

Package: PEcAnAssimSequential (via r-universe)

October 16, 2026

Type Package

Title PEcAn Functions Used for Ecological Forecasts and Reanalysis

Version 1.10.0.9000

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.

URL <https://pecanproject.github.io>

BugReports <https://github.com/PecanProject/pecan/issues>

Depends R (>= 4.1.0)

Imports coda, dplyr, foreach, furrr, future, ggplot2, lubridate (>= 1.6.0), Matrix, ncd4, nimble, PEcAn.DB, PEcAn.logger, PEcAn.remote, PEcAn.settings, PEcAn.uncertainty, PEcAn.workflow, purrr, reticulate, rlang, stringr

Suggests abind, corrplot, doSNOW, exactextractr, mvtnorm, ggrepel, glue, ggpubr, gridExtra, keras3 (>= 1.0.0), itertools, magic (>= 1.5.0), methods, Metrics, PEcAn.benchmark, PEcAn.data.land, PEcAn.data.remote, PEcAn.utils, PEcAn.visualization, plotrix, plyr (>= 1.8.4), randomForest, raster, readr, reshape2 (>= 1.4.2), rlist, sf, sp, stats, terra, testthat (>= 3.1.9), tictoc, tidyr, utils, xgboost, XML

License BSD_3_clause + file LICENSE

Copyright Authors

Encoding UTF-8

RoxygenNote 7.3.3

X-schema.org-keywords state-data-assimilation, ensemble-kalman-filter,
particle-filter, ecological-forecasting, forecast-verification

Config/pak/sysreqs

libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make libmagick++-
dev gsfonts jags libicu-dev libpng-dev libuv1-dev libxml2-dev libnetcdf-dev libssl-
dev libproj-dev python3 libsqlite3-dev libudunits2-dev libx11-dev xz-utils zlib1g-dev

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

Date/Publication 2026-10-16 08:17:39 UTC

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

RemoteRef HEAD

RemoteSha fdb9b89d6133f78473728f63b7c1b3deb4ee29ec

RemoteSubdir modules/assim.sequential

Contents

adj.ens	4
aggregate	5
alltocs	6
alr	7
Analysis.sda	7
analysis_sda_block	8
assess.params	10
block.2.vector	10
block_matrix	11
build.block.xy	12
build_X	12
check.qsub.job.info	13
conj_wt_wishart_sampler	14
Construc_H	15
Construct.H.multisite	15
Construct.R	16
construct_nimble_H	17
Construct.Pf	18
Create_Site_PFT_CSV	19
debias.average	20
debias.extract.cov	20
debias.map.products	21
debias.ML	21
debias.res.lag.calc	22
debias.residual.calc	23
debias.train.accuracy	23
downscale_main	24
downscale_metrics	25
downscale_qsub_main	26
dwtmnorm	27
EnKF	27

EnKF.MultiSite	28
ensemble_downscale	29
GEF	29
GEF.MultiSite.Nimble	31
get_ensemble_weights	31
GrabFillMatrix	32
hop_test	32
interactive.plotting.sda	33
inv.alr	36
load_data_paleon_sda	36
load_nimble	37
Local.support	37
matrix_network	38
MCMC_block_function	38
MCMC_function	39
MCMC_Init	39
metSplit	40
Obs.data.prepare.MultiSite	41
obs_timestep2timepoint	41
outlier_detector_boxplot	42
parallel_prediction	42
parallel_train	43
pecan_settings_2_pts	44
piecew.poly.local	44
Prep_OBS_SDA	45
prepare_train_dat	46
qsub_sda	46
qsub_sda_batch	48
Remote_Sync_launcher	48
rescaling_stateVars	49
rwtmnorm	49
sample.parameters	50
sample_met	50
sampler_toggle	51
sda.bias.correction	51
sda.enkf	52
sda.enkf.multisite	53
sda.enkf.original	55
sda.enkf_local	56
sda.qsub.job.submission	58
sda_assemble	59
SDA_control	59
SDA_downscale	60
SDA_downscale_hrly	61
SDA_downscale_metrics	62
SDA_downscale_preprocess	63
sda_matchparam	64
SDA_OBS_Assembler	64

SDA_remote_launcher	65
sda_weights_site	66
simple.local	67
stack_covariates_2_df	67
stack_covariates_2_geotiff	68
subset_ensemble	69
tobit.model	70
tobit_model_censored	70
tobit2space.model	71
update_q	71

Index	73
--------------	-----------

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 Dokoochaki

aggregate	<i>Aggregation Function</i>
-----------	-----------------------------

Description

Aggregation Function

Usage

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

Arguments

<code>downscaled_output</code>	Raster file output from <code>downscale_function.R</code> . Read file in this way if stored locally: <code>downscaled_output <- readRDS("xxx.rds")</code>
<code>polygon_data</code>	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").
<code>func</code>	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	<i>Additive Log Ratio transform</i>
-----	-------------------------------------

Description

Additive Log Ratio transform

Usage

```
alr(y)
```

Arguments

y	state var
---	-----------

Analysis.sda	<i>Analysis.sda</i>
--------------	---------------------

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 Dokoochaki

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

<code>settings</code>	pecan standard multi-site settings list.
<code>block.list.all</code>	Lists of forecast and analysis outputs for each time point of each block. If <code>t=1</code> , we initialize those outputs of each block with NULL from the ‘sda.enkf.multisite’ function.
<code>X</code>	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.
<code>obs.mean</code>	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.
<code>obs.cov</code>	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.
<code>t</code>	time point in format of YYYY-MM-DD.
<code>nt</code>	total length of time steps, corresponding to the ‘nt’ variable in the ‘sda.enkf.multisite’ function.
<code>MCMC.args</code>	arguments for the MCMC sampling, details can be found in the roxygen structure for control list in the ‘sda.enkf.multisite’ function.
<code>block.list.all.pre</code>	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

<code>assess.params</code>	<i>assess.params</i>
----------------------------	----------------------

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

<code>dat</code>	MCMC output
<code>Xt</code>	ensemble output matrix
<code>wts</code>	ensemble weights
<code>mu_f_TRUE</code>	muf before tobit2space
<code>P_f_TRUE</code>	Pf before tobit2space

Value

make plots

Author(s)

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

<code>block.2.vector</code>	<i>block.2.vector</i>
-----------------------------	-----------------------

Description

block.2.vector

Usage

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

Arguments

<code>block.list</code>	lists of blocks generated by the ‘build.block.xy’ function.
<code>X</code>	A matrix contains ensemble forecasts.
<code>H</code>	H index created by the ‘construct_nimble_H’ function.
<code>adjustment</code>	logical variable determine if we want to adjust the analysis ensembles based on likelihood.

Value

It returns a list of analysis results by MCMC sampling.

Author(s)

Dongchen Zhang

<code>block_matrix</code>	<i>block_matrix</i>
---------------------------	---------------------

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. <code>#'</code> .

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

<code>build.block.xy</code>	<i>build.block.xy</i>
-----------------------------	-----------------------

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

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

Details

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

Author(s)

Dongchen Zhang

<code>build_X</code>	<i>build_X</i>
----------------------	----------------

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

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

Value

X ready to be passed to SDA Analysis code

Author(s)

Alexis Helgeson

`check.qsub.job.info` *sda_assemble*

Description

This function can help to assemble sda outputs (analysis and forecasts) from each job execution.

Usage

```
check.qsub.job.info(username)
```

Arguments

username character: username for the HPC.

Value

list: number of jobs and vector of job ids.

Author(s)

Dongchen Zhang.

conj_wt_wishart_sampler
<i>Weighted conjugate wishart</i>

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

`Construc_H`*Construc_H*

Description

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

Usage

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

Arguments

<code>choose</code>	a vector of observations indices ordered based on their appearances in the list of state variable names.
<code>Y</code>	vector of observations
<code>X</code>	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 Dokoochaki

`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, products = NULL)
```

Arguments

<code>site.ids</code>	a vector name of site ids
<code>var.names</code>	vector names of state variable names
<code>obs.t.mean</code>	list of vector of means for the time t for different sites.
<code>products</code>	vector names of products for each observation.

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

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

construct_nimble_H	<i>construct_nimble_H</i>
--------------------	---------------------------

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",
  products = NULL
)
```

Arguments

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

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

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

<code>site.ids</code>	a vector name of site ids.
<code>var.names</code>	vector names of state variable names.
<code>X</code>	a matrix of state variables. In this matrix rows represent ensembles, while columns show the variables for different sites.
<code>localization.FUN</code>	This is the function that performs the localization of the Pf matrix and it returns a localized matrix with the same dimensions.
<code>t</code>	not used
<code>blocked.dis</code>	passed to ‘localization.FUN’
<code>...</code>	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	<i>Title Identify pft for each site of a multi-site settings using NLCD and Eco-region</i>
---------------------	--

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

<code>settings</code>	a multi-site settings
<code>Ecoregion</code>	path of Ecoregion data (*.shp)
<code>NLCD</code>	path of NLCD img file
<code>con</code>	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)
```

<code>debias.average</code>	<i>debias.average</i>
-----------------------------	-----------------------

Description

This function helps to average the predicted residuals based on certain criteria. The default function is averaging. TODO: we will be averaging residuals based on ML precision for each residual prediction.

Usage

```
debias.average(residuals, by_product_accuracy = NULL)
```

Arguments

<code>residuals</code>	vector: predicted residuals across products.
<code>by_product_accuracy</code>	vector: the out-of-sample RMSE for each product.

Value

numeric: an averaged residual.

Author(s)

Dongchen Zhang

<code>debias.extract.cov</code>	<i>debias.extract.cov</i>
---------------------------------	---------------------------

Description

This function helps to extract the covariates based on date and coordinates from the settings object.

Usage

```
debias.extract.cov(cov.dir, date, site.info)
```

Arguments

<code>cov.dir</code>	character: physical path to the directory contains the time series covariate maps.
<code>date</code>	numeric or character: date that used to find the corresponding covariate file.
<code>site.info</code>	data.frame: data.frame that contains longitude and latitude in its first and second column.

Value

data.frame: a data frame containing M (rows) of locations across N (columns) variables.

Author(s)

Dongchen Zhang

debias.map.products	<i>debias.map.products</i>
---------------------	----------------------------

Description

This function helps to create the mapping operator that maps between product names and variable names by locations.

Usage

```
debias.map.products(obs.mean.t, obs_prep)
```

Arguments

obs.mean.t	list: lists of sites that contain observations for each variable.
obs_prep	list: lists contains basic information for each product of each observation.

Value

matrix: a matrix containing M (rows) of locations*variables across N (columns) products.

Author(s)

Dongchen Zhang

debias.ML	<i>sda.bias.correction</i>
-----------	----------------------------

Description

This function helps to correct the forecasts' biases based on ML (random forest) training on the previous time points.

Usage

```
debias.ML(pred.name, cov.names, dat.train, dat.pred, var.name, py.init)
```

Arguments

<code>pred.name</code>	character: the name for the predictive variable.
<code>cov.names</code>	character: the name for the covariates.
<code>dat.train</code>	data.frame: data frame containing associated covariates and predictive variable for ML training.
<code>dat.pred</code>	data.frame: data frame containing associated covariates for prediction.
<code>var.name</code>	character: variable name to be predicted.
<code>py.init</code>	R function: R function to initialize the python functions. Default is NULL. the default random forest will be used if 'py.init' is NULL.

Value

list: the ML predicted residuals and other ML outputs.

Author(s)

Dongchen Zhang

`debias.res.lag.calc` *debias.res.lag.calc*

Description

This function helps to calculate the lagged residual error time series.

Usage

```
debias.res.lag.calc(res.ts)
```

Arguments

`res.ts` list: lists of residuals at each time point.

Value

list: lists of lagged residuals between time points.

Author(s)

Dongchen Zhang

 debias.residual.calc *debias.residual.calc*

Description

This function helps to calculate the residual error for a certain time point and variable.

Usage

```
debias.residual.calc(obs.mean, all.X, t, var.name, data.source)
```

Arguments

<code>obs.mean</code>	List: 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.
<code>all.X</code>	list: lists of data frame of model forecast from the beginning to the current time points that has n (ensemble size) rows and n.var (number of variables) times n.site (number of locations) columns. (e.g., 100 ensembles, 4 variables, and 8,000 locations will end up with data.frame of 100 rows and 32,000 columns).
<code>t</code>	numeric: the current number of time points (e.g., t=1 for the beginning time point).
<code>var.name</code>	character: variable name to be predicted.
<code>data.source</code>	character: product name of the data.

Value

list: lists of residuals between forecasts and observations across at time t.

Author(s)

Dongchen Zhang

 debias.train.accuracy
 debias.train.accuracy

Description

This function helps to calculate the out-of-sample accuracy of residual predictions.

Usage

```

debias.train.accuracy(
  pred.name,
  cov.names,
  dat.train,
  var.name,
  py.init,
  ratio = 0.8
)

```

Arguments

<code>pred.name</code>	character: the name for the predictive variable.
<code>cov.names</code>	character: the name for the covariates.
<code>dat.train</code>	data.frame: data frame containing associated covariates and predictive variable for ML training.
<code>var.name</code>	character: variable name to be predicted.
<code>py.init</code>	R function: R function to initialize the python functions. Default is NULL. the default random forest will be used if 'py.init' is NULL.
<code>ratio</code>	numeric from 0 to 1: define the ratio of samples used for training. The rest samples will be used for calculating the out-of-sample accuracy. Default is 0.8.

Value

list: the variable importance of the ML and RMSE of the out-of-sample predictions.

Author(s)

Dongchen Zhang

<code>downscale_main</code>	<i>downscale_main</i>
-----------------------------	-----------------------

Description

This is the main function to execute the machine learning training and prediction. Note it will be deployed by each node you requested if the qsub feature is enabled below.

Usage

```

downscale_main(
  settings,
  analysis,
  covariates.dir,
  time,

```

```

    variable,
    outdir,
    base.map.dir,
    method = "randomForest",
    cores = parallel::detectCores()
  )

```

Arguments

settings	character: physical path that points to the pecan settings XML file.
analysis	numeric: data frame (rows: ensemble member; columns: site*state_variables) of updated ensemble analysis results from the ‘sda_enkf’ function.
covariates.dir	character: path to the exported covariates GeoTIFF file.
time	character: the time tag used to differentiate the outputs from others.
variable	character: name of state variable. It should match up with the column names of the analysis data frame.
outdir	character: the output directory where the downscaled maps will be stored.
base.map.dir	character: path to the GeoTIFF file within which the extents and CRS will be used to generate the ensemble maps.
method	character: machine learning method, default is randomForest (currently support randomForest and xgboost).
cores	numeric: how many CPUs to be used in the calculation, the default is the total CPU number you have.

Value

paths to the ensemble downscaled maps.

Author(s)

Dongchen Zhang

downscale_metrics	<i>Calculate Metrics for Downscaling Results</i>
--------------------------	--

Description

This function takes the output from the 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

```
downscale_metrics(downscale_output)
```

Arguments**downscale_output**

List. Output from the downscale function, containing data, models, maps, predictions, and test predictions for each ensemble.

Details

This function calculates performance metrics for the downscaling results. It computes Root Mean Squared Error (RMSE), 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 data.frame with mean, RMSE, MAE, R-squared, and coefficient of variation for each ensemble.

Author(s)

Sambhav Dixit, David LeBauer

downscale_qsub_main	<i>downscale_qsub_main</i>
---------------------	----------------------------

Description

This qsub function helps to run the submitted qsub jobs for running the downscale_main function.

Usage

```
downscale_qsub_main(folder.path)
```

Arguments

folder.path Character: physical path to which the job file is located.

Author(s)

Dongchen Zhang

dwtmnorm	<i>weighted multivariate normal density</i>
----------	---

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	<i>EnKF</i>
------	-------------

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 process 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 Dokoochaki

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 Dokoochaki

ensemble_downscale	<i>Ensemble Downscale</i>
--------------------	---------------------------

Description

Ensemble Downscale

Usage

```
ensemble_downscale(ensemble_data, site_coords, covariates)
```

Arguments

- ensemble_data** EFI standard tibble or data.frame. Contains carbon data for downscaling.
- site_coords** data.frame, tibble, or sf object. Design points. If not sf object, must have 'lon' and 'lat' columns. Must have a unique 'site_id' field.
- covariates** table containing numeric predictors to be used in downscaling. Must have unique identifier 'site_id' field and predictor attributes

Details

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

Value

A list containing the model, predictions for all values of covariates as well as test data and test predictions for downstream statistics.

Author(s)

Joshua Ploshay, Sambhav Dixit, David LeBauer

GEF	<i>GEF</i>
-----	------------

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 Dokoochaki

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

<code>settings</code>	PEcAn settings object
<code>time_do</code>	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	<i>GrabFillMatrix</i>
----------------	-----------------------

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	<i>hop_test</i>
----------	-----------------

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, bq	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	<i>inverse of ALR transform</i>
---------	---------------------------------

Description

inverse of ALR transform

Usage

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

Arguments

alr	state var
-----	-----------

load_data_paleon_sda	<i>load_data_paleon_sda</i>
----------------------	-----------------------------

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	<i>load_nimble</i>
-------------	--------------------

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 Dokoochaki

Local.support	<i>Local.support</i>
---------------	----------------------

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 correlation function

Author(s)

Hamze Dokoochaki

<code>matrix_network</code>	<i>matrix_network</i>
-----------------------------	-----------------------

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

<code>MCMC_block_function</code>	<i>MCMC_block_function</i>
----------------------------------	----------------------------

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	<i>MCMC_function</i>
---------------	----------------------

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 impenmented 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	<i>MCMC_Init</i>
-----------	------------------

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

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

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

<code>obs.path</code>	Path to the obs data which is expected to be an .Rdata.
<code>site.ids</code>	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

<code>start.date</code>	start date when the first observation was taken.
<code>end.date</code>	end date when the last observation was taken.
<code>timestep</code>	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
```

Replace outliers using boxplot statistics

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

```
parallel_prediction    parallel_prediction
```

Description

This function helps to predict the target variable observations based on the covariates. The prediction is working in parallel across vegetated pixels.

Usage

```
parallel_prediction(
  base.map.dir,
  models,
  cov.vecs,
  non.na.inds,
  outdir,
  name,
  cores = parallel::detectCores()
)
```

Arguments

<code>base.map.dir</code>	character: path to the GeoTIFF file within which the extents and CRS will be used to generate the ensemble maps.
<code>models</code>	list: trained models across ensemble members generated by the ‘parallel_train’ function.
<code>cov.vecs</code>	numeric: data frame containing covaraites across vegetated pixels generated from the ‘stack_covariates_2_df’ function.
<code>non.na.inds</code>	numeric: the corresponding index of vegetated pixels generated from the ‘stack_covariates_2_df’ function.
<code>outdir</code>	character: the output directory where the downscaled maps will be stored.
<code>name</code>	list: containing the time and variable name to create the final GeoTIFF file name.
<code>cores</code>	numeric: how many CPUs to be used in the calculation, the default is the total CPU number you have.

Value

paths to the ensemble downscaled maps.

Author(s)

Dongchen Zhang

<code>parallel_train</code>	<i>parallel_train</i>
-----------------------------	-----------------------

Description

This function helps to train the ML model across ensemble members in parallel.

Usage

```
parallel_train(
  full_data,
  method = "randomForest",
  cores = parallel::detectCores()
)
```

Arguments

<code>full_data</code>	numeric: the matrix generated using the ‘prepare_train_dat’ function.
<code>method</code>	character: machine learning method (currently support randomForest and xgboost).
<code>cores</code>	numeric: how many CPUs to be used in the calculation, the default is the total CPU number you have.

Value

list of trained models across ensemble members.

Author(s)

Dongchen Zhang

`pecan_settings_2_pts` *pecan_settings_2_pts*

Description

convert settings to geospatial points in terra.

Usage

```
pecan_settings_2_pts(settings)
```

Arguments

settings PEcAn settings: either a character that points to the settings or shape file or the actual pecan settings object will be accepted.

Value

terra spatial points object.

Author(s)

Dongchen Zhang

`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

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

Value

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

Author(s)

Hamze Dokoochaki

`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

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

Value

mean and covariance of observations

<code>prepare_train_dat</code>	<i>prepare_train_dat</i>
--------------------------------	--------------------------

Description

This function helps to create the training dataset of specific variable type and locations for downscaling. TODO: Add a ratio argument (training sample size/total sample size) so that we could calculate the out-of-sample accuracy.

Usage

```
prepare_train_dat(pts, analysis, covariates.dir, variable)
```

Arguments

pts	spatialpoints: spatial points returned by ‘terra::vectors’ function.
analysis	numeric: data frame (rows: ensemble member; columns: site*state_variables) of updated ensemble analysis results from the ‘sda_enkf’ function.
covariates.dir	character: path to the exported covariates GeoTIFF file.
variable	character: name of state variable. It should match up with the column names of the analysis data frame.

Value

matrix (num.sites, num.variables * num.ensemble + num.covariates) within which the first sets of columns contain values of state variables for each ensemble member of every site, and the rest columns contain the corresponding covariates.

Author(s)

Dongchen Zhang

<code>qsub_sda</code>	<i>qsub_sda</i>
-----------------------	-----------------

Description

This function provides means to split large SDA runs into separate ‘qsub’ jobs. Including job creation, submission, and assemble.

Usage

```
qsub_sda(
  settings,
  obs.mean,
  obs.cov,
  Q,
  pre_enkf_params,
  ensemble.samples,
  outdir,
  control,
  block.index = NULL,
  debias = list(cov.dir = NULL, start.year = NULL, t.start = NULL, residual.lag = NULL,
    fun = NULL),
  prefix = "batch"
)
```

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.
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.
outdir	Physical path to the folder where the SDA outputs will be stored. The default is NULL, where we will be using outdir from the settings object.
control	List of flags controlling the behavior of the SDA. ‘TimeseriesPlot’ for post analysis examination; ‘OutlierDetection’ decide if we want to execute the outlier detection each time after the model forecasting; ‘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; ‘MCMC.args’ include lists for controlling the MCMC sampling process (iteration, nchains, burnin, and nthin.).
block.index	list of site ids for each block, default is NULL. This is used when the localization turns on. Please keep using the default value because the localization feature is still in development.
debias	List: R list containing the covariance directory (‘cov.dir’), start year (‘start.year’), start time point (‘t.start’), if we want to include the ‘residual.lag’ as additional covariate, and the ML function (‘fun’). covariance

directory should include GeoTIFF files named by year. start time point is numeric input which decide when to start the debiasing feature. 'residual.lag' is either TRUE or FALSE.

prefix character: the desired folder name to store the outputs.

Value

NONE

Author(s)

Dongchen Zhang

qsub_sda_batch	<i>qsub_sda_batch</i>
----------------	-----------------------

Description

This function can help to execute sda function.

Usage

qsub_sda_batch(folder.path)

Arguments

folder.path character: path where the 'configs.rds' file is stored.

Author(s)

Dongchen Zhang.

Remote_Sync_launcher	<i>Remote_Sync_launcher</i>
----------------------	-----------------------------

Description

Remote_Sync_launcher

Usage

Remote_Sync_launcher(settingPath, remote.path, PID)

Arguments

<code>settingPath</code>	Path to your local setting .
<code>remote.path</code>	Path generated by SDA_remote_launcher which shows the path to your remote SDA run.
<code>PID</code>	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

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

Value

rescaled Matrix

`rwtmnorm` *random weighted multivariate normal*

Description

random weighted multivariate normal

Usage

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

Arguments

<code>n</code>	sample size
<code>mean</code>	mean
<code>prec</code>	precision
<code>wt</code>	weight

<code>sample.parameters</code>	<i>sample parameters</i>
--------------------------------	--------------------------

Description

sample parameters

Usage

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

Arguments

<code>ne</code>	number of ensemble members
<code>settings</code>	PEcAn settings object
<code>con</code>	PEcAn database connection

Value

data frame of sampled parameters from the posterior distribution

Author(s)

Michael Dietze <dietze@bu.edu>

<code>sample_met</code>	<i>Sample meteorological ensembles</i>
-------------------------	--

Description

Sample meteorological ensembles

Usage

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

Arguments

<code>settings</code>	PEcAn settings list
<code>nens</code>	number of ensemble members to be sampled

sampler_toggle	<i>sampler toggling</i>
----------------	-------------------------

Description

sampler toggling

Usage

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

Arguments

model	model
mvSaved	copied to
target	thing being targetted
control	unused

sda.bias.correction	<i>sda.bias.correction</i>
---------------------	----------------------------

Description

This function helps to correct the forecasts' biases based on ML (random forest) training on the previous time points.

Usage

```
sda.bias.correction(  
  settings,  
  t,  
  t.start,  
  dates,  
  all.X,  
  obs.mean,  
  state.interval,  
  cov.dir,  
  residual.lag = FALSE,  
  py.init = NULL  
)
```

Arguments

<code>settings</code>	PEcAn settings object.
<code>t</code>	numeric: the current number of time points (e.g., <code>t=1</code> for the beginning time point).
<code>t.start</code>	numeric: the user-defined time point to avoid the initial burnin period.
<code>dates</code>	vector: a vector of dates used for extracting covariates through time.
<code>all.X</code>	list: lists of data frame of model forecast from the beginning to the current time points that has <code>n</code> (ensemble size) rows and <code>n.var</code> (number of variables) times <code>n.site</code> (number of locations) columns. (e.g., 100 ensembles, 4 variables, and 8,000 locations will end up with data.frame of 100 rows and 32,000 columns)
<code>obs.mean</code>	List: 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.
<code>state.interval</code>	matrix: containing the upper and lower boundaries for each state variable.
<code>cov.dir</code>	character: physical path to the directory contains the time series covariate maps.
<code>residual.lag</code>	logical: decide if we want to include the lagged residual (difference in residual between time points) in the ML process.
<code>py.init</code>	R function: R function to initialize the python functions. Default is NULL. the default random forest will be used if 'py.init' is NULL.

Value

list: the current X after the bias-correction; the ML outputs for each variable; predicted residuals.

Author(s)

Dongchen Zhang

<code>sda.enkf</code>	<i>State Variable Data Assimilation: Ensemble Kalman Filter and Generalized ensemble filter</i>
-----------------------	---

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	<i>State Variable Data Assimilation: Ensemble Kalman Filter and Generalized ensemble filter</i>
---------------------------	---

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(TimeseriesPlot = FALSE, OutlierDetection = FALSE, send_email = NULL,
        keepNC = TRUE, forceRun = TRUE, run_parallel = TRUE, MCMC.args = NULL, merge_nc =
        TRUE, execution = "local"),
    debias = list(cov.dir = NULL, t.start = NULL, residual.lag = 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	list of ensemble parameters across PFTs. Default is NULL.
control	List of flags controlling the behavior of the SDA. ‘TimeseriesPlot’ for post analysis examination; ‘OutlierDetection’ decide if we want to execute the outlier detection each time after the model forecasting; ‘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 controlling the MCMC sampling process (iteration, nchains, burnin, and nthin.). ‘merge_nc’ determine if we want to merge all netCDF files across sites and ensembles. If it’s set as ‘TRUE’, we will then combine all netCDF files into the ‘merged_nc’ folder within the ‘outdir’. ‘execution’ decide the way we want to execute model including ‘local’, where we execute the

	model locally; ‘qsub’, where we use the traditional ‘start_model_runs’ function for submission; ‘qsub_parallel’, where we first combine jobs and submit them into the SCC.
debias	List: R list containing the covariance directory, the start time point, and if we want to include the ‘residual.lag’ as additional covariate. covariance directory should include GeoTIFF files named by year. start time point is numeric input which decide when to start the debiasing feature. ‘residual.lag’ is either TRUE or FALSE.
...	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	<i>State Variable Data Assimilation: Ensemble Kalman Filter</i>
--------------------------	---

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.enkf_local	<i>sda.enkf_local</i>
----------------	-----------------------

Description

This function provides complete support for the multi-core and multi-node computation on the general HPC system. Thus, this script will be more computationally efficient, making it possible to run SDA over thousands of locations.

Usage

```
sda.enkf_local(
  settings,
  obs.mean,
  obs.cov,
  Q = NULL,
  pre_enkf_params = NULL,
  ensemble.samples = NULL,
  outdir = NULL,
  control = list(TimeseriesPlot = FALSE, OutlierDetection = FALSE, send_email = NULL,
    keepNC = TRUE, forceRun = TRUE, MCMC.args = NULL, merge_nc = TRUE),
  debias = list(cov.dir = NULL, start.year = NULL, t.start = NULL, residual.lag = NULL,
    fun = NULL)
)
```

Arguments

<code>settings</code>	PEcAn settings object
<code>obs.mean</code>	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.
<code>obs.cov</code>	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.
<code>Q</code>	Process covariance matrix given if there is no data to estimate it.
<code>pre_enkf_params</code>	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.
<code>ensemble.samples</code>	list of ensemble parameters across PFTs. Default is NULL.
<code>outdir</code>	physical path to the folder that stores the SDA outputs. Default is NULL.
<code>control</code>	List of flags controlling the behavior of the SDA. ‘TimeseriesPlot’ for post analysis examination; ‘OutlierDetection’ decide if we want to execute the outlier detection each time after the model forecasting; ‘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; ‘MCMC.args’ include lists for controlling the MCMC sampling process (iteration, nchains, burnin, and nthin.). ‘merge_nc’ determine if we want to merge all netCDF files across sites and ensembles. If it’s set as ‘TRUE’, we will then combine all netCDF files into the ‘merged_nc’ folder within the ‘outdir’.
<code>debias</code>	List: R list containing the covariance directory (‘cov.dir’), start year (‘start.year’), start time point (‘t.start’), if we want to include the ‘residual.lag’ as additional covariate, and the ML function (‘fun’). covariance directory should include GeoTIFF files named by year. start time point is numeric input which decide when to start the debiasing feature. ‘residual.lag’ is either TRUE or FALSE.

Value

NONE

Author(s)

Dongchen Zhang <zhangdc@bu.edu>

```
sda.qsub.job.submission  
      sda_assemble
```

Description

This function can help to assemble sda outputs (analysis and forecasts) from each job execution.

Usage

```
sda.qsub.job.submission(  
  batch.folder,  
  username,  
  outdir = NULL,  
  past.job.ids = NULL,  
  max.job = 20,  
  prefix = "sda.all.forecast.analysis.Rdata",  
  resources = list(hour = 24, memory = "4G", ncpu = 28)  
)
```

Arguments

<code>batch.folder</code>	character: path where the SDA batch jobs stored.
<code>username</code>	character: username for the HPC.
<code>outdir</code>	character: path where to stored the assembled results. Default is NULL.
<code>past.job.ids</code>	vector: vector of the past submitted job ids. Default is NULL.
<code>max.job</code>	numeric: maximum allowance for the number of running jobs on HPC.
<code>prefix</code>	character: file name to be determined if a job is finished or not.
<code>resources</code>	list: computation resources (time, memory, cores) when used to submit jobs.

Value

vector or numeric: return -1 when all jobs are finished; return vector of submitted job ids when there is any job that is not finished.

Author(s)

Dongchen Zhang.

sda_assemble	<i>sda_assemble</i>
--------------	---------------------

Description

This function can help to assemble sda outputs (analysis and forecasts) from each job execution.

Usage

```
sda_assemble(batch.folder, outdir)
```

Arguments

<code>batch.folder</code>	character: path where the SDA batch jobs stored.
<code>outdir</code>	character: path where we want to store the assembled analysis and forecasts.

Author(s)

Dongchen Zhang.

SDA_control	<i>SDA_control</i>
-------------	--------------------

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

<code>trace</code>	Logical if code should print out the progress of SDA .
<code>ForewardForecast</code>	Logical if the foreward forecast estimates needs to be read and visualized in time series plots.
<code>interactivePlot</code>	Logical if time series plots need to be generated.
<code>TimeseriesPlot</code>	Logical if time series plots need to be generated.
<code>BiasPlot</code>	Logical if bias plots need to be generated
<code>plot.title</code>	Character defining the title of times series plots
<code>facet.plots</code>	Logical if the timeseries plots should be faceted based on state variables
<code>debug</code>	Logical if the code should stop at some milestones and open an interactive debugging environment
<code>pause</code>	Logical if code needs to be paused and wait for further instruction after the analysis step
<code>Profiling</code>	Logical if code should keep track of how much time each step took
<code>OutlierDetection</code>	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

<code>SDA_downscale</code>	<i>SDA Downscale Function</i>
----------------------------	-------------------------------

Description

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

Usage

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

Arguments

preprocessed	List. Preprocessed data returned as an output from the SDA_downscale_preprocess function.
carbon_pool	Character. Carbon pool of interest. Name must match the carbon pool in 'preprocessed'.
covariates	SpatRaster stack or sf object. Used as predictors in downscaling. If providing a raster stack, layers should be named. If providing an sf object, predictor attributes should be present.
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 checks the input data, ensuring that the required date and carbon pool exist, and that the site coordinates are valid.

Usage

```
SDA_downscale_preprocess(ensemble_data, site_coords, date, carbon_pool)
```

Arguments

ensemble_data	List where each element is a data.frame indexed by date.
site_coords	A data.frame with 'lon' and 'lat' columns, or an sf object with point geometries.
date	Date. The date for the run, which must occur within 'ensemble_data'.
carbon_pool	Character. Carbon pool of interest. Name must match a column in 'ensemble_data'.

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 requires site coordinates and carbon data to have the same number of rows.

Value

A list containing the input data, cleaned site coordinates, and preprocessed carbon data.

Author(s)

Sambhav Dixit

David LeBauer

<code>sda_matchparam</code>	<i>sda_matchparam</i>
-----------------------------	-----------------------

Description

`sda_matchparam`

Usage

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

Arguments

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

Value

`new.params` object used to

Author(s)

Alexis Helgeson

<code>SDA_OBS_Assembler</code>	<i>Assembler for preparing obs.mean and obs.cov for the SDA workflow</i>
--------------------------------	--

Description

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

Usage

```
SDA_OBS_Assembler(settings)
```

Arguments

<code>settings</code>	the settings object followed by <code>PEcAn.settings</code> 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	<i>Calculate ensemble weights for each site at time t.</i>
------------------	--

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 Dokoochaki

simple.local	<i>simple.local</i>
--------------	---------------------

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 correlation function

Author(s)

Hamze Dokoohaki

stack_covariates_2_df	<i>stack_covariates_2_df</i>
-----------------------	------------------------------

Description

This function helps to build the data frame (pixels by data columns) for only vegetated pixels to improve the efficiency. Note that the 'LC' field using the 'MODIS land cover' observations (MCD12Q1.061) must be supplied in the covariates to make this function work.

Usage

```
stack_covariates_2_df(rast.dir, cores = parallel::detectCores())
```

Arguments

<code>rast.dir</code>	character: a character that points to the covariates raster file generated by the ‘ <code>stack_covariates_2_geotiff</code> ’ function.
<code>cores</code>	numeric: how many CPUs to be used in the calculation, the default is the total CPU number you have.

Value

list containing the data frame of covariates for vegetated pixels and the corresponding index of the pixels.

Author(s)

Dongchen Zhang

```
stack_covariates_2_geotiff
      stack_covariates_2_geotiff
```

Description

This function helps to stack target data layers from various GeoTIFF maps (with different extents, CRS, and resolutions) to a single map.

Usage

```
stack_covariates_2_geotiff(
  outdir,
  year,
  base.map.dir,
  cov.tif.file.list,
  normalize = T,
  cores = parallel::detectCores()
)
```

Arguments

<code>outdir</code>	character: the output directory where the stacked GeoTIFF file will be generated.
<code>year</code>	numeric: the year of when the covariates are stacked.
<code>base.map.dir</code>	character: path to the GeoTIFF file within which the extents and CRS will be used to generate the final map.
<code>cov.tif.file.list</code>	list: a list contains sub-lists with each including path to the corresponding map and the variables to be extracted (e.g., <code>list(LC = list(dir = "path/to/landcover.tif", var.name = "LC"))</code>).

normalize	boolean: decide if we want to normalize each data layer, the default is TRUE.
cores	numeric: how many CPUs to be used in the calculation, the default is the total CPU number you have.

Value

path to the exported GeoTIFF file.

Author(s)

Dongchen Zhang

subset_ensemble	<i>Subset ensemble data for downscaling</i>
------------------------	---

Description

Subset ensemble data for downscaling

Usage

```
subset_ensemble(ensemble_data, site_coords, date, carbon_pool)
```

Arguments

ensemble_data	EFI standard tibble or data.frame
site_coords	A data.frame with a unique 'site_id' column.
date	Date. The date for the run, must be a date within 'ensemble_data'.
carbon_pool	Character. Carbon pool of interest. Name must match a value in 'ensemble_data\$variable'.

Details

This function subsets ensemble data and ensures that the specified date and carbon pool are present in the ensemble data.

Value

A data.frame containing 'site_id', 'ensemble', and 'prediction' for the specified date and carbon pool.

Author(s)

Samhavad Dixit, David LeBauer

<code>tobit.model</code>	<i>TWEnF</i>
--------------------------	--------------

Description

TWEnF

Usage

`tobit.model`

Format

TBD

<code>tobit_model_censored</code>	<i>tobit_model_censored</i>
-----------------------------------	-----------------------------

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

tobit2space.model	<i>Fit tobit prior to ensemble members</i>
-------------------	--

Description

Fit tobit prior to ensemble members

Usage

```
tobit2space.model
```

Format

TBD

update_q	<i>update__q</i>
----------	------------------

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, 31
 - tobit.model, 70
 - tobit2space.model, 71
- adj.ens, 4
- aggregate, 5
- alltocs, 6
- alr, 7
- Analysis.sda, 7
- analysis_sda_block, 8
- assess.params, 10
- assessParams (*assess.params*), 10
- block.2.vector, 10
- block_matrix, 11
- build.block.xy, 12
- build_X, 12
- check.qsub.job.info, 13
- conj_wt_wishart_sampler, 14
- Construc_H, 15
- Construct.H.multisite, 15
- Construct.R, 16
- Construct_H (*Construc_H*), 15
- construct_nimble_H, 17
- Construct.Pf, 18
- Create_Site_PFT_CSV, 19
- debias.average, 20
- debias.extract.cov, 20
- debias.map.products, 21
- debias.ML, 21
- debias.res.lag.calc, 22
- debias.residual.calc, 23
- debias.train.accuracy, 23
- downscale_main, 24
- downscale_metrics, 25
- downscale_qsub_main, 26
- dwtmnorm, 27
- EnKF, 27
- EnKF.MultiSite, 28
- ensemble_downscale, 29
- GEF, 29
- GEF.MultiSite.Nimble, 31
- get_ensemble_weights, 31
- GrabFillMatrix, 32
- hop_test, 32
- interactive.plotting.sda, 33
- inv.alr, 36
- load_data_paleon_sda, 36
- load_nimble, 37
- Local.support, 37
- matrix_network, 38
- MCMC_block_function, 38
- MCMC_function, 39
- MCMC_Init, 39
- metSplit, 40
- Obs.data.prepare.MultiSite, 41
- obs_timestep2timepoint, 41
- outlier_detector_boxplot, 42
- parallel_prediction, 42
- parallel_train, 43
- pecan_settings_2_pts, 44
- piecew.poly.local, 44
- post.analysis.ggplot
 - (*interactive.plotting.sda*), 33
- post.analysis.multisite.ggplot
 - (*interactive.plotting.sda*), 33
- postana.bias.plotting.sda
 - (*interactive.plotting.sda*), 33

postana.timeser.plotting.sda
 (*interactive.plotting.sda*),
 33

Prep_OBS_SDA, 45

prepare_train_dat, 46

qsub_sda, 46

qsub_sda_batch, 48

Remote_Sync_launcher, 48

rescaling_stateVars, 49

rwtmnorm, 49

sample.parameters, 50

sample_met, 50

sampler_toggle, 51

sda.bias.correction, 51

sda.enkf, 52

sda.enkf.multisite, 53

sda.enkf.original, 55

sda.enkf_local, 56

sda.qsub.job.submission, 58

sda_assemble, 59

SDA_control, 59

SDA_downscale, 60

SDA_downscale_hrly, 61

SDA_downscale_metrics, 62

SDA_downscale_preprocess, 63

sda_matchparam, 64

SDA_OBS_Assembler, 64

SDA_remote_launcher, 65

SDA_timeseries_plot
 (*interactive.plotting.sda*),
 33

sda_weights_site, 66

simple.local, 67

stack_covariates_2_df, 67

stack_covariates_2_geotiff, 68

subset_ensemble, 69

tobit.model, 70

tobit2space.model, 71

tobit_model_censored, 70

update_q, 71

y_star_create (*load_nimble*), 37