can use the ... to send any other variables to your desired analysis function.

Usage

```
Analysis.sda(
  settings,
  FUN,
  Forecast = list(Pf = NULL, mu.f = NULL, Q = NULL, X = NULL),
  Observed = list(R = NULL, Y = NULL),
  H,
  extraArg,
  ...
)
```

Arguments

`settings`	pecan standard settings list.
`FUN`	A Function for performing the analysis step. Two available options are: 1-EnKF and 2-GEF.

`Forecast`	A list containing the forecasts variables including Q (process variance) and X (a dataframe of forecasts state variables for different ensemble)
`Observed`	A list containing the observed variables including R (cov of observed state variables) and Y (vector of estimated mean of observed state variables)
`H`	is a matrix of 1's and 0's specifying which observations go with which variables.
`extraArg`	This argument is a list containing aqq, bqq and t. The aqq and bqq are shape parameters estimated over time for the proccess covariance and t gives the time in terms of index of obs.list. See Details.
`...`	Extra argument sent to the analysis function. In case you're using the 'GEF' function, this function requires nt, obs.mean, obs.cov, which are the total number of steps, list of observed means and list of observed cov respectively.

Value

Returns whatever the FUN is returning. In case of EnKF and GEF, this function returns a list with estimated mean and cov matrix of forecast state variables as well as mean and cov estimated as a result of assimilation/analysis .

Author(s)

Michael Dietze `<dietze@bu.edu>`, Ann Raiho and Hamze Dokoohaki

`analysis_sda_block` *analysis_sda_block*

Description

This function provides the block-based MCMC sampling approach.

Usage

```
analysis_sda_block(
  settings,
  block.list.all,
  X,
  obs.mean,
  obs.cov,
  t,
  nt,
  MCMC.args,
  block.list.all.pre = NULL
)
```

Arguments

`settings`	pecan standard multi-site settings list.
`block.list.all`	Lists of forecast and analysis outputs for each time point of each block. If t=1, we initialize those outputs of each block with NULL from the 'sda.enkf.multisite' function.
`X`	A matrix contains ensemble forecasts with the dimensions of '[ensemble number, site number * number of state variables]'. The columns are matched with the site.ids and state variable names of the inside the 'FORECAST' object in the 'sda.enkf.multisite' script.
`obs.mean`	Lists of date times named by time points, which contains lists of sites named by site ids, which contains observation means for each state variables of each site for each time point.
`obs.cov`	Lists of date times named by time points, which contains lists of sites named by site ids, which contains observation covariances for all state variables of each site for each time point.
`t`	time point in format of YYYY-MM-DD.
`nt`	total length of time steps, corresponding to the 'nt' variable in the 'sda.enkf.multisite' function.
`MCMC.args`	arguments for the MCMC sampling, details can be found in the roxygen strucutre for control list in the 'sda.enkf.multisite' function.
`block.list.all.pre`	pre-existed block.list.all object for passing the aqq and bqq to the current SDA run, the default is NULL. Details can be found in the roxygen structure for 'pre_enkf_params' of the 'sda.enkf.multisite' function

Details

This function will add data and constants into each block that are needed for the MCMC sampling.

Value

It returns the 'build.block.xy' object and the analysis results.

Author(s)

Dongchen Zhang

`assess.params` *assess.params*

Description

Assessing parameter estimations after mapping model output to tobit space

Usage

```
assessParams(dat, Xt, wts = NULL, mu_f_TRUE = NULL, P_f_TRUE = NULL)
```

Arguments

`dat`	MCMC output
`Xt`	ensemble output matrix
`wts`	ensemble weights
`mu_f_TRUE`	muf before tobit2space
`P_f_TRUE`	Pf before tobit2space

Value

make plots

Author(s)

Michael Dietze and Ann Raiho `<dietze@bu.edu>`

`block.2.vector` *block.2.vector*

Description

block.2.vector

Usage

```
block.2.vector(block.list, X, H)
```

Arguments

`block.list`	lists of blocks generated by the 'build.block.xy' function.
`X`	A matrix contains ensemble forecasts.
`H`	H index created by the 'construct_nimble_H' function.

Value

It returns a list of analysis results by MCMC sampling.

Author(s)

Dongchen Zhang

`block_matrix` *block_matrix*

Description

This function is adopted from migest package.

Usage

```
block_matrix(x = NULL, b = NULL, byrow = FALSE, dimnames = NULL)
```

Arguments

`x`	Vector of numbers to identify each block.
`b`	Numeric value for the size of the blocks within the matrix ordered depending on byrow
`byrow`	logical value. If FALSE (the default) the blocks are filled by columns, otherwise the blocks in the matrix are filled by rows.
`dimnames`	Character string of name attribute for the basis of the block matrix. If NULL a vector of the same length of b provides the basis of row and column names.#'.

Value

Returns a matrix with block sizes determined by the b argument. Each block is filled with the same value taken from x.

Author(s)

Guy J. Abel

`build.block.xy` *build.block.xy*

Description

This function split long vector and covariance matrix into blocks corresponding to the localization.

Usage

```
build.block.xy(settings, block.list.all, X, obs.mean, obs.cov, t)
```

Arguments

`settings`	pecan standard multi-site settings list.
`block.list.all`	List contains nt empty sub-elements.
`X`	A matrix contains ensemble forecasts.
`obs.mean`	List of dataframe of observation means, named with observation datetime.
`obs.cov`	List of covariance matrices of state variables , named with observation datetime.
`t`	time point.

Details

This function will add data and constants into each block that are needed for the MCMC sampling.

Value

It returns the 'build.block.xy' object with data and constants filled in.

Author(s)

Dongchen Zhang

build_X *build_X*

Description

builds X matrix for SDA

Usage

```
build_X(
  out.configs,
  settings,
  new.params,
  nens,
  read_restart_times,
  outdir,
  t = 1,
  var.names,
  my.read_restart,
  restart_flag = FALSE
)
```

Arguments

out.configs	object created for build_X passed from sda.enkf_MultiSite
settings	settings object, passed from sda.enkf_MultiSite
new.params	object created from sda_matchparam, passed from sda.enkf_MultiSite
nens	number of ensemble members i.e. runs
read_restart_times	passed from sda.enkf_MultiSite
outdir	location of previous run output folder containing .nc files
t	Default t=1, for function to work within time loop
var.names	list of state variables taken from settings object
my.read_restart	object that points to the model restart function i.e. read_restart.SIPNET
restart_flag	flag if it's a restart stage. Default is FALSE.

Value

X ready to be passed to SDA Analysis code

Author(s)

Alexis Helgeson

`conj_wt_wishart_sampler` *Weighted conjugate wishart*

Description

Weighted conjugate wishart

Usage

```
conj_wt_wishart_sampler(model, mvSaved, target, control)
```

Arguments

`model`	model
`mvSaved`	copied to
`target`	thing being targetted
`control`	unused

`Construct.H.multisite` *Construct.H.multisite*

Description

This function is makes the blocked mapping function.

Usage

```
Construct.H.multisite(site.ids, var.names, obs.t.mean)
```

Arguments

`site.ids`	a vector name of site ids
`var.names`	vector names of state variable names
`obs.t.mean`	list of vector of means for the time t for different sites.

Value

Returns a matrix with block sizes determined by the b argument. Each block is filled with the same value taken from x.

Author(s)

Hamze

Construct.R *Construct.R*

Description

Make sure that both lists are named with siteids.

Usage

```
Construct.R(site.ids, var.names, obs.t.mean, obs.t.cov)
```

Arguments

site.ids	a vector name of site ids
var.names	vector names of state variable names
obs.t.mean	list of vector of means for the time t for different sites.
obs.t.cov	list of list of cov for the time for different sites.

Value

This function returns a list with Y and R ready to be sent to the analysis functions.

Author(s)

Hamze Dokoohaki

construct_nimble_H *construct_nimble_H*

Description

This function is an upgrade to the Construct.H.multisite function which provides the index by different criteria.

Usage

```
construct_nimble_H(site.ids, var.names, obs.t, pft.path = NULL, by = "single")
```

Arguments

site.ids	a vector name of site ids
var.names	vector names of state variable names
obs.t	list of vector of means for the time t for different sites.
pft.path	physical path to the pft.csv file.
by	criteria, it supports by variable, site, pft, all, and single Q.

Value

Returns one vector containing index for which Q to be estimated for which variable, and the other vector gives which state variable has which observation (= element.W.Data).

Author(s)

Dongchen Zhang

`Construc_H` *Construc_H*

Description

This function creates a matrix mapping obsereved data to their forecast state variable.

Usage

```
Construct_H(choose, Y, X)
```

Arguments

`choose`	a vector of observations indices oredered based on their appearances in the list of state variable names.
`Y`	vector of observations
`X`	Dataframe or matrix of forecast state variables for different ensembles.

Value

This returns a matrix specifying which observation go with which state variables.

Author(s)

Hamze Dokoohaki

`Contruct.Pf` *Contruct.Pf*

Description

The argument X needs to have an attribute pointing the state variables to their corresponding site. This attribute needs to be called 'Site'. At the moment, the cov between state variables at blocks defining the cov between two sites are assumed zero.

Usage

```
Contruct.Pf(
  site.ids,
  var.names,
  X,
  localization.FUN = NULL,
  t = 1,
  blocked.dis = NULL,
  ...
)
```

Arguments

`site.ids`	a vector name of site ids.
`var.names`	vector names of state variable names.
`X`	a matrix of state variables. In this matrix rows represent ensembles, while columns show the variables for different sites.
`localization.FUN`	This is the function that performs the localization of the Pf matrix and it returns a localized matrix with the same dimensions.
`t`	not used
`blocked.dis`	passed to 'localization.FUN'
`...`	passed to 'localization.FUN'

Value

It returns the var-cov matrix of state variables at multiple sites.

Author(s)

Hamze Dokoohaki

`Create_Site_PFT_CSV` *Title Identify pft for each site of a multi-site settings using NLCD and Eco-region*

Description

Title Identify pft for each site of a multi-site settings using NLCD and Eco-region

Usage

```
Create_Site_PFT_CSV(settings, Ecoregion, NLCD, con)
```

Arguments

`settings`	a multi-site settings
`Ecoregion`	path of Ecoregion data (*.shp)
`NLCD`	path of NLCD img file
`con`	connection to bety

Value

pft info with each site

Examples

```
## Not run:
 NLCD <- file.path(
   "/fs", "data1", "pecan.data", "input",
   "nlcd_2001_landcover_2011_edition_2014_10_10",
   "nlcd_2001_landcover_2011_edition_2014_10_10.img")
 Ecoregion <- file.path(
   "/projectnb", "dietzelab", "dongchen",
   "All_NEON_SDA", "NEON42", "eco-region", "us_eco_l3_state_boundaries.shp")
 settings <- PEcAn.settings::read.settings(
   "/projectnb/dietzelab/dongchen/All_NEON_SDA/NEON42/pecan.xml")
 con <- PEcAn.DB::db.open(settings$database$bety)
   site_pft_info <- Create_Site_PFT_CSV(settings, Ecoregion, NLCD, con)

## End(Not run)
```

`dwtmnorm` *weighted multivariate normal density*

Description

weighted multivariate normal density

Usage

```
dwtmnorm(x, mean, prec, wt, log = 0)
```

Arguments

`x`	random variable
`mean`	mean
`prec`	precision
`wt`	weight
`log`	log

`EnKF` *EnKF*

Description

Given the Forecast and Observed this function performs the Ensemble Kalamn Filter.

Usage

```
EnKF(settings, Forecast, Observed, H, extraArg = NULL, ...)
```

Arguments

`settings`	pecan standard settings list.
`Forecast`	A list containing the forecasts variables including Q (process variance) and X (a dataframe of forecasts state variables for different ensemble)
`Observed`	A list containing the observed variables including R (cov of observed state variables) and Y (vector of estimated mean of observed state variables)
`H`	is a matrix of 1's and 0's specifying which observations go with which variables.
`extraArg`	This argument is NOT used inside this function but it is a list containing aqq, bqq and t. The aqq and bqq are shape parameters estimated over time for the proccess covariance and t gives the time in terms of index of obs.list. See Details.
`...`	Extra argument sent to the analysis function.

Value

It returns a list with estimated mean and cov matrix of forecast state variables as well as mean and cov estimated as a result of assimilation/analysis .

Author(s)

Michael Dietze <dietze@bu.edu>, Ann Raiho and Hamze Dokoohaki

---

`EnKF.MultiSite` *EnKF.MultiSite*

Description

Given the Forecast and Observed this function performs the Ensemble Kalamn Filter.

Usage

```
EnKF.MultiSite(settings, Forecast, Observed, H, extraArg = NULL, ...)
```

Arguments

`settings`	pecan standard settings list.
`Forecast`	A list containing the forecasts variables including Q (process variance) and X (a dataframe of forecasts state variables for different ensemble)
`Observed`	A list containing the observed variables including R (cov of observed state variables) and Y (vector of estimated mean of observed state variables)
`H`	is a matrix of 1's and 0's specifying which observations go with which state variables.
`extraArg`	This argument is NOT used inside this function but it is a list containing aqq, bqq and t. The aqq and bqq are shape parameters estimated over time for the proccess covariance and t gives the time in terms of index of obs.list.
`...`	Extra argument sent to the analysis function.

Details

This function is different than 'EnKF' function in terms of how it creates the Pf matrix.

Value

It returns a list with estimated mean and cov matrix of forecast state variables as well as mean and cov estimated as a result of assimilation/analysis .

Author(s)

Michael Dietze <dietze@bu.edu>, Ann Raiho and Hamze Dokoohaki

GEF *GEF*

Description

Given the Forecast and Observed this function performs the Generalized Ensemble Kalamn Filter. The generalized ensemble filter follows generally the three steps of sequential state data assimilation. But, in the generalized ensemble filter we add a latent state vector that accounts for added process variance. Furthermore, instead of solving the analysis analytically like the EnKF, we have to estimate the mean analysis vector and covariance matrix with MCMC.

Usage

```
GEF(
  settings,
  Forecast,
  Observed,
  H,
  extraArg,
  nitr = 50000,
  nburnin = 10000,
  ...
)

GEF.MultiSite(settings, Forecast, Observed, H, extraArg, ...)
```

Arguments

settings	pecan standard settings list.
Forecast	A list containing the forecasts variables including Q (process variance) and X (a dataframe of forecast state variables for different ensemble)
Observed	A list containing the observed variables including R (cov of observed state variables) and Y (vector of estimated mean of observed state variables)
H	not used
extraArg	This argument is a list containing aqq, bqq and t. The aqq and bqq are shape parameters estimated over time for the process covariance and t gives the time in terms of index of obs.list. See Details.
nitr	Number of iterations to run each MCMC chain.
nburnin	Number of initial, pre-thinning, MCMC iterations to discard.
...	This function requires nt, obs.mean, obs.cov, which are the total number of steps, list of observed means and list of observed cov respectively.

Value

It returns a list with estimated mean and cov matrix of forecast state variables as well as mean and cov estimated as a result of assimilation/analysis .

Author(s)

Michael Dietze <dietze@bu.edu>, Ann Raiho and Hamze Dokoohaki

---

GEF.MultiSite.Nimble *multisite TWEnF*

Description

multisite TWEnF

Usage

```
GEF.MultiSite.Nimble
```

Format

TBD

---

get_ensemble_weights *get_ensemble_weights*

Description

Creates file of ensemble weights in format needed for SDA

Usage

```
get_ensemble_weights(settings, time_do)
```

Arguments

settings	PEcAn settings object
time_do	Give user specific time so you don't have to have it be annual

Value

NONE

Author(s)

Ann Raiho <ann.raiho@gmail.com>

GrabFillMatrix *GrabFillMatrix*

Description

GrabFillMatrix

Usage

```
GrabFillMatrix(M, ind, M1 = NULL)
```

Arguments

M	source matrix that will be either subtracted or filled in.
ind	vector of index that of where to be subtracted or filled in.
M1	additional matrix used to fill in the source matrix, the default it NULL.

Details

This function helps subtract or fill in a matrix given the index.

Author(s)

Dongchen Zhang

hop_test *hop_test*

Description

Hop test. This script tests that the model successfully reads it's own restart and can restart without loss of information.

Usage

```
hop_test(settings, ens.runid = NULL, nyear)
```

Arguments

settings	SDA PEcAn settings object
ens.runid	run id. If not provided, is looked up from [settings$outdir]/runs.txt
nyear	number of years to run hop test over

Value

NONE

Author(s)

Ann Raiho <araiho@nd.edu>

`interactive.plotting.sda` *Internal functions for plotting SDA outputs. Interactive, post analysis time-series and bias plots in base plotting system and ggplot*

Description

Internal functions for plotting SDA outputs. Interactive, post analysis time-series and bias plots in base plotting system and ggplot

Usage

```
interactive.plotting.sda(
  settings,
  t,
  obs.times,
  obs.mean,
  obs.cov,
  obs,
  X,
  FORECAST,
  ANALYSIS
)

postana.timeser.plotting.sda(
  settings,
  t,
  obs.times,
  obs.mean,
  obs.cov,
  obs,
  X,
  FORECAST,
  ANALYSIS
)

postana.bias.plotting.sda(
  settings,
  t,
  obs.times,
  obs.mean,
  obs.cov,
  obs,
```

```
  X,
  FORECAST,
  ANALYSIS
)

postana.bias.plotting.sda.corr(t, obs.times, X, aqq, bqq)

post.analysis.ggplot(
  settings,
  t,
  obs.times,
  obs.mean,
  obs.cov,
  obs,
  X,
  FORECAST,
  ANALYSIS,
  plot.title = NULL
)

post.analysis.ggplot.violin(
  settings,
  t,
  obs.times,
  obs.mean,
  obs.cov,
  obs,
  X,
  FORECAST,
  ANALYSIS,
  plot.title = NULL
)

post.analysis.multisite.ggplot(
  settings,
  t,
  obs.times,
  obs.mean,
  obs.cov,
  FORECAST,
  ANALYSIS,
  plot.title = NULL,
  facetg = FALSE,
  readsFF = NULL,
  Add_Map = FALSE
)

SDA_timeseries_plot(
```

```
  ANALYSIS,
  FORECAST,
  obs.mean = NULL,
  obs.cov = NULL,
  outdir,
  pft.path = NULL,
  by = "site",
  types = c("FORECAST", "ANALYSIS", "OBS"),
  CI = c(0.025, 0.975),
  unit = list(AbvGrndWood = "Mg/ha", LAI = "m2/m2", SoilMoistFrac = "", TotSoilCarb =
    "kg/m2"),
  style = list(general_color = c(FORECAST = "blue", ANALYSIS = "red", OBS = "black"),
    fill_color = c(FORECAST = "yellow", ANALYSIS = "green", OBS = "grey"), title_color =
    "red"),
  PDF_w = 20,
  PDF_h = 16,
  t.inds = NULL
)
```

Arguments

settings	pecan standard settings list.
t	current time - int number giving the position of the current time in obs.time.
obs.times	vector of dates of measurements
obs.mean	obs.mean
obs.cov	obs.cov
obs	list containing the mean and cov object
X	dataframe of state variables for each ensemble
FORECAST	Forecast object from the sda.output.Rdata.
ANALYSIS	Analysis object from the sda.output.Rdata.
aqq, bqq	shape parameters estimated over time for the process covariance
plot.title	character giving the title for post visualization ggplots
facetg	logical: Create a subpanel for each variable?
readsFF	optional forward forecast
Add_Map	Bool variable decide if we want to export the GIS map of Ecoregion.
outdir	physical path where the pdf will be stored.
pft.path	Physical path of pft.csv file to allow by = pft option.
by	arrange figures by var, pft, or site.
types	data types that shown in the figure.
CI	range of confidence interval.
unit	list of unit used for y axis label.
style	color option.

PDF_w	width of exported PDF file, passed on to 'base::pdf()'.
PDF_h	height of exported PDF file, passed on to 'base::pdf()'.
t.inds	index of period that will be plotted.

Author(s)

Dongchen Zhang

---

inv.alr *inverse of ALR transform*

Description

inverse of ALR transform

Usage

```
inv.alr(alr)
```

Arguments

alr	state var

---

load_data_paleon_sda *load_data_paleon_sda*

Description

Load data function for paleon SDA data products

Usage

```
load_data_paleon_sda(settings)
```

Arguments

settings	PEcAn SDA settings object

Value

obs.mean and obs.cov for sda.enkf

Author(s)

Ann Raiho <araiho@nd.edu>

`load_nimble` *load_nimble*

Description

This functions is internally used to register a series of nimble functions inside GEF analysis function.

Usage

```
y_star_create(X)
```

Arguments

`X`	state var

Author(s)

Ann Raiho, Hamze Dokoohaki

`Local.support` *Local.support*

Description

distance.mat matrix doesn't need to be just the physical distance, however it represent a measure of similarity between state variables in different sites.

Usage

```
Local.support(Pf, distance.mat, scalef = 1)
```

Arguments

`Pf`	Forecast error covariance matrix
`distance.mat`	is matrix of distances between sites.
`scalef`	scalef is a numeric value that requires tuning and it controls the shape of the corrolation function

Value

It returns a localized covariance matrix by taking a Schur product between Pf and a corrolation function

Author(s)

Hamze Dokoohaki

`matrix_network` *matrix_network*

Description

matrix_network

Usage

```
matrix_network(mat)
```

Arguments

`mat`	a boolean matrix representing the interactions between any sites.

Value

It returns lists of index representing each network.

Author(s)

Dongchen Zhang

`MCMC_block_function` *MCMC_block_function*

Description

MCMC_block_function

Usage

```
MCMC_block_function(block)
```

Arguments

`block`	each block within the 'block.list' lists.

Value

It returns the 'block' object with analysis results filled in.

Author(s)

Dongchen Zhang

`MCMC_function` *MCMC_function*

Description

MCMC_function

Usage

```
MCMC_function(data)
```

Arguments

`data`	list containing everything needed for the MCMC sampling.

Details

This function replace the previous code where implenmented the MCMC sampling part, which allows the MCMC sampling of multiple chains under parallel mode.

Author(s)

Michael Dietze `<dietze@bu.edu>`, Ann Raiho, Hamze Dokoohaki, and Dongchen Zhang.

`MCMC_Init` *MCMC_Init*

Description

MCMC_Init

Usage

```
MCMC_Init(block.list, X)
```

Arguments

`block.list`	lists of blocks generated by the 'build.block.xy' function.
`X`	A matrix contains ensemble forecasts.

Details

This function helps create initial conditions for the MCMC sampling.

Value

It returns the 'block.list' object with initial conditions filled in.

Author(s)

Dongchen Zhang

`metSplit` *metSplit*

Description

metSplit

Usage

```
metSplit(
  conf.settings,
  inputs,
  settings,
  model,
  no_split = FALSE,
  obs.times,
  t,
  nens,
  restart_flag = FALSE,
  my.split_inputs
)
```

Arguments

`conf.settings`	listed multisite settings object generated by sda.enkf_MultiSite
`inputs`	listed object containing met ensemble members generated by input.ens.gen
`settings`	settings object passed to sda.enkf_MultiSite
`model`	model name ex. SIPNET
`no_split`	TRUE/FALSE if model requires met split
`obs.times`	matrix of dates used for assimilation
`t`	number of dates in obs.times
`nens`	number of ensemble members, taken from settings object
`restart_flag`	TRUE/FALSE taken from restart arguement
`my.split_inputs`	generated by sda.enkf_MultiSite ex. split_inputs.SIPNET

Value

input.split object with split met filepaths

Author(s)

Alexis Helgeson

`Obs.data.prepare.MultiSite` *Obs.data.prepare.MultiSite*

Description

Obs.data.prepare.MultiSite

Usage

```
Obs.data.prepare.MultiSite(obs.path, site.ids)
```

Arguments

`obs.path`	Path to the obs data which is expected to be an .Rdata.
`site.ids`	a character vector of site ids which need to be extracted.

Value

a list of observed mean and cov as the SDA expected it to be.

`obs_timestep2timepoint` *convert from timestep to actual time points. supports year, month, week, and day as time unit.*

Description

convert from timestep to actual time points. supports year, month, week, and day as time unit.

Usage

```
obs_timestep2timepoint(start.date, end.date, timestep)
```

Arguments

`start.date`	start date when the first observation was taken.
`end.date`	end date when the last observation was taken.
`timestep`	a list includes time unit and number of time unit per timestep.

Value

timepoints from start to end date given the number of time unit per timestep.

Author(s)

Dongchen Zhang

`outlier.detector.boxplot` *outlier.detector.boxplot*

Description

This function performs a simple outlier replacement on all the columns of dataframes inside a list

Usage

```
outlier.detector.boxplot(X)
```

Arguments

`X`	A list of dataframes

Value

A list the same dimension as X, with each column of each dataframe modified by replacing outlier points with the column median

---

`piecew.poly.local` *piecew.poly.local*

Description

5th order piecewise polynomial adopted from Data assimilation for spatio-temporal processes - p250 - Sebastian Reich

Usage

```
piecew.poly.local(Pf, distance.mat, scalef = 2)
```

Arguments

`Pf`	Forecast error covariance matrix
`distance.mat`	is matrix of distances between sites.
`scalef`	scalef is a numeric value that requires tuning and it controls the shape of the corrolation function

Value

It returns a localized covariance matrix by taking a Schur product between Pf and a corrolation function

Author(s)

Hamze Dokoohaki

`Prep_OBS_SDA` *SDA observation preparation function for LAI and AGB*

Description

SDA observation preparation function for LAI and AGB

Usage

```
Prep_OBS_SDA(settings, out_dir, AGB_dir, Search_Window = 30)
```

Arguments

`settings`	multi.settings objects that contains multiple sites info
`out_dir`	output dir
`AGB_dir`	AGB data dir
`Search_Window`	search window for locate available LAI values

Value

mean and covariance of observations

`Remote_Sync_launcher` *Remote_Sync_launcher*

Description

Remote_Sync_launcher

Usage

```
Remote_Sync_launcher(settingPath, remote.path, PID)
```

Arguments

`settingPath`	Path to your local setting .
`remote.path`	Path generated by SDA_remote_launcher which shows the path to your remote SDA run.
`PID`	PID generated by SDA_remote_launcher which shows the active PID running your SDA job.

rescaling_stateVars *rescaling_stateVars*

Description

This function uses a set of scaling factors defined in the pecan XML to scale a given matrix

Usage

```
rescaling_stateVars(settings, X, multiply = TRUE)
```

Arguments

settings	pecan xml settings where state variables have the scaling_factor tag
X	Any Matrix with column names as variable names
multiply	TRUE = multiplication, FALSE = division

Value

rescaled Matrix

rwtmnorm *random weighted multivariate normal*

Description

random weighted multivariate normal

Usage

```
rwtmnorm(n, mean, prec, wt)
```

Arguments

n	sample size
mean	mean
prec	precision
wt	weight

`sample.parameters` *sample parameters*

Description

sample parameters

Usage

```
sample.parameters(ne, settings, con)
```

Arguments

`ne`	number of ensemble members
`settings`	PEcAn settings object
`con`	PEcAn database connection

Value

data frame of sampled parameters from the posterior distribution

Author(s)

Michael Dietze `<dietze@bu.edu>`

`sampler_toggle` *sampler toggling*

Description

sampler toggling

Usage

```
sampler_toggle(model, mvSaved, target, control)
```

Arguments

`model`	model
`mvSaved`	copied to
`target`	thing being targetted
`control`	unused

`sample_met` *Sample meteorological ensembles*

Description

Sample meteorological ensembles

Usage

```
sample_met(settings, nens = 1)
```

Arguments

`settings`	PEcAn settings list
`nens`	number of ensemble members to be sampled

`sda.enkf` *State Variable Data Assimilation: Ensemble Kalman Filter and Generalized ensemble filter*

Description

Restart mode: Basic idea is that during a restart (primary case envisioned as an iterative forecast), a new workflow folder is created and the previous forecast for the start_time is copied over. During restart the initial run before the loop is skipped, with the info being populated from the previous run. The function then dives right into the first Analysis, then continues on like normal.

Usage

```
sda.enkf(
  settings,
  obs.mean,
  obs.cov,
  Q = NULL,
  restart = NULL,
  control = list(trace = TRUE, interactivePlot = TRUE, TimeseriesPlot = TRUE, BiasPlot =
    FALSE, plot.title = NULL, debug = FALSE, pause = FALSE),
  ...
)
```

Arguments

settings	PEcAn settings object
obs.mean	List of dataframe of observation means, named with observation datetime.
obs.cov	List of covariance matrices of state variables , named with observation datetime.
Q	Process covariance matrix given if there is no data to estimate it.
restart	Used for iterative updating previous forecasts. When the restart is TRUE it read the object in SDA folder written from previous SDA.
control	List of flags controlling the behaviour of the SDA. trace for reporting back the SDA outcomes, interactivePlot for plotting the outcomes after each step, TimeseriesPlot for post analysis examination, BiasPlot for plotting the correlation between state variables, plot.title is the title of post analysis plots and debug mode allows for pausing the code and examining the variables inside the function.
...	Additional arguments, currently ignored

Value

NONE

Author(s)

Michael Dietze and Ann Raiho <dietze@bu.edu>

---

sda.enkf.multisite *State Variable Data Assimilation: Ensemble Kalman Filter and Generalized ensemble filter*

Description

Check out SDA_control function for more details on the control arguments.

Usage

```
sda.enkf.multisite(
  settings,
  obs.mean,
  obs.cov,
  Q = NULL,
  restart = NULL,
  pre_enkf_params = NULL,
  ensemble.samples = NULL,
  control = list(trace = TRUE, TimeseriesPlot = FALSE, debug = FALSE, pause = FALSE,
    Profiling = FALSE, OutlierDetection = FALSE, parallel_qsub = TRUE, send_email = NULL,
    keepNC = TRUE, forceRun = TRUE, run_parallel = TRUE, MCMC.args = NULL),
  ...
)
```

Arguments

settings	PEcAn settings object
obs.mean	Lists of date times named by time points, which contains lists of sites named by site ids, which contains observation means for each state variables of each site for each time point.
obs.cov	Lists of date times named by time points, which contains lists of sites named by site ids, which contains observation covariances for all state variables of each site for each time point.
Q	Process covariance matrix given if there is no data to estimate it.
restart	Used for iterative updating previous forecasts. Default NULL. List object includes file path to previous runs and start date for SDA.
pre_enkf_params	Used for passing pre-existing time-series of process error into the current SDA runs to ignore the impact by the differences between process errors.
ensemble.samples	Pass ensemble.samples from outside to avoid GitHub check issues.
control	List of flags controlling the behavior of the SDA. 'trace' for reporting back the SDA outcomes; 'TimeseriesPlot' for post analysis examination; 'debug' decide if we want to pause the code and examining the variables inside the function; 'pause' decide if we want to pause the SDA workflow at current time point t; 'Profiling' decide if we want to export the temporal SDA outputs in CSV file; 'OutlierDetection' decide if we want to execute the outlier detection each time after the model forecasting; 'parallel_qsub' decide if we want to execute the 'qsub' job submission under parallel mode; 'send_email' contains lists for sending email to report the SDA progress; 'keepNC' decide if we want to keep the NetCDF files inside the out directory; 'forceRun' decide if we want to proceed the Bayesian MCMC sampling without observations; 'run_parallel' decide if we want to run the SDA under parallel mode for the 'future_map' function; 'MCMC.args' include lists for controling the MCMC sampling process (iteration, nchains, burnin, and nthin.).
...	Additional arguments, currently ignored

Details

Restart mode: Basic idea is that during a restart (primary case envisioned as an iterative forecast), a new workflow folder is created and the previous forecast for the start_time is copied over. During restart the initial run before the loop is skipped, with the info being populated from the previous run. The function then dives right into the first Analysis, then continues on like normal.

Value

NONE

Author(s)

Michael Dietze, Ann Raiho and Alexis Helgeson <dietze@bu.edu>

`sda.enkf.original` *State Variable Data Assimilation: Ensemble Kalman Filter*

Description

Restart mode: Basic idea is that during a restart (primary case envisioned as an iterative forecast), a new workflow folder is created and the previous forecast for the start_time is copied over. During restart the initial run before the loop is skipped, with the info being populated from the previous run. The function then dives right into the first Analysis, then continues on like normal.

Usage

```
sda.enkf.original(
  settings,
  obs.mean,
  obs.cov,
  IC = NULL,
  Q = NULL,
  adjustment = TRUE,
  restart = NULL
)
```

Arguments

`settings`	PEcAn settings object
`obs.mean`	list of observations of the means of state variable (time X nstate)
`obs.cov`	list of observations of covariance matrices of state variables (time X nstate X nstate)
`IC`	initial conditions
`Q`	process covariance matrix given if there is no data to estimate it
`adjustment`	flag for using ensemble adjustment filter or not
`restart`	Used for iterative updating previous forecasts. This is a list that includes ens.inputs, the list of inputs by ensemble member, params, the parameters, and old_outdir, the output directory from the previous workflow. These three things are needed to ensure that if a new workflow is started that ensemble members keep there run-specific met and params. See Details

Value

NONE

Author(s)

Michael Dietze and Ann Raiho <dietze@bu.edu>

`SDA_control` *SDA_control*

Description

SDA_control

Usage

```
SDA_control(
  trace = TRUE,
  ForewardForecast = FALSE,
  interactivePlot = FALSE,
  TimeseriesPlot = FALSE,
  BiasPlot = FALSE,
  plot.title = NULL,
  facet.plots = FALSE,
  debug = FALSE,
  pause = FALSE,
  Profiling = FALSE,
  OutlierDetection = FALSE
)
```

Arguments

`trace`	Logical if code should print out the progress of SDA .
`ForewardForecast`	Logical if the foreward forecast estimates needs to be read and visualized in time series plots.
`interactivePlot`	Logical if time series plots need to be generated.
`TimeseriesPlot`	Logical if time series plots need to be generated.
`BiasPlot`	Logical if bias plots need to be generated
`plot.title`	Character defining the title of times series plots
`facet.plots`	Logical if the timeseries plots should be faceted based on state variables
`debug`	Logical if the code should stop at some milestones and open an interactive debugging environment
`pause`	Logical if code needs to be paused and wait for further instruction after the analysis step
`Profiling`	Logical if code should keep track of how much time each step took
`OutlierDetection`	Logical if TRUE then a simple method will be used to replace simulations of outside 3IQR with the median of ensembles.

Value

list of all arguments needed to setup the SDA function

`SDA_downscale` *SDA Downscale Function*

Description

This function uses either Random Forest or Convolutional Neural Network model based on the model_type parameter.

Usage

```
SDA_downscale(
  preprocessed,
  date,
  carbon_pool,
  covariates,
  model_type = "rf",
  seed = NULL
)
```

Arguments

`preprocessed`	List. Preprocessed data returned as an output from the SDA_downscale_preprocess function.
`date`	Date. If SDA site run, format is yyyy/mm/dd; if NEON, yyyy-mm-dd. Restricted to years within file supplied to 'preprocessed' from the 'data_path'.
`carbon_pool`	Character. Carbon pool of interest. Name must match carbon pool name found within file supplied to 'preprocessed' from the 'data_path'.
`covariates`	SpatRaster stack. Used as predictors in downscaling. Layers within stack should be named. Recommended that this stack be generated using 'covariates' instructions in assim.sequential/inst folder
`model_type`	Character. Either "rf" for Random Forest or "cnn" for Convolutional Neural Network. Default is Random Forest.
`seed`	Numeric or NULL. Optional seed for random number generation. Default is NULL.

Details

This function will downscale forecast data to unmodeled locations using covariates and site locations

Value

A list containing the training and testing data sets, models, predicted maps for each ensemble member, and predictions for testing data.

Author(s)

Joshua Ploshay, Sambhav Dixit

`SDA_downscale_hrly` *SDA Downscale Function for Hourly Data*

Description

This function uses the randomForest model to downscale forecast data (hourly) to unmodeled locations using covariates and site locations

Usage

```
SDA_downscale_hrly(nc_file, coords, yyyy, covariates)
```

Arguments

`nc_file`	In quotes, file path for .nc containing ensemble data.
`coords`	In quotes, file path for .csv file containing the site coordinates, columns named "lon" and "lat".
`yyyy`	In string, format is yyyy(year of interest)
`covariates`	SpatRaster stack, used as predictors in randomForest. Layers within stack should be named. Recommended that this stack be generated using 'covariates' instructions in assim.sequential/inst folder

Value

It returns the 'downscale_output' list containing lists for the training and testing data sets, models, and predicted maps for each ensemble member.

Author(s)

Harunobu Ishii

`SDA_downscale_metrics` *Calculate Metrics for Downscaling Results*

Description

This function takes the output from the SDA_downscale function and computes various performance metrics for each ensemble. It provides a way to evaluate the accuracy of the downscaling results without modifying the main downscaling function.

Usage

```
SDA_downscale_metrics(downscale_output, carbon_pool)
```

Arguments

`downscale_output`	List. Output from the SDA_downscale function, containing data, models, maps, and predictions for each ensemble.
`carbon_pool`	Character. Name of the carbon pool used in the downscaling process.

Details

This function calculates performance metrics for the downscaling results. It computes Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared for each ensemble. The function uses the actual values from the testing data and the predictions generated during the downscaling process.

Value

A list of metrics for each ensemble, where each element contains MAE , MSE ,R_squared ,actual values from testing data and predicted values for the testing data

Author(s)

Sambhav Dixit

`SDA_downscale_preprocess` *Preprocess Data for Downscaling*

Description

This function reads and checks the input data, ensuring that the required date and carbon pool exist, and that the site coordinates are valid.

Usage

```
SDA_downscale_preprocess(data_path, coords_path, date, carbon_pool)
```

Arguments

`data_path`	Character. File path for .rds containing ensemble data.
`coords_path`	Character. File path for .csv file containing the site coordinates, with columns named "lon" and "lat".
`date`	Date. If SDA site run, format is yyyy/mm/dd; if NEON, yyyy-mm-dd. Restricted to years within the file supplied to 'data_path'.
`carbon_pool`	Character. Carbon pool of interest. Name must match the carbon pool name found within the file supplied to 'data_path'.

Details

This function ensures that the specified date and carbon pool are present in the input data. It also checks the validity of the site coordinates and aligns the number of rows between site coordinates and carbon data.

Value

A list containing The read .rds data , The cleaned site coordinates, and the preprocessed carbon data.

Author(s)

Sambhav Dixit

`sda_matchparam` *sda_matchparam*

Description

sda_matchparam

Usage

```
sda_matchparam(settings, ensemble.samples, site.ids, nens)
```

Arguments

`settings`	settings object passed from sda.enkf_MultiSite
`ensemble.samples`	taken from sample.Rdata object
`site.ids`	character object passed from sda.enkf_MultiSite
`nens`	number of ensemble members in model runs, taken from restart$runids

Value

new.params object used to

Author(s)

Alexis Helgeson

`SDA_OBS_Assembler` *Assembler for preparing obs.mean and obs.cov for the SDA workflow*

Description

Assembler for preparing obs.mean and obs.cov for the SDA workflow

Usage

```
SDA_OBS_Assembler(settings)
```

Arguments

`settings`	the settings object followed by PEcAn.settings format.

Value

list of obs.mean and obs.cov

Author(s)

Dongchen Zhang

Examples

```
## Not run:
settings_dir <- "/projectnb/dietzelab/dongchen/All_NEON_SDA/NEON42/IC/pecan.xml"
settings <- PEcAn.settings::read.settings(settings_dir)
OBS <- SDA_OBS_Assembler(settings)

## End(Not run)
```

SDA_remote_launcher *SDA_remote_launcher*

Description

SDA_remote_launcher

Usage

```
SDA_remote_launcher(settingPath, ObsPath, run.bash.args)
```

Arguments

settingPath	The Path to the setting that will run SDA
ObsPath	Path to the obs data which is expected to be an .Rdata.
run.bash.args	Shell commands to be run on the remote host before launching the SDA. See examples

Value

This function returns a list of two pieces of information. One the remote path that SDA is running and the PID of the active run.

Examples

```
## Not run:
 # This example can be found under inst folder in the package
 library(PEcAn.all)
 library(purrr)

 run.bash.args <- c(
  "#$ -l h_rt=48:00:00",
  "#$ -pe omp 28 # Request a parallel environment with 4 cores",
  "#$ -l mem_per_core=1G # and 4G memory for each",
  "#$ -l buyin",
  "module load R/3.5.2",
  "module load python/2.7.13"
  )
 settingPath <-"pecan.SDA.4sites.xml"

 ObsPath <- "Obs/LandTrendr_AGB_output50s.RData"

 SDA_remote_launcher(settingPath, ObsPath, run.bash.args)

## End(Not run)
```

sda_weights_site *Calculate ensemble weights for each site at time t.*

Description

Calculate ensemble weights for each site at time t.

Usage

```
sda_weights_site(FORECAST, ANALYSIS, t, ens)
```

Arguments

FORECAST	FORECAST object built within the sda.enkf_MultiSite function.
ANALYSIS	ANALYSIS object built within the Analysis_sda_multisite function.
t	exact number of t inside the sda.enkf_MultiSite function.
ens	number of ensemble members.

Value

list of weights associated with each ensemble member of each site.

Author(s)

Dongchen Zhang and Hamze Dokoohaki

simple.local *simple.local*

Description

Adopted from Data assimilation for spatio-temporal processes - p250 - Sebastian Reich

Usage

```
simple.local(Pf, distance.mat, scalef = 2)
```

Arguments

Pf	Forecast error covariance matrix
distance.mat	is matrix of distances between sites.
scalef	scalef is a numeric value that requires tuning and it controls the shape of the corrolation function

Value

It returns a localized covariance matrix by taking a Schur product between Pf and a corrolation function

Author(s)

Hamze Dokoohaki

`tobit.model` *TWEnF*

Description

TWEnF

Usage

```
tobit.model
```

Format

TBD

`tobit2space.model` *Fit tobit prior to ensemble members*

Description

Fit tobit prior to ensemble members

Usage

```
tobit2space.model
```

Format

TBD

tobit_model_censored *tobit_model_censored*

Description

tobit_model_censored

Usage

```
tobit_model_censored(settings, X, var.names, mu.f, Pf, t)
```

Arguments

settings	(list) pecan standard settings list.
X	(numeric) A matrix contains ensemble forecasts (ensembles * variables).
var.names	(character) variable names.
mu.f	(numeric) forecast mean values.
Pf	(numeric) forecast covariance matrix.
t	(numeric) timestep. If t=1, initial values are imputed for zero values in mu.f

Value

list with updated mu.f, pf, X, and indication of which y values are censored

update_q *update_q*

Description

update_q

Usage

```
update_q(
  block.list.all,
  t,
  nt,
  aqq.Init = NULL,
  bqq.Init = NULL,
  MCMC_dat = NULL,
  block.list.all.pre = NULL
)
```

Arguments

`block.list.all`	each block within the 'block.list' lists.
`t`	time point.
`nt`	total length of time steps.
`aqq.Init`	the initial values of aqq, the default is NULL.
`bqq.Init`	the initial values of bqq, the default is NULL.
`MCMC_dat`	data frame of MCMC samples, the default it NULL.
`block.list.all.pre`	pre-existed block.list.all object for passing the aqq and bqq to the current SDA run, the default is NULL.

Value

It returns the 'block.list.all' object with initialized/updated Q filled in.

Author(s)

Dongchen Zhang

Index

∗ **datasets**
GEF.MultiSite.Nimble, 21
tobit.model, 48
tobit2space.model, 48

adj.ens, 3
aggregate, 4
alltocs, 5
alr, 6
Analysis.sda, 6
analysis_sda_block, 7
assess.params, 9
assessParams *(assess.params)*, 9

block.2.vector, 9
block_matrix, 10
build.block.xy, 11
build_X, 12

conj_wt_wishart_sampler, 13
Construc_H, 15
Construct.H.multisite, 13
Construct.R, 14
Construct_H *(Construc_H)*, 15
construct_nimble_H, 14
Contruct.Pf, 16
Create_Site_PFT_CSV, 17

dwtmnorm, 18

EnKF, 18
EnKF.MultiSite, 19

GEF, 20
GEF.MultiSite.Nimble, 21
get_ensemble_weights, 21
GrabFillMatrix, 22

hop_test, 22

interactive.plotting.sda, 23
inv.alr, 26

load_data_paleon_sda, 26
load_nimble, 27
Local.support, 27

matrix_network, 28
MCMC_block_function, 28
MCMC_function, 29
MCMC_Init, 29
metSplit, 30

Obs.data.prepare.MultiSite, 31
obs_timestep2timepoint, 31
outlier.detector.boxplot, 32

piecew.poly.local, 32
post.analysis.ggplot
(interactive.plotting.sda), 23
post.analysis.multisite.ggplot
(interactive.plotting.sda), 23
postana.bias.plotting.sda
(interactive.plotting.sda), 23
postana.timeser.plotting.sda
(interactive.plotting.sda), 23
Prep_OBS_SDA, 33

Remote_Sync_launcher, 33
rescaling_stateVars, 34
rwtmnorm, 34

sample.parameters, 35
sample_met, 36
sampler_toggle, 35
sda.enkf, 36
sda.enkf.multisite, 37
sda.enkf.original, 39

SDA_control, 40
SDA_downscale, 41
SDA_downscale_hrly, 42
SDA_downscale_metrics, 43
SDA_downscale_preprocess, 43
sda_matchparam, 44
SDA_OBS_Assembler, 45
SDA_remote_launcher, 46
SDA_timeseries_plot *(interactive.plotting.sda)*, 23
sda_weights_site, 47
simple.local, 47

tobit.model, 48
tobit2space.model, 48
tobit_model_censored, 49

update_q, 49

y_star_create *(load_nimble)*, 27