Package ‘mbr’

October 13, 2022

Type Package
Title Mass Balance Reconstruction
Version 0.0.1
Author Hung Nguyen
Maintainer Hung Nguyen <ntthung@gmail.com>
Description Mass-balance-adjusted Regression algorithm for streamflow reconstruction at sub-
annual resolution (e.g., seasonal or monthly). The algorithm implements a penalty term to mini-
mimize the differences between the total sub-annual flows and the annual flow. The method is de-
License GPL (>= 2.0)
Encoding UTF-8
LazyData true
Depends R (>= 3.5)
Imports data.table, dplR, MASS, Matrix, Rfast, stats
RoxygenNote 7.1.1
URL https://github.com/ntthung/mbr
BugReports https://github.com/ntthung/mbr/issues
Suggests knitr, rmarkdown, testthat
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2021-02-16 09:10:02 UTC

R topics documented:

back_trans .......................................................... 2
calculate_metrics .................................................. 3
colScale .............................................................. 3
colUnscale ........................................................... 4
Transform the reconstructed values back to the flow space and convert to data.table

Usage

```r
back_trans(hat, years, mus, sigmas, log.trans, N, season.names)
```

Arguments

- `hat`: A vector of estimated flow in the transformed space.
- `years`: A vector of all years in the study period.
- `mus`: A vector of means, one for each target.
- `sigmas`: A vector of the standard deviations, one for each target.
- `log.trans`: A vector containing the indices of the columns to be log-transformed.
- `N`: The number of targets (number of seasons plus one for the annual reconstruction).
- `season.names`: A character vector containing the names of the seasons.

Value

A `data.table` with three columns: Q (the back-transformed streamflow), season, and year.
**calculate_metrics**

*Reconstruction metrics*

**Description**

Calculate reconstruction metrics from the instrumental period

**Usage**

```r
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**

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

---

**colScale**

*Scale columns of a matrix*

**Description**

Same as `base::scale()` but much faster.

**Usage**

```r
colScale(x, add_attr = TRUE)
```

**Arguments**

- `x` A matrix.
- `add_attr` If TRUE, the column means and standard deviations are returned as attributes. This is consistent with `base::scale()`.
Value

The scaled matrix.

Reference

This function was adopted from John Muschelli’s code on StackOverflow, but I changed the underlying functions to calculate mean and standard deviation from matrixStats to Rfast, which is much faster.

colUnscale

Unscale columns of a matrix

Description

Backtransform a matrix that was scaled before.

Usage

colUnscale(x, cm, csd)

Arguments

x
A matrix.
cm
A vector of column means
csd
A vector of column standard deviations

Value

The unscaled matrix

cv_mb

Cross-validation

Description

Cross-validation
Usage

cv.mb(
  instQ,
  pc.list,
  cv.folds,
  start.year,
  lambda = 1,
  log.trans = NULL,
  force.standardize = FALSE,
  return.type = c("fval", "metrics", "metric means", "Q")
)

Arguments

instQ Instrumental data, in the same order as pc.list. The "season" column must be a factor.

pc.list List of PC matrices

cv.folds A list containing the cross validation folds

start.year The first year of record

lambda The penalty weight

log.trans A vector containing indices of the targets to be log-transformed. If no transformation is needed, provide NULL.

force.standardize If TRUE, all observations are standardized. See Details.

return.type The type of results to be returned. Several types are possible to suit multiple use cases.

  fval Only the objective function value (penalized least squares) is returned; this is useful for the outer optimization for site selection.

  metrics all performance metrics are returned.

  metric means the Tukey’s biweight robust mean of each metric is returned.

  Q The predicted flow in each cross-validation run is returned. This is the most basic output, so that you can use it to calculate other metrics that are not provided by the package.

Value

A data.table containing cross-validation results (metrics, fval, or metric means) for each target.

Examples

cvFolds <- make_Z(1922:2003, nRuns = 50, frac = 0.25, contiguous = TRUE)
cv <- cv.mb(p1Seasonal, pc3seasons, cvFolds, 1750, log.trans = 1:3, return.type = 'metrics')
KGE  

**Kling-Gupta Efficiency**

**Description**

Kling-Gupta Efficiency

**Usage**

KGE(yhat, y)

**Arguments**

- **yhat**: Model outputs
- **y**: Observations

**Value**

KGE value

**Examples**

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

---

lsq_mb  

**Least square with mass balance penalty**

**Description**

Least square with mass balance penalty

**Usage**

lsq_mb(hat, obs, lambda, mus, sigmas, log.seasons, log.ann, N, sInd)

**Arguments**

- **hat**: A vector of estimated flow in the transformed space.
- **obs**: A vector of observed flow in the transformed space.
- **lambda**: Penalty weight.
- **mus**: A vector of means, one for each target.
- **sigmas**: A vector of the standard deviations, one for each target.
- **log.seasons**: A vector containing the indices of the seasons that are log-transformed.
- **log.ann**: TRUE if the annual reconstruction is log-transformed.
- **N**: The number of targets (number of seasons plus one for the annual reconstruction).
- **sInd**: Indices of the seasons, i.e., 1...N-1
**Value**

Objective function value: least squares plus a penalty term.

---

**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**

```r
Z <- make_Z(p1Seasonal$Qa, nRuns = 30, frac = 0.25, contiguous = TRUE)
```

---

**mb_fit**

*Fit parameters with mass balance criterion*

**Description**

Fit parameters with mass balance criterion

**Usage**

`mb_fit(X, Y, lambda, mus, sigmas, log.seasons, log.ann, N, sInd)`
**mb_reconstruction**

**Arguments**

- **X**
  Inputs, must have columns of 1 added

- **Y**
  Observed Dry, Wet, and Annual log-transformed flows

- **lambda**
  Penalty weight.

- **mus**
  A vector of means, one for each target.

- **sigmas**
  A vector of the standard deviations, one for each target.

- **log.seasons**
  A vector containing the indices of the seasons that are log-transformed.

- **log.ann**
  TRUE if the annual reconstruction is log-transformed.

- **N**
  The number of targets (number of seasons plus one for the annual reconstruction).

- **sInd**
  Indices of the seasons, i.e., 1...N-1

**Value**

A one-column matrix of beta value

**Description**

Mass-balance-adjusted reconstruction

**Usage**

```r
mb_reconstruction(
  instQ,
  pc.list,
  start.year,
  lambda = 1,
  log.trans = NULL,
  force.standardize = FALSE
)
```

**Arguments**

- **instQ**
  Instrumental data, in the same order as pc.list. The "season" column must be a factor.

- **pc.list**
  List of PC matrices. The first element is for the first season, second element for second season, and so on. The last element is for the annual reconstruction.

- **start.year**
  The first year of record

- **lambda**
  The penalty weight

- **log.trans**
  A vector containing indices of the targets to be log-transformed. If no transformation is needed, provide NULL.

- **force.standardize**
  If TRUE, all observations are standardized. See Details.
nRMSE

Value
A data.table with the following columns: season, year, Q, and lambda.

Details
If some targets are log transformed and some are not, they will have different scales, which affects the objective function. In this case the observations will be standardized so that they are in the same range. Otherwise, standardization are skipped for speed. However, in some cases you may want to standardize any ways, for example when flows in some months are much larger than in other months. In this case, set force.standardize = TRUE.

Examples
mb_reconstruction(p1Seasonal, pc3seasons, 1750, lambda = 1, log.trans = 1:3)

---

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 value

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 value

**Examples**

\[ NSE(rnorm(100), rnorm(100)) \]

---

**obj_fun**  

**Objective function from parameters**

**Description**

This is a wrapper for `lsq_mb()`. It first calculates `hat`, then calls `lsq_mb()`. This is used in `optim()`, so it returns a scalar.

**Usage**

\[ obj_fun(beta, X, Y, lambda, mus, sigmas, log.seasons, log.ann, N, sInd) \]

**Arguments**

- `beta`: Parameters
- `X`: Inputs, must have columns of 1 added
- `Y`: Observed Dry, Wet, and Annual log-transformed flows
- `lambda`: Penalty weight.
- `mus`: A vector of means, one for each target.
- `sigmas`: A vector of the standard deviations, one for each target.
log. seasons  A vector containing the indices of the seasons that are log-transformed.
log. ann    TRUE if the annual reconstruction is log-transformed.
N           The number of targets (number of seasons plus one for the annual reconstruction).
sInd       Indices of the seasons, i.e., 1...N-1

Value

Objective function value

---

p1Seasonal  *Seasonal streamflow at P.1 station*

**Description**

Streamflow at P.1 station (Chiang Mai, Thailand) for three reconstruction targets: dry season (NJ, Nov-Jun), wet season (JO, Jul-Oct), and water year (WY, Nov-Oct), as used by Nguyen et al (2020).

**Usage**

p1Seasonal

**Format**

A data table with 246 rows and 3 variables:

- **season** a factor with three levels: "NJ", "JO", and "WY"
- **year** integer, from 1922 to 2003
- **Qa** Annual flow for each target

**Source**


**References**

**pc3seasons**

*Principal components of tree rings*

**Description**

Principal components of the Southeast Asian Dendrochronology Network, after appropriate sites have been selected for each season.

**Usage**

`pc3seasons`

**Format**

A list with three elements (NJ, JO, and WY), each element is a principal component matrix.

**Source**


**References**


---

**prepend_ones**

*Prepend a column of ones*

**Description**

Prepend a column of ones

**Usage**

`prepend_ones(x)`

**Arguments**

- `x` The input matrix

**Value**

`x` with a column of ones prepended, which is named 'Int' for 'intercept'
**RE**

*Reduction of Error*

**Description**

Reduction of Error

**Usage**

\[ \text{RE}(\text{yhat}, y, \text{yc}_{\text{bar}}) \]

**Arguments**

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

**Value**

RE value

**Examples**

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

---

**rowScale**

*Scale rows of a Matrix*

**Description**

Similar to colScale

**Usage**

```
rowScale(x, add_attr = TRUE)
```

**Arguments**

- **x**: A matrix.
- **add_attr**: If TRUE, the column means and standard deviations are returned as attributes. This is consistent with `base::scale()`.

**Value**

The scaled matrix.
rowUnscale  

Unscale rows of a matrix

Description
Backtransform a matrix that was scaled before.

Usage
rowUnscale(x, rm, rsd)

Arguments
- `x`  
  A matrix.
- `rm`  
  A vector of row means
- `rsd`  
  A vector of row standard deviations

Value
The unscaled matrix
Index

* datasets
  p1Seasonal, 11
  pc3seasons, 12

back_trans, 2
base::scale(), 3, 13

calculate_metrics, 3
colScale, 3, 13
colUnscale, 4
cv_mb, 4

KGE, 6

lsq_mb, 6

make_Z, 7
mb_fit, 7
mb_reconstruction, 8

nRMSE, 9
NSE, 10

obj_fun, 10

p1Seasonal, 11
pc3seasons, 12
prepend_ones, 12

RE, 13
rowScale, 13
rowUnscale, 14