hpgltools examples using the fission dataset

atb abelew@gmail.com

2016-03-13

Example hpgltool usage with a real data set (fission)

This document aims to provide further examples in how to use the hpgltools.

Note to self, the header has rmarkdown::pdf_document instead of html_document or html_vignette because it gets some bullcrap error 'margins too large'...

Setting up

Here are the commands I invoke to get ready to play with new data, including everything required to install hpgltools, the software it uses, and the fission data.

```
## These first 4 lines are not needed once hpgltools is installed.
## source("http://bioconductor.org/biocLite.R")
## biocLite("devtools")
## library(devtools)
## install_github("elsayed-lab/hpgltools")
library(hpgltools)
require.auto("fission")
library(fission)
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: methods
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following object is masked from 'package:stats':
##
##     xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, as.vector, cbind,
##     colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
##     intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unlist, unsplit

## Loading required package: S4Vectors

## Loading required package: stats4

## Creating a generic function for 'nchar' from package 'base' in package 'S4Vectors'

## Loading required package: IRanges

## Loading required package: GenomeInfoDb
```

```
data(fission)
knitr::opts_knit$set(progress=TRUE, verbose=TRUE, error=TRUE,  fig.width=7, fig.height=7)
```

Data import

All the work I do in Dr. El-Sayed's lab makes some pretty hard assumptions about how data is stored. As a result, to use the fission data set I will do a little bit of shenanigans to match it to the expected format. Now that I have played a little with fission, I think its format is quite nice and am likely to have my experiment class instead be a SummarizedExperiment.

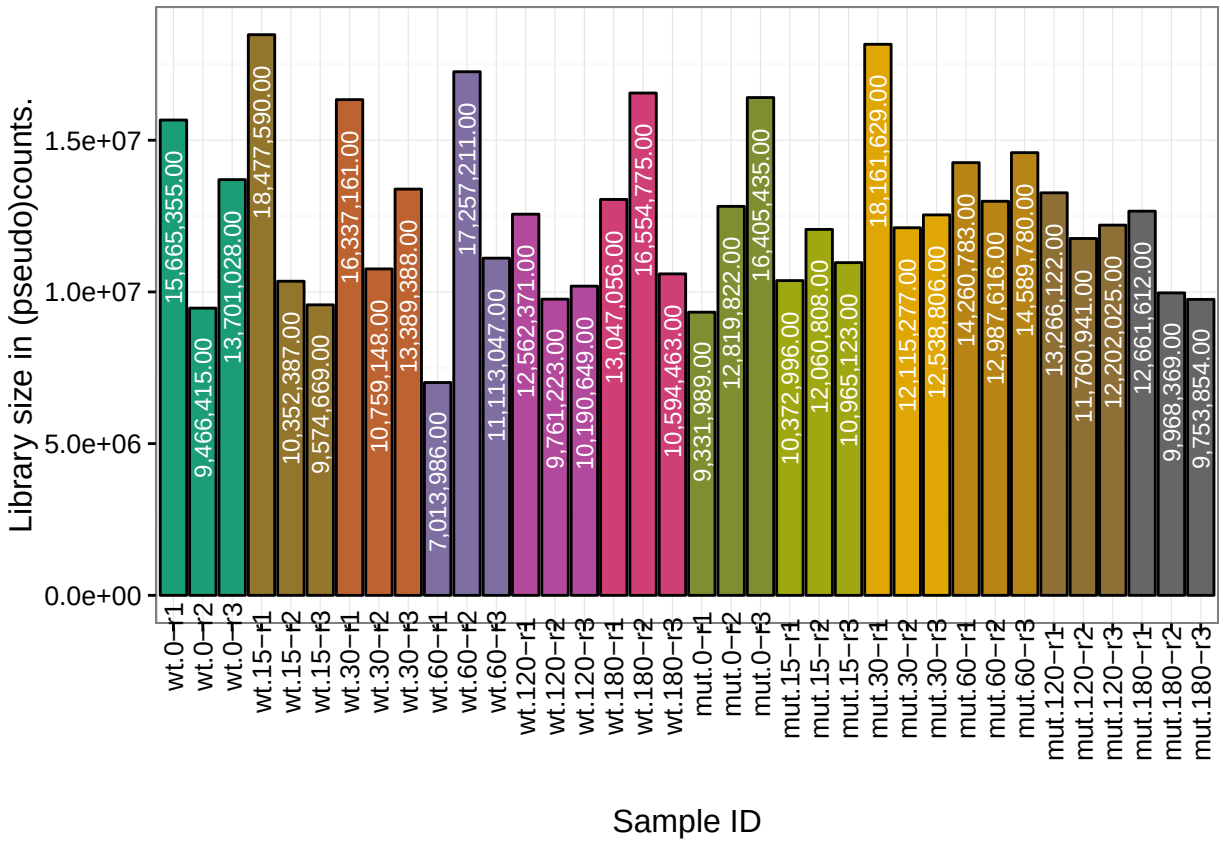
```
## Extract the meta data from the fission dataset
meta <- as.data.frame(fission@colData)
## Make conditions and batches
meta$condition <- paste(meta$strain, meta$minute, sep=".")
meta$batch <- meta$replicate
meta$sample.id <- rownames(meta)
## Grab the count data
fission_data <- fission@assays$data$counts
## This will make an experiment superclass called 'expt' and it contains
## an ExpressionSet along with any arbitrary additional information one might want to include.
## Along the way it writes a Rdata file which is by default called 'expt.Rdata'
fission_expt <- create_expt(meta_dataframe=meta, count_dataframe=fission_data)
```

```
## create_expt(): This needs columns with conditions and batches in the sample sheet.

## create_experiment():  This function assumes some columns in the sample sheet:

## Sample.ID, Stage, Type, condition, batch

## Loading required namespace: Biobase
```

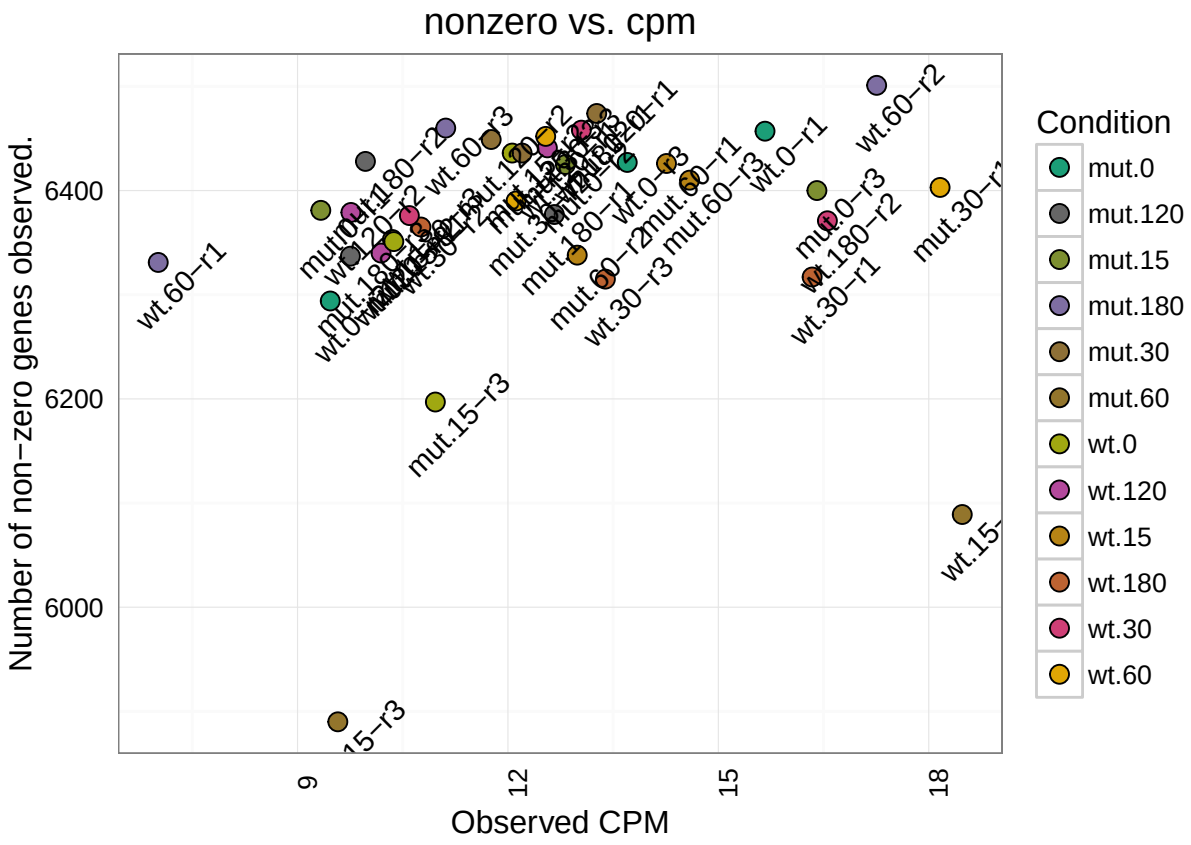
Normalizing and exploring data

There are lots of toys we have learned to use to play with with raw data and explore stuff like batch effects or non-canonical distributions or skewed counts. hpgltools provides some functionality to make this process easier. The graphs shown below and many more are generated with the wrapper 'graph_metrics()' but that takes away the chance to explain the graphs as I generate them.

```
## First make a bar plot of the library sizes in the experiment.
## Notice that the colors were auto-chosen by create_expt() and they should
## be maintained throughout this process
fis_libsize <- hpgl_libsize(fission_expt)
fis_libsize
```



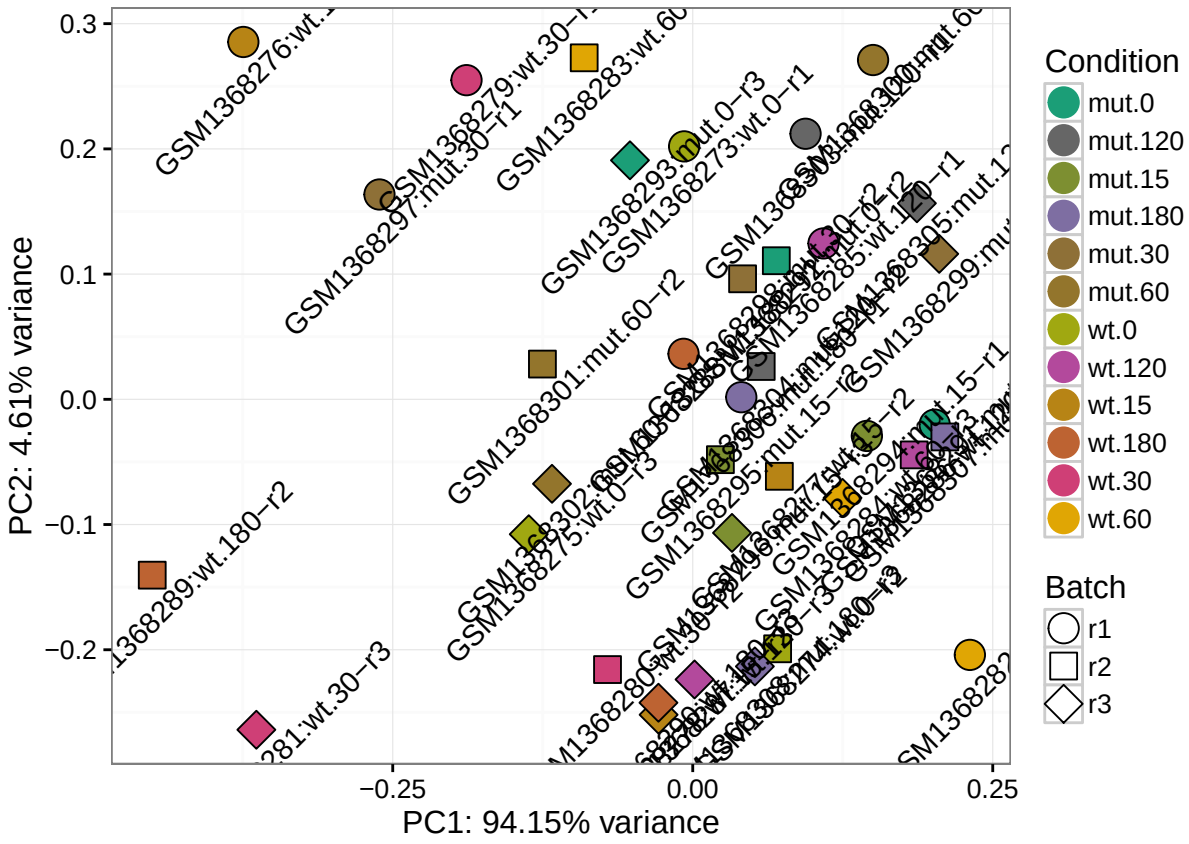

```
## Here we see that the wild type replicate 3 sample for 15 minutes has fewer non-zero genes than all i
fis_nonzero <- hpgl_nonzero(fission_expt, labels="boring", title="nonzero vs. cpm")
fis_nonzero
```



An initial pca plot

In most cases, raw data does not cluster very well, lets see if that is also true for the fission experiment. Assuming it doesn't, lets normalize the data using the defaults (cpm, quantile, log2) and try again.

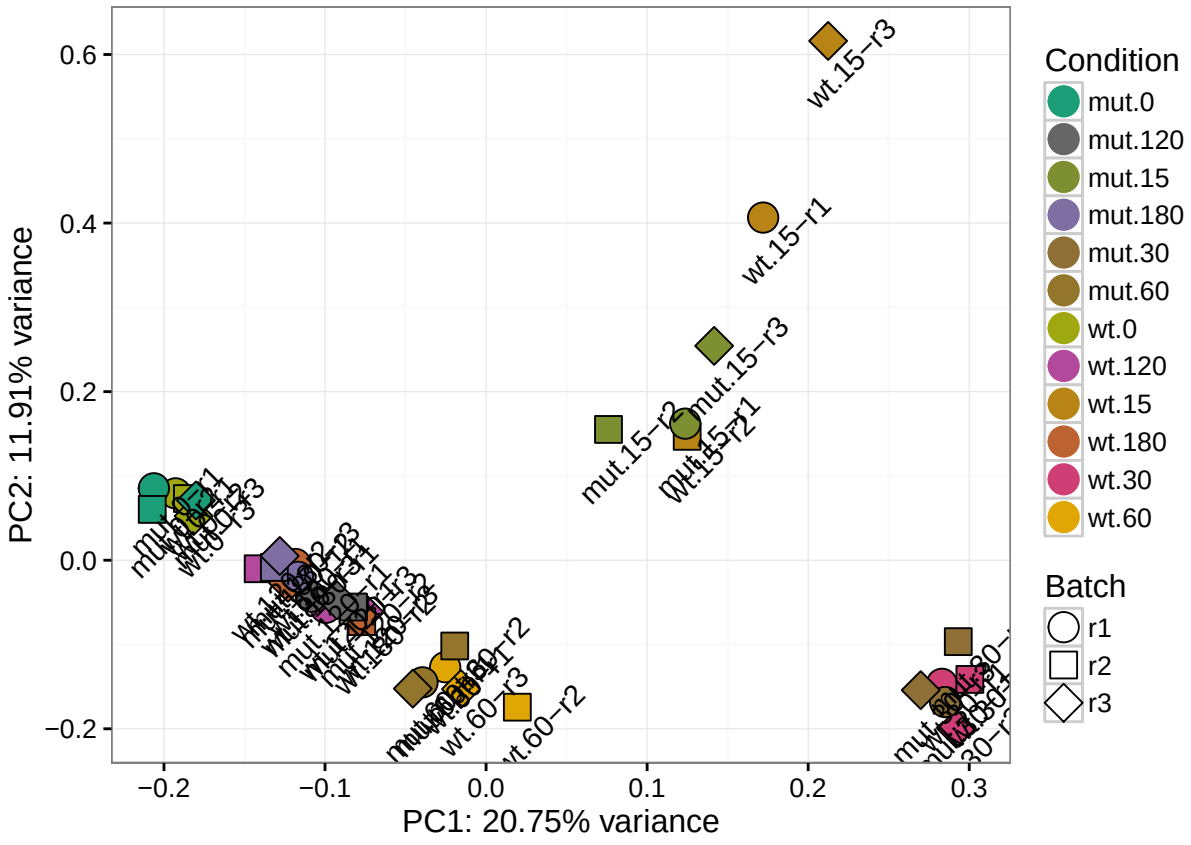
```
## Something in this is causing a build loop on travis...

## I am no longer certain that code is maintained, what remains from it I might pull in to my own.
## require.auto("kokrah/cbcbSEQ")  ## Install Kwame's cbcbSEQ
## Unsurprisingly, the raw data doesn't cluster well at all...
fis_rawpca <- hpgl_pca(fission_expt, expt_labels=fission_expt$condition)
fis_rawpca$plot
```




```
## So, normalize the data
norm_expt <- normalize_expt(fission_expt, transform="log2", norm="quant", convert="cpm")
```

```
## This function will replace the expt$expressionset slot with:

## log2(quant(cpm(data)))

## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize

## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)

## Not correcting the count-data for batch effects.  If batch is
##  included in EdgerR/limma's model, then this is probably wise; but in extreme
##  batch effects this is a good parameter to play with.
```

```
## And try the pca again
fis_normpca <- hpgl_pca(norm_expt, plot_labels="boring", title="normalized pca")
fis_normpca$plot
```

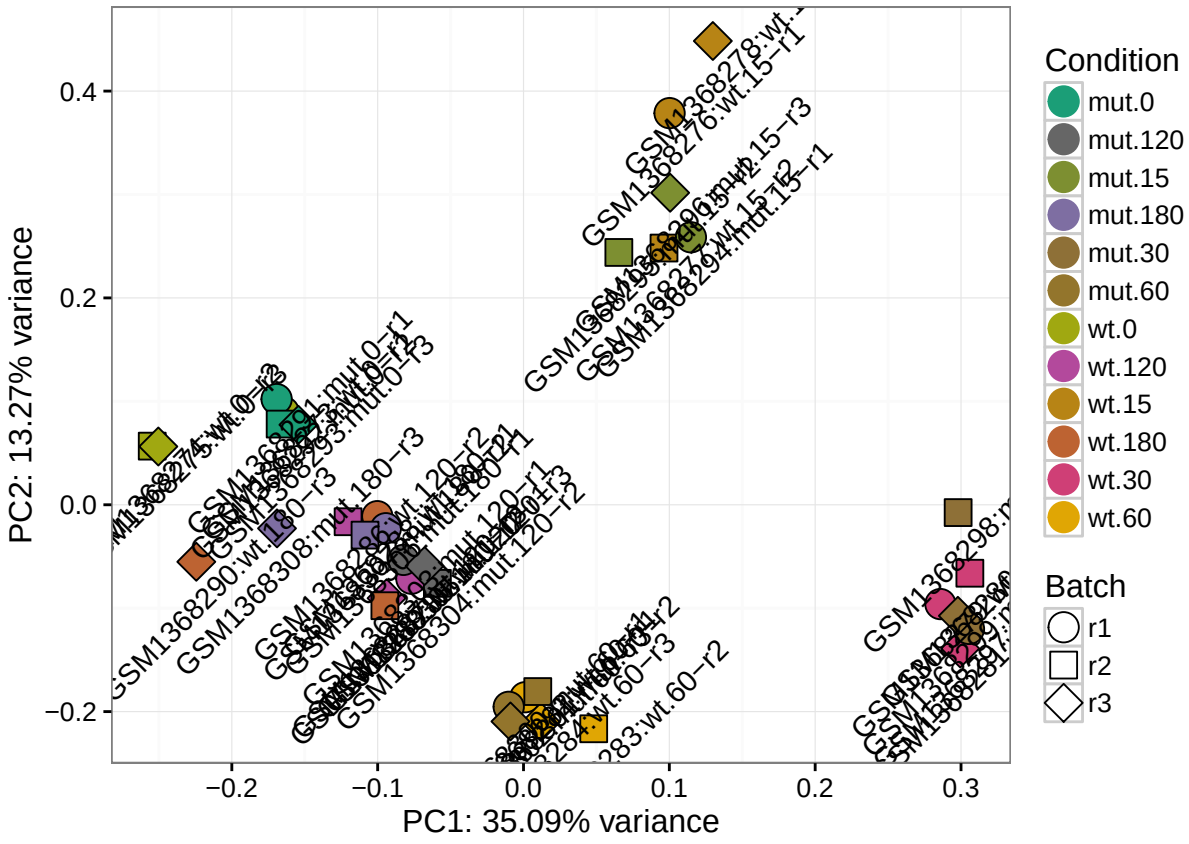



```
normbatch_expt <- normalize_expt(fission_expt, transform="log2", norm="quant", convert="cpm", batch="sva
```

```
## This function will replace the expt$expressionset slot with:

## log2(quant(cpm(batch-correct(data))))

## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize

## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)

## batch_counts: Before batch correction, 47195 entries 0<x<1.
```

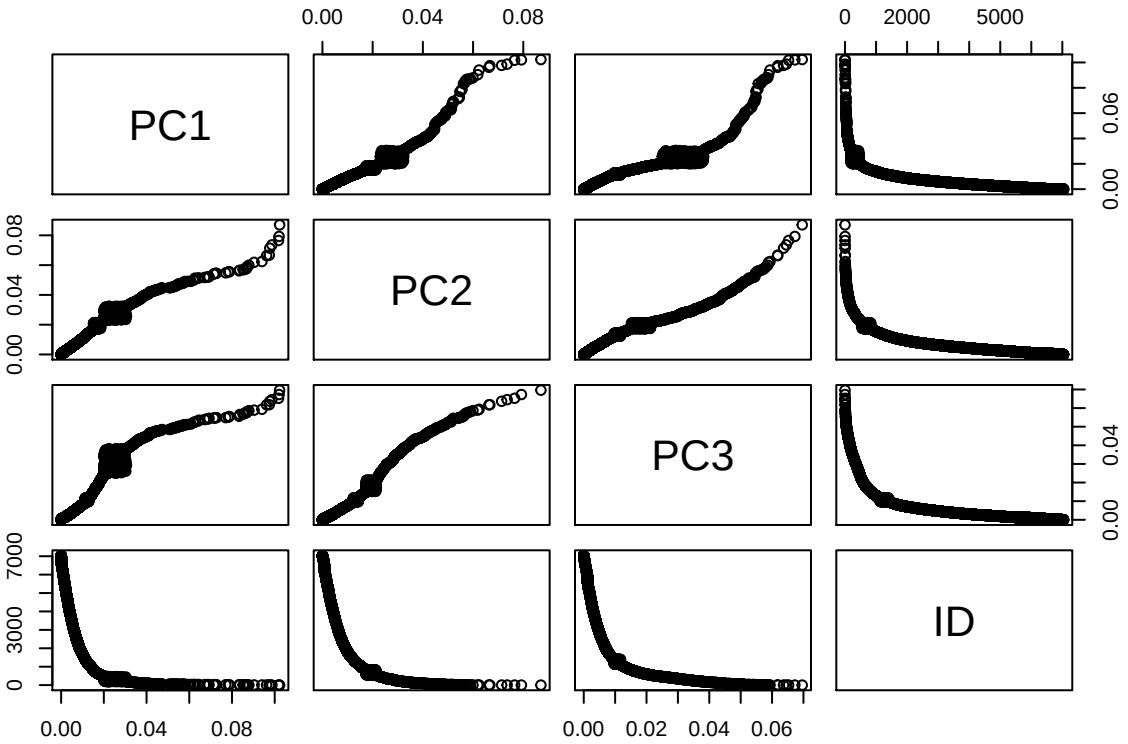
```
## batch_counts: Using sva::fsva for batch correction.

## Number of significant surrogate variables is:  1
## Iteration (out of 5 ):1  2  3  4  5

## The number of elements which are < 0 after batch correction is: 1383

## transform_counts: Found 1383 values equal to 0, adding 0.5
##  to the matrix.
```

```
fis_normbatchpca <- hpgl_pca(normbatch_expt, title="Normalized PCA with batch effect correction.")
fis_normbatchpca$plot
```

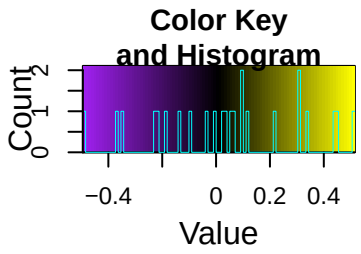


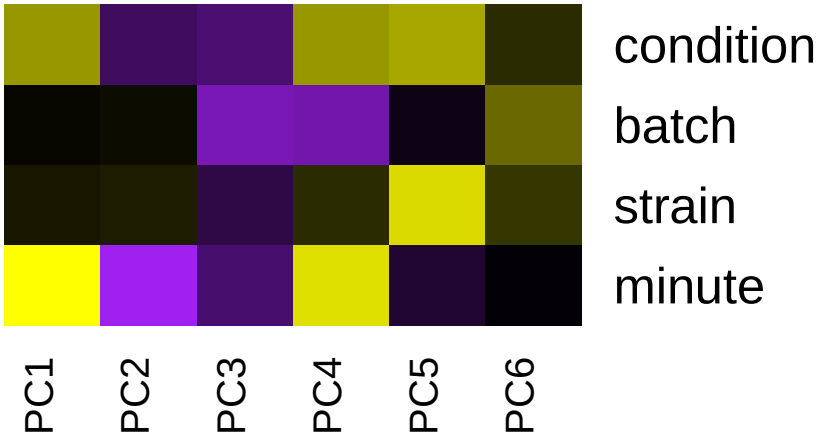

```
## ok, that caused the 0, 60, 15, and 30 minute samples to cluster nicely
## the 120 and 180 minute samples are still a bit tight

## pca_information provides some more information about the call to
## fast.svd that went into making the pca plot
fis_info <- pca_information(norm_expt, expt_factors=c("condition","batch","strain","minute"), num_compo
```

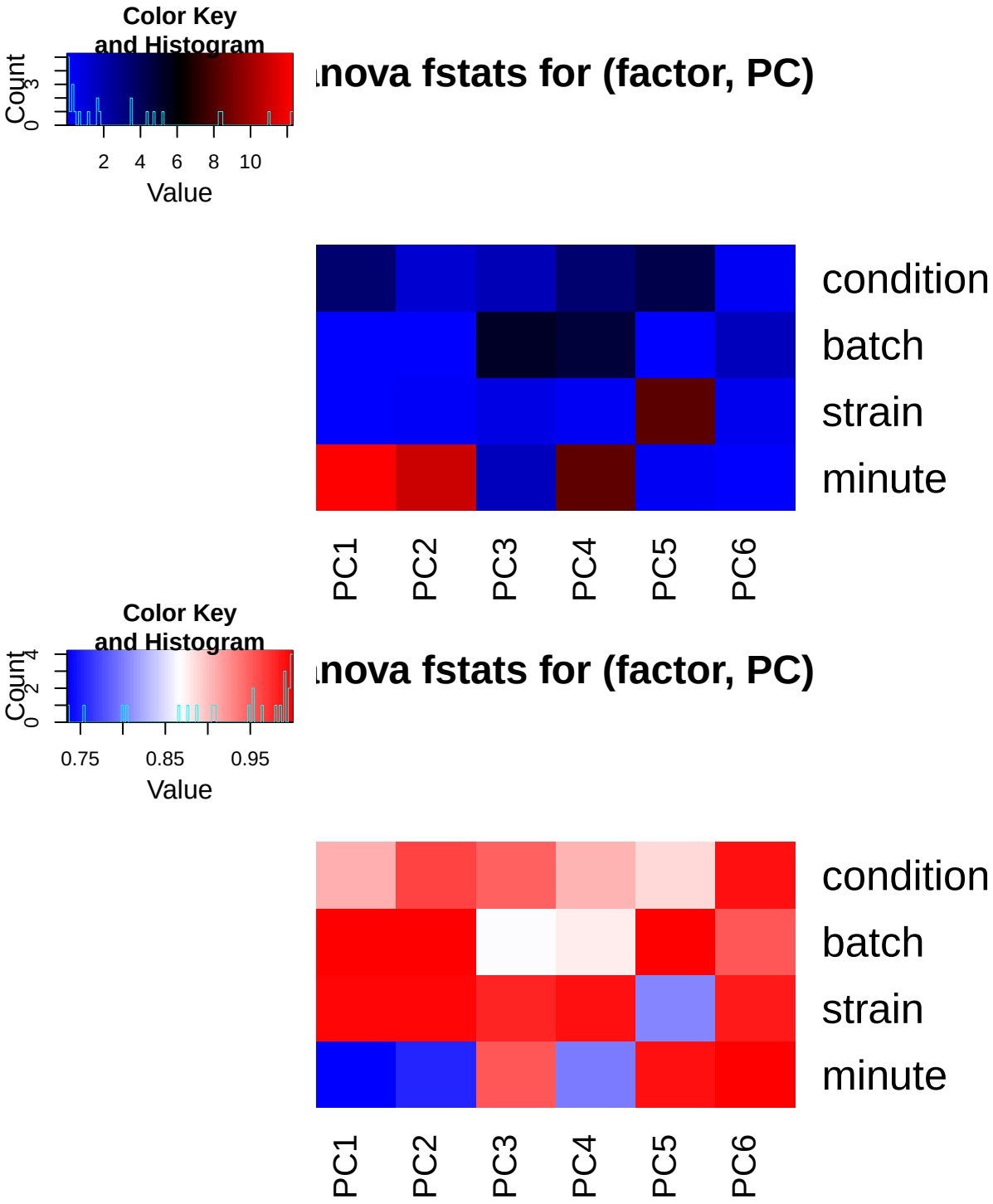
```
## The more shallow the curves in these plots, the more genes responsible for this principle component.
```

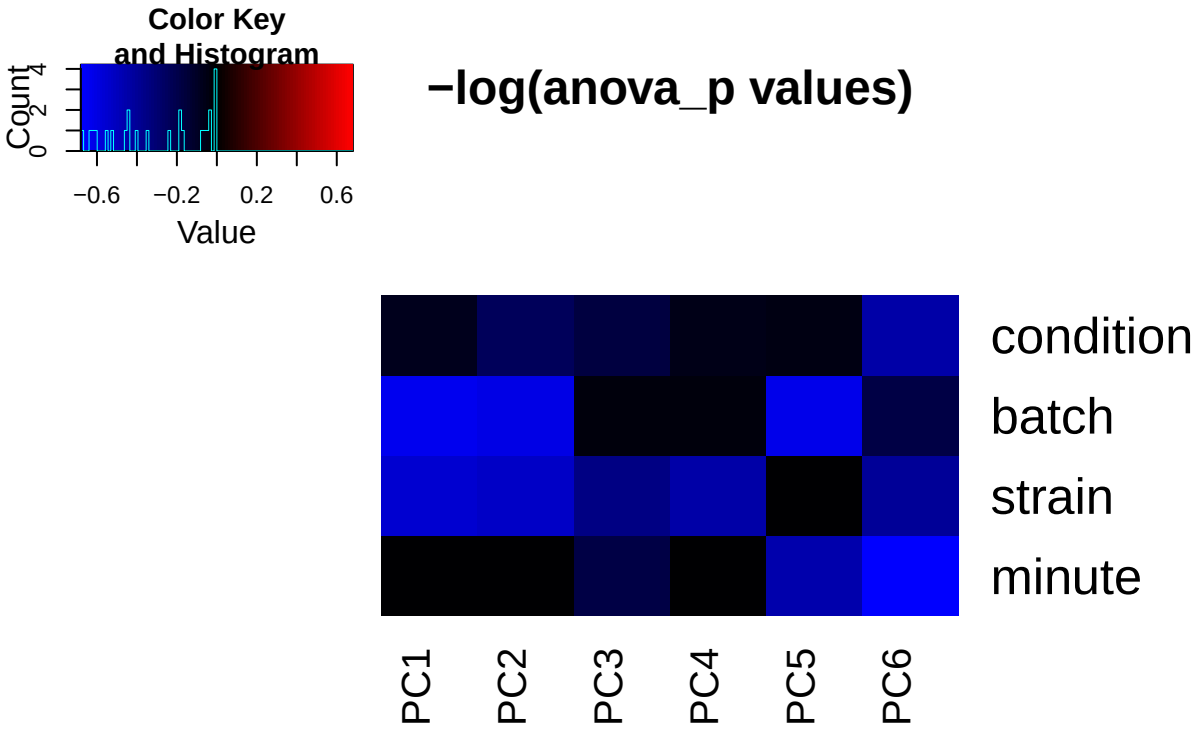



```
## [1] "PC1: 20.75% variance"
## [1] "PC2: 11.91% variance"
```






Color Key
and Histogram
Count
0
3
2
4
6
8
10
Value
nova fstats for (factor, PC)
condition
batch
strain
minute
PC1
PC2
PC3
PC4
PC5
PC6
Color Key
and Histogram
Count
0
2
4
0.75
0.85
0.95
Value
nova fstats for (factor, PC)
condition
batch
strain
minute
PC1
PC2
PC3
PC4
PC5
PC6




```
## The r^2 table shows that quite a lot of the variance in the data is explained by condition
head(fis_info$rsquared_table)
```

```
##   prop_var cumulative_prop_var condition  batch strain minute
## 1   20.752              20.752    98.743  0.069  0.315 98.053
## 2   11.911              32.663    87.067  0.772  0.443 80.859
## 3    7.702              40.365    15.586 13.626  1.889 11.256
## 4    6.138              46.503    76.204 12.331  0.997 65.848
## 5    4.863              51.366    70.220  0.917 19.633 37.284
## 6    3.891              55.257    74.218  4.921  1.369 67.245
```

```
## We can look at the correlation between the principle components and the factors in the experiment
## in this case looking at condition/batch vs the first 4 components.
fis_info$pca_cor
```

```
##                  PC1         PC2        PC3         PC4         PC5
## condition 0.30380317 -0.18690226 -0.2253680  0.30765103  0.33650975
## batch     0.02397345  0.03691367 -0.3645445 -0.35037843 -0.03137802
## strain    0.05616874  0.06653382 -0.1374555  0.09987154  0.44308965
## minute    0.51541477 -0.49466299 -0.2140827  0.44642328 -0.09814799
##                  PC6
## condition  0.099040592
## batch      0.215088641
## strain     0.117025631
## minute    -0.005377346
```

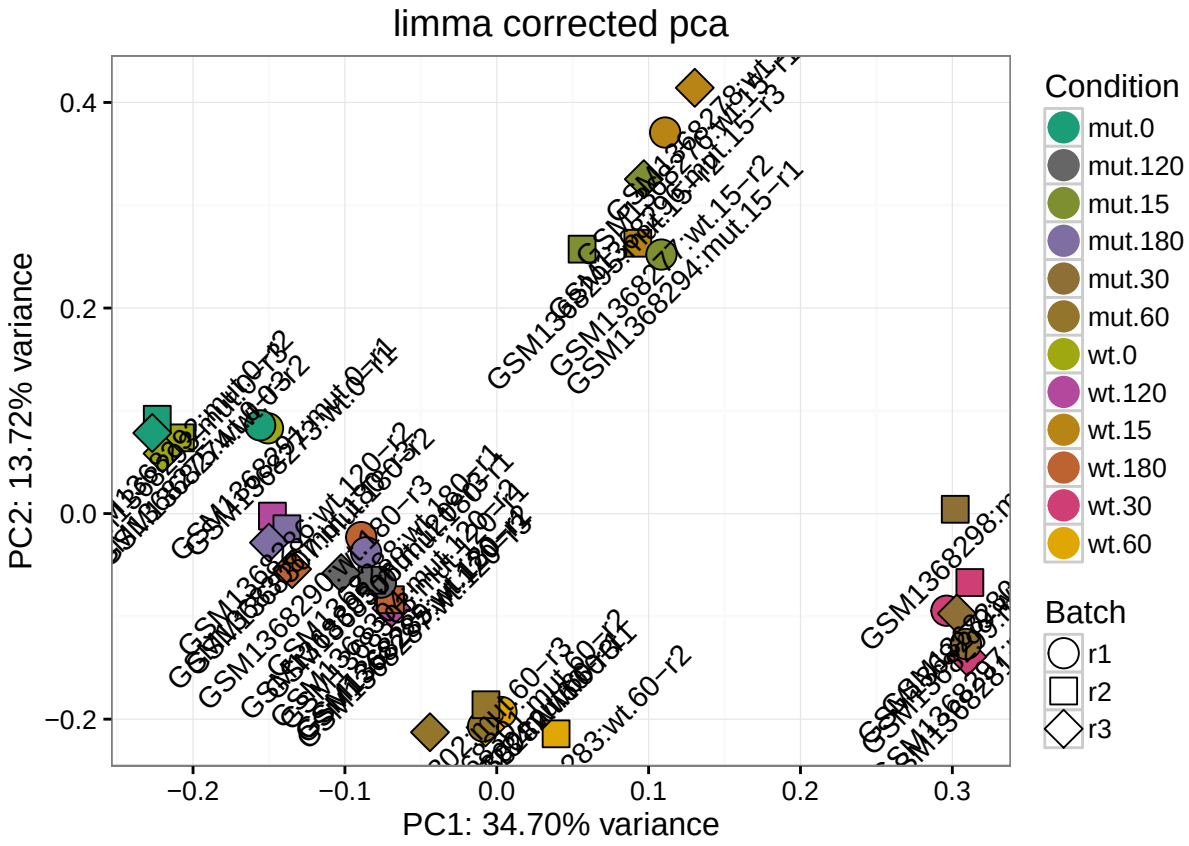
```
## And p-values to lend some credence(or not to those assertions)
fis_info$anova_p
```

```
##                   PC1         PC2        PC3         PC4         PC5
## condition 0.071650310 0.275057211 0.18631370 0.067953327 0.044775771
## batch     0.889620757 0.830751187 0.02882176 0.036170535 0.855842993
## strain    0.744896750 0.699835860 0.42403995 0.562229292 0.006801267
## minute    0.001295445 0.002163481 0.20992903 0.006348466 0.569028441
##                 PC6
## condition 0.5655026
## batch     0.2077438
## strain    0.4966850
## minute    0.9751694
```

```
## Try again with batch removed data
batchnorm_expt <- normalize_expt(fission_expt, batch="limma", norm="quant", transform="log2", convert="
```

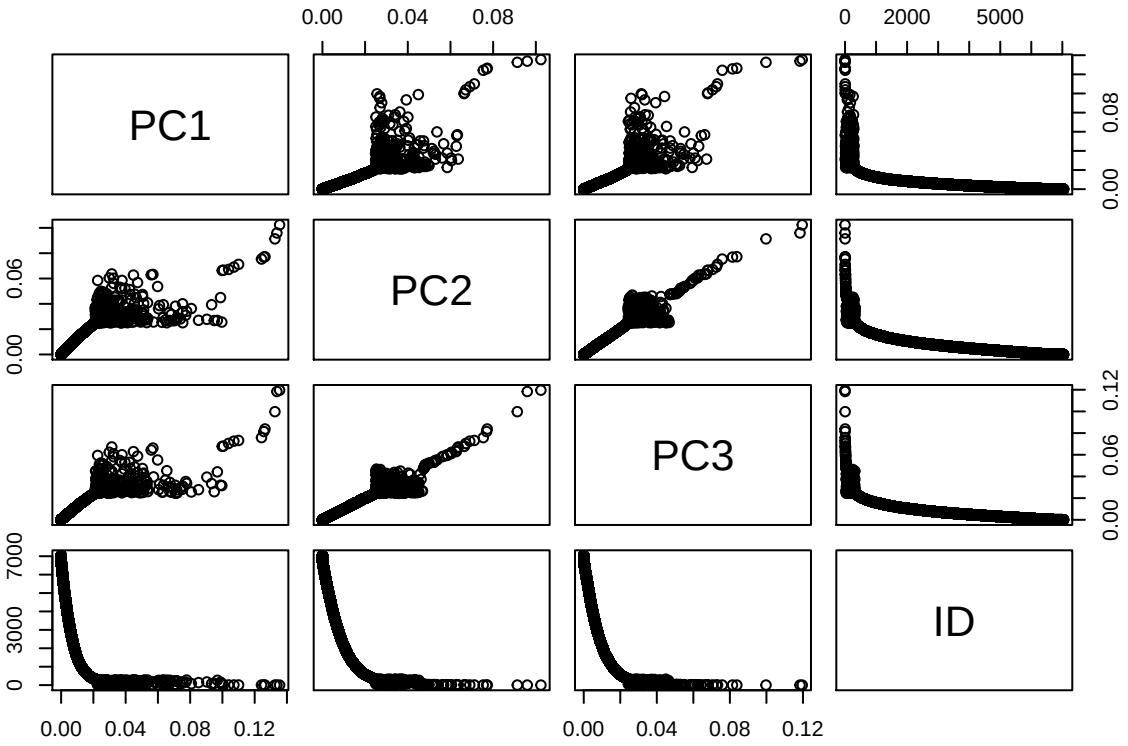
```
## This function will replace the expt$expressionset slot with:

## log2(quant(cpm(batch-correct(data))))

## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize

## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)

## batch_counts: Before batch correction, 47195 entries 0<x<1.

## batch_counts: Using limma's removeBatchEffect to remove batch effect.

## The number of elements which are < 0 after batch correction is: 1689

## transform_counts: Found 1689 values equal to 0, adding 0.5
##  to the matrix.
```

```
fis_batchnormpca <- hpgl_pca(batchnorm_expt, plot_title="limma corrected pca")
fis_batchnormpca$plot
```

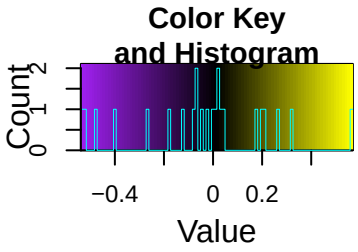


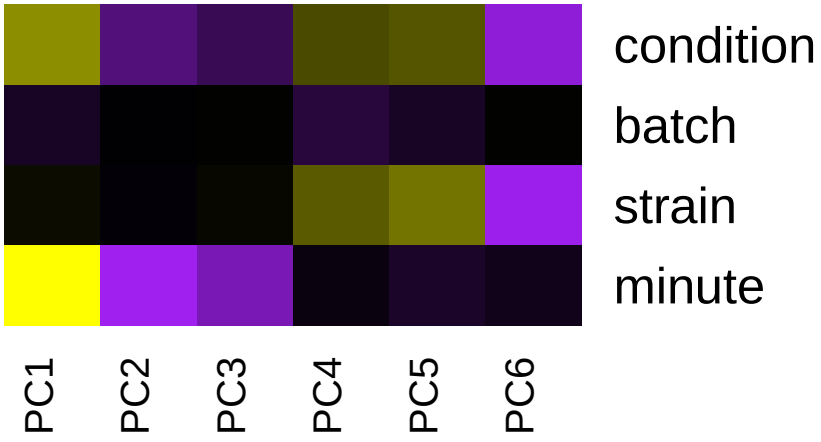

```
test_pca <- pca_information(batchnorm_expt, expt_factors=c("condition","batch","strain","minute"), num_
```

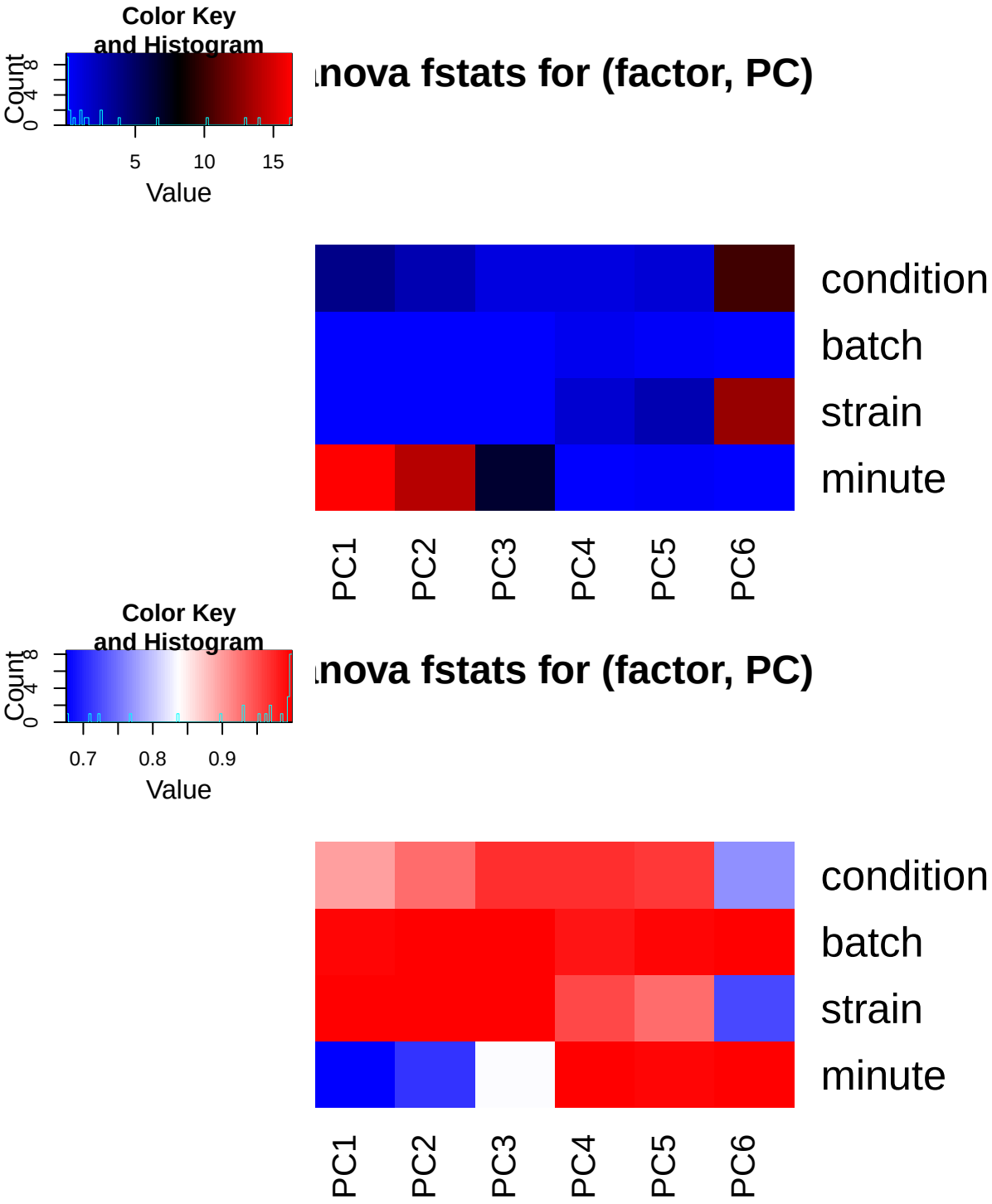
```
## The more shallow the curves in these plots, the more genes responsible for this principle component.
```

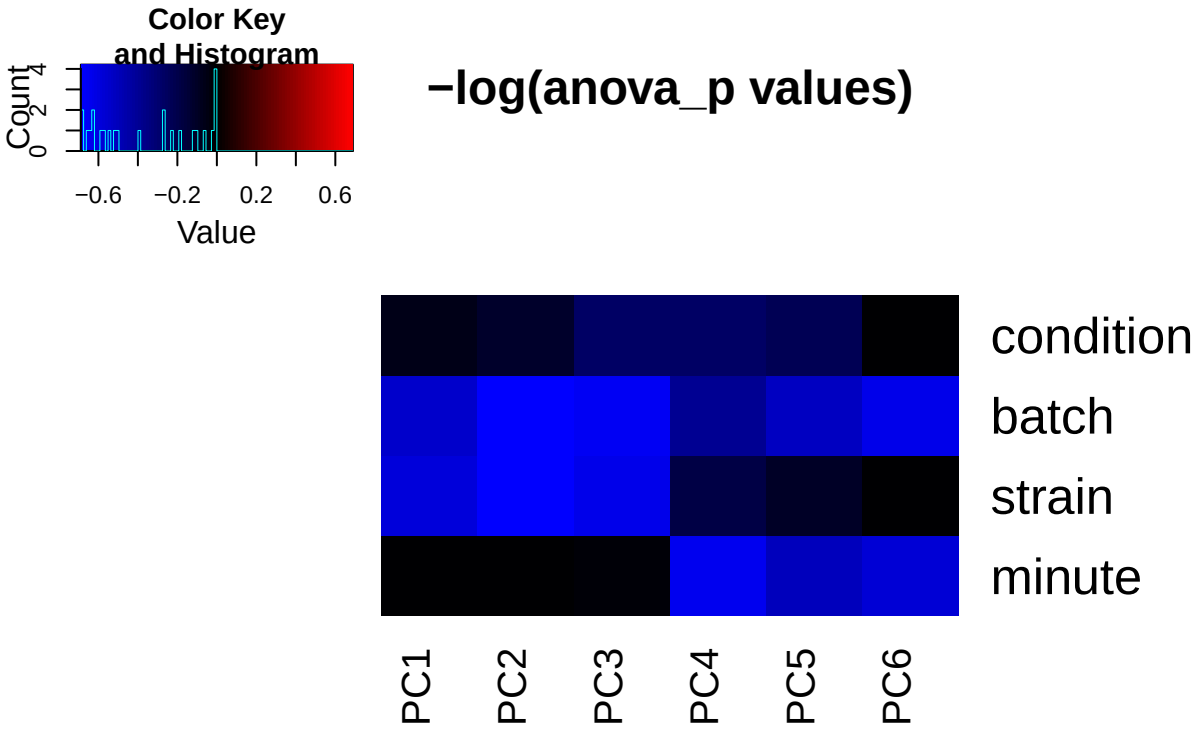



```
## [1] "PC1: 34.70% variance"
## [1] "PC2: 13.72% variance"
```






Color Key
and Histogram
Count
0
4
8
5
10
15
Value
nova fstats for (factor, PC)
condition
batch
strain
minute
PC1
PC2
PC3
PC4
PC5
PC6
Color Key
and Histogram
Count
0
4
8
0.7
0.8
0.9
Value
nova fstats for (factor, PC)
condition
batch
strain
minute
PC1
PC2
PC3
PC4
PC5
PC6



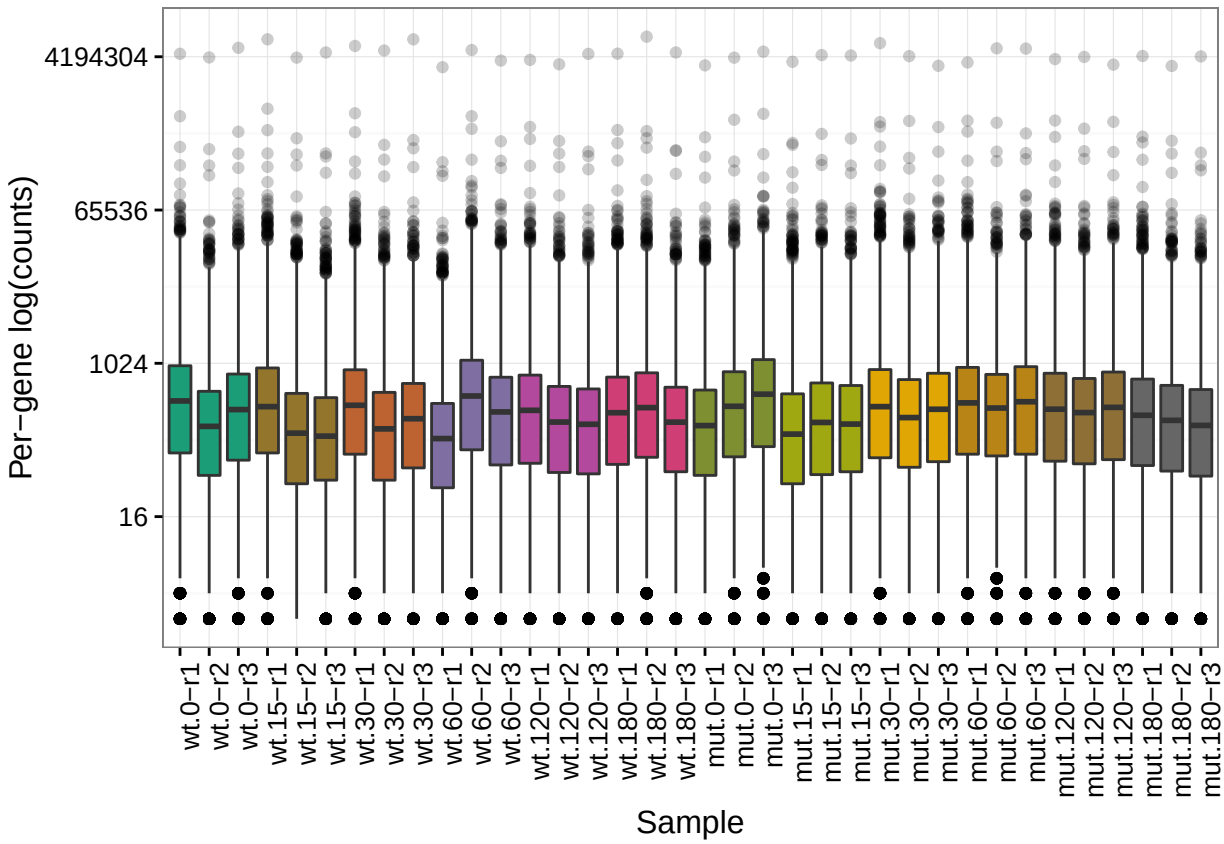
Interesting, the batch normalized pca plot looks much the same as the normalized. The variances are in fact pretty much the exact same...

Look at the data distributions

We have some tools which provide visualizations of the distribution of the data:

```
hpgl_boxplot(fission_expt)
```

```
## I am reasonably sure this should be log scaled and am setting it.
##   If this is incorrect, set scale='raw'

## Warning: Removed 24130 rows containing non-finite values (stat_boxplot).
```



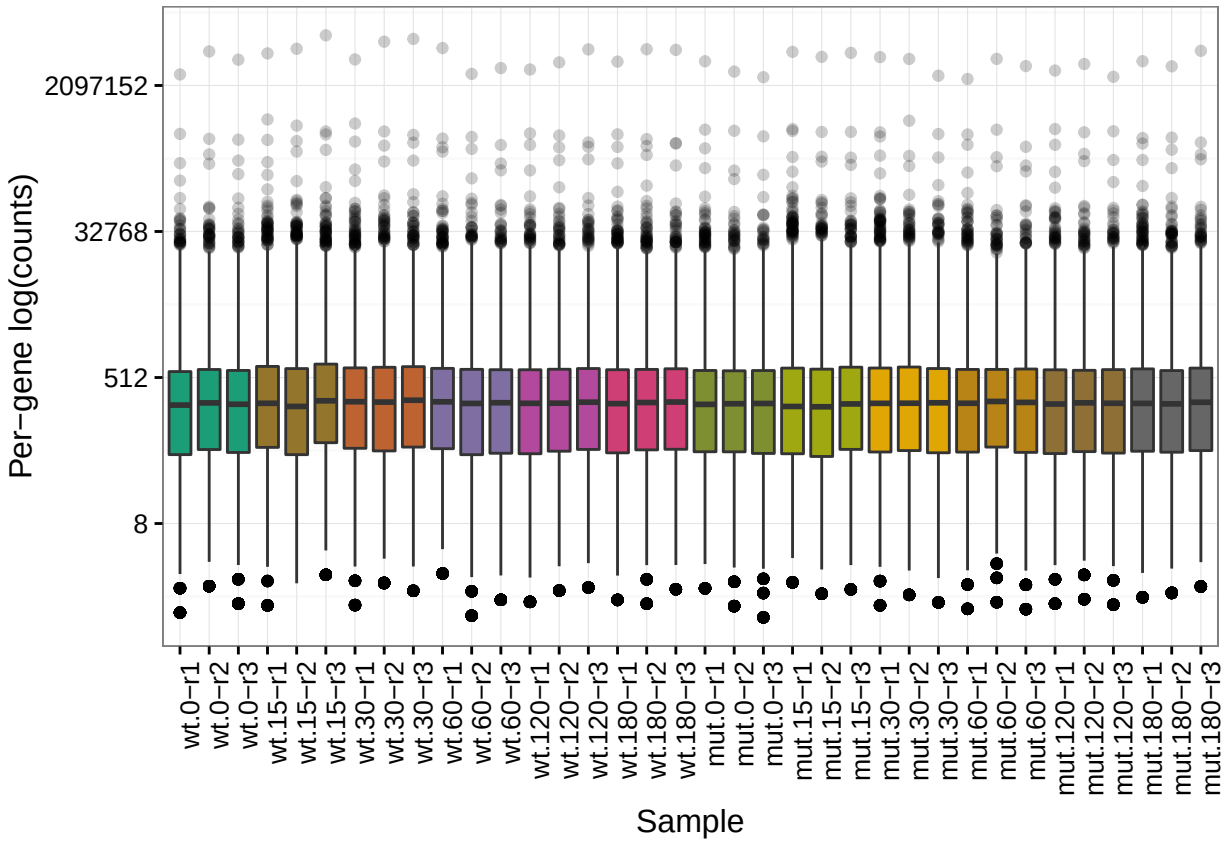

```
sf_expt <- normalize_expt(fission_expt, norm="sf")
```

```
## This function will replace the expt$expressionset slot with:

## sf(data)

## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize

## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)

## Leaving the data in its current base format, keep in mind that
##  some metrics are easier to see when the data is log2 transformed, but
##  EdgeR/DESeq don't like transformed data.

## Leaving the data unconverted.  It is often advisable to cpm/rpkm
##  the data to normalize for sampling differences, keep in mind though that rpkm
##  has some annoying biases, and voom() by default does a cpm (though hpgl_voom()
##  will try to detect this).
```

```
## Not correcting the count-data for batch effects.  If batch is
##  included in EdgerR/limma's model, then this is probably wise; but in extreme
##  batch effects this is a good parameter to play with.

## Warning: replacing previous import 'S4Vectors::Position' by
## 'ggplot2::Position' when loading 'DESeq2'
```

```
hpgl_boxplot(sf_expt)
```

```
## I am reasonably sure this should be log scaled and am setting it.
##   If this is incorrect, set scale='raw'

## Warning: Removed 24130 rows containing non-finite values (stat_boxplot).
```



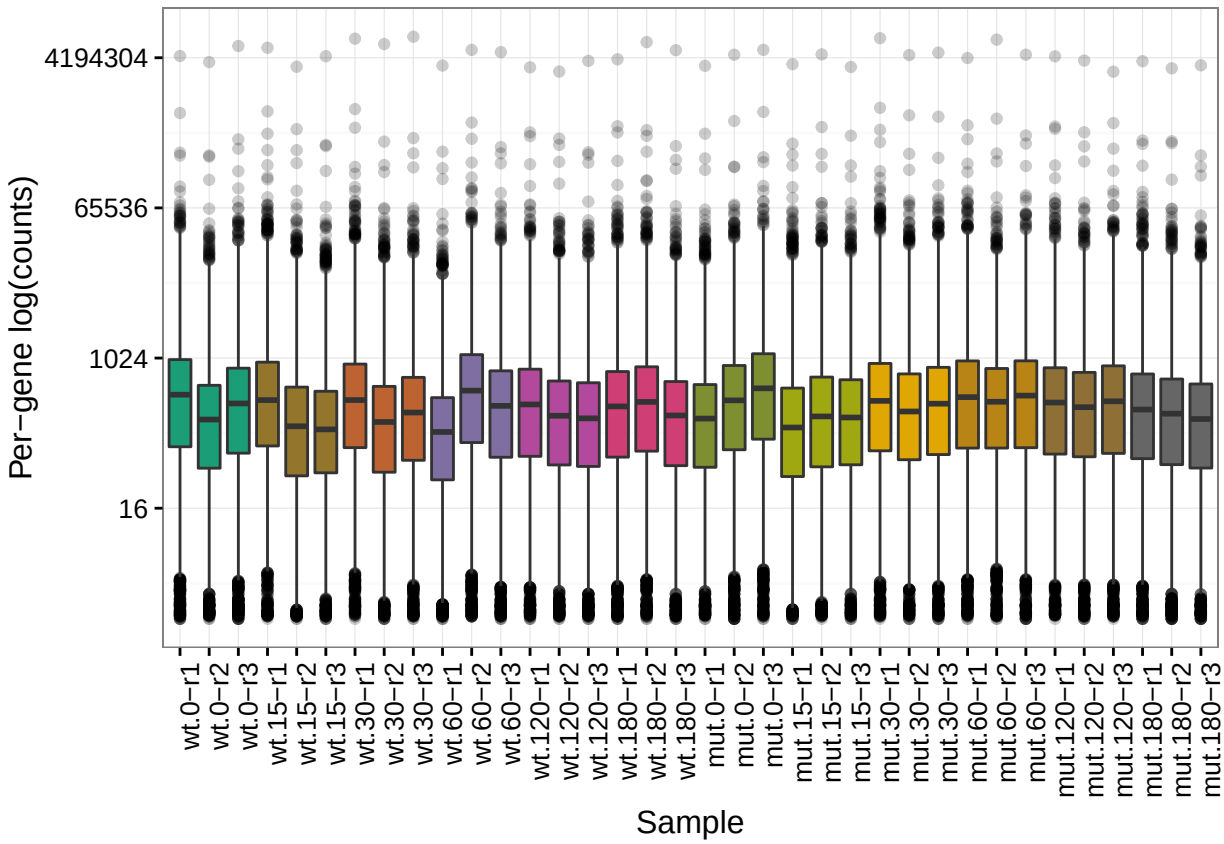

```
tm_expt <- normalize_expt(fission_expt, norm="tmm")
```

```
## This function will replace the expt$expressionset slot with:

## tmm(data)

## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
```

```
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize

## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)

## Leaving the data in its current base format, keep in mind that
##  some metrics are easier to see when the data is log2 transformed, but
##  EdgeR/DESeq don't like transformed data.

## Leaving the data unconverted.  It is often advisable to cpm/rpkm
##  the data to normalize for sampling differences, keep in mind though that rpkm
##  has some annoying biases, and voom() by default does a cpm (though hpgl_voom()
##  will try to detect this).

## Not correcting the count-data for batch effects.  If batch is
##  included in EdgerR/limma's model, then this is probably wise; but in extreme
##  batch effects this is a good parameter to play with.
```

```
hpgl_boxplot(tm_expt)
```

```
## I am reasonably sure this should be log scaled and am setting it.
##   If this is incorrect, set scale='raw'

## Warning: Removed 24130 rows containing non-finite values (stat_boxplot).
```




```
rle_expt <- normalize_expt(fission_expt, norm="rle")
```

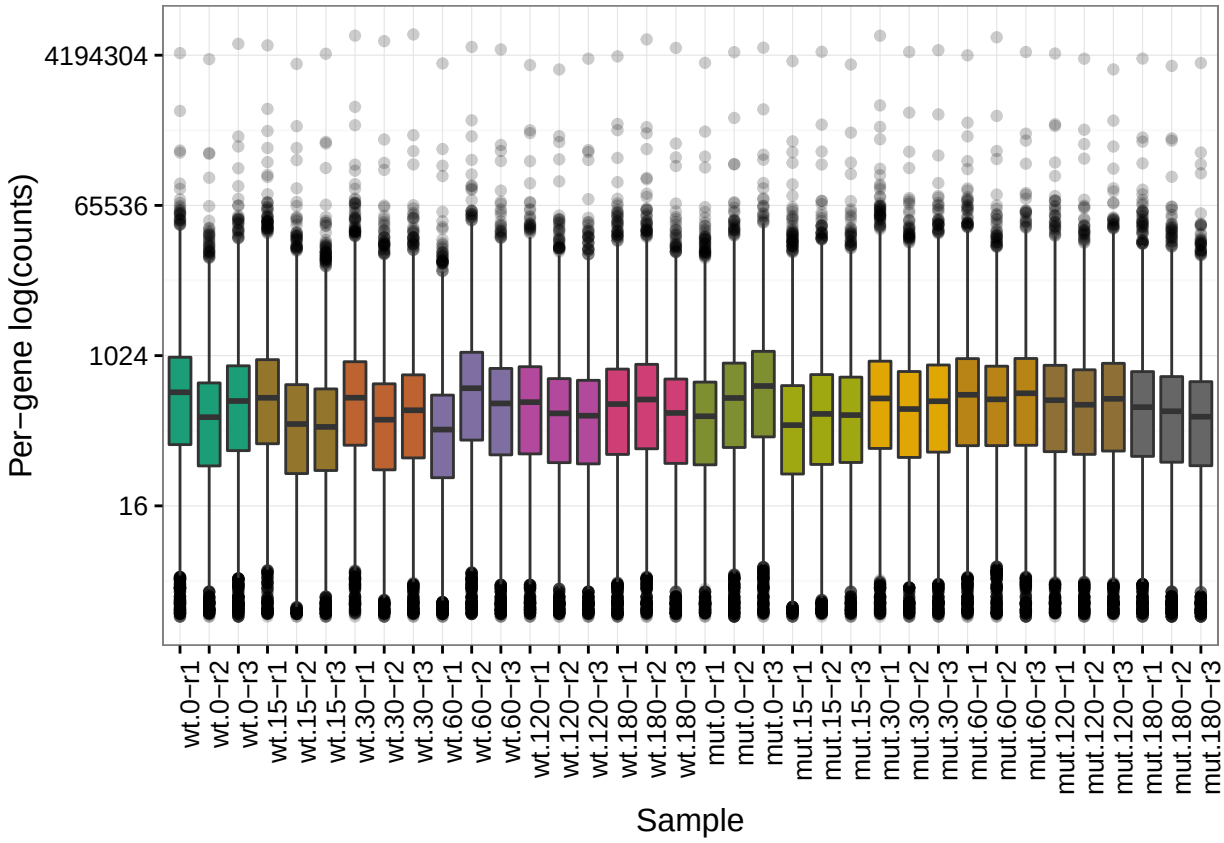
```
## This function will replace the expt$expressionset slot with:

## rle(data)

## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize

## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)

## Leaving the data in its current base format, keep in mind that
##  some metrics are easier to see when the data is log2 transformed, but
##  EdgeR/DESeq don't like transformed data.

## Leaving the data unconverted.  It is often advisable to cpm/rpkm
##  the data to normalize for sampling differences, keep in mind though that rpkm
##  has some annoying biases, and voom() by default does a cpm (though hpgl_voom()
##  will try to detect this).
```

```
## Not correcting the count-data for batch effects.  If batch is
##  included in EdgerR/limma's model, then this is probably wise; but in extreme
##  batch effects this is a good parameter to play with.
```

```
hpgl_boxplot(rle_expt)
```

```
## I am reasonably sure this should be log scaled and am setting it.
##   If this is incorrect, set scale='raw'
```

```
## Warning: Removed 24130 rows containing non-finite values (stat_boxplot).
```




```
up_expt <- normalize_expt(fission_expt, norm="upperquartile")
```

```
## This function will replace the expt$expressionset slot with:
```

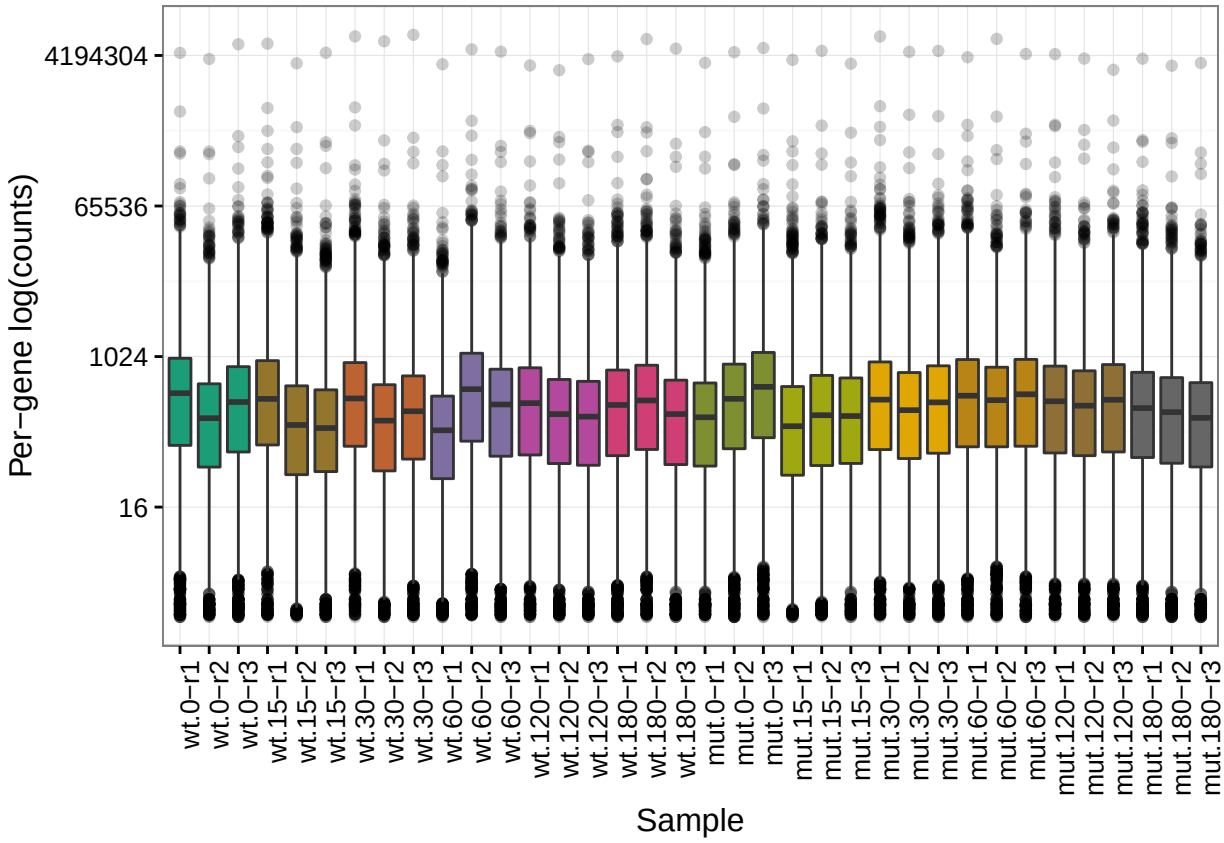
```
## upperquartile(data)
```

```
## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize
```

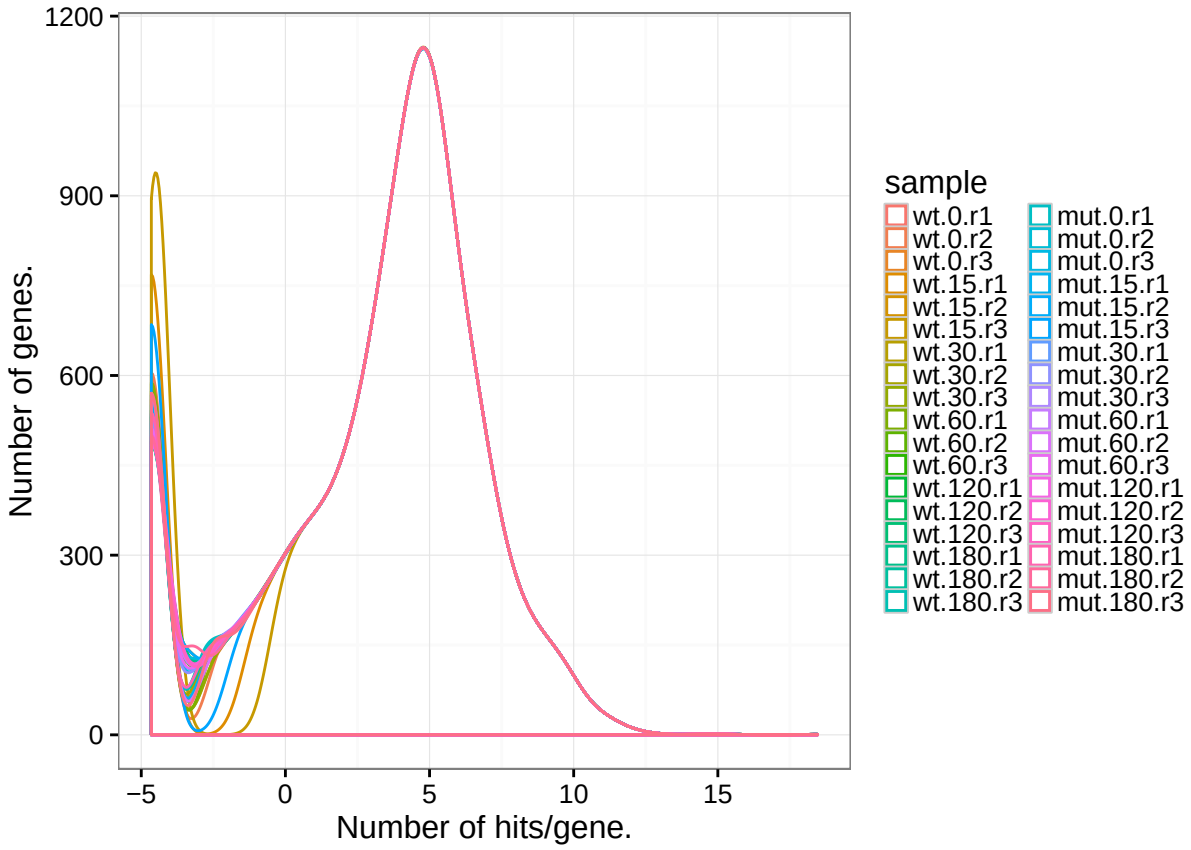
```
## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)

## Leaving the data in its current base format, keep in mind that
##  some metrics are easier to see when the data is log2 transformed, but
##  EdgeR/DESeq don't like transformed data.

## Leaving the data unconverted.  It is often advisable to cpm/rpkm
##  the data to normalize for sampling differences, keep in mind though that rpkm
##  has some annoying biases, and voom() by default does a cpm (though hpgl_voom()
##  will try to detect this).

## Not correcting the count-data for batch effects.  If batch is
##  included in EdgerR/limma's model, then this is probably wise; but in extreme
##  batch effects this is a good parameter to play with.
```

```
hpgl_boxplot(up_expt)
```

```
## I am reasonably sure this should be log scaled and am setting it.
##   If this is incorrect, set scale='raw'

## Warning: Removed 24130 rows containing non-finite values (stat_boxplot).
```

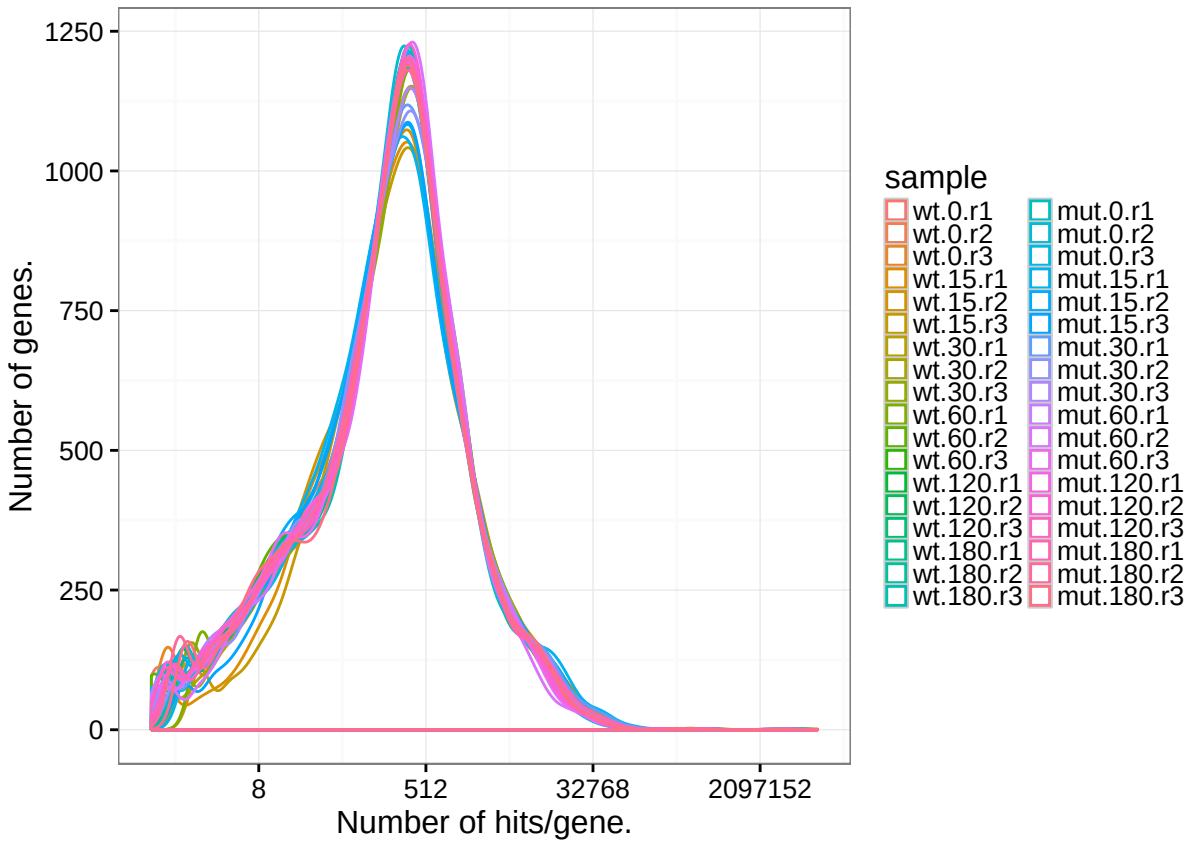


```
hpgl_density(norm_expt)
```



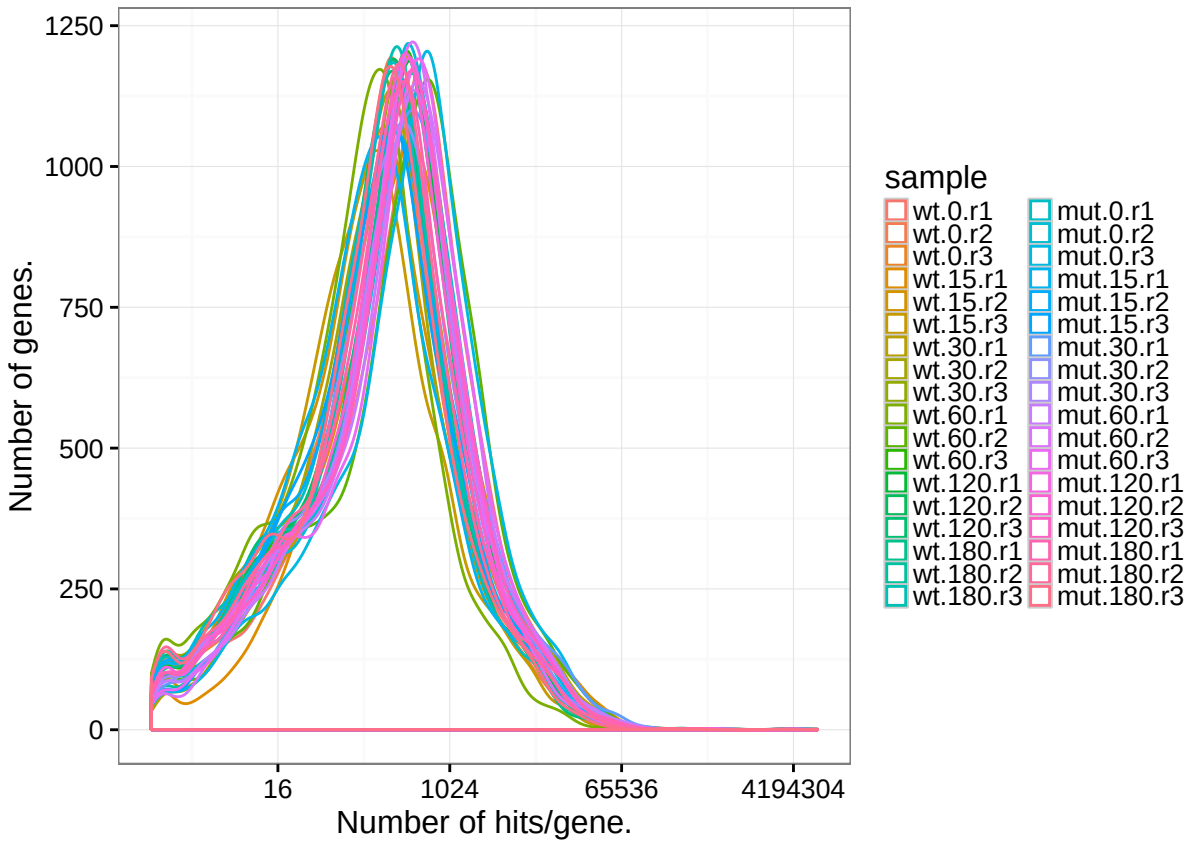

```
hpgl_density(sf_expt)
```

```
## This data will benefit from being displayed on the log scale.

## If this is not desired, set scale='raw'

## Warning: Removed 24130 rows containing non-finite values (stat_density).
```

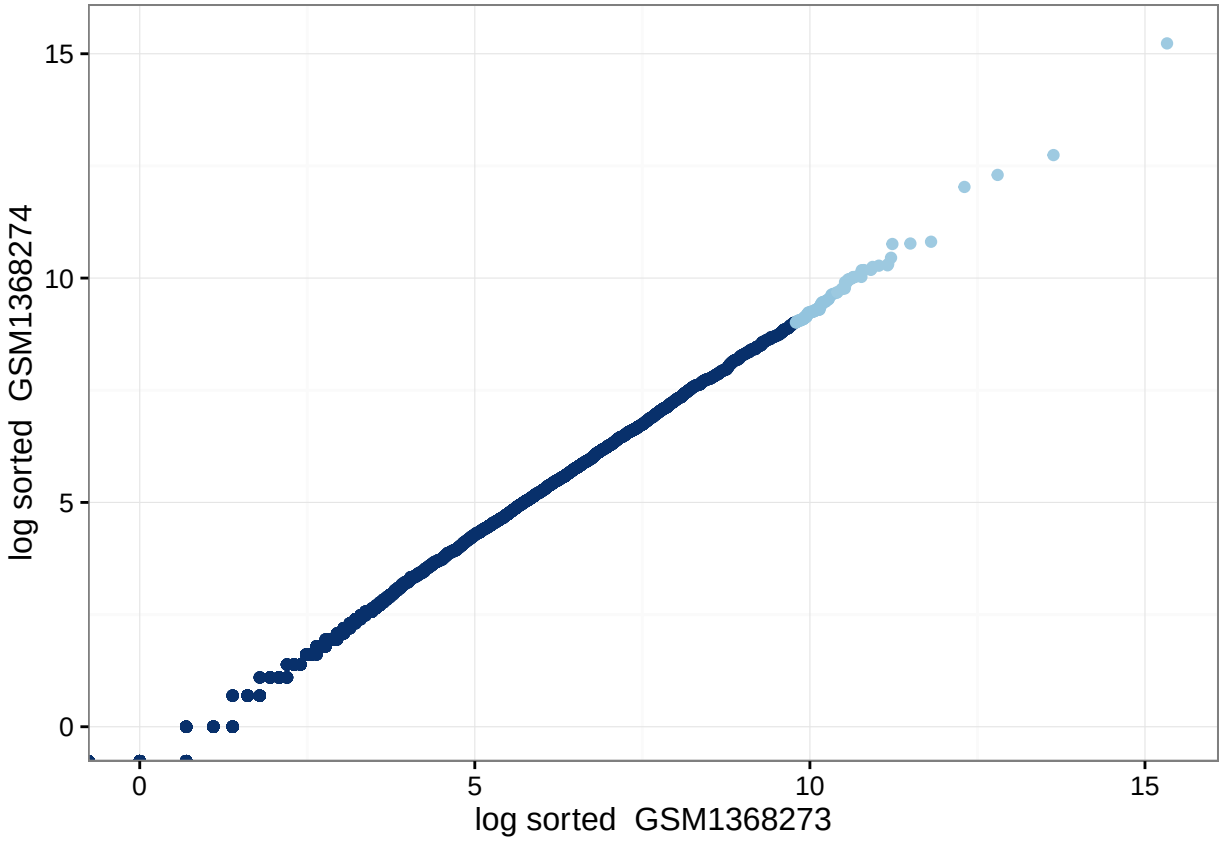



```
hpgl_density(tm_expt)
```

```
## This data will benefit from being displayed on the log scale.
## If this is not desired, set scale='raw'

## Warning: Removed 24130 rows containing non-finite values (stat_density).
```



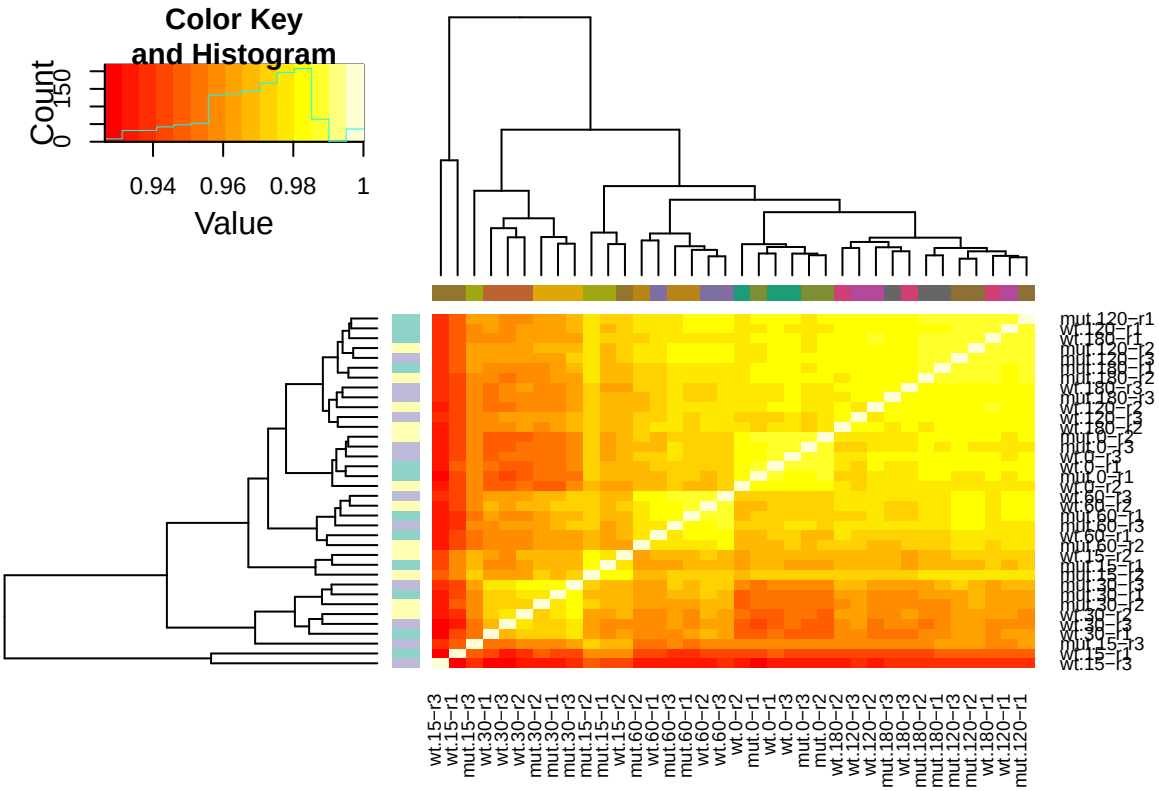

```
compare_12 <- hpgl_qq_plot(fission_expt, x=1, y=2)
compare_12$log
```



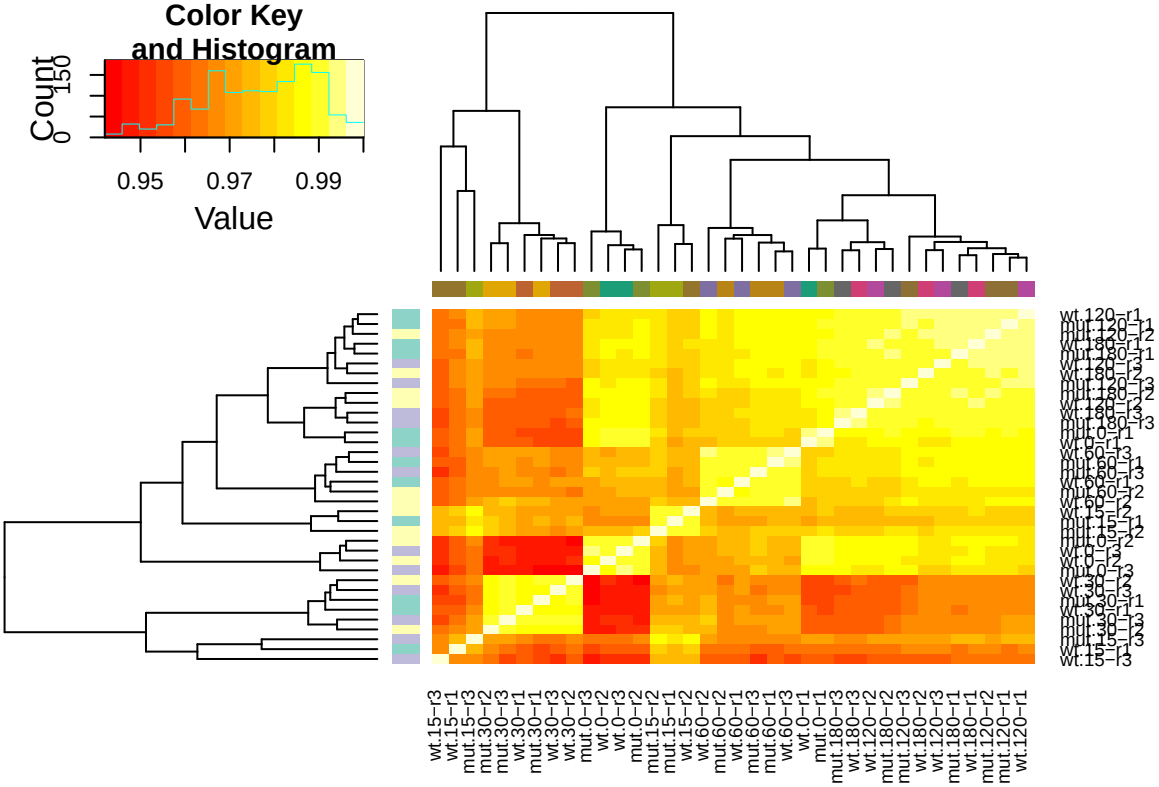
See how they cluster

Ok, so we can further check out how the data cluster with respect to one another...

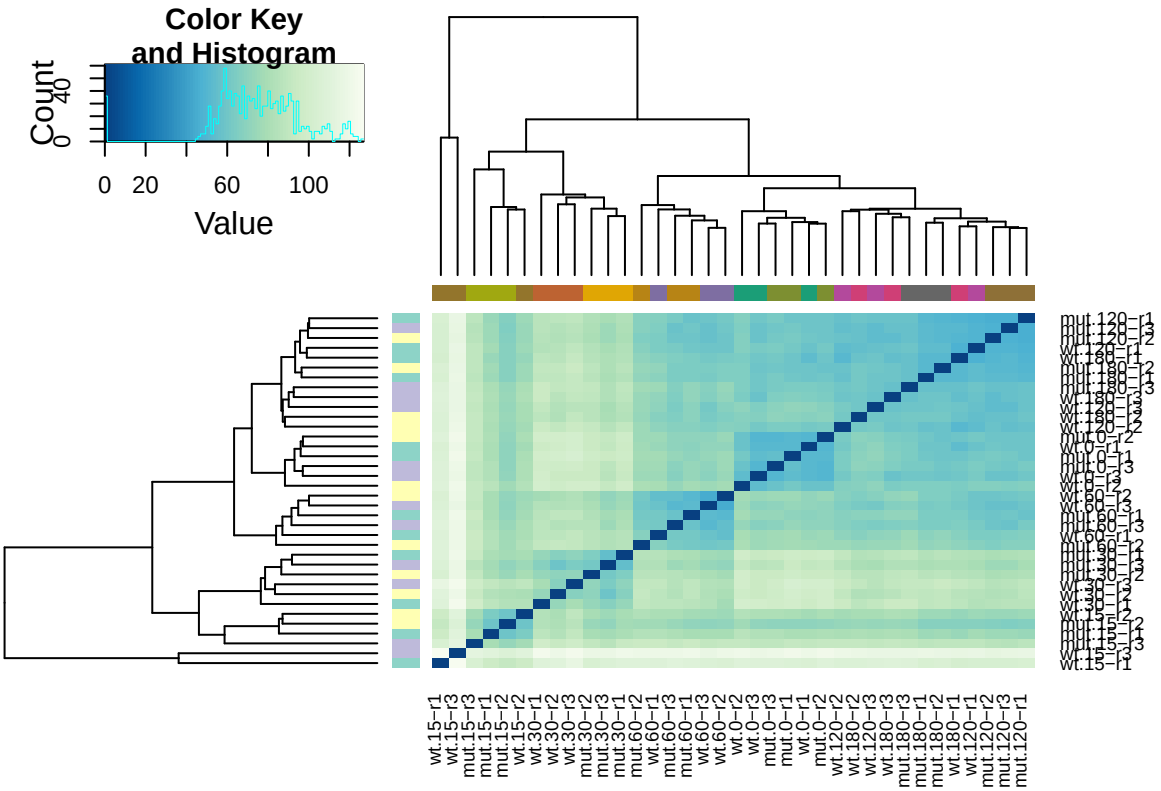
```
hpgl_corheat(norm_expt)
```



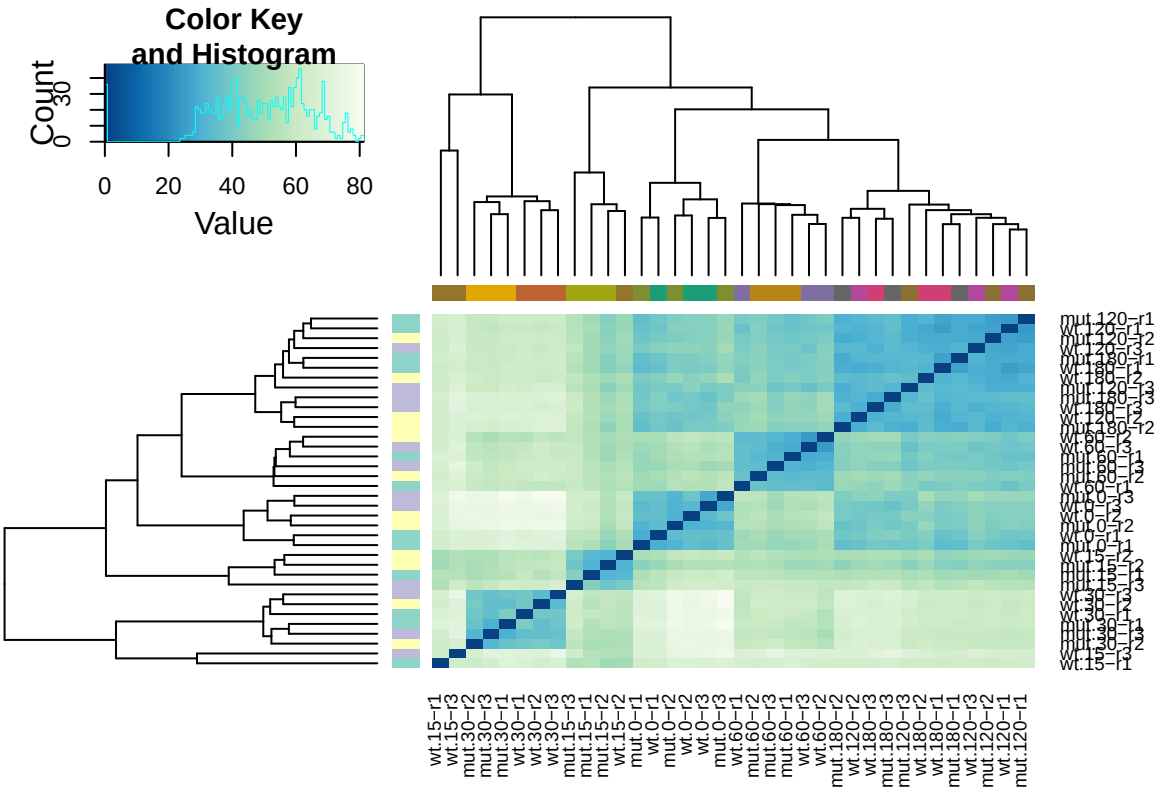

```
hpgl_corheat(batchnorm_expt)
```



```
hpgl_disheat(norm_expt)
```




```
hpgl_disheat(batchnorm_expt)
```



Some simple differential expression analyses

Travis wisely imposes a limit on the amount of time for building vignettes. My tools by default will attempt all possible pairwise comparisons, which takes a long time. Therefore I am going to take a subset of the data and limit these comparisons to that.

```
fun_data <- expt_subset(fission_expt, subset="condition=='wt.120'|condition=='mut.120'")
fun_norm <- normalize_expt(fun_data, batch="limma", norm="quant", transform="log2", convert="cpm")
```

```
## This function will replace the expt$expressionset slot with:

## log2(quant(cpm(batch-correct(data))))

## It saves the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept in the slot called:
##  'new_expt$normalized$normalized_counts$libsize' which is copied into
##  new_expt$best_libsize

## Filter low is false, this should likely be set to something, good
##  choices include cbcb, kofa, pofa (anything but FALSE).  If you want this to
##  stay FALSE, keep in mind that if other normalizations are performed, then the
##  resulting libsizes are likely to be strange (potentially negative!)
```

```
## batch_counts: Before batch correction, 7600 entries 0<x<1.

## batch_counts: Using limma's removeBatchEffect to remove batch effect.

## The number of elements which are < 0 after batch correction is: 175

## transform_counts: Found 175 values equal to 0, adding 0.5
##  to the matrix.
```

Try using limma first

```
limma_comparison <- limma_pairwise(fun_norm)
```

```
## Starting limma pairwise comparison.

## libsize was not specified, this parameter has profound effects on limma's result.

## Using the libsize from expt$best_libsize.

## Limma step 1/6: choosing model.

## Limma step 2/6: running voom

## limma step 3/6: running lmFit

## Limma step 4/6: making and fitting contrasts.

## As a reference, the identity is: mut.120 = mut.120,

## As a reference, the identity is: wt.120 = wt.120,

## Limma step 5/6: Running eBayes and topTable.

## Limma step 6/6: Writing limma outputs.

## limma step 6/6: 1/3: Printing table: mut.120.

## limma step 6/6: 2/3: Printing table: wt.120.

## limma step 6/6: 3/3: Printing table: wt.120_vs_mut.120.
```

```
names(limma_comparison$all_tables)
```

```
## [1] "mut.120"           "wt.120"            "wt.120_vs_mut.120"
```

```
summary(limma_comparison$all_tables$wt.120_vs_mut.120)
```

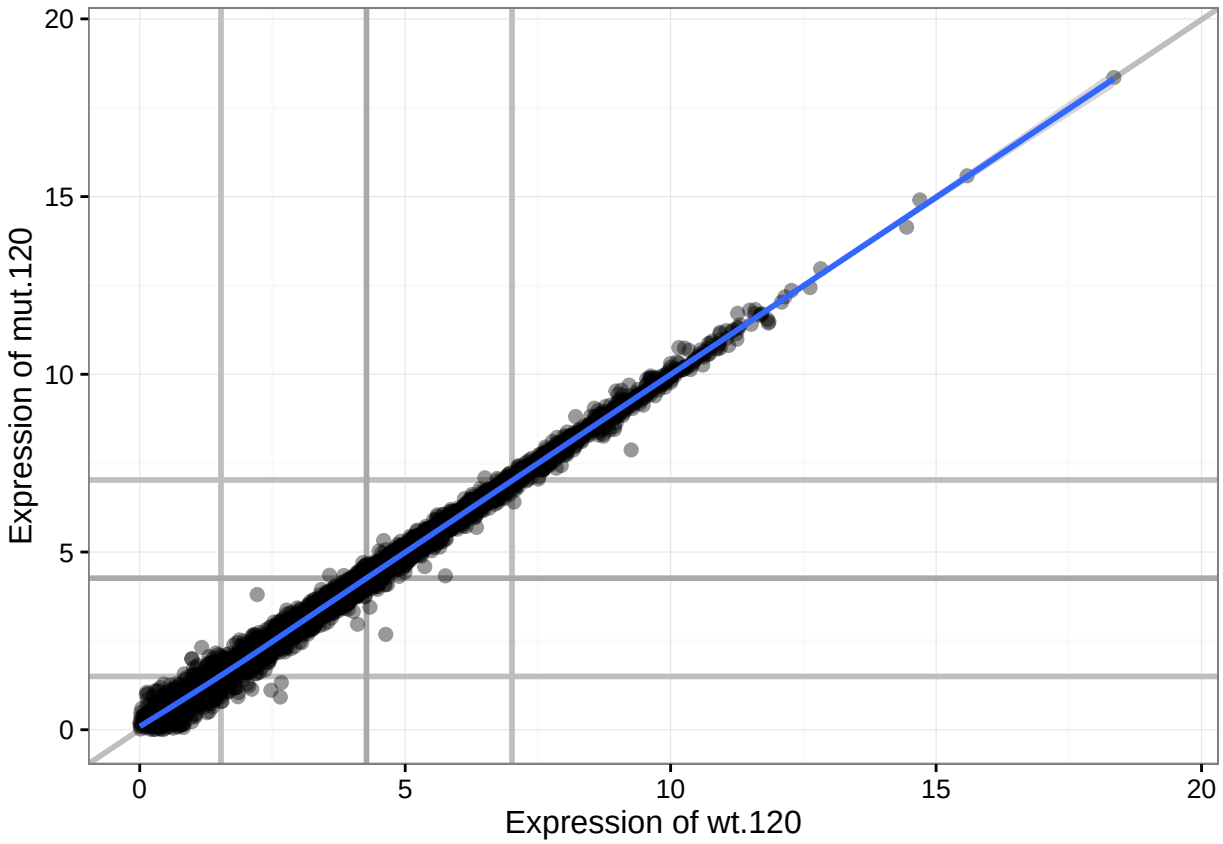
```
##      logFC              AveExpr              t
##  Min.   :-1.5910000   Min.   :-0.8809   Min.   :-31.41000
##  1st Qu.:-0.0958500   1st Qu.: 1.6650   1st Qu.: -1.33750
##  Median : 0.0000002   Median : 4.2660   Median :  0.00002
##  Mean   : 0.0052609   Mean   : 3.8557   Mean   : -0.15042
##  3rd Qu.: 0.0976950   3rd Qu.: 5.7225   3rd Qu.:  1.14750
##  Max.   : 2.1010000   Max.   :18.3500   Max.   : 36.59000
##    P.Value           adj.P.Val              B             qvalue
##  Length:7039        Length:7039        Min.   :-7.571   Length:7039
##  Class :character   Class :character   1st Qu.:-7.067   Class :character
##  Mode  :character   Mode  :character   Median :-6.421   Mode  :character
##                                        Mean   :-5.701
##                                        3rd Qu.:-5.029
##                                        Max.   : 6.493
```

```
wt_120 <- limma_comparison$all_tables$wt.120
mut_120 <- limma_comparison$all_tables$mut.120
scatter_wt_mut <- limma_coefficient_scatter(limma_comparison, x="wt.120", y="mut.120", gvis_filename=NU
```

```
## This can do comparisons among the following columns in the limma result:

## mut.120wt.120wt.120_vs_mut.120

## Actually comparing wt.120 and mut.120.

## Setting binwidth to 0.0384765861670427 in order to have 500 bins.
```

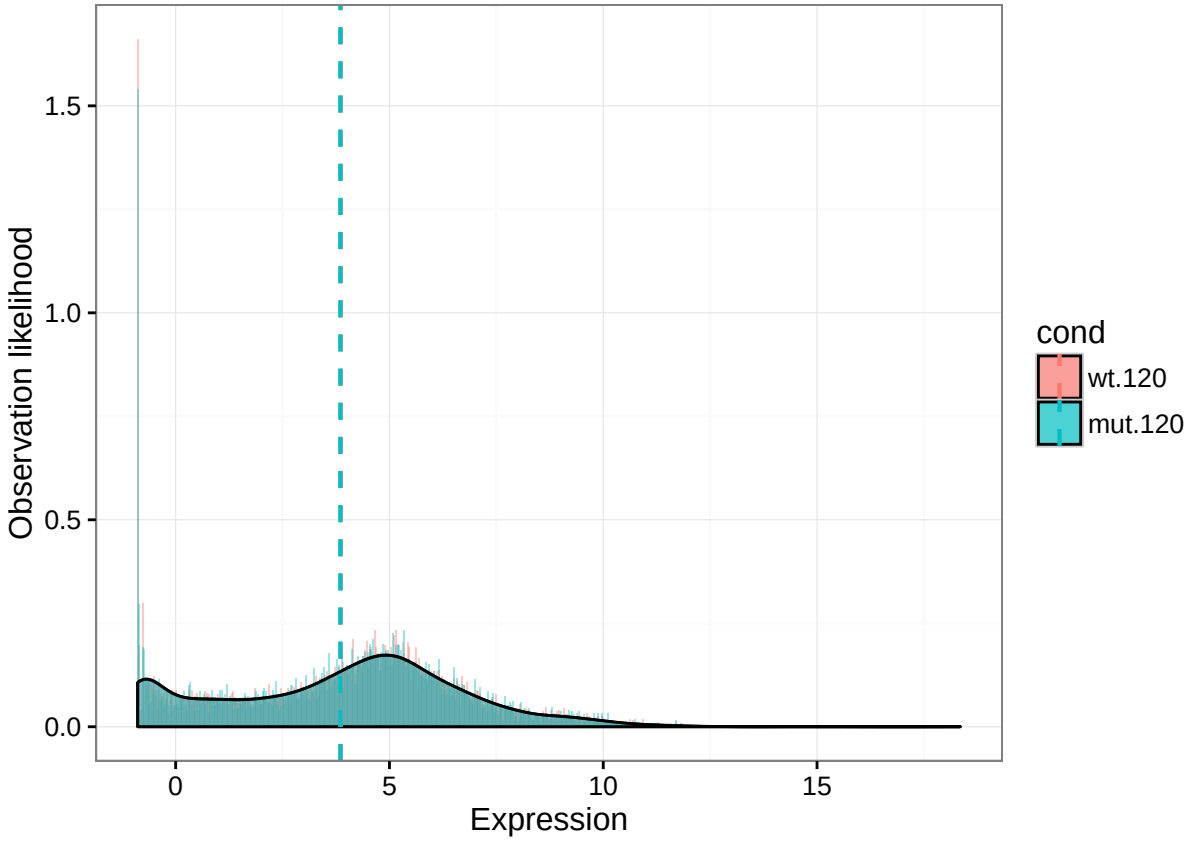
```
scatter_wt_mut$scatter
```

```
## Warning: Removed 1069 rows containing non-finite values (stat_smooth).

## Warning: Removed 1069 rows containing missing values (geom_point).
```




```
scatter_wt_mut$both_histogram
```

```
## $plot
```




```
##
## $data_summary
##      wt.120           mut.120
##  Min.   :-0.8833   Min.   :-0.8883
##  1st Qu.: 1.6569   1st Qu.: 1.6557
##  Median : 4.2711   Median : 4.2644
##  Mean   : 3.8583   Mean   : 3.8531
##  3rd Qu.: 5.7084   3rd Qu.: 5.7203
##  Max.   :18.3500   Max.   :18.3500
##
## $uncor_t
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  play_all$expression and play_all$cond
##
##         wt.120
## mut.120 0.91
##
## P value adjustment method: none
##
## $bon_t
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  play_all$expression and play_all$cond
##
```

```
##         wt.120
## mut.120 0.91
##
## P value adjustment method: bonferroni
```

Then DESeq2

```
deseq_comparison <- deseq2_pairwise(fun_data, model_batch=TRUE)
```

```
## Starting DESeq2 pairwise comparisons.

## DESeq2 step 1/5: Including batch and condition in the deseq model.

## factor levels were dropped which had no samples

## DESeq2 step 2/5: Estimate size factors.

## DESeq2 step 3/5: estimate Dispersions.

## DESeq2 step 4/5: nbinomWaldTest.

## DESeq2 step 5/5: 1/1: Printing table: wt.120_vs_mut.120

## Collected coefficients for: mut.120

## Collected coefficients for: wt.120
```

```
summary(deseq_comparison$all_tables$wt.120_vs_mut.120)
```

```
##     baseMean           logFC             lfcSE             stat
##  Min.   :      0   Min.   :-0.4275   Min.   :0.0115   Min.   :-4.2710
##  1st Qu.:     30   1st Qu.:-0.0436   1st Qu.:0.0982   1st Qu.:-0.4824
##  Median :    215   Median :-0.0001   Median :0.1089   Median :-0.0026
##  Mean   :   1645   Mean   : 0.0036   Mean   :0.0985   Mean   : 0.0299
##  3rd Qu.:    597   3rd Qu.: 0.0480   3rd Qu.:0.1154   3rd Qu.: 0.5072
##  Max.   :3976000   Max.   : 0.5620   Max.   :0.1182   Max.   : 4.7570
##                    NA's   :387       NA's   :387      NA's   :387
##     P.Value           adj.P.Val           qvalue
##  Min.   :0.000002   Min.   :0.006838   Min.   :0.01383
##  1st Qu.:0.406350   1st Qu.:1.000000   1st Qu.:1.00000
##  Median :0.646500   Median :1.000000   Median :1.00000
##  Mean   :0.615159   Mean   :0.982937   Mean   :0.99856
##  3rd Qu.:0.854750   3rd Qu.:1.000000   3rd Qu.:1.00000
##  Max.   :1.000000   Max.   :1.000000   Max.   :1.00000
##
```

And EdgeR

```
edger_comparison <- edger_pairwise(fun_data, model_batch=TRUE)
```

```
## Starting edgeR pairwise comparisons.

## EdgeR step 1/9: normalizing data.

## EdgeR step 2/9: Estimating the common dispersion.

## EdgeR step 3/9: Estimating dispersion across genes.

## EdgeR step 4/9: Estimating GLM Common dispersion.

## EdgeR step 5/9: Estimating GLM Trended dispersion.

## EdgeR step 6/9: Estimating GLM Tagged dispersion.

## EdgeR step 7/9: Running glmFit.

## EdgeR step 8/9: Making pairwise contrasts.

## As a reference, the identity is: mut.120 = mut.120,

## As a reference, the identity is: wt.120 = wt.120,

## EdgeR step 9/9: 1/1: Printing table: wt.120_vs_mut.120.
```

```
summary(edger_comparison$all_tables$wt.120_vs_mut.120)
```

```
##      logFC              logCPM             LR              PValue
##  Min.   :-4.72200   Min.   :-2.554   Min.   : 0.00000   Length:7039
##  1st Qu.:-0.12195   1st Qu.: 1.494   1st Qu.: 0.06314   Class :character
##  Median : 0.00000   Median : 4.234   Median : 0.36910   Mode  :character
##  Mean   :-0.03085   Mean   : 3.599   Mean   : 1.01156
##  3rd Qu.: 0.09977   3rd Qu.: 5.700   3rd Qu.: 1.26400
##  Max.   : 5.20200   Max.   :18.400   Max.   :48.90000
##      FDR                qvalue
##  Length:7039        Length:7039
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

My stupid basic comparison

```
basic_comparison <- basic_pairwise(fun_data)
```

```
## Starting basic pairwise comparison.

## Basic step 1/3: Creating median and variance tables.

## Basic step 2/3: Performing comparisons.

## Basic step 2/3: 1/1: Performing log2 subtraction: wt.120_vs_mut.120

## Basic step 3/3: Creating faux DE Tables.

## Basic: Returning tables.
```

```
summary(basic_comparison$all_tables$wt.120_vs_mut.120)
```

```
##  numerator_median   denominator_median  numerator_var
##  Min.   :     0.0   Min.   :     0.04   Length:7039
##  1st Qu.:     2.4   1st Qu.:     2.85   Class :character
##  Median :    17.6   Median :    19.43   Mode  :character
##  Mean   :   141.5   Mean   :   137.59
##  3rd Qu.:    49.2   3rd Qu.:    53.69
##  Max.   :350300.0   Max.   :296700.00
##  denominator_var          t                p
##  Length:7039        Min.   :-7.7150   Length:7039
##  Class :character   1st Qu.: 0.1175   Class :character
##  Mode  :character   Median : 0.8897   Mode  :character
##                     Mean   : 0.8374
##                     3rd Qu.: 1.6050
##                     Max.   :14.6000
##      logFC
##  Min.   :-3.63900
##  1st Qu.:-0.28705
##  Median :-0.12100
##  Mean   :-0.13283
##  3rd Qu.: 0.05047
##  Max.   : 4.51700
```

Combine them all

```
all_comparisons <- all_pairwise(fun_data, model_batch=TRUE)
```

```
## Starting limma pairwise comparison.

## libsize was not specified, this parameter has profound effects on limma's result.

## Using the libsize from expt$normalized$normalized_counts.
```

```
## Limma step 1/6: choosing model.

## Limma step 2/6: running voom

## The voom input was not cpm, converting now.

## The voom input was not log2, transforming now.

## limma step 3/6: running lmFit

## Limma step 4/6: making and fitting contrasts.

## As a reference, the identity is: mut.120 = mut.120,

## As a reference, the identity is: wt.120 = wt.120,

## Limma step 5/6: Running eBayes and topTable.

## Limma step 6/6: Writing limma outputs.

## limma step 6/6: 1/3: Printing table: mut.120.

## limma step 6/6: 2/3: Printing table: wt.120.

## limma step 6/6: 3/3: Printing table: wt.120_vs_mut.120.

## Starting DESeq2 pairwise comparisons.

## DESeq2 step 1/5: Including batch and condition in the deseq model.

## factor levels were dropped which had no samples

## DESeq2 step 2/5: Estimate size factors.

## DESeq2 step 3/5: estimate Dispersions.

## DESeq2 step 4/5: nbinomWaldTest.

## DESeq2 step 5/5: 1/1: Printing table: wt.120_vs_mut.120

## Collected coefficients for: mut.120

## Collected coefficients for: wt.120

## Starting edgeR pairwise comparisons.

## EdgeR step 1/9: normalizing data.
```

```
## EdgeR step 2/9: Estimating the common dispersion.
## EdgeR step 3/9: Estimating dispersion across genes.
## EdgeR step 4/9: Estimating GLM Common dispersion.
## EdgeR step 5/9: Estimating GLM Trended dispersion.
## EdgeR step 6/9: Estimating GLM Tagged dispersion.
## EdgeR step 7/9: Running glmFit.
## EdgeR step 8/9: Making pairwise contrasts.
## As a reference, the identity is: mut.120 = mut.120,
## As a reference, the identity is: wt.120 = wt.120,
## EdgeR step 9/9: 1/1: Printing table: wt.120_vs_mut.120.
## Starting basic pairwise comparison.
## Basic step 1/3: Creating median and variance tables.
## Basic step 2/3: Performing comparisons.
## Basic step 2/3: 1/1: Performing log2 subtraction: wt.120_vs_mut.120
## Basic step 3/3: Creating faux DE Tables.
## Basic: Returning tables.
## 1/1: Comparing analyses: wt.120_vs_mut.120
all_combined <- combine_de_tables(all_comparisons)
## Working on table 1/1: wt.120_vs_mut.120
## The table is: wt.120_vs_mut.120
sig_genes <- extract_significant_genes(all_combined, excel=NULL)
## Writing excel data sheet 1/1
## Assuming the fold changes are on the log scale and so taking >< 0
## After (adj)p filter, the up genes table has 7 genes.
## After (adj)p filter, the down genes table has 8 genes.
## Assuming the fold changes are on the log scale and so taking -1 * fc
## After fold change filter, the up genes table has 7 genes.
## After fold change filter, the down genes table has 8 genes.
## Not printing excel sheets for the significant genes.
```

Ontology searches

The following works, but on travis to causes the vignette build time to exceed the maximum allowed, which is weird because on my computer it only takes like 4 minutes.

```
limma_results <- limma_comparison$all_tables
## The set of comparisons performed
names(limma_results)
table <- limma_results$wt.120_vs_mut.120
dim(table)
gene_names <- rownames(table)

updown_genes <- get_sig_genes(table)
##orthologs <- read.table("ftp://ftp.ebi.ac.uk/pub/databases/pombase/pombe/orthologs/cerevisiae-ortholog
##colnames(orthologs) <- c("pombe","cerevisiae")

##head(updown_genes$up_genes)
##updown_genes$up_genes = merge(updown_genes$up_genes, orthologs, by.x="row.names", by.y="pombe")
##rownames(updown_genes$up_genes) = make.names(updown_genes$up_genes$cerevisiae, unique=TRUE)
##updown_genes$down_genes = merge(updown_genes$down_genes, orthologs, by.x="row.names", by.y="pombe")
##rownames(updown_genes$down_genes) = make.names(updown_genes$down_genes$cerevisiae, unique=TRUE)

require.auto("GenomicFeatures")
require.auto("biomaRt")
ensembl_pombe <- biomaRt::useMart("fungal_mart", dataset="spombe_eg_gene", host="fungi.ensembl.org")
pombe_filters <- biomaRt::listFilters(ensembl_pombe)
head(pombe_filters, n=20) ## 11 looks to be my guy

## getBM(attributes=c('hgnc_symbol', 'chromosome_name', 'start_position', 'end_position'), filters='with
pombe_goids <- biomaRt::getBM(attributes=c('pombase_gene_name', 'go_accession'), values=gene_names, mart
colnames(pombe_goids) <- c("ID","GO")
pombe <- GenomicFeatures::makeTxDbFromBiomart(biomart ="fungal_mart", dataset = "spombe_eg_gene", host=
pombe_transcripts <- as.data.frame(GenomicFeatures::transcriptsBy(pombe))
lengths <- pombe_transcripts[,c("group_name","width")]
colnames(lengths) <- c("ID","width")
## Something useful I didn't notice before:
## makeTranscriptDbFromGFF()  ## From GenomicFeatures, much like my own gff2df()

goseq_search <- simple_goseq(de_genes=updown_genes$up_genes, lengths=lengths, goids=pombe_goids)
goseq_search$mfp_plot

##cluster_search = simple_clusterprofiler(de_genes=updown_genes$up_genes, goids=pombe_goids, gff="pombe
topgo_search = simple_topgo(de_genes=updown_genes$up_genes, goids_df=pombe_goids)
topgo_search$pvalue_plots$BP

gff_from_txdb <- GenomicFeatures::asGFF(pombe)
## why is GeneID: getting prefixed to the IDs!?
gff_from_txdb$ID <- gsub(x=gff_from_txdb$ID, pattern="GeneID:", replacement="")
written_gff <- rtracklayer::export.gff3(gff_from_txdb, con="pombe.gff")
gostats_search = simple_gostats(updown_genes$up_genes, "pombe.gff", pombe_goids, gff_type="gene")
## Oops I forgot to add my 25 character wrapper for these plots, whatever
## that is weird I thought I did!
gostats_search$pvalue_plots$mfp_plot_over
head(gostats_search$mf_over_all)
```