Package ‘teigen’

December 7, 2016

Type Package

Title Model-Based Clustering and Classification with the Multivariate t Distribution

Version 2.2.0

Date 2016-12-06

Author Jeffrey L. Andrews, Jaymeson R. Wickins, Nicholas M. Boers, Paul D. McNicholas

Maintainer Jeffrey L. Andrews <jeff.andrews@ubc.ca>

Description Fits mixtures of multivariate t-distributions (with eigen-decomposed covariance structure) via the expectation conditional-maximization algorithm under a clustering or classification paradigm.

License GPL (>= 2)

LazyLoad yes

Imports parallel

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-12-07 08:44:00

R topics documented:

teigen-package *teigen: Model-Based Clustering and Classification with the Multivariate t Distribution*

Description

Fits mixtures of multivariate t-distributions (with eigen-decomposed covariance structure) via the expectation conditional-maximization algorithm under a clustering or classification paradigm.

Details

Package:	teigen
Type:	Package
Version:	2.2.0
Date:	2016-12-06
License:	GPL (>=2)
LazyLoad:	yes

Author(s)

Jeffrey L. Andrews, Jaymeson R. Wickins, Nicholas M. Boers, Paul D. McNicholas

Maintained by: Jeffrey L. Andrews <jeff.andrews@ubc.ca>

References

Andrews JL and McNicholas PD (2012). “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.

See Also

`teigen` for main function

ckd *Indian Chronic Kidney Disease Data*

Description

This is a cleaned up version of the Chronic Kidney Disease data set available from the UCI learning repository:

http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease

Nominal variables have been removed and rows with missing values recorded on the remaining variables have also been removed.

Usage

```
data("ckd")
```

Format

This data frame contains 203 rows (observations) and 13 columns (variables): 1.) ckdclass: There are 2 classes, ckd or notckd 2.) age: in years 3.) blood.pressure: in mm/Hg 4.) blood.glucose.random: in mgs/dl 5.) blood.urea: in mgs/dl 6.) serum.creatinine: in mgs/dl 7.) sodium: in mEq/L 8.) potassium: in mEq/L 9.) hemoglobin: in gms 10.) packed.cell.volume 11.) white.blood.cell.count: in cells/cmm 12.) red.blood.cell.count: in cells/cmm

Source

See http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease for original source.

References

Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Examples

```
data(ckd)
hclustres <- cutree(hclust(dist(ckd[,-1])),3)
table(ckd[,1], hclustres)
```

`plot.teigen` *plot.teigen: Plotting Function for tEIGEN Objects*

Description

For multivariate data, outputs marginal contour or uncertainty plots to the graphics device for objects of class `teigen`. For univariate data, plot a univariate density plot.

Usage

```
## S3 method for class 'teigen'
plot(x, xmarg = 1, ymarg = 2, res = 200,
what = c("contour", "uncertainty"),alpha = 0.4, col = rainbow(x$G),
pch = 21, cex = NULL, bg = NULL, lty = 1, uncmult = 0,
levels = c(seq(0.01, 1, by = 0.025), 0.001),
main=NULL, xlab=NULL, draw.legend=TRUE, ...)
```

Arguments

`x` An object of class `teigen`

`xmarg` Scalar argument giving the number of the variable to be used on the x-axis

`ymarg` Scalar argument giving the number of the variable to be used on the y-axis. If `NULL`, the teigen object will be interpreted as univariate using x[,xmarg] as the data.

`res` Scalar argument giving the resolution for the calculation grid required for the contour plot. Default is 200, which results in a 200x200 grid. Also determines how smooth the univariate density curves are (higher res, smoother curves). Ignored for uncertainty plots.

`what` Only available if the model provided by x is multivariate. Character vector stating which plots should be sent to the graphics device. Choices are `"contour"` or `"uncertainty"`. Default is to plot both (see Details).

`alpha` A factor modifying the opacity alpha for the plotted points. Typically in [0,1].

`col` A specification for the default plotting color. See section 'Color Specification' in `par`. Note that the number of colors provided must equal to the number of groups in the teigen object (extra colors ignored).

`pch` Either an integer specifying a symbol or a single character to be used as the default in plotting points. See `points` for possible values and their interpretation. If pch is one of 21:25, see bg for coloring.

`cex` A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. For uncertainty plots, cex changes the size of the smallest sized point. The relative sizes amongst the points remains the same. As a result, the sizes of all the points change.

`bg` Background (fill) color for the open plot symbols if pch is one of 21:25. If pch is in 21:25 point color will be black (col = "black" will be used). If no bg is given to color the inside of the points, col will be used.

`lty` The line type for univariate plotting. See `par` for more information. Only updates the group curves, not the density or mixture curves.

`uncmult` A multiplier for the points on the uncertainty plot. The larger the number, the more the size difference becomes magnified. Large points will get larger faster than smaller points.

`levels` Numeric vector giving the levels at which contours should be drawn. Default is to draw a contour in 0.25 steps, plus a contour at 0.001. This may result in more/less contours than desired depending on the resulting density.

`main` Optional character string for title of plot. Useful default if left as `NULL`.

`xlab` Optional character string for x-axis label.

`draw.legend` Logical for a default generation of a legend to the right of the plot.

`...` Options to be passed to `plot`.

Details

`"contour"` plots the marginal distribution of the mixture distribution. For univariate data, or if `ymarg` is `NULL`, a univariate marginal is provided that includes the kernel density estimate from `density()`, the mixture distribution, and colour-coded component densities.

`"uncertainty"` plots the uncertainty of each observation's classification - the larger the point, the more uncertainty associated with that observation. Uncertainty in this context refers to the probability that the observation arose from the mixture component specified by the colour in the plot rather than the other components.

The default behavior of the function is to specify both plot types. This opens up an interactive menu from which the user may switch back and forth between both graphs. On exiting the menu, the graph that was plotted last will remain in the open device.

Author(s)

Jeffrey L. Andrews, Jaymeson R. Wickins, Nicholas M. Boers

See Also

`teigen`

Examples

```
set.seed(2521)
tfaith <- teigen(faithful, models = "CCCC", Gs = 1:4, verbose = FALSE)

plot(tfaith, what = "uncertainty", cex = 1.5, uncmult = 1.5)
plot(tfaith, what = "contour")
plot(tfaith, ymarg = NULL, lwd = 2)
```

`predict.teigen` *predict.teigen: Predicting Function for tEIGEN Objects*

Description

Provides the fuzzy probability matrix and classification vector for inputted observations assuming the model provided by the `teigen` object.

Usage

```
## S3 method for class 'teigen'
predict(object, newdata=NULL, modelselect="BIC", ...)
```

Arguments

`object` An object of class `teigen`

`newdata` Data frame or matrix of new observations on the same variables used in the fitting of the `teigen` object. For predicting one observation, a vector is permitted. If `NULL`, then the observations used in the fitting of the `teigen` object are inputted.

`modelselect` A character string of either `"BIC"` (default) or `"ICL"` indicating the desired model-selection criteria to apply to the `teigen` object.

`...` Arguments to be passed to other functions.

Details

Note that the scale argument from the `teigen` object is passed along to the `predict`function. See examples below for plotting.

Value

`fuzzy` Matrix of fuzzy classification probabilities

`classification` Vector of maximum a posteriori classifications

Author(s)

Jeffrey L. Andrews

See Also

`teigen`

Examples

```
set.seed(2521)
ind <- sample(1:nrow(faithful), 20)
set.seed(256)
tfaith_unscaled <- teigen(faithful[-ind,], models = "UUUU", Gs = 2, verbose = FALSE, scale=FALSE)
pred_unscaled <- predict(tfaith_unscaled, faithful[ind,])
set.seed(256)
tfaith_scaled <- teigen(faithful[-ind,], models = "UUUU", Gs = 2, verbose = FALSE, scale=TRUE)
pred_scaled <- predict(tfaith_scaled, faithful[ind,])
identical(pred_unscaled$classification, pred_scaled$classification)

##Plotting UNSCALED
plot(tfaith_unscaled, what="contour")
points(faithful[ind,1], faithful[ind,2], pch=15)
plotcolours <- rainbow(tfaith_unscaled$G)
points(faithful[ind,1], faithful[ind,2], pch=20, col=plotcolours[pred_unscaled$classification])

##Plotting SCALED
plot(tfaith_scaled, what="contour")
points((faithful[ind,1]-tfaith_scaled$info$scalemeans[1])/tfaith_scaled$info$scalesd[1],
       (faithful[ind,2]-tfaith_scaled$info$scalemeans[2])/tfaith_scaled$info$scalesd[2],
       pch=15)
plotcolours <- rainbow(tfaith_scaled$G)
points((faithful[ind,1]-tfaith_scaled$info$scalemeans[1])/tfaith_scaled$info$scalesd[1],
       (faithful[ind,2]-tfaith_scaled$info$scalemeans[2])/tfaith_scaled$info$scalesd[2],
       pch=20, col=plotcolours[pred_scaled$classification])
```

`print.teigen` *print.teigen: Print Function for tEIGEN Objects*

Description

Outputs short, concise information on the teigen object: BIC value, best model, and best group. Same info for ICL is output if it disagrees with the BIC value.

Usage

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

Arguments

`x`	An object of class `teigen`
`...`	Options to be passed to `print`.

Author(s)

Jaymeson R. Wickins, Nicholas M. Boers, Jeffrey L. Andrews

See Also

`teigen`

`summary.teigen` *summary.teigen: Summary Function for tEIGEN Objects*

Description

Summary method for class "teigen". Gives summary information.

Usage

```
## S3 method for class 'teigen'
summary(object, ...)
## S3 method for class 'summary.teigen'
print(x, ...)
```

Arguments

`object`	An object of class `teigen`
`x`	An object of class "`summary.teigen`".
`...`	Options to be passed to `summary`.

Value

An object of class "`summary.teigen`" that has a specialized `print` method. The object is a list containing the BIC and ICL values, as well as loglik value, model number and group number for the BIC. These values are also stored for ICL if it disagrees with the BIC value.

Author(s)

Jaymeson R. Wickins, Nicholas M. Boers, Jeffrey L. Andrews

See Also

`teigen`

---

teigen *teigen: Function for Model-Based Clustering and Classification with the Multivariate t Distribution*

Description

Fits multivariate t-distribution mixture models (with eigen-decomposed covariance structure) to the given data within a clustering paradigm (default) or classification paradigm (by giving either training index or percentage of data taken to be known). Can be run in parallel.

Usage

```
teigen(x, Gs = 1:9, models = "all", init = "kmeans", scale = TRUE, dfstart = 50,
known = NULL, training = NULL, gauss = FALSE, dfupdate = "approx",
eps = c(0.001, 0.1), verbose = TRUE, maxit = c(Inf,Inf),
convstyle = "aitkens", parallel.cores = FALSE,
ememargs = list(25, 5, "UUUU", "hard"))
```

Arguments

`x`	A numeric matrix, data frame, or vector (for univariate data) .
`Gs`	A number or vector indicating the number of groups to fit. Default is 1-9.
`models`	A character vector giving the models to fit. See details for a comprehensive list of choices.

`init` A list of initializing classification of the form that `init[[G]]` contains the initializing vector for all G considered (see example below). Alternatively, the user can use a character string indicating initialization method. Currently the user can choose from `"kmeans"` (default), ‘hard’ random - `"hard"`, ‘soft’ random - `"soft"`, ``` ``emem'' ``` (see `ememargs` below for description), and `"uniform"` (classification only).

`scale` Logical indicating whether or not the function should scale the data. Default is `TRUE` and is the prescribed method — tEIGEN models are not scale invariant.

`dfstart` The initialized value for the degrees of freedom. The default is 50.

`training` Optional indexing vector for the observations whose classification is taken to be known.

`known` A vector of known classifications that can be numeric or character - optional for clustering, necessary for classification. Must be the same length as the number of rows in the data set. If using in a true classification sense, give samples with unknown classification the value `NA` within `known` (see training example below).

`gauss` Logical indicating if the algorithm should use the gaussian distribution. If `models="mclust"` or `"gaussian"` then `gauss=TRUE` is forced.

`dfupdate` Character string or logical indicating how the degrees of freedom should be estimated. The default is `"approx"` indicating a closed form approximation be used. Alternatively, `"numeric"` can be specified which makes use of `uniroot`. If `FALSE`, the value from `dfstart` is used and the degrees of freedom are not updated. If `TRUE`, `"numeric"` will be used for back-compatibility.

`eps` Vector (of size 2) giving tolerance values for the convergence criterion. First value is the tolerance level for iterated M-steps. Second value is tolerance for the EM algorithm: convergence is based on Aitken’s acceleration, see cited papers for more information.

`verbose` Logical indicating whether the running output should be displayed. This option is not available in parallel. What is displayed depends on the width of the R window. With a width of 80 or larger: time run, estimated time remaining, percent complete are all displayed.

`maxit` Vector (of size 2) giving maximum iteration number for the iterated M-steps and EM algorithm, respectively. A warning is displayed if either of these maximums are met, default for both is Inf (aka, no limit).

`convstyle` Character string specifying the method of determining convergence. Default is "aitkens" which uses a criteria based on Aitken’s acceleration, but "lop" (lack of progress) may be used instead.

`parallel.cores` Logical indicating whether to run teigen in parallel or not. If `TRUE`, then the function determines the number of cores available and uses all of them. Alternatively, a positive integer may be provided indicating the number of cores the user wishes to use for running in parallel.

`ememargs` A list of the controls for the emEM initialization with named elements: `numstart` - numeric, number of starts (default 25) `iter` - numeric, number of EM iterations (default 5) `model` - character string, model name to be used (default “UUUU” from C,U,I nomenclature...see details below) `init` - character string, initialization method for emEM (default `hard`, or `soft`, or `kmeans`). The emEM initialiazation will run multiple, randomized initialization attempts for a limited number of iterations, and then continue the model-fitting process.

Details

Model specification (via the `models` argument) follows either the nomenclature discussed in Andrews and McNicholas (2012), or via the nomenclature popularized in other packages. In both cases, the nomenclature refers to the decomposition and constraints on the covariance matrix:

$$\Sigma_g = \lambda_g D_g A_g D_g'$$

The nomenclature from Andrews and McNicholas (2012) gives four letters, each letter referring to (in order) λ, D, A, and the degrees of freedom. Possible letters are `"U"` for unconstrained, `"C"` for constrained (across groups), and `"I"` for when the parameter is replaced by the appropriately sized identity matrix (where applicable). As an example, the string `"UICC"` would refer to the model where $\Sigma_g = \lambda_g A$ with degrees of freedom held equal across groups.

The alternative nomenclature describes (in order) the volume (λ), shape (A), orientation (D), and degrees of freedom in terms of `"V"`ariable, `"E"`qual, or the `"I"`dentity matrix. The example model discussed in the previous paragraph would then be called by `"VEIE"`.

Possible univariate models are `c("univUU", "univUC", "univCU", "univCC")` where the first capital letter describes `"U"`nconstrained or `"C"`onstrained variance and the second capital letter refers to the degrees of freedom. Once again, `"V"`ariable or `"E"`qual can replace U and C, but this time the orders match between the nomenclatures.

As many models as desired can be selected and ran via the vector supplied to `models`. More commonly, subsets can be called by the following character strings: `"all"` runs all 28 tEIGEN models (default), `"dfunconstrained"` runs the 14 unconstrained degrees of freedom models, `"dfconstrained"` runs the 14 constrained degrees of freedom models, `"mclust"` runs the 10 MCLUST models using the multivariate Gaussian distribution rather than the multivariate t, `"gaussian"` is similar but includes four further mixture models than MCLUST, `"univariate"` runs the univariate models - will automatically be called if one of the previous shortcuts is used on univariate data.

Note that adding `"alt"` to the beginning of those previously mentioned characters strings will run the same models, but return results with the V-E-I nomenclature.

Also note that for G=1, several models are equivalent (for example, UUUU and CCCC). Thus, for G=1 only one model from each set of equivalent models will be run.

Value

`x` Data used for clustering/classification.

`index` Indexing vector giving observations taken to be known (only available when classification is performed).

`classification` Vector of group classifications as determined by the BIC.

`bic` BIC of the best fitted model.

`modelname` Name of the best model according to the BIC.

`allbic` Matrix of BIC values according to model and G. A value of -Inf is returned when the model did not converge.

`bestmodel` Character string giving best model (BIC) details.

`G` Value corresponding to the number of components chosen by the BIC.

`tab` Classification table for BIC-selected model (only available when known is given). When classification is used the “known” observations are left out of the table.

`fuzzy` The fuzzy clustering matrix for the model selected by the BIC.

`logl` The log-likelihood corresponding to the model with the best BIC.

`iter` The number of iterations until convergence for the model selected by the BIC.

`parameters` List containing the fitted parameters: `mean` - matrix of means where the rows correspond to the component and the columns are the variables; `sigma` - array of covariance matrices (multivariate) or variances (univariate); `lambda` - vector of scale parameters, or constants of proportionality; `d` - eigenvectors, or orientation matrices; `a` - diagonal matrix proportional to eigenvalues, or shape matrices; `df` - vector containing the degrees of freedom for each component; `weights` - matrix of the expected value of the characteristic weights; `pig` - a vector giving the mixing proportions.

`iclresults` List containing all the previous outputs, except `x` and `index`, pertaining to the model chosen by the best ICL (all under the same name except `allicl` and `icl` are the equivalent of `allbic` and `bic`, respectively).

`info` List containing a few of the original user inputs, for use by other dedicated functions of the `teigen` class.

Author(s)

Jeffrey L. Andrews, Jaymeson R. Wickins, Nicholas M. Boers, Paul D. McNicholas

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.

See Also

See package manual `tEIGEN`

Examples

```
###Note that only one model is run for each example
###in order to reduce computation time

#Clustering old faithful data with hard random start
tfaith <- teigen(faithful, models="UUUU", Gs=1:3, init="hard")
plot(tfaith, what = "uncertainty")
summary(tfaith)

#Clustering old faithful with hierarchical starting values
initial_list <- list()
```

```
clustree <- hclust(dist(faithful))
for(i in 1:3){
initial_list[[i]] <- cutree(clustree,i)
}
tfaith <- teigen(faithful, models="CUCU", Gs=1:3, init=initial_list)
print(tfaith)

#Classification with the iris data set
#Introducing NAs is not required; this is to illustrate a `true' classification scenario
irisknown <- iris[,5]
irisknown[134:150] <- NA
triris <- teigen(iris[,-5], models="CUUU", init="uniform", known=irisknown)

##Parallel examples:
###Note: parallel.cores set to 2 in order to comply
###with CRAN submission policies (set to higher
###number or TRUE to automatically use all available cores)

#Clustering old faithful data with tEIGEN
tfaith <- teigen(faithful, models="UUUU",
parallel.cores=2, Gs=1:3, init="hard")
plot(tfaith, what = "contour")

#Classification with the iris data set
irisknown <- iris[,5]
irisknown[sample(1:nrow(iris),50)] <- NA
tiris <- teigen(iris[,-5], parallel.cores=2, models="CUUU",
init="uniform", known=irisknown)
tiris$tab
```

Index