Package 'clustEff'

November 4, 2026

Type Package

Title Clusters of Effects Curves in Quantile Regression Models

Version 1.0.0

Description Clustering of curves based on a dissimilarity that combines the concordance of their shapes and their closeness, as proposed by Sottile and Adelfio (2019) <doi:10.1007/s00180-018-0817-8>. The method can be applied to the effects curves of quantile regression coefficients models fitted with 'qrcm', with univariate or multivariate responses, and to curves observed in time, as in functional data analysis. The package also implements the clustering of waveforms by trimmed k-means of functional principal component scores (Adelfio, Chiodi, D'Alessandro and Luzio, 2011).

License GPL-2

URL https://github.com/gianluca-sottile/clustEff

BugReports https://github.com/gianluca-sottile/clustEff/issues

Depends R (>= 4.1.0)

Imports cluster, graphics, grDevices, splines, stats

Suggests fda, knitr, qrcm (>= 3.0), rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Gianluca Sottile [aut, cre] (ORCID: <https://orcid.org/0000-0001-9347-7251>), Giada Adelfio [aut]

Maintainer Gianluca Sottile <gianluca.sottile@unipa.it>

Repository CRAN

Date/Publication 2026-11-03 23:20:02 UTC

Contents

clustEff 2
distshape 5
extract_curves 7
fpcac 9
fpca_bspline 11
opt.fpcac 12
plot.clustEff 13
plot.fpcac 14
summary.clustEff 15
summary.fpcac 15

Index **17**

`clustEff` *Clusters of Effects Curves*

Description

Hierarchical clustering of curves, such as the effects curves $\beta_j(p)$ of quantile regression coefficients models or curves observed in time, based on the shape-distance dissimilarity of Sottile and Adelfio (2019).

Usage

```
clustEff(
  curves,
  lower = NULL,
  upper = NULL,
  p = NULL,
  k = NULL,
  alpha = NULL,
  step = c("both", "shape", "distance"),
  deriv = 2,
  combine = c("product", "mean"),
  shape.tol = 1e-06,
  criterion = c("height", "pi.out", "dist.rel", "silhouette", "conf.int"),
  method = "ward.D2",
  diss = NULL,
  smooth = FALSE,
  df = NULL,
  conf.level = 0.9,
  stand = FALSE,
  legacy = FALSE,
  verbose = FALSE,
  ...
)
```

```
## S3 method for class 'clustEff'
print(x, ...)
```

Arguments

curves	a numeric matrix with one column for each curve and one row for each value of p; or an object returned by extract_curves(); or an object of class "iqr" (see qrcm::iqr()) or a list of such objects, whose effects curves are clustered.
lower, upper	optional matrices (same dimensions as curves) with the lower and upper pointwise confidence bands of the curves.
p	the grid on which the curves are evaluated (quantile orders, or times). By default (1:n)/n, or the grid of an "effects_curves" object.
k	the number of clusters: a single value, or a range c(kmin, kmax) in which the number of clusters is selected with criterion. The default is c(2, min(5, q - 1)) for q curves.
alpha	the quantile of the pointwise distances used by distshape(). The default is 0.25 when bands are available (effects curves) and 0.5 otherwise (waveforms), as suggested by Sottile and Adelfio (2019).
step, deriv, combine, shape.tol	arguments of distshape().
criterion	the criterion used to select the number of clusters when k is a range, see Details (default "height").
method	the agglomeration method of stats::hclust() (default "ward.D2").
diss	an optional dissimilarity (an object of class "dist", e.g. computed by distshape()); if supplied, alpha, step, deriv, combine and shape.tol are ignored.
smooth	logical: if TRUE, each curve (and its bands) is smoothed with stats::smooth.spline() before computing the dissimilarity (useful for noisy curves, as the shape component depends on derivatives).
df	degrees of freedom of the smoothing splines (default: chosen by generalized cross-validation).
conf.level	when the bands are not available, the bands of each cluster are the pointwise quantiles of order (1 - conf.level)/2 and (1 + conf.level)/2 of its curves.
stand	logical: standardize each curve (center and scale over the grid)?
legacy	logical: reproduce 'clustEff' 0.3.x (dissimilarity with legacy = TRUE in distshape(), alpha = 0.5, criterion "silhouette" unless specified). Remaining differences with 0.3.x: "height" (formerly cut.method = "length") searches only within the range k; with stand = TRUE also the bands are standardized; without bands the cluster bands use conf.level (0.3.x always used the quantiles 0.1 and 0.9).
verbose	logical: print the progress?
...	arguments of 'clustEff' 0.3.x that were removed (they give an informative error, see clustEff-defunct).
x	an object of class "clustEff".

Details

The algorithm (Sottile and Adelfio, 2019, Section 3.2) computes the dissimilarity matrix with `distshape()`, builds a dendrogram with `stats::hclust()`, selects the number of clusters l^* and computes the mean curve (and mean bands) of each cluster.

Criteria for the number of clusters l:

- `"height"` (default): l^* corresponds to the largest gap between consecutive heights of the dendrogram.
- `"pi.out"` (effects curves with bands): with $\bar{LB}_j(p)$ and $\bar{UB}_j(p)$ the mean bands of cluster j,

$$\pi_{out}(l) = l^{-1} \sum_{j=1}^{l} q_j^{-1} \sum_{i \in j} \int [1 - I\{\bar{LB}_j(p) \le \beta_i(p) \le \bar{UB}_j(p)\}]\, dp$$

  is the average proportion of the curves lying outside the mean bands of their cluster (Sottile and Adelfio, 2019, Section 3.1). It decreases with l; l^* is the elbow, i.e. the value at which the decrease slows down most.
- `"dist.rel"` (waveforms): $dist_{rel}(l) = l \, \max_j q_j^{-1} \sum_{i \in j} \int |\bar{\beta}_j(p) - \beta_i(p)|\, dp$ (Section 3.1), and l^* is the first number of clusters at which $dist_{rel}$ stops decreasing.
- `"silhouette"`: l^* maximizes the average silhouette width (default criterion of 'clustEff' 0.3.x).
- `"conf.int"`: criterion `"conf.int"` of 'clustEff' 0.3.x (coverage of the mean bands weighted by the silhouette widths).

The paper selects l^* minimizing $\pi_{out}(l) - \pi_{out}(l+1)$ or $dist_{rel}(l) - dist_{rel}(l+1)$, and writes π_{out} with the indicator of the curves inside the bands and a factor l. In simulations with 2 to 4 groups of effects curves these literal rules rarely found the true number of clusters, while the rules above did better and `"height"` was the most accurate; see the vignette.

Value

An object of class `"clustEff"`, a list with items:

`clusters`	the cluster of each curve (clusters are numbered by increasing size).
`k`	the number of clusters.
`p`, `curves`, `lower`, `upper`	the input (possibly smoothed or standardized).
`mean`, `mean.lower`, `mean.upper`	matrices of the mean curves and mean bands of each cluster.
`signif`	logical matrix: are the mean bands of each cluster away from zero at each p? (`NULL` without bands).
`diss`, `hclust`, `silhouette`	the dissimilarity, the dendrogram and the silhouette (see `cluster::silhouette()`).
`criterion`	the values of the criterion for each number of clusters.
`measures`	a data frame with, for each cluster, its size, the average area between the curves and the mean curve (eq. 2 of the paper), the correlation distance $\rho_{dist} = 1 -$ average pairwise correlation between the curves of the cluster (eq. 3), the average silhouette width and the average within-cluster dissimilarity.
`settings`	the options used.

References

Sottile, G. and Adelfio, G. (2019). Clusters of effects curves in quantile regression models. *Computational Statistics*, 34, 551-569. doi:10.1007/s0018001808178

See Also

`distshape()`, `extract_curves()`, `summary.clustEff()`, `plot.clustEff()`, `fpcac()`.

Examples

```
## Waveforms: three groups of noisy curves
set.seed(1234)
tt <- seq(0, 1, length.out = 100)
W <- cbind(sapply(1:5, function(i) sin(3 * pi * tt) + rnorm(100, 0, 0.1)),
           sapply(1:5, function(i) cos(3 * pi * tt) + rnorm(100, 0, 0.1)),
          sapply(1:5, function(i) sin(3 * pi * tt) * cos(pi * tt) + rnorm(100, 0, 0.1)))
fit <- clustEff(W, p = tt, k = c(2, 5), smooth = TRUE)
fit
summary(fit)
plot(fit)

## Effects curves with confidence bands: three groups of four curves
p <- seq(0.01, 0.99, 0.01)
B <- cbind(sapply(1:4, function(i) 1 + p + rnorm(1, 0, 0.05)),
           sapply(1:4, function(i) -1 + 2 * p^2 + rnorm(1, 0, 0.05)),
           sapply(1:4, function(i) 2 - 3 * p^3 + rnorm(1, 0, 0.05)))
fit2 <- clustEff(B, lower = B - 0.2, upper = B + 0.2, p = p, k = c(2, 5))
fit2$clusters
plot(fit2)
```

`distshape` *Shape-Distance Dissimilarity between Curves*

Description

Computes the dissimilarity between curves of Sottile and Adelfio (2019, eq. 1), which combines the concordance of their *shapes* (the signs of their second derivatives) and their *closeness* (their distance relative to the distances between all pairs of curves), pointwise on a grid.

Usage

```
distshape(
  curves,
  p = NULL,
  alpha = 0.5,
  step = c("both", "shape", "distance"),
  deriv = 2,
  combine = c("product", "mean"),
```

```
  d = NULL,
  shape.tol = 1e-06,
  legacy = FALSE,
  ...
)
```

Arguments

curves	a numeric matrix with one column for each curve and one row for each value of p, or an object returned by extract_curves().
p	the grid on which the curves are evaluated (quantile orders, or times). By default p = (1:n)/n, or the grid of an "effects_curves" object.
alpha	the order of the quantile of the pointwise distances between all pairs of curves that defines two curves as close (default 0.5). Sottile and Adelfio (2019) suggest 0.25 for effects curves and 0.5 for waveforms.
step	which components are used: "both" (default), "shape" or "distance".
deriv	the order of the derivative used for the shape component: 2 (default, convexity, as in the paper) or 1 (monotonicity).
combine	how the two components are combined when step = "both": "product" (default, eq. 1 of the paper: two curves are similar at p if they are concordant in shape *and* close) or "mean" (average of the two proportions, as in 'clustEff' 0.3.x).
d	an optional matrix of derivatives of order deriv of the curves (same dimensions as curves). If missing they are computed by interpolating splines, or taken from an "effects_curves" object (second derivatives).
shape.tol	derivatives smaller than shape.tol times the scale of the curve (the largest absolute value of the curve or of its derivative) are treated as zero, so that, e.g., two linear curves are concordant in shape.
legacy	logical: if TRUE, the dissimilarity of 'clustEff' 0.3.x is computed (first derivative, combine = "mean", grid (1:n)/n, equal weights); the arguments p, deriv, combine, d and shape.tol are then ignored.
...	arguments of 'clustEff' 0.3.x that were removed (they give an informative error).

Details

For two curves β_i and $\beta_{i'}$,

$$d_{shape}(p) = I\{sign(\beta_i''(p)) = sign(\beta_{i'}''(p))\}, \qquad d_{distance}(p) = I\{|\beta_i(p) - \beta_{i'}(p)| \le f_\alpha(p)\},$$

where $f_\alpha(p)$ is the α-quantile of the distances $|\beta_j(p) - \beta_{j'}(p)|$ between all pairs of curves at p. The dissimilarity is

$$d(i, i') = 1 - \int d_{shape}(p)\, d_{distance}(p)\, dp,$$

where the integral (normalized by the length of the interval) is computed with the trapezoidal rule on the grid p. The computations are vectorized over the pairs of curves and need $O(q^2)$ memory for q curves.

The paper defines the shape concordance as $sign(\beta_i'') \times sign(\beta_{i'}'') = 1$; here derivatives below `shape.tol` are set to zero and two zero derivatives are concordant, which avoids that, e.g., two linear curves are always discordant. The shape component alone is a Hamming distance between the sign patterns; the distance component, based on a threshold, does not satisfy the triangle inequality in general. With only two curves the distance component is always 1, and `step = "both"` reduces to `"shape"`.

Value

An object of class `"dist"` (see `stats::dist()`) with attributes `alpha`, `step`, `deriv`, `combine` and `legacy`.

References

Sottile, G. and Adelfio, G. (2019). Clusters of effects curves in quantile regression models. *Computational Statistics*, 34, 551-569. doi:10.1007/s0018001808178

See Also

`clustEff()`

Examples

```
tt <- seq(0, 1, length.out = 100)
curves <- cbind(a1 = sin(3 * pi * tt), a2 = sin(3 * pi * tt) + 0.1,
                b1 = cos(3 * pi * tt), b2 = cos(3 * pi * tt) - 0.1)
distshape(curves, p = tt)
distshape(curves, p = tt, step = "shape")
```

---

extract_curves *Effects Curves of Quantile Regression Coefficients Models*

Description

Extracts the estimated quantile regression coefficient functions $\hat{\beta}_j(p)$ (the *effects curves*), their pointwise confidence bands and their second derivatives from one or more models fitted with `qrcm::iqr()`, in the format used by `clustEff()`.

Usage

```
extract_curves(object, p = seq(0.01, 0.99, 0.01), which, level = 0.95)

## S3 method for class 'effects_curves'
print(x, ...)
```

Arguments

`object`	an object of class `"iqr"`, or a (possibly named) list of such objects, for example one model for each of several responses.
`p`	the grid of quantile orders on which the curves are evaluated.
`which`	the coefficients to extract: a vector of names or of indices of the rows of the coefficient matrix. By default all of them except the intercept.
`level`	confidence level of the pointwise bands (default 0.95).
`x`	an object of class `"effects_curves"`.
`...`	not used.

Details

The bands are $\hat{\beta}_j(p) \pm z_{(1+level)/2}\, se\{\hat{\beta}_j(p)\}$, with standard errors from `predict(object, type = "beta", se = TRUE)`. The second derivatives $\hat{\beta}_j''(p)$, used by the shape component of the dissimilarity of `distshape()`, are computed by central differences of the smooth fitted functions (step 10^{-3}; exact up to rounding errors when $\beta(p)$ is a polynomial of degree up to 3 in p, and very accurate otherwise, unlike derivatives of interpolating splines of the estimates).

With a list of models (e.g., the same covariates for different responses, as in the multivariate case of Sottile and Adelfio, 2019), the curves are named `"model:coefficient"`, where `model` is the name of the list element (or its position).

Value

An object of class `"effects_curves"`, a list with items `curves`, `lower`, `upper` and `d2` (matrices with one row for each value of `p` and one column for each curve), `p`, `level` and `info` (a data frame with the model and the coefficient of each curve).

References

Sottile, G. and Adelfio, G. (2019). Clusters of effects curves in quantile regression models. *Computational Statistics*, 34, 551-569. doi:10.1007/s0018001808178

See Also

`clustEff()`, `qrcm::iqr()`

Examples

```
if (requireNamespace("qrcm", quietly = TRUE)) {
  set.seed(1)
  n <- 300
  x <- runif(n, 0, 5)
  # five responses with three different effects of x
  slopes <- list(function(u) 0.5 + u, function(u) 0.5 + u, function(u) -1 + 0.5 * u,
                 function(u) -1 + 0.5 * u, function(u) 2 * u^2)
  fits <- lapply(slopes, function(b) {
    u <- runif(n)
    y <- qnorm(u) + b(u) * x
```

```
    qrcm::iqr(y ~ x, formula.p = ~ I(p) + I(p^2))
  })
  names(fits) <- paste0("y", 1:5)
  ec <- extract_curves(fits, which = "x")
  ec
  fit <- clustEff(ec, k = 3)
  fit$clusters
}
```

---

fpcac *Functional PCA-based Clustering of Curves (FPCAC)*

Description

Clusters curves (e.g., seismic waveforms) by a trimmed k-means applied to their functional principal component scores (Adelfio et al., 2011).

Usage

```
fpcac(
  curves,
  t = NULL,
  K = 2,
  nbasis = 5,
  norder = 3,
  nharmonics = 3,
  alpha = 0,
  nstart = 30,
  iter.max = 25,
  init = c("kmeans++", "random"),
  conf.level = 0.9,
  verbose = FALSE,
  ...
)

## S3 method for class 'fpcac'
print(x, ...)
```

Arguments

curves	a numeric matrix with one column for each curve and one row for each time point.
t	the time points (default: equally spaced on [0, 1]).
K	the number of clusters.
nbasis, norder	number and order of the B-spline basis functions.
nharmonics	number of functional principal components.

alpha	trimming proportion: the N - floor(N * (1 - alpha)) curves farthest from their centers are not assigned to any cluster (label 0).
nstart	number of random starts.
iter.max	maximum number of concentration steps for each start.
init	initialization of the centers: "kmeans++" (default) or "random" (as in 'clustEff' 0.3.x).
conf.level	level of the pointwise bands of the curves of each cluster.
verbose	logical: print the objective function of each start?
...	arguments of 'clustEff' 0.3.x that were removed (informative error).
x	an object of class "fpcac".

Details

Each curve is represented on a B-spline basis and summarized by the scores of its first nharmonics functional principal components (see fpca_bspline()). The trimmed k-means of García-Escudero and Gordaliza finds K centers minimizing the mean squared distance of the closest $\lfloor N(1-\alpha) \rfloor$ curves to their nearest center. Use set.seed() for reproducible results.

Value

An object of class "fpcac", a list with items clusters (0 = trimmed), K, centers, obj.function, radius, scores, fpca (the output of fpca_bspline()), curves, t, mean, mean.lower, mean.upper (mean curve and pointwise bands of each cluster), mean.dist (correlation distance of each curve from the mean of its cluster) and call.

References

Adelfio, G., Chiodi, M., D'Alessandro, A. and Luzio, D. (2011). FPCA algorithm for waveform clustering. *Journal of Communication and Computer*, 8, 494-502.

García-Escudero, L. A. and Gordaliza, A. (2005). A proposal for robust curve clustering. *Journal of Classification*, 22, 185-201. doi:10.1007/s0035700500138

See Also

opt.fpcac(), clustEff()

Examples

```
set.seed(1)
tt <- seq(0, 1, length.out = 100)
X <- cbind(sapply(1:10, function(i) sin(3 * pi * tt) + rnorm(100, 0, 0.2)),
           sapply(1:10, function(i) cos(3 * pi * tt) + rnorm(100, 0, 0.2)),
           sapply(1:2, function(i) rnorm(100, 0, 0.2)))          # two outliers
fit <- fpcac(X, t = tt, K = 2, nbasis = 10, alpha = 0.1)
fit
plot(fit)
```

`fpca_bspline` *Functional Principal Components Analysis of Curves on a B-spline Basis*

Description

Represents each curve on a B-spline basis by least squares and computes the functional principal components (harmonics) and the scores, as in Ramsay and Silverman (2005, Chapter 8). It is used by `fpcac()`.

Usage

```
fpca_bspline(
  curves,
  t = NULL,
  nbasis = 5,
  norder = 3,
  nharmonics = 3,
  lambda = 1e-08
)
```

Arguments

`curves`	a numeric matrix with one column for each curve and one row for each time point.
`t`	the time points (default: equally spaced on [0, 1]).
`nbasis`	number of B-spline basis functions.
`norder`	order of the B-splines (degree + 1; default 3, quadratic).
`nharmonics`	number of principal components.
`lambda`	ridge penalty on the basis coefficients (default 1e-8, for numerical stability).

Value

A list with items `scores` (curves x harmonics), `values` (the eigenvalues), `varprop` (proportion of variance explained), `coefs` (basis coefficients of the curves), `harmonics` (basis coefficients of the harmonics), `mean` (basis coefficients of the mean curve), `fitted` (the smoothed curves on `t`), `basis` (the basis matrix on `t`) and `gram` (the Gram matrix of the basis).

References

Ramsay, J. O. and Silverman, B. W. (2005). *Functional Data Analysis*, 2nd ed. Springer.

Examples

```
tt <- seq(0, 1, length.out = 100)
X <- sapply(1:20, function(i) sin(2 * pi * tt + runif(1)) + rnorm(100, 0, 0.1))
f <- fpca_bspline(X, tt, nbasis = 10, nharmonics = 2)
f$varprop
```

`opt.fpcac` *Number of Clusters for FPCAC*

Description

Fits `fpcac()` for $K = 1, \ldots, k.max$ and computes the average silhouette width or the total within-cluster sum of squares.

Usage

```
opt.fpcac(
  curves,
  t = NULL,
  k.max = 5,
  criterion = c("silhouette", "wss"),
  nbasis = 5,
  norder = 3,
  nharmonics = 3,
  alpha = 0,
  nstart = 30,
  iter.max = 25,
  init = c("kmeans++", "random"),
  diss = NULL,
  ...
)

## S3 method for class 'opt.fpcac'
print(x, ...)

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

Arguments

`curves`, `t`, `nbasis`, `norder`, `nharmonics`, `alpha`, `nstart`, `iter.max`, `init`	arguments of `fpcac()`.
`k.max`	maximum number of clusters.
`criterion`	`"silhouette"` (default: K maximizing the average silhouette width, $K \geq 2$) or `"wss"` (K with the largest decrease of the within-cluster sum of squares from $K-1$, as in 'clustEff' 0.3.x).
`diss`	an optional dissimilarity between the curves (class `"dist"`). By default the Euclidean distance between the vectors of scores, consistent with the trimmed k-means (`'clustEff'` 0.3.x used the correlation distance $\sqrt{2(1-r)}$ between the vectors of scores, which is degenerate with one or two harmonics).
`...`	arguments of 'clustEff' 0.3.x that were removed (informative error).
`x`	an object of class `"opt.fpcac"`.

Value

An object of class "opt.fpcac", a list with items criterion (the values for each K), K.opt, clusters (matrix of the clusters for each K), fits (the fpcac() fits) and criterion.name. Trimmed curves are excluded from the computation of the criteria.

See Also

fpcac()

Examples

```
set.seed(1)
tt <- seq(0, 1, length.out = 100)
X <- cbind(sapply(1:8, function(i) sin(3 * pi * tt) + rnorm(100, 0, 0.2)),
           sapply(1:8, function(i) cos(3 * pi * tt) + rnorm(100, 0, 0.2)),
          sapply(1:8, function(i) sin(3 * pi * tt) * cos(pi * tt) + rnorm(100, 0, 0.2)))
o <- opt.fpcac(X, t = tt, k.max = 5, nbasis = 10)
o$K.opt
plot(o)
```

---

plot.clustEff *Plot Clusters of Curves*

Description

Plot Clusters of Curves

Usage

```
## S3 method for class 'clustEff'
plot(
  x,
  type = c("clusters", "dendrogram", "silhouette", "dissimilarity", "criterion"),
  which,
  polygon = TRUE,
  col,
  ...
)
```

Arguments

x	an object of class "clustEff".
type	what to plot: "clusters" (mean curves with their bands, or the curves of one cluster if which has length one), "dendrogram", "silhouette" (boxplots of the silhouette widths by cluster), "dissimilarity" (boxplots of the within-cluster dissimilarities) or "criterion" (the criterion against the number of clusters).

`which`	the clusters to plot (for `type = "clusters"`).
`polygon`	logical: draw the bands as shaded areas (otherwise dashed lines)?
`col`	colours of the clusters.
`...`	further graphical parameters (`xlab`, `ylab`, `main`, `ylim`, `lwd`).

Value

No return value, called for its side effects.

See Also

`clustEff()`

`plot.fpcac` *Plot an FPCAC Fit*

Description

Plot an FPCAC Fit

Usage

```
## S3 method for class 'fpcac'
plot(x, which, polygon = TRUE, col, ...)
```

Arguments

`x`	an object of class `"fpcac"`.
`which`	the clusters to plot. With a single cluster, its curves are drawn.
`polygon`	logical: draw the bands as shaded areas?
`col`	colours of the clusters.
`...`	further graphical parameters (`xlab`, `ylab`, `main`, `ylim`, `lwd`).

Value

No return value, called for its side effects.

See Also

`fpcac()`

summary.clustEff *Summarize Clusters of Curves*

Description

Summarize Clusters of Curves

Usage

```
## S3 method for class 'clustEff'
summary(object, ...)

## S3 method for class 'summary.clustEff'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

object	an object of class "clustEff".
...	not used.
x	an object of class "summary.clustEff".
digits	number of significant digits.

Value

An object of class "summary.clustEff": a list with the number of clusters, the number of curves and grid points, the options used, the clustering table, the within-cluster measures (with their weighted average) and the criterion used to select the number of clusters.

See Also

clustEff()

summary.fpcac *Summarize an FPCAC Fit*

Description

Summarize an FPCAC Fit

Usage

```
## S3 method for class 'fpcac'
summary(object, ...)

## S3 method for class 'summary.fpcac'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`object`	an object of class `"fpcac"`.
`...`	not used.
`x`	an object of class `"summary.fpcac"`.
`digits`	number of significant digits.

Value

An object of class `"summary.fpcac"` with the clustering table, the number of trimmed curves, the proportion of variance of each harmonic and, for each cluster, the average correlation distance of its curves from the mean curve.

See Also

`fpcac()`

Index

clustEff, 2
clustEff(), *7, 8*, *10*, *14, 15*
clustEff-defunct, *3*
cluster::silhouette(), *4*

distshape, 5
distshape(), *3–5*, *8*

extract_curves, 7
extract_curves(), *3*, *5, 6*

fpca_bspline, 11
fpca_bspline(), *10*
fpcac, 9
fpcac(), *5*, *11–14*, *16*

opt.fpcac, 12
opt.fpcac(), *10*

plot.clustEff, 13
plot.clustEff(), *5*
plot.fpcac, 14
plot.opt.fpcac *(*opt.fpcac*)*, 12
print.clustEff *(*clustEff*)*, 2
print.effects_curves *(*extract_curves*)*, 7
print.fpcac *(*fpcac*)*, 9
print.opt.fpcac *(*opt.fpcac*)*, 12
print.summary.clustEff
*(*summary.clustEff*)*, 15
print.summary.fpcac *(*summary.fpcac*)*, 15

qrcm::iqr(), *3*, *7, 8*

set.seed(), *10*
stats::dist(), *7*
stats::hclust(), *3, 4*
stats::smooth.spline(), *3*
summary.clustEff, 15
summary.clustEff(), *5*
summary.fpcac, 15