Package ‘clr’

July 29, 2019

Type Package
Title Curve Linear Regression via Dimension Reduction
Version 0.1.2
Author Amandine Pierrot
with contributions and/or help from Qiwei Yao, Haeran Cho, Yannig Goude and
Tony Aldon.
Maintainer Amandine Pierrot <amandine.m.pierrot@gmail.com>
Copyright EDF R&D 2017
Description A new methodology for linear regression with both curve response
and curve regressors, which is described in Cho, Goude, Brossat and Yao
<doi:10.1007/978-3-319-18732-7_3>. The key idea behind this methodology is
dimension reduction based on a singular value decomposition in a Hilbert
space, which reduces the curve regression problem to several scalar linear
regression problems.
License LGPL (>= 2.0)
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
Depends R (>= 2.10)
Imports magrittr, lubridate, dplyr, stats
NeedsCompilation no
Repository CRAN
Date/Publication 2019-07-29 09:00:02 UTC

R topics documented:

  clr-package ......................................................... 2
  clr .............................................................. 2
  clrdatal .......................................................... 4
  clust_test ......................................................... 5
Description

`clr` provides functions for curve linear regression via dimension reduction.

Details

The package implements a new methodology for linear regression with both curve response and curve regressors, which is described in Cho et al. (2013) and Cho et al. (2015). The CLR model performs a data-driven dimension reduction, based on a singular value decomposition in a Hilbert Space, as well as a data transformation so that the relationship between the transformed data is linear and can be captured by simple regression models.

Author(s)

Amandine Pierrot <amandine.m.pierrot@gmail.com>

with contributions and help from Qiwei Yao, Haeran Cho, Yannig Goude and Tony Aldon.

References

These provide details for the underlying `clr` methods.


description

Fits a curve linear regression (CLR) model to data, using dimension reduction based on singular value decomposition.

Usage

```r
clr(Y, X, clust = NULL, qx_estimation = list(method = "pctvar", param = 0.999), ortho_Y = TRUE, qy_estimation = list(method = "pctvar", param = 0.999), d_estimation = list(method = "cor", param = 0.5))
```
Arguments

Y  An object of class `clrdata` or `matrix`, of the response curves (one curve a row).

X  An object of class `clrdata` or `matrix`, of the regressor curves (one curve a row).

clust  If needed, a list of row indices for each cluster, to obtain (approximately) homogeneous dependence structure inside each cluster.

qx_estimation  A list containing both values for 'method' (among 'ratio', 'ratioM', 'pctvar', 'fixed') and for 'param' (depending on the selected method), in order to choose how to estimate the dimension of X (in the sense that its Karhunen-Loève decomposition has qx terms only).

ortho_Y  If TRUE then Y is orthogonalized.

qy_estimation  Same as for qx_estimation, if ortho_Y is set to TRUE.

d_estimation  A list containing both values for 'method' (among 'ratio', 'pctvar', 'cor') and for 'param' (depending on the selected method), in order to choose how to estimate the correlation dimension.

Value

An object of class `clr`, which can be used to compute predictions. This `clr` object is a list of lists: one list by cluster of data, each list including:

- residuals  The matrix of the residuals of d_hat simple linear regressions.
- b_hat  The vector of the estimated coefficient of the d_hat simple straight line regressions.
- eta  The matrix of the projections of X.
- xi  The matrix of the projections of Y.
- qx_hat  The estimated dimension of X.
- qy_hat  The estimated dimension of Y.
- d_hat  The estimated correlation dimension.
- X_mean  The mean of the regressor curves.
- X_sd  The standard deviation of the regressor curves.
- Y_mean  The mean of the response curves.
- ortho_Y  The value which was selected for ortho_Y.
- GAMMA  The standardized transformation for X.
- INV_DELTA  The standardized transformation for Y to predict if ortho_Y was set to TRUE.
- phi  The eigenvectors for Y to predict if ortho_Y was set to FALSE.
- idx  The indices of the rows selected from X and Y for the current cluster.

See Also

`clr-package`, `clrdata` and `predict.clr`. 
Examples

library(clr)
data(gb_load)
data(clust_train)

clr_load <- clrdata(x = gb_load$ENGLAND_WALES_DEMAND,
                   order_by = gb_load$TIMESTAMP,
                   support_grid = 1:48)

## data cleaning: replace zeros with NA
clr_load[rowSums((clr_load == 0) * 1) > 0, ] <- NA
matplot(t(clr_load), ylab = 'Daily loads', type = 'l')

Y <- clr_load[2:nrow(clr_load), ]
X <- clr_load[1:(nrow(clr_load) - 1), ]

begin_pred <- which(substr(rownames(Y), 1, 4) == '2016')[1]
Y_train <- Y[1:(begin_pred - 1), ]
X_train <- X[1:(begin_pred - 1), ]

## Example without any cluster
model <- clr(Y = Y_train, X = X_train)

## Example with clusters
model <- clr(Y = Y_train, X = X_train, clust = clust_train)

clrdata

Create an object of clrdata

Description

clrdata is used to create a clrdata object from raw data inputs.

Usage

clrdata(x, order_by, support_grid)

Arguments

x A vector containing the time series values
order_by A corresponding vector of unique time-dates - must be of class 'POSIXct'
support_grid A vector corresponding to the support grid of functional data

Value

An object of class clrdata with one function a row. As it inherits the matrix class, all matrix methods remain valid. If time-dates are missing in x, corresponding NA functions are added by clrdata so that time sequence is preserved between successive rows.
clust_test

Examples

```r
library(clr)
data(gb_load)

clr_load <- clrdata(x = gb_load$ENGLAND_WALES_DEMAND,
    order_by = gb_load$TIMESTAMP,
    support_grid = 1:48)

head(clr_load)
dim(clr_load)
summary(clr_load)

matplot(t(clr_load), ylab = 'Daily loads', type = 'l')
lines(colMeans(clr_load, na.rm = TRUE),
    col = 'black', lwd = 2)

clr_weather <- clrdata(x = gb_load$TEMPERATURE,
    order_by = gb_load$TIMESTAMP,
    support_grid = 1:48)

summary(clr_weather)
plot(1:48,
    colMeans(clr_weather, na.rm = TRUE),
    xlab = 'Instant', ylab = 'Mean of temperatures',
    type = 'l', col = 'cornflowerblue')
```

---

clust_test  Electricity load example: clusters on test set

Description

A list with observations by cluster for prediction

Usage

clust_test

Format

A list of length 14:

14 clusters of loads, depending on both daily and seasonal classification, banking holidays being removed

Author(s)

Amandine Pierrot <amandine.m.pierrot@gmail.com>
clust_train  

*Electricity load example: clusters on train set*

**Description**

A list with observations by cluster for fitting

**Usage**

```r
clust_train
```

**Format**

A list of length 14:

- 14 clusters of loads, depending on both daily and seasonal classification, banking holidays being removed

**Author(s)**

Amandine Pierrot <amandine.m.pierrot@gmail.com>

gb_load  

*Electricity load from Great Britain*

**Description**

A dataset containing half-hourly electricity load from Great Britain from 2011 to 2016, together with observed temperatures. Temperatures are computed from weather stations all over the country. It is a weighted averaged temperature depending on population geographical distribution.

**Usage**

```r
gb_load
```

**Format**

A data frame with 105216 rows and 7 variables:

- `SETTLEMENT_DATE` date, the time zone being Europe/London
- `SETTLEMENT_PERIOD` time of the day
- `TIMESTAMP` date-time, the time zone being Europe/London
- `ENGLAND_WALES_DEMAND` British electric load, measured in MW, on average over the half hour
- `TEMPERATURE` observed temperature in Celsius
**predict.clr**

**MV** percentage of missing values when averaging over weather stations, depending on the weight of the station

**DAY_TYPE** type of the day of the week, from 1 for Sunday to 7 for Saturday, 8 being banking holidays

**Author(s)**

Amandine Pierrot <amandine.m.pierrot@gmail.com>

**Source**

National Grid

National Centers for Environmental Information

---

**predict.clr**

*Prediction from fitted CLR model(s)*

**Description**

Takes a fitted `clr` object produced by `clr()` and produces predictions given a new set of functions or the original values used for the model fit.

**Usage**

```r
## S3 method for class 'clr'
predict(object, newX = NULL, newclust = NULL,
        newXmean = NULL, simplify = FALSE, ...)
```

**Arguments**

- **object** A fitted `clr` object produced by `clr()`.
- **newX** An object of class `clrdata` or a matrix with one function a row. If this is not provided then predictions corresponding to the original data are returned. If `newX` is provided then it should contain the same type of functions as the original ones (same dimension, same clusters eventually, ...).
- **newclust** A new list of indices to obtain (approximately) homogeneous dependence structure inside each cluster of functions.
- **newXmean** To complete when done
- **simplify** If TRUE, one matrix of predicted functions is returned instead of a list of matrices (one matrix by cluster). In the final matrix, rows are sorted by increasing row numbers.
- **...** Further arguments are ignored.

---


2 [https://gis.ncdc.noaa.gov/maps/ncei/cdo/alltimes](https://gis.ncdc.noaa.gov/maps/ncei/cdo/alltimes)
Value

predicted functions

Examples

```r
library(clr)
data(gb_load)

clr_load <- clrdata(x = gb_load$ENGLAND_WALES_DEMAND,
                     order_by = gb_load$TIMESTAMP,
                     support_grid = 1:48)

# data cleaning: replace zeros with NA
clr_load[rowSums((clr_load == 0) * 1) > 0, ] <- NA

Y <- clr_load[2:nrow(clr_load), ]
X <- clr_load[1:(nrow(clr_load) - 1), ]

begin_pred <- which(substr(rownames(Y), 1, 4) == '2016')[1]
Y_train <- Y[1:(begin_pred - 1), ]
X_train <- X[1:(begin_pred - 1), ]
Y_test <- Y[begin_pred:nrow(Y), ]
X_test <- X[begin_pred:nrow(X), ]

## Example without any cluster
model <- clr(Y = Y_train, X = X_train)
pred_on_train <- predict(model)
head(pred_on_train[[1]])
pred_on_test <- predict(model, newX = X_test)
head(pred_on_test[[1]])

## Example with clusters
model <- clr(Y = Y_train, X = X_train, clust = clust_train)
pred_on_train <- predict(model)
str(pred_on_train)
head(pred_on_train[[1]])
pred_on_test <- predict(model, newX = X_test, newclust = clust_test,
                        simplify = TRUE)
str(pred_on_test)
head(pred_on_test)

# With dates as row names
rownames(pred_on_test) <- rownames(Y_test)[as.numeric(rownames(pred_on_test))]
```