Package: ldsr (via r-universe)

September 12, 2024

Type Package

Title Linear Dynamical System Reconstruction

Version 0.0.3

Description Streamflow (and climate) reconstruction using Linear Dynamical Systems. The advantage of this method is the additional state trajectory which can reveal more information about the catchment or climate system. For details of the method please refer to Nguyen and Galelli (2018) <doi:10.1002/2017WR022114>.

License GPL (>= 2.0)

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports data.table, dplR, foreach, MASS, Rcpp, stats

RoxygenNote 7.1.0

Roxygen list(markdown = TRUE)

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/ntthung/ldsr

BugReports https://github.com/ntthung/ldsr/issues

Suggests GA, knitr, rmarkdown, testthat, ggplot2, patchwork, doFuture, future

VignetteBuilder knitr

Repository https://critical-infrastructure-systems-lab.r-universe.dev

RemoteUrl https://github.com/critical-infrastructure-systems-lab/ldsr

RemoteRef HEAD

RemoteSha 593eeb790c754525ac793c3922b1e785d1780030

Contents

`calculate_metrics` *Reconstruction metrics*

Description

Calculate reconstruction metrics from the instrumental period

Usage

```
calculate_metrics(sim, obs, z, norm.fun = mean)
```

Arguments

sim	A vector of reconstruction output for instrumental period
obs	A vector of all observations
z	A vector of left out indices in cross validation
norm.fun	The function (unquoted name) used to calculate the normalizing constant. Default is mean(), but other functions such as sd() can also be used. THe function must take a vector as input and return a scalar as output, and must have an argument na.rm = TRUE.

Value

A named vector of performance metrics

Examples

```
calculate_metrics(rnorm(100), rnorm(100), z = 1:10)
calculate_metrics(rnorm(100), rnorm(100), z = 1:10, norm.fun = sd)
```

---

call_method *Call a reconstruction method*

Description

Call a reconstruction method subroutine according to the method required

Usage

```
call_method(
  y,
  u,
  v,
  method,
  init,
  num.restarts,
  return.init,
  ub,
  lb,
  num.islands,
  pop.per.island,
  niter,
  tol
)
```

Arguments

y	Catchment output, preprocessed from data
u	Input matrix for a single-model reconstruction, or a list of input matrices for an ensemble reconstruction.
v	Same as u.
method	By default this is "EM". There are experimental methods but you should not try.
init	A list, each element is a vector of initial values for the parameters. If NULL, will be created by make_init(). See make_init for details.
num.restarts	The number of initial conditions to start the EM search; ignored if init is provided.
return.init	If TRUE, the list of initial values (init) will be returned. This can be useful if you want to reproduce the model from this one set of initial values.
ub	Upper bounds, a vector whose length is the number of parameters
lb	Lower bounds
num.islands	Number of islands (if method is GA; experimental)
pop.per.island	Initial population per island (if method is GA; experimental)
niter	Maximum number of iterations, default 1000
tol	Tolerance for likelihood convergence, default 1e-5. Note that the log-likelihood is normalized by dividing by the number of observations.

Value

The results as produced by LDS_EM_restart() when the default method (EM) is called. Other methods are experimental.

---

corr *Pearson's correlation*

Description

Calculate the Pearson's correlation using the numerically stable formulation (see References). Internal function.

Usage

```
corr(x, y)
```

Arguments

x	First variable
y	Second variable

Value

Pearson's correlation

Reference John D. Cook's article at https

//www.johndcook.com/blog/2008/11/05/how-to-calculate-pearson-correlation-accurately/

`cvLDS` *Cross validate LDS model. This is a wrapper for LDS_reconstruction*

Description

Cross validate LDS model. This is a wrapper for LDS_reconstruction

Usage

```
cvLDS(
  Qa,
  u,
  v,
  start.year,
  method = "EM",
  transform = "log",
  num.restarts = 50,
  Z = make_Z(Qa$Qa),
  metric.space = "original",
  use.raw = FALSE,
  ub = NULL,
  lb = NULL,
  num.islands = 4,
  pop.per.island = 100,
  niter = 1000,
  tol = 1e-05,
  use.robust.mean = TRUE
)
```

Arguments

`Qa`	Observations: a data.frame of annual streamflow with at least two columns: year and Qa.
`u`	Input matrix for a single-model reconstruction, or a list of input matrices for an ensemble reconstruction.
`v`	Same as u.
`start.year`	Starting year of the climate proxies, i.e, the first year of the paleo period. `start.year + nrow(pc) - 1` will determine the last year of the study horizon, which must be greater than or equal to the last year in `Qa`.

method	By default this is "EM". There are experimental methods but you should not try.
transform	Flow transformation, either "log", "boxcox" or "none". Note that if the Box-Cox transform is used, the confidence interval after back-transformation is simply the back-transform of the trained onfidence interval; this is hackish and not entirely accurate.
num.restarts	The number of initial conditions to start the EM search; ignored if init is provided.
Z	A list of cross-validation folds. If NULL, will be created with make_Z() with default settings. Users are advised to use make_Z() to create the cross-validation folds beforehand. See make_Z for details.
metric.space	Either "transformed" or "original", the space to calculate the performance metrics.
use.raw	Whether performance metrics are calculated on the raw time series. Experimental; don't use.
ub	Upper bounds, a vector whose length is the number of parameters
lb	Lower bounds
num.islands	Number of islands (if method is GA; experimental)
pop.per.island	Initial population per island (if method is GA; experimental)
niter	Maximum number of iterations, default 1000
tol	Tolerance for likelihood convergence, default 1e-5. Note that the log-likelihood is normalized by dividing by the number of observations.
use.robust.mean	If TRUE (the default), use Tukey's biweight robust mean when calculating mean scores across the cross-validation run. If FALSE, use arithmetic mean. The Tukey's biweight robust mean function is dplR::tbrm().

Value

A list of cross validation results

- metrics.dist: distribution of performance metrics across all cross-validation runs; a matrix, one column for each metric, with column names.
- metrics: average performance metrics; a named vector.
- target: the (transformed) observations used for cross-validation; a data.table with two columns (year, y)
- Ycv: the predicted streamflow in each cross validation run; a matrix, one column for each cross-validation run
- Z: the cross-validation

Examples

```
# Make a shorter time horizon so that this example runs fast
u <- v <- t(NPpc[601:813])
# We run this example without parallelism.
foreach::registerDoSEQ()
```

```
cvLDS(NPannual, u, v, start.year = 1800, num.restarts = 2,
      Z = make_Z(NPannual$Qa, nRuns = 1))
# Please refer to the vignette for the full run with parallel options. It takes a minute or two.
```

---

cvPCR *Cross validation of PCR reconstruction.*

Description

Cross validation of PCR reconstruction.

Usage

```
cvPCR(
  Qa,
  pc,
  start.year,
  transform = "log",
  Z = NULL,
  metric.space = "original",
  use.robust.mean = TRUE
)
```

Arguments

Qa	Observations: a data.frame of annual streamflow with at least two columns: year and Qa.
pc	For a single model: a data.frame, one column for each principal component. For an ensemble reconstruction: a list, each element is a data.frame of principal components.
start.year	Starting year of the climate proxies, i.e, the first year of the paleo period. `start.year + nrow(pc) - 1` will determine the last year of the study horizon, which must be greater than or equal to the last year in `Qa`.
transform	Flow transformation, either "log", "boxcox" or "none". Note that if the Box-Cox transform is used, the confidence interval after back-transformation is simply the back-transform of the trained onfidence interval; this is hackish and not entirely accurate.
Z	A list of cross-validation folds. If `NULL`, will be created with `make_Z()` with default settings. Users are advised to use `make_Z()` to create the cross-validation folds beforehand. See make_Z for details.
metric.space	Either "transformed" or "original", the space to calculate the performance metrics.
use.robust.mean	If TRUE (the default), use Tukey's biweight robust mean when calculating mean scores across the cross-validation run. If FALSE, use arithmetic mean. The Tukey's biweight robust mean function is `dplR::tbrm()`.

Value

A list of cross validation results

- metrics.dist: distribution of performance metrics across all cross-validation runs; a matrix, one column for each metric, with column names.
- metrics: average performance metrics; a named vector.
- obs: the (transformed) observations, a data.table with two columns (year, y)
- Ycv: the predicted streamflow in each cross validation run; a matrix, one column for each cross-validation run
- Z: the cross-validation fold

Examples

```
cvPCR(NPannual, NPpc, start.year = 1200)
```

`exp_ci` *Exponential confidence interval*

Description

Get the confidence interval of Q = exp(Y) when Y is normal, i.e, Q is log-normal.

Usage

```
exp_ci(y, sigma)
```

Arguments

`y`	A vector of model estimates
`sigma`	The standard deviation of y as learned from a model

Value

A data.table with the updated confidence intervals

inv_boxcox *Inverse Box-Cox transform*

Description

Inverse Box-Cox transform

Usage

```
inv_boxcox(x, lambda)
```

Arguments

x	A numeric vector to be transformed
lambda	Lambda parameter

Value

The inverse Box-Cox

Examples

```
inv_boxcox(x = rnorm(10), lambda = 1)
inv_boxcox(x = rnorm(10), lambda = 0) # exp(x)
```

Kalman_smoother *Implement Kalman smoothing*

Description

Estimate the hidden state and expected log-likelihood given the observations, exogeneous input and system parameters. This is an internal function and should not be called directly.

Usage

```
Kalman_smoother(y, u, v, theta, stdlik = TRUE)
```

Arguments

y	Observation matrix (may need to be normalized and centered before hand) (q rows, T columns)
u	Input matrix for the state equation (m_u rows, T columns)
v	Input matrix for the output equation (m_v rows, T columns)
theta	A list of system parameters (A, B, C, D, Q, R)'
stdlik	Boolean, whether the likelihood is divided by the number of observations. Standardizing the likelihood this way may speed up convergence in the case of long time series.

Value

A list of fitted elements (X, Y, V, J, and lik)

Note

This code only works on one dimensional state and output at the moment. Therefore, transposing is skipped, and matrix inversion is treated as /, and log(det(Sigma)) is treated as log(Sigma).

KGE *Kling-Gupta Efficiency*

Description

Kling-Gupta Efficiency

Usage

```
KGE(yhat, y)
```

Arguments

yhat	Model outputs
y	Observations

Value

KGE

Examples

```
KGE(rnorm(100), rnorm(100))
```

LDS_BFGS *Learn LDS with L-BFGS-B*

Description

Warning This is an experimental feature. Use with care.

Usage

```
LDS_BFGS(y, u, v, ub, lb, num.restarts = 100, parallel = TRUE)
```

Arguments

y	Transformed and standardized streamflow
u	Input matrix for a single-model reconstruction, or a list of input matrices for an ensemble reconstruction.
v	Same as u.
ub	Upper bounds, a vector whose length is the number of parameters
lb	Lower bounds
num.restarts	The number of initial conditions to start the EM search; ignored if `init` is provided.
parallel	Logical, whether parallel computation is used

Value

A list of reconstruction results; see LDS_reconstruction

LDS_BFGS_with_update *Learn LDS with L-BFGS-B*

Description

Warning This is an experimental feature. Use with care.

Usage

```
LDS_BFGS_with_update(
  y,
  u,
  v,
  lambda = 1,
  ub,
  lb,
  num.restarts = 100,
  parallel = TRUE
)
```

Arguments

y	Transformed and standardized streamflow
u	Input matrix for a single-model reconstruction, or a list of input matrices for an ensemble reconstruction.
v	Same as u.
lambda	weight for penalty
ub	Upper bounds, a vector whose length is the number of parameters

lb	Lower bounds
num.restarts	The number of initial conditions to start the EM search; ignored if init is provided.
parallel	Logical, whether parallel computation is used

Value

A list of reconstruction results; see LDS_reconstruction

LDS_EM *Learn LDS model*

Description

Estimate the hidden state and model parameters given observations and exogenous inputs using the EM algorithm. This is the key backend routine of this package.

Usage

```
LDS_EM(y, u, v, theta0, niter = 1000L, tol = 1e-05)
```

Arguments

y	Observation matrix (may need to be normalized and centered before hand) (q rows, T columns)
u	Input matrix for the state equation (m_u rows, T columns)
v	Input matrix for the output equation (m_v rows, T columns)
theta0	A vector of initial values for the parameters
niter	Maximum number of iterations, default 1000
tol	Tolerance for likelihood convergence, default 1e-5. Note that the log-likelihood is normalized

Value

A list of model results

- theta: model parameters (A, B, C, D, Q, R, mu1, V1) resulted from Mstep
- fit: results of Estep
- liks : vector of loglikelihood over the iteration steps

Note

This code only works on one dimensional state and output at the moment. Therefore, transposing is skipped, and matrix inversion is treated as /, and log(det(Sigma)) is treated as log(Sigma).

LDS_EM_restart *Learn LDS model with multiple initial conditions*

Description

This is the backend computation for LDS_reconstruction. You should not use this directly.

Usage

```
LDS_EM_restart(y, u, v, init, niter = 1000, tol = 1e-05, return.init = TRUE)
```

Arguments

y	Observation matrix (may need to be normalized and centered before hand) (q rows, T columns)
u	Input matrix for the state equation (m_u rows, T columns)
v	Input matrix for the output equation (m_v rows, T columns)
init	A list of initial parameters for the EM search. See make_init.
niter	Maximum number of iterations, default 1000
tol	Tolerance for likelihood convergence, default 1e-5. Note that the log-likelihood is normalized by dividing by the number of observations.
return.init	Indicate whether the initial condition that results in the highest

Value

a list as produced by LDS_EM. If return.init is true, a vector of initial condition is included in the list as well.

LDS_GA *Learn a linear dynamical system using Genetic Algorithm.*

Description

Warning This is an experimental feature. Use with care.

Usage

```
LDS_GA(
  y,
  u,
  v,
  lambda = 1,
  ub,
  lb,
```

```
  num.islands = 4,
  pop.per.island = 100,
  niter = 1000,
  parallel = TRUE
)
```

Arguments

y	Transformed and standardized streamflow
u	Input matrix for a single-model reconstruction, or a list of input matrices for an ensemble reconstruction.
v	Same as u.
lambda	weight for penalty
ub	Upper bounds, a vector whose length is the number of parameters
lb	Lower bounds
num.islands	Number of islands (if method is GA; experimental)
pop.per.island	Initial population per island (if method is GA; experimental)
niter	Maximum number of iterations, default 1000
parallel	Logical, whether parallel computation is used

Value

A list of reconstruction results; see LDS_reconstruction

`LDS_reconstruction` *Learn LDS model.*

Description

The initial conditions can either be randomized (specifiled by num.restarts) or provided beforehand.

Usage

```
LDS_reconstruction(
  Qa,
  u,
  v,
  start.year,
  method = "EM",
  transform = "log",
  init = NULL,
  num.restarts = 50,
  return.init = FALSE,
  ub = NULL,
  lb = NULL,
```

```
  num.islands = 4,
  pop.per.island = 250,
  niter = 1000,
  tol = 1e-05,
  return.raw = FALSE
)
```

Arguments

Qa	Observations: a data.frame of annual streamflow with at least two columns: year and Qa.
u	Input matrix for a single-model reconstruction, or a list of input matrices for an ensemble reconstruction.
v	Same as u.
start.year	Starting year of the climate proxies, i.e, the first year of the paleo period. start.year + nrow(pc) - 1 will determine the last year of the study horizon, which must be greater than or equal to the last year in Qa.
method	By default this is "EM". There are experimental methods but you should not try.
transform	Flow transformation, either "log", "boxcox" or "none". Note that if the Box-Cox transform is used, the confidence interval after back-transformation is simply the back-transform of the trained onfidence interval; this is hackish and not entirely accurate.
init	A list, each element is a vector of initial values for the parameters. If NULL, will be created by make_init(). See make_init for details.
num.restarts	The number of initial conditions to start the EM search; ignored if init is provided.
return.init	If TRUE, the list of initial values (init) will be returned. This can be useful if you want to reproduce the model from this one set of initial values.
ub	Upper bounds, a vector whose length is the number of parameters
lb	Lower bounds
num.islands	Number of islands (if method is GA; experimental)
pop.per.island	Initial population per island (if method is GA; experimental)
niter	Maximum number of iterations, default 1000
tol	Tolerance for likelihood convergence, default 1e-5. Note that the log-likelihood is normalized by dividing by the number of observations.
return.raw	If TRUE, state and streamflow estimates without measurement updates will be returned.

Value

A list of the following elements

- rec: reconstruction results, a data.table with the following columns
  - year: calculated from Qa and the length of u

  - X: the estimated hidden state
  - Xl, Xu: lower and upper range for the 95\
  - Q: the reconstructed streamflow
  - Ql, Qu: lower and upper range for the 95\
- theta: model parameters
- lik: maximum likelihood
- init: the initial condition that resulted in the maximum likelihood (if `return.init = TRUE`).

Examples

```
# Make a shorter time horizon so that this example runs fast
u <- v <- t(NPpc[601:813])
# We run this example without parallelism
foreach::registerDoSEQ()
LDS_reconstruction(NPannual, u, v, start.year = 1800, num.restarts = 1)
# Please refer to the vignette for the full run with parallel options. It takes a second or two.
```

`LDS_rep` *Multiple LDS replicates*

Description

Generate multiple stochastic time series from an LDS model. This is a convenient wrapper for one_LDS_rep.

Usage

```
LDS_rep(
  theta,
  u = NULL,
  v = NULL,
  years,
  num.reps = 100,
  mu = 0,
  exp.trans = TRUE
)
```

Arguments

`theta`	A list of parameters: A, B, C, D, Q, R, x0, v0
`u`	Input matrix for the state equation (m_u rows, T columns)
`v`	Input matrix for the output equation (m_v rows, T columns)
`years`	The years of the study horizon
`num.reps`	The number of stochastic replicates#'
`mu`	Mean of the log-transformed streamflow process
`exp.trans`	Whether exponential transformation back to the streamflow space is required. If TRUE, both Y and Q are returned, otherwise only Y.

Value

Same as one_LDS_rep, but the data.table consists of multiple replicates.

Examples

```
LDS_rep(theta, t(NPpc), t(NPpc), 1200:2012, num.reps = 10, mu = mean(log(NPannual$Qa)))
```

`make_init` *Make initial values for parameters.*

Description

If init is a vector, make it a list of one element. If init is NULL, randomize it. In this case, the function will randomize the initial value by sampling uniformly within the range for each parameters (A in [0, 1], B in [-1, 1], C in [0, 1] and D in [-1, 1]).

Usage

```
make_init(p, q, num.restarts)
```

Arguments

`p`	Dimension of u
`q`	Dimension of v
`num.restarts`	Number of randomized initial values

Value

A list of initial conditions, each element is an object of class `theta`.

Examples

```
make_init(5, 5, 1)
make_init(5, 5, 2)
```

make_Z *Make cross-validation folds.*

Description

Make a list of cross-validation folds. Each element of the list is a vector of the cross-validation points for one cross-validation run.

Usage

```
make_Z(obs, nRuns = 30, frac = 0.1, contiguous = TRUE)
```

Arguments

obs	Vector of observations.
nRuns	Number of repetitions.
frac	Fraction of left-out points. For leave-one-out, use frac = 1, otherwise use any value less than 1. Default is 0.1 (leave-10%-out).
contiguous	Logical. If TRUE, the default, the left-out points are made in contiguous blocks; otherwise, they are scattered randomly.

Value

A list of cross-validation folds

Examples

```
Z <- make_Z(NPannual, nRuns = 30, frac = 0.25, contiguous = TRUE)
```

---

metrics_with_transform
Transform the estimates before calculating metrics

Description

If you already ran the cross-validation on transformed output and now wanted to calculate performance on the back-transformed one, or vice-versa, you don't have to rerun the whole cross-validation, but just need to transform or back-transform the cross-validation Ycv. This function helps you do that.

Usage

```
metrics_with_transform(cv, transform, lambda = NULL)
```

Arguments

cv	Cross-validation output as produced by cvLDS or cvPCR
transform	Either "log", "exp", "boxcox" or "inv_boxcox"
lambda	Lambda value used in Box-Cox or inverse Box-Cox

Value

A new cv object wit hthe new metrics

Examples

```
# Cross-validate with log-transform
cv <- cvPCR(NPannual, NPpc, start.year = 1200, transform = 'log', metric.space = 'transformed')
# Calculate metrics based on back-transformed values
m <- metrics_with_transform(cv, 'exp')
```

Mstep *Maximizing expected likelihood using analytical solution*

Description

Maximizing expected likelihood using analytical solution

Usage

```
Mstep(y, u, v, fit)
```

Arguments

y	Observation matrix (may need to be normalized and centered before hand) (q rows, T columns)
u	Input matrix for the state equation (m_u rows, T columns)
v	Input matrix for the output equation (m_v rows, T columns)
fit	result of Kalman_smoother

Value

An object of class theta: a list of

negLogLik *Calculates the negative log-likelihood*

Description

Calculates the negative log-likelihood

Usage

```
negLogLik(theta, u, v, y)
```

Arguments

theta	A vector of model parameters, to be converted to a list
u	Input matrix
v	Input matrix
y	Observations

Value

The negative log-likelihood

NPannual *Annual streamflow at Nakhon Phanom*

Description

A dataset contaning annual streamflow of the Mekong River measured at station Nakhon Phanom.

Usage

```
NPannual
```

Format

A data.table with two columns

year from 1960 to 2005

Qa mean annual streamflow, cubic meter per second

Source

Mekong River Commission

NPpc *Selected principal components*

Description

A dataset contaning the principal components of MADA surrounding NP

Usage

```
NPpc
```

Format

A data.table

nRMSE *Normalized root-mean-square error*

Description

RMSE is normalized by the normalization constant

Usage

```
nRMSE(yhat, y, normConst)
```

Arguments

yhat	Model outputs
y	Observations
normConst	The normalization constant

Value

normalized RMSE

Examples

```
x <- rnorm(100)
y <- rnorm(100)
nRMSE(x, y, sd(y))
```

NSE *Nash-Sutcliffe Efficiency*

Description

Nash-Sutcliffe Efficiency

Usage

```
NSE(yhat, y)
```

Arguments

yhat	Model outputs
y	Observations

Value

NSE

Examples

```
NSE(rnorm(100), rnorm(100))
```

one_lds_cv *One cross-validation run*

Description

Make one prediction for one cross-validation run. This is a subroutine that is called by cvLDS, without any checks. You should not need to use this directly.

Usage

```
one_lds_cv(
  z,
  instPeriod,
  mu,
  y,
  u,
  v,
  method = "EM",
  num.restarts = 20,
  ub = NULL,
  lb = NULL,
  num.islands = 4,
```

```
  pop.per.island = 100,
  niter = 1000,
  tol = 1e-06,
  use.raw = FALSE
)
```

Arguments

z	A vector of left-out points, indexed according to the intrumental period
instPeriod	indices of the instrumental period in the whole record
mu	Mean of the observations
y	Catchment output, preprocessed from data
u	Input matrix for a single-model reconstruction, or a list of input matrices for an ensemble reconstruction.
v	Same as u.
method	By default this is "EM". There are experimental methods but you should not try.
num.restarts	The number of initial conditions to start the EM search; ignored if `init` is provided.
ub	Upper bounds, a vector whose length is the number of parameters
lb	Lower bounds
num.islands	Number of islands (if method is GA; experimental)
pop.per.island	Initial population per island (if method is GA; experimental)
niter	Maximum number of iterations, default 1000
tol	Tolerance for likelihood convergence, default 1e-5. Note that the log-likelihood is normalized by dividing by the number of observations.
use.raw	Whether performance metrics are calculated on the raw time series. Experimental; don't use.

Value

A vector of prediction.

`one_LDS_rep` *One LDS replicate*

Description

Generate a single stochastic time series from an LDS model

Usage

```
one_LDS_rep(
  rep.num,
  theta,
  u = NULL,
  v = NULL,
  years,
  mu = 0,
  exp.trans = TRUE
)
```

Arguments

rep.num	The ID number of the replicate
theta	A list of parameters: A, B, C, D, Q, R, x0, v0
u	Input matrix for the state equation (m_u rows, T columns)
v	Input matrix for the output equation (m_v rows, T columns)
years	The years of the study horizon
mu	Mean of the log-transformed streamflow process
exp.trans	Whether exponential transformation back to the streamflow space is required. If TRUE, both Y and Q are returned, otherwise only Y.

Value

A data.table. The first column is the years of the study horizon, as supplied by year. Subsequent columns are simX, simY, and simQ which are the simulated catchment state (X), log-transformed and centralized flow (Y) and flow (Q). The last column is the replicate ID number.

Examples

```
# Learn theta
one_LDS_rep(1, theta, t(NPpc), t(NPpc), 1200:2012, mu = mean(log(NPannual$Qa)))
```

---

one_pcr_cv *One cross-validation run*

Description

Make one prediction for one cross-validation run. This is a subroutine to be called by other cross-validation functions.

Usage

```
one_pcr_cv(Xmat, y, z)
```

Arguments

Xmat	The matrix of training data in the instrumental period.
y	A vector of observations.
z	A vector of left-out points.

Value

A vector of prediction.

`PCR_ensemble_selection` *Select the best reconstruction*

Description

Select the best reconstruction from an ensemble. Experimental, API may change.

Usage

```
PCR_ensemble_selection(
  Qa,
  pc,
  start.year,
  transform = "log",
  Z = NULL,
  agg.type = c("best member", "best overall"),
  criterion = c("RE", "CE", "nRMSE", "KGE"),
  return.all.metrics = FALSE
)
```

Arguments

Qa	Observations: a data.frame of annual streamflow with at least two columns: year and Qa.
pc	For a single model: a data.frame, one column for each principal component. For an ensemble reconstruction: a list, each element is a data.frame of principal components.
start.year	Starting year of the climate proxies, i.e, the first year of the paleo period. `start.year + nrow(pc) - 1` will determine the last year of the study horizon, which must be greater than or equal to the last year in `Qa`.
transform	Flow transformation, either "log", "boxcox" or "none". Note that if the Box-Cox transform is used, the confidence interval after back-transformation is simply the back-transform of the trained onfidence interval; this is hackish and not entirely accurate.

`Z` A list of cross-validation folds. If `NULL`, will be created with `make_Z()` with default settings. Users are advised to use `make_Z()` to create the cross-validation folds beforehand. See make_Z for details.

`agg.type` Type of ensemble aggregate. There are 2 options: 'best member': the member with the best performance score is used; 'best overall': if the ensemble average is better than the best member, it will be used, otherwise the best member will be used.

`criterion` The performance criterion to be used.

`return.all.metrics` Logical, if TRUE, all members' performance scores (and the ensemble average's score, if `agg.type == 'best overall'`) are returned.

Value

A list of two elements:

- choice: The index of the selection. If the ensemble is selected, returns 0.
- cv: the cross-validation results of the choice, see cvPCR for details.
- all.metrics: all members' scores, and if `agg.type == 'best overall'`, the ensemble average's scores as well, in the last column.

Examples

```
PCR_ensemble_selection(NPannual, list(NPpc, NPpc[, 1:2]), start.year = 1200,
                       agg.type = 'best overall', criterion = 'KGE')
PCR_ensemble_selection(NPannual, list(NPpc, NPpc[, 1:2]), start.year = 1200,
                       agg.type = 'best overall', criterion = 'KGE')
```

PCR_reconstruction *Principal Component Regression Reconstruction*

Description

Reconstruction with principal component linear regression.

Usage

```
PCR_reconstruction(Qa, pc, start.year, transform = "log")
```

Arguments

`Qa` Observations: a data.frame of annual streamflow with at least two columns: year and Qa.

`pc` For a single model: a data.frame, one column for each principal component. For an ensemble reconstruction: a list, each element is a data.frame of principal components.

start.year	Starting year of the climate proxies, i.e, the first year of the paleo period. start.year + nrow(pc) - 1 will determine the last year of the study horizon, which must be greater than or equal to the last year in Qa.
transform	Flow transformation, either "log", "boxcox" or "none". Note that if the Box-Cox transform is used, the confidence interval after back-transformation is simply the back-transform of the trained onfidence interval; this is hackish and not entirely accurate.

Value

A list of reconstruction results, with the following elements:

For a single-model reconstruction::

- rec: reconstructed streamflow with 95% prediction interval; a data.table with four columns: year, Q, Ql (lower bound), and Qu (upper bound).
- coeffs: the regression coefficients.
- sigma: the residual standard deviation.

For an ensemble reconstruction::

- rec: the ensemble average reconstruction; a data.table with two columns: year and Q.
- ensemble: a list of ensemble members, each element is reconstructed from one element of pc and is itself a list of three elements: Q (a vector of reconstructed flow), coeffs and sigma. Note that for ensemble reconstruction, ldsr does not provide uncertainty estimates. It is up to the user to do so, for example, using ensemble spread.

Examples

```
PCR_reconstruction(NPannual, NPpc, start.year = 1200)
```

---

penalized_likelihood *Penalized likelihood objective function*

Description

Kalman_smoother returns X and likelihood. The penalized likelihood is the likelihood minus the sum-of-squares of the measurement update. This is used as the fitness function in genetic algorihm.

Usage

```
penalized_likelihood(y, u, v, theta.vec, lambda)
```

Arguments

y	Observation matrix (may need to be normalized and centered before hand) (q rows, T columns)
u	Input matrix for the state equation (m_u rows, T columns)
v	Input matrix for the output equation (m_v rows, T columns)
theta.vec	a vector of parameter elements (i.e, the vectorized version of theta in Kalman_smoother)
lambda	weight of the penalty

Value

The penalized likelihood (a real number)

`propagate` *State propagation*

Description

This function propagates the state trajectory based on the exogenous inputs only (without measurement update), and calculates the corresponding log-likelihood

Usage

```
propagate(theta, u, v, y, stdlik = TRUE)
```

Arguments

`theta`	A list of system parameters (A, B, C, D, Q, R)'
`u`	Input matrix for the state equation (m_u rows, T columns)
`v`	Input matrix for the output equation (m_v rows, T columns)
`y`	Observations
`stdlik`	Boolean, whether the likelihood is divided by the number of observations. Standardizing the likelihood this way may speed up convergence in the case of long time series.

Value

A list of predictions and log-likelihood (X, Y, V, lik)

Note

This code only works on one dimensional state and output at the moment. Therefore, transposing is skipped, and matrix inversion is treated as /, and log(det(Sigma)) is treated as log(Sigma).

RE *Reduction of Error*

Description

Reduction of Error

Usage

```
RE(yhat, y, yc_bar)
```

Arguments

yhat	Model outputs in the validation set
y	Observations in the validation set
yc_bar	Mean observations in the calibration set

Value

RE

Examples

```
x <- rnorm(100)
y <- rnorm(100)
yc_bar <- mean(x[1:50])
RE(x[51:100], y[51:100], yc_bar)
```

theta *LDS parameters*

Description

Theta values for Nakhon Phanom, precomputed to use in examples.

Usage

```
theta
```

Format

A list with elements A, B, C, D, Q, R, mu1, V1

`vec_to_list` *Converts theta from a vector (as used in GA) to list (as used in Kalman smoothing)*

Description

Converts theta from a vector (as used in GA) to list (as used in Kalman smoothing)

Usage

```
vec_to_list(theta.vec, d)
```

Arguments

`theta.vec`	a vector of parameter elements
`d`	dimention of inputs

Value

A theta object, see make_init

Index