Package: PEcAn.assim.batch (via r-universe)

September 18, 2024

Type Package

Title PEcAn Functions Used for Ecological Forecasts and Reanalysis

Version 1.8.0.9000

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.

VignetteBuilder knitr

Imports abind, BayesianTools, coda (>= 0.18), MASS, methods, mlegp,
ellipse, graphics, grDevices, IDPmisc, lubridate (>= 1.6.0),
ncdf4 (>= 1.15), parallel, PEcAn.benchmark, PEcAn.DB,
PEcAn.emulator, PEcAn.logger, PEcAn.MA, PEcAn.remote,
PEcAn.settings, PEcAn.uncertainty, PEcAn.utils, PEcAn.workflow,
rjags, stats, prodlim, MCMCpack, TruncatedNormal (>= 2.2),
utils, XML, lqmm, mvtnorm

Suggests knitr (>= 1.42), rmarkdown (>= 2.19), testthat (>= 1.0.2)

License BSD_3_clause + file LICENSE

Copyright Authors

LazyLoad yes

LazyData FALSE

Encoding UTF-8

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

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

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

RemoteRef HEAD

RemoteSha f22a7c4bbc532e4551f7bc9624cef649da317ac1

Contents

`assim.batch` *Run Batch PDA*

Description

Run Batch PDA

Usage

```
assim.batch(settings)
```

Arguments

`settings`	a PEcAn settings list

Value

Updated settings list

Author(s)

Ryan Kelly

autoburnin *Automatically calculate and apply burnin value*

Description

Automatically calculate and apply burnin value

Usage

```
autoburnin(jags_out, return.burnin = FALSE, ...)
```

Arguments

`jags_out`	JAGS output
`return.burnin`	Logical. If `TRUE`, return burnin value in addition to samples (as list). Default = FALSE.
`...`	Additional arguments for `getBurnin`, `gelman_diag_mw`, and `gelman.diag`.

Author(s)

Michael Dietze, Alexey Shiklomanov

Examples

```
    z1 <- coda::mcmc(c(rnorm(2500, 5), rnorm(2500, 0)))
    z2 <- coda::mcmc(c(rnorm(2500, -5), rnorm(2500, 0)))
    z <- coda::mcmc.list(z1, z2)
    z_burned <- autoburnin(z)
```

bounded *bounded*

Description

bounded

Usage

```
bounded(xnew, rng)
```

Arguments

`xnew`	new x coordinate
`rng`	range

`calculate.prior` *calculate.prior*

Description

calculate.prior

Usage

```
calculate.prior(samples, priors)
```

Arguments

`samples`	Matrix of MCMC samples
`priors`	prior list

`correlationPlot` *Flexible function to create correlation density plots*

Description

numeric matrix or data.frame

Usage

```
correlationPlot(
  mat,
  density = "smooth",
  thin = "auto",
  method = "pearson",
  whichParameters = NULL
)
```

Arguments

`mat`	matrix or data frame of variables
`density`	type of plot to do
`thin`	thinning of the matrix to make things faster. Default is to thin to 5000
`method`	method for calculating correlations
`whichParameters`	all params or some

Author(s)

Florian Hartig

References

The code for the correlation density plot originates from Hartig, F.; Dislich, C.; Wiegand, T. & Huth, A. (2014) Technical Note: Approximate Bayesian parameterization of a process-based tropical forest model. Biogeosciences, 11, 1261-1272.

`ddist` *ddist*

Description

ddist

Usage

```
ddist(x, prior)
```

Arguments

`x`	vector of values (e.g. observations) to be evaluated by the specified probability density function
`prior`	data.frame specifying a prior probability distribution in terms of the distribution name (distn) and first and second parameters (parama, paramb)

`gelman_diag_gelmanPlot` *Calculate Gelman Diagnostic using coda::gelman.plot*

Description

Calculates Gelman diagnostic cumulatively. This is a much more conservative approach than the moving-window method.

Usage

```
gelman_diag_gelmanPlot(x, ...)
```

Arguments

`x`	MCMC samples
`...`	additional arguments

Author(s)

Alexey Shiklomanov

`gelman_diag_mw` *Calculate Gelman diagnostic on moving window*

Description

Calculate Gelman diagnostic on moving window

Usage

```
gelman_diag_mw(
  x,
  width_fraction = 0.1,
  width = ceiling(coda::niter(x) * width_fraction),
  njump = 50,
  include.mpsrf = TRUE,
  ...
)
```

Arguments

`x`	MCMC samples, of class `mcmc` or `mcmc.list`
`width_fraction`	Fractional width of moving window. Default=0.1.
`width`	Width of moving window. Default is niter(x)*width_fraction
`njump`	Number of windows to calculate over
`include.mpsrf`	Whether to calculate multivariate PSRF and include in output (default = FALSE).
`...`	additional arguments

Value

Gelman Diagnostic 3D array. First dim – mean (1) and 95% confidence (2). Second dim – iteration

Author(s)

Alexey Shiklomanov

`generate_hierpost` *Helper function that generates the hierarchical posteriors*

Description

Helper function that generates the hierarchical posteriors

Usage

```
generate_hierpost(mcmc.out, prior.fn.all, prior.ind.all)
```

Arguments

`mcmc.out`	hierarchical MCMC outputs
`prior.fn.all`	list of all prior functions
`prior.ind.all`	indices of the targeted params

Value

hierarchical MCMC outputs in original parameter space

Author(s)

Istem Fer

`getBurnin` *Calculate burnin value*

Description

Automatically detect burnin based on one of several methods.

Usage

```
getBurnin(
  jags_out,
  threshold = 1.1,
  use.confidence = TRUE,
  method = "moving.window",
  plotfile = "/dev/null",
  ...
)
```

Arguments

`jags_out`	List of MCMC sample matrices or `mcmc.list` object
`threshold`	Maximum value of Gelman diagnostic
`use.confidence`	Logical. If TRUE (default), use 95% confidence interval for Gelman Diagnostic. If FALSE, use the point estimate.
`method`	Character string indicating method. Options are "moving.window" (default) or "gelman.plot".
`plotfile`	path
`...`	Other parameters to methods

Details

See "gelman_diag_mw" and "gelman_diag_gelmanPlot"

Author(s)

Alexey Shiklomanov, Michael Dietze

Examples

```
    z1 <- coda::mcmc(c(rnorm(2500, 5), rnorm(2500, 0)))
    z2 <- coda::mcmc(c(rnorm(2500, -5), rnorm(2500, 0)))
    z <- coda::mcmc.list(z1, z2)
    burnin <- getBurnin(z, threshold = 1.05)
```

`get_ss` *get_ss*

Description

get_ss

Usage

```
get_ss(gp, xnew, pos.check)
```

Arguments

`gp`	Gaussian Process
`xnew`	new x coordinate
`pos.check`	check if value needs to be positive (if TRUE, returns -Inf when GP is negative)

get_y *get_y*

Description

get_y

Usage

```
get_y(SSnew, xnew, llik.fn, priors, llik.par)
```

Arguments

`SSnew`	new summary statistic
`xnew`	new x coordinate
`llik.fn`	list that contains likelihood functions
`priors`	prior list
`llik.par`	parameters to be passed llik functions

gpeval *gpeval*

Description

Calculates the probability of a set of parameter values, given by xnew

Usage

```
gpeval(xnew, k, mu, tau, psi, x, rng, splinefuns)
```

Arguments

`xnew`	new x coordinate
`k`	Specific absorption coefficient (400 - 2500nm)
`mu`	The mean parameter of the distribution; NOTE this is not equal to the mean
`tau`	spatial var
`psi`	spatial corr
`x`	Name of variable to plot on X axis
`rng`	range
`splinefuns`	spline functions

Author(s)

Michael Dietze

hier.mcmc *Hierarchical MCMC using emulator*

Description

Hierarchical MCMC

Usage

```
hier.mcmc(
  settings,
  gp.stack,
  nstack = NULL,
  nmcmc,
  rng_orig,
  jmp0,
  mu_site_init,
  nparam,
  nsites,
  llik.fn,
  prior.fn.all,
  prior.ind.all
)
```

Arguments

`settings`	a pecan settings list
`gp.stack`	list of GPs
`nstack`	list of number of observations, currently not used
`nmcmc`	number of MCMC iterations
`rng_orig`	range of knots
`jmp0`	initial jump vars
`mu_site_init`	initial parameter values per site
`nparam`	number of parameters
`nsites`	number of sites
`llik.fn`	list of likelihood functions
`prior.fn.all`	list of prior functions
`prior.ind.all`	indices of targeted parameters

Author(s)

Istem Fer

`is.accepted` *is.accepted*

Description

is.accepted

Usage

```
is.accepted(ycurr, ynew, format = "lin")
```

Arguments

`ycurr`	current value on y axis
`ynew`	new y coordinate
`format`	lin = lnlike fcn, log = log(lnlike)

`load.pda.data` *Load Ameriflux L2 Data From NetCDF*

Description

Load Ameriflux L2 Data From NetCDF

Load Dataset for Paramater Data Assimilation

Usage

```
load.L2Ameriflux.cf(file.in)

load.pda.data(settings, bety, external.formats = NULL)
```

Arguments

`file.in`	= the netcdf file of L2 data
`settings`	= PEcAn settings list
`bety`	database connection object
`external.formats`	formats list

Value

A data frame of all variables in the netcdf

A list containg the loaded input data, plus metadata

Author(s)

Ryan Kelly

Ryan Kelly, Istem Fer

`load_pda_history` *Helper function that loads history from previous PDA run, but returns only requested objects*

Description

Helper function that loads history from previous PDA run, but returns only requested objects

Usage

```
load_pda_history(workdir, ensemble.id, objects)
```

Arguments

`workdir`	path of working dir e.g. '/fs/data2/output/PEcAn_***'
`ensemble.id`	ensemble id of a previous PDA run, from which the objects will be retrieved
`objects`	object names that are common to all multi PDA runs, e.g. llik.fn, prior.list etc.

Value

a list of objects that will be used in joint and hierarchical PDA

Author(s)

Istem Fer

`makeMCMCList` *Make MCMC list from samples list*

Description

Make MCMC list from samples list

Usage

```
makeMCMCList(samps)
```

Arguments

`samps`	samples list (output from invert.custom)

mcmc.GP *mcmc.GP*

Description

Function to sample from a GP model that is assumed to be a -lnLikelihood surface with flat priors and bounded region

Usage

```
mcmc.GP(
  gp,
  x0,
  nmcmc,
  rng,
  format = "lin",
  mix = "joint",
  splinefuns = NULL,
  jmp0 = 0.35 * (rng[, 2] - rng[, 1]),
  ar.target = 0.5,
  priors = NA,
  settings,
  run.block = TRUE,
  n.of.obs,
  llik.fn,
  hyper.pars,
  resume.list = NULL
)
```

Arguments

`gp`	Gaussian Process
`x0`	initial values
`nmcmc`	number of iterations
`rng`	range of knots
`format`	lin = lnlike fcn, log = log(lnlike)
`mix`	each = jump each dim. independently, joint = jump all at once
`splinefuns`	spline functions, not used
`jmp0`	initial jump variances
`ar.target`	acceptance rate target
`priors`	prior list
`settings`	PEcAn settings list
`run.block`	is this a new run or making the previous chain longer

`n.of.obs`	number of observations
`llik.fn`	list that contains likelihood functions
`hyper.pars`	hyper parameters
`resume.list`	list of needed info if we are running the chain longer

Author(s)

Michael Dietze

`minimize.GP` *minimize.GP*

Description

minimize.GP

Usage

```
minimize.GP(gp, rng, x0, splinefuns = NULL)
```

Arguments

`gp`	Gaussian Process
`rng`	range
`x0`	initial values
`splinefuns`	spline functions

Author(s)

Michael Dietze

`pda.adjust.jumps` *Adjust PDA MCMC jump size*

Description

Adjust PDA MCMC jump size

Usage

```
pda.adjust.jumps(settings, jmp.list, accept.rate, pnames = NULL)
```

Arguments

`settings`	a PEcAn settings list
`jmp.list`	list of jump variances
`accept.rate`	acceptance rate
`pnames`	parameter names

Value

A PEcAn settings list updated to reflect adjusted jump distributions

Author(s)

Ryan Kelly

`pda.adjust.jumps.bs` *Adjust PDA block MCMC jump size*

Description

Adjust PDA block MCMC jump size

Usage

```
pda.adjust.jumps.bs(settings, jcov, accept.count, params.recent)
```

Arguments

`settings`	a PEcAn settings list
`jcov`	jump covariance matrix
`accept.count`	aceeptance count
`params.recent`	parameters accepted since previous adjustment

Value

A PEcAn settings list updated to reflect adjusted jump distributions

Author(s)

Ryan Kelly

`pda.autocorr.calc` *autocorrelation correction*

Description

autocorrelation correction

Usage

```
pda.autocorr.calc(input, model = "heteroskedastic.laplacian")
```

Arguments

`input`	list that contains time-series data vector and parameters for heteroskedastic.laplacian
`model`	data model type, for flux data heteroskedastic laplacian, normal is an example

Value

rho AR(1)

Author(s)

Istem Fer

`pda.bayesian.tools` *Paramater Data Assimilation using BayesianTools*

Description

Paramater Data Assimilation using BayesianTools R Package

Usage

```
pda.bayesian.tools(
  settings,
  external.data = NULL,
  external.priors = NULL,
  external.formats = NULL,
  ensemble.id = NULL,
  params.id = NULL,
  param.names = NULL,
  prior.id = NULL,
  chain = NULL,
  iter = NULL,
  adapt = NULL,
  adj.min = NULL,
  ar.target = NULL,
  jvar = NULL,
  remote = FALSE,
  ...
)
```

Arguments

`settings`	= a pecan settings list
`external.data`	list of external inputs
`external.priors`	list of external priors
`external.formats`	bety formats used when function is used without a DB connection, e.g. remote
`ensemble.id`	ensemble IDs
`params.id`	id of pars
`param.names`	names of pars
`prior.id`	ids of priors
`chain`	how many chains
`iter`	how many iterations
`adapt`	adaptation intervals
`adj.min`	to be used in adjustment
`ar.target`	acceptance rate target
`jvar`	jump variance
`remote`	logical, if TRUE no DB connection is established
`...`	additional arguments

Value

settings

Author(s)

Istem Fer

`pda.calc.error` *Calculate sufficient statistics*

Description

Calculate sufficient statistics

Usage

```
pda.calc.error(settings, con, model_out, run.id, inputs, bias.terms)
```

Arguments

`settings`	list
`con`	DB connection
`model_out`	list
`run.id`	run ID
`inputs`	list
`bias.terms`	matrix

Value

pda.errors

Author(s)

Istem Fer

`pda.calc.llik` *Calculate Likelihoods for PDA*

Description

Calculate Likelihoods for PDA

Usage

```
pda.calc.llik(pda.errors, llik.fn, llik.par)
```

Arguments

`pda.errors`	calculated errors
`llik.fn`	list of likelihood fcns
`llik.par`	parameters to be passed llik functions

Value

Total log likelihood (i.e., sum of log likelihoods for each dataset)

Author(s)

Ryan Kelly, Istem Fer

`pda.calc.llik.par` *pda.calc.llik.par*

Description

Calculate likelihood parameters

Usage

```
pda.calc.llik.par(settings, n, error.stats, hyper.pars)
```

Arguments

`settings`	list
`n`	named vector, sample sizes of inputs
`error.stats`	list, Sufficient Statistics
`hyper.pars`	list, hyperparameters

Author(s)

Istem Fer

`pda.create.btprior` *Create priors for BayesianTools*

Description

Helper function for creating log-priors compatible with BayesianTools package

Usage

```
pda.create.btprior(prior.sel)
```

Arguments

`prior.sel`	`data.frame` containing prior distributions of the selected parameters

Details

`prior.sel` must contain the following columns:

- `distn` – String describing a distribution; e.g. `norm` for `dnorm`, `rnorm`, etc.
- `parama`, `paramb` – First and second parameters, respectively, of the corresponding distribution

Optionally, `prior.sel` may also contain the following columns:

- `param_name` – Parameter name, which will be carried through to the prior object and sampler
- `lower`, `upper` – Lower and upper bounds, respectively. These can be leveraged by the BayesianTools samplers.
- `best` – Best guess for a parameter estimate. BayesianTools can also use this, though I'm not sure how...

Value

out Prior class object for BayesianTools package

Author(s)

Istem Fer, Alexey Shiklomanov

`pda.create.ensemble` *Create ensemble record for PDA ensemble*

Description

Create PDA Ensemble

Usage

```
pda.create.ensemble(settings, con, workflow.id)
```

Arguments

`settings`	a PEcAn settings list
`con`	DB connection
`workflow.id`	workflow ID

Value

Ensemble ID of the created ensemble

Author(s)

Ryan Kelly

`pda.define.llik.fn` *Define PDA Likelihood Functions*

Description

Define PDA Likelihood Functions

Usage

```
pda.define.llik.fn(settings)
```

Arguments

`settings`	PEcAn settings list

Value

List of likelihood functions, one for each dataset to be assimilated against.

Author(s)

Ryan Kelly, Istem Fer

`pda.define.prior.fn` *Define PDA Prior Functions*

Description

Define PDA Prior Functions

Usage

```
pda.define.prior.fn(prior)
```

Arguments

`prior`	prior dataframe

Value

List of prior functions containing dprior, rprior, qprior, dmvprior, rmvprior. Each of these is a list with one distribution function per parameter.

Author(s)

Ryan Kelly

pda.emulator *Paramater Data Assimilation using emulator*

Description

Paramater Data Assimilation using emulator

Usage

```
pda.emulator(
  settings,
  external.data = NULL,
  external.priors = NULL,
  external.knots = NULL,
  external.formats = NULL,
  ensemble.id = NULL,
  params.id = NULL,
  param.names = NULL,
  prior.id = NULL,
  chain = NULL,
  iter = NULL,
  adapt = NULL,
  adj.min = NULL,
  ar.target = NULL,
  jvar = NULL,
  n.knot = NULL,
  individual = TRUE,
  remote = FALSE
)
```

Arguments

`settings`	a pecan settings list
`external.data`	list of external inputs
`external.priors`	list of external priors
`external.knots`	list of external knots
`external.formats`	bety formats used when function is used without a DB connection, e.g. remote
`ensemble.id`	ensemble IDs
`params.id`	id of pars
`param.names`	names of pars
`prior.id`	ids of priors
`chain`	how many chains

`iter`	how many iterations
`adapt`	adaptation intervals
`adj.min`	to be used in adjustment
`ar.target`	acceptance rate target
`jvar`	jump variance
`n.knot`	number of knots requested
`individual`	logical, if TRUE it becomes a site-level PDA
`remote`	logical, if TRUE runs are submitted to remote and objects prepared accordingly

Value

nothing. Diagnostic plots, MCMC samples, and posterior distributions are saved as files and db records.

Author(s)

Mike Dietze

Ryan Kelly, Istem Fer

`pda.emulator.ms` *Paramater Data Assimilation using emulator on multiple sites in three modes: local, global, hierarchical First draft, not complete yet*

Description

Paramater Data Assimilation using emulator on multiple sites in three modes: local, global, hierarchical First draft, not complete yet

Usage

```
pda.emulator.ms(multi.settings)
```

Arguments

`multi.settings`	= a pecan multi-settings list

Value

settings

Author(s)

Istem Fer

`pda.generate.externals`
This is a helper function for preparing PDA external objects, but it doesn't cover all the cases yet, use it with care You can use this function just to generate either one of the external. PDA objects, but note that some args cannot be blank depending on what you aim to generate*

Description

This is a helper function for preparing PDA external objects, but it doesn't cover all the cases yet, use it with care You can use this function just to generate either one of the external.* PDA objects, but note that some args cannot be blank depending on what you aim to generate

Usage

```
pda.generate.externals(
  external.data = FALSE,
  obs = NULL,
  varn = NULL,
  varid = NULL,
  n_eff = NULL,
  align_method = "match_timestep",
  par = NULL,
  model_data_diag = FALSE,
  model.out = NULL,
  start_date = NULL,
  end_date = NULL,
  external.formats = FALSE,
  external.priors = FALSE,
  prior.list = NULL,
  external.knots = FALSE,
  knots.list = NULL,
  ind.list = NULL,
  nknots = NULL
)
```

Arguments

`external.data`	boolean, if TRUE function will generate external.data for PDA, then you need to pass varn and obs too, as well as align_method if different than "match_timestep"
`obs`	your data as a(n ordered) list where each sublist corresponds to a data frame of your constraining variable with two columns, variable name - posix IMPORTANT: your obs must be in the same units as PEcAn standards already, this function doesn't do unit conversions! IMPORTANT: your obs must be ready to compare with model outputs in general, e.g. if you're passing flux data it

	should already be ustar filtered e.g. obs[[1]] NEE posix 4.590273e-09 2017-01-01 00:00:00 NA 2017-01-01 00:30:00 NA 2017-01-01 01:00:00 NA 2017-01-01 01:30:00 NA 2017-01-01 02:00:00 4.575248e-09 2017-01-01 02:30:00 if you have more than variable make sure the order you pass the data is the same as varn. E.g. for varn=c("NEE", "Qle"), external.data should be obs[[1]] NEE posix NA 2018-05-09 NA 2018-05-10 NA 2018-05-11 NA 2018-05-12 ... ... obs[[2]] Qle posix NA 2018-05-09 NA 2018-05-10 NA 2018-05-11 NA 2018-05-12 ... ...
`varn`	a vector of PEcAn standard variable name(s) to read from model outputs, e.g. c("NEE", "Qle")
`varid`	a vector of BETY variable id(s) of your constraints, e.g. for varn = c("NEE", "Qle"), varid = c(297, 298)
`n_eff`	effective sample size of constraints, PDA functions estimates it for NEE and LE, and uses it in the heteroskedastic Laplacian only, if you already know it passing it now will save you some time
`align_method`	one of the benchmark::align_data align_method options "match_timestep" or "mean_over_larger_timestep", defaults to "match_timestep"
`par`	list with vector sublists of likelihood parameters of heteroskedastic laplacian for flux data, function calculates it if NULL for NEE, FC, and Qle. Leave empty for other variables e.g. AMF.params <- PEcAn.uncertainty::flux.uncertainty(...fill in...) par <- list(c(AMF.params$intercept, AMF.params$slopeP, AMF.params$slopeN))
`model_data_diag`	optional for diagnostics, if you want to check whether your model and data will be aligned in PDA properly you can return a dataframe as well as plot a quick & dirty timerseries graph
`model.out`	an example model output folder to align your data with model, e.g. "/data/workflows/PEcAn_1500000011
`start_date`	the start date of the model.out run, e.g. "2017-01-01"
`end_date`	the end date of the model.out run, e.g. "2018-12-31"
`external.formats`	boolean, if TRUE make sure to pass the varn argument
`external.priors`	boolean, if TRUE pass prior.list argument too
`prior.list`	a list of prior dataframes (one per pft, make sure the order is the same as it is in your <assim.batch> block), if you're using this make sure the targeted parameters are on the list e.g. prior.list <- list(data.frame(distn = c("norm", "beta"), parama = c(4, 1), paramb = c(7,2), n = rep(NA, 2), row.names = c("growth_resp_factor", "leaf_turnover_rate")), data.frame(distn = c("unif", "unif"), parama = c(10, 4), paramb = c(40,27), n = rep(NA, 2), row.names = c("psnTOpt", "half_saturation_PAR")))
`external.knots`	boolean, if TRUE pass prior.list, ind.list, nknots OR knots.list arguments too
`knots.list`	a list of dataframes (one per pft) where each row is a parameter vector, i.e. training points for the emulator. If not NULL these are used, otherwise knots will be generated using prior.list, ind.list and nknots.
`ind.list`	a named list of vectors (one per pft), where each vector indicates the indices of the parameters on the prior.list targeted in the PDA e.g. ind.list <- list(temperate.deciduous = c(2), temperate.conifer = c(1,2))
`nknots`	number of knots you want to train the emulator on

Examples

```
## Not run:
pda.externals <-  pda.generate.externals(external.data   = TRUE, obs = obs,
varn = "NEE", varid = 297, n_eff = 106.9386,
external.formats = TRUE, model_data_diag = TRUE,
model.out = "/tmp/out/outdir",
start_date = "2017-01-01", end_date = "2018-12-31")

## End(Not run)
```

`pda.generate.knots` *Generate Parameter Knots for PDA Emulator*

Description

Generate Parameter Knots for PDA Emulator

Usage

```
pda.generate.knots(
  n.knot,
  sf,
  probs.sf,
  n.param.all,
  prior.ind,
  prior.fn,
  pname
)
```

Arguments

`n.knot`	number of knots
`sf`	scaling factor
`probs.sf`	values for sf
`n.param.all`	number of all params
`prior.ind`	indices of targeted parameters in the prior dataframe
`prior.fn`	list of prior functions
`pname`	name of parameters

Value

A list of probabilities and parameter values, with one row for each knot in the emulator.

Author(s)

Ryan Kelly, Istem Fer

`pda.generate.sf` *Generate scaling factor knots for PDA Emulator*

Description

Generate scaling factor knots for PDA Emulator

Usage

```
pda.generate.sf(n.knot, sf, prior.list)
```

Arguments

`n.knot`	number of knots
`sf`	scaling factor
`prior.list`	list of prior dataframes

Author(s)

Istem Fer

`pda.get.model.output` *Get Model Output for PDA*

Description

Get Model Output for PDA

Usage

```
pda.get.model.output(settings, run.id, bety, inputs, external.formats = NULL)
```

Arguments

`settings`	PEcAn settings list
`run.id`	run ID
`bety`	database connection
`inputs`	inputs list
`external.formats`	format list

Value

A list containing model outputs extracted to correspond to each observational dataset being used for PDA.

Author(s)

Ryan Kelly, Istem Fer

`pda.init.params` *Initialise Parameter Matrix for PDA*

Description

Initialise Parameter Matrix for PDA

Usage

```
pda.init.params(settings, chain, pname, n.param.all)
```

Arguments

`settings`	a PEcAn settings list
`chain`	number of chain
`pname`	parameter name
`n.param.all`	number of all params

Value

A list containing 'start' and 'finish' counters for MCMC, as well as the params table, which is an empty matrix concatenated to any param samples from a previous PDA run, if provided.

Author(s)

Ryan Kelly

`pda.init.run` *Initialise Model Runs for PDA*

Description

Initialise Model Runs for PDA

Usage

```
pda.init.run(
  settings,
  con,
  my.write.config,
  workflow.id,
  params,
  n = ifelse(is.null(dim(params)), 1, nrow(params)),
  run.names = paste("run", 1:n, sep = ".")
)
```

Arguments

`settings`	a PEcAn settings list
`con`	DB connection
`my.write.config`	model write config fcn name
`workflow.id`	workflow ID
`params`	parameters of the run
`n`	number of runs
`run.names`	names of runs

Value

Vector of run IDs for all model runs that were set up (including write.configs)

Author(s)

Ryan Kelly

`pda.load.priors` *Load Priors for Paramater Data Assimilation*

Description

Load Priors for Paramater Data Assimilation

Usage

```
pda.load.priors(settings, con, extension.check = FALSE)
```

Arguments

`settings`	a PEcAn settings list
`con`	database connection
`extension.check`	check if this is another round or longer run

Value

A previously-generated posterior distribution, to be used as the prior for PDA.

Author(s)

Ryan Kelly, Istem Fer

pda.mcmc *Paramater Data Assimilation using MCMC*

Description

Paramater Data Assimilation using MCMC

Usage

```
pda.mcmc(
  settings,
  params.id = NULL,
  param.names = NULL,
  prior.id = NULL,
  chain = NULL,
  iter = NULL,
  adapt = NULL,
  adj.min = NULL,
  ar.target = NULL,
  jvar = NULL,
  n.knot = NULL
)
```

Arguments

`settings`	= a pecan settings list
`params.id`	id of pars
`param.names`	names of pars
`prior.id`	ids of priors
`chain`	how many chains
`iter`	how many iterations
`adapt`	adaptation intervals
`adj.min`	to be used in adjustment
`ar.target`	acceptance rate target
`jvar`	jump variance
`n.knot`	number of knots requested

Details

Brute-force, only to be used on simple models

Value

nothing. Diagnostic plots, MCMC samples, and posterior distributions are saved as files and db records.

Author(s)

Mike Dietze

Ryan Kelly

---

pda.mcmc.bs *Paramater Data Assimilation using MCMC*

Description

Parameter Data Assimilation using MCMC with block sampling

Usage

```
pda.mcmc.bs(
  settings,
  params.id = NULL,
  param.names = NULL,
  prior.id = NULL,
  chain = NULL,
  iter = NULL,
  adapt = NULL,
  adj.min = NULL,
  ar.target = NULL,
  jvar = NULL,
  n.knot = NULL
)
```

Arguments

`settings`	= a pecan settings list
`params.id`	id of pars
`param.names`	names of pars
`prior.id`	ids of priors
`chain`	how many chains
`iter`	how many iterations
`adapt`	adaptation intervals
`adj.min`	to be used in adjustment
`ar.target`	acceptance rate target
`jvar`	jump variance
`n.knot`	number of knots requested

Details

Brute-force, only to be used on simple models

Value

nothing. Diagnostic plots, MCMC samples, and posterior distributions are saved as files and db records.

Author(s)

Mike Dietze

Ryan Kelly

`pda.mcmc.recover` *Clean up a failed PDA run*

Description

Clean up a failed PDA run

Usage

```
pda.mcmc.recover(
  settings,
  params.id = NULL,
  param.names = NULL,
  prior.id = NULL,
  chain = NULL,
  iter = NULL,
  adapt = NULL,
  adj.min = NULL,
  ar.target = NULL,
  jvar = NULL,
  n.knot = NULL,
  burnin = NULL
)
```

Arguments

`settings`	PEcAn param list
`params.id`	id of pars
`param.names`	names of pars
`prior.id`	ids of priors
`chain`	how many chains
`iter`	how many iterations
`adapt`	adaptation intervals
`adj.min`	to be used in adjustment
`ar.target`	acceptance rate target

`jvar`	jump variance
`n.knot`	number of knots requested
`burnin`	burnin

Value

An updated settings list

Author(s)

Ryan Kelly

`pda.neff.calc` *Calculate N_eff*

Description

Autocorelation correction and efficient sample size calculation on latent process

Usage

```
pda.neff.calc(inputs, recalculate = FALSE)
```

Arguments

`inputs`	list
`recalculate`	repeat neff calculation or not

Details

What we're trying to do is to calculate the autocorrelation of the latent state, after attempting to "remove" the observation error. The first step is thus to estimate the latent state using a simple 'process free' state-space model (e.g. random walk).

Value

inputs list, updated inputs with n_eff

Author(s)

Istem Fer

`pda.plot.params` *Plot PDA Parameter Diagnostics*

Description

Plot PDA Parameter Diagnostics

Usage

```
pda.plot.params(
  settings,
  mcmc.param.list,
  prior.ind,
  par.file.name = NULL,
  sffx
)
```

Arguments

`settings`	PEcAn settings list
`mcmc.param.list`	MCMC param list to be sorted
`prior.ind`	indices of the targeted parameters
`par.file.name`	output file name
`sffx`	suffix to the output file names

Value

Nothing. Plot is generated and saved to PDF.

Author(s)

Ryan Kelly, Istem Fer

`pda.postprocess` *Postprocessing for PDA Results*

Description

Postprocessing for PDA Results

Usage

```
pda.postprocess(
  settings,
  con,
  mcmc.param.list,
  pname,
  prior,
  prior.ind,
  sffx = NULL
)
```

Arguments

`settings`	PEcAn settings list
`con`	DB connection
`mcmc.param.list`	output of PDA MCMC
`pname`	parameter names
`prior`	prior list
`prior.ind`	indices of targeted parameters
`sffx`	suffix to the output files, e.g. "hierarchical"

Value

PEcAn settings list, updated with <params.id> pointing to the new params file.

Author(s)

Ryan Kelly, Istem Fer

`pda.settings` *Set PDA Settings*

Description

Set PDA Settings

Usage

```
pda.settings(
  settings,
  params.id = NULL,
  param.names = NULL,
  prior.id = NULL,
  chain = NULL,
  iter = NULL,
  adapt = NULL,
  adj.min = NULL,
  ar.target = NULL,
  jvar = NULL,
  n.knot = NULL,
  run.round = FALSE
)
```

Arguments

`settings`	a PEcAn settings list
`params.id`	id of pars
`param.names`	names of pars
`prior.id`	ids of priors
`chain`	how many chains
`iter`	how many iterations
`adapt`	adaptation intervals
`adj.min`	to be used in adjustment
`ar.target`	acceptance rate target
`jvar`	jump variance
`n.knot`	number of knots requested
`run.round`	another round or not

Value

An updated settings list

Author(s)

Ryan Kelly, Istem Fer

`pda.settings.bt` *Apply settings for BayesianTools*

Description

Helper function for applying BayesianTools specific settings from PEcAn general settings

Usage

```
pda.settings.bt(settings)
```

Arguments

`settings`	PEcAn settings

Value

bt.settings list of BayesianTools::runMCMC settings

Author(s)

Istem Fer

`pda.sort.params` *Function to sort Hierarchical MCMC samples*

Description

Function to sort Hierarchical MCMC samples

Usage

```
pda.sort.params(
  mcmc.out,
  sub.sample = "mu_global_samp",
  ns = NULL,
  prior.all,
  prior.ind.all.ns,
  sf = NULL,
  n.param.orig,
  prior.list,
  prior.fn.all
)
```

Arguments

`mcmc.out`	MCMC samples
`sub.sample`	which subsample to return
`ns`	site number
`prior.all`	prior dataframe
`prior.ind.all.ns`	indices of targeted parameters on the prior.all dataframe
`sf`	scaling factor if used
`n.param.orig`	original indices of parameters on the prior.list
`prior.list`	list of prior dataframes
`prior.fn.all`	prior functions

prepare_pda_remote *helper function for submitting remote pda runs*

Description

helper function for submitting remote pda runs

Usage

```
prepare_pda_remote(settings, site = 1, multi_site_objects)
```

Arguments

settings	PEcAn settings list
site	site number (which site)
multi_site_objects	information needed for remote runs

return.bias *return.bias*

Description

return.bias

Usage

```
return.bias(
  settings,
  isbias,
  model.out,
  inputs,
  prior.list.bias,
  run.round = FALSE,
  pass2bias = NULL
)
```

Arguments

settings	settings list
isbias	bias variable index
model.out	model output list
inputs	inputs list

`prior.list.bias`	prior list, bias prior to be added
`run.round`	extension flag
`pass2bias`	if this is another round, this is re-sampled MCMC samples, will go with the rest of model params

Author(s)

Istem Fer

`return_hyperpars` *return_hyperpars*

Description

return_hyperpars

Usage

```
return_hyperpars(assim.settings, inputs)
```

Arguments

`assim.settings`	PEcAn settings list
`inputs`	inputs list

Author(s)

Istem Fer

`return_multi_site_objects` *This is a helper function partly uses pda.emulator code*

Description

This is a helper function partly uses pda.emulator code

Usage

```
return_multi_site_objects(multi.settings)
```

Arguments

`multi.settings`	PEcAn multi settings object

`runModule.assim.batch` *Run Batch module*

Description

Run Batch module

Usage

```
runModule.assim.batch(settings)
```

Arguments

`settings`	a PEcAn settings list

`sample_MCMC` *Helper function to sample from previous MCMC chain while proposing new knots*

Description

Helper function to sample from previous MCMC chain while proposing new knots

Usage

```
sample_MCMC(
  mcmc_path,
  n.param.orig,
  prior.ind.orig,
  n.post.knots,
  knots.params.temp,
  prior.list,
  prior.fn,
  sf,
  sf.samp
)
```

Arguments

`mcmc_path`	path to previous emulator mcmc samples object
`n.param.orig`	vector, number of parameters targeted in each (pft) sublist
`prior.ind.orig`	list, actual indices of parameters targeted in each (pft) sublist
`n.post.knots`	number of new samples requested
`knots.params.temp`	list of parameter samples proposed from the original PDA-prior

`prior.list`	PDA-prior list
`prior.fn`	list for parameter d/r/q/p functions
`sf`	SF parameter names
`sf.samp`	SF parameters MCMC samples

Author(s)

Istem Fer

`sync_pda_remote` *helper function for syncing remote pda runs this function resembles remote.copy.from but we don't want to sync everything back*

Description

helper function for syncing remote pda runs this function resembles remote.copy.from but we don't want to sync everything back

Usage

```
sync_pda_remote(multi.settings, ensembleidlist, register = FALSE)
```

Arguments

`multi.settings`	PEcAn multi settings
`ensembleidlist`	ensemble id list for remote runs
`register`	if register==TRUE, the last files returned will be registered to the DB, TO BE DONE

`write_sf_posterior` *Function to write posterior distributions of the scaling factors*

Description

Function to write posterior distributions of the scaling factors

Usage

```
write_sf_posterior(sf.samp.list, sf.prior, sf.samp.filename)
```

Arguments

`sf.samp.list`	scaling factor MCMC samples
`sf.prior`	scaling factor prior
`sf.samp.filename`	scaling factor posterior output file name

Index