mdgsa Library

David Montaner

(2014-11-24)

Contents

1 Introduction 1
  1.1 Citation .................................................. 1

2 Functional Profiling of Gene Expression Data 1
  2.1 Univariate Gene Set Analysis ........................ 3
  2.2 Multivariate Gene Set Analysis ....................... 7

3 Appendix 13
  3.1 1.- Multidimensional Functional Classification .... 13

4 Session Info 14

1 Introduction

The mdgsa library implements the gene set analysis methodology developed in Montaner and Dopazo (2010). It presents a flexible framework for analyzing the enrichment of gene sets along a given ranking of genes. The novelty is that, not just one ranking index but two, may be analyzed and jointly explored in a multidimensional gene set analysis.

As classical GSEA, our approach allows for the functional profiling of isolated genomic characteristics; differential gene expression, copy number analyses, or variant to disease associations may be interpreted in terms of gene sets using the mdgsa package. But more interestingly, our multivariate approach may be used to find out gene set enrichments due to the combined effect of two of such genomic dimensions. We could for instance detect gene sets affected by the interaction of gene expression changes and copy number alterations.

1.1 Citation

Further description of the mdgsa methods may be found at:

Multidimensional gene set analysis of genomic data.
David Montaner and Joaquin Dopazo.

2 Functional Profiling of Gene Expression Data

In this tutorial we use the data in the Acute Lymphocytic Leukemia expression dataset package of Bioconductor. First we will use the limma library to compute a differential gene expression analysis. Then we will use the functions uvGsa and mdGsa in the mdgsa package to perform uni-dimensional and bi-dimensional gene set analyses respectively. The functional interpretation will be done in terms of KEGG Pathways. The annotation will be taken from the hgu95av2.db library in Bioconductor.

First we load the data and describe the design matrix of the experiment
library (ALL)
data (ALL)

des.mat <- model.matrix (~ 0 + mol.biol, data = ALL)
colnames (des.mat) <- c("ALL", "BCR", "E2A", "NEG", "NUP", "p15")

head (des.mat)
## ALL BCR E2A NEG NUP p15
## 01005 0 1 0 0 0 0
## 01010 0 0 0 1 0 0
## 03002 0 1 0 0 0 0
## 04006 1 0 0 0 0 0
## 04007 0 0 0 1 0 0
## 04008 0 0 0 1 0 0

Then we can use limma to carry out some gene expression comparisons. We can for instance compare \textit{ALL} samples to \textit{NEG} control samples or explore gene differential expression between \textit{BCR} and \textit{NEG}.

library (limma)
cont.mat <- makeContrasts (ALL-NEG, BCR-NEG, levels = des.mat)
cont.mat
## Contrasts
## Levels ALL - NEG BCR - NEG
## ALL 1 0
## BCR 0 1
## E2A 0 0
## NEG -1 -1
## NUP 0 0
## p15 0 0

fit <- lmFit (ALL, design = des.mat)
fit <- contrasts.fit (fit, cont.mat)
fit <- eBayes (fit)

From this analysis we get \textit{test statistics} and \textit{p-values} for each of the two contrasts

fit$t[1:3,]
## Contrasts
## ALL - NEG BCR - NEG
## 1000_at -2.2610931 -0.7684296
## 1001_at -1.0962463 0.2064388
## 1002_f_at 0.7978798 -1.7527367

fit$p.value[1:3,]
## Contrasts
## ALL - NEG BCR - NEG
## 1000_at 0.02548234 0.44368153
## 1001_at 0.27507879 0.83678412
## 1002_f_at 0.42645360 0.08209929

These gene level information may be now interpreted in terms of \textit{gene sets}. For this example we will carry out a \textit{gene set analysis} using the functional blocks described in \textit{KEGG}, but any other functional data base such as the \textit{Gene Ontology} or even a customized one may be analyzed using mdgsa. We can get the \textit{KEGG} annotation from the \textit{hgu95av2.db} library as follows.

library (hgu95av2.db)
anmat <- toTable (hgu95av2PATH)
anmat[1:3,]
2.1 Univariate Gene Set Analysis

We can now carry out the functional interpretation of the contrast “BCR - NEG” for instance. The data needed for the gene set analysis are the p-values and test statistics returned by limma at gene level.

```r
fit$t[1:3, "BCR - NEG"]
## 1000_at 1001_at 1002_f_at
## -0.7684296 0.2064388 -1.7527367
fit$p.value[1:3, "BCR - NEG"]
## 1000_at 1001_at 1002_f_at
## 0.44368153 0.83678412 0.08209929
```

We load the mdgsa library.

```r
library (mdgsa)
```

The first step in the procedure is to combine p-values and test statistics in a single ranking value.

```r
rindex <- pval2index (pval = fit$p.value[,"BCR - NEG"], sign = fit$t[,"BCR - NEG"])
```

```r
rindex <- indexTransform (rindex)
```

```r
rindex[1:3]
## 1000_at 1001_at 1002_f_at
## -0.4038315 0.4309088 -1.2145063
```

This ranking index keeps the sign of the test statistic; that is, the information of whether the gene is over or underexpressed.

```r
plot (fit$t[,"BCR - NEG"], rindex)
```
but the evidence of the differential expression is taken directly from the p-value

```r
plot(fit$p.value[, "BCR - NEG"], rindex)
```
Next we will need to format the annotation. The function `annotMat2list` in the `mdgsa` library converts the annotation matrix into a list.

```r
anmat[1:3,]
## probe_id path_id
## 1 1000_at 04010
## 2 1000_at 04012
## 3 1000_at 04062
annot <- annotMat2list(anmat)
length(annot)
## [1] 228
```

Each element of the list contains the gene names of a gene set.

```r
lapply(annot[1:3], head, n= 3)
## $`00010`
## [1] "2035_s_at" "31488_s_at" "32210_at"
##
## $`00020`
## [1] "160044_g_at" "32332_at" "32546_at"
##
## $`00030`
## [1] "31485_at" "32210_at" "32336_at"
```

It is also important to make sure that the gene universe described by the ranking index and the annotation are concordant. The function `annotFilter` encompasses the annotation list to the names in the ranking index; it is also used to exclude functional blocks too small to be considered gene sets, or too big to be specific of any biological process of interest.
annot <- annotFilter (annot, rindex)

Now everything is ready to carry out the univariate gene set analysis.

res.uv <- uvGsa (rindex, annot)
## user system elapsed
## 9.976 0.012 10.078

The output of the uvGsa function is a data frame which rows correspond to functional blocks analyzed.

res.uv[1:3,]
## N lor pval padj
## 00010 65 -0.08900436 0.47430323 1.0000000
## 00020 33 -0.26525586 0.12586858 1.0000000
## 00030 21 -0.54531247 0.01420588 0.2497736

The uvGsa function fits a logistic regression model relating, the probability of genes belonging to the gene set, with the value of the ranking statistic.

Significant and positive log odds ratio (lor) indicate that the gene set is enriched in those genes with high values of the ranking statistic. In our example this means that genes up regulated in BCR compared to NEG are more likely to belong to the functional block. We could also say that the block of genes shows a significant degree of over expression BCR compared to NEG.

On the other hand, when a gene set has a negative log odds ratio we can say that the genes in the set are more likely to be associated to low values, negative in our case, of the ranking statistics. In our case this means that the gene set is down regulated in BCR compared to NEG.

The function uvPat helps you classifying the analyzed gene sets

res.uv[, "pat"] <- uvPat (res.uv, cutoff = 0.05)
table (res.uv[, "pat"])
##
## -1 0 1
## 10 156 42

Positive values (1) correspond to significant and positive log odds ratios. Negative values (-1) correspond to significant and negative log odds ratios. Zeros correspond to non enriched blocks.

As in this example we are analyzing KEGG pathways we can use the function getKEGGnames to find out the “name” of the pathways1.

res.uv[, "KEGG"] <- getKEGGnames (res.uv)

Finally, the function uvSignif may help us displaying just the enriched blocks.

res <- uvSignif (res.uv)
res[, c("pat", "KEGG")]
## pat KEGG
## 05416 1 Viral myocarditis
## 05332 1 Graft-versus-host disease
## 04612 1 Antigen processing and presentation
## 05330 1 Allograft rejection
## 04145 1 Phagosome
## 04940 1 Type I diabetes mellitus
## 05323 1 Rheumatoid arthritis
## 05145 1 Toxoplasmosis
## 05131 1 Shigellosis

1Similar function getGOnames is available to be used with Gene Ontology terms.
### 2.2 Multivariate Gene Set Analysis

But with the mdgsa library we can analyze not just one but two ranking statistics at a time. In our example we were interested not in just one differential expression contrast but in two: ALL vs. NEG and BCR vs. NEG. We used limma to fit this two contrasts (see previous sections) and computed gene statistics and p-values for for each of them.
We can combine this two matrices in a single one containing a ranking statistic for each contrast, just as we did in the univariate example.

```r
rindex <- pval2index (pval = fit$p.value, sign = fit$t)
rindex <- indexTransform (rindex)
rindex[1:3,]
## Contrasts
##    ALL - NEG   BCR - NEG
## 1000_at  -1.6649597 -0.4038315
## 1001_at  -0.9075447  0.4309088
## 1002_f_at  0.6548653 -1.2145063
```

Now we can explore the bi-dimensional distribution of this ranking indexes

```r
plot (rindex, pch = ".")
```
and search for gene sets enrichment patterns in both dimensions.

The same annotation list we filtered in the previous section may be used in this second example. Thus everything is ready to carry out our multidimensional analysis.

```r
res.md <- mdGsa (rindex, annot)
## user system elapsed
## 11.744 0.000 11.875
```

As in the univariate analysis, the output of the mdGsa function is a data frame with a row per analyzed gene set.

```r
res.md[1:3,]
## user system elapsed
## 11.744 0.000 11.875
```

The column of the row contain log odds ratios and p-values for each of the analyzed dimensions and also for their interaction effect. The function mdPat helps clarifying the bi-dimensional pattern of enrichment.

```r
res.md[,"pat"] <- mdPat (res.md)
table (res.md[,"pat"])
```

```r
## user system elapsed
## 11.744 0.000 11.875
```
And as before we can incorporate the KEGG names to our results.

```r
res.md[, "KEGG"] <- getKEGGnames(res.md)
res.md[1:3,]
```

Thus we could for instance explore the KEGG classified as having a with a q3f pattern. The q3f classification means that the genes of this gene set are located in the third quadrant of the bivariate ranking index representation.

The plotMdGsa function help us understanding such pattern.

```r
Q3 <- rownames(res.md)[res.md$pat == "q3f"]
Q3
## [1] "03030"
plotMdGsa(rindex, block = annot[[Q3]], main = res.md[Q3, "KEGG"])
```

Red dots in the figure represent the genes within the gene set. They show, in both dimensions of our ranking, values significantly more negatives that the remaining genes in the study. This same pattern may be appreciated in the ellipses drawn in the figure. The blue one represents a confidence region for all the genes in the study. The red one shows the same confidence region but just for those genes within the gene set. We can see how the distribution of the genes in KEGG 03030 is displaced towards the third quadrant of the plot. This indicates us that “DNA replication” is a pathway jointly down regulated in both ALL and BCR when compared to the controls in the NEG group.

Similarly we can explore a bimodal (b13) KEGG. This pattern classification indicates us that the functional block has two sub-modules of genes with opposite patterns of expression. One subset of the KEGG is up-regulated in both conditions while the other subset is down-regulated also in both dimensions analyzed.

It may be worth pointing here that, this bimodal pattern will be missed by standard univariate gene set methods, and that it may be just detected in the multidimensional analysis.
As third example we could display some KEGG enriched in one dimension but not in the other. The **yh** pattern indicates gene set over-representation just in the $Y$ axis but not in the horizontal axis. In our case, KEGG pathways with at **yh** pattern are those over expressed in $BRC$ compared to $NEG$ but not enriched in the comparison between $ALL$ and $NEG$.

Similarly an **yl** pattern indicates a down-regulation of the block in the $BRC$ vs. $NEG$ comparison but not in the $ALL$ vs. $NEG$ one.

The plot below displays one of such **yl** classified KEGGs.
Valine, leucine and isoleucine degradation

All possible multidimensional enrichment patterns are listed in the Appendix.

```r
rownames(res.md)[res.md$pat == "yh"]
## [1] "04010" "04012" "04060" "04062" "04142" "04144" "04145" "04210"
## [9] "04350" "04380" "04510" "04514" "04520" "04530" "04612" "04621"
## [17] "04630" "04662" "04666" "04670" "04672" "04722" "04810" "04940"
## [25] "05100" "05120" "05130" "05131" "05142" "05144" "05145" "05146"
## [33] "05150" "05220" "05310" "05320" "05322" "05323" "05330" "05332"
## [41] "05412" "05416"
YH <- "05332"
plotMdGsa(rindex, block = annot[[YH]], main = res.md[YH, "KEGG"])
```
3 Appendix

3.1 1.- Multidimensional Functional Classification

All possible functional block classifications in the bi-dimensional gene set analysis are:

- \( q_{1i} \): block displaced toward quadrant 1 \((0 < X & 0 < Y) \) with interaction.
- \( q_{2i} \): block displaced toward quadrant 2 \((0 > X & 0 < Y) \) with interaction.
- \( q_{3i} \): block displaced toward quadrant 3 \((0 > X & 0 > Y) \) with interaction.
- \( q_{4i} \): block displaced toward quadrant 4 \((0 < X & 0 > Y) \) with interaction.
- \( q_{1f} \): block displaced toward quadrant 1, no interaction.
- \( q_{2f} \): block displaced toward quadrant 2, no interaction.
- \( q_{3f} \): block displaced toward quadrant 3, no interaction.
- \( q_{4f} \): block displaced toward quadrant 4, no interaction.
- \( xh \): block shifted to positive \( X \) values.
- \( xl \): block shifted to negative \( X \) values.
- \( yh \): block shifted to positive \( Y \) values.
- \( yl \): block shifted to negative \( Y \) values.
- \( b_{13} \): bimodal block. Half of the genes displaced towards quadrant 1 and the other half towards quadrant 3.
- \( b_{24} \): bimodal block. Half of the genes displaced towards quadrant 2 and the other half towards quadrant 4.
- \( NS \): non significant block.

A detailed description of each of the patterns can be found in Montaner and Dopazo (2010).

The function \texttt{mdPat} in the \texttt{mdgsa} package is devised to help the user classifying bi-dimensional GSA results in such
4 Session Info

```r
sessionInfo()
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.2 LTS

## Matrix products: default
## BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so

## locale:
## [1] LC_CTYPE=en_US.UTF-8   LC_NUMERIC=C
## [3] LC_TIME=en_US.UTF-8    LC_COLLATE=C
## [5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=en_US.UTF-8   LC_NAME=C
## [9] LC_ADDRESS=C           LC_TELEPHONE=C

## attached base packages:
## [1] stats4   parallel   stats    graphics   grDevices   utils    datasets
## [8] methods  base

## other attached packages:
## [1] mdgsa_1.8.0  hgu95av2.db_3.2.3  org.Hs.eg.db_3.4.1
## [4] AnnotationDbi_1.38.0  IRanges_2.10.0  S4Vectors_0.14.0
## [7] limma_3.32.0  ALL_1.17.0   Biobase_2.36.0
## [10] BiocGenerics_0.22.0 knitr_1.15.1  BiocStyle_2.4.0

## loaded via a namespace (and not attached):
## [1] Rcpp_0.12.10   cluster_2.0.6  magrittr_1.5  lattice_0.20-35
## [5] KEGG.db_3.2.3   stringr_1.2.0  tools_3.4.0  grid_3.4.0
## [9] DBI_0.6-1   htmltools_0.3.5  yaml_2.1.14  rprojroot_1.2
## [13] digest_0.6.12  Matrix_1.2-9  memoise_1.1.0  evaluate_0.10
## [17] RSQLite_1.1-2  rmarkdown_1.4  stringi_1.1.5  G0.db_3.4.1
## [21] compiler_3.4.0  backports_1.0.5
```