Supplementary File: mvgraphnorm: An R package which generates constrained covariance matrix for a given graph to generate samples from a Gaussian graphical model

Frank Emmert-Streib, Shailesh Tripathi and Matthias Dehmer

Abstract

This is the supplementary file containing instructions and various examples to generate constrained covariance matrix using different algorithms.

Contents

1 Example: Obtaining constrained covariance matrix by using different algorithms

The following code allows to generate samples from a multivariate normal distribution using a constrained covariance matrix which is generated by using *HTF*, *IPF* or *KIM* algorithm. The constrained covariance matrix is generated by using an input network or an adjacency matrix of a network.

```
> ############ Figure A #############
> library("mvgraphnorm")
> ## input network ##
> g <- barabasi.game(100, directed=FALSE)
> zz1 <- rmvggm(net.str=g, method="htf")

[1] "htf"

> zz2 <- rmvggm(net.str=g, method="ipf")

[1] "ipf"

> zz3 <- rmvggm(net.str=g, method="kim")

[1] "kim"

> names(zz1)

[1] "dat"     "sigma"   "net.str"

> head(zz1[[2]][1:10,1:10])
```

```
             1           2             3             4             5
1  1.071253691 -0.11811419  0.0011071669  0.1030279683 -0.0013754710
2 -0.118114192  1.31062068 -0.0122864682 -0.0113595290  0.0152170695
3  0.001107167 -0.01228647  1.1234271516  0.0001069020 -0.0001429917
4  0.103027968 -0.01135953  0.0001069020  1.2045720295 -0.0001344431
5 -0.001375471  0.01521707 -0.0001429917 -0.0001344431  1.0040193618
6  0.001933726 -0.02145705  0.0002011500  0.0001859744 -0.0002491288
              6             7             8             9            10
1  0.0019337264  7.303545e-06 -9.337562e-05 -2.518770e-05 -0.0035882442
2 -0.0214570478 -8.080036e-05  1.029019e-03  2.794882e-04  0.0398166864
3  0.0002011500  7.592643e-07 -9.354947e-06 -2.620074e-06 -0.0003723912
4  0.0001859744  7.138725e-07 -9.223693e-06 -2.422405e-06 -0.0003449317
5 -0.0002491288 -5.331193e-03  6.792275e-02  3.245021e-06  0.0004611878
6  0.9151198030  1.322837e-06 -1.684676e-05 -1.191987e-02 -0.0006518656
```

2 Example: Obtaining constrained covariance matrix by defining a random correlation matrix using different algorithms

The following code allows to generate samples from a multivariate normal distribution using a user defined correlation matrix. If the matrix is not positive definite matrix we use Higham (2002) algorithm to convert the matrix into the nearest positive definite matrix. This Input matrix is used to obtain a constrained covariance matrix for *HTF* and *IPF* algorithms.

```
> ############ Figure A #############
> library("mvgraphnorm")
> ## input network ##
> g <- barabasi.game(50, directed=F)
> ## Generate a random correlation matrix ##
> m <- matrix(0,50,50)
> m[upper.tri(m)] <- runif(min=-.8, max=.8, 25*49)
> m <- m+t(m)
> diag(m) <- 1
> zz <- rmvggm(20, net.str=g, method="htf"
+ , cor=m)

[1] "htf"

> head(zz[[2]][1:10,1:10])

            1           2           3          4           5           6
1  1.55133124  0.141506724 -0.020897203  0.74492075  0.085448127  0.531191422
2  0.14150672  0.842090609 -0.124356968  0.06794893  0.007794261  0.048453326
3 -0.02089720 -0.124356968  1.231842645 -0.01003445 -0.001151028 -0.007155413
```

```
4  0.74492075  0.067948928 -0.010034453  1.27299995  0.041030620  0.255068359
5  0.08544813  0.007794261 -0.001151028  0.04103062  1.158848148  0.029258298
6  0.53119142  0.048453326 -0.007155413  0.25506836  0.029258298  0.982185871
             7            8             9           10
1 -0.069734908  0.042108252  0.0118612602 -0.028716291
2 -0.414989012  0.250581474  0.0010819389 -0.170887420
3  0.061284130 -0.037004987 -0.0001597768  0.025236051
4 -0.033485421  0.020219609  0.0056955592 -0.013789035
5 -0.003841031  0.002319343  0.1608626015 -0.001581708
6 -0.023877934  0.014418289  0.0040614148 -0.009832747

>
```

3 Example: plots of edge and non-edge components of constrained covariance matrix

The following example provides a of distribution of non edge and edge components of inverse of constrained covariance matrix ($\Sigma(r)$) and the inverse of sample covariance matrix(S).

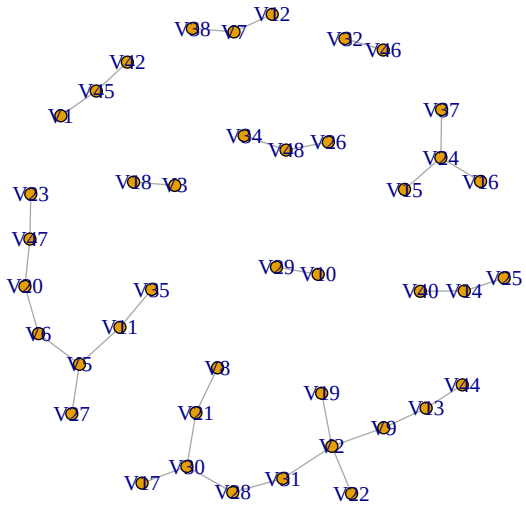
```
> ############ Figure A #############
> library("mvgraphnorm")
> ## input network ##
> g <- barabasi.game(50, directed=F)
> ## Generate a random correlation matrix ##
> m <- matrix(0,50,50)
> m[upper.tri(m)] <- runif(min=-.8, max=.8, 25*49)
> m <- m+t(m)
> diag(m) <- 1
> zz <- rmvggm(20, net.str=g, method="htf"
+ , cor=m)

[1] "htf"

> p <- NULL
> p <- viz.rmvggm(zz, net=FALSE)
> dev.off()

null device
          1

> print(p$covp)
> print(p$covsmp)
>
>
```



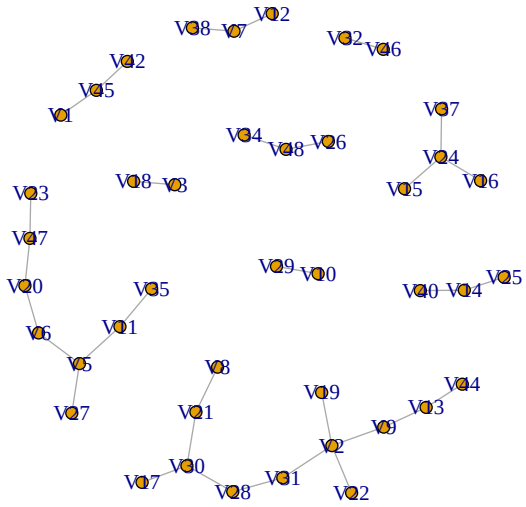
4 Example: Inferring a network structure using conditional independence test

The following example provides a network inferred from samples generated from multivariate normal distribution using covariance matrix ($\Sigma(r)$) generated by using *HTF*, *IPF* or *KIM* algorithm. For inferring the network we use conditional independence test using Grow-Shrink (GS) method.

```
> ############ Figure A #############
> library("mvgraphnorm")
> ## input network ##
> g <- barabasi.game(50, directed=F)
> ## Generate a random correlation matrix ##
> m <- matrix(0,50,50)
> m[upper.tri(m)] <- runif(min=-.8, max=.8, 25*49)
> m <- m+t(m)
> diag(m) <- 1
> zz <- rmvggm(20, net.str=g, method="htf", cor=m)
```

```
[1] "htf"
```

```
> p <- NULL
> p <- viz.rmvggm(zz, net=TRUE, test="cor", undirected = TRUE)
> print(p$netp)
>
>
```



```
> sessionInfo()

R version 3.3.3 (2017-03-06)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X Yosemite 10.10.5

locale:
[1] C/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] mvgraphnorm_1.81 qpgraph_2.8.3    mvtnorm_1.0-6    igraph_1.1.2

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.14               GenomeInfoDb_1.10.3
 [3] XVector_0.14.1             GenomicFeatures_1.26.4
 [5] bitops_1.0-6               zlibbioc_1.20.0
 [7] biomaRt_2.30.0             digest_0.6.13
 [9] bit_1.1-14                 annotate_1.52.1
[11] RSQLite_2.1.1              memoise_1.1.0
[13] lattice_0.20-35            pkgconfig_2.0.2
[15] Matrix_1.2-12              graph_1.52.0
[17] DBI_1.0.0                  Rgraphviz_2.18.0
[19] parallel_3.3.3             rtracklayer_1.34.2
[21] Biostrings_2.42.1          S4Vectors_0.12.2
[23] IRanges_2.8.2              stats4_3.3.3
[25] bit64_0.9-7                grid_3.3.3
[27] Biobase_2.34.0             AnnotationDbi_1.36.2
[29] XML_3.98-1.9               qtl_1.42-8
[31] BiocParallel_1.8.2         bnlearn_4.4
[33] corpcor_1.6.9              blob_1.1.1
[35] magrittr_1.5               GenomicAlignments_1.10.1
[37] Rsamtools_1.26.2           BiocGenerics_0.20.0
[39] GenomicRanges_1.26.4       SummarizedExperiment_1.4.0
[41] xtable_1.8-3               RCurl_1.95-4.10
```