Package 'fkcentroids'

March 24, 2026

Title Functional K-Centroids Clustering Using Phase and Amplitude Components

Version 0.0.3

Description Cluster functional data using phase and amplitude components of each function. By weighting phase and amplitude variation differently, clustering results can be obtained from multiple perspectives. Routines for synchronization, functional k-means clustering, and functional k-medians clustering are provided.

URL `https://github.com/seungwoo-stat/fkcentroids`

BugReports `https://github.com/seungwoo-stat/fkcentroids/issues`

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports fdasrvf, graphics, Kmedians, stats, withr

Depends R (>= 3.5)

LazyData true

NeedsCompilation no

Author Seungwoo Kang [aut, cre] (ORCID: <`https://orcid.org/0000-0001-8082-0794`>), Hee-Seok Oh [aut]

Maintainer Seungwoo Kang <`kangsw@skku.edu`>

Repository CRAN

Date/Publication 2026-03-24 09:10:08 UTC

Contents

fkmeans — *Functional k-Means Clustering Using Phase and Amplitude Components*

Description

Conducts functional k-means clustering by jointly considering phase and amplitude variation. The relative importance of the two components can be explicitly controlled by the user via the multiview parameter α. Optionally, k-means clustering can be performed directly on the observed curves, rather than on their phase and amplitude components. See Details below.

Usage

```
fkmeans_pre(
  Xclrv,
  Y,
  t,
  alpha_scale = 1,
  k,
  itermax = 10,
  nstart = 1,
  algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
  trace = FALSE
)

fkmeans(
  Ytilde,
  x,
  t,
  sync_map = c("auc", "fr", "none"),
  sync_args = NULL,
  alpha_scale = 1,
  k,
  itermax = 10,
  nstart = 1,
  algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"),
  trace = FALSE
)

## S3 method for class 'fkmeans'
fitted(object, method = c("centers", "classes"), ...)
```

Arguments

`Xclrv`	A $(T-1) \times n$ matrix of centered log-ratio velocity transformed phase components evaluated over the intervals defined by the time points `t`. Refer to `X2Xclrv()`.

`Y`	A $T \times n$ matrix of amplitude components evaluated at the time points `t`.
`t`	A numeric vector of length T giving the time points at which the phase and amplitude components are evaluated. This vector must start at 0 and end at 1.
`alpha_scale`	A numeric indicating the value of multiview parameter. The multiview parameter α is set to $\alpha = \alpha_0 \times$`alpha_scale`. See the details below. By default, set to 1.
`k`	A numeric indicating the number of clusters.
`itermax`	A numeric indicating the maximum number of iterations allowed in the k-means algorithm. By default, set to 10.
`nstart`	A numeric indicating the number of initial sets. By default, set to 1.
`algorithm`	A character string indicating the algorithm for k-means clustering. `"Hartigan-Wong"` algorithm is set as default.
`trace`	A boolean. If `TRUE` and `algorithm == "Hartigan-Wong"`, it prints tracing information on the console.
`Ytilde`	A $m \times n$ matrix whose ith column contains the values of the ith observed function evaluated at the m time points `x`.
`x`	A numeric vector of length m giving the observed time points corresponding to `Ytilde`.
`sync_map`	A character string. If `"auc"` (the default), AUC time-synchronizing mapping is used. If `"fr"`, FR time-synchronizing mapping is used. Refer to `auc_sync()` and `fr_sync()`. If `"none"`, time-synchronizing mapping is not used, and the functional clustering is conducted on the observed curves `Ytilde`. Hence, for `"none"`, the arguments `t`, `sync_args`, and `alpha_scale` are ignored.
`sync_args`	If `sync_map == "auc"` it represents a numeric indicating the parameter p used in AUC time-synchronizing mapping. If `sync_map == "fr"` it represent the template function used in FR time-synchronizing mapping.
`...`	Not used.
`object`	A `fkmeans` object, obtained as a result of the function `fkmeans_pre()` or `fkmeans()`.
`method`	A character string. • `"centers"`: Returns cluster centers for each curve. • `"classes"`: Returns a vector of class assignments.

Details

The distance between two observed functions is defined in terms of their phase and amplitude components. For two functions with components (X_1, Y_1) and (X_2, Y_2), the distance is given by

$$\left\{\alpha \|\texttt{clrv}(X_1) - \texttt{clrv}(X_2)\|_2^2 + \|Y_1 - Y_2\|_2^2\right\}^{1/2},$$

where $\|\cdot\|_2$ denotes the usual $\mathbb{L}^2$ norm and $\alpha \geq 0$ is the multiview parameter. For the `clrv` transformation, refer to `X2Xclrv()`.

Based on this distance, k-means clustering is performed.

A reference value α_0, which serves as a baseline around which α may be varied, is selected as follows. The value α_0 is defined as the ratio of the total sum of squares of the amplitude components to that of the phase components.

See the documentation for `stats::kmeans()` and Kang and Oh (2026) for further details.

Value

fkmeans_pre() and fkmeans() return an object of class fkmeans, which is a list containing the following components:

cluster	A vector of integers (from 1:k) indicating the cluster to which each function is allocated.
centers.Xclrv	A $(T-1)\times k$ matrix of phase components' cluster centers (centered log-ratio velocity transformed). This component is not returned when sync_map == "none".
centers.Y	A $T \times k$ matrix of amplitude components' cluster centers. This component is not returned when sync_map == "none".
centers.Ytilde	A $T \times k$ matrix of raw functions' cluster centers. This component is only returned when sync_map == "none".
totss	The total sum of squares.
withinss	A vector of within-cluster sum of squares, one component per cluster.
tot.withinss	Total within-cluster sum of squares, i.e., sum(withinss).
betweenss	The between-cluster sum of squares, i.e., totss - tot.withinss.
size	The number of functions in each cluster.
iter	The number of (outer) iterations.
ifault	Indicator of a possible algorithm problem; refer to stats::kmeans().
alpha0	The reference value α_0. This component is not returned when sync_map == "none".

print() and fitted() methods are supported for the object of class fkmeans. fitted.fkmeans() with method = "centers" returns cluster centers (one for each input point) and method = "classes" returns a vector of class assignments.

References

Kang S. and Oh H.-S. (2026) "Multiview functional clustering using latent representations of phase and amplitude components," *Unpublished Manuscript*.

See Also

stats::kmeans() for multivariate k-means clustering. fkmedians_pre() and fkmedians() for robust functional k-medians clustering. auc_sync() and fr_sync() for time-synchronizing mappings. X2Xclrv() for centered log-ratio velocity transformation.

Examples

```
t <- seq(0, 1, length.out = 100)
sync <- auc_sync(seoul_bike$Ytilde[,1:10], seoul_bike$x, t)
fkmeans_pre(X2Xclrv(sync), sync$Y, t, alpha_scale = 1, k = 2)
fkmeans(seoul_bike$Ytilde[,1:10], seoul_bike$x, t, sync_map = "auc",
  sync_args = 1, alpha_scale = 1, k = 2)
```

fkmedians — *Functional k-Medians Clustering Using Phase and Amplitude Components*

Description

Conducts functional k-medians clustering by jointly considering phase and amplitude variation. The relative importance of the two components can be explicitly controlled by the user via the multiview parameter α. Optionally, k-medians clustering can be performed directly on the observed curves, rather than on their phase and amplitude components. See Details below.

Usage

```
fkmedians_pre(Xclrv, Y, t, alpha_scale = 1, k, niter = 20, nstart = 1)

fkmedians(
  Ytilde,
  x,
  t,
  sync_map = c("auc", "fr", "none"),
  sync_args = NULL,
  alpha_scale = 1,
  k,
  niter = 20,
  nstart = 1
)

## S3 method for class 'fkmedians'
fitted(object, method = c("centers", "classes"), ...)
```

Arguments

`Xclrv`	A $(T-1) \times n$ matrix of centered log-ratio velocity transformed phase components evaluated over the intervals defined by the time points `t`. Refer to `X2Xclrv()`.
`Y`	A $T \times n$ matrix of amplitude components evaluated at the time points `t`.
`t`	A numeric vector of length T giving the time points at which the phase and amplitude components are evaluated. This vector must start at 0 and end at 1.
`alpha_scale`	A numeric indicating the value of multiview parameter. The multiview parameter α is set to $\alpha = \alpha_0 \times$`alpha_scale`. See the details below. By default, set to 1.
`k`	A numeric indicating the number of clusters.
`niter`	A numeric indicating the number of iterations for the k-medians algorithm. By default, set to 20.
`nstart`	A numeric indicating the number of initial sets. By default, set to 1.

`Ytilde`	A $m \times n$ matrix whose ith column contains the values of the ith observed function evaluated at the m time points `x`.
`x`	A numeric vector of length m giving the observed time points corresponding to `Ytilde`.
`sync_map`	A character string. If `"auc"` (the default), AUC time-synchronizing mapping is used. If `"fr"`, FR time-synchronizing mapping is used. Refer to `auc_sync()` and `fr_sync()`. If `"none"`, time-synchronizing mapping is not used, and the functional clustering is conducted on the observed curves `Ytilde`. Hence, for `"none"`, the arguments `t`, `sync_args`, and `alpha_scale` are ignored.
`sync_args`	If `sync_map == "auc"` it represents a numeric indicating the parameter p used in AUC time-synchronizing mapping. If `sync_map == "fr"` it represent the template function used in FR time-synchronizing mapping.
`...`	Not used.
`object`	A `fkmedians` object, obtained as a result of the function `fkmedians_pre()` or `fkmedians()`.
`method`	A character string. • `"centers"`: Returns cluster centers for each curve. • `"classes"`: Returns a vector of class assignments.

Details

The distance between two observed functions is defined in terms of their phase and amplitude components. For two functions with components (X_1, Y_1) and (X_2, Y_2), the distance is given by

$$\left\{\alpha \|\texttt{clrv}(X_1) - \texttt{clrv}(X_2)\|_2^2 + \|Y_1 - Y_2\|_2^2\right\}^{1/2},$$

where $\|\cdot\|_2$ denotes the usual $\mathbb{L}^2$ norm and $\alpha \geq 0$ is the multiview parameter. For the `clrv` transformation, refer to `X2Xclrv()`.

Based on this distance, k-medians clustering is performed. In particular, Weiszfeld algorithm is used to find the geometric median function for each cluster, implemented in `Kmedians::Kmedians()`.

A reference value α_0, which serves as a baseline around which α may be varied, is selected as follows. The value α_0 is defined as the ratio of the total sum of squares of the amplitude components to that of the phase components.

See the documentation for `Kmedians::Kmedians()` (Godichon-Baggioni and Surendran, 2024) and Kang and Oh (2026) for further details.

Value

`fkmedians_pre()` and `fkmedians()` return an object of class `fkmedians`, which is a list containing the following components:

`cluster`	A vector of integers (from `1:k`) indicating the cluster to which each function is allocated.
`centers.Xclrv`	A $(T-1) \times k$ matrix of phase components' cluster centers (centered log-ratio velocity transformed). This component is not returned when `sync_map == "none"`.

`centers.Y`	A $T \times k$ matrix of amplitude components' cluster centers. This component is not returned when `sync_map == "none"`.
`centers.Ytilde`	A $T \times k$ matrix of raw functions' cluster centers. This component is only returned when `sync_map == "none"`.
`withinsrs`	A vector of within-cluster sum of residuals, one component per cluster.
`tot.withinsrs`	Total within-cluster sum of residuals, i.e., `sum(withinsrs)`.
`size`	The number of functions in each cluster.
`iter`	The number of (outer) iterations.
`alpha0`	The reference value α_0. This component is not returned when `sync_map == "none"`.

`print()` and `fitted()` methods are supported for the object of class `fkmedians`. `fitted.fkmedians()` with `method = "centers"` returns cluster centers (one for each input point) and `method = "classes"` returns a vector of class assignments.

References

Godichon-Baggioni A. and Surendran S. (2024) “A penalized criterion for selecting the number of clusters for K-medians,” *Journal of Computational and Graphical Statistics*, **33**(4), 1298–1309.

Kang S. and Oh H.-S. (2026) “Multiview functional clustering using latent representations of phase and amplitude components,” *Unpublished Manuscript*.

See Also

`Kmedians::Kmedians()` for multivariate k-medians clustering. `fkmeans_pre()` and `fkmeans()` for functional k-means clustering. `auc_sync()` and `fr_sync()` for time-synchronizing mappings. `X2Xclrv()` for centered log-ratio velocity transformation.

Examples

```
t <- seq(0, 1, length.out = 100)
sync <- auc_sync(seoul_bike$Ytilde[,1:10], seoul_bike$x, t)
fkmedians_pre(X2Xclrv(sync), sync$Y, t, alpha_scale = 1, k = 2, nstart = 10)
fkmedians(seoul_bike$Ytilde[,1:10], seoul_bike$x, t, sync_map = "auc",
  sync_args = 1, alpha_scale = 1, k = 2, nstart = 10)
```

`seoul_bike` *Seoul Public Bike Rental Records*

Description

Functions generated from Seoul's public bike rental records collected between 00:00 and 24:00 on April 1st, 2025 (Tuesday). Each function represents a single rental station that has at least 24 rentals over the day.

Each function is constructed through a two-step process: (1) each rental record is converted to time in hours, that is, a numeric value between 0 and 24; and (2) Gaussian convolution with bandwidth 1 is applied to the rental times to generate a smooth function for each station.

Usage

```
data(seoul_bike)
```

Format

A list of length 2 containing the following components:

- `Ytilde`: A 73×1784 matrix whose ith column contains the values of the ith observed function (bike station) evaluated at the time points `x`.
- `x`: A length 73 numeric vector representing time in hours at 20-minute intervals. For example, 18.333 corresponds to 6:20 p.m., i.e., `seq(0, 1, length.out = 73)`.

Source

Seoul Open Data Plaza : https://data.seoul.go.kr/dataList/OA-15182/F/1/datasetView.do

References

Kang S. and Oh H.-S. (2026) "Multiview functional clustering using latent representations of phase and amplitude components," *Unpublished Manuscript*.

`syncftn` *Time-Synchronizing Mappings*

Description

Area-under-the-curve (AUC) time-synchronizing mapping and Fisher–Rao (FR) time-synchronizing mapping.

Usage

```
auc_sync(Ytilde, x, t, p = 1)

fr_sync(Ytilde, x, t, template)

## S3 method for class 'syncftn'
plot(x, phase_mode = c("raw", "clrv"), ...)
```

Arguments

`Ytilde`	A $m \times n$ matrix whose ith column contains the values of the ith observed function evaluated at the m time points `x`.
`x`	A numeric vector or a `syncftn` object, depending on the function. • `auc_sync()` and `fr_sync()`: A numeric vector of length m giving the observed time points corresponding to `Ytilde`. • `plot.syncftn()`: A `syncftn` object, obtained as a result of the function `auc_sync()` or `fr_sync()`.

`t`	A numeric vector of length T giving the time points at which the phase and amplitude components are evaluated. This vector must start at 0 and end at 1.
`p`	A numeric value specifying the power parameter of the AUC time-synchronizing mapping. The default is `p = 1`.
`template`	A numeric vector of length T giving the template function value evaluated at the time points `t`.
`phase_mode`	A character string. • `raw` (the default): Plots phase components in their original form. • `clrv`: Plots phase components after centered log-ratio velocity transformation. Refer to `X2Xclrv()`.
`...`	Further graphical parameters supplied to the internal `graphics::matplot()` function. `main` and `ylab` arguments are ignored.

Details

Let $\tilde{Y}(x)$ be an observed function defined on $x \in [T_1, T_2]$. The AUC time-synchronizing mapping is defined as

$$\varphi_{\tilde{Y}}(x) := \left(\frac{\int_{T_1}^{x} |\tilde{Y}(s)|^p \, \mathrm{d}s}{\int_{T_1}^{T_2} |\tilde{Y}(s)|^p \, \mathrm{d}s} \right)^{1/p}, \quad x \in [T_1, T_2].$$

For further details, see Liu and Müller (2004) and Kang and Oh (2026).

The FR time-synchronizing mapping is defined as

$$\varphi_{\tilde{Y}} := \underset{\varphi}{\operatorname{argmin}}\, d(\tilde{Y} \circ \varphi^{-1}, Y_0),$$

where

$$d(Y_1, Y_2) := \left\| \operatorname{sgn}(DY_1)\sqrt{|DY_1|} - \operatorname{sgn}(DY_2)\sqrt{|DY_2|} \right\|_2,$$

with $\operatorname{sgn}(u) = 1$ if $u \geq 0$ and -1 otherwise. The phase component is then defined as $X(t) := \varphi_{\tilde{Y}}^{-1}(t)$, and the amplitude component is defined as $Y(t) := (\tilde{Y} \circ \varphi_{\tilde{Y}}^{-1})(t)$ for $t \in [0, 1]$. For further details, see Srivastava et al. (2011) and Kang and Oh (2026).

Value

`auc_sync()` and `fr_sync()` returns a object of class `syncftn`, which is a list containing the following components, obtained from AUC synchronization and FR synchronization, respectively:

`X`	A $T \times n$ matrix of phase components evaluated at the time points `t`.
`Y`	A $T \times n$ matrix of amplitude components evaluated at the time points `t`.

`plot.syncftn()` plots phase and amplitude components of each observed function.

References

Kang S. and Oh H.-S. (2026) “Multiview functional clustering using latent representations of phase and amplitude components,” *Unpublished Manuscript*.

Liu X. and Müller H.-G. (2004).“Functional convex averaging and synchronization for time-warped random curve,” *Journal of the American Statistical Association*, **99**(467), 687–699.

Srivastava A., Wu W., Kurtek S., Klassen E., and Marron J. S. (2011) “Registration of functional data using Fisher–Rao metric,” *arXiv preprint arXiv:1103.3817*.

See Also

fdasrvf::pair_align_functions() for FR synchronization method. X2Xclrv() for centered log-ratio velocity transformation. fkmeans() and fkmedians() for k-centroids clustering using phase and amplitude components.

Examples

```
t <- seq(0, 1, length.out = 100)
sync <- auc_sync(seoul_bike$Ytilde[,1:10], seoul_bike$x, t)
plot(sync)
oldpar <- par(mfrow = c(1,2))
plot(sync, col = 1)

template <- 5 * dnorm(t, 0.2, 0.1) + 5 * dnorm(t, 0.8, 0.1)
sync <- fr_sync(seoul_bike$Ytilde[,1:10], seoul_bike$x, t, template)
plot(sync, col = 1)
lines(t, template, col = 2)
par(oldpar)
```

X2Xclrv *Centered Log-Ratio Velocity Transformation*

Description

Transform phase components using centered log-ratio velocity (clrv) transformation.

Usage

```
X2Xclrv(X, t)

## S3 method for class 'matrix'
X2Xclrv(X, t)

## S3 method for class 'syncftn'
X2Xclrv(X, t = attr(X, "t"))

## S3 method for class 'Xclrv'
plot(x, ...)
```

Arguments

X	A matrix representing phase components, or an object of class syncftn. If a matrix is provided, it must be a $T \times n$ matrix of n phase components evaluated at the time points t.
t	A numeric vector of length T giving the time points at which the phase components are evaluated. This vector must start at 0 and end at 1.

`x`	A `Xclrv` object, obtained as a result of the function `X2Xclrv()`.
`...`	Further graphical parameters supplied to the internal `graphics::matplot()` function.

Details

Let $X(t)$ be a phase component defined on $t \in [0,1]$. centered log-ratio velocity (clrv) transformation is defined as

$$\texttt{clrv}(X)(t) := \log(DX(t)) - \int_0^1 \log(DX(s))\mathrm{d}s, \quad t \in [0,1],$$

where D is a differential operator.

Value

`X2Xclrv()` returns an object of class `Xclrv`, which is a $(T-1) \times n$ matrix of clrv transformed phase components evaluated over the intervals defined by the time points `t`.

`plot.Xclrv()` plots phase components after centered log-ratio velocity transformation.

References

Kang S. and Oh H.-S. (2026) “Multiview functional clustering using latent representations of phase and amplitude components,” *Unpublished Manuscript*.

See Also

`auc_sync()` and `fr_sync()` for time-synchronizing mappings.

Examples

```
t <- seq(0, 1, length.out = 100)
sync <- auc_sync(seoul_bike$Ytilde[,1:10], seoul_bike$x, t)
plot(X2Xclrv(sync))
```

`Xclrv2X` *Inverse Centered Log-Ratio Velocity Transformation*

Description

Inverse of the centered log-ratio velocity (clrv) transformation.

Usage

```
Xclrv2X(Xclrv, t, x)
```

Arguments

`Xclrv`	A $(T-1) \times n$ matrix of clrv transformed phase components evaluated over the intervals defined by the time points `t`.
`t`	A numeric vector of length T giving the time points at which the phase components are evaluated. This vector must start at 0 and end at 1.
`x`	A numeric vector of length m giving the observed time points.

Details

Let $X(t)$ be a phase component defined on $t \in [0, 1]$. centered log-ratio velocity (clrv) transformation is defined as

$$\texttt{clrv}(X)(t) := \log(DX(t)) - \int_0^1 \log(DX(s))\mathrm{d}s, \quad t \in [0, 1],$$

where D is a differential operator.

If $X : [0, 1] \to [T_1, T_2]$ for $T_1 < T_2$, the inverse of the clrv transformation is defined as

$$T_1 + (T_2 - T_1)\frac{\int_0^t \exp\{\texttt{clrv}(X)(s)\}\mathrm{d}s}{\int_0^1 \exp\{\texttt{clrv}(X)(s)\}\mathrm{d}s}, \quad t \in [0, 1].$$

Value

A $T \times n$ matrix of inverse-clrv transformed phase components evaluated at the time points `t`.

References

Kang S. and Oh H.-S. (2026) “Multiview functional clustering using latent representations of phase and amplitude components,” *Unpublished Manuscript*.

See Also

`X2Xclrv()` for clrv transformation.

Examples

```
t <- seq(0, 1, length.out = 100)
sync <- auc_sync(seoul_bike$Ytilde[,1:10], seoul_bike$x, t)
xclrv <- X2Xclrv(sync)
range(Xclrv2X(xclrv, t, seoul_bike$x) - sync$X)
```

Index