hpgltools

atb abelew@gmail.com

2015-07-31

Hpgltools: Stupid R tricks.

The following block shows how I handle autloading requisite libraries for my code. This makes it easier for me to download/install the R requirements on a new computer, something which I have found myself needing to do more than I would have guessed.

```r
## This block serves to load requisite libraries and set some options.
library("hpgltools")
## To set up an initial vignette, use the following line:
## devtools::use_vignette("hpgltools")
autoloads_all()
opts_knit$set(progress=TRUE, verbose=TRUE, purl=FALSE, error=TRUE, stop_on_error=FALSE, fig.width=7, fi
options(java.parameters="-Xmx8g")  ## used for xlconnect -- damn 4g wasn't enough
theme_set(theme_bw(base_size=10))
set.seed(1)
```

Rendering the vignette

The following block has a few lines I use to load data, save it, and render pdf/html reports. I do this under the veritable editor, 'emacs,' with the key combination "Control-c, Control-n" for each line I want to evaluate in R, or "Control-c, Control-c" for a paragraph.

```r
load("RData")
rm(list=ls())
save(list=ls(all=TRUE), file="RData")
render("hpgltools.Rmd", output_format="pdf_document")
render("hpgltools.Rmd", output_format="html_document")
```

Tasks that hpgltools helps me perform

This code was written to speed up and simplify a few specific tasks:

- Reading RNA sequencing count tables (in R/count_tables.R)
- Normalization of data (R/normalization.R)
- Graphing metrics of data to check and evaludate batch effects (R/plots.R)
- Performing contrasts of the data using voom/limma (R/misc_functions.R)
- Plotting RNA abundances by condition/batch (R/plots.R)
- Simplifying ontology/KEGG searches (R/ontology.R)

The following paragraphs will attempt to show how I use it.

Annotation information

Every RNA sequencing experiment I have played with has required a different handling of the genome's annotation. Most, but not all, have kept the data of interest in a gff file. Here is an example of how I process one of those files and make a data frame of genes as well as tooltips, which will be used for googleVis graphs later. In every experiment I have played with, I make a 'reference' directory into which I copy the current annotation data, this way I have a consistent and known version of the annotation. In the example below, this is the TriTrypDB version 8.1 of the T. cruzi genome.

```
tcruzi_annotations = import.gff3("reference/gff/clbrener_8.1_complete.gff.gz")
annotation_info = as.data.frame(tcruzi_annotations)

genes = annotation_info[annotation_info$type=="gene",]
gene_annotations = genes
rownames(genes) = genes$Name
tooltip_data = genes
tooltip_data = tooltip_data[,c(11,12)]
tooltip_data$tooltip = paste(tooltip_data$Name, tooltip_data$description, sep=": ")
tooltip_data$tooltip = gsub("\\+", " ", tooltip_data$tooltip)
rownames(tooltip_data) = tooltip_data$Name
tooltip_data = tooltip_data[-1]
tooltip_data = tooltip_data[-1]
colnames(tooltip_data) = c("name.tooltip")
head(tooltip_data)
```

Reading count tables

In Dr. El-Sayed's lab, there is a very specific naming convention for RNA sequencing experiments. Every sequencing run has an 'HPGL' (host pathogen genomics lab) identifier. All experiments have associated metadata, including the condition in the experiment, the batch, bioanalyzer reports, etc. When I play with data, I keep all this information in a csv file 'samples.csv' and the processed count-tables for the experiment in a specific directory: processed_data/. Therefore, I have a couple functions which automate the import of data into R in the hopes that no mistakes are made.

Here is an example from a recent experiment.

```
samples = read.csv("data/all_samples.csv")
knitr::kable(head(samples))
```

Sample.ID	Type	Stage	batch	Media	SRA	Reads.Passed	ncRNA	X..ncRNA	Remaining	Genome	X..G
HPGL0406	WT	EL	1	THY	NA	19026277	353992	1.86%	18672285	17810587	95.3
HPGL0407	WT	EL	2	THY	NA	15074073	259613	1.72%	14814460	14334043	96.7
HPGL0408	mga	EL	1	THY	NA	17112233	293752	1.72%	16818481	15769581	93.7
HPGL0409	mga	EL	2	THY	NA	18298278	339862	1.86%	17958416	16553148	92.1
HPGL0149	WT	LL	1	THY	NA	39107368	8055417	20.60%	31051951	26285560	84.6
HPGL0150	WT	LL	2	THY	NA	35429033	3705275	10.46%	31723758	30012962	94.6

Since I didn't want to copy over all my count tables, you, dear reader, will have to trust that there is a file for each entry in the above table which corresponds to the Sample.ID. These may be organized by sample name or condition. The following code shows how I create an expressionset and fill it with the count data.

```r
example_data = counts(make_exampledata(ngenes=10000, columns=24))
## create_expt() usually expects that there are a bunch of count tables
## from htseq in the directory: processed_data/count_tables/
## These may be organised in separate directories by condition(type)
## in one directory each by sample.  By default, this assumes they will be
## named sample_id.count.gz, but this may be changed with the suffix argument.
all_expt = create_expt("data/all_samples.csv", count_dataframe=example_data)
```

```
## [1] "This function needs the conditions and batches to be an explicit column in the sample sheet."
## [1] "Please note that thus function assumes a specific set of columns in the sample sheet:"
## [1] "The most important ones are: Sample.ID, Stage, Type."
## [1] "Other columns it will attempt to create by itself, but if"
## [1] "batch and condition are provided, that is a nice help."
```

Examining data

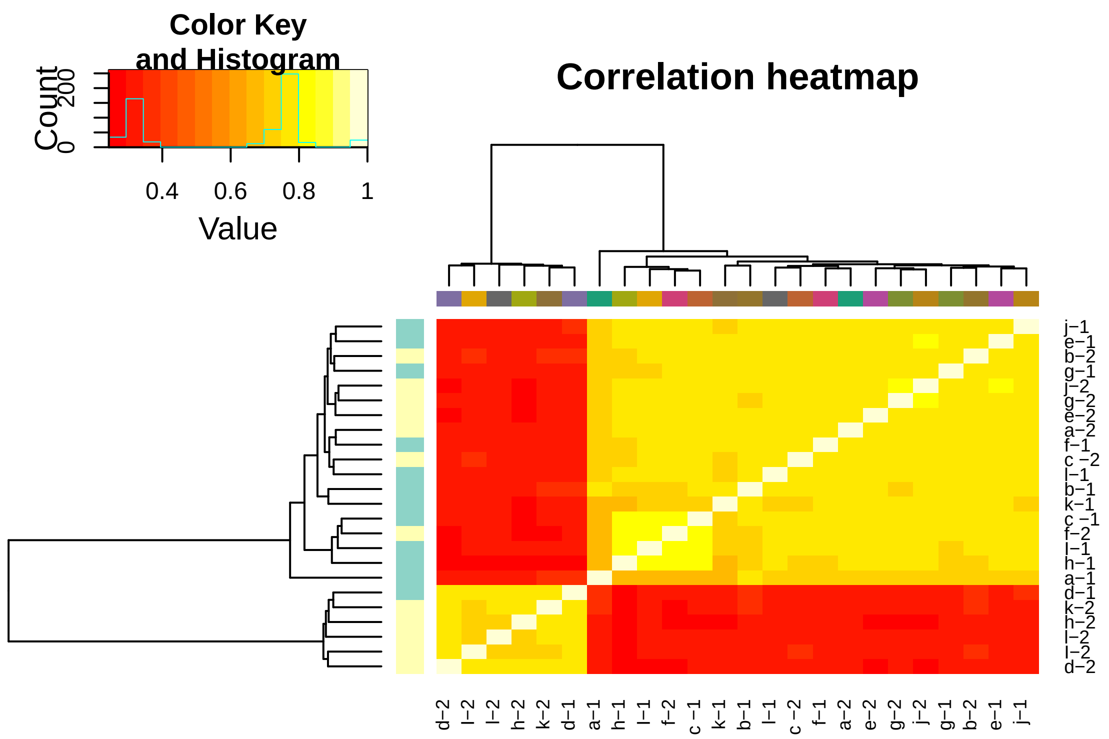
Once the data is read in, the first task is always to look at it and evaluate for batch effects and thus decide what to do about them. However, different normalization methods are appropriate in different data sets, therefore I have some functions which attempt to make this easier. For this, I will make a dummy data set using limma's makeExampleData()

```r
## graph_metrics() performs the following:
## runs a libsize plot, non-zero genes plot, boxplot, correlation/distance heatmaps, and pca plots
## It performs a normalization of the data (log2(quantile(cpm)) by default), and does it again
## It then uses limma's removeBatchEffect() to make a stab at removing batch effect, and does it again.

## An important thing to remember: the data from makeExampleData() is not very interesting, so the resul
## plots are also not interesting...
fun = graph_metrics(expt=all_expt)
```

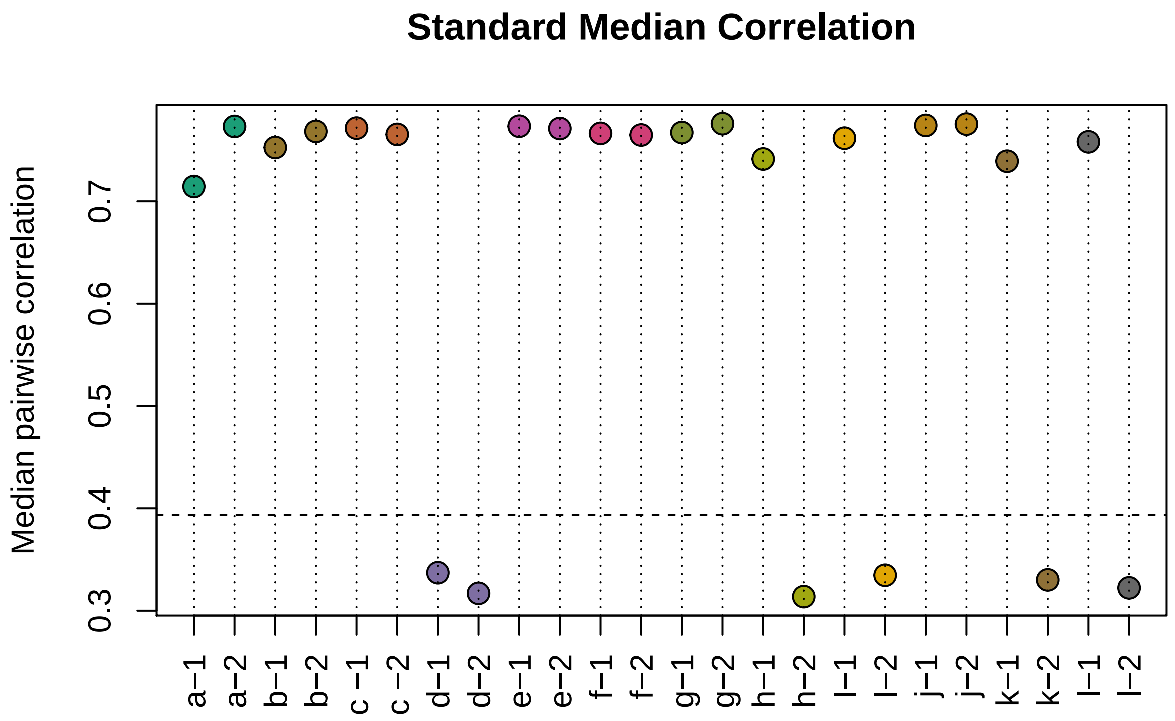
```
## Graphing number of non-zero genes with respect to CPM by library.
## Graphing library sizes.
## Adding log10
## Graphing a boxplot on log scale.
```

```
## [1] "I think this probably should be put on a log scale to be visible."
## [1] "Run this function with 'scale=\"log\"' to try it out."
```

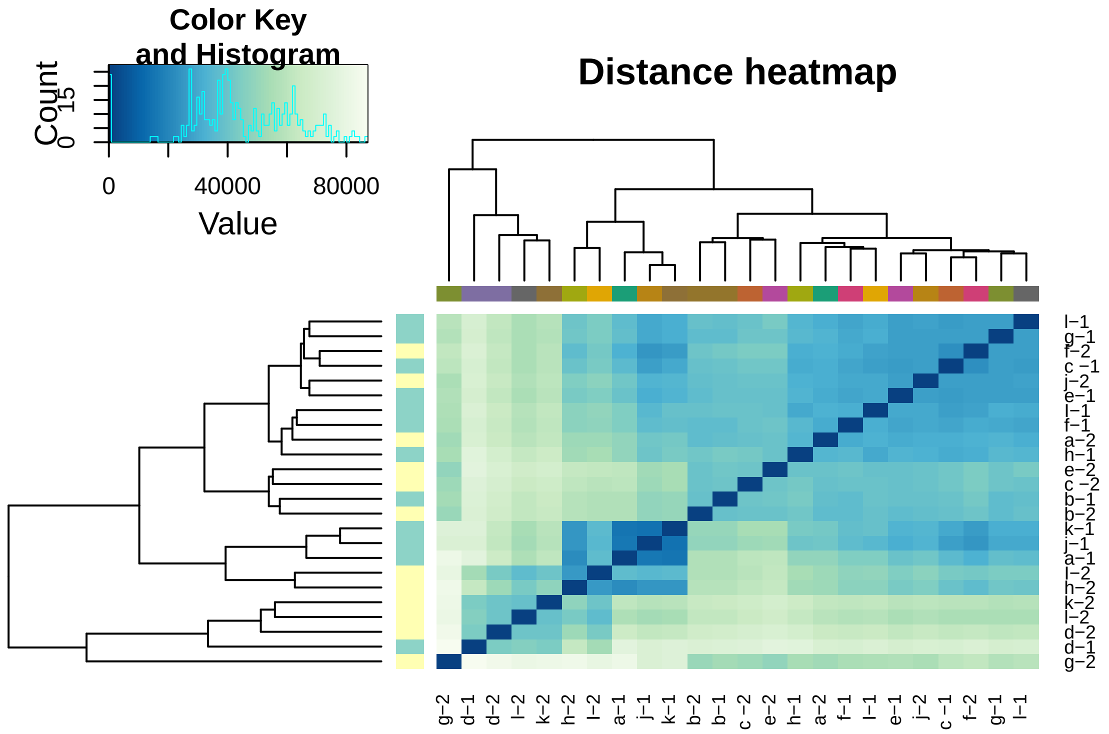
```
## Graphing a correlation heatmap.
## Graphing a standard median correlation.
```



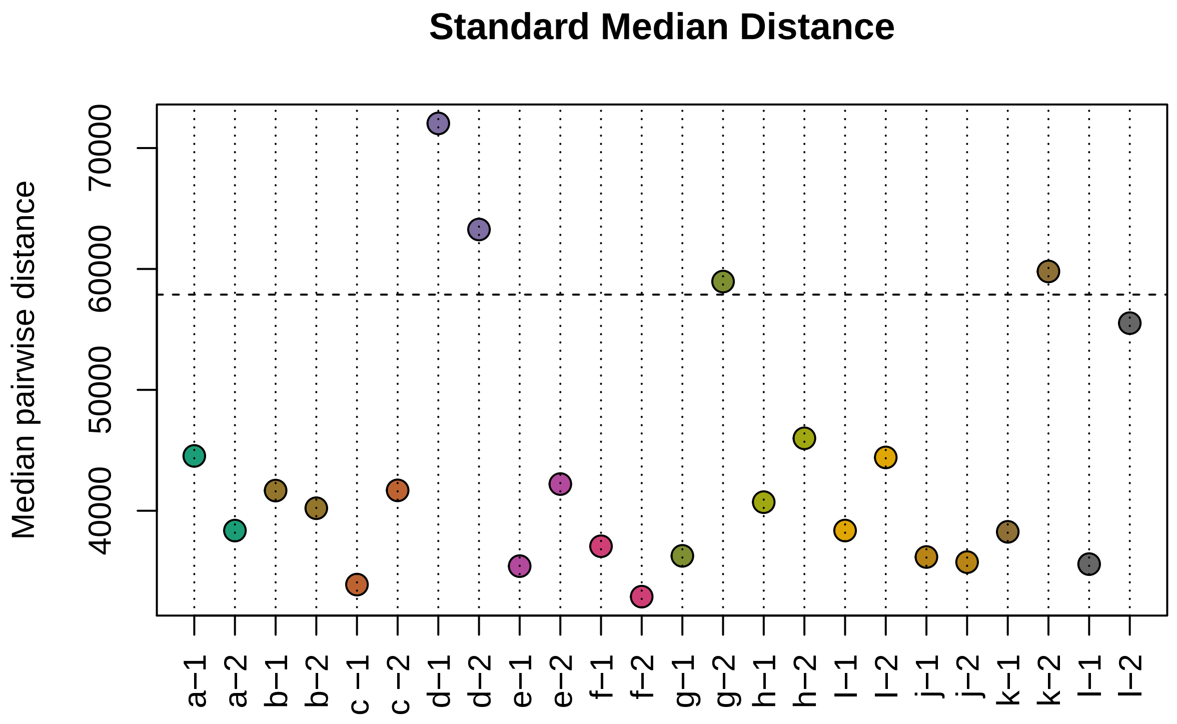
```
## Graphing a distance heatmap.
```



```
## Graphing a standard median distance.
```

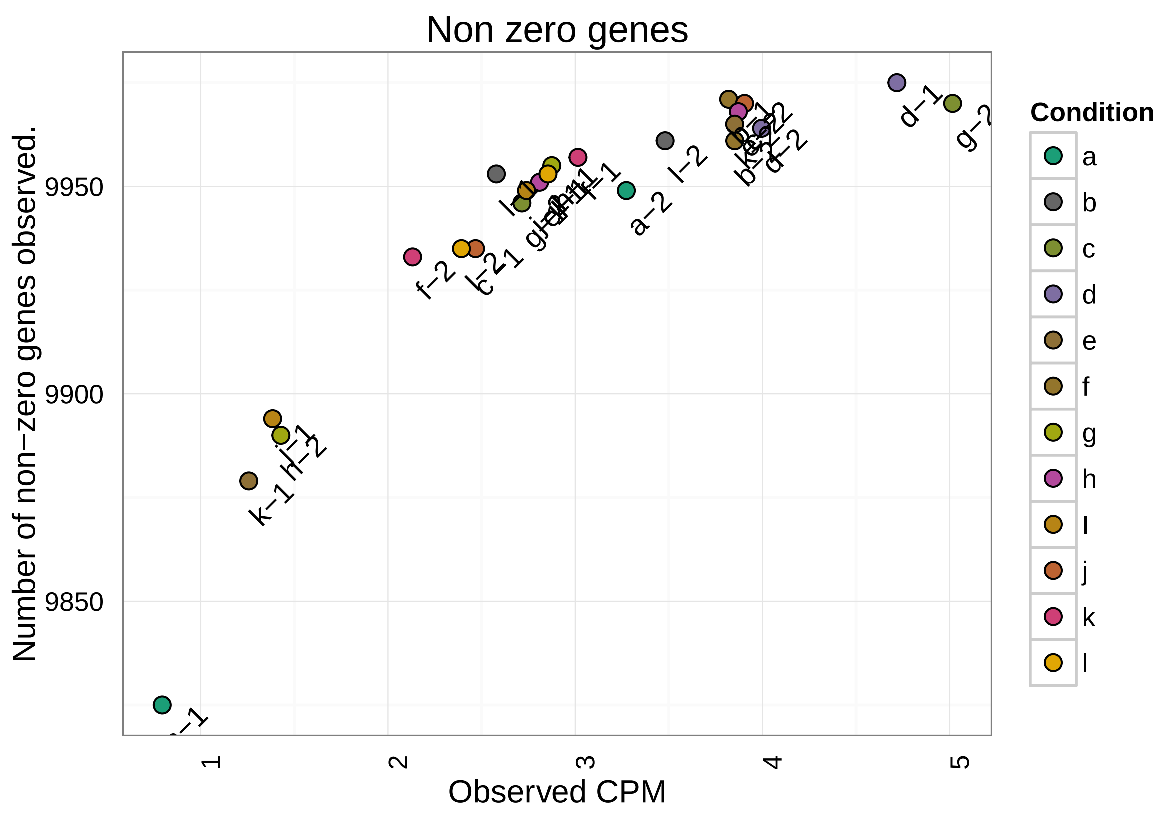


```
## Graphing a PCA plot.
## Plotting a density plot.

## [1] "Perhaps this data should be plotted on the log scale, add log=TRUE to try it out."

## Using  as id variables
```

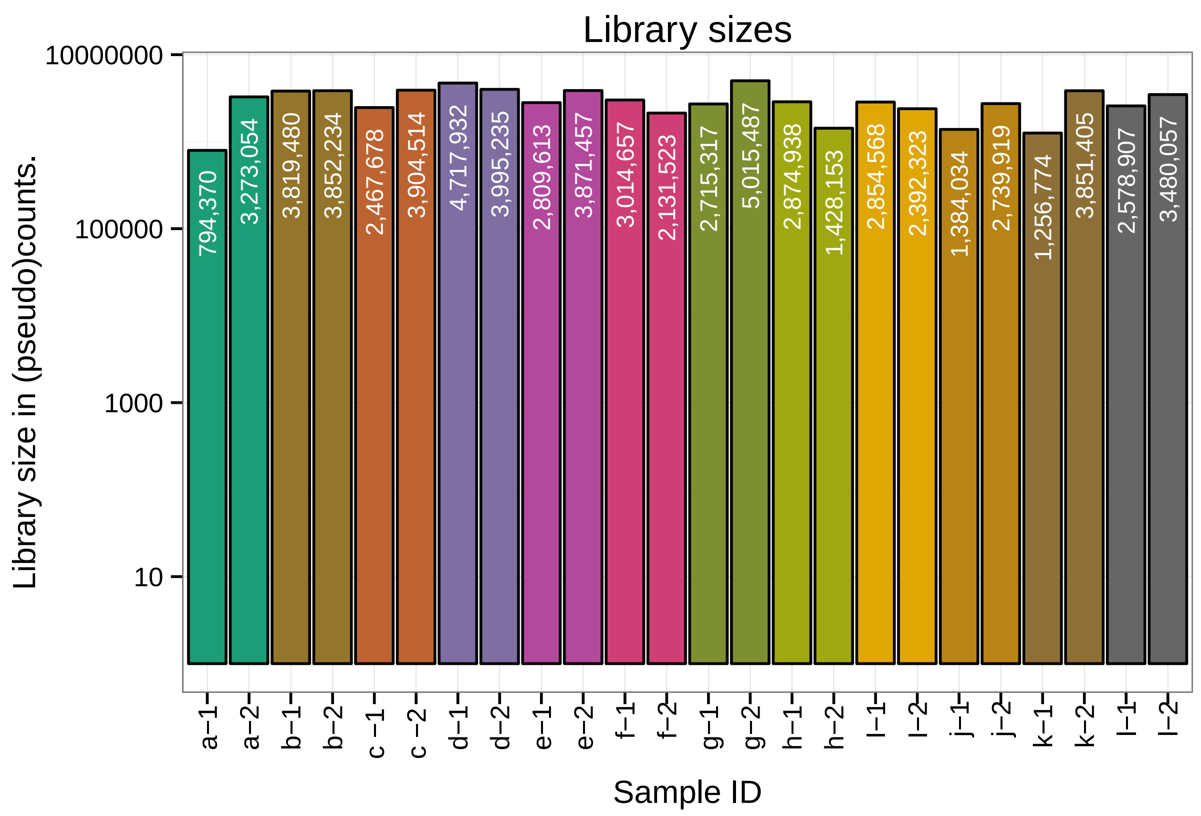


```
fun
```

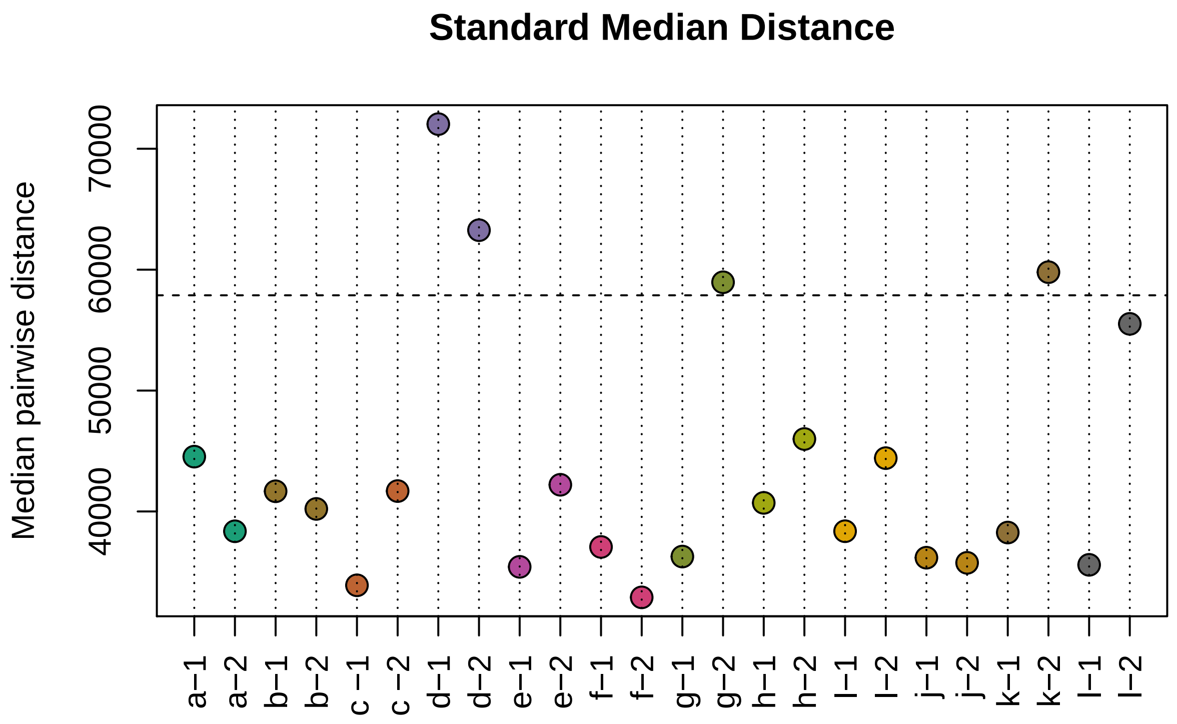
```
## $nonzero
```



```
##
## $libsize
```

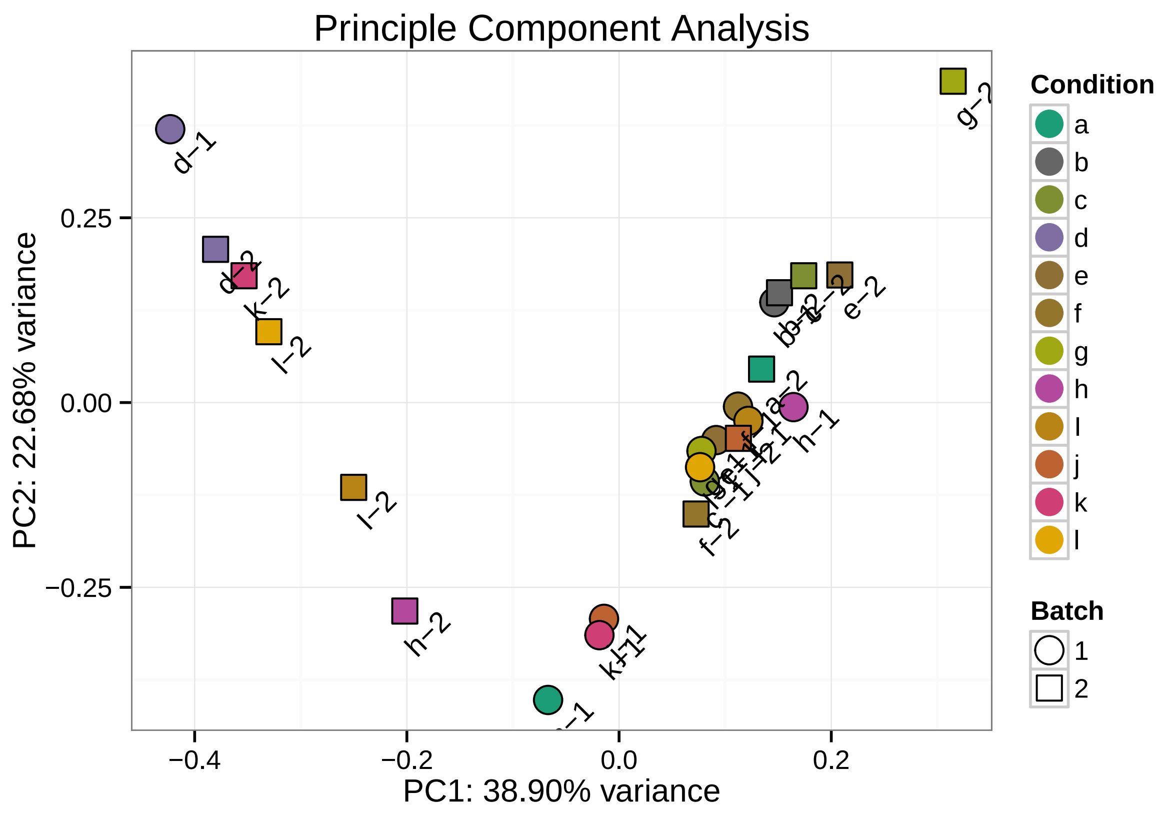

Library sizes
Library size in (pseudo)counts.
10000000
100000
1000
10
794,370
3,273,054
3,819,480
3,852,234
2,467,678
3,904,514
4,717,932
3,995,235
2,809,613
3,871,457
3,014,657
2,131,523
2,715,317
5,015,487
2,874,938
1,428,153
2,854,568
2,392,323
1,384,034
2,739,919
1,256,774
3,851,405
2,578,907
3,480,057
a−1
a−2
b−1
b−2
c −1
c −2
d−1
d−2
e−1
e−2
f−1
f−2
g−1
g−2
h−1
h−2
I−1
I−2
j−1
j−2
k−1
k−2
l−1
l−2
Sample ID

```
##
## $boxplot
```

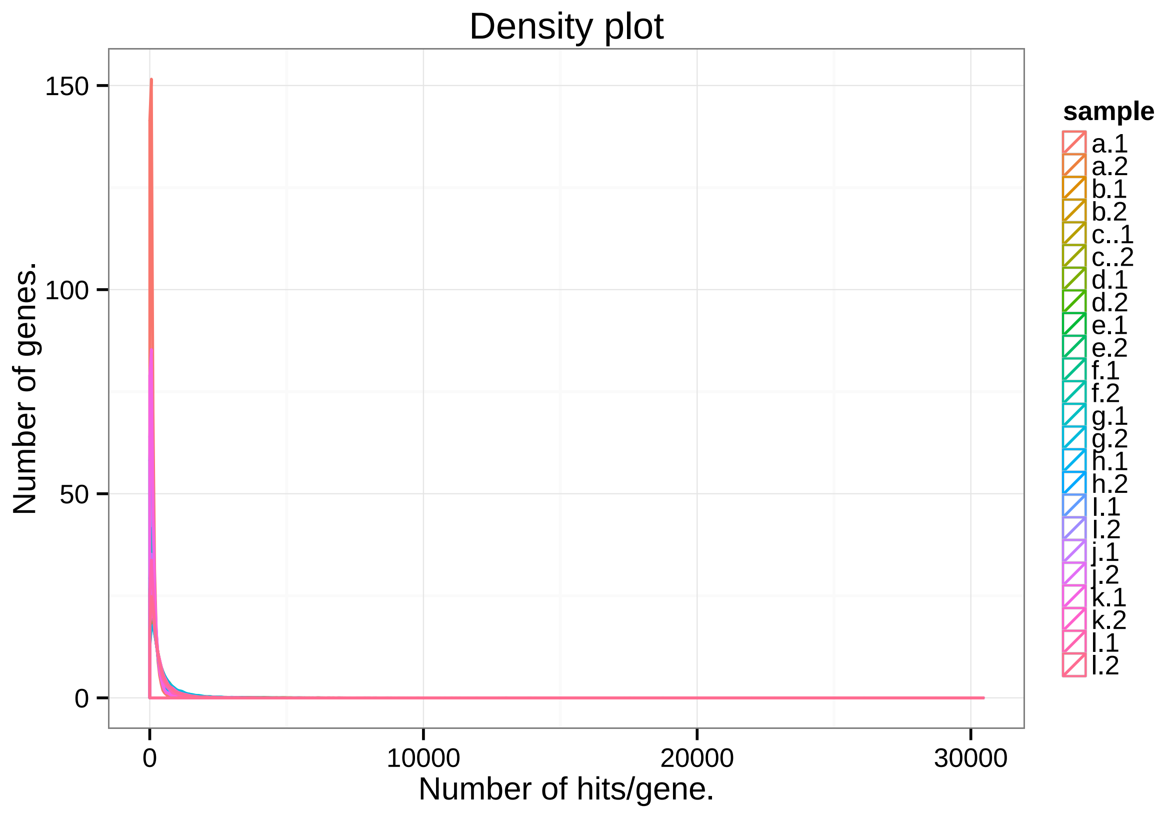

Standard Median Distance
Median pairwise distance
40000
50000
60000
70000
a−1
a−2
b−1
b−2
c −1
c −2
d−1
d−2
e−1
e−2
f−1
f−2
g−1
g−2
h−1
h−2
I−1
I−2
j−1
j−2
k−1
k−2
l−1
l−2

```
##
```

```
## $corheat
##
## $smc
##
## $disheat
##
## $smd
##
## $pcaplot
```



```
##
## $pcatable
##           SampleID condition batch batch_int         PC1          PC2
## HPGL0406       a-1         a     1         1 -0.06691092 -0.402355331
## HPGL0407       a-2         a     2         2  0.13431332  0.045259223
## HPGL0408       b-1         b     1         1  0.14634248  0.135682093
## HPGL0409       b-2         b     2         2  0.15119716  0.148552727
## HPGL0149      c -1        c      1         1  0.08078548 -0.106755934
## HPGL0150      c -2        c      2         2  0.17403628  0.171498664
## HPGL0147       d-1         d     1         1 -0.42304652  0.369802408
## HPGL0148       d-2         d     2         2 -0.38027775  0.207290683
## HPGL0410       e-1         e     1         1  0.09148588 -0.050776835
## HPGL0411       e-2         e     2         2  0.20806087  0.172566966
## HPGL0412       f-1         f     1         1  0.11211147 -0.005541911
## HPGL0413       f-2         f     2         2  0.07258768 -0.150795269
## HPGL0414       g-1         g     1         1  0.07767734 -0.065569614
## HPGL0416       g-2         g     2         2  0.31491903  0.434733324
## HPGL0415      h-1         h     1         1  0.16438931 -0.006171446
## HPGL0417      h-2         h     2         2 -0.20194956 -0.282064901
## HPGL0418      I-1         I     1         1  0.12188367 -0.024827186
## HPGL0419      I-2         I     2         2 -0.25012222 -0.114756653
## HPGL0420      j-1         j     1         1 -0.01422104 -0.292626128
## HPGL0421      j-2         j     2         2  0.11244544 -0.048430048
## HPGL0422      k-1         k     1         1 -0.01852333 -0.314655028
## HPGL0423      k-2         k     2         2 -0.35343798  0.171561901
## HPGL0424      l-1         l     1         1  0.07632821 -0.087452287
## HPGL0425      l-2         l     2         2 -0.33007430  0.095830582
##           colors labels
## HPGL0406 #1B9E77    a-1
## HPGL0407 #1B9E77    a-2
## HPGL0408 #93752C    b-1
## HPGL0409 #93752C    b-2
## HPGL0149 #BD6332   c -1
## HPGL0150 #BD6332   c -2
## HPGL0147 #7E6EA2    d-1
## HPGL0148 #7E6EA2    d-2
## HPGL0410 #B3499C    e-1
## HPGL0411 #B3499C    e-2
## HPGL0412 #CF3F76    f-1
## HPGL0413 #CF3F76    f-2
## HPGL0414 #7D8F31    g-1
## HPGL0416 #7D8F31    g-2
## HPGL0415 #A0A811    h-1
## HPGL0417 #A0A811    h-2
## HPGL0418 #E0A604    I-1
## HPGL0419 #E0A604    I-2
## HPGL0420 #B78415    j-1
## HPGL0421 #B78415    j-2
## HPGL0422 #8E7037    k-1
## HPGL0423 #8E7037    k-2
## HPGL0424 #666666    l-1
## HPGL0425 #666666    l-2
##
## $pcares
##    propVar cumPropVar cond.R2 batch.R2
## 1    38.90      38.90   65.58     2.02
## 2    22.68      61.58   48.03    12.08
## 3     4.18      65.76   55.85     2.90
## 4     3.47      69.23   60.55     1.22
## 5     3.06      72.29   13.77    16.57
## 6     2.91      75.20   51.94     0.01
## 7     2.83      78.03   58.16     2.50
## 8     2.60      80.63   40.48     3.92
## 9     2.57      83.20   29.00     2.97
## 10    2.50      85.70   31.46     6.33
## 11    1.99      87.69   55.96     4.23
## 12    1.64      89.33   50.20     9.96
## 13    1.56      90.89   52.36     0.16
## 14    1.43      92.32   33.43     0.01
## 15    1.27      93.59   45.31     0.17
## 16    1.25      94.84   50.07    16.02
## 17     1.18       96.02    46.71      3.30
## 18     1.12       97.14    60.63      0.07
## 19     1.00       98.14    40.19      0.77
## 20     0.84       98.98    54.10      6.48
## 21     0.45       99.43    53.26      8.16
## 22     0.34       99.77    54.93      0.10
## 23     0.22       99.99    48.03      0.06
##
## $pcavar
##  [1] 38.90 22.68  4.18  3.47  3.06  2.91  2.83  2.60  2.57  2.50  1.99
## [12]  1.64  1.56  1.43  1.27  1.25  1.18  1.12  1.00  0.84  0.45  0.34
## [23]  0.22
##
## $density
```



```
##
## $qq
## NULL
##
## $ma
## NULL
```

```
## The following are some examples of other ways to make use of these plots:

##fun_boxplot = hpgl_boxplot(df=fun)
##print(fun_boxplot)
##log_boxplot = hpgl_boxplot(df=fun, scale="log")
```

```
##print(log_boxplot)
##hpgl_corheat(df=fun, colors=hpgl_colors)
##hpgl_disheat(df=fun, colors=hpgl_colors)
##hpgl_smc(df=fun, colors=hpgl_colors)
##hpgl_libsize(df=fun)
##hpgl_qq_all(df=fun)
```

Normalizing data

RNAseq data must be normalized. Here is one easy method:

```
## normalize_expt will do this on the expt class, replace the expressionset therein, and
## make a backup of the data inside the expt class.
norm_expt = normalize_expt(all_expt)
```

```
## [1] "This function will replace the expt$expressionset slot with the raw(raw(raw))'d data."
## [1] "It saves the current data into a slot named: expt$backup_expressionset"
## [1] "It will also save copies of each step along the way in expt$normalized with the corresponding li
## [1] "Keep the libsizes in mind when invoking limma.  The appropriate libsize is the non-log(cpm(norma
## [1] "This is most likely kept in the slot called: 'new_expt$normalized$normalized_counts$libsize' whi
## [1] "Filter low is false, this should likely be set to something, good choices include cbcb, kofa, po
## [1] "Leaving the data in its current base format, keep in mind that some metrics are easier to see wh
## [1] "Leaving the data unconverted.  It is often advisable to cpm/rpkm the data to normalize for sampl
## [1] "Leaving the data unnormalized.  This is necessary for DESeq, but EdgeR/limma might benefit from
## [1] "Not correcting the count-data for batch effects.  If batch is included in EdgerR/limma's model,
```

```
head(exprs(norm_expt$expressionset))
```

```
##          HPGL0406 HPGL0407 HPGL0408 HPGL0409 HPGL0149 HPGL0150 HPGL0147
## gene_1_F       10      155      326      552      248       92      206
## gene_2_T      127     1375      923      405       96      614      325
## gene_3_F        5       45       18       63       38       38       49
## gene_4_F        6       62       70       33       29       15       38
## gene_5_F       13      149      134      133      153       90      273
## gene_6_F      441      960     1884      650      604      817     1096
##          HPGL0148 HPGL0410 HPGL0411 HPGL0412 HPGL0413 HPGL0414 HPGL0416
## gene_1_F      218       81      323      243       72      185      576
## gene_2_T      304      422      722      698      367      956     1380
## gene_3_F       52       47       35       76       28       34       86
## gene_4_F       37       30       26       24       10       22       34
## gene_5_F       87      187      228      132      185      107      138
## gene_6_F     1450      781     1252     1101     1017      706      833
##          HPGL0415 HPGL0417 HPGL0418 HPGL0419 HPGL0420 HPGL0421 HPGL0422
## gene_1_F      374       92      144      163      121       78       57
## gene_2_T      260      121      769      175      285      199      182
## gene_3_F       45       19       38       27       22        7       47
## gene_4_F       29       17       27       16       21       55        6
## gene_5_F      168       49      134      117       92       78       99
## gene_6_F      313      254      590      485      522      847      300
##          HPGL0423 HPGL0424 HPGL0425
## gene_1_F      273      243      192
## gene_2_T      219      310      287
## gene_3_F       15       38       47
## gene_4_F       92       16       70
## gene_5_F      139       63      136
## gene_6_F     1064      241     1083
```

```
## size factor, tmm, rle, upperQuartile all require a design matrix.
norm_boxplot = hpgl_boxplot(expt=norm_expt)
```

```
## Error in hpgl_boxplot(expt = norm_expt): argument "data" is missing, with no default
```

```
print(norm_boxplot)
```

```
## Error in print(norm_boxplot): error in evaluating the argument 'x' in selecting a method for functio
```

```
norm_disheat = hpgl_disheat(expt=norm_expt)
```

```
## Error in hpgl_heatmap(data, colors = colors, design = design, method = method, : argument "data" is
```

```
print(norm_disheat)
```

```
## Error in print(norm_disheat): error in evaluating the argument 'x' in selecting a method for functio
```

Voom/limma etc

There are a couple ways to call limma using the expt class. In some cases, it might be useful to pull out a subset of the data and only compare the samples of specific conditions/batches/etc.

```
## el_subset means to pull out only those samples which represent 'Early Log' growth.
el_subset = expt_subset(norm_expt, "stage=='EL'")
## Conversely, one may pull samples which are early log and also wild type
elwt_subset = expt_subset(norm_expt, "stage=='EL'&type=='WT'")
## These subsets may be characterized with the plots as above
## Here is a qq plot as an example.
elwt_qqs = hpgl_qq_all(expt=elwt_subset)
```

```
## Error in hpgl_qq_all(expt = elwt_subset): unused argument (expt = elwt_subset)
```

```
## Simple comparison will take the first condition as control and the second
## as experimental, if we look at el_subset, we will see that means conditions
## 'a' and 'b'.  Thus performing simple_comparison will look for differentially
## expressed genes between them.
head(el_subset$design)
```

```
##             sample stage type condition batch   color  counts intercounts
## HPGL0406 HPGL0406    EL   WT         a     1 #1B9E77 unknown     unknown
## HPGL0407 HPGL0407    EL   WT         a     2 #1B9E77 unknown     unknown
## HPGL0408 HPGL0408    EL  mga         b     1 #93752C unknown     unknown
## HPGL0409 HPGL0409    EL  mga         b     2 #93752C unknown     unknown
```

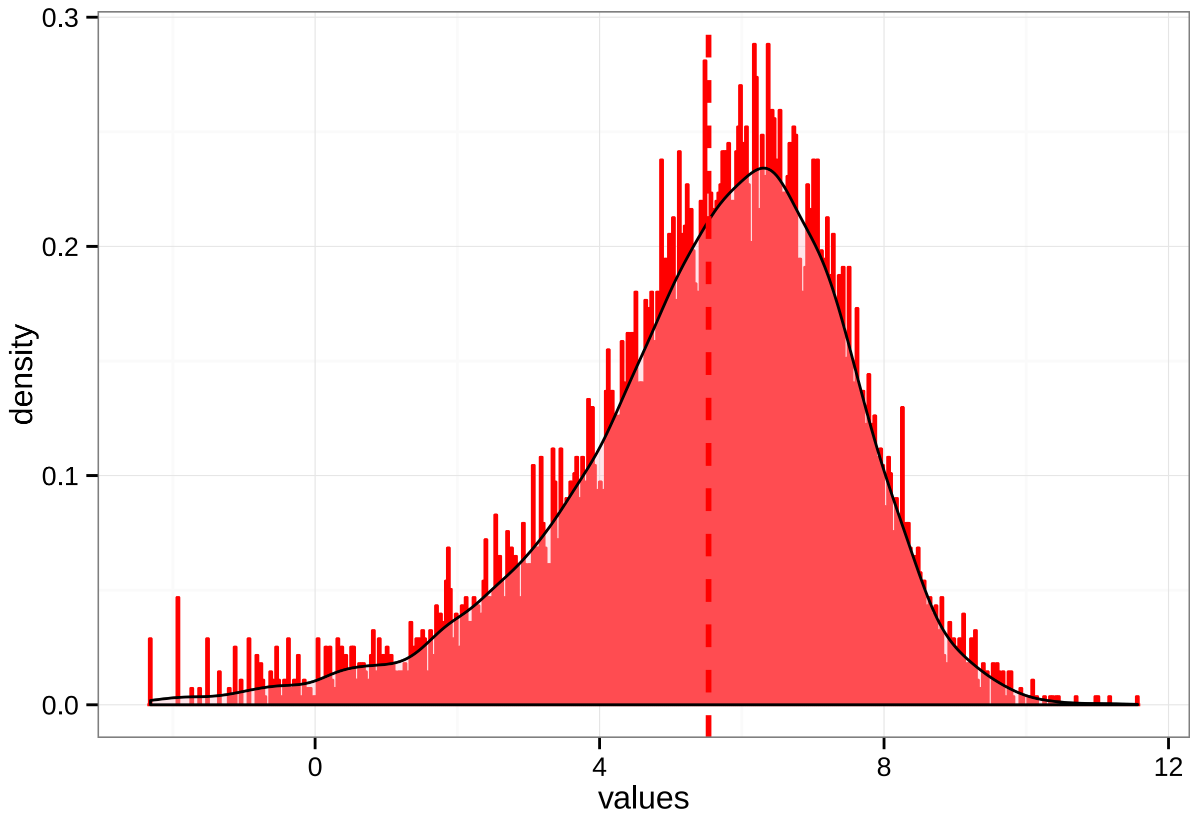
```r
ab_comparison = simple_comparison(el_subset)
```

```
## The voom input was not cpm, converting now.
## The voom input was not log2, transforming now.
## No binwidth nor bins provided, setting it to 0.0299012603878232 in order to have 500 bins.
```

```r
## A summary of the data will show the data provided:
## The following plots and pieces of data show the output provided by simple_comparison()
## This function isn't really intended to be used, but provides a reference point for performing other a
summary(ab_comparison)
```

```
##                         Length Class         Mode
## amean_histogram             9  gg            list
## coef_amean_cor              9  htest         list
## coefficient_scatter         9  gg            list
## coefficient_x               9  gg            list
## coefficient_y               9  gg            list
## coefficient_both            4  -none-        list
## coefficient_lm             22  lmrob         list
## coefficient_lmsummary      15  summary.lmrob list
## coefficient_weights     10000  -none-        numeric
## comparisons             10000  MArrayLM      list
## contrasts               10000  MArrayLM      list
## contrast_histogram          9  gg            list
## downsignificant             6  data.frame    list
## fit                     30000  MArrayLM      list
## ma_plot                     9  gg            list
## psignificant                6  data.frame    list
## pvalue_histogram            9  gg            list
## table                       6  data.frame    list
## upsignificant               6  data.frame    list
## volcano_plot                9  gg            list
## voom_data               40000  EList         list
## voom_plot                   9  gg            list
```

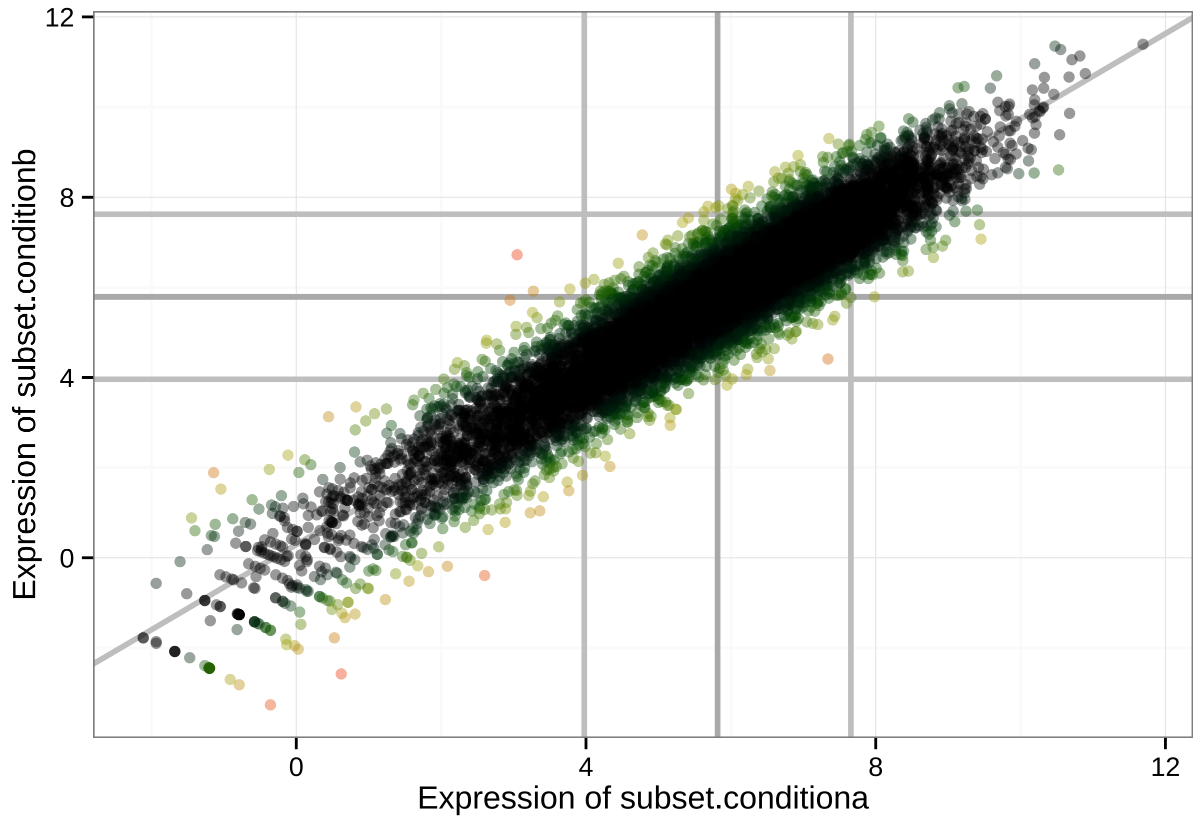
```r
print(ab_comparison$amean_histogram)  ## A histogram of the per-gene mean values
```



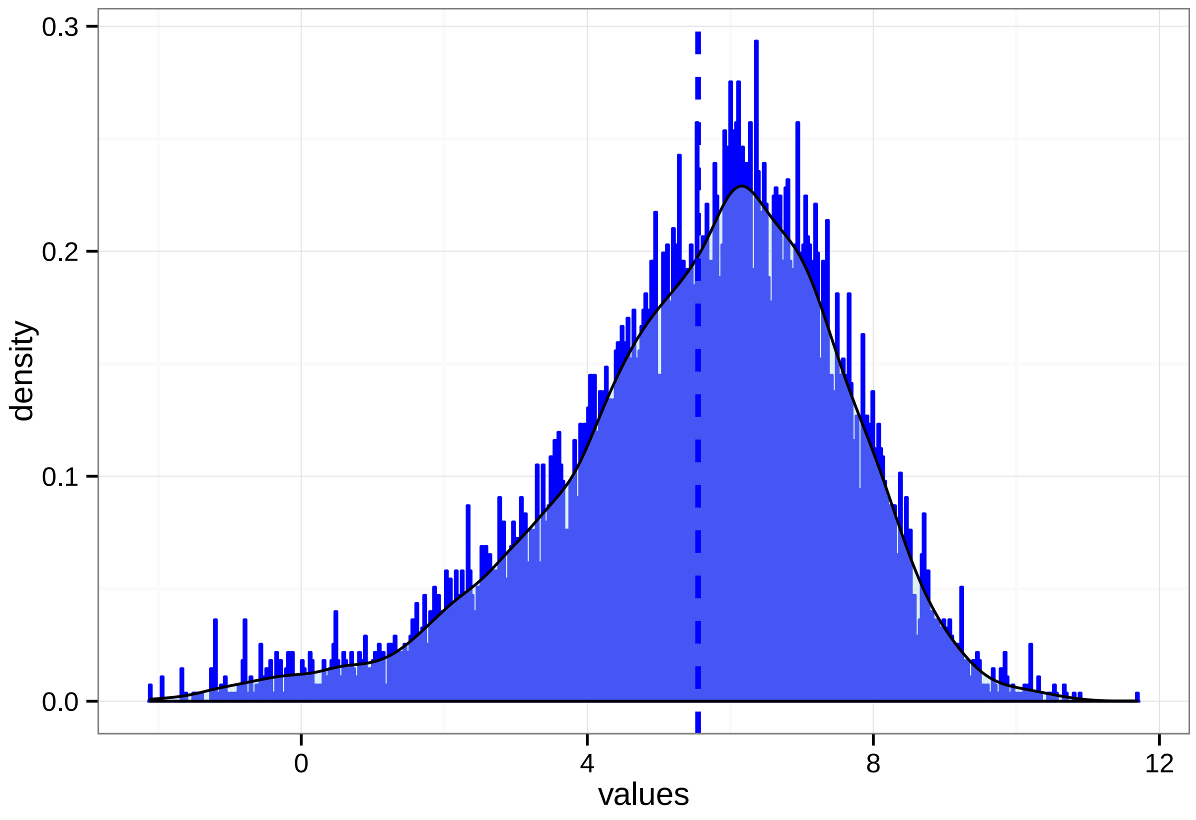
```
print(ab_comparison$coef_amean_cor)   ## The correlation of the means (should not be significant)
```

```
##
##  Pearson's product-moment correlation
##
## data:  cond_contrasts$coefficients and cond_contrasts$Amean
## t = -4.3772, df = 9998, p-value = 0.00001214
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.06328073 -0.02415554
## sample estimates:
##        cor
## -0.0437349
```

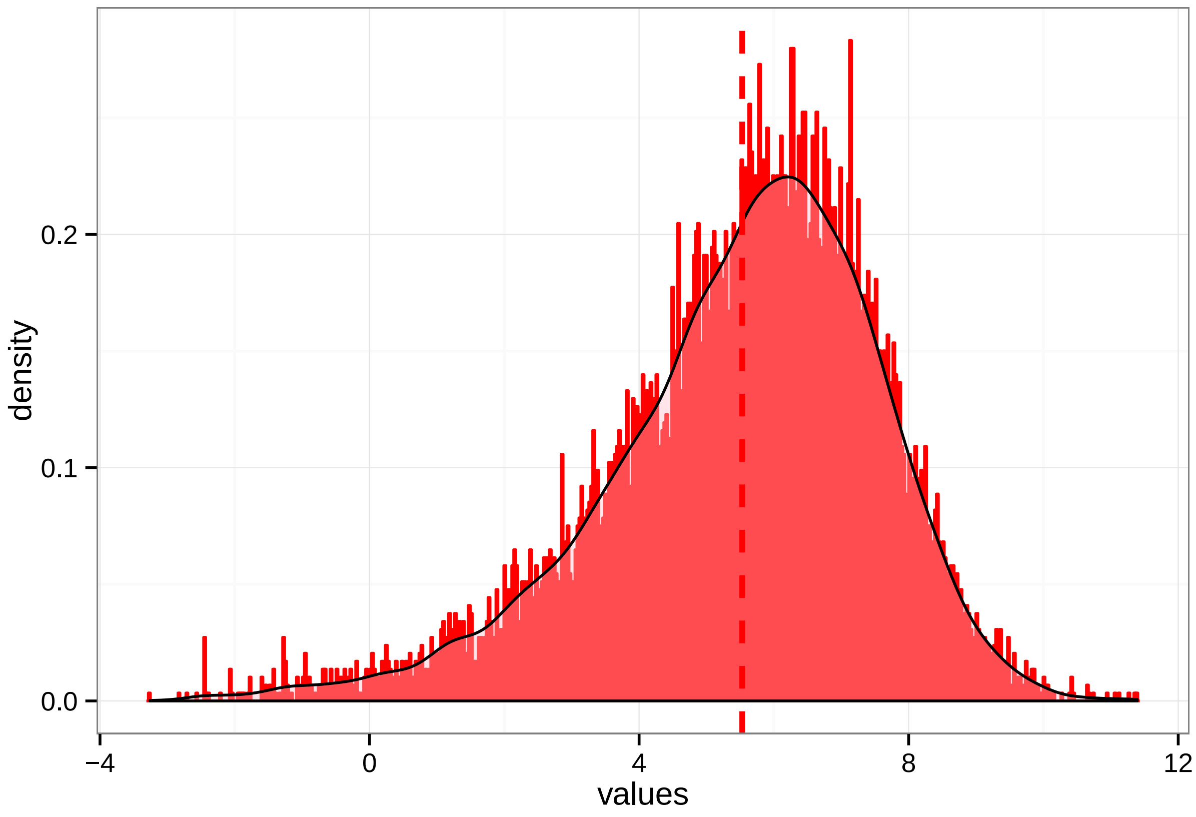
```
print(ab_comparison$coefficient_scatter) ## A scatter plot of condition b with respect to a
```



```r
print(ab_comparison$coefficient_x) ## A histogram of the gene abundances of a
```

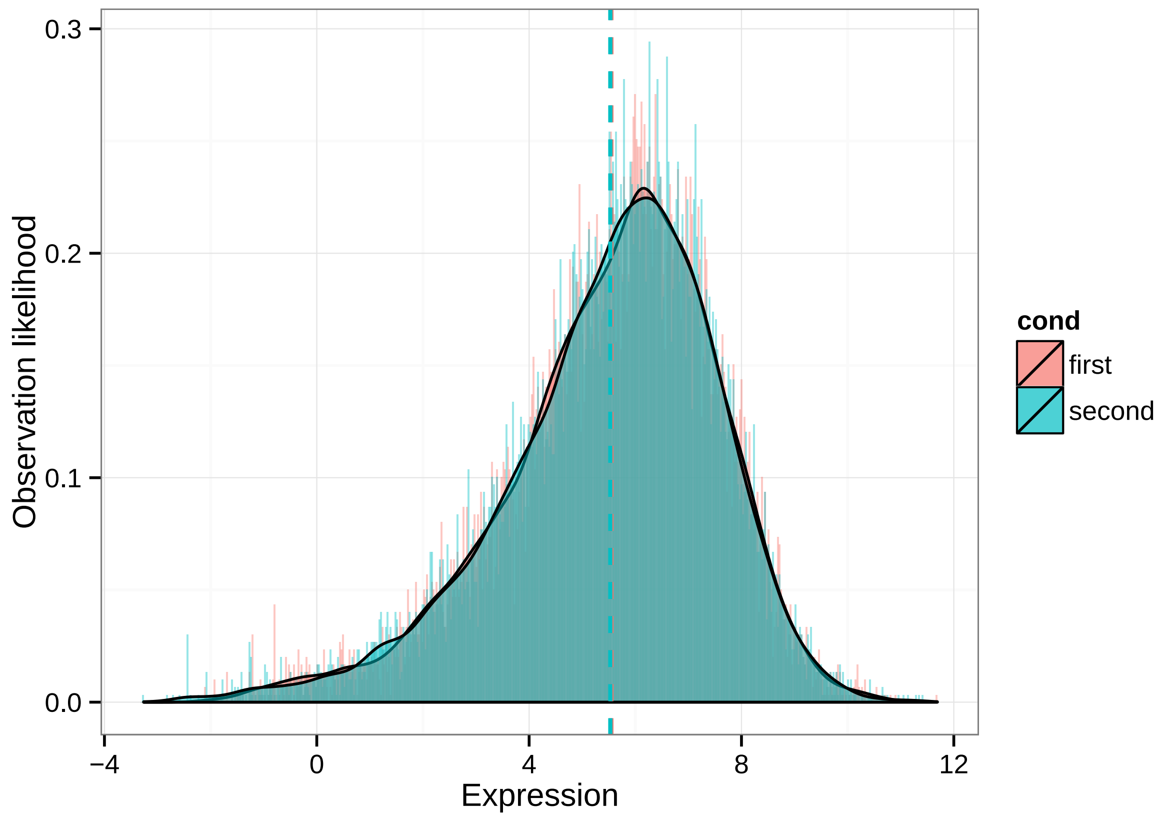


```
print(ab_comparison$coefficient_y) ## A histogram of the gene abundances of b
```



```
print(ab_comparison$coefficient_both) ## A histogram of the gene abundances of a and b
```

```
## $plot
```



```
##
## $data_summary
##      first            second
##  Min.   :-2.113   Min.   :-3.261
##  1st Qu.: 4.421   1st Qu.: 4.411
##  Median : 5.817   Median : 5.791
##  Mean   : 5.548   Mean   : 5.530
##  3rd Qu.: 6.942   3rd Qu.: 6.925
##  Max.   :11.690   Max.   :11.393
##
## $uncor_t
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  play_all$expression and play_all$cond
##
##        first
## second 0.51
##
## P value adjustment method: none
##
## $bon_t
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  play_all$expression and play_all$cond
##
##        first
## second 0.51
##
## P value adjustment method: bonferroni
```

```r
## Note to self, I keep meaning to change the colors of that to match the others
print(ab_comparison$coefficient_lm) ## The description of the line which describes the relationship
```

```
##
## Call:
## robustbase::lmrob(formula = second ~ first, data = df, method = "SMDM")
##
## Coefficients:
## (Intercept)        first
##      0.2936       0.9446
```

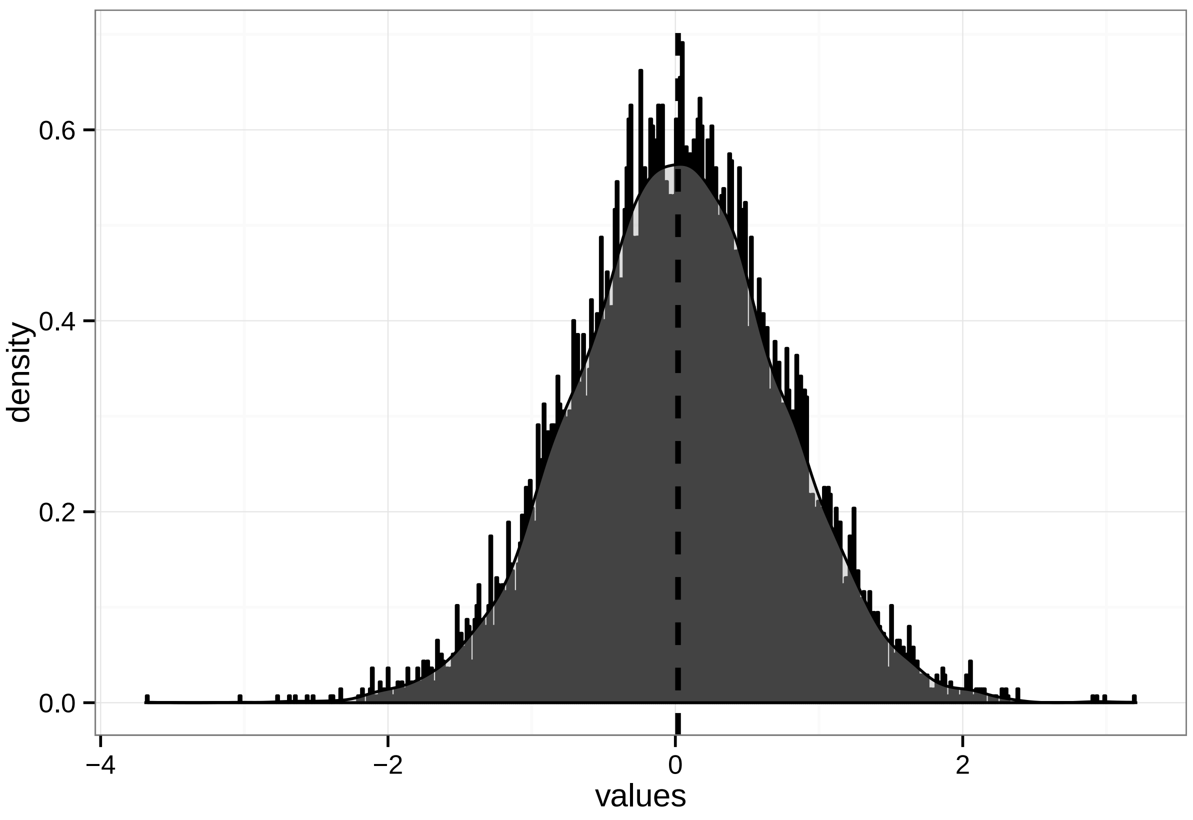
```r
## of all of the genes in a to those in b
print(ab_comparison$coefficient_lmsummary) ## A summary of the robust linear model in coefficient_lm
```

```
##
## Call:
## robustbase::lmrob(formula = second ~ first, data = df, method = "SMDM")
##  \--> method = "SMDM"
## Residuals:
##       Min        1Q    Median        3Q       Max
## -3.454612 -0.462071  0.005723  0.466988  3.549615
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 0.293646   0.021566   13.62 <0.0000000000000002 ***
## first       0.944617   0.003646  259.09 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Robust residual standard error: 0.699
## Multiple R-squared:  0.8757, Adjusted R-squared:  0.8757
## Convergence in 7 IRWLS iterations
##
## Robustness weights:
##  6951 weights are ~= 1. The remaining 3049 ones are summarized as
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07042 0.81910 0.92800 0.86890 0.97920 0.99900
## Algorithmic parameters:
##       tuning.chi1       tuning.chi2       tuning.chi3       tuning.chi4
## -0.50000000000000  1.50000000000000                NA  0.50000000000000
##                bb       tuning.psi1       tuning.psi2       tuning.psi3
##  0.50000000000000 -0.50000000000000  1.50000000000000  0.95000000000000
##       tuning.psi4        refine.tol           rel.tol         solve.tol
##                NA  0.00000010000000  0.00000010000000  0.00000010000000
##       eps.outlier             eps.x warn.limit.reject warn.limit.meanrw
##  0.00001000000000  0.00000000002126  0.50000000000000  0.50000000000000
##      nResample         max.it       best.r.s       k.fast.s          k.max
##            500             50              2              1            200
##    maxit.scale      trace.lev            mts     compute.rd      numpoints
##            200               0           1000               0             10
## fast.s.large.n
##           2000
##                  psi           subsampling                   cov
##                "lqq"         "nonsingular"             ".vcov.w"
## compute.outlier.stats
##                "SMDM"
## seed : int(0)
```

```
## This has some neat things like the R-squared value and the parameters used to arrive at the linear mo
## ab_comparison$coefficient_weights ## a list of weights by gene, bigger weights mean closer to the lin
## ab_comparison$comparisons ## the raw output from limma
print(ab_comparison$contrasts)  ## The output from limma's makeContrasts()
```

```
## An object of class "MArrayLM"
## $coefficients
##    gene_1_F    gene_2_T    gene_3_F    gene_4_F    gene_5_F
## -2.06830281  0.73688637  0.05179155  0.10814455 -0.25655242
## 9995 more rows ...
##
## $stdev.unscaled
##  gene_1_F  gene_2_T  gene_3_F  gene_4_F  gene_5_F
## 0.4937486 0.4386966 0.5666225 0.5255633 0.4906638
## 9995 more rows ...
##
## $sigma
## [1] 1.1499226 2.9749817 0.7076561 2.2577474 1.5163376
## 9995 more elements ...
##
## $df.residual
## [1] 1 1 1 1 1
## 9995 more elements ...
##
## $cov.coefficients
##                     Contrasts
## Contrasts            changed_v_control
##   changed_v_control                 1
##
## $rank
## [1] 3
##
## $Amean
## gene_1_F gene_2_T gene_3_F gene_4_F gene_5_F
## 5.719066 7.669256 3.226942 3.653567 4.963363
## 9995 more elements ...
##
## $method
## [1] "ls"
##
## $design
##   changed control subset_batch2
## 1       1       0             0
## 2       1       0             1
## 3       0       1             0
## 4       0       1             1
## attr(,"assign")
## [1] 1 1 2
## attr(,"contrasts")
## attr(,"contrasts")$`subset$condition`
## [1] "contr.treatment"
##
## attr(,"contrasts")$`subset$batch`
## [1] "contr.treatment"
##
##
## $contrasts
##               Contrasts
## Levels          changed_v_control
##   changed                       1
##   control                      -1
##   subset_batch2                 0
```

```
print(ab_comparison$contrast_histogram)  ## A histogram of the values of b-a for each gene
```



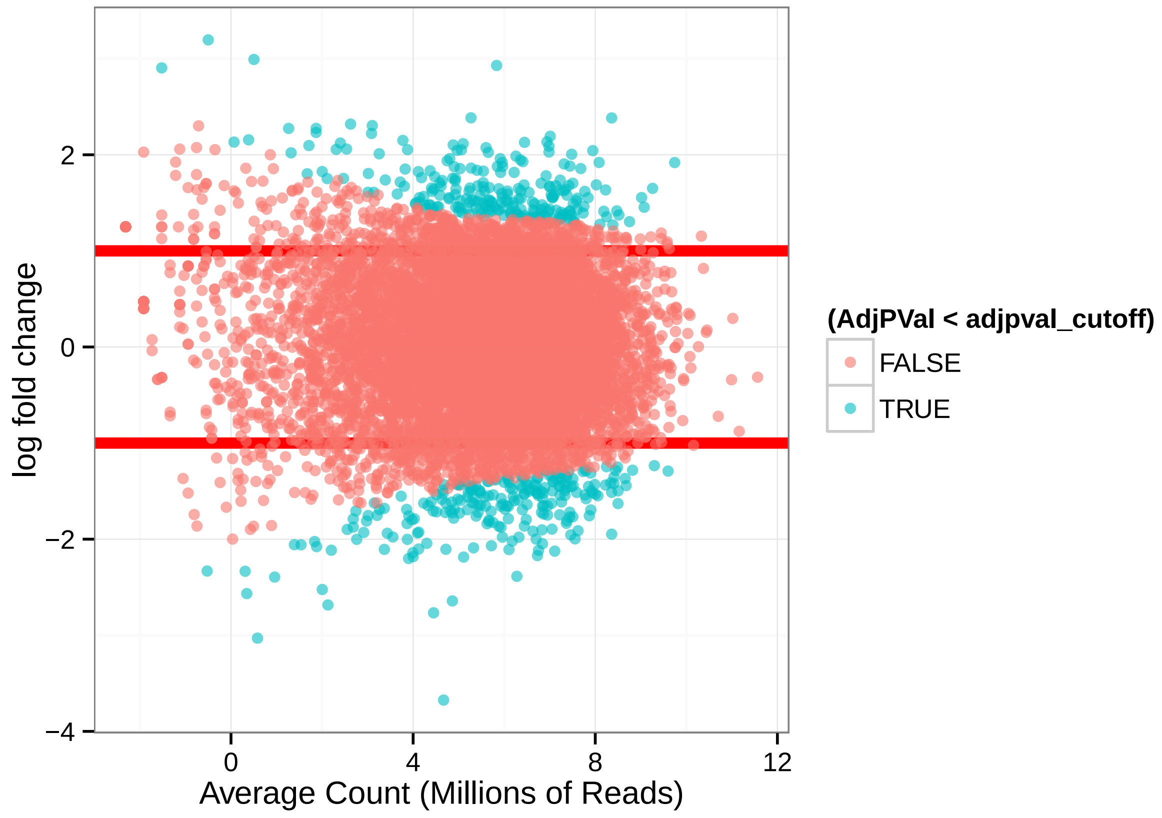
```
head(ab_comparison$downsignificant)  ## The list of genes which are significantly down in b vs a
```

```
##                 logFC   AveExpr         t       P.Value adj.P.Val
## gene_9453_F -3.674407 4.6687613 -4.751151 0.00000205079 0.0205079
## gene_6962_F -3.030009 0.5821939 -2.894170 0.00380984540 0.8466323
## gene_5901_F -2.766474 4.4491470 -3.583991 0.00033999470 0.8440245
## gene_2815_F -2.684763 2.1287330 -2.908773 0.00363649503 0.8440245
## gene_2096_F -2.642624 4.8614813 -3.465371 0.00053172354 0.8440245
## gene_2865_F -2.565979 0.3463398 -2.441604 0.01463940433 0.9979641
##                     B
## gene_9453_F -1.244491
## gene_6962_F -3.870834
## gene_5901_F -2.764891
## gene_2815_F -3.707017
## gene_2096_F -2.868598
## gene_2865_F -4.114582
```

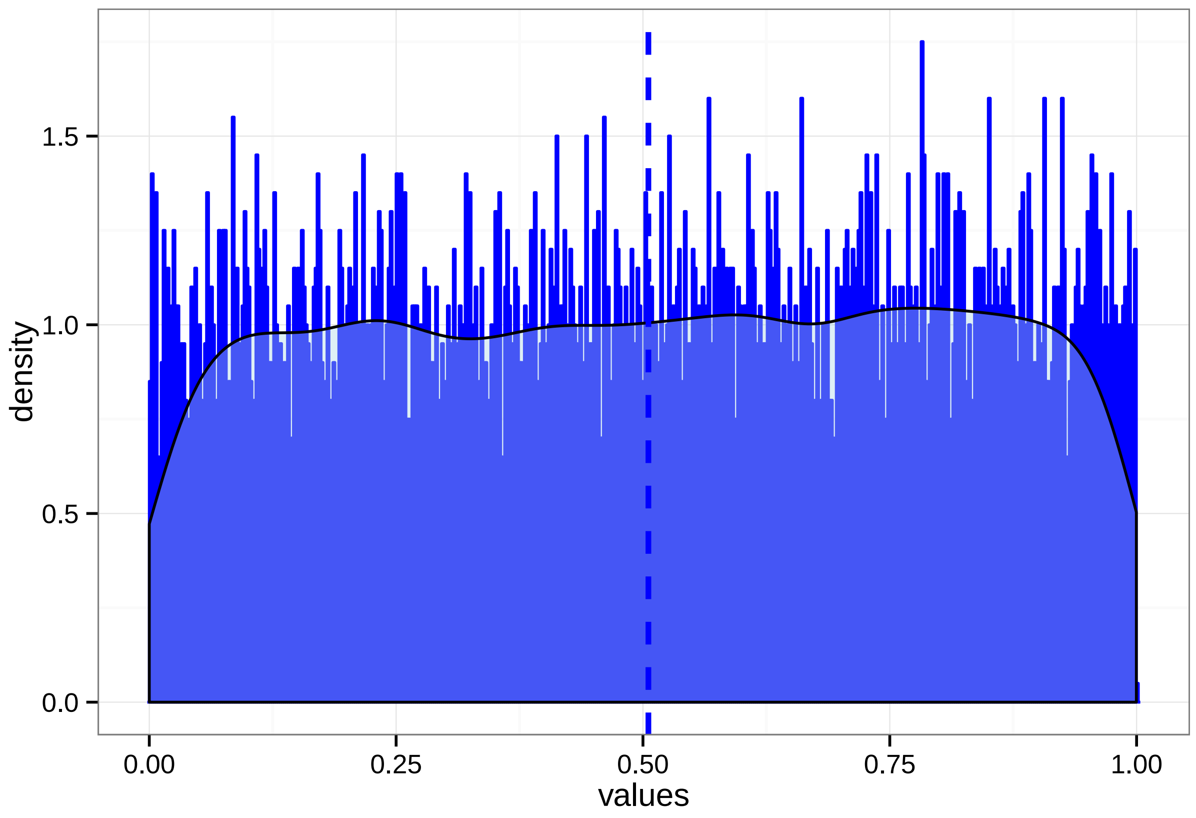
```
dim(ab_comparison$downsignificant)
```

```
## [1] 1903    6
```

```
## ab_comparison$fit ## the result from lmFit()
print(ab_comparison$ma_plot)  ## An ma plot of b vs a
```



```
print(ab_comparison$pvalue_histogram) ## A histogram of the p-values, one would hope to see a spike in t
```



```
head(ab_comparison$table) ## The full contrast table
```

```
##                  logFC    AveExpr         t       P.Value  adj.P.Val
## gene_9453_F -3.674407  4.6687613 -4.751151 0.00000205079 0.02050790
## gene_1147_F  3.195330 -0.5000518  2.662580 0.00776681755 0.88924321
## gene_6962_F -3.030009  0.5821939 -2.894170 0.00380984540 0.84663231
## gene_7746_F  2.991793  0.5039044  2.940911 0.00327998028 0.84402453
## gene_5460_F  2.929328  5.8334237  4.295525 0.00001759273 0.08796365
## gene_745_T   2.903883 -1.5219175  2.258808 0.02391675544 0.99796408
##                     B
## gene_9453_F -1.244491
## gene_1147_F -4.116820
## gene_6962_F -3.870834
## gene_7746_F -3.809143
## gene_5460_F -1.412287
## gene_745_T  -4.309751
```

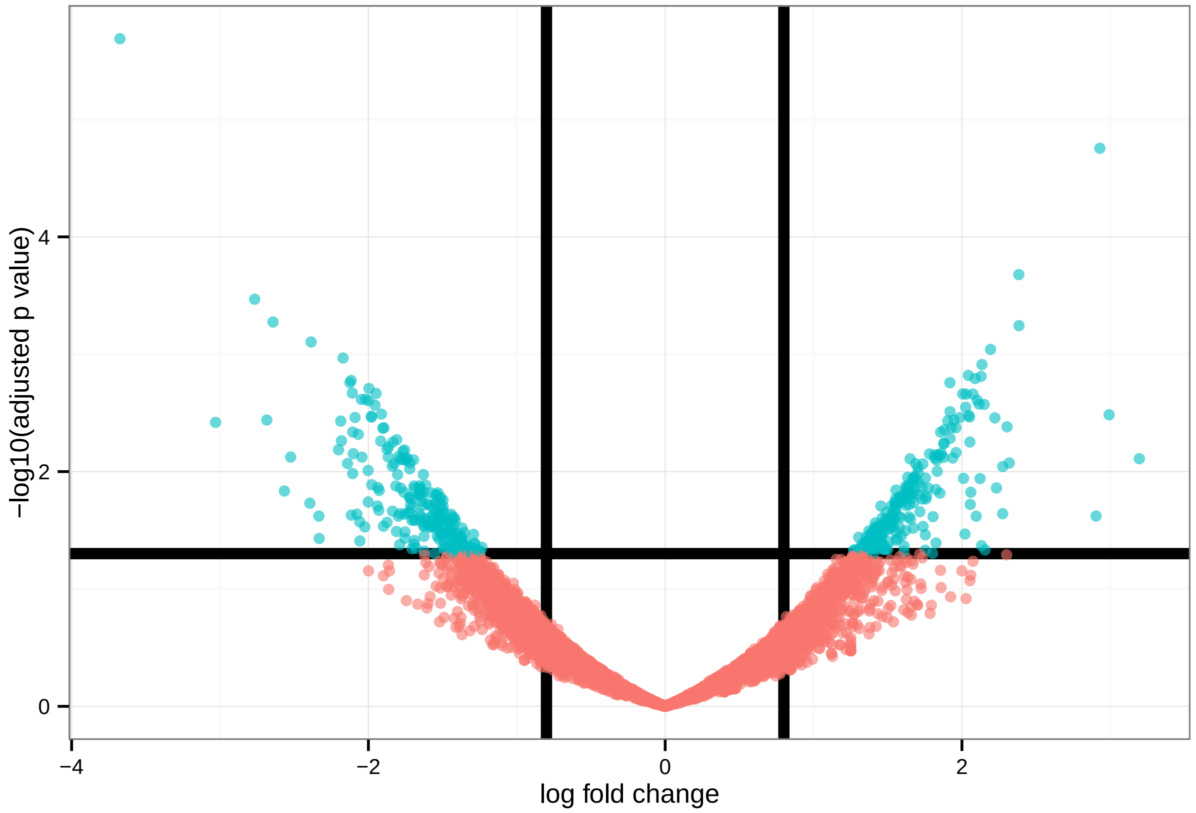
```
head(ab_comparison$upsignificant)  ## The list of genes which are significantly up in b vs a
```

```
##                 logFC    AveExpr        t       P.Value  adj.P.Val
## gene_1147_F 3.195330 -0.5000518 2.662580 0.00776681755 0.88924321
## gene_7746_F 2.991793  0.5039044 2.940911 0.00327998028 0.84402453
## gene_5460_F 2.929328  5.8334237 4.295525 0.00001759273 0.08796365
## gene_745_T  2.903883 -1.5219175 2.258808 0.02391675544 0.99796408
## gene_6055_F 2.384475  5.2700388 3.446193 0.00057088066 0.84402453
## gene_5451_T 2.383115  8.3595141 3.708323 0.00020976507 0.69921690
##                     B
## gene_1147_F -4.116820
## gene_7746_F -3.809143
## gene_5460_F -1.412287
## gene_745_T  -4.309751
## gene_6055_F -2.663534
## gene_5451_T -2.111578
```

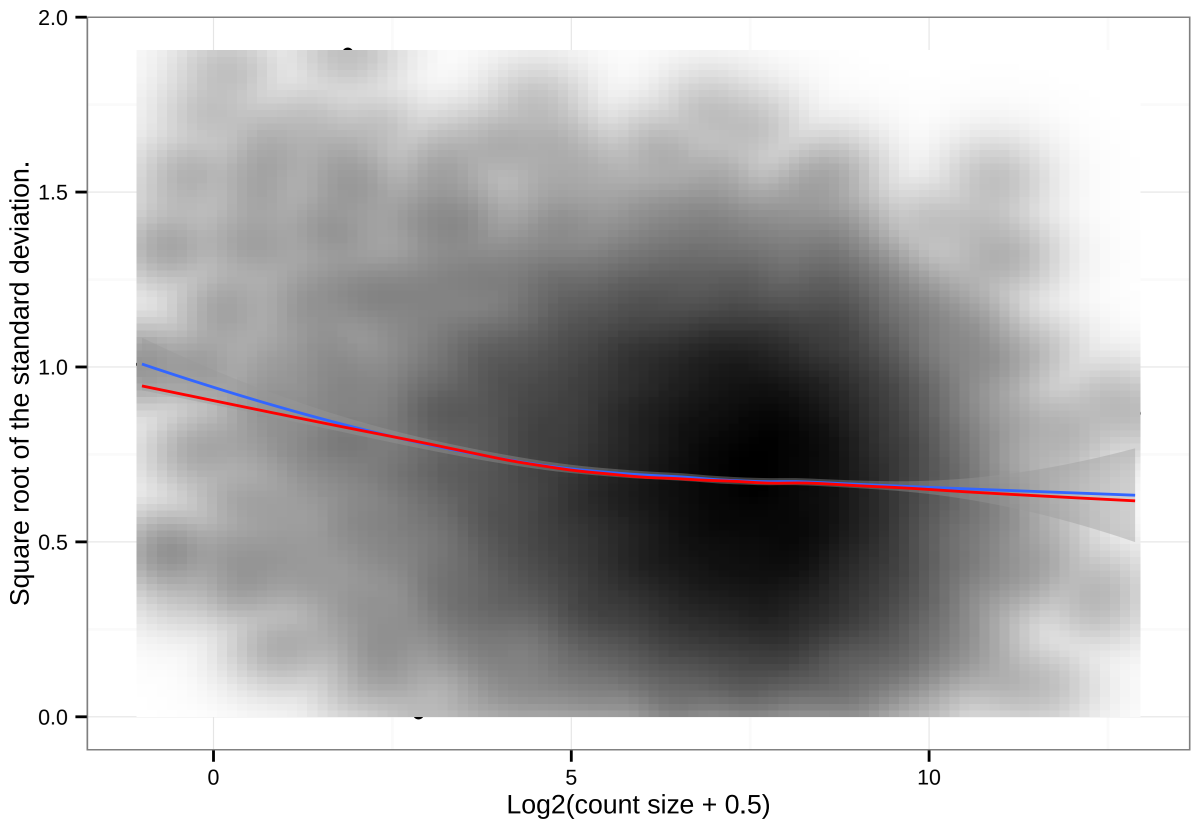
```
dim(ab_comparison$upsignificant)
```

```
## [1] 2029    6
```

```
print(ab_comparison$volcano_plot) ## A Volcano plot of b vs a
```



```
## ab_comparison$voom_data  ## The output from voom()
print(ab_comparison$voom_plot) ## A ggplot2 version of the mean/variance trend provided by voom()
```



```
## The data structure ab_comparison$comparisons contains the output from eBayes() which comprises the la
## limma step...
funkytown = write_limma(data=ab_comparison$comparisons, excel=FALSE, csv=FALSE)
```

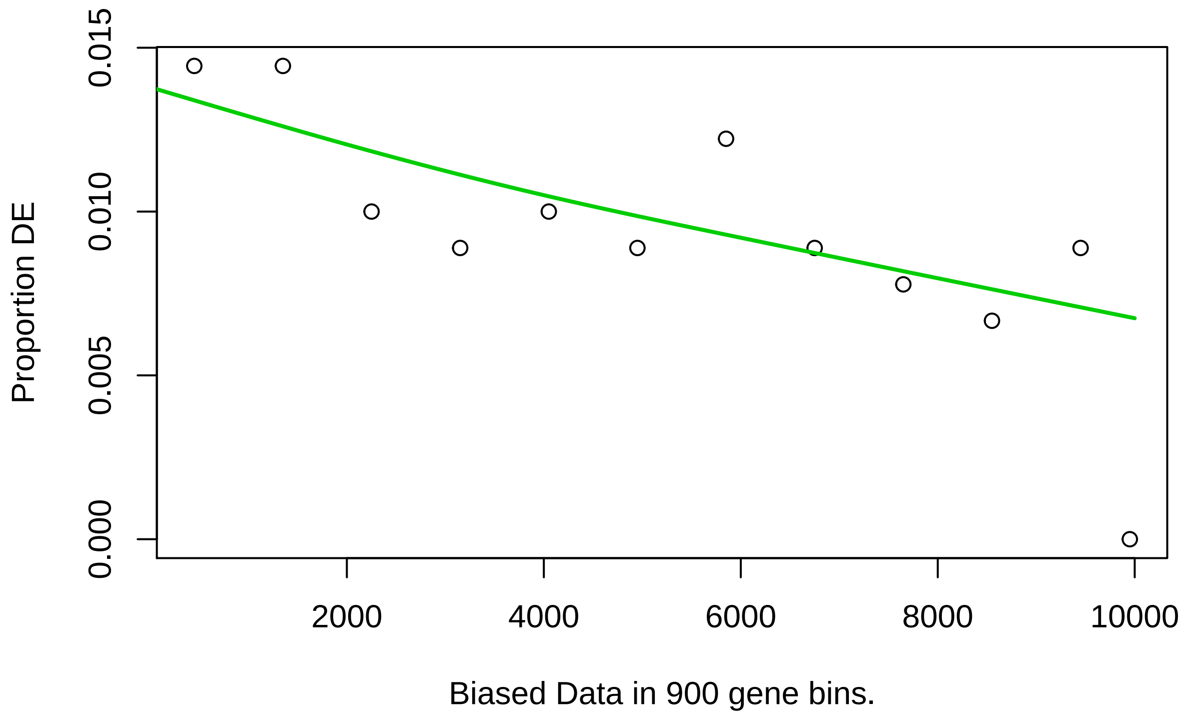
```
## 1/1: Printing table: changed_v_control
```

```
## Lets make up some gene lengths
gene_lengths = funkytown[[1]]
gene_lengths$width = sample(nrow(gene_lengths))
gene_lengths$ID = rownames(gene_lengths)
gene_lengths = gene_lengths[,c("ID","width")]

## And some GO categories
goids=funkytown[[1]]
all_go_categories = AnnotationDbi::keys(GO.db)
goids$GO = sample(all_go_categories, nrow(gene_lengths))
goids$ID = rownames(goids)
goids = goids[,c("ID","GO")]

ontology_fun = limma_ontology(funkytown, gene_lengths=gene_lengths, goids=goids, n=100, overwrite=TRUE)
```

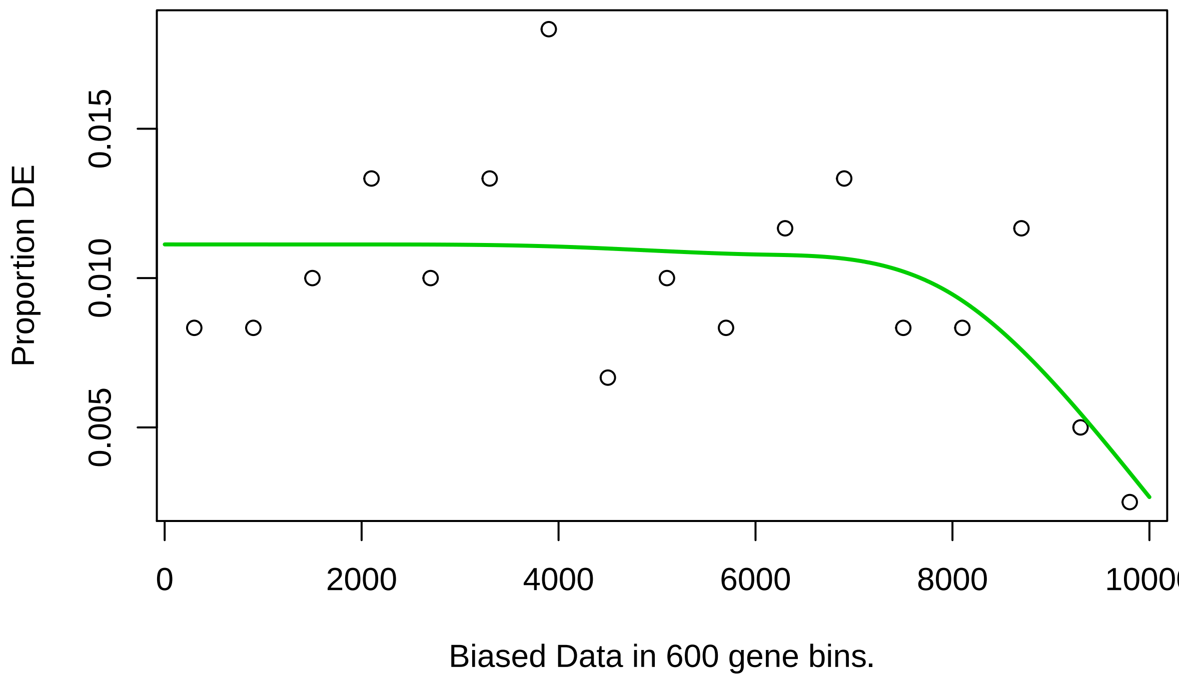
```
## This function expects a list of limma contrast tables and some annotation information.
## The annotation information would be gene lengths and ontology ids
## Performing ontology search of:changed_v_control
## simple_goseq() makes some pretty hard assumptions about the data it is fed:
## It requires 2 tables, one of GOids which must have columns (gene)ID and GO(category)
## The other table is of gene lengths with columns (gene)ID and (gene)width.
## Other columns are fine, but ignored.
## Using the length data to fill in the de vector.
## Using manually entered categories.
## Calculating the p-values...
## Calculating q-values
## There are no genes with an adjusted pvalue < 0.1 using method: BH.
## Providing genes with an un-adjusted pvalue < 0.1
## Filling godata table with term information, this takes a while.
```

```
## [1] "Testing that go categories are defined."
## [1] "Removing undefined categories."
## [1] "Gathering synonyms."
## [1] "Gathering secondary ids."
## [1] "Gathering category definitions."
```

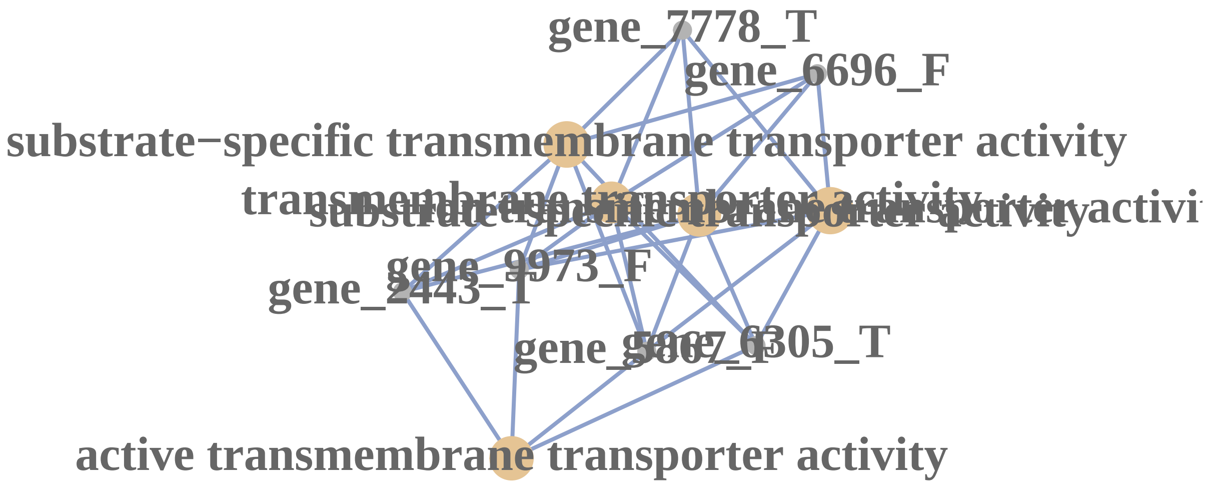
```
## Making pvalue plots for the ontologies.
## simple_goseq() makes some pretty hard assumptions about the data it is fed:
## It requires 2 tables, one of GOids which must have columns (gene)ID and GO(category)
## The other table is of gene lengths with columns (gene)ID and (gene)width.
## Other columns are fine, but ignored.
## Using the length data to fill in the de vector.
```

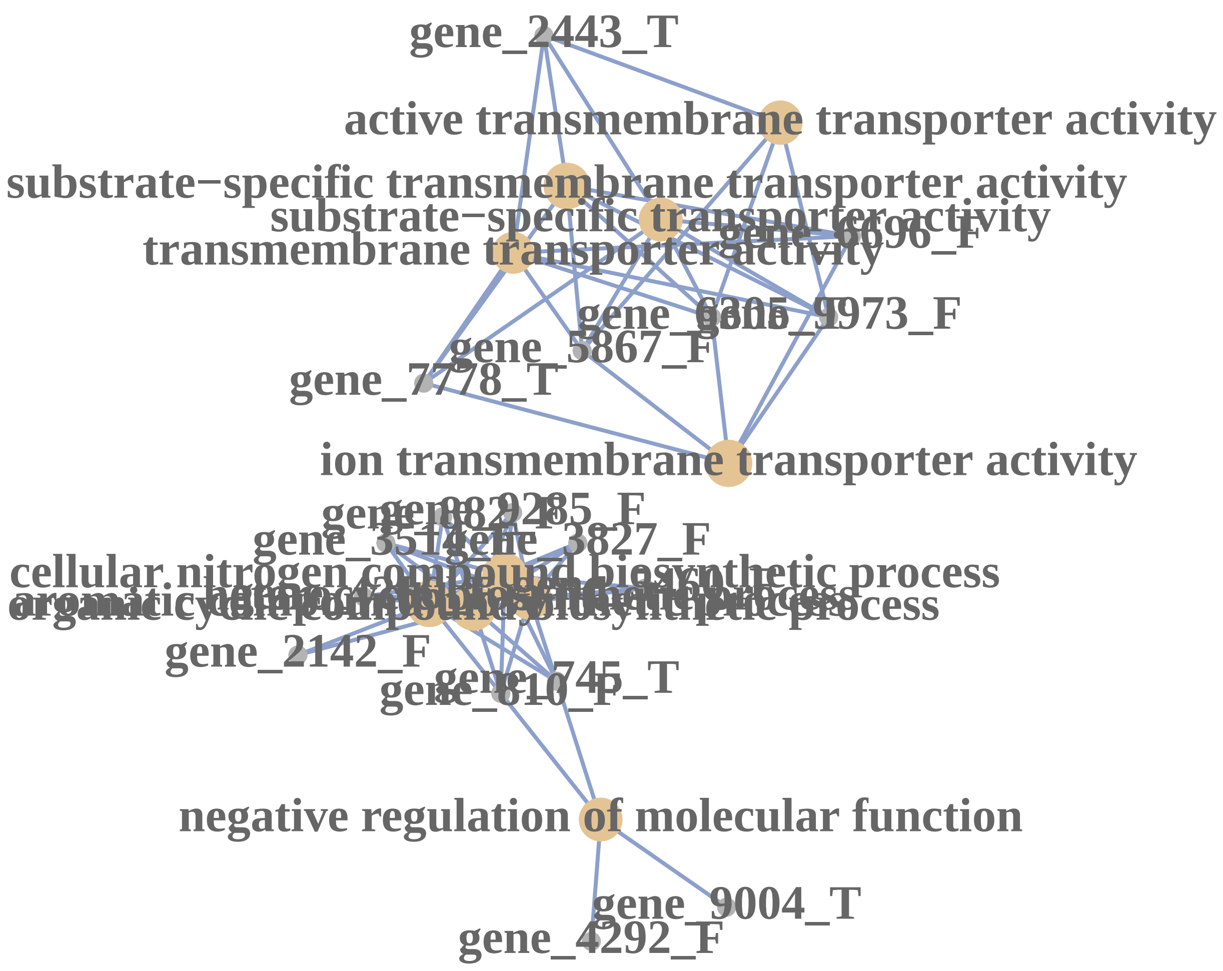


```
## Using manually entered categories.
## Calculating the p-values...
## Calculating q-values
## There are no genes with an adjusted pvalue < 0.1 using method: BH.
## Providing genes with an un-adjusted pvalue < 0.1
## Filling godata table with term information, this takes a while.
## [1] "Testing that go categories are defined."
## [1] "Removing undefined categories."
## [1] "Gathering synonyms."
## [1] "Gathering secondary ids."
## [1] "Gathering category definitions."

## Making pvalue plots for the ontologies.

## Warning in readChar(con, 5L, useBytes = TRUE): cannot open compressed file
## 'geneTable.rda', probable reason 'No such file or directory'

## Generating the geneTable.rda
## Gene Table file save in the working directory.

## Warning in readChar(con, 5L, useBytes = TRUE): cannot open compressed file
## 'GO2EG.rda', probable reason 'No such file or directory'

## Generating GO mapping data for cluster profiler from the goids data.

## [1] "GO Annotation Mapping files save in the working directory."

## Starting MF(molecular function) analysis

## The minimum observed adjusted pvalue is: 0.019450
## The minimum observed adjusted pvalue is: 0.092317

## Starting BP(biological process) analysis

## The minimum observed adjusted pvalue is: 0.005319
## The minimum observed adjusted pvalue is: 0.142921

## Starting CC(cellular component) analysis

## The minimum observed adjusted pvalue is: 0.040365
## The minimum observed adjusted pvalue is: 0.354832

## Attempting to include the cnetplots from clusterProfiler.
## They fail often, if this is causing errors, set:
## include_cnetplots to FALSE
```

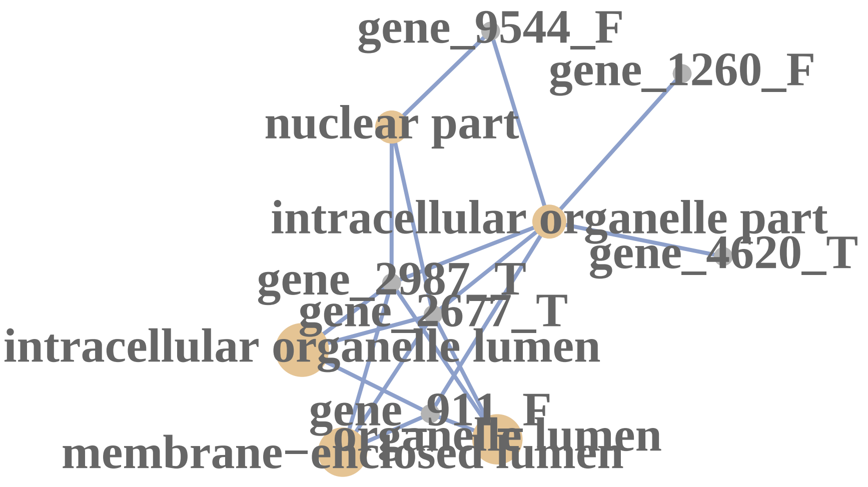


```
## cnetplot just failed for the BP ontology.  Do not be concerned with the previous error.
## cnetplot just failed for the CC ontology.  Do not be concerned with the previous error.
```

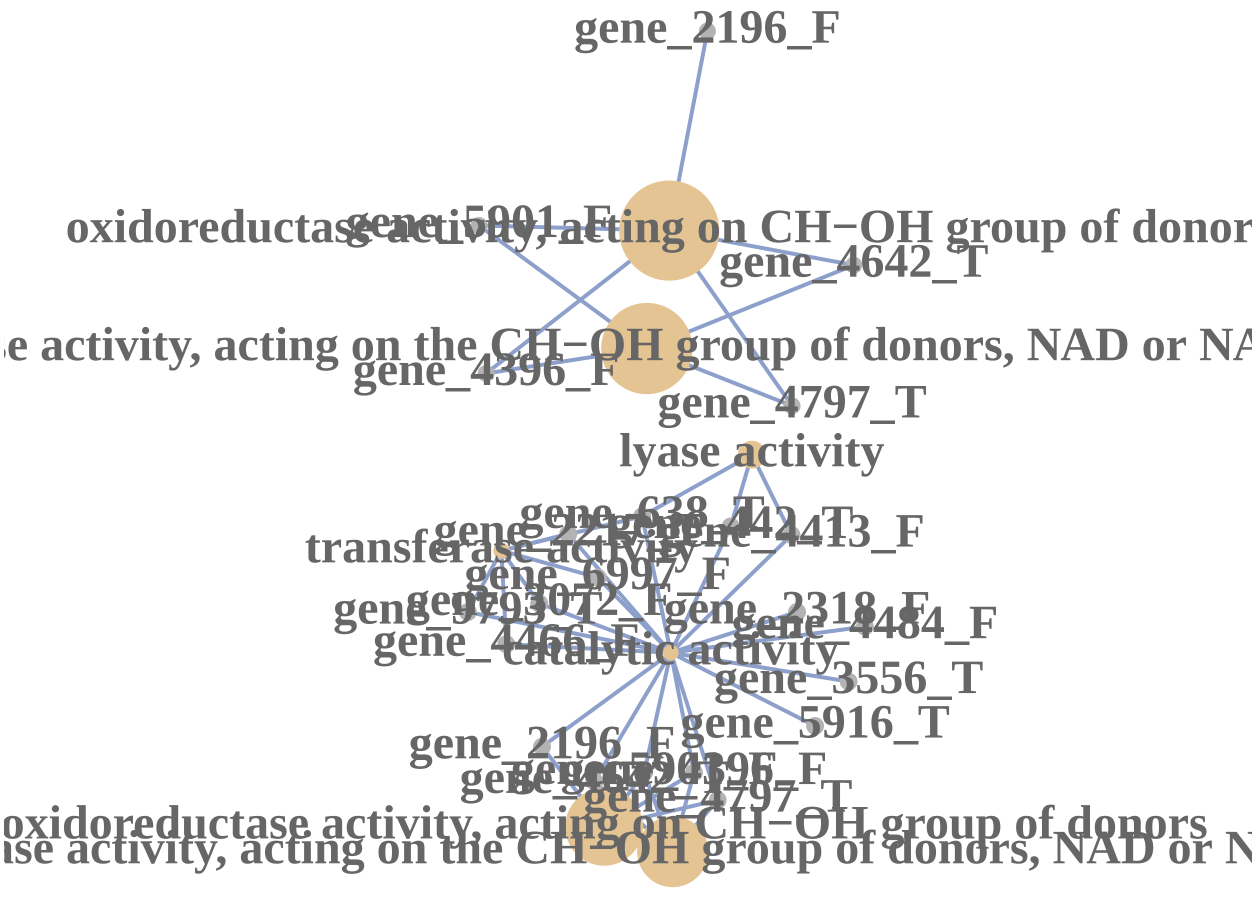

gene_7778_T
gene_6696_F
substrate−specific transmembrane transporter activity
transmembrane transporter activity
substrate−specific transporter activity
gene_9973_F
gene_2443_T
gene_5867_T
gene_6305_T
active transmembrane transporter activity

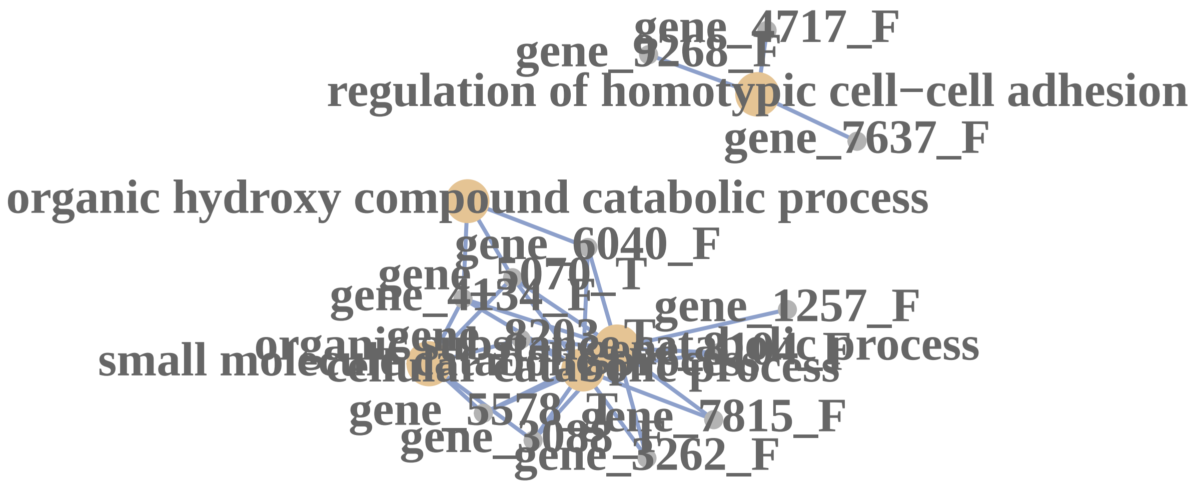

gene_2443_T
active transmembrane transporter activity
substrate−specific transmembrane transporter activity
substrate−specific transporter activity
gene_6696_F
transmembrane transporter activity
gene_5973_F
gene_5867_F
gene_7778_T
ion transmembrane transporter activity
gene_3827_F
gene_2142_F
gene_745_T
cellular nitrogen compound biosynthetic process
organic cyclic compound biosynthetic process
negative regulation of molecular function
gene_9004_T
gene_4292_F

```
## Using GO mapping data located in GO2EG.rda
## Starting MF(molecular function) analysis

## The minimum observed adjusted pvalue is: 0.003653
## The minimum observed adjusted pvalue is: 0.027214

## Starting BP(biological process) analysis

## The minimum observed adjusted pvalue is: 0.009256
## The minimum observed adjusted pvalue is: 0.260803

## Starting CC(cellular component) analysis

## The minimum observed adjusted pvalue is: 0.023982
## The minimum observed adjusted pvalue is: 0.135569

## Attempting to include the cnetplots from clusterProfiler.
## They fail often, if this is causing errors, set:
## include_cnetplots to FALSE
```



```
## cnetplot just failed for the BP ontology.  Do not be concerned with the previous error.
## cnetplot just failed for the CC ontology.  Do not be concerned with the previous error.
```




gene_4717_F
gene_9268_F
regulation of homotypic cell−cell adhesion
gene_7637_F
organic hydroxy compound catabolic process
gene_6040_F
gene_5070_T
gene_4134_F
gene_1257_F
small molecule catabolic process
organic substance catabolic process
cellular catabolic process
gene_8203_T
gene_3104_F
gene_5578_T
gene_7815_F
gene_3088_T
gene_3262_F

```
## Attempting to generate a id2go file in the format expected by topGO.

##
## Building most specific GOs ..... ( 27 GO terms found. )
##
## Build GO DAG topology .......... ( 143 GO terms and 171 relations. )
##
## Annotating nodes ............... ( 27 genes annotated to the GO terms. )
##
## Building most specific GOs ..... ( 65 GO terms found. )
##
## Build GO DAG topology .......... ( 869 GO terms and 1835 relations. )
##
## Annotating nodes ............... ( 65 genes annotated to the GO terms. )
##
## Building most specific GOs ..... ( 8 GO terms found. )
##
## Build GO DAG topology .......... ( 81 GO terms and 148 relations. )
##
## Annotating nodes ............... ( 8 genes annotated to the GO terms. )
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 105 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 528 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 51 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
```

```
##
##          -- Classic Algorithm --
##
##      the algorithm is scoring 143 nontrivial nodes
##      parameters:
##          test statistic:  KS tests
##          score order:  increasing
##
##          -- Classic Algorithm --
##
##      the algorithm is scoring 869 nontrivial nodes
##      parameters:
##          test statistic:  KS tests
##          score order:  increasing
##
##          -- Classic Algorithm --
##
##      the algorithm is scoring 81 nontrivial nodes
##      parameters:
##          test statistic:  KS tests
##          score order:  increasing
##
##          -- Elim Algorithm --
##
##      the algorithm is scoring 143 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 13:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 12:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 11:  3 nodes to be scored    (0 eliminated genes)
##
##   Level 10:  4 nodes to be scored    (0 eliminated genes)
##
##   Level 9:   7 nodes to be scored    (0 eliminated genes)
##
##   Level 8:   9 nodes to be scored    (0 eliminated genes)
##
##   Level 7:   17 nodes to be scored   (0 eliminated genes)
##
##   Level 6:   30 nodes to be scored   (0 eliminated genes)
##
##   Level 5:   30 nodes to be scored   (0 eliminated genes)
##
##   Level 4:   24 nodes to be scored   (0 eliminated genes)
##
##   Level 3:   11 nodes to be scored   (0 eliminated genes)
##
##   Level 2:   4 nodes to be scored    (0 eliminated genes)
##
```

```
##   Level 1:   1 nodes to be scored    (0 eliminated genes)
##
##          -- Elim Algorithm --
##
##      the algorithm is scoring 869 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 15:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 14:  4 nodes to be scored    (0 eliminated genes)
##
##   Level 13:  11 nodes to be scored   (0 eliminated genes)
##
##   Level 12:  20 nodes to be scored   (0 eliminated genes)
##
##   Level 11:  34 nodes to be scored   (0 eliminated genes)
##
##   Level 10:  57 nodes to be scored   (0 eliminated genes)
##
##   Level 9:   73 nodes to be scored   (0 eliminated genes)
##
##   Level 8:   98 nodes to be scored   (0 eliminated genes)
##
##   Level 7:   128 nodes to be scored  (0 eliminated genes)
##
##   Level 6:   150 nodes to be scored  (0 eliminated genes)
##
##   Level 5:   146 nodes to be scored  (0 eliminated genes)
##
##   Level 4:   88 nodes to be scored   (0 eliminated genes)
##
##   Level 3:   40 nodes to be scored   (6 eliminated genes)
##
##   Level 2:   17 nodes to be scored   (6 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (6 eliminated genes)
##
##          -- Elim Algorithm --
##
##      the algorithm is scoring 81 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 15:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 14:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 13:  2 nodes to be scored    (0 eliminated genes)
##
```

```
##   Level 12:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 11:  5 nodes to be scored    (0 eliminated genes)
##
##   Level 10:  10 nodes to be scored   (0 eliminated genes)
##
##   Level 9:   11 nodes to be scored   (0 eliminated genes)
##
##   Level 8:   8 nodes to be scored    (0 eliminated genes)
##
##   Level 7:   7 nodes to be scored    (0 eliminated genes)
##
##   Level 6:   6 nodes to be scored    (0 eliminated genes)
##
##   Level 5:   7 nodes to be scored    (0 eliminated genes)
##
##   Level 4:   8 nodes to be scored    (0 eliminated genes)
##
##   Level 3:   6 nodes to be scored    (0 eliminated genes)
##
##   Level 2:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (0 eliminated genes)
##
##           -- Weight Algorithm --
##
##       The algorithm is scoring 105 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test : ratio
##
##   Level 13:  1 nodes to be scored.
##
##   Level 12:  2 nodes to be scored.
##
##   Level 11:  3 nodes to be scored.
##
##   Level 10:  4 nodes to be scored.
##
##   Level 9:   6 nodes to be scored.
##
##   Level 8:   5 nodes to be scored.
##
##   Level 7:   11 nodes to be scored.
##
##   Level 6:   20 nodes to be scored.
##
##   Level 5:   22 nodes to be scored.
##
##   Level 4:   16 nodes to be scored.
##
##   Level 3:   10 nodes to be scored.
##
##   Level 2:   4 nodes to be scored.
##
```

```
##   Level 1:   1 nodes to be scored.
##
##          -- Weight Algorithm --
##
##      The algorithm is scoring 528 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test : ratio
##
##   Level 15:  2 nodes to be scored.
##
##   Level 14:  2 nodes to be scored.
##
##   Level 13:  5 nodes to be scored.
##
##   Level 12:  10 nodes to be scored.
##
##   Level 11:  18 nodes to be scored.
##
##   Level 10:  31 nodes to be scored.
##
##   Level 9:   42 nodes to be scored.
##
##   Level 8:   54 nodes to be scored.
##
##   Level 7:   70 nodes to be scored.
##
##   Level 6:   86 nodes to be scored.
##
##   Level 5:   98 nodes to be scored.
##
##   Level 4:   60 nodes to be scored.
##
##   Level 3:   34 nodes to be scored.
##
##   Level 2:   15 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
##
##          -- Weight Algorithm --
##
##      The algorithm is scoring 51 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test : ratio
##
##   Level 12:  1 nodes to be scored.
##
##   Level 11:  2 nodes to be scored.
##
##   Level 10:  4 nodes to be scored.
##
##   Level 9:   8 nodes to be scored.
##
##   Level 8:   5 nodes to be scored.
##
```

```
##   Level 7:   4 nodes to be scored.
##
##   Level 6:   3 nodes to be scored.
##
##   Level 5:   6 nodes to be scored.
##
##   Level 4:   7 nodes to be scored.
##
##   Level 3:   5 nodes to be scored.
##
##   Level 2:   5 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
##
## Building most specific GOs ..... ( 24 GO terms found. )
##
## Build GO DAG topology ......... ( 123 GO terms and 151 relations. )
##
## Annotating nodes .............. ( 24 genes annotated to the GO terms. )
##
## Building most specific GOs ..... ( 67 GO terms found. )
##
## Build GO DAG topology ......... ( 969 GO terms and 2118 relations. )
##
## Annotating nodes .............. ( 67 genes annotated to the GO terms. )
##
## Building most specific GOs ..... ( 9 GO terms found. )
##
## Build GO DAG topology ......... ( 52 GO terms and 83 relations. )
##
## Annotating nodes .............. ( 9 genes annotated to the GO terms. )
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 77 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 673 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 18 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 123 nontrivial nodes
##       parameters:
```

```
##          test statistic:  KS tests
##          score order:  increasing
##
##          -- Classic Algorithm --
##
##      the algorithm is scoring 969 nontrivial nodes
##      parameters:
##          test statistic:  KS tests
##          score order:  increasing
##
##          -- Classic Algorithm --
##
##      the algorithm is scoring 52 nontrivial nodes
##      parameters:
##          test statistic:  KS tests
##          score order:  increasing
##
##          -- Elim Algorithm --
##
##      the algorithm is scoring 123 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 14:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 13:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 12:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 11:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 10:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 9:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 8:   6 nodes to be scored    (0 eliminated genes)
##
##   Level 7:   14 nodes to be scored   (0 eliminated genes)
##
##   Level 6:   27 nodes to be scored   (0 eliminated genes)
##
##   Level 5:   25 nodes to be scored   (0 eliminated genes)
##
##   Level 4:   18 nodes to be scored   (0 eliminated genes)
##
##   Level 3:   15 nodes to be scored   (0 eliminated genes)
##
##   Level 2:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (0 eliminated genes)
##
##          -- Elim Algorithm --
```

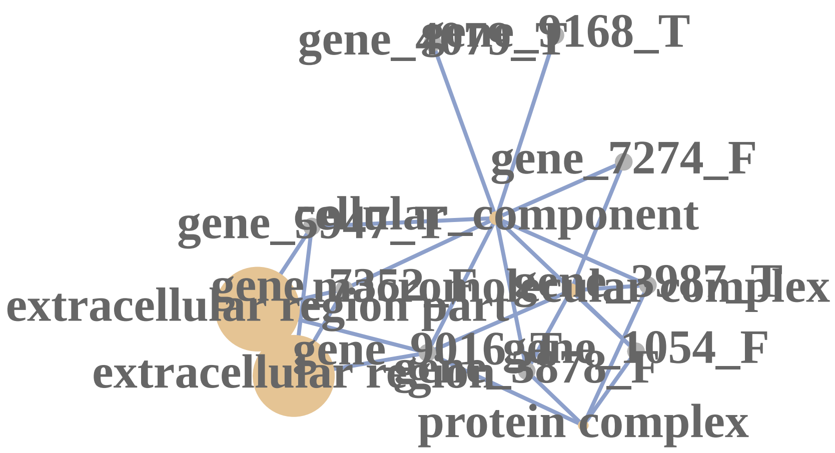
```
##
##      the algorithm is scoring 969 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 19:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 18:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 17:  3 nodes to be scored    (0 eliminated genes)
##
##   Level 16:  4 nodes to be scored    (0 eliminated genes)
##
##   Level 15:  7 nodes to be scored    (0 eliminated genes)
##
##   Level 14:  14 nodes to be scored   (0 eliminated genes)
##
##   Level 13:  26 nodes to be scored   (0 eliminated genes)
##
##   Level 12:  38 nodes to be scored   (0 eliminated genes)
##
##   Level 11:  50 nodes to be scored   (0 eliminated genes)
##
##   Level 10:  64 nodes to be scored   (0 eliminated genes)
##
##   Level 9:   79 nodes to be scored   (0 eliminated genes)
##
##   Level 8:   95 nodes to be scored   (0 eliminated genes)
##
##   Level 7:   131 nodes to be scored  (0 eliminated genes)
##
##   Level 6:   161 nodes to be scored  (7 eliminated genes)
##
##   Level 5:   142 nodes to be scored  (7 eliminated genes)
##
##   Level 4:   88 nodes to be scored   (7 eliminated genes)
##
##   Level 3:   44 nodes to be scored   (7 eliminated genes)
##
##   Level 2:   19 nodes to be scored   (7 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (7 eliminated genes)
##
##          -- Elim Algorithm --
##
##      the algorithm is scoring 52 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 10:  2 nodes to be scored    (0 eliminated genes)
```

```
##
##   Level 9:   3 nodes to be scored    (0 eliminated genes)
##
##   Level 8:   3 nodes to be scored    (0 eliminated genes)
##
##   Level 7:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 6:   9 nodes to be scored    (0 eliminated genes)
##
##   Level 5:   8 nodes to be scored    (0 eliminated genes)
##
##   Level 4:   9 nodes to be scored    (0 eliminated genes)
##
##   Level 3:   7 nodes to be scored    (0 eliminated genes)
##
##   Level 2:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (0 eliminated genes)
##
##           -- Weight Algorithm --
##
##       The algorithm is scoring 77 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test : ratio
##
##   Level 14:  1 nodes to be scored.
##
##   Level 13:  1 nodes to be scored.
##
##   Level 12:  2 nodes to be scored.
##
##   Level 11:  1 nodes to be scored.
##
##   Level 10:  2 nodes to be scored.
##
##   Level 9:   3 nodes to be scored.
##
##   Level 8:   3 nodes to be scored.
##
##   Level 7:   9 nodes to be scored.
##
##   Level 6:   16 nodes to be scored.
##
##   Level 5:   13 nodes to be scored.
##
##   Level 4:   11 nodes to be scored.
##
##   Level 3:   10 nodes to be scored.
##
##   Level 2:   4 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
##
##           -- Weight Algorithm --
```

```
##
##       The algorithm is scoring 673 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test : ratio
##
##   Level 19:  1 nodes to be scored.
##
##   Level 18:  2 nodes to be scored.
##
##   Level 17:  3 nodes to be scored.
##
##   Level 16:  4 nodes to be scored.
##
##   Level 15:  6 nodes to be scored.
##
##   Level 14:  11 nodes to be scored.
##
##   Level 13:  20 nodes to be scored.
##
##   Level 12:  30 nodes to be scored.
##
##   Level 11:  38 nodes to be scored.
##
##   Level 10:  38 nodes to be scored.
##
##   Level 9:   51 nodes to be scored.
##
##   Level 8:   60 nodes to be scored.
##
##   Level 7:   82 nodes to be scored.
##
##   Level 6:   106 nodes to be scored.
##
##   Level 5:   104 nodes to be scored.
##
##   Level 4:   67 nodes to be scored.
##
##   Level 3:   35 nodes to be scored.
##
##   Level 2:   14 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
##
##           -- Weight Algorithm --
##
##       The algorithm is scoring 18 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test : ratio
##
##   Level 7:   1 nodes to be scored.
##
##   Level 6:   1 nodes to be scored.
##
##   Level 5:   2 nodes to be scored.
```

```
##
##   Level 4:   3 nodes to be scored.
##
##   Level 3:   6 nodes to be scored.
##
##   Level 2:   4 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
```

```
## Warning in data.frame(infoMat, annoStat, apply(1, 2, format.FUN, dig = 2, :
## row names were found from a short variable and have been discarded
```



```
testme = head(funkytown[[1]], n=40)
tt = simple_clusterprofiler(testme, goids=goids, gff=goids)
```

```
## Using GO mapping data located in GO2EG.rda
## Starting MF(molecular function) analysis
```

```
## The minimum observed adjusted pvalue is: 0.240804
## The minimum observed adjusted pvalue is: 0.417056
```

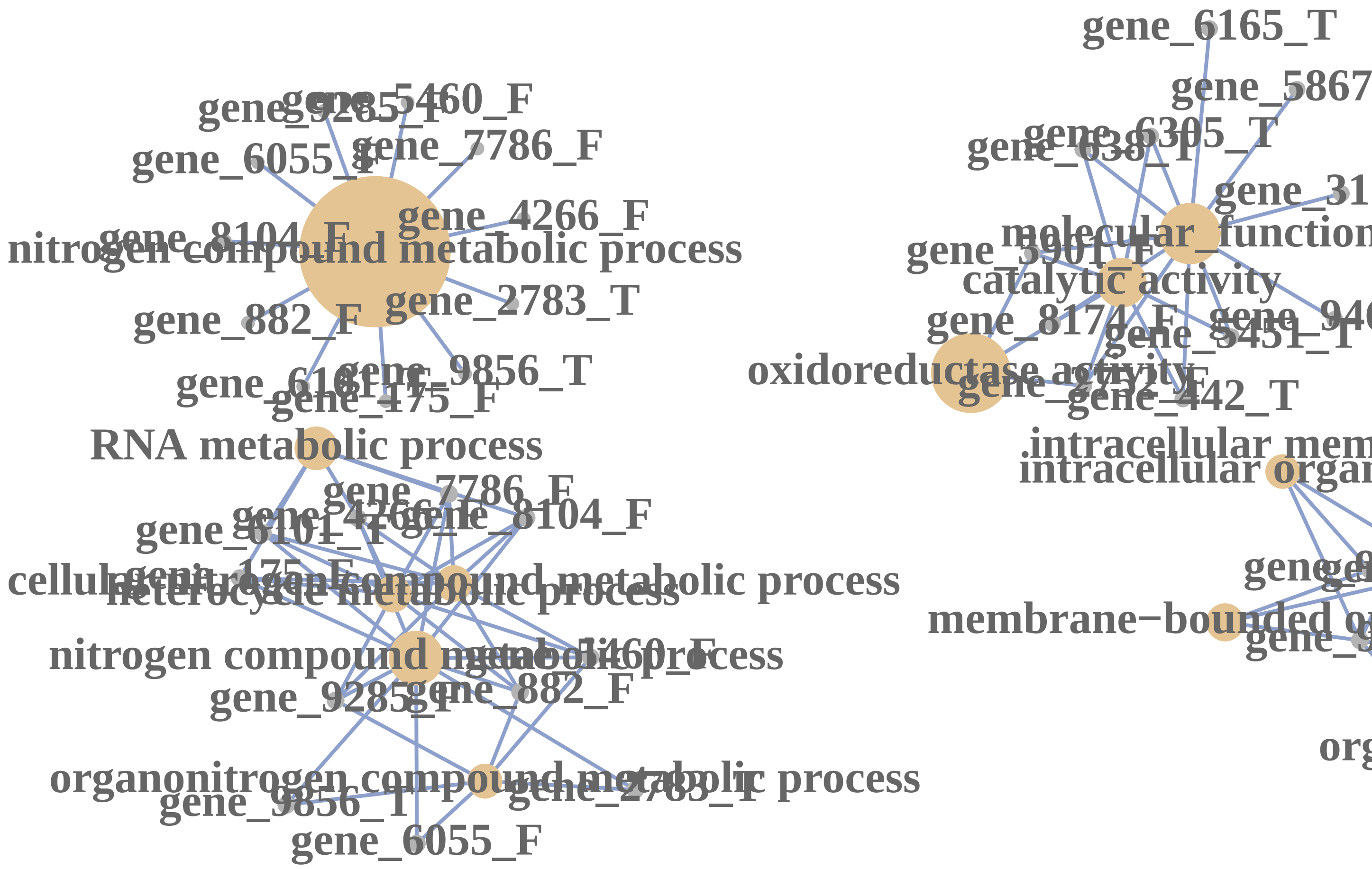
```
## Starting BP(biological process) analysis
```

```
## The minimum observed adjusted pvalue is: 0.000889
## The minimum observed adjusted pvalue is: 0.053347
```

```
## Starting CC(cellular component) analysis
```

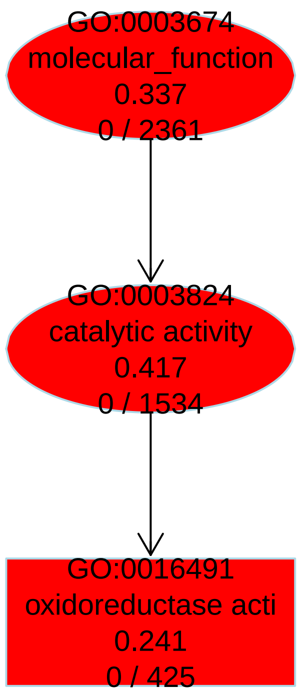
```
## The minimum observed adjusted pvalue is: 0.061645
## The minimum observed adjusted pvalue is: 0.359815
```

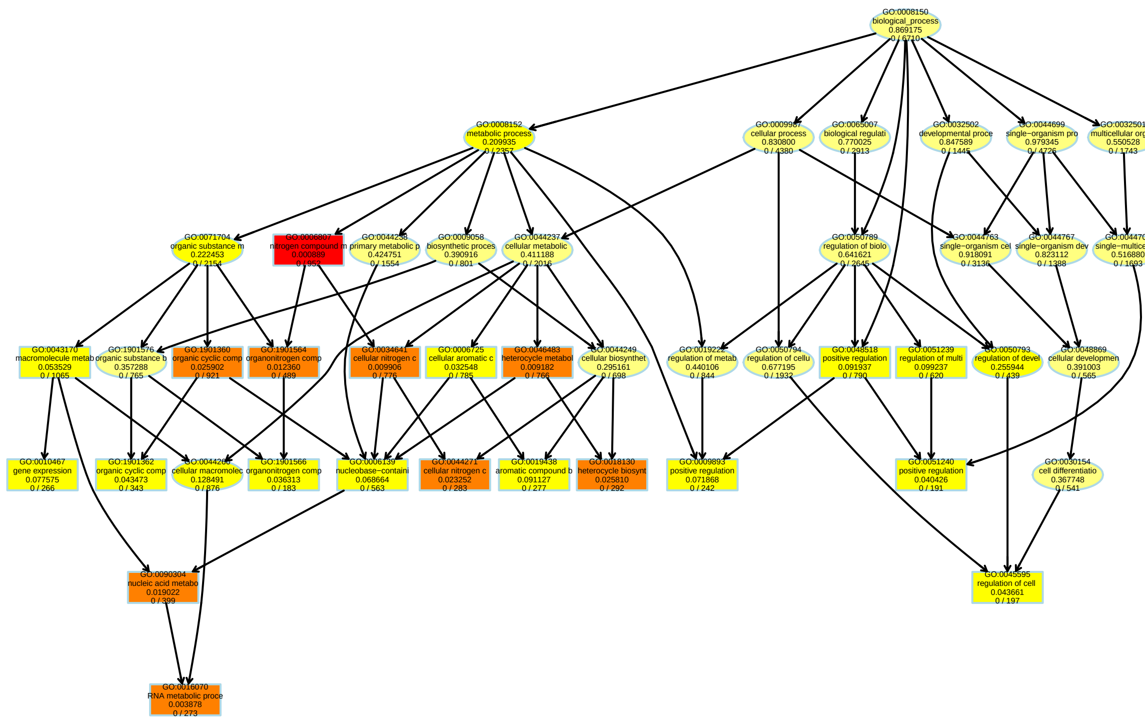
```
## Attempting to include the cnetplots from clusterProfiler.
## They fail often, if this is causing errors, set:
## include_cnetplots to FALSE
## cnetplot just failed for the MF ontology.  Do not be concerned with the previous error.
## cnetplot just failed for the CC ontology.  Do not be concerned with the previous error.
```

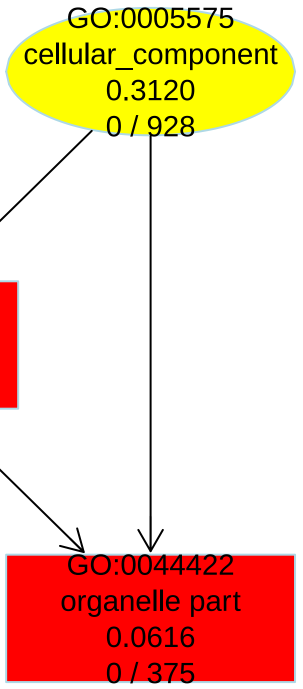


```r
ttt = cluster_trees(testme, tt)
```

```
##
## Building most specific GOs ..... ( 2361 GO terms found. )
##
## Build GO DAG topology ......... ( 3342 GO terms and 4180 relations. )
##
## Annotating nodes ............... ( 2361 genes annotated to the GO terms. )
##
## Building most specific GOs ..... ( 6710 GO terms found. )
##
## Build GO DAG topology ......... ( 14519 GO terms and 34643 relations. )
##
## Annotating nodes ............... ( 6710 genes annotated to the GO terms. )
##
## Building most specific GOs ..... ( 928 GO terms found. )
##
## Build GO DAG topology ......... ( 1539 GO terms and 2950 relations. )
##
## Annotating nodes ............... ( 928 genes annotated to the GO terms. )
```







```
tttt = simple_topgo(testme)
```

```
##
## Building most specific GOs ..... ( 11 GO terms found. )
##
## Build GO DAG topology .......... ( 74 GO terms and 86 relations. )
##
## Annotating nodes ............... ( 11 genes annotated to the GO terms. )
##
```

```
## Building most specific GOs ..... ( 24 GO terms found. )
##
## Build GO DAG topology ......... ( 486 GO terms and 1067 relations. )
##
## Annotating nodes .............. ( 24 genes annotated to the GO terms. )
##
## Building most specific GOs ..... ( 5 GO terms found. )
##
## Build GO DAG topology ......... ( 58 GO terms and 106 relations. )
##
## Annotating nodes .............. ( 5 genes annotated to the GO terms. )
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 74 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 486 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 58 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 74 nontrivial nodes
##       parameters:
##           test statistic:  KS tests
##           score order:  increasing
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 486 nontrivial nodes
##       parameters:
##           test statistic:  KS tests
##           score order:  increasing
##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 58 nontrivial nodes
##       parameters:
##           test statistic:  KS tests
##           score order:  increasing
##
##           -- Elim Algorithm --
##
##       the algorithm is scoring 74 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 13:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 12:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 11:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 10:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 9:   3 nodes to be scored    (0 eliminated genes)
##
##   Level 8:   3 nodes to be scored    (0 eliminated genes)
##
##   Level 7:   7 nodes to be scored    (0 eliminated genes)
##
##   Level 6:   15 nodes to be scored   (0 eliminated genes)
##
##   Level 5:   14 nodes to be scored   (0 eliminated genes)
##
##   Level 4:   11 nodes to be scored   (0 eliminated genes)
##
##   Level 3:   11 nodes to be scored   (0 eliminated genes)
##
##   Level 2:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (0 eliminated genes)
##
##          -- Elim Algorithm --
##
##      the algorithm is scoring 486 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test
##          cutOff:  0.01
##          score order:  increasing
##
##   Level 19:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 18:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 17:  3 nodes to be scored    (0 eliminated genes)
##
##   Level 16:  3 nodes to be scored    (0 eliminated genes)
##
##   Level 15:  4 nodes to be scored    (0 eliminated genes)
##
##   Level 14:  8 nodes to be scored    (0 eliminated genes)
##
##   Level 13:  13 nodes to be scored   (0 eliminated genes)
##
##   Level 12:  21 nodes to be scored   (0 eliminated genes)
```

```
##
##   Level 11:  22 nodes to be scored   (0 eliminated genes)
##
##   Level 10:  22 nodes to be scored   (0 eliminated genes)
##
##   Level 9:   33 nodes to be scored   (0 eliminated genes)
##
##   Level 8:   43 nodes to be scored   (0 eliminated genes)
##
##   Level 7:   55 nodes to be scored   (0 eliminated genes)
##
##   Level 6:   75 nodes to be scored   (0 eliminated genes)
##
##   Level 5:   82 nodes to be scored   (0 eliminated genes)
##
##   Level 4:   53 nodes to be scored   (0 eliminated genes)
##
##   Level 3:   32 nodes to be scored   (0 eliminated genes)
##
##   Level 2:   13 nodes to be scored   (0 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (15 eliminated genes)
##
##           -- Elim Algorithm --
##
##       the algorithm is scoring 58 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test
##           cutOff:  0.01
##           score order:  increasing
##
##   Level 12:  1 nodes to be scored    (0 eliminated genes)
##
##   Level 11:  2 nodes to be scored    (0 eliminated genes)
##
##   Level 10:  4 nodes to be scored    (0 eliminated genes)
##
##   Level 9:   8 nodes to be scored    (0 eliminated genes)
##
##   Level 8:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 7:   5 nodes to be scored    (0 eliminated genes)
##
##   Level 6:   4 nodes to be scored    (0 eliminated genes)
##
##   Level 5:   7 nodes to be scored    (0 eliminated genes)
##
##   Level 4:   8 nodes to be scored    (0 eliminated genes)
##
##   Level 3:   7 nodes to be scored    (0 eliminated genes)
##
##   Level 2:   6 nodes to be scored    (0 eliminated genes)
##
##   Level 1:   1 nodes to be scored    (0 eliminated genes)
```

```
##
##          -- Weight Algorithm --
##
##      The algorithm is scoring 74 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test : ratio
##
##   Level 13:  1 nodes to be scored.
##
##   Level 12:  1 nodes to be scored.
##
##   Level 11:  1 nodes to be scored.
##
##   Level 10:  1 nodes to be scored.
##
##   Level 9:   3 nodes to be scored.
##
##   Level 8:   3 nodes to be scored.
##
##   Level 7:   7 nodes to be scored.
##
##   Level 6:   15 nodes to be scored.
##
##   Level 5:   14 nodes to be scored.
##
##   Level 4:   11 nodes to be scored.
##
##   Level 3:   11 nodes to be scored.
##
##   Level 2:   5 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
##
##          -- Weight Algorithm --
##
##      The algorithm is scoring 486 nontrivial nodes
##      parameters:
##          test statistic:  Fisher test : ratio
##
##   Level 19:  1 nodes to be scored.
##
##   Level 18:  2 nodes to be scored.
##
##   Level 17:  3 nodes to be scored.
##
##   Level 16:  3 nodes to be scored.
##
##   Level 15:  4 nodes to be scored.
##
##   Level 14:  8 nodes to be scored.
##
##   Level 13:  13 nodes to be scored.
##
##   Level 12:  21 nodes to be scored.
```

```
##
##   Level 11:  22 nodes to be scored.
##
##   Level 10:  22 nodes to be scored.
##
##   Level 9:   33 nodes to be scored.
##
##   Level 8:   43 nodes to be scored.
##
##   Level 7:   55 nodes to be scored.
##
##   Level 6:   75 nodes to be scored.
##
##   Level 5:   82 nodes to be scored.
##
##   Level 4:   53 nodes to be scored.
##
##   Level 3:   32 nodes to be scored.
##
##   Level 2:   13 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
##
##           -- Weight Algorithm --
##
##       The algorithm is scoring 58 nontrivial nodes
##       parameters:
##           test statistic:  Fisher test : ratio
##
##   Level 12:  1 nodes to be scored.
##
##   Level 11:  2 nodes to be scored.
##
##   Level 10:  4 nodes to be scored.
##
##   Level 9:   8 nodes to be scored.
##
##   Level 8:   5 nodes to be scored.
##
##   Level 7:   5 nodes to be scored.
##
##   Level 6:   4 nodes to be scored.
##
##   Level 5:   7 nodes to be scored.
##
##   Level 4:   8 nodes to be scored.
##
##   Level 3:   7 nodes to be scored.
##
##   Level 2:   6 nodes to be scored.
##
##   Level 1:   1 nodes to be scored.
```

A cell-means model using all conditions and batches

```
## acb stands for "kept_conditions_batches"  which takes too long to
## type when setting up the contrasts.
acb = paste0(kept_qcpml2$conditions, kept_qcpml2$batches)
kept_data = exprs(kept_qcpml2$expressionset)
table(acb)
## The invocation of table() keptows me to count up the contribution of
## each condition/batch combination to the whole data set.

## Doing this (as I understand it) means I do nothave to worry about
## balanced samples so much, but must be more careful to understand
## the relative contribution of each sample type to the entire data
## set.

complete_model = model.matrix(~0 + acb)
complete_fit = lmFit(kept_data, complete_model)
complete_voom = hpgl_voom(kept_data, complete_model)
complete_voom$plot
complete_model
## This is an example of what happens when I have heterogenous numbers of samples
## on each side of a contrast, so that a normal design matrix of conditions + batches
## would not work, so instead I add up the contributions of each batch (capital letters)
## and average them out, then use the resulting terms in the various contrasts below.
epi_cl14 = "acbcl14_epiF"
epi_clbr = "acbclbr_epiE"
tryp_cl14 = "(acbcl14_trypB + acbcl14_trypD + acbcl14_trypG) / 3"
tryp_clbr = "acbclbr_trypG"
a60_cl14 =  "(acbcl14_a60A * 2/3) + (acbcl14_a60B * 1/3)"
a60_clbr = "acbclbr_a60A"
a96_cl14 = "acbcl14_a96C"
a96_clbr = "acbclbr_a96C"
epi_cl14clbr = paste0("(",epi_cl14,")", "  -  ", "(",epi_clbr,")")
tryp_cl14clbr = paste0("(",tryp_cl14,")", "  -  ", "(",tryp_clbr,")")
a60_cl14clbr = paste0("(",a60_cl14,")", "  -  ", "(",a60_clbr,")")
a96_cl14clbr = paste0("(",a96_cl14,")", "  -  ", "(",a96_clbr,")")
epitryp_cl14 = paste0("(",tryp_cl14,")", "  -  ", "(",epi_cl14,")")
epitryp_clbr = paste0("(",tryp_clbr,")", "  -  ", "(",epi_clbr,")")
epia60_cl14 = paste0("(",a60_cl14,")", "  -  ", "(",epi_cl14,")")
epia60_clbr = paste0("(",a60_clbr,")", "  -  ", "(",epi_clbr,")")
a60a96_cl14 = paste0("(",a96_cl14,")", "  -  ", "(",a60_cl14,")")
a60a96_clbr = paste0("(",a96_clbr,")", "  -  ", "(",a60_clbr,")")
a60tryp_cl14 = paste0("(",tryp_cl14,")", "  -  ", "(",a60_cl14,")")
a60tryp_clbr = paste0("(",tryp_clbr,")", "  -  ", "(",a60_clbr,")")
## The following contrast is messed up in some as of yet unknown way.
epitryp_cl14clbr = paste0("(",epitryp_cl14,")", "  -  ", "(",epitryp_clbr,")")
## So I will add some more contrasts using data which doesn't get screwed up
epia60_cl14clbr = paste0("(",epia60_cl14,")", "  -  ", "(",epia60_clbr,")")
a60tryp_cl14clbr = paste0("(",a60tryp_cl14,")", "  -  ", "(",a60tryp_clbr,")")
a60a96_cl14clbr = paste0("(",a60a96_cl14,")", "  -  ", "(",a60a96_clbr,")")

complete_contrasts_v2 = makeContrasts(
    epi_cl14=epi_cl14,
    epi_clbr=epi_clbr,
    tryp_cl14=tryp_cl14,
    tryp_clbr=tryp_clbr,
    a60_cl14=a60_cl14,
    a60_clbr=a60_clbr,
    a96_cl14=a96_cl14,
    a96_clbr=a96_clbr,
    epi_cl14clbr=epi_cl14clbr,
    tryp_cl14clbr=tryp_cl14clbr,
    a60_cl14clbr=a60_cl14clbr,
    a96_cl14clbr=a96_cl14clbr,
    epitryp_cl14=epitryp_cl14,
    epitryp_clbr=epitryp_clbr,
    epia60_cl14=epia60_cl14,
    epia60_clbr=epia60_clbr,
    a60a96_cl14=a60a96_cl14,
    a60a96_clbr=a60a96_clbr,
    a60tryp_cl14=a60tryp_cl14,
    a60tryp_clbr=a60tryp_clbr,
    epitryp_cl14clbr=epitryp_cl14clbr,
    epia60_cl14clbr=epia60_cl14clbr,
    a60tryp_cl14clbr=a60tryp_cl14clbr,
    a60a96_cl14clbr=a60a96_cl14clbr,
    levels=complete_voom$design)
## This colnames() is annoyingly necessary to avoid really obnoxious contrast names.
colnames(complete_contrasts_v2) = c("epi_cl14","epi_clbr","tryp_cl14","tryp_clbr","a60_cl14","a60_clbr"
kept_fits = contrasts.fit(complete_fit, complete_contrasts_v2)
kept_comparisons = eBayes(kept_fits)
```

Clean conditions, batches

On the other hand, I would like to perform arbitrary comparisons among my data even when the batches and conditions look good, so I set up my model/contrast matrices a little strangely even then:

```
all_data = exprs(norm_expt$expressionset)
complete_model = model.matrix(~0 + all_human_expt$conditions + all_human_expt$batches)
## Shorten the column names of the model so I don't have to type so much later...
tmpnames = colnames(complete_model)
tmpnames = gsub("all_human_expt[[:punct:]]","", tmpnames)
tmpnames = gsub("conditions","", tmpnames)
colnames(complete_model) = tmpnames
rm(tmpnames)

complete_voom = hpgl_voom(all_data, complete_model)
complete_voom$plot
complete_fit = lmFit(complete_voom, complete_model)

all_contrasts = makeContrasts(
    ## Start with the simple coefficient groupings for each condition
    none4=none4,
    none24=none24,
    none48=none48,
    none72=none72,
```

```r
    bead4=bead4,
    bead24=bead24,
    bead48=bead48,
    bead72=bead72,
    maj4=maj4,
    maj24=maj24,
    maj48=maj48,
    maj72=maj72,
    ama4=ama4,
    ama24=ama24,
    ama48=ama48,
    ama72=ama72,
    ## Now do a few simple comparisons
    ## compare beads to uninfected
    beadnone_4=bead4-none4,
    beadnone_24=bead24-none24,
    beadnone_48=bead48-none48,
    beadnone_72=bead72-none72,
    majnone_4=maj4-none4,
    majnone_24=maj24-none24,
    majnone_48=maj48-none48,
    majnone_72=maj72-none72,
    amanone_4=ama4-none4,
    amanone_24=ama24-none24,
    amanone_48=ama48-none48,
    amanone_72=ama72-none72,
    ## compare samples to beads
    majbead_4=maj4-bead4,
    majbead_24=maj24-bead24,
    majbead_48=maj48-bead48,
    majbead_72=maj72-bead72,
    amabead_4=ama4-bead4,
    amabead_24=ama24-bead24,
    amabead_48=ama48-bead48,
    amabead_72=ama72-bead72,
    ## (x-z)-(a-b)
    ## Use this to compare major and amazonensis
    amamaj_bead_4=(ama4-bead4)-(maj4-bead4),
    amamaj_bead_24=(ama24-bead24)-(maj24-bead24),
    amamaj_bead_48=(ama48-bead48)-(maj48-bead48),
    amamaj_bead_72=(ama72-bead72)-(maj72-bead72),
    ## (c-d)-(e-f) where c/d are: (amazon|major/none)/(beads/none)
    majbead_none_4=(maj4-none4)-(bead4-none4),
    majbead_none_24=(maj24-none24)-(bead24-none24),
    majbead_none_48=(maj48-none48)-(bead48-none48),
    majbead_none_72=(maj72-none72)-(bead72-none72),
    amabead_none_4=(ama4-none4)-(bead4-none4),
    amabead_none_24=(ama24-none24)-(bead24-none24),
    amabead_none_48=(ama48-none48)-(bead48-none48),
    amabead_none_72=(ama72-none72)-(bead72-none72),
    levels=complete_voom$design)
all_fits = contrasts.fit(complete_fit, all_contrasts)
all_comparisons = eBayes(all_fits)
```

```
limma_list = write_limma(data=all_comparisons)

all_table = topTable(all_comparisons, adjust="fdr", n=nrow(all_data))
write.csv(all_comparisons, file="excel/all_tables.csv")
## write_limma() is a shortcut for writing out all the data structures
all_comparison_tables = write_limma(all_comparisons, excel=FALSE)
```

Ontology searches

The following is an example of a simplified GO search given 20 groups of genes which are from an unannotated organism, but for which blast2GO was performed.

```
ontology_info = read.csv(file="data/trinotate_go_trimmed.csv.gz", header=FALSE, sep="\t")
##ontology_info = read.csv(file="data/transcript_go.csv.gz", header=FALSE, sep="\t")
colnames(ontology_info) = c("gene_id","transcript_id","group","startend","blast_go","pfam_go")
## Drop any entries which don't have a putative length
ontology_info = subset(ontology_info, startend != 0)
## Split the column 'startend' into two columns by the '-' sign
ontology_info = as.data.frame(transform(ontology_info, startend=reshape::colsplit(startend, split="\\-"
## Make the resulting pieces into two separate columns, start and end.
ontology_info$start = ontology_info$startend$start
ontology_info$end = ontology_info$startend$end
## Use start and end to make length
ontology_info$length = abs(ontology_info$start - ontology_info$end)
## Drop the unneeded columns
ontology_info = ontology_info[,c("gene_id","transcript_id","group","start","end","length","blast_go","p
head(ontology_info)
```

```
##    gene_id transcript_id group start  end length
## 1  c111_g1    c111_g1_i1     1   833 2068   1235
## 2  c753_g1    c753_g1_i1     1    50 1660   1610
## 3  c777_g1    c777_g1_i1     1  1433 3670   2237
## 4  c777_g1    c777_g1_i2     1  1553 3790   2237
## 5 c1076_g1   c1076_g1_i1     1   133 2601   2468
## 6 c2482_g1   c2482_g1_i2     1  1112 1612    500
##
## 1
## 2 GO:0031225 GO:0046658 GO:0048046 GO:0005618 GO:0016020 GO:0009505 GO:0005886 GO:0009506 GO:0005774
## 3                                                                                 GO:0016021 GO:0005886
## 4                                                                                 GO:0016021 GO:0005886
## 5                                                                                 GO:0030176 GO:0008219
## 6
##      pfam_go
## 1          0
## 2
## 3
## 4
## 5          0
## 6 GO:0005515
## goseq() requires mappings between ID/length and ID/GO category
## Currently I have my toy set to assume column names, which is admittedly stupid.
gene_lengths = ontology_info[,c("transcript_id","length")]
colnames(gene_lengths) = c("ID","width")
split_go = ontology_info[,c("transcript_id","blast_go")]
split_go$blast_go = as.character(split_go$blast_go)

## The following few lines were pulled from the internet
## they serve to generate a data structure in the format expected by goseq()
## It simply splits all space separated GO categories into separate rows
## with the same ID
require.auto("splitstackshape")
id_go = concat.split.multiple(split_go, "blast_go", seps=" ", "long")
```

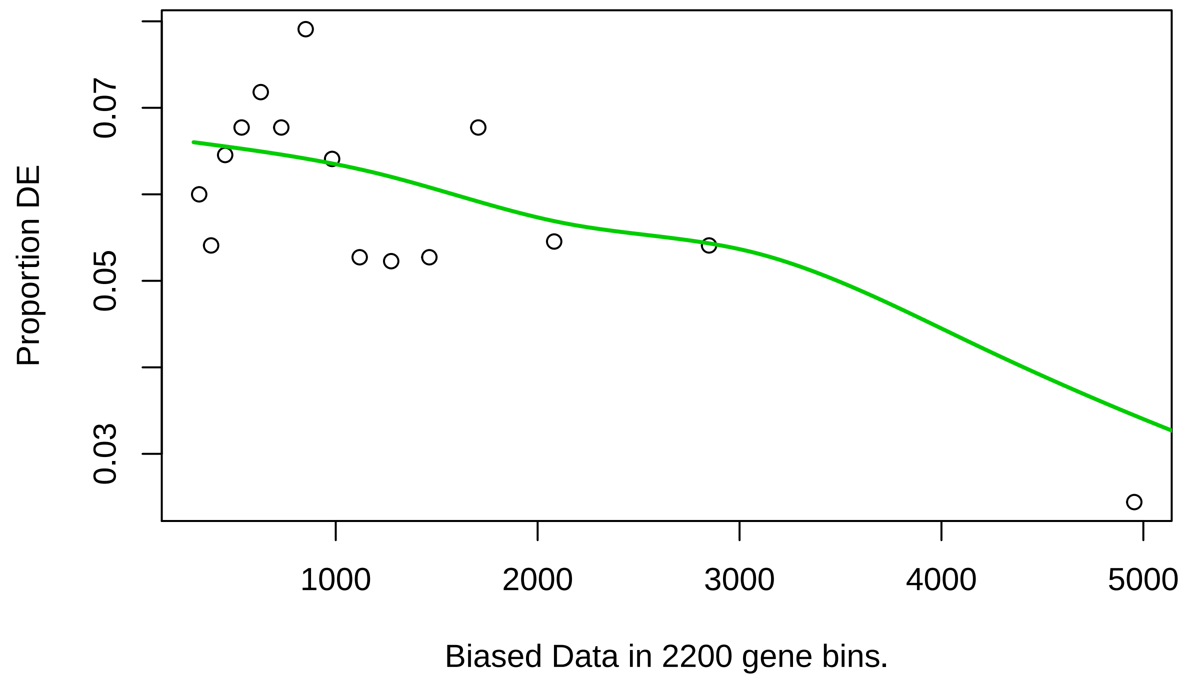
```
## This function is deprecated. Use `cSplit` instead.
```

```
id_go = as.data.frame(id_go)
colnames(id_go) = c("ID","GO")
go_ids = subset(id_go, GO != 0)

## Pull out all entries from group 1
group_one = subset(ontology_info, group == "1")
group_one = group_one[,c("transcript_id","start","end")]
colnames(group_one) = c("ID","start","end")

## Perform the goseq() analysis
group_one_go = simple_goseq(group_one, lengths=gene_lengths, goids=go_ids)
```

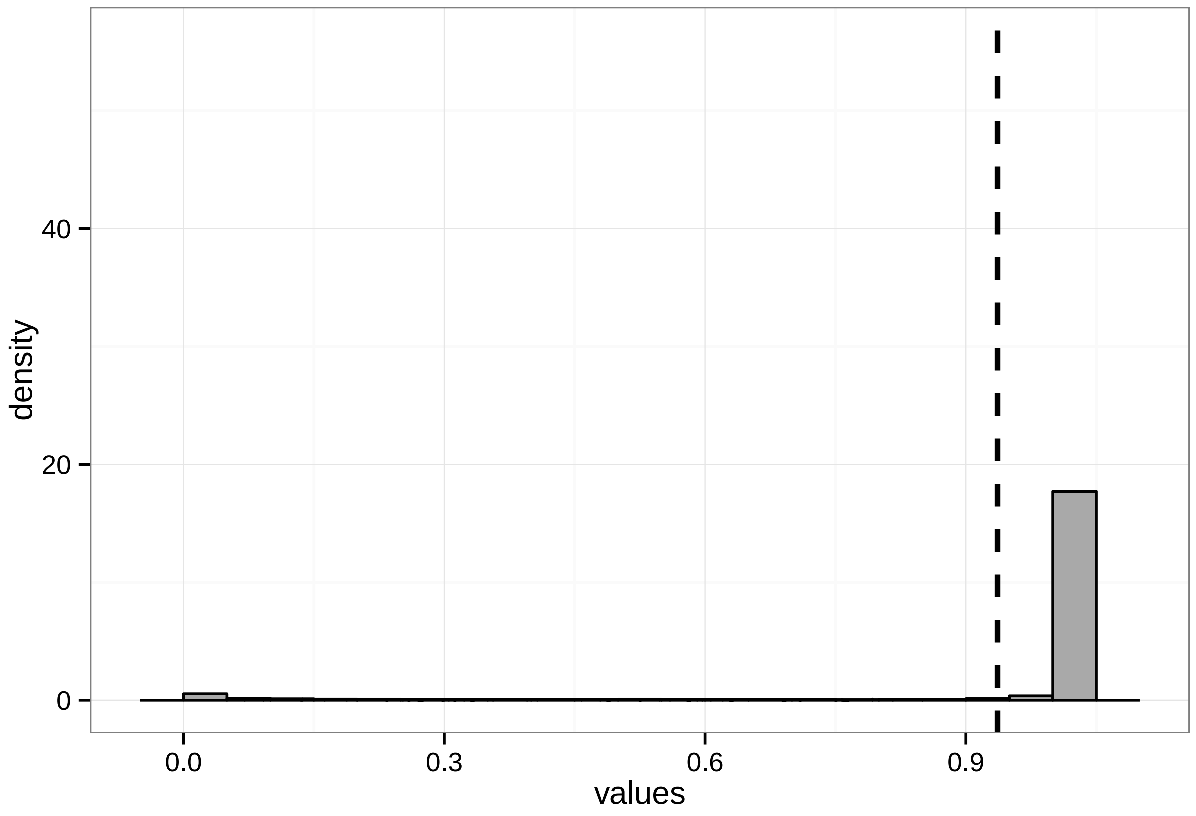
```
## simple_goseq() makes some pretty hard assumptions about the data it is fed:
## It requires 2 tables, one of GOids which must have columns (gene)ID and GO(category)
## The other table is of gene lengths with columns (gene)ID and (gene)width.
## Other columns are fine, but ignored.
## Using the length data to fill in the de vector.
## Using manually entered categories.
## Calculating the p-values...
## Calculating q-values
## Filling godata table with term information, this takes a while.
```

```
## [1] "Testing that go categories are defined."
## [1] "Removing undefined categories."
## [1] "Gathering synonyms."
## [1] "Gathering secondary ids."
## [1] "Gathering category definitions."
```

```
## Making pvalue plots for the ontologies.
```



```
group_one_go$pvalue_histogram
```



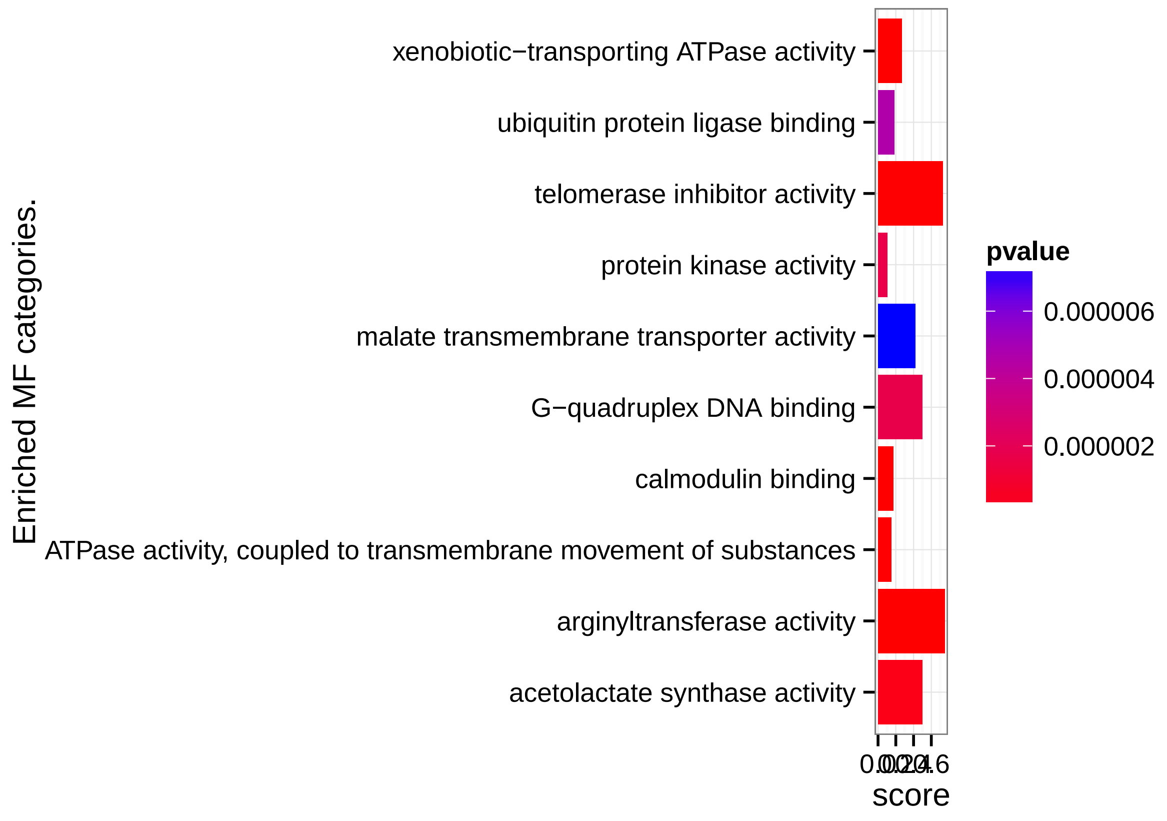
```
head(group_one_go$godata_interesting)
```

```
##         category numDEInCat numInCat over_represented_pvalue
## 2415 GO:0008911         10       10   0.0000000000004151763
## 714  GO:0004057         12       16   0.0000000000024209997
## 3664 GO:0016598         12       16   0.0000000000024209997
## 6529 GO:0050994         12       16   0.0000000000024209997
## 4931 GO:0034077          8        8   0.0000000001485178138
## 5763 GO:0045151          8        8   0.0000000001485178138
##      under_represented_pvalue           qvalue ontology
## 2415                        1 0.000000003264946       MF
## 714                         1 0.000000004759685       MF
## 3664                        1 0.000000004759685       BP
## 6529                        1 0.000000004759685       BP
## 4931                        1 0.000000194657348       BP
## 5763                        1 0.000000194657348       BP
##                                    term
## 2415   lactaldehyde dehydrogenase activity
## 714            arginyltransferase activity
## 3664                  protein arginylation
## 6529 regulation of lipid catabolic process
## 4931          butanediol metabolic process
## 5763          acetoin biosynthetic process
##
## 2415
## 714  arginine transferase activity, arginyl-transfer ribonucleate-protein aminoacyltransferase activi
## 3664
## 6529
## 4931
## 5763
##       secondary
## 2415
## 714  GO:0042172
## 3664 GO:0019130
## 6529
## 4931
## 5763
##
## 2415                                                                      Catalysis of the reaction
## 714                                                                           Catalysis of the reac
## 3664 The conjugation of arginine to the N-terminal aspartate or glutamate of a protein; required for
## 6529                   Any process that modulates the frequency, rate, or extent of the chemical re
## 4931                        The chemical reactions and pathways involving butanediol; the biologica
## 5763                                                          The chemical reactions and pathways re
```

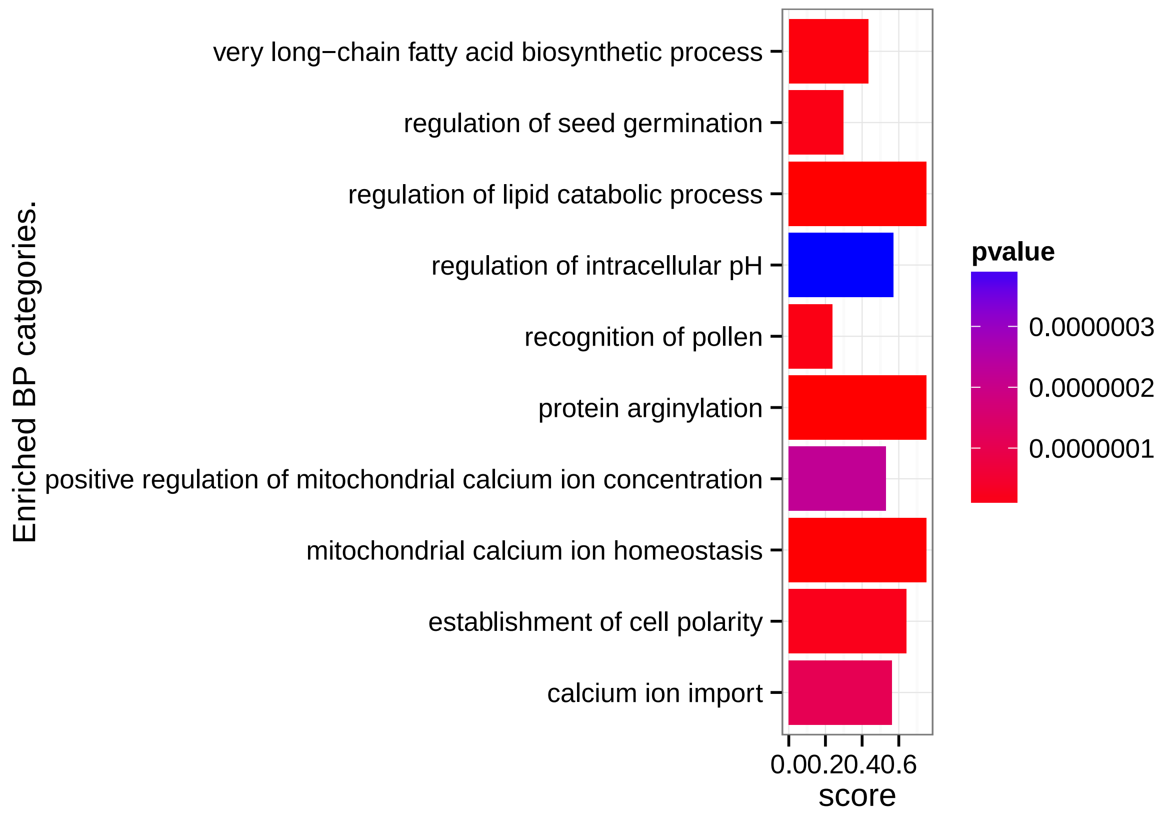
```r
head(group_one_go$mf_subset)
```

```
##        category over_represented_pvalue under_represented_pvalue
## 2415 GO:0008911   0.0000000000004151763                        1
## 714  GO:0004057   0.0000000000024209997                        1
## 2303 GO:0008559   0.0000000002259948329                        1
## 1243 GO:0005516   0.0000000013380245983                        1
## 5382 GO:0042626   0.0000000246020005545                        1
## 3138 GO:0010521   0.0000000357102309135                        1
##      numDEInCat numInCat
## 2415         10       10
## 714          12       16
## 2303          23       86
## 1243          40      232
## 5382          42      281
## 3138           8       11
##                                                                 term
## 2415                              lactaldehyde dehydrogenase activity
## 714                                      arginyltransferase activity
## 2303                            xenobiotic-transporting ATPase activity
## 1243                                              calmodulin binding
## 5382 ATPase activity, coupled to transmembrane movement of substances
## 3138                                   telomerase inhibitor activity
##      ontology            qvalue
## 2415       MF 0.000000003264946
## 714        MF 0.000000004759685
## 2303       MF 0.000000253889052
## 1243       MF 0.000001169136160
## 5382       MF 0.000011380596021
## 3138       MF 0.000015601403106
```

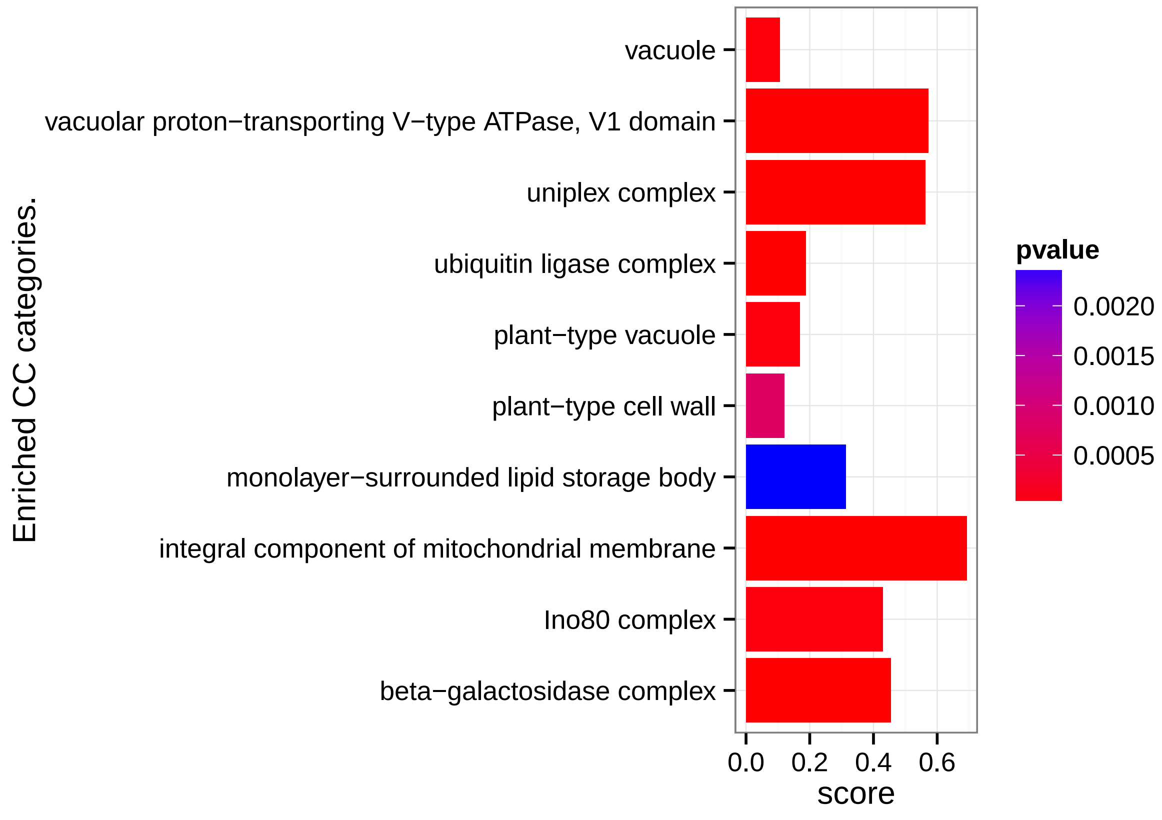
```
group_one_go$mfp_plot
```



```
group_one_go$bpp_plot
```



```
group_one_go$ccp_plot
```



```
## Print trees of the goseq() data
initial_trees = goseq_trees(group_one, group_one_go, goids_df=go_ids)
```

```
## Error in .local(.Object, ...): allGenes must be a factor with 2 levels
```

```
initial_trees$MF
```

```
## Error in eval(expr, envir, enclos): object 'initial_trees' not found
```

```
initial_trees$BP
```

```
## Error in eval(expr, envir, enclos): object 'initial_trees' not found
```

```
initial_trees$CC
```

```
## Error in eval(expr, envir, enclos): object 'initial_trees' not found
```

Vignette Info

Note the various macros within the `vignette` setion of the metadata block above. These are required in order to instruct R how to build the vignette. Note that you should change the `title` field and the `\VignetteIndexEntry` to match the title of your vignette.

Styles

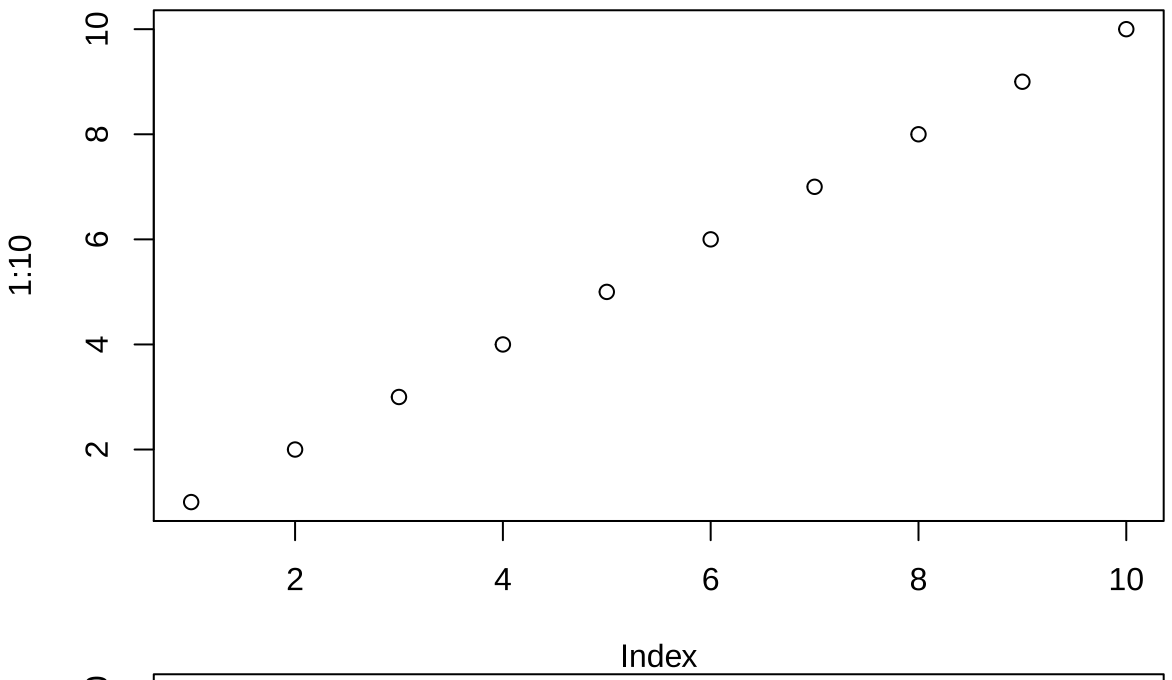
The `html_vignette` template includes a basic CSS theme. To override this theme you can specify your own CSS in the document metadata as follows:

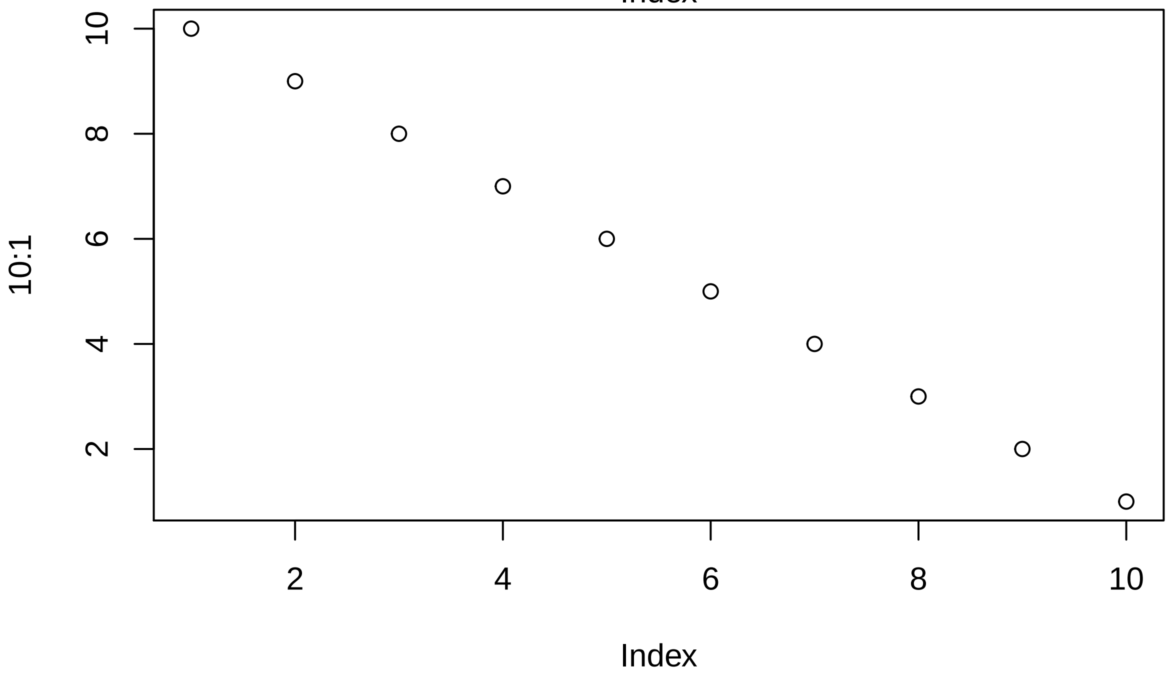
```
output:
  rmarkdown::html_vignette:
    css: mystyles.css
```

Figures

The figure sizes have been customised so that you can easily put two images side-by-side.

```
plot(1:10)
plot(10:1)
```





You can enable figure captions by `fig_caption: yes` in YAML:

```
output:
  rmarkdown::html_vignette:
    fig_caption: yes
```

Then you can use the chunk option `fig.cap = "Your figure caption."` in **knitr**.

More Examples

You can write math expressions, e.g. $Y = X\beta + \epsilon$, footnotes[1], and tables, e.g. using `knitr::kable()`.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

Also a quote using `>`:

> "He who gives up [code] safety for [code] speed deserves neither." (via)

[1] A footnote here.