Package ‘snipEM’

February 20, 2015

Type Package

Title Snipping methods for robust estimation and clustering

Version 1.0

Date 2014-10-02

Author Alessio Farcomeni, Andy Leung

Maintainer Alessio Farcomeni <alessio.farcomeni@uniroma1.it>

Description Snipping methods

License GPL (>= 2)

Depends R (>= 3.0.0), Rcpp (>= 0.10.0), mvtnorm, MASS, GSE

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-10-06 15:09:20

R topics documented:

  ldmvnorm ......................................................... 2
  sclust .......................................................... 2
  skmeans .......................................................... 4
  snipEM ............................................................ 6
  stEM .............................................................. 8
  sumlog ............................................................ 10

Index 12
ldmvnorm | *Multivariate Normal Log-Density for Complete and Incomplete Data*

**Description**

This function provides the log-density function for the multivariate normal distribution with mean equal to mu and covariance matrix Sigma. Marginal distributions will be used when the vector (or matrix) of quantiles is incomplete. That is, when the vector (or matrix) of quantiles contain NA.

**Usage**

```r
ldmvnorm(x, mu, Sigma, onNA=0)
```

**Arguments**

- `x`: Vector or matrix of quantiles. If `x` is a matrix, each row is taken to be a quantile.
- `mu`: Mean vector, default is `rep(0, length = ncol(x))`
- `Sigma`: Covariance matrix, default is `diag(ncol(x))`.
- `onNA`: Action for a row on NAs. Default is to return 0.

**Author(s)**

Alessio Farcomeni <alessio.farcomeni@uniroma1.it>, Andy Leung <andy.leung@stat.ubc.ca>

**Examples**

```r
## Not run:
x <- matrix(rnorm(1000),100,10)
u <- matrix(rbinom(1000, 1, 0.1), 100, 10)
x[ u == 1 ] <- NA
mu <- rep(0,10)
Sigma <- diag(10)
ldmvnorm(x, mu, Sigma)
```

## End(Not run)

---

sclust | *Snipping for robust model based clustering analysis with cellwise outliers*

**Description**

Estimates a finite Gaussian mixture model optimized over a snipping set.
Usage

```r
sclust(X, k, V, R, restr_fact=12, tol = 1e-04, maxiters = 100,
       maxiters.S = 1000, print.it = FALSE)
```

Arguments

- `X` Data.
- `k` Number of clusters
- `V` Binary matrix of the same size as `X`. Zeros correspond to initial snipped entries.
- `R` Initial guess for cluster labels, 1 to `k`.
- `restr_fact` Restriction factor, i.e., constraint on the condition number of all covariance matrices for each cluster. Default is 12.
- `tol` Tolerance for convergence. Default is `1e-4`.
- `maxiters` Maximum number of iterations for the SM algorithm. Default is 100.
- `maxiters.S` Maximum number of iterations of the inner greedy snipping algorithm. Default is 1000.
- `print.it` Logical; if `TRUE`, partial results are print. Default is `FALSE`.

Details

This function computes the `sclust` estimator of Farcomeni (2014). It leads to robust mixture modeling in presence of entry-wise outliers. It is based on a classification-expectation-snip-maximize (CESM) algorithm. At the S step, the likelihood is optimized over the set of snipped entries, at the M step the location and scatter estimates are updated. The S step is based on a greedy algorithm, unlike the one proposed in Farcomeni (2014,2014a). The number of snipped entries `sum(1-V)` is kept fixed throughout. Note that initializing with labels arising from classical (non-robust) clustering methods may be detrimental for the final performance of `sclust` and may even yield an error due to empty clusters.

Value

A list with the following elements:

- `R` Final cluster labels.
- `mu` Estimated location matrix.
- `S` Array of estimated scatter matrices.
- `V` Final (optimal) V matrix.
- `lik` Gaussian log-likelihood at convergence.
- `iter` Number of outer iterations before convergence.

Author(s)

Alessio Farcomeni <alessio.farcomeni@uniroma1.it>, Andy Leung <andy.leung@stat.ubc.ca>
References


See Also

*snipEM, stEM, generate.cellcontam, sumlog, ldmvnorm*

Examples

```r
## not run:
X <- matrix(NA,1000,5)
# two clusters
k <- 2
X[1:500,] <- rnorm(500*5)
X[501:1000,] <- rnorm(500*5,15)
R <- rep(c(1,2), each=500)

# 5% cellwise outliers
s <- sample(1000*5,1000*5*0.05)
X[s] <- rnorm(1000*5*0.05,-100,100)
V <- X
V[s] <- 0
V[-s] <- 1

# Initial V and R
Vinit <- matrix(1, nrow(X), ncol(X))
Vinit[which(X > quantile(X,0.975) | X < quantile(X,0.025))] <- 0
Rinit <- kmeans(X,2)$clust

# Snipped robust clustering
sc <- sclust(X,2,Vinit,Rinit)
table(R,Rinit)
table(R,sc$R)

## End(not run)
```

**skmeans**

Snipped k-means clustering with cellwise outliers

**Description**

Perform k-means clustering on a data matrix with cellwise outliers using a snipping algorithm.

**Usage**

`skmeans(X, k, V, clust, s, itersmax = 10^5, D = 1e-1)`
Arguments

- **X**: Data.
- **k**: Integer; number of clusters, \( k > 1 \).
- **V**: Binary matrix of the same size as \( X \). Zeros correspond to initial snipped entries.
- **clust**: Vector of size \( n \) containing values from 1 to \( k \). Starting solution for class labels.
- **itersmax**: Max number of iterations of the algorithm. Default is \( 3 \times 10^5 \).
- **s**: Binary vector of size \( n \) for trimming, starting solution. Number of zeros will be preserved and correspond to trimmed rows. If the vector is \( \text{rep}(1, n) \), it performs no trimming. Default is \( \text{rep}(1, n) \).
- **D**: Tuning parameter for the fitting algorithm. Corresponds approximately to the maximal change in loss by switching two non outlying entries. Comparing different choices is recommended. Default is \( 1e^{-1} \).

Details

This function computes the skmeans estimator of Farcomeni (2014). It leads to robust \( k \)-means in presence of entry-wise and cell-wise outliers. The number of snipped entries \( \sum(1-V) \) and trimmed rows \( \sum(1-s) \) is kept fixed throughout. Initial estimates for \( V, s \) and \( \text{clust} \) should be provided. Note that initializing with labels arising from classical (non-robust) clustering methods may be detrimental for the final performance of skmeans and may even yield an error due to empty clusters.

Value

A list with the following elements:

- **loss**: Loss function (the total sum of squares) at convergence.
- **mu**: Estimated locations.
- **s**: Final (optimal) trimmed rows in vector of size \( n \).
- **V**: Final (optimal) \( V \) matrix.
- **clust**: Final (optimal) class labels as vector of size \( n \).

Author(s)

Alessio Farcomeni <alessio.farcomeni@uniroma1.it>, Andy Leung <andy.leung@stat.ubc.ca>

References


See Also

- sclust, stEM, snipEM, generate.casecontam, generate.cellcontam
Examples

```r
## Not run:
x <- matrix(NA, 1000, 5)
# two clusters
k <- 2
x[1:500,] <- rnorm(500*5)
x[501:1000,] <- rnorm(500*5, 15)
clust <- rep(c(1,2), each=500)

# 5% cellwise outliers
s <- sample(1000*5, 1000*5*0.05)
x[s] <- runif(1000*5*0.05, -100, 100)
V <- x
V[s] <- 0
V[-s] <- 1

# Initial V and R
Vinit <- matrix(1, nrow(x), ncol(x))
Vinit[which(x > quantile(x, 0.975) | x < quantile(x, 0.025))] <- 0
km <- kmeans(x, k)
clustinit <- km$clust

# Snipped robust clustering
skm <- skmeans(x, k, Vinit, clustinit)

table(clust, km$clust)
table(clust, skm$clust)

## End(Not run)
```

### Description

Computes an estimator optimizing the Gaussian likelihood over a snipping set. The function `snipEM(initialV)` can be used to perform some iterations to initialize `V`.

### Usage

```r
snipEM(X, V, tol = 1e-04, maxiters = 500, maxiters.S = 1000, print.it = FALSE)

snipEM.initialV(X, V, mu0, S0, maxiters.S = 100, greedy = TRUE)
```

### Arguments

- **X**: Data.
- **V**: Binary matrix of the same size as `X`. Zeros correspond to initial snipped entries.
tol: Tolerance for convergence. Default is 1e-4.
maxiters: Maximum number of iterations for the SM algorithm. Default is 500.
maxiters.S: Maximum number of iterations of the inner greedy snipping algorithm. Default is 1000.
print.it: Logical; if TRUE, partial results are print. Default is FALSE.
mu0: Initial estimate for the mean vector that is used in the initialization stage.
S0: Initial estimate for the covariance matrix that is used in the initialization stage.
greedy: Logical; if TRUE, perform the greedy snipping algorithm in search for the binary matrix that gives the largest likelihood value throughout maxiters.S iterations. If FALSE, stop right after the snipping algorithm finds a binary matrix that gives a larger likelihood value than the initial one. Default is TRUE.

Details
This function computes the $sclust$ estimator of Farcomeni (2014) with $k = 1$. It therefore provides a robust estimate of location and scatter in presence of entry-wise outliers. It is based on a snip-maximize (SM) algorithm. At the S step, the likelihood is optimized over the set of snipped entries, at the M step the location and scatter estimates are updated. The S step is based on a greedy algorithm, unlike the one proposed in Farcomeni (2014, 2014a). The number of snipped entries $\sum(1-V)$ is kept fixed throughout.

Results depend on good initialization of the $V$ matrix. A boxplot rule (see examples) usually works well. The function snipEM.initialV can be used to improve the initial choice through some iterations updating only $V$ from initial (robust) estimates $\mu0$ and $S0$. In the example, the EMVE is used to obtain $\mu0$ and $S0$.

Value
A list with the following elements:

- $\mu$: Estimated location.
- $S$: Estimated scatter matrix.
- $V$: Final (optimal) $V$ matrix.
- $lik$: Gaussian log-likelihood at convergence.
- $iter$: Number of outer iterations before convergence.

Author(s)
Alessio Farcomeni <alessio.farcomeni@uniroma1.it>, Andy Leung <andy.leung@stat.ubc.ca>

References
See Also

`sclust, stEM, generate.cellcontam, sumlog, ldmvnorm`

Examples

```r
## not run:
# 10% cellwise outliers
n=100
p=10
Xc <- generate.cellcontam(100, 10, 100, 5, 0.10)
A <- Xc$A  # true correlation matrix
Uc <- Xc$u  # (true outlier) binary matrix
Xc <- Xc$x

# initial V
V <- matrix(1,n,p)
V[!is.na(match(as.vector(Xc), boxplot(as.vector(Xc), plot=FALSE)$out))] <- 0
Xna <- Xc
Xna[ which( V == 0 ) ] <- NA
resEMVE <- emve(Xna)
V <- snipEM.initialV(Xc, V, resEMVE@mu, resEMVE@S)$V

# Performance in LRT distance
resSEM <- snipEM(Xc, V)
slrt(resSEM@S, A)
slrt(cov(Xc), A)

## End(not run)
```

---

**stEM**

*Snipping and trimming for location and scatter estimation with case-wise and cellwise outliers*

Description

Computes an estimator optimizing the Gaussian likelihood over a snipping and trimming set.

Usage

```r
stEM(X, V, tol = 1e-4, maxiters = 500, maxiters.S = 1000, print.it = FALSE)
```

Arguments

- **X** Data.
- **V** Binary matrix of the same size as X. Zeros correspond to initial snipped entries, rows of zeros correspond to initial trimmed entries.
- **tol** Tolerance for convergence. Default is 1e-4.
- **maxiters** Maximum number of iterations for the SM algorithm. Default is 500.
stEM

maxiters.S Maximum number of iterations of the inner greedy snipping algorithm. Default is 1000.
print.it Logical; if TRUE, partial results are print. Default is FALSE.

Details

This function combines computes the snipEM estimator of Farcomeni (2014) with trimming. Optimization over a trimming set is performed via usual concentration steps (Rousseeuw and van Driessen, 1999). It therefore provides a robust estimate of location and scatter in presence of entry-wise and case-wise outliers. The number of snipped entries and trimmed rows is kept fixed throughout. V must contain at least one row of zeros (otherwise use snipEM).

Value

A list with the following elements:

mu Estimated location.
S Estimated scatter matrix.
V Final (optimal) V matrix.
lik Gaussian log-likelihood at convergence.
iter Number of outer iterations before convergence.

Author(s)

Alessio Farcomeni <alessio.farcomeni@uniroma1.it>, Andy Leung <andy.leung@stat.ubc.ca>

References

Farcomeni, A. (2014) Robust constrained clustering in presence of entry-wise outliers, Technometrics, 56, 102-111

See Also

sclust, snipEM, generate.casecontam, generate.cellcontam, emve, sumlog, ldmvnorm

Examples

```R
## Not run:
# 10% cellwise outliers
Xc <- generate.cellcontam(100, 10, 100, 5, 0.10)
A <- Xc$A   # true correlation matrix
Uc <- Xc$u  # (true outlier) binary matrix
Xc <- Xc$x
```
# 5% case-wise outliers
Xct <- generate.casecontam(100, 10, 100, 10, 0.05)
xct <- rbind(xct, xct[s(1)]
trim <- rep(1,105)
trim[101:105] <- 0  # (true casewise outlier) binary vector
Uc <- rbind(Uc, matrix(1,5,10))

# initial V
V <- matrix(1,105,10)
# identify rows to be trimmed
km <- kmeans(xc,2)$clust
wtrim <- which(km==which.min(table(km)))
V[wtrim,] <- 0
Vtmp <- V[-wtrim,]

# identify cells to be snipped
Vtmp[is.na(match(as.vector(xc[-wtrim,]), boxplot(as.vector(xc[-wtrim,]), plot=FALSE)$out))] <- 0
Xna <- xc[-wtrim,]
Xna[ which( Vtmp == 0 ) ] <- NA
resEMVE <- emve(Xna)
Vtmp <- snipEM(initialV(Xc[-wtrim,], Vtmp, resEMVE@mu, resEMVE@S)
V[-wtrim,] <- Vtmp$V

# Performance in LRT distance
resSTEM <- stem(Xc, V)
slrt(resSTEM$S, A)
slrt(cov(Xc), A)

## End(Not run)

---

### sumlog

Log-sum from log data

**Description**

Obtain log(sum(x)) from log(x), without passing to exponentials. It is based on the fact that log(a + b) = log(a) + log (1 + exp(log(b) - log(a))).

**Usage**

```r
sumlog(x, lower=-745, upper=709)
```

**Arguments**

- `x` Vector of log-values
- `lower` Value such that exp(lower-epsilon)=0
- `upper` Value such that exp(upper+epsilon)=Inf
**Details**

This function computes the logarithm of the sum of \( \exp(x) \), without passing through exponentials. It shall be used to avoid under/over flow. It has proven useful in computing the likelihood of finite mixture models, normalization constants, importance sampling, etc. It is described in the appendix of Farcomeni (2012).

**Value**

A scalar equal to \( \log(\text{sum}(\exp(x))) \).

**Author(s)**

Alessio Farcomeni <alessio.farcomeni@uniroma1.it>, Andy Leung <andy.leung@stat.ubc.ca>

**References**


**Examples**

```r
## Not run:
# complete underflow without sumlog
x <- c(-750,-752)
log(sum(exp(x)))
sumlog(x)

# imprecise sum
x <- c(-745,-752)
log(sum(exp(x)))
sumlog(x)

# no issues
x <- c(log(3),log(2))
log(5)
log(sum(exp(x)))
sumlog(x)

## End(Not run)
```
Index

emve, 9

generate.casecontam, 5, 9
generate.cellcontam, 4, 5, 8, 9

ldmvnorm, 2, 4, 8, 9

sclust, 2, 5, 8, 9
skmeans, 4
snipEM, 4, 5, 6, 9
steEM, 4, 5, 8, 8
sumlog, 4, 8, 9, 10