False discovery rate control for sparse correlation matrix

Arief Gusnanto
Department of Statistics, University of Leeds (UK)
A.Gusnanto@leeds.ac.uk
April 9, 2025

This document outlines some examples on how to use the package `scfdr` to create sparse correlation matrix where the non-zero correlations are controlled in terms of their false discovery rate.

1 History of update

2 April 2025: This document is created.

2 Downloading the `R` package

The package can be downloaded from the following GitHub repository:

```
https://github.com/ariefgusnanto/scfdr
```

At the time of the writing of this vignette, the package is currently being prepared for submission to CRAN. Any update on this will be included in the next relevant version of this vignette.

3 Installing the `R` package

The package is currently only for Linux/Unix operating system. We will update the vignette when it is available for Windows operating system. At the moment, it is recommended to download the package from GitHub first and install it locally from the file.

From inside `R`, you can install it by running

```
install.packages("scfdr_0.0.1.tar.gz", lib="xxx")
```

from the folder where the file `scfdr_0.0.1.tar.gz` is located. Note that the version number of the latest version of the package may be different to the above. The `xxx` denotes the path destination of the installation for the package. If you do not have administrative privileges to install in the default location for `R` packages, then you can specify the directory to install in your home directory (for example) or any other location where you have writing rights. If you do have administrative privileges to install the R package in the default location, then leave out the argument `lib` and run

```
install.packages("scfdr_0.0.1.tar.gz")
```

4 Importing the package

To import the package, we run

```
library("scfdr", lib="xxx")
```

where `xxx` here is the directory where the library is located. This is relevant, for example, when you installed the library *not* in the default location. If you previously installed the package in the default position, then the argument `lib` should be left out (automatically set to the default). Importing the `scfdr` package will automatically import the following dependency packages: `RSC`, `parallel`, and `MASS`.

The package `RSC` is for the calculation of the robust median absolute deviation (RMAD) correlation as shown in Serra *et al.* (2018). The package `parallel` will facilitate multi-core calculation of correlations (both the observed correlations and correlations under the null hypothesis of no true correlation). The package `MASS` is simply to facilitate the examples in the

5 Gene expression data

The gene expression data needs to be put in the ‘wide’ format: the rows of the data should correspond to observations/patients/arrays/samples and the columns should correspond to the genomic features (probesets/genes/loci). In the github page, you can download the data used in the main paper (GSE36133).

```
load("GSE36133-glio.RData")
```

The main R object is `glio` which contains the gene expression of 13,044 probesets from 43 patients.

```
> dim(glio)
[1] 13044    43
```

Note that the object `glio` is having the rows as probesets/genes, while the columns as patients.

6 Calculating observed correlation

To calculate the observed correlation matrix, we run

```
set.seed(121212)
cor.obs <- corrs(t(glio), "rmad")
```

where the `t()` will transpose the data so that the rows correspond to patients and the columns correspond to probesets/genes. The second argument indicates that the correlation calculated is the robust MAD. For Pearson correlation, then the second argument is left blank (default). Other options are `"spearman"` and `"kendall"`. Setting the seed (using `set.seed()`) above is not required, but this is to make the results from the paper reproducible.

7 Generating distribution of correlations under the null hypothesis

To generate correlations under the null hypothesis (z^*’s), we perform the following

```
temp1 <- cor.null.sampling(t(glio), "rmad", sampling=0.1, mc.cores=1, niter=8)
```

The input should be a data matrix with observations in the rows and genomic features on the columns (hence the transposition). The sampling is for the genes/probesets to be included to calculate correlations under the null distribution. The argument `mc.cores` should be set to 1 when using RMAD correlations, because the RSC package alrealy implemented multicore calculations. For other correlations (Pearson, Spearman, and Kendall), this can be set to the number of cores available in your computer. The `niter` indicates number of iterations in the sampling. This scheme is generally appropriate for gene expression data with $p > 10,000$ and will give approximately $\approx 0.04p^2$ correlations under the null hypothesis. For data with strong correlation structures, such as copy number alterations or DNA methylation data, then it is highly recommended to increase the sampling proportion to 0.3-0.4. Note that the resulting object `temp1` here is a list.

8 Estimating the local false discovery rate for each correlation

To estimate fdr(z), we run the following `R` command

```
res.temp <- est.prob(mat2vec(cor.obs), unlist(temp1), pplot=FALSE)
```

The above command will estimate the local false discovery rate for each correlation. The function `mat2vec()` above will take the lower diagonal element of the observed correlation matrix `cor.obs` and make them a vector. The function `vec2mat` will basically reverse this (to construct the correlation matrix based on the vectorised lower diagonal elements of the correlation matrix). This vector will be the first input to the function `est.prob`, which represents the observed correlation $f(z)$ in the main paper. The second input is the correlations under the null hypothesis (note that they need to be `unlist`-ed because the outcome of function `cor.null.sampling()` is a list. This second input represents the correlation under the null ($f_0(z)$) in the main paper. The null proportion π_0 is automatically estimated in the function `est.prob`.

In the above function, it is important that the argument `pplot` to be set as `FALSE` (the default). If this argument is set to `TRUE`, the function will take almost forever to plot the posterior probability of correlation for *each* correlation (in total, it can be in hundreds of millions).

9 Thresholding the correlation values

In order to create the sparse correlation matrix, we need to set to zero many correlations that are below the threshold. This is done by running

```
tr.cor.temp <- thresh.cor(res.temp, FDR=0.001)
```

The first input is the result of function `est.prob` that contains information on the observed correlation and local false discovery rate of each correlation value. The second argument here is the FDR (global false discovery rate) that we wish to control. It is recommended to use the above example for 0.001, before relaxing it.

To check the threshold value used for the FDR control set, then we can run

```
attr(tr.cor.temp,"threshold")
```

as it is set as an attribute in the object `tr.cor.temp`.

To check the proportion of the correlations that are now set to zero, we can run

```
sum(tr.cor.temp==0)/length(tr.cor.temp)
```

10 Creating sparse correlation matrix

The outcome of the function `thresh.cor()` above is still a vector (of lower diagonal elements of the sparse correlation matrix). To create the sparse correlation matrix, we run the following command to create the matrix

```
sp.cor <- vec2mat(tr.cor.temp)
```

11 Plotting

In the current version of the package, the plotting functionality is not yet fully implemented. So, to make some plots, you can use this “ad hoc” procedure.

The following will create the histogram of the observed (dense) correlation matrix (taking only the lower diagonal values).

```
hist(mat2vec(cor.obs), xlim=c(-1,1), las=1,
prob=T, main="Observed (GSE36133)",
xlab="RMAD")
```

This will create the histogram of the correlations under the null hypothesis.

```
hist(unlist(temp1), xlim=c(-1,1), las=1,
prob=T, main="Under H_0 (GSE36133)",
xlab="RMAD")
```

This will create a plot of densities for each $f(z)$ and $f_0(z)$.

```
plot(density(mat2vec(cor.obs)), xlim=c(-1,1), ylim=c(0,1.6), las=1,
main="Observed and under H_0 (GSE36133)",
xlab="RMAD", lwd=2)
lines(density(unlist(temp1)), lwd=2, col="indianred")
legend(-1, 1.6, c("f(z)", "f_0(z)"), lwd=2, col=c("black", "indianred"))
```

This will plot the posterior probability (to be truly correlated) as a function of the observed correlation. Note that here, we need to resample the observed correlation because the full result contains hundreds of millions of values that will unnecessarily burden the computer. Note that the results of the functions above carry the information for *all* correlations in the data, but we sample some of them here for plotting purposes *only*.

```
set.seed(121213)
xxx <- sample(1:length(res.temp$prob), 100000)
plot(res.temp$cor.obs[xxx], res.temp$prob[xxx], xlim=c(-1,1),
ylim=c(0,1), pch=19, cex=0.7,
xlab="RMAD", ylab="1-fdr(z)",
main="GSE36133")
rm(xxx)
```

12 Creating graph/network

The package contains a function `cor2adj` that will create adjacency matrix from correlation matrix. This function can be utilised to construct graphs/network from adjacency matrix.

13 Frequently asked questions

1. *My installation of the package gave an error message*
   There are some potential reasons for this. First, please check first that you have administrative privileges to install the package in the default location. If not, consider installing the package in home directory or any directory where you have writing permission. Second, at the moment, we can only install the package from a local file. So, make sure that you have downloaded the package file *and* you have included the path to the file when installing it.

2. *Can we run the package for analysis of copy number alterations data or DNA methylation data?*
   Computationally, the package can be used to analyse other genomic data. However, due to the correlation structure in the CNA and DNA methylation data, the sampling proportion in the estimation of $f_0(z)$ needs to be increased from the default 0.1 to 0.3-0.4. This is a guidance only, and the user needs to check this with their specific data.

3. *Can the package be run on a smaller dataset?*
   Yes it can, but to facilitate better estimation of $f_0(z)$, then it is important to increase sampling proportion in `cor.null.sampling` function to be higher (closer to one) and the argument `niter` to (for example) 20 or 25.

References

Serra A, Coretto P, Fratello M, Tagliaferri R, Stegle O. (2018) Robust and sparse correlation matrix estimation for the analysis of high-dimensional genomics data. *Bioinformatics*, **34**(4):625-634. doi: 10.1093/bioinformatics/btx642