tmod: Analysis of Transcriptional Modules

January Weiner

May 18th, 2015

The package `tmod` uses blood transcriptional modules described by Chaussabel et al. [2] and by Li et al. [3]. Furthermore, the package includes tools for testing the significance of enrichment of the modules as well as visualisation of the genes and modules. This vignette is a tutorial for the package.

Contents

1 The Gambia data set

1.1 Basic data analysis

In the following, we will use the Egambia data set included in the package. The data set has been generated by Maertzdorf et al. (2011)[4] and has the GEO ID GSE28623. The data is already background corrected and normalized, so we can proceed with a differential gene expression analysis. Note that only a bit over 5000 genes from the original set of over 45000 probes is included.

```
> library(limma)
> library(tmod)
> design <- cbind(Intercept=rep(1, 30), TB=rep(c(0,1), each= 15))
```

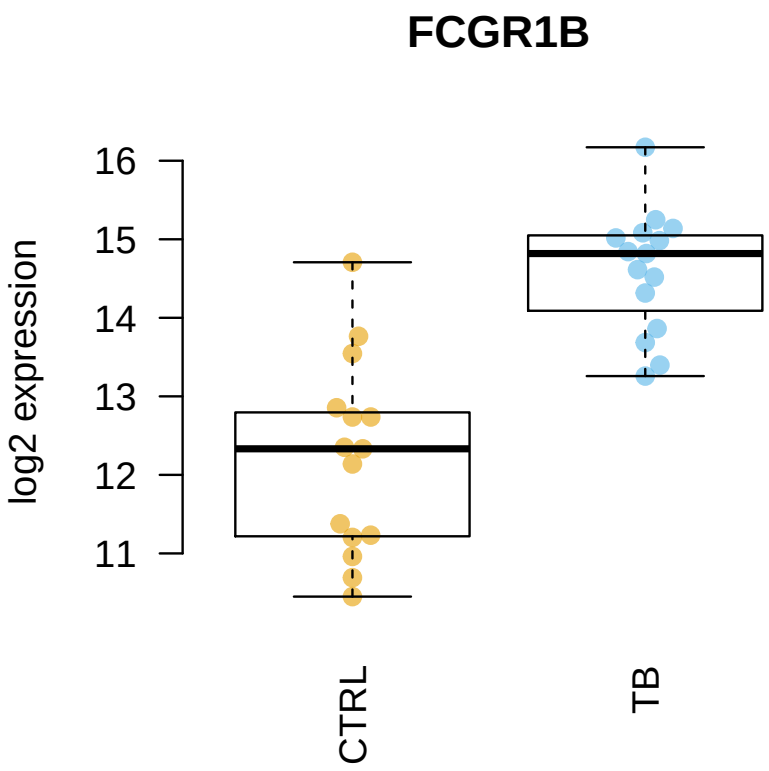
```
> E <- as.matrix(Egambia[,-c(1:3)])
> fit <- eBayes( lmFit(E, design))
> tt <- topTable(fit, coef=2, number=Inf,
   genelist=Egambia[,1:3] )
> head(tt, 10)

      GENE_SYMBOL
4178       FAM20A
20799      FCGR1B
4122        BATF2
23567     ANKRD22
20498       SEPT4
20360       CD274
2513         AIM2
24032      GOLSYN
1337         ETV7
467       SERPING1
                                                          GENE_NAME
4178                  family with sequence similarity 20, member A"
20799        Fc fragment of IgG, high affinity Ib, receptor (CD64)"
4122          basic leucine zipper transcription factor, ATF-like 2
23567                                      ankyrin repeat domain 22
20498                                                     septin 4
20360                                               CD274 molecule
2513                                           absent in melanoma 2
24032                                       Golgi-localized protein
1337                                                   ets variant 7
467   serpin peptidase inhibitor, clade G (C1 inhibitor), member 1"
         EG     logFC   AveExpr         t      P.Value
4178   54757  2.955829  4.007327  6.200637 3.423267e-07
20799   2210  2.391490 13.401207  5.946113 7.552423e-07
4122  116071  2.680837 10.398520  5.797752 1.198442e-06
23567 118932  2.763908  8.651749  5.624092 2.057601e-06
20498   5414  3.286528  4.223270  5.480564 3.215558e-06
20360  29126  2.377399  7.334747  5.463149 3.394453e-06
2513    9447  1.966342  9.933621  5.462879 3.397298e-06
24032  55638 -2.534812  2.221666 -5.362575 4.639596e-06
1337   51513  2.844012  8.075046  5.345142 4.897651e-06
467      710  2.639069  7.708228  5.150375 8.958000e-06
        adj.P.Val        B
4178  0.001898886 6.457171
20799 0.002094665 5.741043
4122  0.002215920 5.322491
23567 0.002692116 4.832003
20498 0.002692116 4.426508
20360 0.002692116 4.377314
```

```
2513   0.002692116 4.376553
24032  0.003018586 4.093323
1337   0.003018586 4.044119
467    0.004969002 3.495088
```

OK, we see some of the genes known to be prominent in the human host response to TB. We can display one of these using tmod's `showGene` function (it's just a boxplot combined with a beeswarm, nothing special):

```
> group <- rep( c("CTRL", "TB"), each=15)
> showGene(E["20799",], group,
   main=Egambia["20799", "GENE_SYMBOL"])
```



Fine, but what about the modules?

1.2 Transcriptional module analysis

There are two main functions to understand which modules are significantly enriched. The first one, tmodHGtest, is simply a hypergeometric test on two groups of genes: 'foreground' (fg), or the list of differentially expressed genes, and 'background' (bg) – the gene universe, i.e., all genes present in the analysis. The gene identifiers used currently by tmod are HGNC identifiers, and we will use the GENE_SYMBOL field from the Egambia data set.

In this particular example, however, we have almost no genes which are significantly differentially expressed after correction for multiple testing: the power of the test with 10 individuals in each group is too low. For the sake of the example, we will therefore relax our selection. Normally, I'd use a q-value threhsold of at least 0.001.

```
> fg <- tt$GENE_SYMBOL[tt$adj.P.Val < 0.05 & abs( tt$logFC ) > 1]
> res <- tmodHGtest(fg=fg, bg=tt$GENE_SYMBOL)
> res
```

```
                ID
LI.M112.0 LI.M112.0
LI.M11.0   LI.M11.0
LI.M75       LI.M75
LI.S4         LI.S4
LI.S5         LI.S5
LI.M165     LI.M165
LI.M4.3     LI.M4.3
LI.M16       LI.M16
                                                  Title
LI.M112.0                       complement activation (I)
LI.M11.0                       enriched in monocytes (II)
LI.M75                            antiviral IFN signature
LI.S4                          Monocyte surface signature
LI.S5                                DC surface signature
LI.M165         enriched in activated dendritic cells (II)
LI.M4.3   myeloid cell enriched receptors and transporters
LI.M16                     TLR and inflammatory signaling
          b  B  n    N        E      P.Value    adj.P.Val
LI.M112.0 4 11 47 4826 37.33849 2.480096e-06 0.0008581134
LI.M11.0  4 20 47 4826 20.53617 3.414323e-05 0.0059067783
LI.M75    3 10 47 4826 30.80426 9.906126e-05 0.0085687989
LI.S4     3 10 47 4826 30.80426 9.906126e-05 0.0085687989
LI.S5     4 34 47 4826 12.08010 2.957367e-04 0.0204649814
LI.M165   3 19 47 4826 16.21277 7.521410e-04 0.0394125446
LI.M4.3   2  5 47 4826 41.07234 9.112727e-04 0.0394125446
LI.M16    2  5 47 4826 41.07234 9.112727e-04 0.0394125446
```

The columns in the above table contain the following:

ID The module ID. IDs starting with "LI" come from Li et al. [3], while IDs starting with "DC" have been defined by Chaussabel et al. [2].

Title The module description

b Number of genes from the given module in the fg set

B Number of genes from the module in the bg set

n Size of the fg set

N Size of the bg set

E Enrichment, calcualted as (b/n)/(B/N)

P.Value P-value from the hypergeometric test

adj.P.Val P-value adjusted for multiple testing using the Benjamini-Hochberg correction

Well, IFN signature in TB is well known. However, the numbers of genes are not high: n is the size of the foreground, and b the number of genes in fg that belong to the given module. N and B are the respective totals – size of bg+fg and number of genes that belong to the module that are found in this totality of the analysed genes. If we were using the full Gambia data set (with all its genes), we would have a different situation.

Another approach is to sort all the genes (for example, by the respective p-value) and perform a U-test on the ranks of (i) genes belonging to the module and (ii) genes that do not belong to the module. This is a bit slower, but often works even in the case if the power of the statistical test for differential expression is low. That is, even if only a few genes or none at all are significant at acceptable thresholds, sorting them by the p-value or another similar metric can nonetheless allow to get meaningful enrichments[1]. Moreover, we do not need to set arbitrary thresholds, like p-value or logFC cutoff.

```
> l    <- tt$GENE_SYMBOL
> res2 <- tmodUtest(l)
> head(res2)
```

```
               ID                                Title
LI.M37.0 LI.M37.0 immune activation - generic cluster
LI.M37.1 LI.M37.1         enriched in neutrophils (I)
LI.S4       LI.S4          Monocyte surface signature
LI.M75     LI.M75             antiviral IFN signature
LI.M11.0 LI.M11.0          enriched in monocytes (II)
LI.M67     LI.M67           activated dendritic cells
```

[1]The rationale is that the non-significant p-values are not associated with the test that we are actually performing, but merely used to sort the gene list. Thus, it does not matter whether they are significant or not.

```
              U  N1       AUC      P.Value    adj.P.Val
LI.M37.0 352659 100 0.7462103 1.597067e-17 5.525852e-15
LI.M37.1  50280  12 0.8703781 4.530577e-06 6.569127e-04
LI.S4     43220  10 0.8974252 6.853638e-06 6.569127e-04
LI.M75    42996  10 0.8927741 8.632649e-06 6.569127e-04
LI.M11.0  74652  20 0.7766542 9.492958e-06 6.569127e-04
LI.M67    28095   6 0.9714730 3.200305e-05 1.811391e-03
```

```
> nrow(res2)
```

```
[1] 25
```

This list makes a lot of sense, and also is more stable than the other one: it does not depend on modules that contain just a few genes. Since the statistics is different, the b, B, n, N and E columns in the output have been replaced by the following:

U The Mann-Whitney U statistics

N1 Number of genes in the module

AUC Area under curve – a measure of the effect size

There are two tests in tmod which both operate on an ordered list of genes: `tmodUtest` and `tmodCERNOtest`. The U test is simple, however has two main issues. Firstly, it detects enrichments as well as depletions – that is, modules which are enriched at the bottom of the list (e.g. modules which are never, ever regulated in a particular comparison) will be detected as well. This is often undesirable. Secondly, large modules will be reported as significant even if the actual effect size (i.e., AUC) is modest or very small, just because of the sheer number of genes in a module. Unfortunately, also the reverse is true: modules with a small number of genes, even if they consist of highly up- or down-regulated genes from the top of the list will not be detected.

The CERNO test, described by Yamaguchi et al. [9], is based on Fisher's method of combining probabilities. In summary, for a given module, the ranks of genes from the module are logarithmized, summed and multiplied by -2:

$$f_{CERNO} = -2 \cdot \sum_{i=1}^{N} \ln \frac{R_i}{N_{tot}}$$

This statitic has the χ^2 distribution with $2 \cdot N$ degrees of freedom, where N is the number of genes in a given module and N_{tot} is the total number of genes [9].

The CERNO test is actually much more practical than the U test for most purposes.

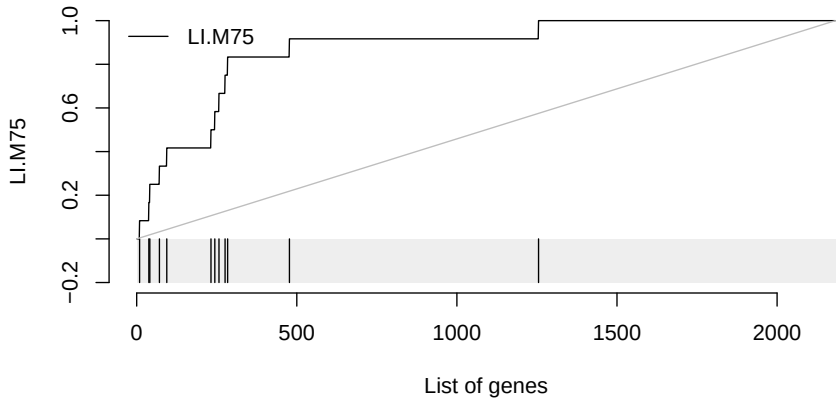
```
> l    <- tt$GENE_SYMBOL
> res2 <- tmodCERNOtest(l)
> head( res2 )
```

```
                ID                                Title
LI.M37.0   LI.M37.0 immune activation - generic cluster
LI.M11.0   LI.M11.0          enriched in monocytes (II)
LI.S4         LI.S4          Monocyte surface signature
LI.M112.0 LI.M112.0           complement activation (I)
LI.M75       LI.M75             antiviral IFN signature
LI.M16       LI.M16      TLR and inflammatory signaling
              cerno  N1       AUC      cES      P.Value
LI.M37.0  426.35781 100 0.7462103 2.131789 1.824844e-18
LI.M11.0  113.80864  20 0.7766542 2.845216 5.255069e-09
LI.S4      76.37298  10 0.8974252 3.818649 1.606057e-08
LI.M112.0  73.67987  11 0.8455773 3.349085 1.722322e-07
LI.M75     65.29854  10 0.8927741 3.264927 1.045914e-06
LI.M16     46.33475   5 0.9790500 4.633475 1.247201e-06
             adj.P.Val
LI.M37.0  6.313962e-16
LI.M11.0  9.091270e-07
LI.S4     1.852319e-06
LI.M112.0 1.489809e-05
LI.M75    7.192190e-05
LI.M16    7.192190e-05
```

Here, the results are similar, however CERNO test was able to detect another module – "TLR and inflammatory signaling". Although only 5 genes are in this module (which is why U test could not detect it), the genes are all on the top of the list of the differentially regulated genes.

Let us now investigate in more detail the module LI.M75, the antiviral interferon signature. We can use the `evidencePlot` function to see how the module is enriched in the list `l`.

```
> evidencePlot(l, "LI.M75")
```



In essence, this is a receiver-operator characteristic (ROC) curve, and the area under the curve (AUC) is related to the U-statistic, from which the P-value in the tmodUtest is calculated, as $\text{AUC} = \frac{U}{n_1 \cdot n_2}$. Both the U statistic and the AUC are reported. Moreover, the AUC can be used to calculate effect size according to the Wendt's formula[8] for rank-biserial correlation coefficient:

$$r = 1 - \frac{2 \cdot U}{n_1 \cdot n_2} = 1 - 2 \cdot \text{AUC}$$

In the above diagram, we see that nine out of the 10 genes that belong to the LI.M75 module and which are present in the Egambia data set are ranked among the top 1000 genes (as sorted by p-value).

2 Using other sets of modules

By default, tmod uses the modules published by Li et al. [3] (LI). A second set of modules was published by Chaussabel et al. [2] (DC); new module definitions were described by Banchereau et al. [1] and can be found on a public website[2].

Depending on the `mset` parameter to the test functions, either the LI or DC sets are used, or both, if the `mset=all` has been specified.

```
> l    <- tt$GENE_SYMBOL
> res2 <- tmodUtest(l, mset="all")
> head( res2 )
```

```
               ID                                Title
LI.M37.0 LI.M37.0 immune activation - generic cluster
DC.M4.2   DC.M4.2                         Inflammation
```

[2] http://www.biir.net/public_wikis/module_annotation/G2_Trial_8_Modules

```
DC.M1.2   DC.M1.2                          Interferon
DC.M3.2   DC.M3.2                         Inflammation
DC.M5.15 DC.M5.15                          Neutrophils
DC.M7.29 DC.M7.29                         Undetermined
              U  N1       AUC      P.Value    adj.P.Val
LI.M37.0 352659 100 0.7462103 1.597067e-17 9.678227e-15
DC.M4.2   91352  20 0.9503953 1.674762e-12 5.074530e-10
DC.M1.2   73612  17 0.9004196 5.703006e-09 9.623646e-07
DC.M3.2   96366  24 0.8361620 6.352241e-09 9.623646e-07
DC.M5.15  65289  16 0.8483498 7.240084e-07 8.774982e-05
DC.M7.29  77738  20 0.8087599 9.084521e-07 9.175366e-05
```

As you can see, the information contained in both module sets is partially redundant.

3 Functional multivariate analysis

Transcriptional modules can help to understand the biological meaning of the calculated multivariate transformations. For example, consider a principal component analysis (PCA), visualised using the pca3d package [6]:

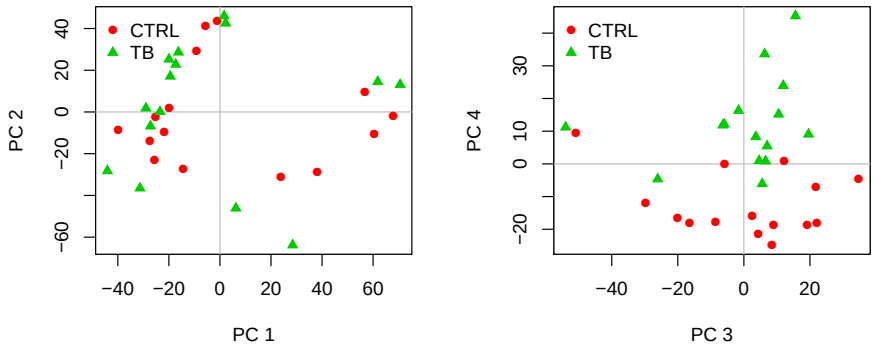
```
> library(pca3d)
> pca <- prcomp(t(E), scale.=TRUE)
> par(mfrow=c(1, 2))
> l<-pca2d(pca, group=group)

Legend:
----------------------
 group:  color,  shape
----------------------
  CTRL:    red,     16
    TB: green3,     17

> cols <- as.character(l$colors)
> legend("topleft", as.character(l$groups),
        pch=l$shapes,
        col=cols, bty="n")
> l<-pca2d(pca, group=group, components=3:4)

Legend:
----------------------
 group:  color,  shape
----------------------
  CTRL:    red,     16
    TB: green3,     17
```

```
> legend("topleft", as.character(l$groups),
        pch=l$shapes,
        col=cols, bty="n")
> par(mfrow=c(1, 1))
```



The fourth component looks really interesting. Does it correspond to the modules which we have found before? Each principal component is, after all, a linear combination of gene expression values multiplied by weights (or scores) which are constant for a given component. The i-th principal component for sample j is given by

$$PC_{i,j} = \sum_k w_{i,k} \cdot x_{k,j}$$

where k is the index of the variables (genes in our case), $w_{i,k}$ is the weight associated with the i-th component and the k-th variable (gene), and $x_{k,j}$ is the value of the variable k for the sample j; that is, the gene expression of gene k in the sample j. Genes influence the position of a sample along a given component the more the larger their absolute weight for that component.

For example, on the right-hand figure above, we see that samples which were taken from TB patients have a high value of the principal component 4; the opposite is true for the healthy controls. The genes that allow us to differentiate between these two groups will have very large, positive weights for genes highly expressed in TB patients, and very large, negative weights for genes which are highly expressed in NID, but not TB.

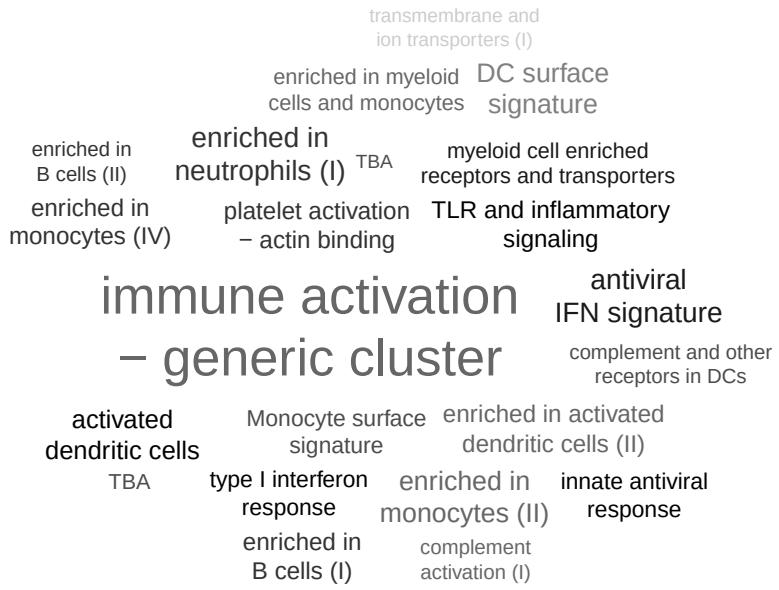
We can sort the genes by their weight in the given component, since the weights are stored in the pca object in the "rotation" slot, and use the tmodUtest function to test for enrichment of the modules.

```
> o <- order(abs(pca$rotation[,4]), decreasing=TRUE)
> l <- Egambia$GENE_SYMBOL[o]
> res <- tmodUtest(l)
> head(res)
```

```
               ID                                Title
LI.M37.0 LI.M37.0 immune activation - generic cluster
LI.M37.1 LI.M37.1         enriched in neutrophils (I)
LI.M75     LI.M75             antiviral IFN signature
LI.M11.0 LI.M11.0          enriched in monocytes (II)
LI.S5       LI.S5                DC surface signature
LI.M67     LI.M67           activated dendritic cells
              U  N1       AUC      P.Value    adj.P.Val
LI.M37.0 339742 100 0.7188785 3.133111e-14 1.084056e-11
LI.M37.1  50096  12 0.8671929 5.405722e-06 6.700097e-04
LI.M75    43379  10 0.9007267 5.809333e-06 6.700097e-04
LI.M11.0  74343  20 0.7734395 1.185187e-05 1.025187e-03
LI.S5    115007  34 0.7058762 1.711493e-05 1.184353e-03
LI.M67    28291   6 0.9782503 2.506730e-05 1.445548e-03
```

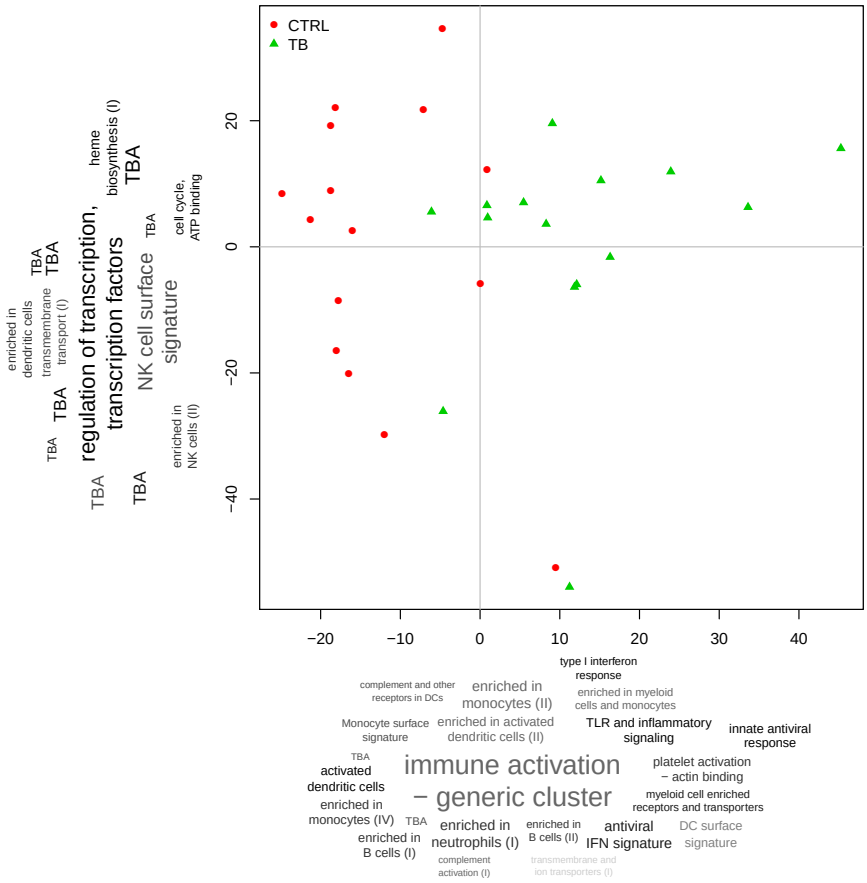
Perfect, this is what we expected: we see that the neutrophil / interferon signature which is the hallmark of the TB biosignature. We can visualise this list of results using the tagcloud package [7]. P-Values will be represented by the size of the tags, while AUC – which is a proxy for the effect size – will be shown by the color of the tag, from grey (AUC=0.5, random) to black (1):

```
> library(tagcloud)
> w <- -log10(res$P.Value)
> c <- smoothPalette(res$AUC, min=0.5)
> tags <- strmultline(res$Title)
> tagcloud(tags, weights=w, col=c)
```



We can now annotate the PCA axes using the tag clouds; however, see below for a shortcut in tmod.

```
> par(mar=c(1,1,1,1))
> o3 <- order(abs(pca$rotation[,3]), decreasing=TRUE)
> l3 <- Egambia$GENE_SYMBOL[o3]
> res3 <- tmodUtest(l3)
> layout(matrix(c(3,1,0,2),2,2,byrow=TRUE),
   widths=c(0.3, 0.7), heights=c(0.7, 0.3))
> # note -- PC4 is now x axis!!
> l<-pca2d(pca, group=group, components=4:3)
> cols <- as.character(l$colors)
> legend("topleft",
   as.character(l$groups),
   pch=l$shapes,
   col=cols, bty="n")
> tagcloud(tags, weights=w, col=c, fvert= 0)
> tagcloud(strmultline(res3$Title),
   weights=-log10(res3$P.Value),
   col=smoothPalette(res3$AUC, min=0.5),
   fvert=1)
```



As mentioned previously, there is a way of doing it all with tmod much more quickly. Note that `plot.params` are just parameters which will be passed to the pca2d function.

```
> tmodPCA(pca,
   genes=Egambia$GENE_SYMBOL,
   components=3:4,
   plot.params=list(group=group))

Legend:
----------------------
 group:  color,  shape
----------------------
  CTRL:    red,     16
    TB: green3,     17
```

4 Accessing the tmod data

The `tmod` package stores its data in two data frames and two lists. These objects are loaded when the package is attached via library(), and can be immediately used without calling data(). The names mimick the various environments from Annotation.dbi packages, but currently the objects are just two lists and two data frames.

tmodMODULES is a data frame which contains general module information as defined in the supplementary materials for Li et al. [3] and Chaussabel et al. [2]

tmodGENES is a data frame which contains general gene information, including columns with HGNC ("primary"), as well as ENTREZ and REFSEQ identifiers.

tmodMODULES2GENES is a list with module IDs (same as in the "ID" column of tmodMODULES) as names. Every element of the list is a character vector with IDs ("primary" column of tmodGENES) of the genes which are included in this module.

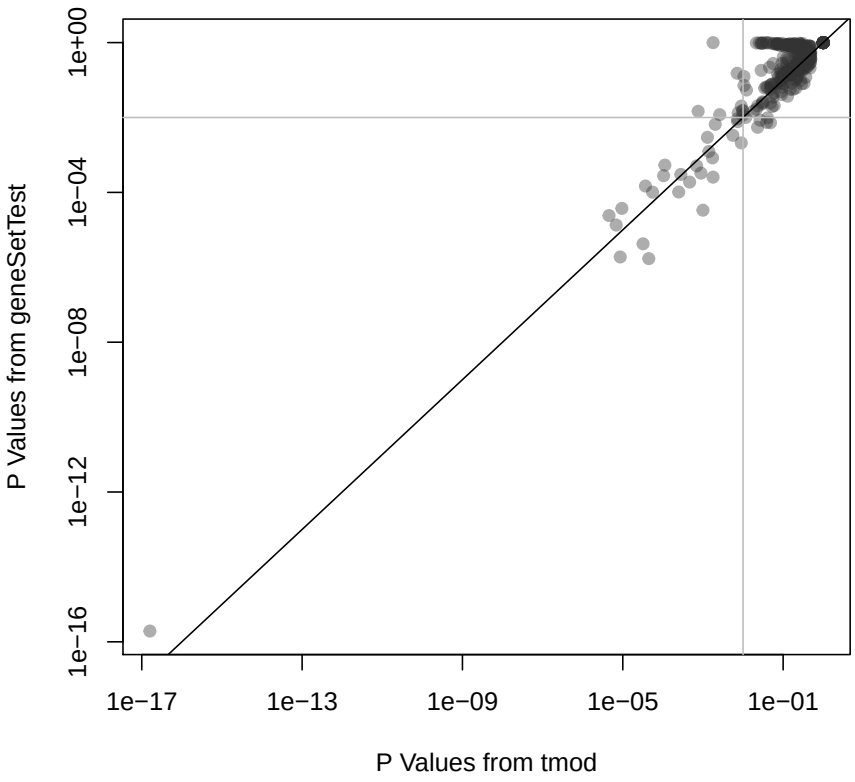
tmodGENES2MODULES is a list with gene IDs (same as in the "primary" column of tmodGENES) as names. Every element of the list is a character vector with IDs of the modules in which the gene is found.

Using these variables, one can apply any other tool for the analysis of enriched module sets available, for example, the `geneSetTest` function from the limma package (Smyth et al. [5]). We will first run `tmodUtest` setting the `qval` to `Inf` to get p-values for all modules. Then, we apply the `geneSetTest` function to each module:

```
> res <- tmodUtest(tt$GENE_SYMBOL, qval=Inf)
> gstest <- function(x) {
   sel <- tt$GENE_SYMBOL %in% tmodMODULES2GENES[[x]]
   geneSetTest(sel, tt$logFC)
 }
> gst <- sapply(res$ID, gstest)
```

Are the results of both statistical approaches similar? `tmod` uses a very simple statistical test. The approach from `geneSetTest` is more complex, but similar in principle.

```
> plot(res$P.Value, gst,
   log="xy", pch=19,
   col="#33333366",
   xlab="P Values from tmod",
   ylab="P Values from geneSetTest")
> abline(0,1)
> abline(h=0.01, col="grey")
> abline(v=0.01, col="grey")
```



On the plot above, the p-values from `tmod` are plotted against the p-values from `geneSetTest`. As you can see, in this particular example, both methods give very similar results.

5 Using and creating custom sets of modules

It is possible to use any kind of arbitrary or custom gene set definitions. These custom definition of gene sets takes form of a list which is then provided as the `mset` parameter to the test functions. The list in question must have the following members:

MODULES A data frame which contains at least the columns "ID" and "Title". The IDs must correspond to the names of MODULES2GENES.

GENES A data frame which contains at least the column "ID". The gene IDs must correspond to the gene IDs used in MODULES2GENES.

MODULES2GENES A list. The names of the list are the IDs from the MODULES data frame. The items in the list are character vectors with names of the genes that are associated with each module.

Here is a minimal definition of such a set:

```
> mymset <- list(
   MODULES=data.frame(ID=c("A", "B"),
                      Title=c("A title",
                                    "B title")),
   GENES=data.frame(ID=c( "G1", "G2", "G3", "G4" )),
   MODULES2GENES=list(
     A=c("G1", "G2"),
     B=c("G3", "G4"))
 )
```

Whether the gene IDs are Entrez, or something else entirely does not matter, as long as they matched the provided input to the test functions.

5.1 MSigDB

The MSigDB database from the Broad institute is an interesting collection of gene sets (actually, multiple collections). Unfortunately, MSigDB cannot be distributed or even accessed without a free registration, which imposes a heavy limination on third party tools such as tmod. Use the following guide to download and parse the database such that you can use it with R and tmod.

First, you will need to download the MSigDB in XML format[3]. This file can be accessed at the URL `http://www.broadinstitute.org/gsea/msigdb/download_file.jsp?filePath=/resources/msigdb/5.0/msigdb_v5.0.xml` – follow the link, register and log in, and save the file on your disk (roughly 65MB).

Importing MSigDB is easy – tmod features a function specifically for that purpose. Once you have downloaded the MSigDB file, you can create the tmod-compatible R object with one command[4]:

```
> msig <- tmodImportMsigDB("msigdb_v5.0.xml")
```

That's it – now you can use the full MSigDB for enrichment tests:

```
> res <- tmodCERNOtest(tt$GENE_SYMBOL, mset=msig )
> head(res)
```

[3] Note that even if you register with MSig, it is not possible to download the database from R

[4] MSigDB gene sets can be also downloaded as "GMT" files. This format contains less information and is therefore less usable. However, the tmod function tmodImportMsigDB() can also use this format, look up the manual page

```
         ID                                          Title
M3408 M3408          GSE1432_CTRL_VS_IFNG_24H_MICROGLIA_DN
M3010 M3010                            HECKER_IFNB1_TARGETS
M3286 M3286    GSE13485_CTRL_VS_DAY3_YF17D_VACCINE_PBMC_DN
M3288 M3288    GSE13485_CTRL_VS_DAY7_YF17D_VACCINE_PBMC_DN
M3311 M3311 GSE13485_PRE_VS_POST_YF17D_VACCINATION_PBMC_DN
M3347 M3347               GSE14000_UNSTIM_VS_4H_LPS_DC_DN
         cerno N1       AUC      cES      P.Value
M3408 239.0983 39 0.8014227 3.065363 2.967858e-18
M3010 244.1219 43 0.8459807 2.838626 4.555892e-17
M3286 247.0915 45 0.7293732 2.745461 1.408943e-16
M3288 272.2570 54 0.7222067 2.520898 3.626792e-16
M3311 229.4948 41 0.7272625 2.798718 6.715323e-16
M3347 272.0698 55 0.7334883 2.473362 9.792737e-16
         adj.P.Val
M3408 2.501904e-14
M3010 1.920308e-13
M3286 3.959129e-13
M3288 7.643464e-13
M3311 1.132204e-12
M3347 1.375880e-12
```

The results are quite typical for MSigDB, which is quite abundant with similar or overlapping gene sets. As the first results, we see, again, interferon response, as well as sets of genes which are significantly upregulated after yellow fever vaccination – and which are also interferon related. We might want to limit our analysis only to the 50 "hallmark" module categories:

```
> sel <- msig$MODULES$Category == "H"
> tmodCERNOtest(tt$GENE_SYMBOL,
   modules=msig$MODULES$ID[sel],
   mset=msig )
```

```
         ID                                       Title
M5913 M5913         HALLMARK_INTERFERON_GAMMA_RESPONSE
M5921 M5921                       HALLMARK_COMPLEMENT
M5911 M5911         HALLMARK_INTERFERON_ALPHA_RESPONSE
M5946 M5946                     HALLMARK_COAGULATION
M5890 M5890           HALLMARK_TNFA_SIGNALING_VIA_NFKB
M5930 M5930 HALLMARK_EPITHELIAL_MESENCHYMAL_TRANSITION
M5932 M5932             HALLMARK_INFLAMMATORY_RESPONSE
M5953 M5953                 HALLMARK_KRAS_SIGNALING_UP
M5892 M5892           HALLMARK_CHOLESTEROL_HOMEOSTASIS
          cerno N1       AUC      cES      P.Value
M5913 221.68317 41 0.7786936 2.703453 8.505170e-15
M5921 217.81028 56 0.6979148 1.944735 8.607634e-09
M5911 108.39559 20 0.7563566 2.709890 3.192325e-08
```

```
M5946 179.24580 50 0.6779481 1.792458 1.966824e-06
M5890 148.95123 47 0.6484665 1.584588 2.657694e-04
M5930 212.53461 73 0.6371808 1.455717 2.701053e-04
M5932 184.53035 62 0.6206393 1.488148 3.457724e-04
M5953 221.76208 82 0.6046637 1.352208 1.790956e-03
M5892  49.14641 14 0.6138968 1.755229 8.040562e-03
         adj.P.Val
M5913 4.252585e-13
M5921 2.151909e-07
M5911 5.320542e-07
M5946 2.458530e-05
M5890 2.250878e-03
M5930 2.250878e-03
M5932 2.469803e-03
M5953 1.119347e-02
M5892 4.466979e-02
```

We see both – the prominent ínterferon response and the complement activation. Also, in addition, TNF-α signalling via NF-$\kappa\beta$.

5.2 Manual creation of tmod module objects: MSigDB

For the purposes of an example, the code below shows how to parse the XML MSigDB file using the R package XML. Essentially, this is the same code that tmodImportMsigDB is using:

```
> library(XML)
> foo  <- xmlParse( "/home/january/Projects/R/pulemodule/vignette/msigdb_v5.0.xml" )
> foo2 <- xmlToList(foo)
```

There are over 10,000 "gene sets" (equivalent to modules in tmod) defined. Each member of foo2 is a named character vector:

```
> path1 <- foo2[[1]]
> class(path1)

[1] "character"

> names(path1)

 [1] "STANDARD_NAME"         "SYSTEMATIC_NAME"
 [3] "HISTORICAL_NAMES"      "ORGANISM"
 [5] "PMID"                  "AUTHORS"
 [7] "GEOID"                 "EXACT_SOURCE"
 [9] "GENESET_LISTING_URL"   "EXTERNAL_DETAILS_URL"
[11] "CHIP"                  "CATEGORY_CODE"
[13] "SUB_CATEGORY_CODE"     "CONTRIBUTOR"
```

```
[15] "CONTRIBUTOR_ORG"      "DESCRIPTION_BRIEF"
[17] "DESCRIPTION_FULL"     "TAGS"
[19] "MEMBERS"              "MEMBERS_SYMBOLIZED"
[21] "MEMBERS_EZID"         "MEMBERS_MAPPING"
[23] "FOUNDER_NAMES"        "REFINEMENT_DATASETS"
[25] "VALIDATION_DATASETS"
```

For our example analysis, we will use only human gene sets. We further need to make sure there are no NULLs in the list.

```
> orgs <- sapply(foo2, function(x) x["ORGANISM"])
> unique(orgs)
> foo3 <- foo2[ orgs == "Homo sapiens" ]
> foo3 <- foo3[ ! sapply(foo3, is.null) ]
```

Next, construct the MODULES data frame. We will use four named fields for each vector, which contain the ID (systematic name), description, category and subcategory:

```
> msig <- list()
> msig$MODULES <- t(sapply(foo3,
   function(x)
     x[ c("SYSTEMATIC_NAME", "STANDARD_NAME", "CATEGORY_CODE", "SUBCATEGORY_CODE") ]))
> colnames(msig$MODULES) <- c( "ID", "Title", "Category", "Subcategory" )
> rownames(msig$MODULES) <- msig$MODULES[,"ID"]
> msig$MODULES <- data.frame(msig$MODULES, stringsAsFactors=FALSE)
```

Then, we create the modules to genes mapping and the GENES data frame. For this, we use the `MEMBERS_SYMBOLIZED` field, which is a comma separated list of gene symbols belonging to a particular module:

```
> msig$MODULES2GENES <- lapply(foo3,
   function(x) strsplit( x["MEMBERS_SYMBOLIZED"], "," )[[1]])
> names(msig$MODULES2GENES) <- msig$MODULES$ID
> msig$GENES <- data.frame( ID=unique(unlist(msig$MODULES2GENES)))
```

From now on, you can use msig with tmod.

5.3 Manual creation of tmod module sets: Wikipathways

Below is an example of how to use the pathway definitions from WikiPathways[5]. First, we download the data (human pathways) and clean it up:

```
> download.file(
 "http://www.wikipathways.org//wpi/batchDownload.php?species=Homo%20sapiens&fileType=txt",
   destfile="human.zip")
```

[5] http://www.wikipathways.org/

```
> files <- unzip( "human.zip", list=T)
> files$ID    <- gsub( ".*_(WP[0-9]*)_.*", "\\1", files$Name )
> files$Title <- gsub( "(.*)_WP[0-9]*_.*", "\\1", files$Name )
```

Since each pathway is in a separate file in the zip archive we downloaded, we have to read each file separately. Below, we create a list, `p2GENES`, which maps the modules to the corresponding genes. To make it consistent, I decided to use gene symbols rather than the Entrez numbers (just because it makes the interpretation of results a bit easier), but actually that is not necessary: tmod does not care what gene symbols are used, as long as the mappings between genes and modules are consistent, and as long as the same identifiers are used in the lists of genes.

Furthermore, note that we filter out anything that is not an ENTREZ gene identifier. This gets rid of entities which are not genes (e.g. biochemical compounds), but also of some genes.

```
> library(org.Hs.eg.db)
> p2GENES <- sapply( files$Name, function(fn) {
   foo <- read.csv( unz( "human.zip",
   filename= fn ), sep="\t" )
   ids <- foo$Identifier[ foo$Identifier %in% ls( org.Hs.egSYMBOL ) ]
   unique(unlist(mget(as.character(ids), org.Hs.egSYMBOL)))
 })
> names(p2GENES) <- files$ID
```

p2GENES is the first of three objects that we need to create. The next one is a data frame containing module definitions. We also calculate the number of associated genes and select pathways that have at least 5 associated ENTREZ genes:

```
> pathways <- data.frame( ID=files$ID,
   Title=files$Title,
   stringsAsFactors=FALSE )
> pathways$N <- sapply(p2GENES, length)
> pathways$URL <-
   paste0("http://www.wikipathways.org/index.php/Pathway:",
   pathways$ID )
> sel <- pathways$N > 4
> pathways <- pathways[ sel, ]
```

Finally, we need a data frame containing the gene IDs[6] and we are good to go: we can build the list that will be the value of the `mset` parameter:

[6]The reason why it is a data frame and not simply a vector of IDs and why it is necessary at all since it can be calculated from the module to gene mapping: it is faster and allows for more than just one gene symbol

```
> GENES <- data.frame( ID=unique(unlist(p2GENES)))
> Hspaths <- list( MODULES=pathways,
       MODULES2GENES=p2GENES,
       GENES=GENES )
```

We can now use the tmodCERNOtest to see whether it works:

```
> tmodCERNOtest(tt$GENE_SYMBOL, mset=Hspaths)
```

```
         ID                                        Title
WP558 WP558      Hs_Complement_and_Coagulation_Cascades
WP545 WP545 Hs_Complement_Activation,_Classical_Pathway
          cerno N1       AUC      cES      P.Value
WP558 107.73082 28 0.6418746 1.923765 4.008176e-05
WP545  45.65536  9 0.7465689 2.536409 3.330196e-04
        adj.P.Val
WP558 0.008617578
WP545 0.035799604
```

Nice – the complement pathway was also found before, when using the default data set. Unfortunately, we don't see anything else: WikiPathways are more oriented on metabolic pathways, while the blood transcriptional modules are particularly good for analyzing immune responses. However, if we were to test a specific hypothesis, we would select modules related to interferon response:

```
> sel <- grep( "Interferon",
   Hspaths$MODULES$Title, ignore.case=T )
> tmodCERNOtest(tt$GENE_SYMBOL, mset=Hspaths,
   modules=Hspaths$MODULES$ID[sel])
```

```
         ID                                    Title   cerno
WP619 WP619 Hs_Type_II_interferon_signaling_(IFNG) 42.1566
      N1       AUC      cES     P.Value   adj.P.Val
WP619  9 0.7050031 2.342033 0.001051527 0.003154582
```

Since the number of tests is lower, the type-II interferon signalling is now significant.

References

[1] Romain Banchereau, Alejandro Jordan-Villegas, Monica Ardura, Asuncion Mejias, Nicole Baldwin, Hui Xu, Elizabeth Saye, Jose Rossello-Urgell, Phuong Nguyen, Derek Blankenship, et al. Host immune transcriptional profiles reflect the variability in clinical disease manifestations in patients with staphylococcus aureus infections. *PLoS One*, 7(4):e34390, 2012.

[2] Damien Chaussabel, Charles Quinn, Jing Shen, Pinakeen Patel, Casey Glaser, Nicole Baldwin, Dorothee Stichweh, Derek Blankenship, Lei Li, Indira Munagala, et al. A modular analysis framework for blood genomics studies: application to systemic lupus erythematosus. *Immunity*, 29(1):150–164, 2008.

[3] Shuzhao Li, Nadine Rouphael, Sai Duraisingham, Sandra Romero-Steiner, Scott Presnell, Carl Davis, Daniel S Schmidt, Scott E Johnson, Andrea Milton, Gowrisankar Rajam, et al. Molecular signatures of antibody responses derived from a systems biology study of five human vaccines. *Nature immunology*, 2013.

[4] Jeroen Maertzdorf, Martin Ota, Dirk Repsilber, Hans J Mollenkopf, January Weiner, Philip C Hill, and Stefan HE Kaufmann. Functional correlations of pathogenesis-driven gene expression signatures in tuberculosis. *PloS one*, 6(10):e26938, 2011.

[5] Gordon K Smyth. Limma: linear models for microarray data. In R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber, editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, pages 397–420. Springer, New York, 2005.

[6] January Weiner. *pca3d: Three dimensional PCA plots*, 2013. R package version 0.4.

[7] January Weiner. *tagcloud: Tag Clouds*, 2014. R package version 0.5.

[8] Hans W Wendt. Dealing with a common problem in social science: A simplified rank-biserial coefficient of correlation based on the u statistic. *European Journal of Social Psychology*, 2(4):463–465, 1972.

[9] Ken D Yamaguchi, Daniel L Ruderman, Ed Croze, T Charis Wagner, Sharlene Velichko, Anthony T Reder, and Hugh Salamon. Ifn-β-regulated genes show abnormal expression in therapy-naïve relapsing–remitting ms mononuclear cells: Gene expression analysis employing all reported protein–protein interactions. *Journal of neuroimmunology*, 195(1):116–120, 2008.