# FUNNEL-GSEA: R Package

*Yun Zhang, Juilee Thakar, Xing Qiu*

*2016-10-01*

## Introduction

```
library(FUNNEL)
```

This R package is built for FUNNEL-GSEA. It is a statistical inference framework for Gene Set Enrichment Analysis (GSEA) based on FUNctioNal ELastic-net regression (FUNNEL), which utilizes the temporal information based on functional principal component analysis (FPCA), and disentangles the effects of **overlapping** genes by a functional extension of the elastic-net regression. It then performs hypothesis testing for gene sets by an extension of Mann-Whitney U test which is based on weighted rank sums computed from correlated observations.

In this vignette, we will introduce the one-step function for carrying out the FUNNEL analysis, and also some useful functions embedded in the FUNNEL framework.

- The one-step function, `FUNNEL.GSEA`, takes pre-processed data as inputs and runs the whole testing procedure automatically. There are also post-analysis tools such as `weightPerGene` and `plotWeight` to interpret and illustrate FUNNEL results.
- For more advanced users, we will introduce individual functions such as `FPCA.Fstats`, `equiv.regression` and `wMWUTest`. These are the key building blocks in the FUNNEL framework, and also quite useful by themselves. The users may find more flexibity by using these functions for their own analyses.

## FUNNEL one-step function

Load sample data.

```
data("H3N2-Subj1")
```

The one-step function is contained in `FUNNEL.GSEA`. It takes pre-processed data as input and runs the FUNNEL test automatically. For details of data pre-processing, please see `help("H3N2-Subj1")`. (It takes about 5~10 minutes to run the following.)

```
system.time(result <- FUNNEL.GSEA(X, tt, genesets))
```

```
## 26 real eigenvalues are negative or zero and are removed!
## Weight calculation...
## Gene set test...
```

```
##    user  system elapsed
## 398.023   4.598 403.067
```

## Some post-analysis tools

### Empirical gene set membership

If the users are interested in some specific genes, they may obtain the estimated weights (a.k.a. empirical gene set membership) by using the following function based on the `FUNNEL.GSEA` result.

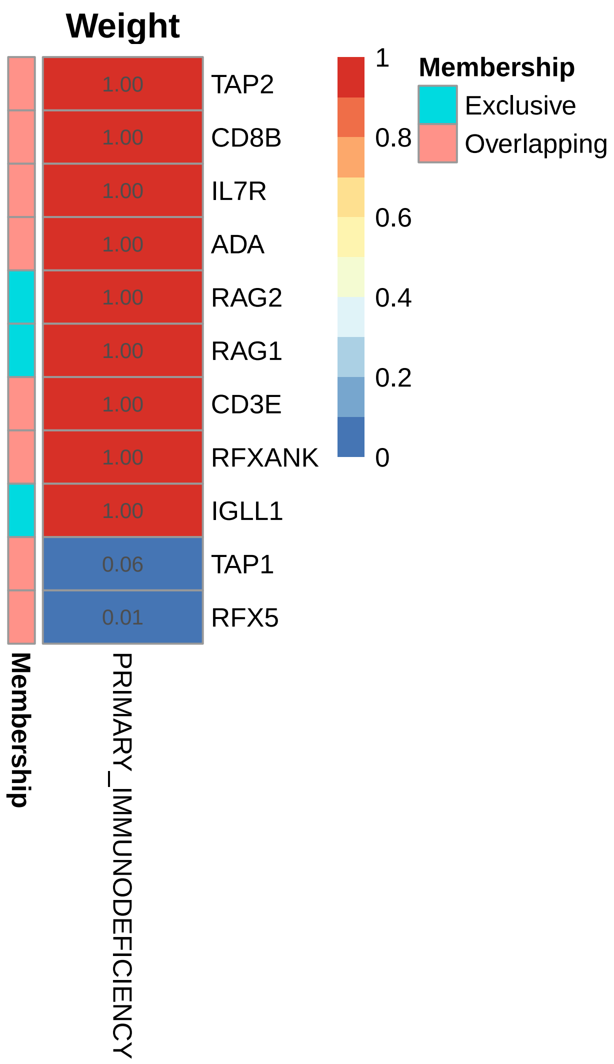
```
est.weights <- weightPerGene(result$weight.list, genesOfInterest=genesets[["PRIMARY_IMMUNODEFICIENCY"]]
```

It returns a list of length of `length(genesOfInterest)`. If a gene belongs multiple pathways, it returns a vector of estimated weights (sum to 1) for each of the overlapping pathways. If a gene is exclusive to one pathway, it returns integer `1`. `NA` means this gene is not presented in the expression data.

### Weight plot

The users can also plot the (non-zero) weights (a.k.a. empirical gene set membership) obtained from `FUNNEL.GSEA` for a specific gene set. For example, the (non-trivial) empirical membership for the Primary Immunodeficiency pathway is

```
plotWeight(result$weight.list, geneset.index="PRIMARY_IMMUNODEFICIENCY")
```



# More advanced functions

In this package, we also provide some individual functions for the key parts of the FUNNEL-GSEA framework. Advanced users may find these functions helpful for conducting their customized analyses.

### Some extra data processing

Firstly, the following data processing is performed internally in the `FUNNEL.GSEA` function.

```r
library(fda)

## Remove genes in predefined gene sets that are not present in X the filtered input data
genenames <- rownames(X)
newGenesets <- lapply(genesets, function(z) { intersect(z, genenames) } )

## Standardize timepoint and X so that the optimum roughness/L1/L2 penality parameters are applicable
tt2 <- (tt - min(tt))/diff(range(tt))
X2 <- t(scale(t(X)))

## Smoothing
mybasis <- create.bspline.basis(range(tt2), length(tt2)+4-2, 4, tt2)
mypar <- fdPar(mybasis, 2, lambda=10^-3.5)
fdexpr <- smooth.basis(tt2, t(X2), mypar)$fd
```

### Functional F-statistics for time-course gene expression data

`FPCA.Fstats` takes time-course expression data and time points as input values. By using Functional Principal Component Analysis (FPCA) techniques, it returns the functional F-statistic for each gene. For more details, please refer to *Wu, S. and Wu, H., 2013. More powerful significant testing for time course gene expression data using functional principal component analysis approaches. BMC bioinformatics, 14(1), p.1.*

```r
## Get functional F-statistics
Fstats <- FPCA.Fstats(X2, tt2)
```

```
## 26 real eigenvalues are negative or zero and are removed!
```

```r
## The following should be the same
identical(Fstats, result$Fstats)
```

```
## [1] TRUE
```

### An equivalence between functional and multivariate regression

`equiv.regression` takes functional covariates (`xfd`) and response (`yfd`) as input, and returns a list of equivalent multivariate covariates (`Xmat`) and response (`y`). Running penalized (or ordinary least square) regression on `y` and `Xmat` is equivalent to the corresponding concurrent functional regression with constant beta.

```r
library(quadrupen)

## Take genes C5AR1 and C3AR1 as two examples
gene.i <- c("C5AR1", "C3AR1")

## They belong to the following two pathways
newGeneset.i <- newGenesets[c("NEUROACTIVE_LIGAND_RECEPTOR_INTERACTION", "COMPLEMENT_AND_COAGULATION_CAS

## The response is just the smoothed curves of these two genes
yfd <- fdexpr[gene.i]

## Let us use the first 3 eigen-functions of both pathways as covariates
xfd <- FUNNEL:::PCA.genesets(fdexpr, newGeneset.i, nharm = 3, centerfns = FALSE)$harmonics

## Calculate the equivalent multivariate regression datasets
equiv <- equiv.regression(yfd, xfd, threshold = 0.01)
Y <- equiv$y; colnames(Y) <- gene.i      #3x2 matrix
X <- equiv$X                             #3x6 matrix
colnames(X) <- paste(rep(names(newGeneset.i), each=3), rep(paste0("eigfun", 1:3),
                     length(newGeneset.i)), sep=".")

## Now we can run multivariate elastinet regression on X and Y, as implemented in package quadrupen
en <- elastic.net(X, Y[, "C3AR1"], lambda1 = 0.4, lambda2 = 0.01,
                  intercept = FALSE, normalize = FALSE)

## beta.en are the regression coefficients
beta.en <- as.numeric(attributes(en)$coef)
names(beta.en) <- colnames(X)
```

### An extended Mann-Whitney U test that incorporates pre-computed weights and correlation

wMWUTest is an extension of the two-sample Mann-Whitney U test (a.k.a. rank sum test) which incorporates pre-calculated weights for the correlated inputs in the test group. Note that the pre-calculated correlation only applies to the test group (the gene set of interest). The correlation of the background genes is assumed to be zero. Pre-calculated weights are typically computed by function FUNNEL.GSEA.

```r
gg1 <- newGenesets[["GLYCOLYSIS_GLUCONEOGENESIS"]]
ww1 <- result$weight.list[["GLYCOLYSIS_GLUCONEOGENESIS"]]
rho <- result$correlation

## The test
o1 <- wMWUTest(gg1, result$Fstats, ww1, rho, df=length(tt2)-1)
## p-value for the gene set test
o1["greater"]
```

```
##   greater
## 0.1935949
```

```r
## Should be the same as the p-value below
result$pvals["GLYCOLYSIS_GLUCONEOGENESIS"]
```

```
## GLYCOLYSIS_GLUCONEOGENESIS
##                  0.1935949
```

# Session Info

```
sessionInfo()
```

```
## R version 3.2.2 (2015-08-14)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.6 (El Capitan)
## 
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
## 
## attached base packages:
## [1] splines   stats     graphics  grDevices utils     datasets  methods  
## [8] base     
## 
## other attached packages:
## [1] quadrupen_0.2-4 ggplot2_2.1.0   Rcpp_0.12.4     fda_2.4.4      
## [5] Matrix_1.2-3    FUNNEL_0.1.2   
## 
## loaded via a namespace (and not attached):
##  [1] knitr_1.12.3       magrittr_1.5       MASS_7.3-45       
##  [4] munsell_0.4.3      colorspace_1.2-6   lattice_0.20-33   
##  [7] stringr_1.0.0      plyr_1.8.3         tools_3.2.2       
## [10] parallel_3.2.2     grid_3.2.2         gtable_0.2.0      
## [13] htmltools_0.2.6    yaml_2.1.13        digest_0.6.9      
## [16] akima_0.5-12       RColorBrewer_1.1-2 reshape2_1.4.1    
## [19] formatR_1.2.1      codetools_0.2-14   evaluate_0.8      
## [22] rmarkdown_0.9.6    sp_1.2-1           pheatmap_1.0.7    
## [25] stringi_1.0-1      scales_0.4.0
```