Package ‘ctmva’

November 20, 2025

Type Package

Title Continuous-Time Multivariate Analysis

Version 1.5.0

Date 2025-11-20

Maintainer Biplab Paul <paul.biplab497@gmail.com>

Description Implements a basis function or functional data analysis framework for several techniques of multivariate analysis in continuous-time setting. Specifically, we introduced continuous-time analogues of several classical techniques of multivariate analysis, such as principal component analysis, canonical correlation analysis, Fisher linear discriminant analysis, K-means clustering, and so on. Details are in Biplab Paul, Philip T. Reiss, Erjia Cui and Noemi Foa (2025) ``Continuous-time multivariate analysis'' <doi:10.1080/10618600.2024.2374570>.

License GPL (>= 2)

Depends R (>= 4.1.0)

Imports fda, polynom, MASS, mgcv, Matrix

Suggests dplyr, ggplot2, wbwdi

RoxygenNote 7.3.3

Author Biplab Paul [aut, cre],
Philip Tzvi Reiss [aut],
Noemi Foa [aut],
Dror Arbiv [aut]

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2025-11-20 15:00:02 UTC

Contents

`cca.ct` *Continuous-time canonical correlation analysis*

Description

A continuous-time version of canonical correlation analysis (CCA).

Usage

```
cca.ct(fdobj1, fdobj2)
```

Arguments

`fdobj1, fdobj2`	a pair of continuous-time multivariate data sets, of class `"fd"`

Value

A list consisting of

`vex1, vex2`	matrices defining the canonical variates. The first columns of each give the coefficients defining the first pair of canonical variates; and so on.
`cor`	canonical correlations, i.e., correlations between the pairs of canonical variates

Note

Columns of the output matrix `vex2` are flipped as needed to ensure positive correlations.

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

cancor, for classical CCA

Examples

```
## Not run:

# CCA relating Canadian daily temperature and precipitation data
require(fda)
data(CanadianWeather)
daybasis <- create.bspline.basis(c(0,365), nbasis=80)
tempfd <- smooth.basis(day.5, CanadianWeather$dailyAv[,,"Temperature.C"], daybasis)$fd
precfd <- smooth.basis(day.5, CanadianWeather$dailyAv[,,"log10precip"], daybasis)$fd
tpcor <- cca.ct(tempfd, precfd)
par(mfrow=1:2)
barplot(tpcor$vex1[,1], horiz=TRUE, las=1, main="Temperature",
            sub="First canonical coefficients vector")
barplot(tpcor$vex2[,1], horiz=TRUE, las=1, main="Log precipitation",
            sub="First canonical coefficients vector")


## End(Not run)
```

---

ccor *Curve correlation*

Description

Inputs raw data representing two curves, applies penalized B-spline smoothing to the two curves, and computes the curve correlation between them via a call to cor.ct.

Usage

```
ccor(y, time, curve = NULL, k = 15, min.overlap = 0, min.n = 8)
```

Arguments

y	either a vector or a two-column matrix, without missing values; see Details
time	a vector of time points
curve	curve indicator; see Details
k	number of B-spline basis functions
min.overlap	minimum overlap of the two curves' time ranges
min.n	minimum number of observations per curve

Details

If y is a two-column matrix, the two curves are observed at the time points given by time; in this case length(time) must equal nrow(y), and curve is ignored. If y is a vector, it must have the same length as both time and curve. In this case y contains the observations on both curves, while elements of time and curve identify the observation time and the curve being observed, respectively.

Value

A list with components

y, time	the supplied y and time
mod1, mod2	models for the two curves, outputted by gam
fd1, fd2	functional data objects (see fd) for the two curves
estcor	estimated curve correlation

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il>, Noemi Foa, Dror Arbiv and Biplab Paul <paul.biplab497@gmail.com>

See Also

cor.ct, b.spline

Examples

```
# see example for ccor_posim
```

---

ccor_boot *Bootstrap confidence interval estimation for curve correlation*

Description

Inputs raw data representing two curves, and computes a bootstrap confidence interval for the curve correlation between them.

Usage

```
ccor_boot(
  y,
  time,
  curve = NULL,
  k = 15,
  ndraw = 299,
  conf = 0.95,
  min.overlap = 0,
  min.n = 8
)
```

Arguments

y, time, curve, k, min.overlap, min.n	see ccor
ndraw	number of bootstrap samples
conf	confidence level

Value

A list with components

cor	curve correlation (for the full data)
boot.cor	curve correlations for the ndraw bootstrap samples
ci	confidence interval for the curve correlation

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il>, Noemi Foa, Dror Arbiv and Biplab Paul <paul.biplab497@gmail.com>

See Also

ccor, ccor_posim

Examples

```
# see example for ccor_posim
```

---

ccor_posim — *Credible interval estimation for curve correlation based on posterior simulation*

Description

Inputs raw data representing two curves, and computes a credible interval for the curve correlation between them simulating from the approximate posterior distribution of the joint spline coefficient vector.

Usage

```
ccor_posim(
  y,
  time,
  curve = NULL,
  method,
  k = 15,
  conf = 0.95,
  ndraw = 999,
  min.overlap = 0
)
```

Arguments

y, time, curve, k, min.overlap
: see ccor

method
: "indep" (curves fitted independently) or "mvn" (curves fitted together, with error correlation estimated based on multivariate normal likelihood)

conf
: confidence level

ndraw
: number of draws from posterior distribution of spline coefficient vector

Value

A list with components

cor
: curve correlation

model
: the model for the two curves (if method=="mvn"), or a list of the two curves' models (if method=="indep")

bsb
: B-spline basis (from package fda) used for the curves

Vc.fda
: corrected posterior covariance matrix for the coefficients with respect to the B-spline basis bsb (see the component $Vc in gamObject)

sims
: curve correlations for the ndraw draws from the posterior distribution

ci
: credible interval for the curve correlation

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il>, Noemi Foa, Dror Arbiv and Biplab Paul <paul.biplab497@gmail.com>

See Also

ccor, ccor_boot, mvn

Examples

```
## Not run:
if (interactive () &&
    requireNamespace("wbwdi", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("dplyr", quietly = TRUE)) {

    # Curve correlation of per capita GDP and fertility rate in Paraguay
    wdi_dat <- wbwdi::wdi_get(entities = c("PRY"), start_year=1960, end_year=2023,
                    indicators = c("NY.GDP.PCAP.KD","SP.DYN.TFRT.IN"), format="wide") |>
        dplyr::rename(percapitaGDP = NY.GDP.PCAP.KD, fertility = SP.DYN.TFRT.IN)
    ggplot2::ggplot(wdi_dat, aes(percapitaGDP, fertility, color=year)) + geom_point()

    y <- as.matrix(wdi_dat[ , c("percapitaGDP", "fertility")])

    set.seed(345)

    ci <- list()
```

```
    ci[[1]] <- ccor_posim(y=y, time=wdi_dat$year, method="indep")
    ci[[2]] <- ccor_posim(y=y, time=wdi_dat$year, method="mvn")
    ci[[3]] <- ccor_boot(y=y, time=wdi_dat$year, ndraw=399)

    tabl <- matrix(NA, 3, 3)
    for (k in 1:3) tabl[k, ] <- c(ci[[k]]$cor, ci[[k]]$ci)
   dimnames(tabl) <- list(c("Posim_indep", "Posim_MVN", "Bootstrap"), c("Est","Lower95","Upper95"))
    round(tabl, 4)
}
## End(Not run)
```

`center.ct` *Center a continuous-time multivariate data set*

Description

Subtracts the (continuous-time) mean of each of the variables. This is analogous to column-centering an $n \times p$ data matrix.

Usage

```
center.ct(fdobj)
```

Arguments

fdobj	continuous-time multivariate data set of class "fd"

Value

A centered version of the input data.

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

standardize.ct

cor.ct *Continuous-time correlation or cross-correlation matrix*

Description

Computes the correlation matrix of a continuous-time multivariate data set represented as an fd object; or the cross-correlation matrix of two such data sets.

Usage

```
cor.ct(fdobj1, fdobj2 = fdobj1, common_trend = FALSE)
```

Arguments

fdobj1	continuous-time multivariate data set of class "fd"
fdobj2	an optional second data set
common_trend	logical: centering wrt mean function if TRUE, without centering if FALSE (the default)

Value

If fdobj1 and fdobj2 each consist of a single curve, the (scalar) CT correlation between them. Otherwise a matrix of (cross-) correlations is returned.

Note

When fdobj1==fdobj2, cor.ct(...) is the same as cov2cor(cov.ct(...)).

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

center.fd, for centering of "fd" objects; inprod.cent

Examples

```
## Not run:

# Canadian temperature data

require(fda)
require(corrplot)
data(CanadianWeather)
daybasis <- create.fourier.basis(c(0,365), nbasis=55)
tempfd <- smooth.basis(day.5, CanadianWeather$dailyAv[,,"Temperature.C"], daybasis)$fd
```

```
## The following yields a matrix of correlations that are all near 1:
rawcor <- cor.ct(tempfd)
corrplot(rawcor, method = 'square', type = 'lower', tl.col="black", tl.cex = 0.6)
## This occurs due to a strong seasonal trend that is common to all stations
## Removing this common trend leads to a more interesting result:
dtcor  <- cor.ct(tempfd, common_trend = TRUE)
ord <- corrMatOrder(dtcor)
dtcord <- dtcor[ord,ord]
corrplot(dtcord, method = 'square', type = 'lower', tl.col="black", tl.cex = 0.6)


## End(Not run)
```

cov.ct — *Continuous-time covariance or cross-covariance matrix*

Description

Computes the covariance matrix of a continuous-time multivariate data set represented as an `fd` object; or the cross-covariance matrix of two such data sets.

Usage

```
cov.ct(fdobj1, fdobj2 = fdobj1, common_trend = FALSE)
```

Arguments

`fdobj1`	continuous-time multivariate data set of class "`fd`"
`fdobj2`	an optional second data set
`common_trend`	logical: centering with respect to the mean function if `TRUE`, without centering if `FALSE` (the default)

Value

A matrix of (cross-) covariances

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

`cor.ct`

Examples

```
# see example for cor.ct, which works similarly
```

inprod.cent *Centered inner product matrix for a basis or pair of bases*

Description

Several methods of continous-time multivariate analysis require a matrix of inner products of pairs of centered functions from a basis, such as a B-spline basis, or pairs consisting of one function from each of two bases. This function computes such matrices via 7-point Newton-Cotes integration, which is exact for cubic B-splines. For a Fourier basis with the inner product taken over the entire range, a simple closed form is used instead of integration.

Usage

```
inprod.cent(basis1, basis2 = basis1, rng = NULL)
```

Arguments

`basis1`	basis object from the `fda` package.
`basis2`	an optional second basis
`rng`	time range. By default, the entire range spanned by the basis, or the intersection of the ranges of the two bases.

Value

Matrix of inner products of each pair of centered basis functions.

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

`create.bspline.basis` from package `fda`, for the most commonly used basis object type.

Examples

```
## Not run:

require(fda)
bbasis6 <- create.bspline.basis(nbasis=6)
inprod.cent(bbasis6)
fbasis7 <- create.fourier.basis(nbasis=7)
inprod.cent(fbasis7)

## End(Not run)
```

`kmeans.ct` *Continuous-time k-means clustering*

Description

A continuous-time version of k-means clustering in which each cluster is a time segments or set of time segments.

Usage

```
kmeans.ct(
  fdobj,
  k,
  common_trend = FALSE,
  init.pts = NULL,
  tol = 0.001,
  max.iter = 100
)
```

Arguments

`fdobj`	continuous-time multivariate data set of class "`fd`"
`k`	number of clusters
`common_trend`	logical: Should the curves be centered with respect to the mean function? Defaults to `FALSE`.
`init.pts`	a set of k time points. The observations at these time points serve as initial values for the k means. Randomly generated if not supplied.
`tol`	convergence tolerance for the k means
`max.iter`	maximum number of iterations

Value

Object of class "`kmeans.ct`", a list consisting of

`fdobj`	the supplied `fdobj`
`means`	means of the k clusters
`transitions`	transition points between segments
`cluster`	cluster memberships in the segments defined by the transitions
`size`	length of each cluster, i.e. sum of lengths of subintervals making up each cluster
`totisd`	total integrated sum of distances from the overall mean, analogous to `totss` from `kmeans`
`withinisd`	within-cluster integrated sum of distances, i.e. integrated sum of distances from each cluster mean
`tot.withinisd`	total within-cluster integrated sum of distances, i.e. `sum(withinisd)`
`betweenisd`	between-cluster integrated sum of distances, i.e. `totisd-tot.withinss`

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

kmeans, plot.kmeans.ct, silhouette.ct

Examples

```
## Not run:

require(fda)
data(CanadianWeather)
daybasis <- create.bspline.basis(c(0,365), nbasis=55)
tempfd <- smooth.basis(day.5, CanadianWeather$dailyAv[,,"Temperature.C"], daybasis)$fd
kmtemp3 <- kmeans.ct(tempfd, 3)
plot(kmtemp3, axes=FALSE, xlab='', ylab="Temperature")
axesIntervals(); box()
plot(silhouette.ct(kmtemp3), axes=FALSE, xlab='')
axesIntervals(); box()

## End(Not run)
```

`lda.ct` *Continuous-time Fisher's linear discriminant analysis*

Description

A continuous-time version of Fisher's LDA, in which segments of the time interval take the place of groups of observations.

Usage

```
lda.ct(fdobj, partition, part.names = NULL)
```

Arguments

`fdobj`	continuous-time multivariate data set of class "fd"
`partition`	a priori break points dividing the time interval into segments
`part.names`	optional character vector of names for the segments

Details

The `means` and `scaling` components of the output are similar to `lda`, but unlike that function, `lda.ct` performs only *Fisher's* LDA and cannot incorporate priors or perform classification.

Value

Object of class "lda.ct", a list consisting of

means	means of the variables within each segment
scaling	matrix of coefficients defining the discriminants (as in lda)
values	eigenvalues giving the ratios of between to within sums of squares
partition	the supplied partition
fdobj	linear discriminants represented as an "fd" object
nld	number of linear discriminants

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

plot.lda.ct; lda, for the classical version

Examples

```
## see end of example in ?pca.ct
```

---

meanbasis *Compute means of basis functions*

Description

Given a basis object as defined in the **fda** package (see basisfd), this function simply computes the vector of means of the basis functions. Used internally.

Usage

```
meanbasis(basis, rng = NULL)
```

Arguments

basis	a basis object of class "basisfd"
rng	time range. By default, the entire interval spanned by the basis. Must be left NULL for Fourier bases.

Value

Vector of means of the basis functions

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

Examples

```
require(fda)
bbasis6 <- create.bspline.basis(nbasis=6)
meanbasis(bbasis6)
meanbasis(bbasis6, c(.3,.6))
fbasis11 <- create.fourier.basis(nbasis=11)
meanbasis(fbasis11)
```

---

pca.ct *Continuous-time principal component analysis*

Description

A continuous-time version of principal component analysis.

Usage

```
pca.ct(fdobj, cor = FALSE, common_trend = FALSE)
```

Arguments

`fdobj`	continuous-time multivariate data set of class "`fd`"
`cor`	logical: use correlation matrix if `TRUE`, covariance if `FALSE` (the default)
`common_trend`	logical: Should the curves be centered with respect to the mean function? Defaults to `FALSE`.

Value

Returns a list including:

`var`	variances of the principal components.
`loadings`	the matrix of loadings (i.e., its columns are the eigenvectors of the continuous-time covariance).
`scorefd`	score functions.

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

`cov.ct`; `princomp`, for the classical version

Examples

```
## Not run:

# Data for one session from a classic EEG data set
require(fda)
require(eegkit)
data(eegdata)
data(eegcoord)
longdat <- subset(eegdata, subject=="co2a0000369" & trial==0)
widedat <- reshape(longdat, direction="wide", drop=c("subject","group","condition","trial"),
                 v.names="voltage",idvar="channel")

# Convert time series for 64 channels to a functional data object
bsb <- create.bspline.basis(c(0,255),nbasis=30)
fdo <- Data2fd(argvals=0:255, y=t(as.matrix(widedat[,-1])), basisobj=bsb)
plot(fdo)

# Now do PCA and display first loadings for 3 PC's,
# along with percent variance explained by each
pcc <- pca.ct(fdo)
pve <- 100*pcc$var/sum(pcc$var)
par(mfrow=c(1,3))
cidx <- match(widedat[,1],rownames(eegcoord))
eegspace(eegcoord[cidx,4:5],pcc$loadings[,1], colorlab="PC1 loadings",
         main=paste0(round(pve[1],0), "%"), mar=c(17,3,12,2), cex.main=2)
eegspace(eegcoord[cidx,4:5],pcc$loadings[,2], colorlab="PC2 loadings",
         main=paste0(round(pve[2],0), "%"), mar=c(17,3,12,2), cex.main=2)
eegspace(eegcoord[cidx,4:5],pcc$loadings[,3], colorlab="PC3 loadings",
         main=paste0(round(pve[3],0), "%"), mar=c(17,3,12,2), cex.main=2)

# Linear discriminant analysis: discriminating among the 1st, 2nd and 3rd portions
#  of the time interval
ld <- lda.ct(fdo, c(85,170))
plot(ld)
eegspace(eegcoord[cidx,4:5],ld$scaling[,1], colorlab="LD1 coefficients",
         mar=c(17,3,12,2), cex.main=2)
eegspace(eegcoord[cidx,4:5],ld$scaling[,2], colorlab="LD2 coefficients",
         mar=c(17,3,12,2), cex.main=2)


## End(Not run)
```

`plot.kmeans.ct` *Plot a kmeans.ct object*

Description

Plots a continuous-time k-means clustering object generated by a call to `kmeans.ct`.

Usage

```
## S3 method for class 'kmeans.ct'
plot(
  x,
  plottype = "functions",
  mark.transitions = TRUE,
  col = NULL,
  lty = NULL,
  xlab = "Time",
  ylab = NULL,
  legend = TRUE,
  ncol.legend = 1,
  cex.legend = 1,
  ...
)
```

Arguments

`x`	clustering object produced by `kmeans.ct`
`plottype`	either `"functions"` (the default), to display each variable as a smooth function of time, or `"distance"`, to plot distances from the k cluster means versus time.
`mark.transitions`	logical: Should transitions between clusters be marked with vertical lines? Defaults to `TRUE`.
`col`	plot colors
`lty`	line type
`xlab, ylab`	x- and y-axis labels
`legend`	either a logical variable (whether a legend should be included) or a character vector to appear in the legend. Default is `TRUE`.
`ncol.legend`	number of columns for legend
`cex.legend`	character expansion factor for legend
`...`	other arguments passed to `matplot`

Value

None; a plot is generated.

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il> and Biplab Paul <paul.biplab497@gmail.com>

See Also

`kmeans.ct`, which includes an example

plot.lda.ct *Plot an lda.ct object*

Description

Plots the Fisher's linear discriminant functions generated by a call to `lda.ct`.

Usage

```
## S3 method for class 'lda.ct'
plot(x, ylab = "Discriminants", xlab = "Time", which = NULL, col = NULL, ...)
```

Arguments

`x`	linear discriminant analysis object produced by `lda.ct`
`ylab, xlab`	y- and x-axis labels
`which`	which of the linear discrminants to plot
`col`	color vector
`...`	other arguments passed to `matplot`

Value

None; a plot is generated.

Author(s)

Biplab Paul <paul.biplab497@gmail.com> and Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

`lda.ct`

Examples

```
## see the example at the end of ?pca.ct
```

`plot.silhouette.ct` *Plot a silhouette.ct object*

Description

Plots the silhouette index, generated by a call to `silhouette.ct`, for a continuous-time k-means clustering object.

Usage

```
## S3 method for class 'silhouette.ct'
plot(x, mark.transitions = TRUE, xlab = "Time", ylab = "Silhouette", ...)
```

Arguments

`x`	silhouette object produced by `silhouette.ct`
`mark.transitions`	logical: Should transitions between clusters be marked with vertical lines? Defaults to `TRUE`.
`xlab, ylab`	x- and y-axis labels
`...`	other arguments passed to `plot`

Value

None; a plot is generated.

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

`kmeans.ct`, which includes an example; `silhouette.ct`

`silhouette.ct` *Silhouettes for continuous-time k-means clustering*

Description

Computes the silhouette index, at a grid of time points, for a continuous-time k-means clustering object produced by `kmeans.ct`.

Usage

```
silhouette.ct(kmobj, ngrid = 5000)
```

Arguments

`kmobj`	continuous-time k-means clustering from `kmeans.ct`
`ngrid`	number of equally spaced grid points at which to compute the silhouette index

Value

Object of class "`silhouette.ct`", a list consisting of

`grid`	grid of `ngrid` points spanning the time range
`value`	silhouette index at each point along the grid
`transitions`	transition points between segments
`cluster`	cluster memberships in the segments defined by the transitions
`mean`	mean silhouette index

Note

An error is issued if the grid of time points contains one or more of the cluster transition points. This should not ordinarily occur, but if it does, it can be remedied by modifying `ngrid`.

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il>

See Also

`kmeans.ct`, which includes an example; `plot.silhouette.ct`

`standardize.ct` *Center and scale a continuous-time multivariate data set*

Description

Subtracts the (continuous-time) mean and divides by the (continuous-time) standard deviation of each of the variables. This is the continuous-time analogue of taking an $n \times p$ data matrix, subtracting the mean of each column, and dividing by the standard deviation of each column, as is done by `scale(..., center=TRUE, scale=TRUE)`.

Usage

```
standardize.ct(fdobj)
```

Arguments

`fdobj`	continuous-time multivariate data set of class "`fd`"

Value

A standardized (centered and scaled) version of the input data.

Author(s)

Philip Tzvi Reiss <reiss@stat.haifa.ac.il> and Biplab Paul <paul.biplab497@gmail.com>

See Also

`center.ct`

Index