Package 'TFunHDDC'

June 4, 2023

Type Package

Title Clustering of Functional Data via Mixtures of t-Distributions

Version 1.0.1

Date 2023-06-01

Depends fda.usc, R (>= 3.5.0)

Imports tclust, stringr, MASS, fda

Description Extension of 'funHDDC' Schmutz et al. (2018) <doi:10.1007/s00180-020-00958-4> for cases including outliers by fitting t-distributions for robust groups. 'TFunHDDC' can cluster univariate or multivariate data produced by the 'fda' package for data using a b-splines or Fourier basis.

License GPL-3

Encoding UTF-8

LazyData true

NeedsCompilation yes

Repository CRAN

Author Cristina Anton [aut, cre],
Iain Smith [aut],
Malcolm Nielsen [aut],
Jeffrey Andrews [ctb],
Jaymeson Wickins [ctb],
Nicholas Boers [ctb],
Paul McNicholas [ctb],
Amandine Schmutz [ctb],
Julien Jacques [ctb],
Charles Bouveyron [ctb]

Maintainer Cristina Anton <popescuc@macewan.ca>

Date/Publication 2023-06-04 12:20:13 UTC

R topics documented:

`contaminatedTriangles` *contaminatedTriangles*

Description

Simulated triangle data produced as an example of mild contaminated data with behavioural outliers.

Value

`fd`	A functional data object representing the fitted triangle data.
`groupd`	Group classifications for each curve as a ordinary behavioral group (1,2,3, or 4) or outlier (4 as outliers to group 1 and 6 as outliers for group 3).

Author(s)

Cristina Anton and Iain Smith

References

- Cristina Anton, Iain Smith Model-based clustering of functional data via mixtures of t distributions. Advances in Data Analysis and Classification (to appear).

`fitNOxBenchmark` *fitNOxBenchmark*

Description

Extract NOx data from fda.usc

Usage

```
fitNOxBenchmark(nbasis=15)
```

Arguments

nbasis	The number of basis functions to fit to the simulated data.

Details

Open NOx data from the poblenou data set of fda.usc. Fit the data to a given number of basis functions and adjust classes for festive days into just weekdays and weekends.

Value

fd	A functional data object representing the fitted NOx data.
groupd	Group classifications for each curve as a curve representing a weekday or weekend/festive day.

Author(s)

Cristina Anton and Iain Smith

References

- Febrero-Bande M, Galeano P, Gonz~alez-Manteiga W (2008) Outlier detection in functional data by depth measures, with application to identify abnormal nox levels. Environmetrics 19:331-345. <//doi.org/10.1002/env.878>.

- Cristina Anton, Iain Smith Model-based clustering of functional data via mixtures of t distributions. Advances in Data Analysis and Classification (to appear).

See Also

plotNOx

Examples

```
# Univariate Contaminated Data
data1<-fitNOxBenchmark(15)
plotNOx(data1)
```

---

genModelFD *genModelFD*

Description

Generate functional data with coefficients distributed according to a finite mixture of contamined normal distributions such that for the *k*th cluster we have the multivariate contaminated normal distribution with density

$$f(\gamma_i;\theta_k) = \alpha_k\phi(\gamma_i;\mu_k,\Sigma_k) + (1-\alpha_k)\phi(\gamma_i;\mu_k,\eta_k\Sigma_k)$$

where $\alpha_k \in (0.5, 1)$ represents the proportion of uncontaminated data, $\eta_k > 1$ is the inflation coefficient due to outliers, and $\phi(\gamma_i;\mu_k,\Sigma_k)$ is the density for the multivariate normal distribution $N(\mu_k,\Sigma_k)$.

Usage

```
genModelFD(ncurves=1000, nsplines=35, alpha=c(0.9,0.9,0.9),
           eta=c(10, 5, 15))
```

Arguments

ncurves	The number of curves total for the simulation.
nsplines	The number of splines to fit to the simulated data.
alpha	The proportion of uncontaminated data in each group.
eta	The inflation coefficient that measures the increase in variability due to the outliers.

Details

The data are generate from the model $FCLM[a_k, b_k, \mathbf{Q}_k, d_k, \alpha_k, \eta_k]$. The number of clusters is fixed to $K = 3$ and the mixing proportions are equal $\pi_1 = \pi_2 = \pi_3 = 1/3$. We consider the following values of the parameters

Group 1:$d = 5$, $a = 150$, $b = 5$, $\mu = (1, 0, 50, 100, 0, \ldots, 0)$

Group 2: $d = 20$, $a = 15$, $b = 8$, $\mu = (0, 0, 80, 0, 40, 2, 0, \ldots, 0)$

Group 3: $d = 10$, $a = 30$, $b = 10$, $\mu = (0, \ldots, 0, 20, 0, 80, 0, 0, 100)$,

where d is the intrinsic dimension of the subgroups, μ is the mean vector of size 70, a is the values of the d-first diagonal elements of $\mathbf{D}$, and b the value of the last $70 - d$- elements. Curves as smoothed using 35 Fourier basis functions.

Value

fd	A functional data object representing the simulated data.
groupd	Group classifications for each curve.

Author(s)

Cristina Anton and Iain Smith

References

- Amovin-Assagba M, Gannaz I, Jacques J (2022) Outlier detection in multivariate functional data through a contaminated mixture model. Comput Stat Data Anal 174. - Cristina Anton, Iain Smith Model-based clustering of functional data via mixtures of t distributions. Advances in Data Analysis and Classification (to appear).

Examples

```
# Univariate Contaminated Data
data <- genModelFD(ncurves=300, nsplines=35, alpha=c(0.9,0.9,0.9),
                   eta=c(10, 7, 17))
plot(data$fd, col = data$groupd)
clm <- data$groupd
```

genTriangles *genTriangles*

Description

Generate contaminated triangle data. Groups 1, 2, 3, and 4 are separable over the two dimensions of functional data. Groups 5 and 6 contain the contaminated curves of groups 1 and 3 respectively.

Usage

```
genTriangles()
```

Details

Group 1:

$X_1(t) = U + (0.6 - U)H_1(t) + \epsilon_1(t)$

$X_2(t) = U + (0.5 - U)H_1(t) + \epsilon_1(t)$

Contaminated $X_1(t) = \sin(t) + (0.6 - U)H_1(t) + \epsilon_2(t)$

Contaminated $X_2(t) = \sin(t) + (0.5 - U)H_1(t) + \epsilon_2(t)$

Group 2:

$X_1(t) = U + (0.6 - U)H_2(t) + \epsilon_1(t)$

$X_2(t) = U + (0.5 - U)H_2(t) + \epsilon_1(t)$

Group 3:

$X_1(t) = U + (0.5 - U)H_1(t) + \epsilon_1(t)$

$X_2(t) = U + (0.6 - U)H_2(t) + \epsilon_1(t)$

Contaminated $X_1(t) = \sin(t) + (0.5 - U)H_1(t) + \epsilon_3(t)$

Contaminated $X_2(t) = \sin(t) + (0.6 - U)H_2(t) + \epsilon_3(t)$

Group 4:

$X_1(t) = U + (0.5 - U)H_2(t) + \epsilon_1(t)$

$X_2(t) = U + (0.6 - U)H_1(t) + \epsilon_1(t)$. Here $t \in [1, 21]$, $H_1(t) = (6 - |t - 7|)_+$, and $H_2(t) = (6 - |t - 15|)_+$, with $(\cdot)_+$ representing the positive part. $U \sim \mathcal{U}(0, 0.1)$, and $\epsilon_1(t) \sim N(0, 0.5)$, $\epsilon_2(t) \sim N(0, 2)$, $\epsilon_3(t) \sim Cauchy(0, 4)$ are mutually independent white noises and independent of U. We simulate 100 curves for each group, groups 1 and 3 consisting of 80 ordinary curves and 20 contaminated curves. Curves are smoothed using a 25 cubic B-spline basis.

Value

`fd`	List of functional data objects representing the two dimensions of triangle data.
`groupd`	Group classification for each curve

Author(s)

Cristina Anton and Iain Smith

References

- C.Bouveyron and J.Jacques (2011), Model-based Clustering of Time Series in Group-specific Functional Subspaces, Advances in Data Analysis and Classification, vol. 5 (4), pp. 281-300, <doi:10.1007/s11634-011-0095-6>

- Schmutz A, Jacques J, Bouveyron C, et al (2020) Clustering multivariate functional data in group-specific functional subspaces. Comput Stat 35:1101-1131

- Cristina Anton, Iain Smith Model-based clustering of functional data via mixtures of t distributions. Advances in Data Analysis and Classification (to appear).

See Also

plotTriangles

Examples

```
# Multivariate Contaminated Triangles
conTrig <- genTriangles()
cls = conTrig$groupd
plotTriangles(conTrig)
```

---

plotNOx *plotNOx*

Description

Plot data returned by fitNOxBenchmark as lines coloured according to the assigned clusters.

Usage

```
plotNOx(fdn)
```

Arguments

fdn	List with an element fd given the functional data, and an element $groupd$ given the classes; usually returned from the function fitNOxBenchmark.

Value

No return value, used for side effects.

Author(s)

Cristina Anton and Iain Smith

See Also

fitNOxBenchmark

Examples

```
# Univariate Contaminated Data
data1<-fitNOxBenchmark(15)
plotNOx(data1)
```

---

plotTriangles *plotTriangles*

Description

Plot data returned by genTriangles as lines coloured according to the assigned clusters.

Usage

```
plotTriangles(fdt)
```

Arguments

fdt	List with an element fd given the functional data, and an element $groupd$ given the classes, as returned from the function genTriangles.

Value

No return value, used for side effects.

Author(s)

Cristina Anton and Iain Smith

See Also

genTriangles

Examples

```
conTrig <- genTriangles()
plotTriangles(conTrig)
```

predict.tfunHDDC *predict.tfunHDDC: Predicting Function for t-funHDDC Objects*

Description

Provides the matrix of classification probabilities and the classification vector for inputted observations assuming the model provided by the tfunHDDC object.

Usage

```
## S3 method for class 'tfunHDDC'
predict(object, data=NULL, ...)
```

Arguments

object	An object of class tfunHDDC
data	Data frame (univariate funtional data) or a list (multivariate functional data) of new observations on the same variables used in the fitting of the tfunHDDC object. If NULL, then the observations used in the fitting of the tfunHDDC object are inputted.
...	Arguments to be passed to other functions.

Value

t	Matrix of classification probabilities
class	Vector of maximum a posteriori classifications

Author(s)

Cristina Anton, Iain Smith

References

-Andrews JL, McNicholas PD (2012) Model-based clustering, classi

cation, and discriminant analysis via mixtures of multivariate t-distributions: The teigen family. Stat Comput 22:10211029. <doi.org/10.1007/ s11222-011-9272-x>

-Andrews JL, Wickins JR, Boers NM, et al (2018) An R package for modelbased clustering and classi

cation via the multivariate t distribution. Journal of Statistical Software 83(7):1-32

- Cristina Anton, Iain Smith Model-based clustering of functional data via mixtures of t distributions. Advances in Data Analysis and Classification (to appear).

See Also

tfunHDDC

Examples

```
set.seed(1027)
#simulataed univariate data

data = genModelFD(ncurves=300, nsplines=35, alpha=c(0.9,0.9,0.9),
                  eta=c(10, 7, 17))

plot(data$fd, col = data$groupd)

clm = data$groupd

model1=c("AkjBkQkDk", "AkjBQkDk", "AkBkQkDk", "ABkQkDk", "AkBQkDk", "ABQkDk")

####################classification example with predictions

training=c(1:50,101:150, 201:250)

test=c(51:100,151:200, 251:300)

known1=clm[training]

t4<-tfunHDDC(data$fd[training],K=3,threshold=0.2,init="kmeans",nb.rep=1,
             dfconstr="no", dfupdate="numeric", model=model1[1],known=known1,
             itermax = 10)

if (!is.null(t4$class)) {
table(clm[training], t4$class)

p1<-predict.tfunHDDC(t4,data$fd[test] )

if (!is.null(p1$class)) table(clm[test], p1$class)
}

############################NOX data

data1=fitNOxBenchmark(15)

plotNOx(data1)

###example for prediction

training=c(1:50)

test=c(51:115)

known1=data1$groupd[training]

t1<-tfunHDDC(data1$fd[training],K=2,threshold=0.6,init="kmeans",nb.rep=10,
             dfconstr="no", model=c("AkjBkQkDk", "AkjBQkDk", "AkBkQkDk",
             "ABkQkDk", "AkBQkDk", "ABQkDk"),known=known1)
```

```
if (!is.null(t1$class)) {
table(data1$groupd[training], t1$class)

p1<-predict.tfunHDDC(t1,data1$fd[test] )

if (!is.null(p1$class)) table(data1$groupd[test], p1$class)
}
```

`tfunHDDC` *tfunHDDC: Function for Model-Based Clustering of Functional Data with Outliers Using the t-Distribution.*

Description

tfunHDDC is an adaptation of funHDDC (Schmutz et al., 2018) that uses t-distributions for robust clustering in the presence of outliers.

Usage

```
tfunHDDC(data, K=1:10, model="AkjBkQkDk", known=NULL,dfstart=50, dfupdate="approx",
           dfconstr="no", threshold=0.1, itermax=200, eps=1e-6, init='random',
           criterion="bic", d_select="Cattell", init.vector=NULL,
          show=TRUE, mini.nb=c(5, 10), min.individuals=2, mc.cores=1, nb.rep=2,
           keepAllRes=TRUE, kmeans.control = list(), d_max=100, d_range=2,
           verbose = TRUE)
```

Arguments

`data`	In the univariate case: a functional data object produced by the fda package. In the multivariate case: a list of functional data objects.
`K`	The number of clusters or list of clusters to try, for example K=2:10.
`dfstart`	The df (degrees of freedom) to which we initialize the t-distribution.
`dfupdate`	Either "numeric", or "approx". The default is "approx" indicating a closed form approximation be used. Alternatively, "numeric" can be specified which makes use of `uniroot`.
`dfconstr`	"yes" when df (degrees of freedom) for the t-distribution should be the same between all clusters; "no" when df may be different between clusters.
`model`	The chosen model among 'AkjBkQkDk', 'AkjBQkDk', 'AkBkQkDk', 'ABkQkDk', 'AkBQkDk', 'ABQkDk'. 'AkjBkQkDk' is the default. We can test multiple models at the same time with the command c(). For example c("AkjBkQkDk","AkjBQkDk").
`threshold`	The threshold of the Cattell' scree-test used for selecting the group-specific intrinsic dimensions.
`known`	A vector of known classifications that can be numeric or `NA`. It is optional for clustering. For classification, curves with unknown classification should be given the value `NA` within `known` (see the examples below). Must be the same length as the number of curves in the data set.

`itermax` The maximum number of iterations.

`eps` The threshold of the convergence criterion.

`init` A character string. It is the way to initialize the EM algorithm. There are five ways of initialization: “kmeans” (default), “param”, “random”, “mini-em”, “vector”, or "tkmeans". See details for more information. It can also be directly initialized with a vector containing the prior classes of the observations.

`criterion` The criterion used for model selection: bic (default) or icl.

`d_select` “Cattell” (default), “BIC”, or "grid". This parameter selects which method to use to select the intrinsic dimensions of subgroups. "grid" will select d based on thecriterion value after running each combination of d1, d2, ..., dK for the groups. d used for each group is based on the values for d_range. "grid" will only work for a single value of K (not a list). See details for more information.

`init.vector` A vector of integers or factors. It is a user-given initialization. It should be of the same length as of the data. Only used when init="vector".

`show` Use show = FALSE to settle off the informations that may be printed.

`mini.nb` A vector of integers of length two. This parameter is used in the “mini-em” initialization. The first integer sets how many times the algorithm is repeated; the second sets the maximum number of iterations the algorithm will do each time. For example, if init=“mini-em” and mini.nb=c(5,10), the algorithm wil be launched 5 times, doing each time 10 iterations; finally the algorithm will begin with the initialization that maximizes the log-likelihood.

`min.individuals` This parameter is used to control for the minimum population of a class. If the population of a class becomes stricly inferior to ’min.individuals’ then the algorithm stops and gives the message: ’pop<min.indiv.’. Here the meaning of "population of a class" is the sum of its posterior probabilities. The value of ’min.individuals’ cannot be lower than 2.

`mc.cores` Positive integer, default is 1. If mc.cores>1, then parallel computing is used, using mc.cores cores. Warning for Windows users only: the parallel computing can sometimes be slower than using one single core (due to how parLapply works).

`nb.rep` A positive integer (default is 1 for kmeans initialization and 20 for random initialization). Each estimation (i.e. combination of (model, K, threshold)) is repeated nb.rep times and only the estimation with the highest log-likelihood is kept.

`keepAllRes` Logical. Should the results of all runs be kept? If so, an argument all_results is created in the results. Default is TRUE.

`kmeans.control` A list. The elements of this list should match the parameters of the `kmeans` initialization (see kmeans help for details). The parameters are “iter.max”, “nstart” and “algorithm”. "alpha" is an added parameter for the `tkmeans` initialization (see tkmeans help for details)

`d_max` A positive integer. The maximum number of dimensions to be computed. Default is 100. It means that the instrinsic dimension of any cluster cannot be larger than d_max. It quickens a lot the algorithm for datasets with a large number of variables (e.g. thousands).

d_range	Vector of values to use for the intrinsic dimension for each group when d_select="grid".
verbose	Whether to print progress and approximate timing information as tfunHDDC executes. TRUE (default when running in serial) or FALSE (default when running parallel).

Details

If we choose init="random", the algorithm is run 20 times with the same model options and the solution which maximises the log-likelihood is printed. This explains why sometimes with this initialization it runs a bit slower than with 'kmeans' initialization.

If the warning message: "In tfunHDDC(...) : All models diverged" is printed, it means that the algorithm found less classes that the chosen number (parameter K). Because the EM algorithm is used, it could be because of a bad initialization of the EM algorithm. So we have to restart the algorithm multiple times in order to check if with a new initialization of the EM algorithm the model converges, or if there is no solution with the chosen number K.

The different initializations are:

"mini-em": it is an initialization strategy for which the classes are randomly initialized and the EM algorithm is run for several iterations. This action is repetead a few times (the default is 5 iterations and 10 times). At the end, the initialization chosen is the one which maximise the log-likelihood (see mini.nb for more information about its parameters).

"random": the classes are randomly given using a multinomial distribution

"kmeans": the classes are initialized using the kmeans function (with algorithm="Hartigan-Wong"; nstart=4; iter.max=50); note that the user can use his own arguments for kmeans using the dot-dot-dot argument

"tkmeans": the classes are initialized using the tkmeans function (with same default initialization as kmeans); note that the user can use his own arguments for tkmeans using the dot-dot-dot argument

A prior class "vector": It can also be directly initialized with a vector containing the prior classes of the observations. To do so use init="vector" and provide the vector in the argument init.vector.

Note that the BIC criterion used in this function is to be maximized and is defined as 2*LL-k*log(n) where LL is the log-likelihood, k is the number of parameters and n is the number of observations.

There are three methods for selecting the intrinsic dimension using d_select:

"Cattell": Runs a Cattell's scree test to approximate the intrinsic dimension that yields the greatest improvement in clustering.

"BIC": At each iteration we tests each value for each group's intrinsic dimension and sets the intrinsic dimension that yields the best BIC.

"grid": Runs every combination of hyperparameters (eg. K=2, threshold = 0.05, model = ...) for every combination of intrinsic dimensions that can be set with the given d_range (with K = 2 and d_range = c(2, 10) it would set (2,2), (2, 10), (10, 2), and (10, 10)). Due to the sharp increase in test cases it is recommended that this mode is run in parallel if possible. Doing an intial short run to approximate the timing with verbose = TRUE is suggested as well.

Value

d	The number of dimensions for each cluster.
a	Values of parameter a for each cluster.

b	Values of parameter b for each cluster.
mu	The mean of each cluster in the original space.
prop	The proportion of individuals in each cluster.
loglik	The maximum of log-likelihood.
loglik_all	The log-likelihood at each iteration.
posterior	The posterior probability for each individual to belong to each cluster.
class	The clustering partition.
BIC	The BIC value.
ICL	The ICL value.
complexity	the number of parameters that are estimated.
all_results	if multiple number of clusters or models are considered, results for each model are stored here
nux	Values for the degrees of freedom of the t-distributions for each group.

Author(s)

Cristina Anton, Iain Smith, and Malcolm Nielsen

References

- Andrews JL and McNicholas PD. "Model-based clustering, classification, and discriminant analysis with the multivariate *t*-distribution: The *t*EIGEN family" *Statistics and Computing* 22(5), 1021–1029.

- Andrews JL, McNicholas PD, and Subedi S (2011) "Model-based classification via mixtures of multivariate t-distributions" *Computational Statistics and Data Analysis* 55, 520–529.

- C.Bouveyron and J.Jacques, Model-based Clustering of Time Series in Group-specific Functional Subspaces, Advances in Data Analysis and Classification, vol. 5 (4), pp. 281-300, 2011 <doi:10.1007/s11634-011-0095-6>

- Schmutz A, Jacques J, Bouveyron C, et al (2020) Clustering multivariate functional data in group-specific functional subspaces. Comput Stat 35:1101-1131

- Cristina Anton, Iain Smith Model-based clustering of functional data via mixtures of t distributions. Advances in Data Analysis and Classification (to appear).

See Also

teigen, kmeans, tkmeans,predict.tfunHDDC

Examples

```
set.seed(1027)
#simulataed univariate data

data = genModelFD(ncurves=300, nsplines=35, alpha=c(0.9,0.9,0.9),
                  eta=c(10, 7, 17))
```

```
plot(data$fd, col = data$groupd)

clm = data$groupd

model1=c("AkjBkQkDk", "AkjBQkDk", "AkBkQkDk", "ABkQkDk", "AkBQkDk", "ABQkDk")

t1<-tfunHDDC(data$fd,K=3,threshold=0.2,init="kmeans",nb.rep=1,dfconstr="no",
             dfupdate="numeric", model=model1[1], itermax=10)

if (!is.null(t1$class)) table(clm, t1$class)

###############example when some classifications are known

known1=rep(NA,1,300)

known1[1]=clm[1]

known1[103]=clm[103]

known1[250]=clm[250]

t2<-tfunHDDC(data$fd,K=3,threshold=0.2,init="kmeans",nb.rep=1,dfconstr="no",
             dfupdate="numeric", model=model1[1],known=known1, itermax=10)
if (!is.null(t2$class)) table(clm, t2$class)

####### example when some classifications are known

known1=rep(NA,1,300)

known1[1:100]=rep(3,1,50)

t3<-tfunHDDC(data$fd,K=3,threshold=0.2,init="kmeans",nb.rep=1,dfconstr="no",
             dfupdate="numeric", model=model1[1],known=known1, itermax=10)

if (!is.null(t3$class)) table(clm, t3$class)

############################multivariate simulated data
set.seed(1027)

conTrig <- genTriangles()

cls = conTrig$groupd # groups 5 and 6 (contaminated) into 1 and 3 respectively

res_s = tfunHDDC(conTrig$fd, K=4, dfconstr="no", dfupdate="numeric",
                 model="ABKQKDK", init="kmeans", threshold=0.2, nb.rep=1,
                 itermax=10)

if (!is.null(res_s$class)) table(cls, res_s$class)
```

Index