Package: EMgaussian (via r-universe)

August 22, 2024

Encoding UTF-8

Type Package

Title Expectation-Maximization Algorithm for Multivariate Normal (Gaussian) with Missing Data

Version 0.2.1

Date 2024-02-15

Maintainer Carl F. Falk <carl.falk@mcgill.ca>

Description Initially designed to distribute code for estimating the Gaussian graphical model with Lasso regularization, also known as the graphical lasso (glasso), using an Expectation-Maximization (EM) algorithm based on work by Städler and Bühlmann (2012) <doi:10.1007/s11222-010-9219-7>. As a byproduct, code for estimating means and covariances (or the precision matrix) under a multivariate normal (Gaussian) distribution is also available.

License GPL (>= 3)

Imports Rcpp, matrixcalc, Matrix, lavaan, glasso, glassoFast, caret

Suggests testthat (>= 3.0.0), psych, bootnet, qgraph, cglasso

LinkingTo Rcpp, RcppArmadillo

Config/testthat/edition 3

RoxygenNote 7.3.1

Repository https://falkcarl.r-universe.dev

RemoteUrl https://github.com/falkcarl/emgaussian

RemoteRef HEAD

RemoteSha 4bc02c4e9e9de1ea18074fb782bfdcb7285c8071

Contents

`em.cov` *EM algorithm for multivariate normal, covariance matrix parameterization*

Description

EM algorithm for multivariate normal, covariance matrix parameterization

Usage

```
em.cov(
  dat,
  max.iter = 500,
  tol = 1e-05,
  start = c("diag", "pairwise", "listwise", "full"),
  debug = 0,
  ...
)
```

Arguments

`dat`	Data frame or matrix that contains the raw data.
`max.iter`	Max number of EM cycles.
`tol`	Tolerance for change in parameter estimates across EM Cycles. If all changes are less than `tol`, the algorithm terminates.
`start`	Starting value method (see details).
`debug`	(Experimental) set an integer value > 1 for some information as the algorithm runs.
`...`	Space for additional arguments, not currently used.

Details

This function computes all means and covariances among a set of variables using the Expectation-Maximization (EM) algorithm to handle missing values, and assuming multivariate normality. The EM code was originally developed for the precision matrix parameterization (`em.prec`), i.e., the parameters are the means and the inverse of the covariance matrix. But, this is easily modifiable to handle a covariance matrix parameterization such that means and covariances are the model parameters.

For starting values, the function accepts either a list that has `mustart` and `covstart` slots corresponding to the starting mean and covariance matrix. This is useful if the user would like to use custom starting values. Otherwise, a character corresponding to any of the options available in the `startvals.cov` function will be used to take a guess at starting values.

Value

A list with the following:

- p.est: all parameter estimates as a vector (means followed by unique elements of covariance matrix; each row below diagonal stacked).
- mu: estimated means.
- S: estimated covariance matrix.
- it: number of EM cycles completed.
- conv: boolean value indicating convergence (TRUE) or not (FALSE).

Examples

```
library(psych)
data(bfi)
test <- em.cov(bfi[,1:25])
```

---

em.prec *EM algorithm for multivariate normal, precision matrix parameterization*

Description

EM algorithm for multivariate normal, precision matrix parameterization

Usage

```
em.prec(
  dat,
  max.iter = 500,
  tol = 1e-05,
  start = c("diag", "pairwise", "listwise", "full"),
  glassoversion = c("none", "glassoFast", "glasso", "glassonostart"),
  debug = 0,
  ...
)
```

Arguments

dat	Data frame or matrix that contains the raw data.
max.iter	Max number of EM cycles.
tol	Tolerance for change in parameter estimates across EM Cycles. If all changes are less than tol, the algorithm terminates.
start	Starting value method (see details).

glassoversion	Character indicating whether to do regularization (lasso), and if so, using which package. "glasso" uses the glasso function and uses the E-step covariance matrix for starting values, "glassoFast" uses glassoFast{glassoFast} which also penalizes the diagonal of the precision matrix by default (glasso does not), "glassonostart" also uses glasso but no "warm" starting values.
debug	(Experimental) set an integer value > 1 for some information as the algorithm runs.
...	Arguments passed down to any of the glasso functions.

Details

This function computes all means and the precision matrix (inverse of covariance matrix) among a set of variables using the Expectation-Maximization (EM) algorithm to handle missing values, and assuming multivariate normality. The EM code was originally developed based on Stadler and Buhlmann (2012) and for use with the graphical lasso (i.e., glasso). This version allows the possibility of using a lasso by specifying something other than "none" for glassoversion. However, it can also be used without regularization to just estimate the precision matrix.

For starting values for the EM algorithm itself (not at the M-step), the function accepts either a list that has mustart and covstart slots corresponding to the starting mean and covariance matrix. This is useful if the user would like to use custom starting values. Otherwise, a character corresponding to any of the options available in the startvals.cov function will be used to take a guess at starting values.

Value

A list with the following:

- p.est: all parameter estimates as a vector (means followed by unique elements of precision matrix; each row below diagonal stacked).
- mu: estimated means.
- S: estimated covariance matrix.
- K: estimated precision matrix.
- it: number of EM cycles completed.
- conv: boolean value indicating convergence (TRUE) or not (FALSE).

References

Städler, N., & Bühlmann, P. (2012). Missing values: sparse inverse covariance estimation and an extension to sparse regression. Statistics and Computing, 22, 219–235. doi:10.1007/s11222010-92197

Examples

```
library(psych)
data(bfi)
test <- em.prec(bfi[,1:25])
```

EMggm *Regularized GGM under missing data with EBIC or k-fold cross validation*

Description

Regularized GGM under missing data with EBIC or k-fold cross validation

Usage

```
EMggm(
  dat,
  max.iter = 500,
  est.tol = 1e-07,
  start = c("diag", "pairwise", "listwise", "full"),
  glassoversion = c("glasso", "glassoFast", "glassonostart", "none"),
  rho = 0,
  rhoselect = c("ebic", "kfold"),
  N = NULL,
  gam = 0.5,
  zero.tol = 1e-10,
  k = 5,
  seed = NULL,
  debug = 0,
  convfail = FALSE,
  ...
)
```

Arguments

dat	A data frame or matrix of the raw data.
max.iter	Maxumum number of EM cycles for the EM algorithm, passed eventually to em.prec.
est.tol	Tolerance for change in parameter estimates across EM Cycles. If all changes are less than tol, the algorithm terminates.
start	Starting value method (see details of em.prec). Note that "none" will not do any regularization, making rho moot.
glassoversion	Character indicating which function to use at the M-step for regularization. See em.prec) for more details.
rho	Vector of tuning parameter values. Defaults to just a single value of 0 (i.e., also no regularization).
rhoselect	Method of selecting the tuning parameter. "ebic" will select the best value in rho based on that yielding the best EBIC. "kfold" will do k-fold cross-validation.
N	Sample size to use in any EBIC calculations. Defaults to average pairwise sample size.

gam	Value of gamma in EBIC calculations. Typical choice, and the default, is .5.
zero.tol	Tolerance in EBIC calculations for declaring edges to be zero. Anything in absolute value below zero.tol will be considered zero and will not count as a parameter in EBIC calculations.
k	If using k-fold cross validation, an integer specifying the number of folds. Defaults to 5.
seed	Random number seed passed to function that does k-fold cross validation. Use if you want folds to be more or less replicable.
debug	(Experimental) Pick an integer above 0 for some messages that may aide in debugging.
convfail	(Experimental) What to do if optimization fails. If TRUE, will fill in the partial correlation matrix with 0's.
...	Other arguments passed down to em.prec and to functions that do the glasso.

Details

This function will estimate a regularized Gaussian graphical model (GGM) using either EBIC or k-fold cross validation to select the tuning parameter. It was written as a wrapper function to the em.prec function, which is an implementation of the EM algorithm from Städler & Bühlmann (2012). These authors also used k-fold cross validation, which is implemented here.

For the tuning parameter (rho), typically a grid of of values is evaluated and the one that results in the best EBIC or cross validation metric is selected and returned as the final model. See rhogrid and examples for some ways to pick these candidate tuning parameter values.

This function is intended to be compatible (to my knowledge) with estimateNetwork in the bootnet package in that one can input this as a custom function and then have all of the benefits of plotting, centrality indices, and so on, that bootnet provides.

Most methods in examples were studied by Falk and Starr (under review). In particular use of this function in conjunction with estimateNetwork worked well for both "ebic" and "kfold" for model selection, though the original article used the glassoFast package for estimation, which by default penalizes the diagonal of the precision matrix. It seems slightly more common to not penalize the diagonal. The below use glasso, which does not penalize the diagonal of the precision matrix; this approach appeared to be sensitive to the choice of starting values, sometimes getting stuck if the "pairwise" starting value approach was used. The most recent set of simulations, in which it did not get stuck, used the "diag" approach. In addition, the two-stage approach studied by Falk and Starr (under review) also performed well, though not as well as the present function; that approach is available in the bootnet package. Note also that cglasso has an implementation of Städler & Bühlmann (2012), but we found in simulations with a high proportion of missing data that our implementation was less likely to encounter estimation problems.

Value

A list with the following elements:

- results: Contains the "best" or selected model, with elements mostly following that of em.prec's output. Additional elements include rho and crit, which are the grid of tuning parameter values and value of the criterion. These are added here because if used with bootnet, it may not save the other slots below.

- rho: Grid for rho.
- crit: Vector of the same length as the grid for rho with the criterion (EBIC or cross-validation). Smaller is better.
- graph: Estimated partial correlation matrix; bootnet appears to expect this in order to do plots, compute centrality indices and so on.

References

Evidence that EBIC and k-fold works well with this package (cite if you use the EMglasso R package): Falk, C. F., & Starr, J. (2023, July 19). Regularized cross-sectional network modeling with missing data: A comparison of methods. Preprint: doi:10.31219/osf.io/dk6zv

Original publication on use of EM algorithm and k-fold cross-validation for glasso: Städler, N., & Bühlmann, P. (2012). Missing values: sparse inverse covariance estimation and an extension to sparse regression. Statistics and Computing,22, 219–235. doi:10.1007/s1122201092197

If you use this package with bootnet: Epskamp,S., Borsboom, D., & Fried, E. I. (2018). Estimating psychological networks and their accuracy: A tutorial paper. Behavior research methods, 50(1), 195–212. doi:10.3758/s1342801708621

If you also use this package with qgraph: Epskamp, S., Cramer, A., Waldorp, L., Schmittmann, V. D., & Borsboom, D. (2012). qgraph: Network visualizations of relationships in psychometric data. Journal of Statistical Software, 48 (1), 1-18.

Foundational article on glasso: Friedman, J. H., Hastie, T., & Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics, 9 (3), 432-441.

Foundational article on use of EBIC with glasso: Foygel, R., & Drton, M. (2010). Extended Bayesian information criteria for Gaussian graphical models.

If glasso is also the estimation method: Friedman, J. H., Hastie, T., & Tibshirani, R. (2014). glasso: Graphical lasso estimation of Gaussian graphical models. Retrieved from https://CRAN.R-project.org/package=glasso

If glassoFast is also the estimation method: Sustik M.A., Calderhead B. (2012). GLASSOFAST: An efficient GLASSO implementation. UTCS Technical Report TR-12-29:1-3.

Examples

```
library(psych)
library(bootnet)
data(bfi)

# Regularized estimation with just a couple of tuning parameter values

# EBIC
rho <- seq(.01,.5,length.out = 50)
ebic1 <- EMggm(bfi[,1:25], rho = rho, glassoversion = "glasso",
               rhoselect = "ebic")

# k-fold
kfold1 <- EMggm(bfi[,1:25], rho = rho, glassoversion = "glasso",
                rhoselect = "kfold")
```

```
plot(rho, ebic1$crit) # values of EBIC along grid
plot(rho, kfold1$crit) # values of kfold along grid

# partial correlation matrix
# ebic1$graph
# kfold1$graph

# Integration with bootnet package
ebic2 <- estimateNetwork(bfi[,1:25], fun = EMggm, rho = rho,
                         glassoversion = "glasso", rhoselect = "ebic")
kfold2 <- estimateNetwork(bfi[,1:25], fun = EMggm, rho = rho,
                         glassoversion = "glasso", rhoselect = "kfold")

# ebic2 and kfold2 now do just about anything one could normally do with an
# object returned from estimateNetwork e.g., plotting
plot(ebic2)
plot(kfold2)

# e.g., centrality indices
library(qgraph)
centralityPlot(ebic2)
# and so on

# Other ways to pick grid for tuning parameter... 100 values using method
# that qgraph basically uses; requires estimate of covariance matrix. Here
# the covariance matrix is obtained directly from the data.
rho <- rhogrid(100, method="qgraph", dat = bfi[,1:25])

ebic3 <- estimateNetwork(bfi[,1:25], fun = EMggm, rho=rho,
                         glassoversion = "glasso", rhoselect = "ebic")
kfold3 <- estimateNetwork(bfi[,1:25], fun = EMggm, rho=rho,
                         glassoversion = "glasso", rhoselect = "kfold")

# look at tuning parameter values vs criterion
plot(ebic3$result$rho, ebic3$result$crit)
plot(kfold3$result$rho, kfold3$result$crit)

# EBIC with bootnet; does listwise deletion by default
ebic.listwise <- estimateNetwork(bfi[,1:25], default="EBICglasso")

# EBIC with bootnet; two-stage estimation
# Note the constant added to bfi tricks lavaan into thinking the data are
# not categorical
ebic.ts <- estimateNetwork(bfi[,1:25] + 1e-10, default="EBICglasso",
                           corMethod="cor_auto", missing="fiml")
```

---

rhogrid *Create sequence of possible tuning parameter values*

Description

Create sequence of possible tuning parameter values

Usage

```
rhogrid(
  n.rho,
  method = c("qgraph", "glassopath"),
  rho.min.ratio = 0.01,
  dat = NULL,
  S = NULL,
  ...
)
```

Arguments

n.rho	Integer corresponding to the number of tuning parameter values.
method	Character corresponding to the method to create tuning parameter values ("qgraph" or "glassopath"); see Details.
rho.min.ratio	Numeric value that mimics EBICglasso behavior for tuning parameter. i.e., "Ratio of lowest (tuning parameter) compared to maximal (tuning parameter)".
dat	The raw data, if S is not provided used to estimate S. Not required if S is provided.
S	Covariance matrix for the data. If provided, supersedes dat.
...	Other arguments passed down to em.prec.

Details

For regularized estimation of the Gaussian graphical model, a sequence or grid of possible tuning parameter values is often tried, with the tuning parameter that optimizes some criterion (EBIC, k-fold cross validation) chosen. This is an attempt to automate some of the sequence creation. Code is borrowed from qgraph and glasso, acknowledged in references below. Both require some estimate of the covariance matrix in order to do regularization; if not provided, em.prec with default options is attempted.

For "qgraph" the max value is determined by the maximum absolute value of the difference between the covariance matrix and an identity matrix. The min is rho.min.ratio times the max value. A sequence that is equally spaced on a log scale is then constructed between these two values.

For "glassopath", the max value is the max absolute value of the covariance matrix. The min is the max divided by the number of desired tuning parameter values. A sequence that is equally spaced between these two values is then constructed.

Value

A vector of possible tuning parameter values.

References

qgraph: Epskamp, S., Cramer, A., Waldorp, L., Schmittmann, V. D., & Borsboom, D. (2012). qgraph: Network visualizations of relationships in psychometric data. Journal of Statistical Software, 48 (1), 1-18.

glasso (i.e., glassopath option): Friedman, J. H., Hastie, T., & Tibshirani, R. (2014). glasso: Graphical lasso estimation of Gaussian graphical models. Retrieved from https://CRAN.R-project.org/package=glasso

Examples

```
library(psych)
data(bfi)

# pick 50 values using the approach qgraph uses; give data as input
rho <- rhogrid(50, method="qgraph", dat = bfi[,1:25])

emresult <- em.cov(bfi[,1:25])
S<-emresult$S

# pick 50 values using the approach glasso uses; give S as input
rho2 <- rhogrid(50, method="glassopath", S = S)
```

`startvals.cov` *Starting values for means and covariances*

Description

Starting values for means and covariances

Usage

```
startvals.cov(dat, start = c("diag", "pairwise", "listwise", "full"))
```

Arguments

`dat`	Data frame or matrix that contains the raw data.
`start`	Starting value method (see details).

Details

Attempts to figure out a starting values for the means and covariances for use with other functions that do the EM algorithm such as `em.prec` or `em.cov`. Note that means are determined univariately using all available cases. For covariances, several options are available:

- "diag" Use all available complete values to compute the variances of each variable and construct a diagonal covariance matrix. - "pairwise" Pairwise (co)variances will be used to construct the starting covariance matrix. - "listwise" Listwise deletion will be used and only those with complete

data will contribute to the starting covariance matrix. - "full" Cheat and use `lavaan` to obtain direct maximum likelihood estimates of covariances. This defeats the purpose to some extent, but not that `lavaan` may be quite slow compared to this implementation.

Value

A list consisting of:

- `mustart` - starting values for means.
- `covstart` - starting values for covariances.

Examples

```
library(psych)
data(bfi)
startvals.cov(bfi[,1:25])
```

Index