Example hpgltools usage with fission

Ashton Trey Belew abelew@gmail.com

2015-04-27

Example hpgltool usage with a real data set (fission)

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

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)
autoloads_all()
```

Import fission

```
library(hpgltools)
autoloads_all()
```

```
##
## groupGOTerms:    GOBPTerm, GOMFTerm, GOCCTerm environments built.

## Warning: replacing previous import by 'graphics::plot' when loading
## 'seqLogo'
```

```
require.auto("fission")
data(fission)
```

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)
## Write it out in the format expected by my toys
write.csv(meta, file="fission.csv")
## 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("fission.csv", count_dataframe=fission_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."
```

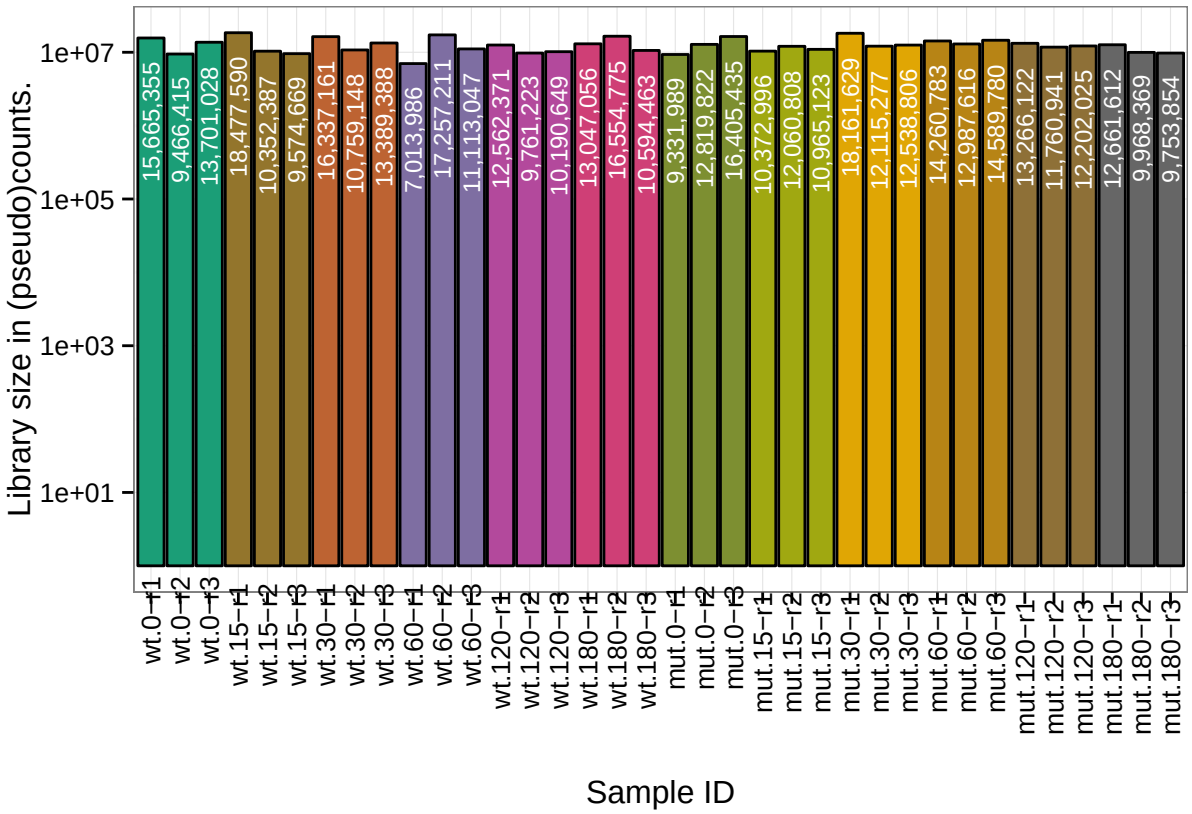
Normalizing and exploring data

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.

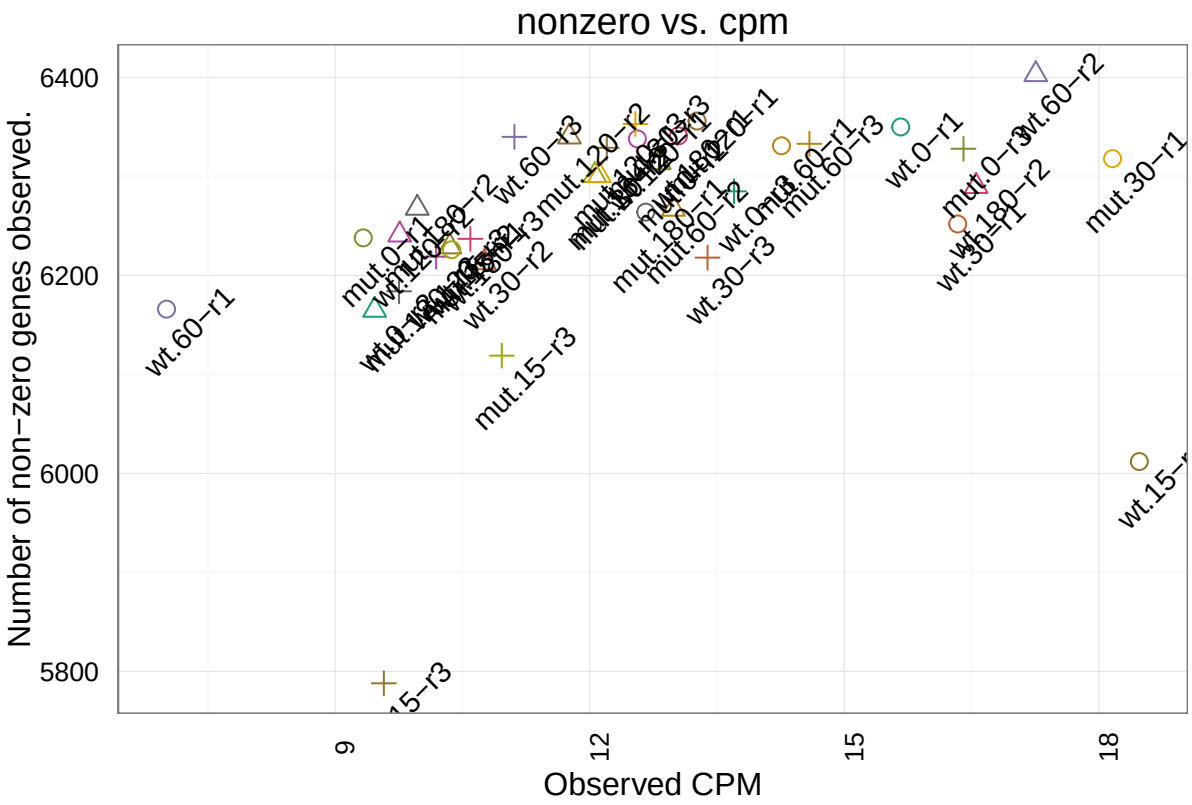
```
full_design = fission_expt$definitions
## First make a bar plot of the library sizes in the experiment.
## Notice that the colors were auto-chosen by create_expt() and they should
## be maintained throughout this process
fis_libsize = hpgl_libsize(expt=fission_expt)
```

```
## Adding log10
```

```
fis_libsize
```



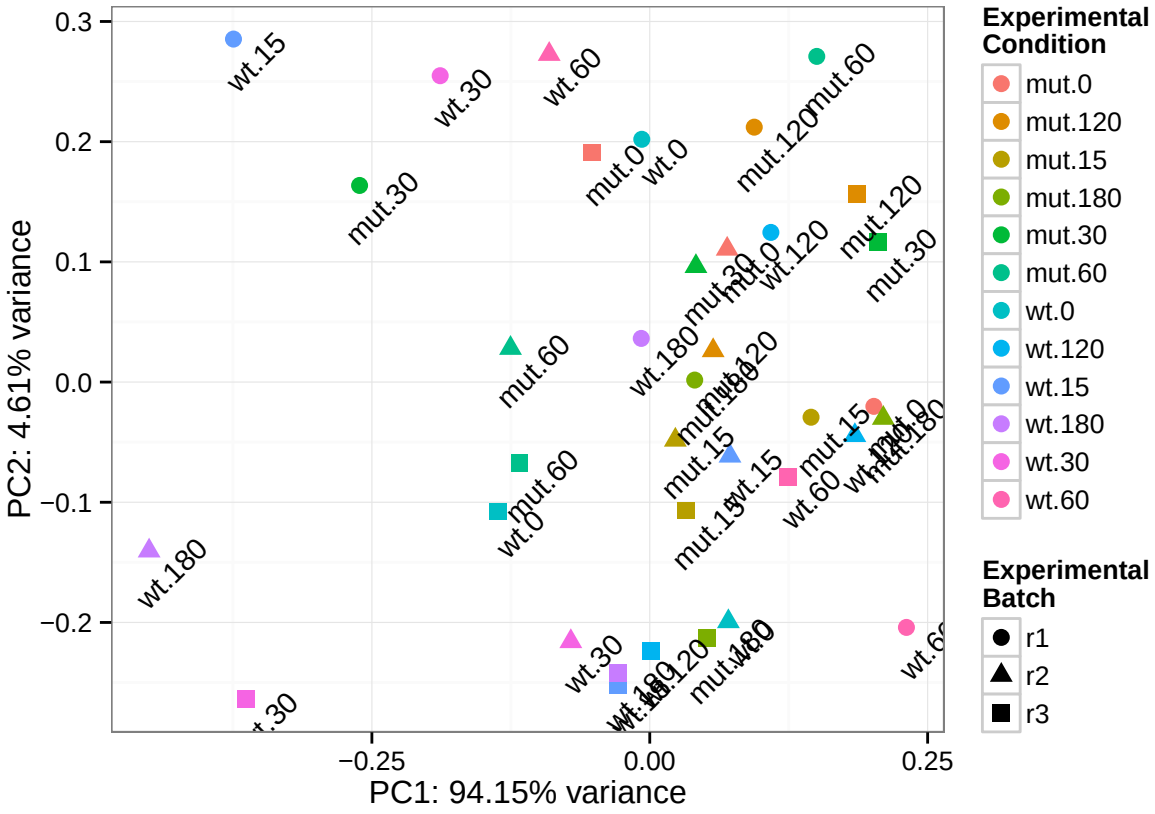

```
## Here we see that the wild type replicate 3 sample for 15 minutes has fewer non-zero genes than all i
fis_nonzero = hpgl_nonzero(expt=fission_expt, labels="boring", title="nonzero vs. cpm")
fis_nonzero
```



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.

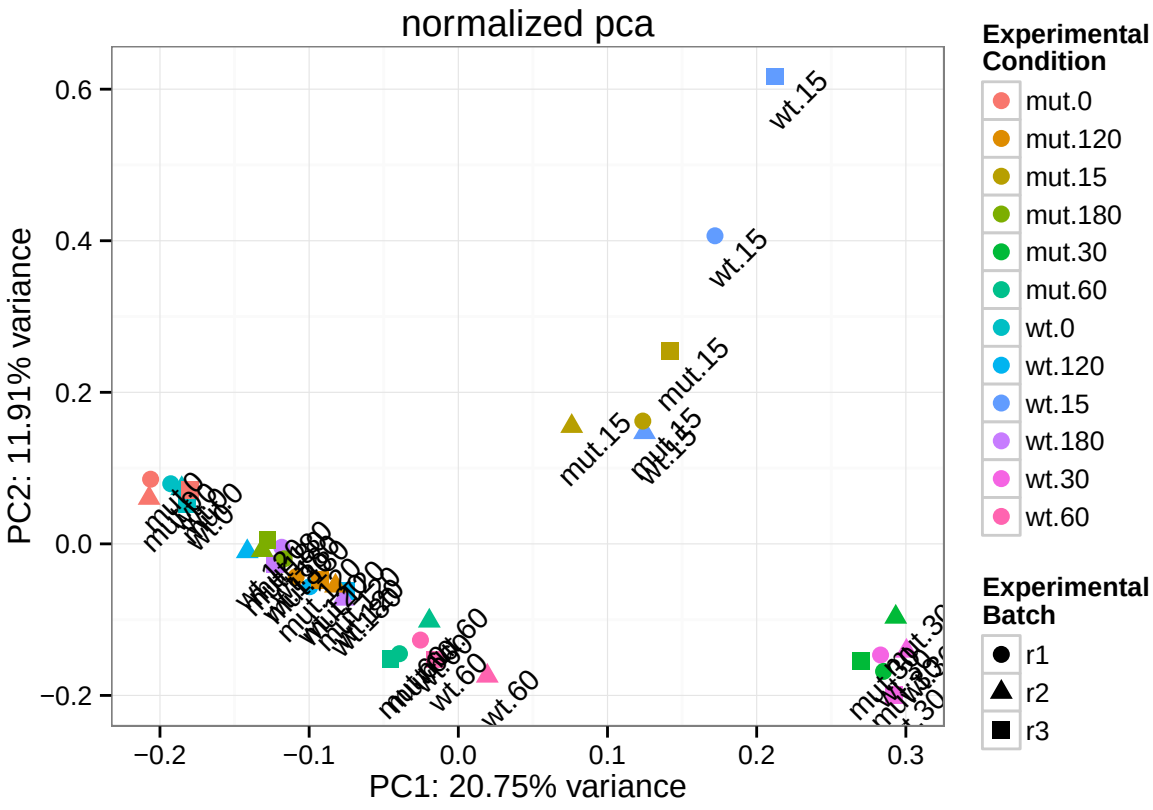
```
## Unsurprisingly, the raw data doesn't cluster well at all...
fis_rawpca = hpgl_pca(expt=fission_expt, labels=fission_expt$condition)
fis_rawpca$plot
```



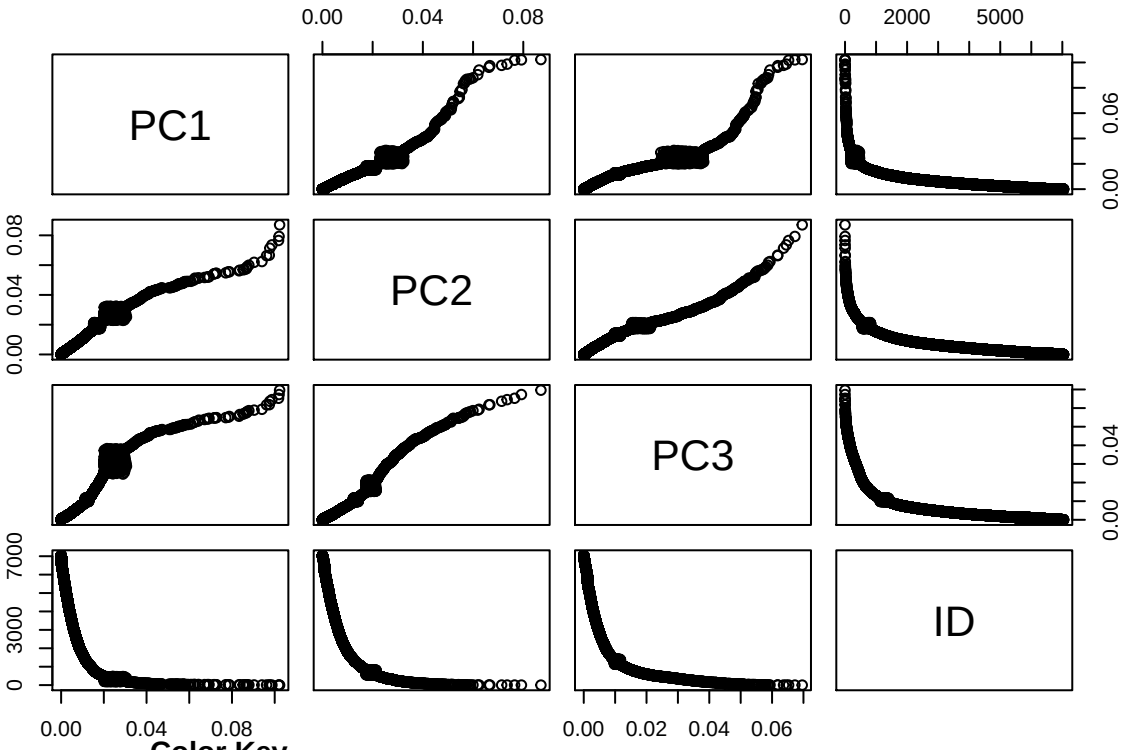

```
## So, normalize the data
norm_expt = normalize_expt(fission_expt, transform="log2", norm="quant", convert="cpm")
```

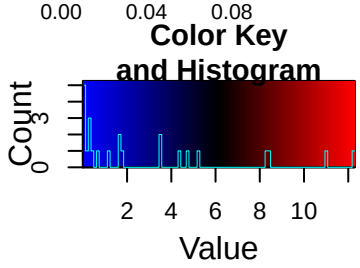
```
## [1] "This function will replace the expt$expressionset slot with the log2(quant(cpm))'d data."
## [1] "It saves the current data into a slot named: expt$backup_expressionset"
## [1] "It will also save copies of each step along the way in expt$normalized with the corresponding li
## [1] "Keep the libsizes in mind when invoking limma.  The appropriate libsize is the non-log(cpm(norma
## [1] "This is most likely kept in the slot called: 'new_expt$normalized$normalized_counts$libsize' whi
## [1] "Filter low is false, this should likely be set to something, good choices include cbcb, kofa, pc
## [1] "Not correcting the count-data for batch effects.  If batch is included in EdgerR/limma's model,
```

```
## And try the pca again
fis_normpca = hpgl_pca(expt=norm_expt, labels=norm_expt$condition, title="normalized pca")
fis_normpca$plot
```

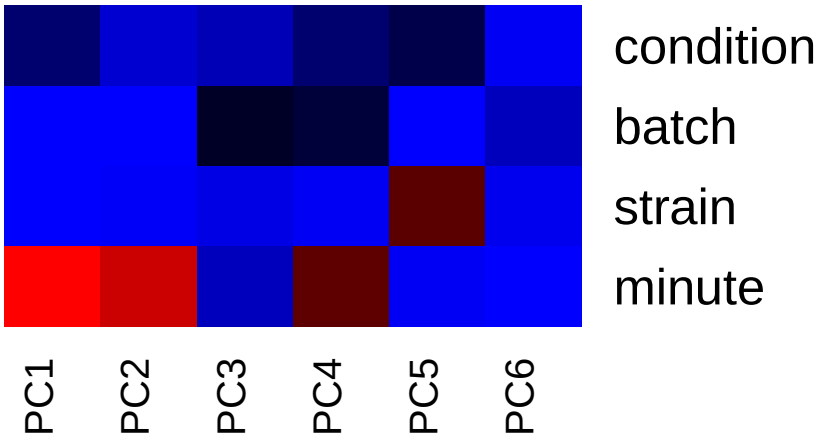


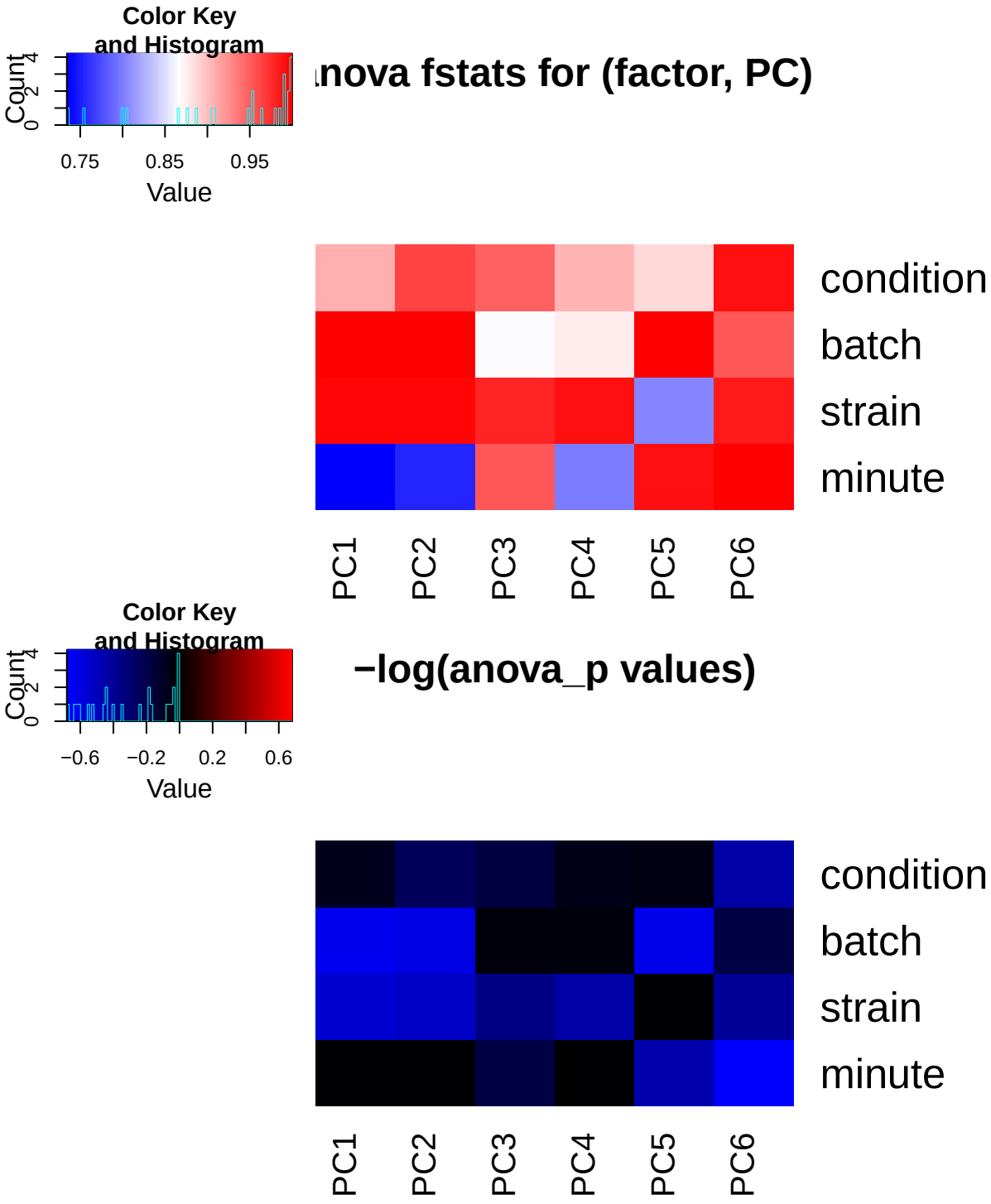

```
## ok, that caused the 0, 60, 15, and 30 minute samples to cluster nicely
## the 120 and 180 minute samples are still a bit tight

## pca_information provides some more information about the call to
## fast.svd that went into making the pca plot
fis_info = pca_information(df=exprs(norm_expt$expressionset), design=full_design, factors=c("condition"

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





nova fstats for (factor, PC)






```
## 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
```

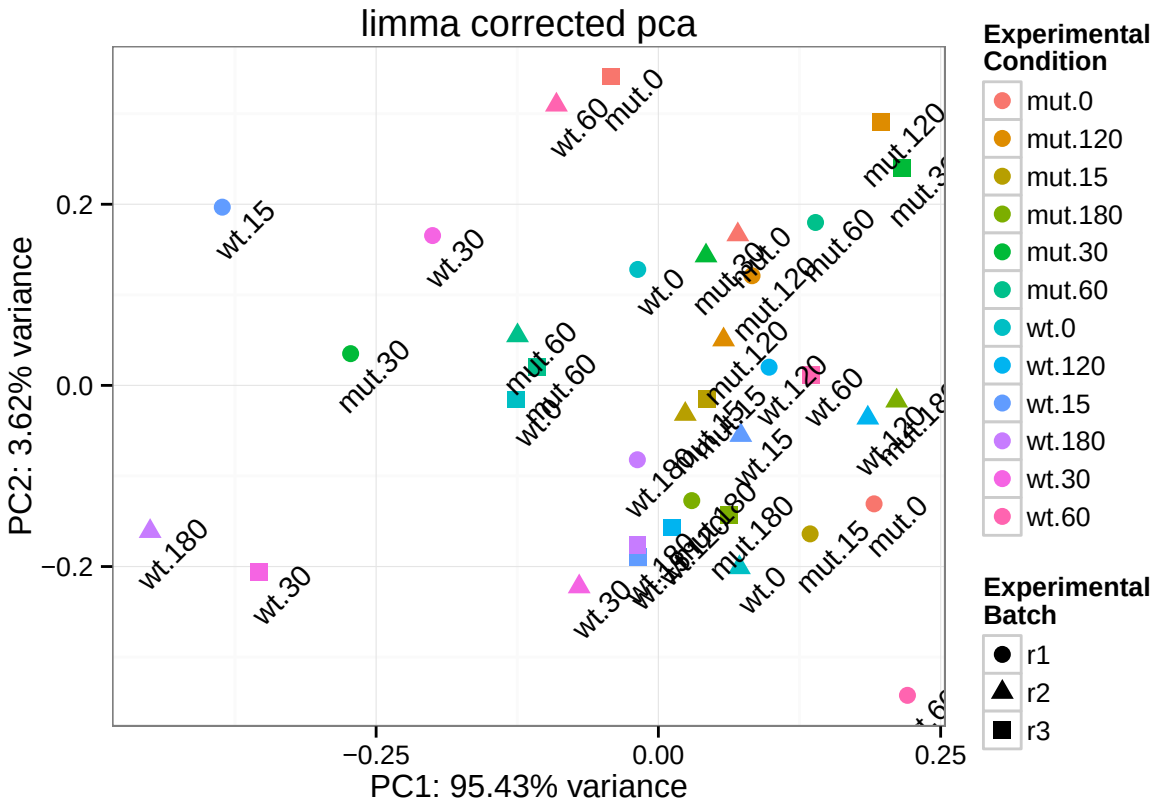
```
## And p-values to lend some credence(or not to those assertions)
fis_info$anova_p
```

```
##                   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")
```

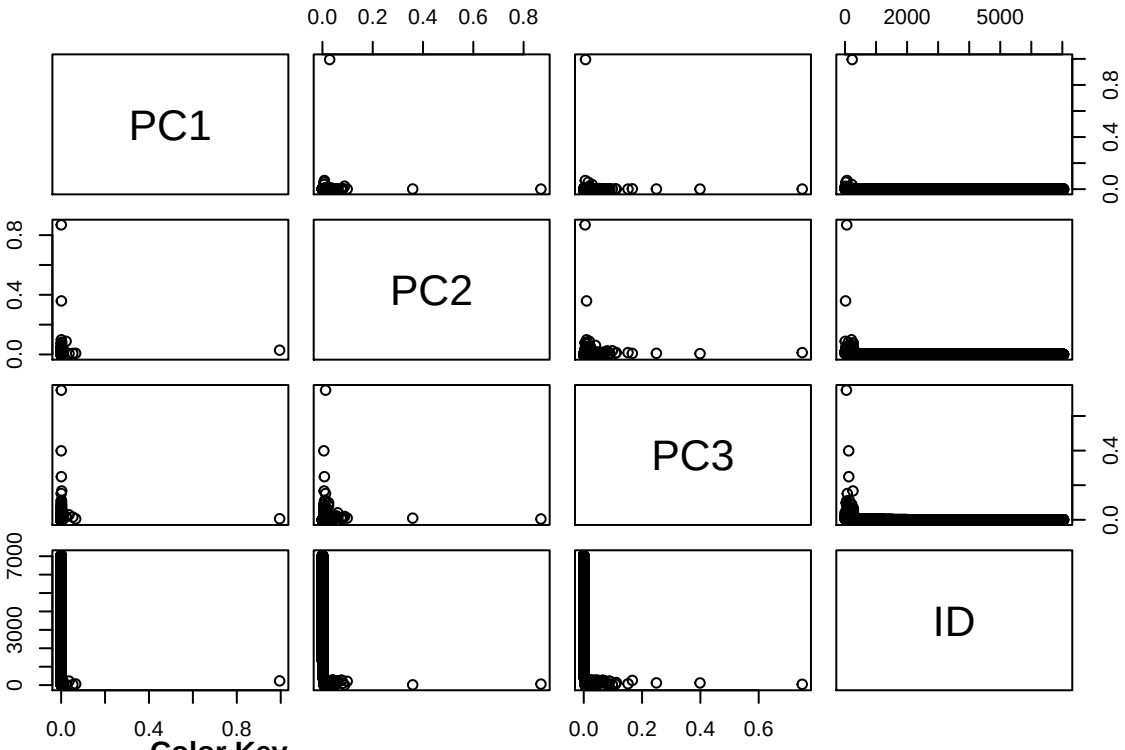
```
## [1] "This function will replace the expt$expressionset slot with the raw(raw(raw))'d data."
## [1] "It saves the current data into a slot named: expt$backup_expressionset"
## [1] "It will also save copies of each step along the way in expt$normalized with the corresponding li
## [1] "Keep the libsizes in mind when invoking limma.  The appropriate libsize is the non-log(cpm(norma
## [1] "This is most likely kept in the slot called: 'new_expt$normalized$normalized_counts$libsize' whi
## [1] "Filter low is false, this should likely be set to something, good choices include cbcb, kofa, pe
## [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

## Using limma's removeBatchEffect to remove batch effect.
```

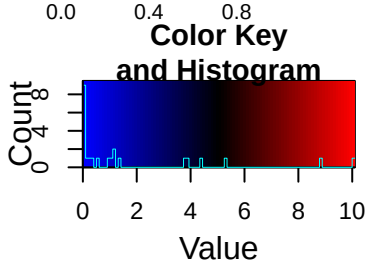
```
fis_batchnormpca = hpgl_pca(expt=batchnorm_expt, labels=norm_expt$condition, title="limma corrected pca
fis_batchnormpca$plot
```



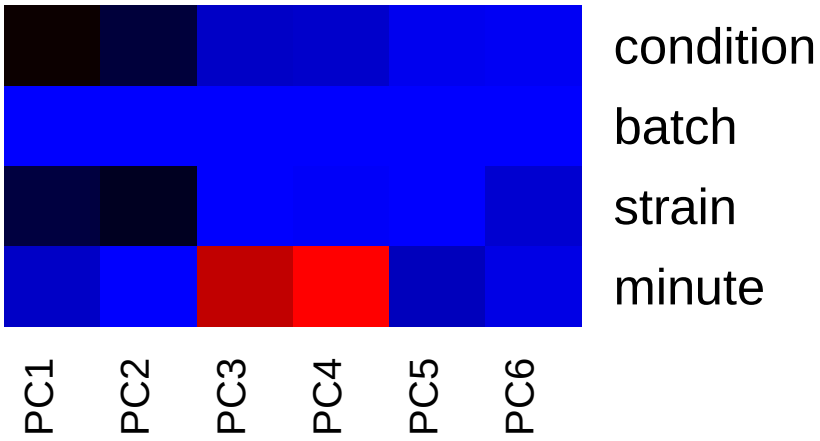

```
test_pca = pca_information(df=exprs(batchnorm_expt$expressionset), design=full_design, factors=c("condi
```

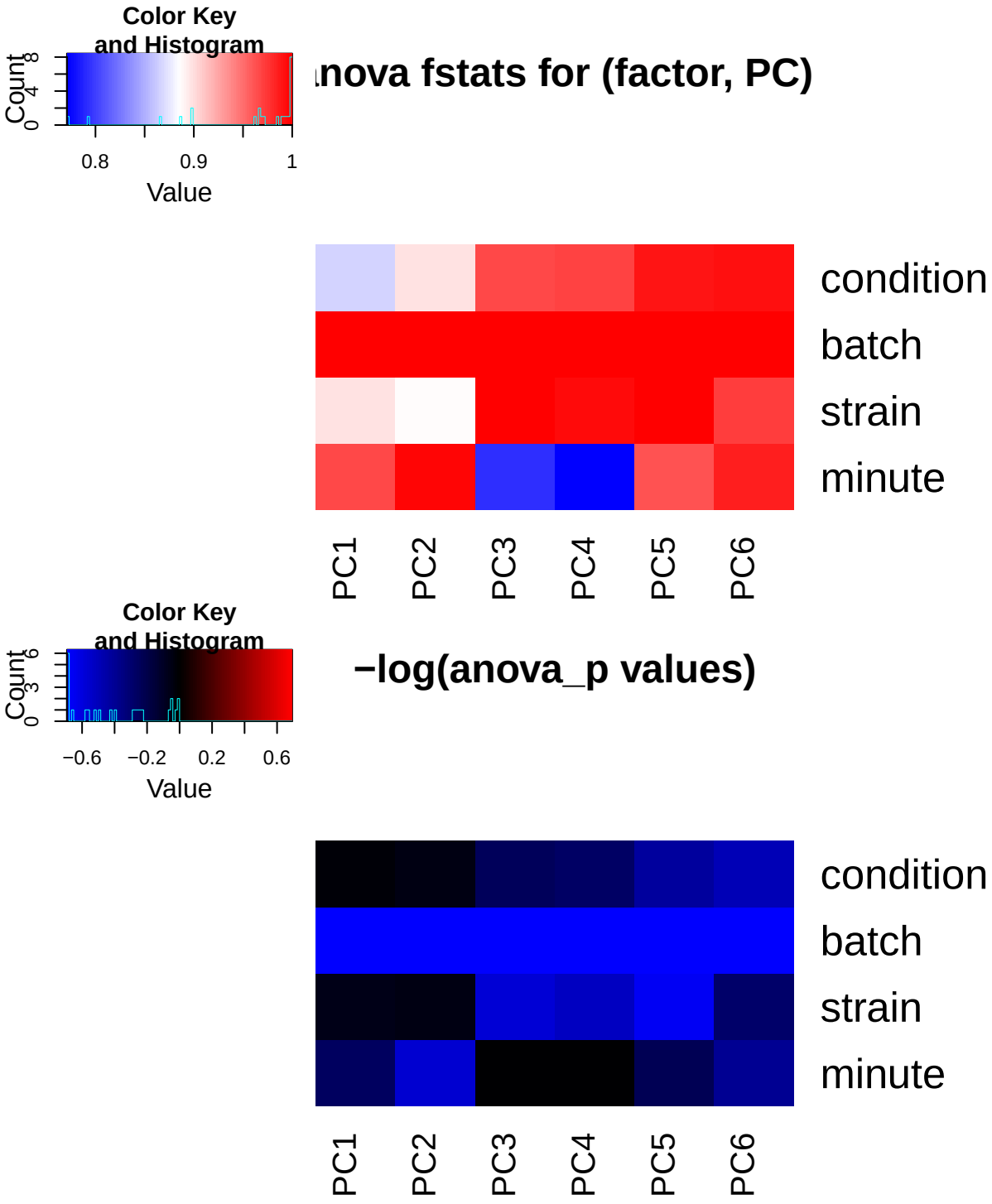
```
## The more shallow the curves in these plots, the more genes responsible for this principle component.
```





nova fstats for (factor, PC)





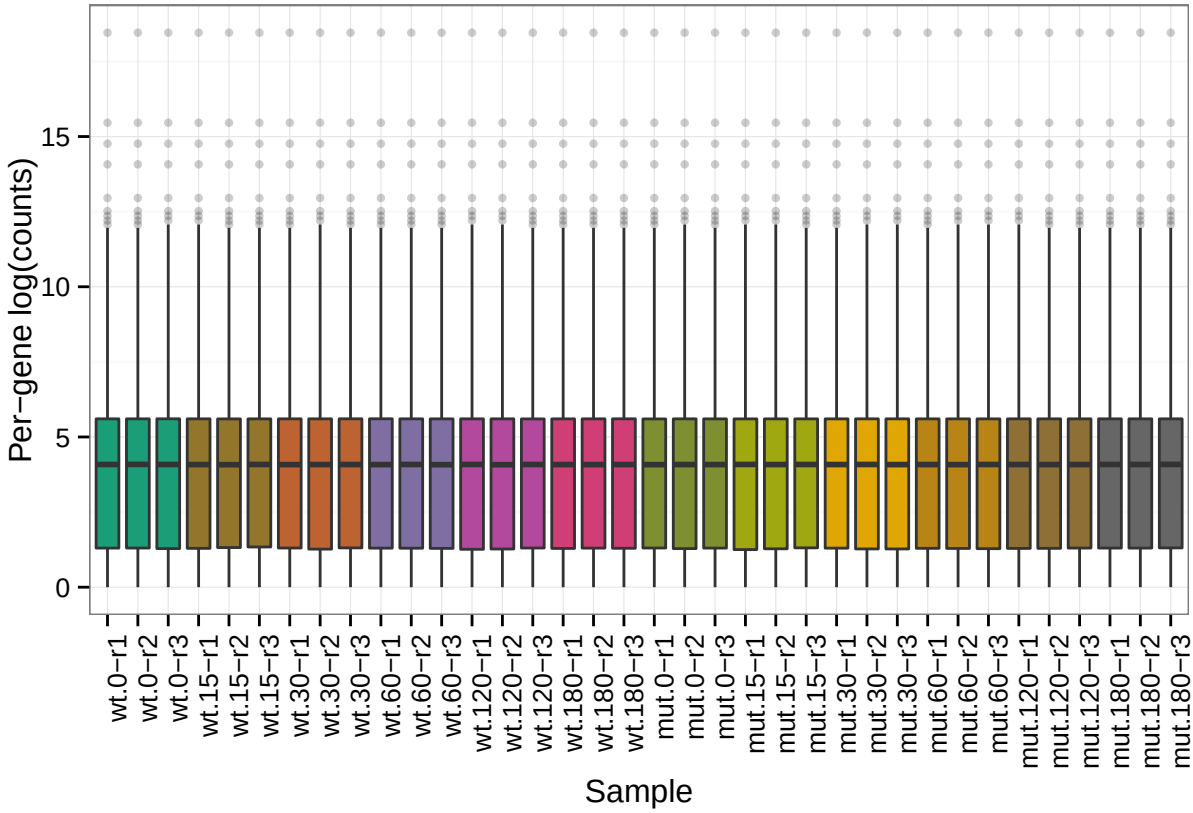

```
##combatnorm_expt = normalize_expt(fission_expt, batch="sva")
##fis_combatnormpca = hpgl_pca(expt=combatnorm_expt, labels=norm_expt$condition, title="sva corrected p
##fis_combatnormpca
##test_pca = pca_information(df=exprs(combatnorm_expt$expressionset), design=full_design, factors=c("co
```

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:

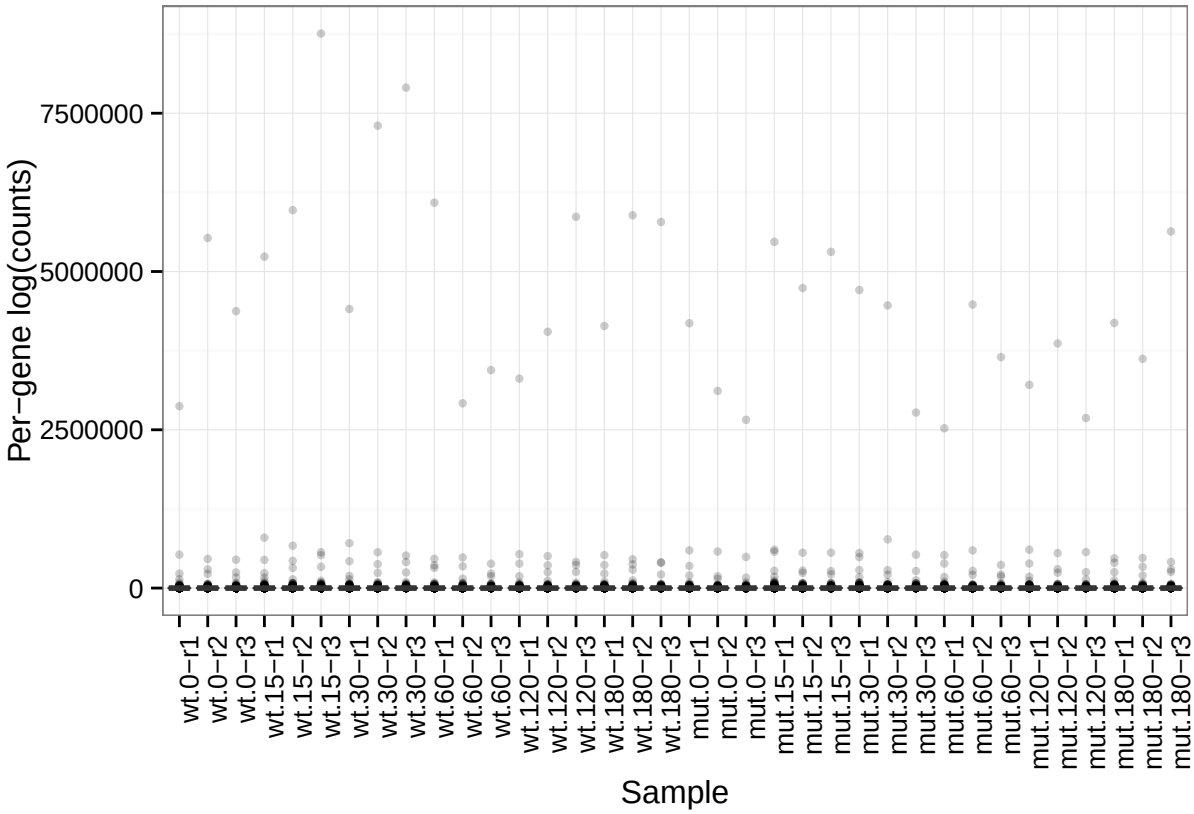
```
hpgl_boxplot(expt=norm_expt)
```




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

```
## [1] "This function will replace the expt$expressionset slot with the raw(sf(raw))'d data."
## [1] "It saves the current data into a slot named: expt$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, pe
## [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] "Not correcting the count-data for batch effects.  If batch is included in EdgerR/limma's model,
```

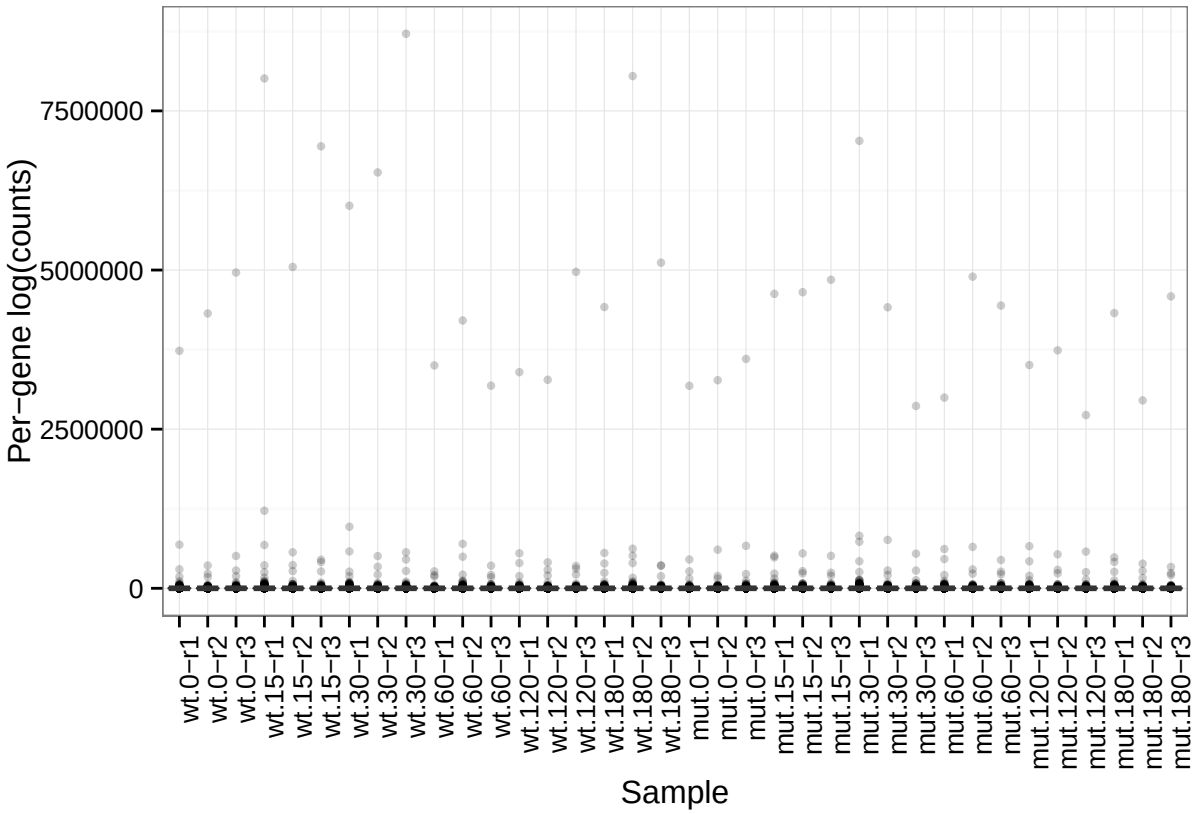
```
hpgl_boxplot(expt=sf_expt)
```




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

```
## [1] "This function will replace the expt$expressionset slot with the raw(tmm(raw))'d data."
## [1] "It saves the current data into a slot named: expt$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] "Not correcting the count-data for batch effects.  If batch is included in EdgerR/limma's model,
```

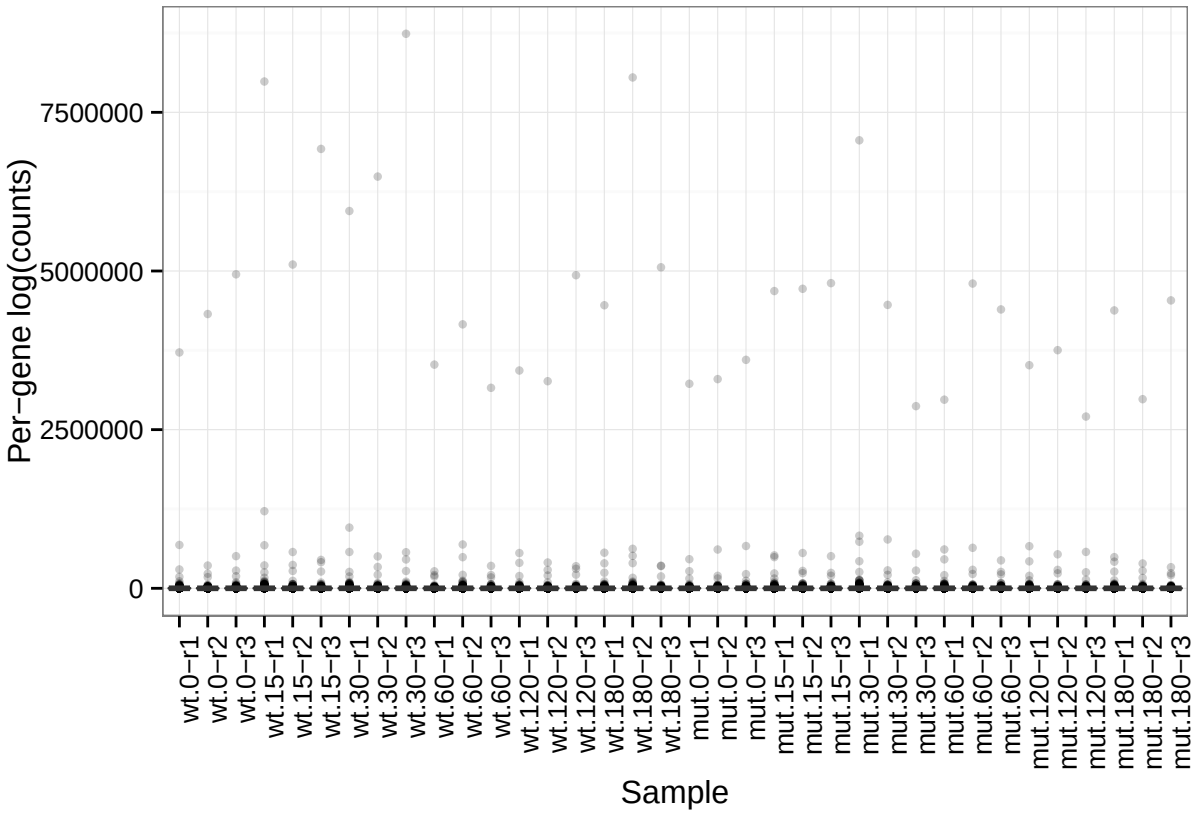
```
hpgl_boxplot(expt=tm_expt)
```




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

```
## [1] "This function will replace the expt$expressionset slot with the raw(rle(raw))'d data."
## [1] "It saves the current data into a slot named: expt$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, pe
## [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] "Not correcting the count-data for batch effects.  If batch is included in EdgerR/limma's model,
```

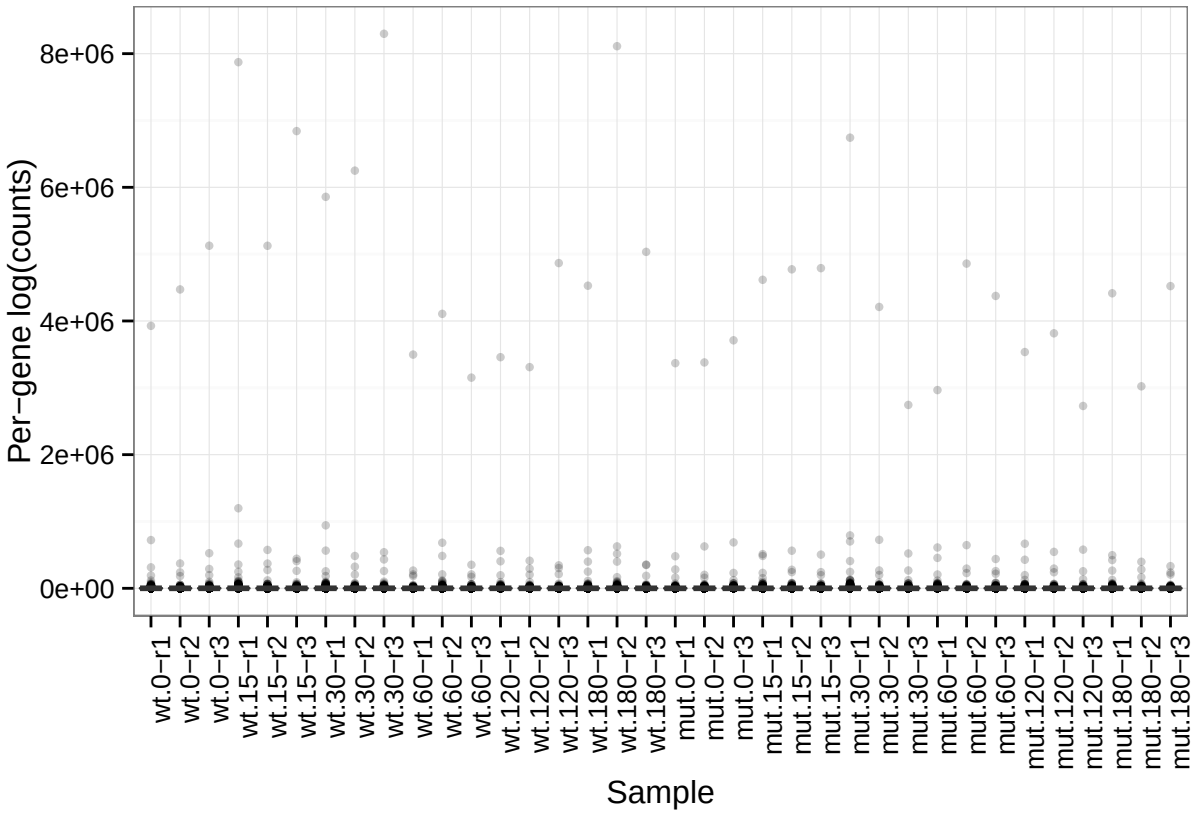
```
hpgl_boxplot(expt=rle_expt)
```




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

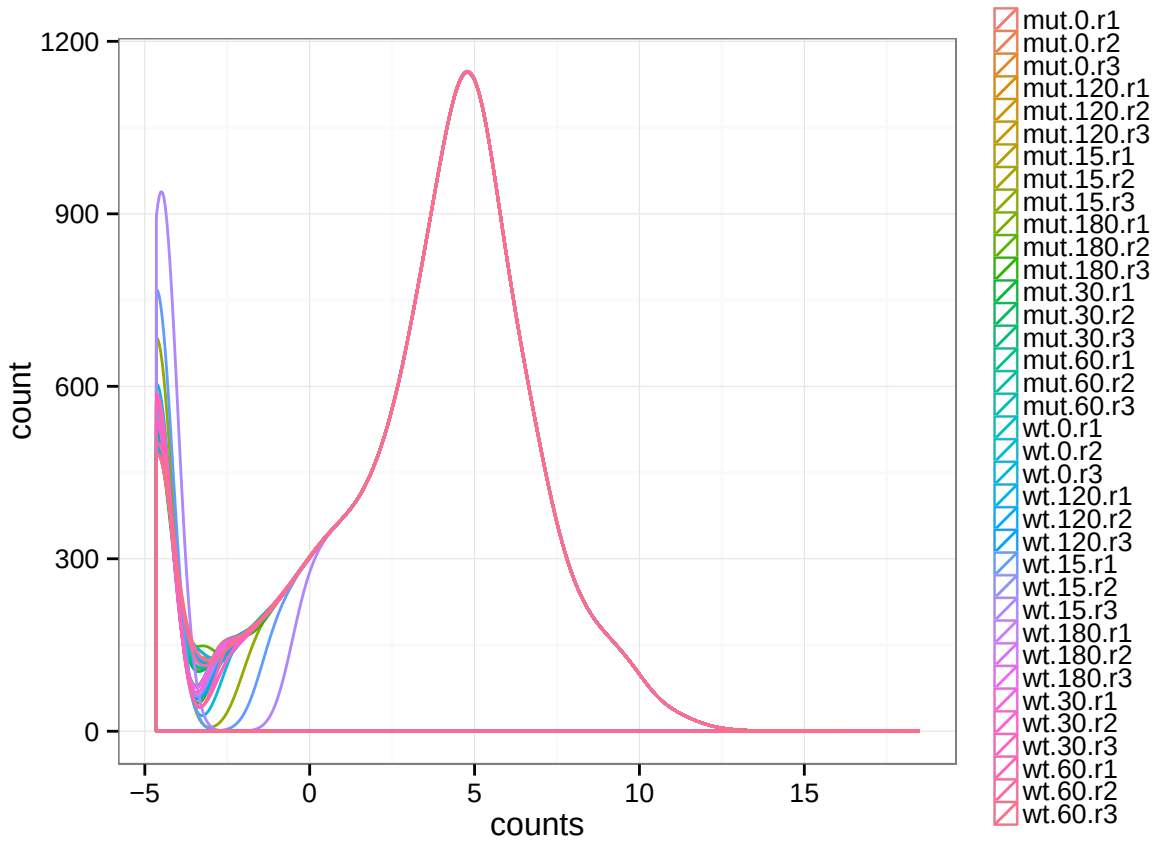
```
## [1] "This function will replace the expt$expressionset slot with the raw(upperquartile(raw))'d data.'
## [1] "It saves the current data into a slot named: expt$backup_expressionset"
## [1] "It will also save copies of each step along the way in expt$normalized with the corresponding li
## [1] "Keep the libsizes in mind when invoking limma.  The appropriate libsize is the non-log(cpm(norma
## [1] "This is most likely kept in the slot called: 'new_expt$normalized$normalized_counts$libsize' whi
## [1] "Filter low is false, this should likely be set to something, good choices include cbcb, kofa, pe
## [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] "Not correcting the count-data for batch effects.  If batch is included in EdgerR/limma's model,
```

```
hpgl_boxplot(expt=up_expt)
```



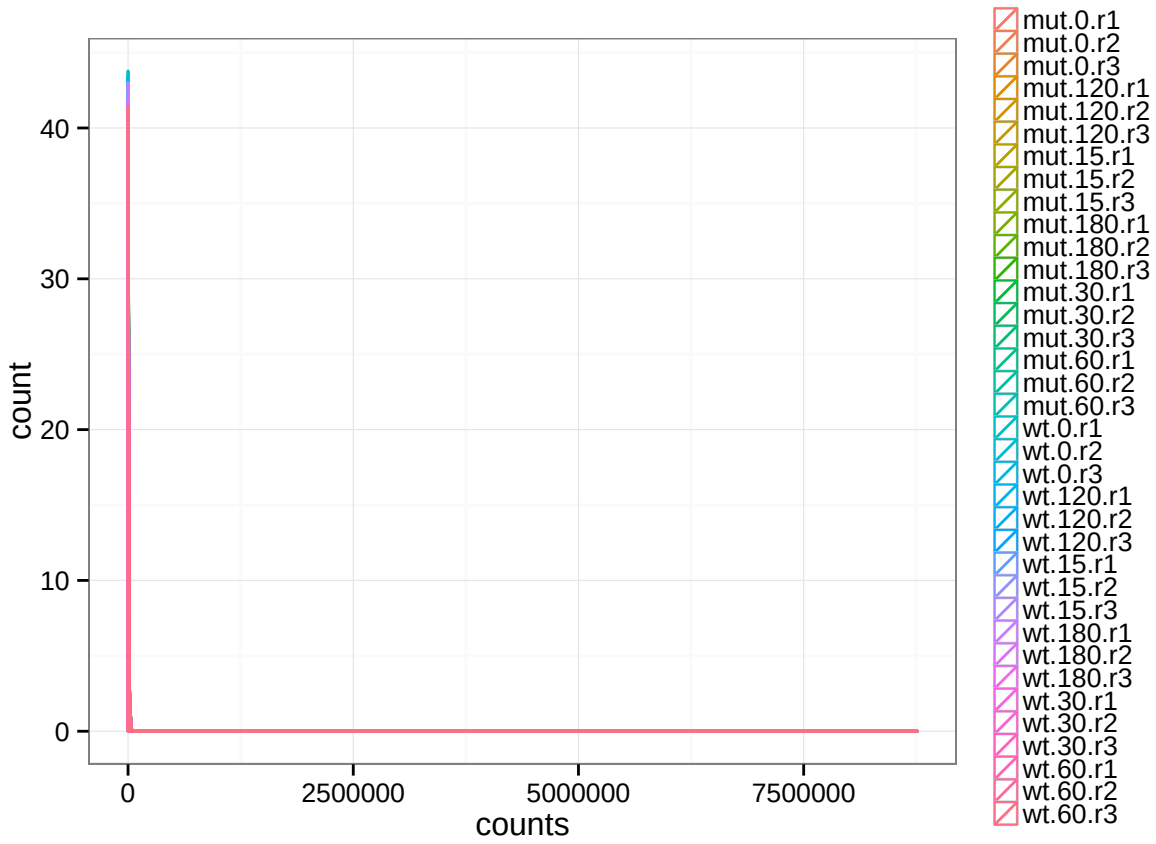

```
hpgl_density_plot(expt=norm_expt)
```

```
## This plot looks neat if you do position='fill' or position='stack'
```



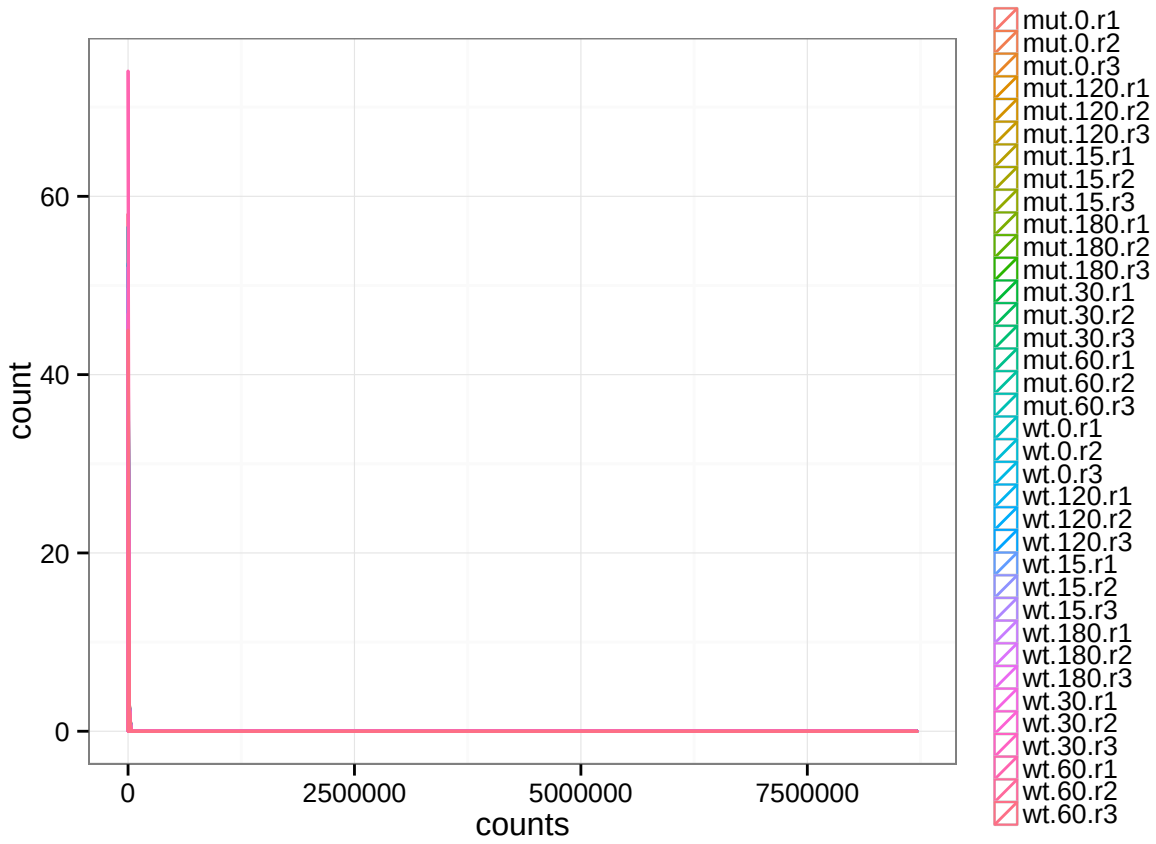

```
hpgl_density_plot(expt=sf_expt)
```

```
## This plot looks neat if you do position='fill' or position='stack'
```




```
hpgl_density_plot(expt=tm_expt)
```

```
## This plot looks neat if you do position='fill' or position='stack'
```

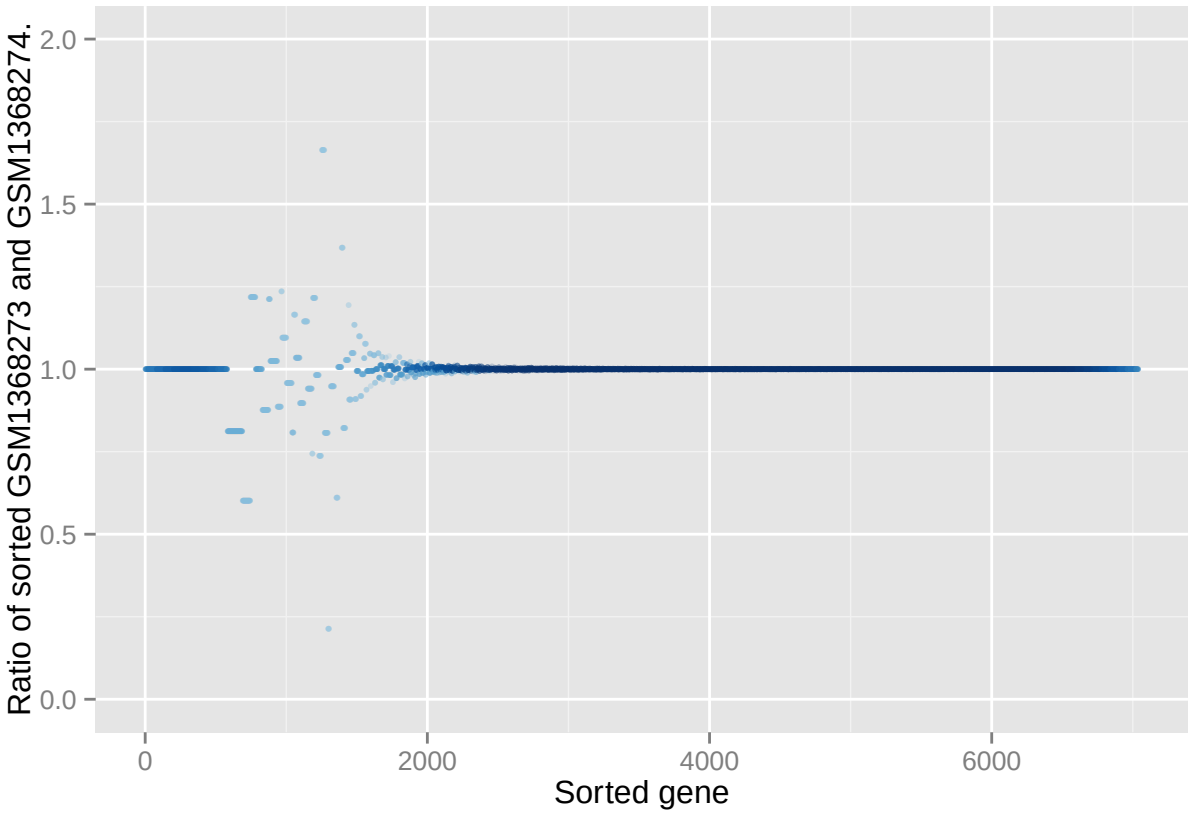



```
compare_12 = hpgl_qq_plot(exprs(norm_expt$expressionset), x=1, y=2)
```

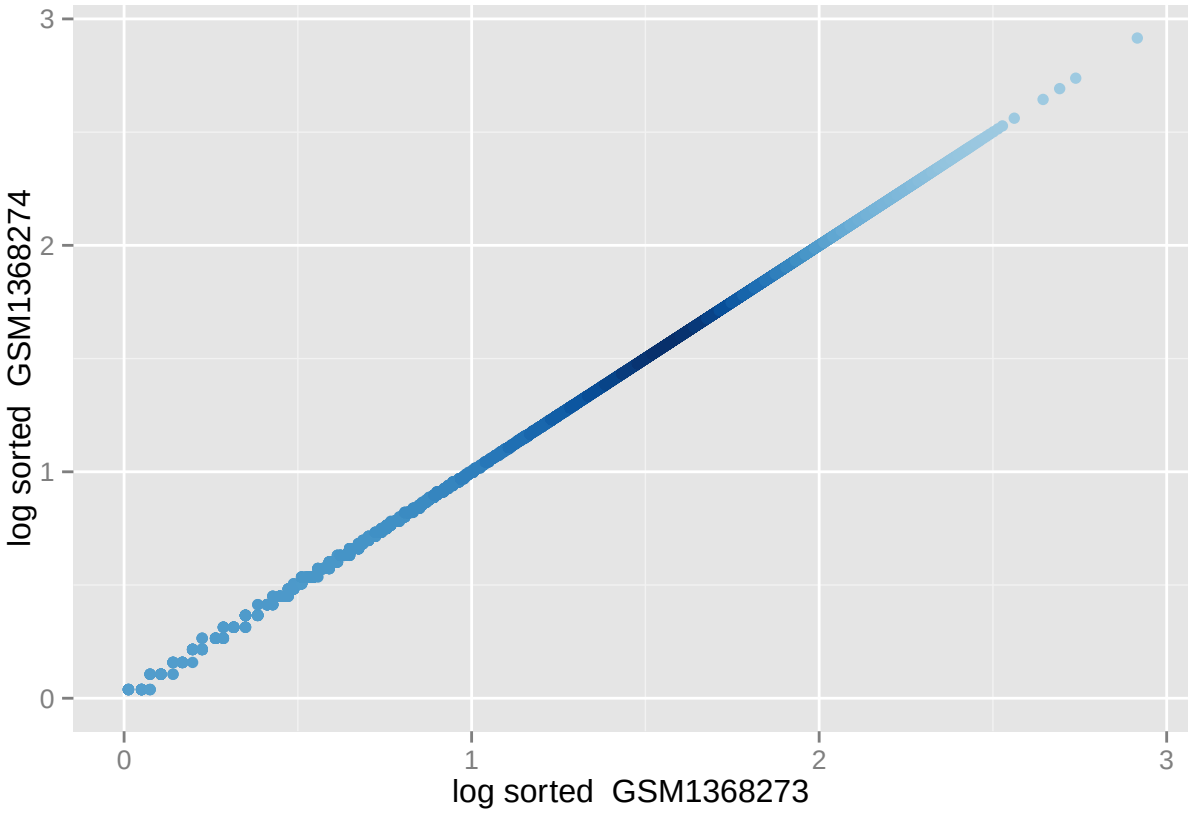
```
## Warning in log(sorted_x): NaNs produced
```

```
## Warning in log(sorted_y): NaNs produced
```

```
compare_12$ratio
```



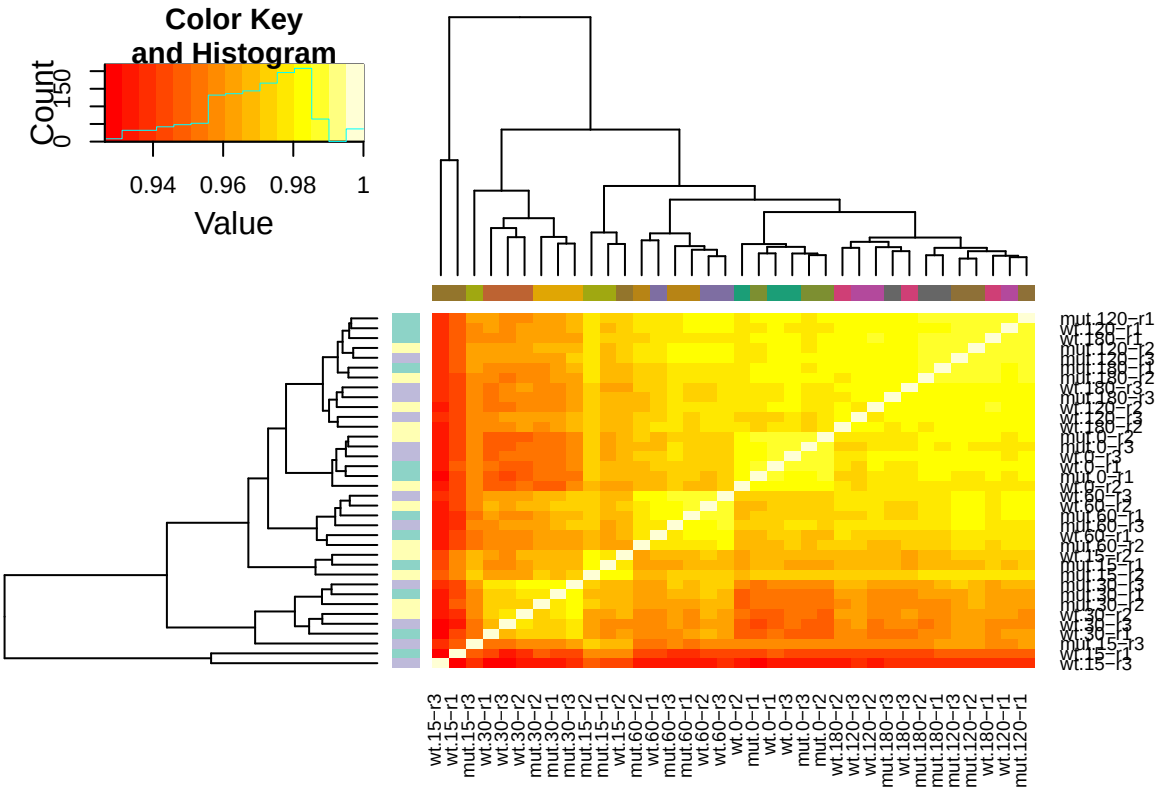

```
compare_12$log
```



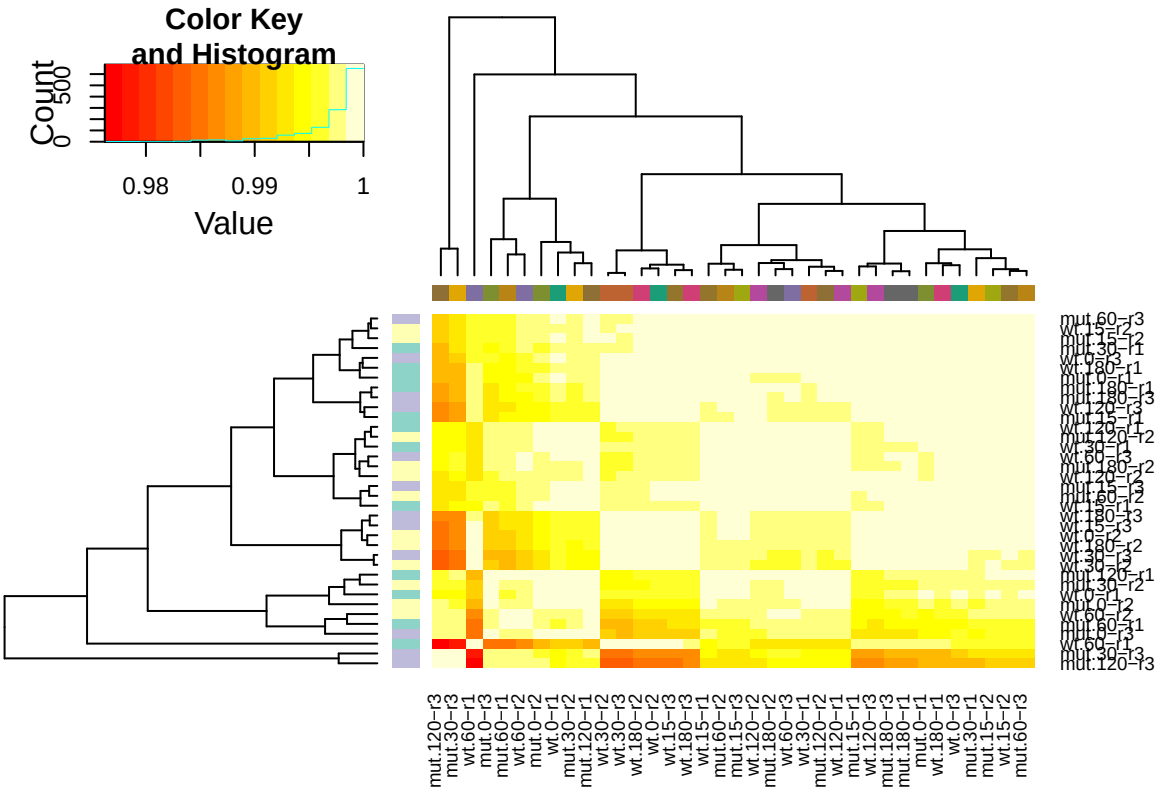
See how they cluster

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

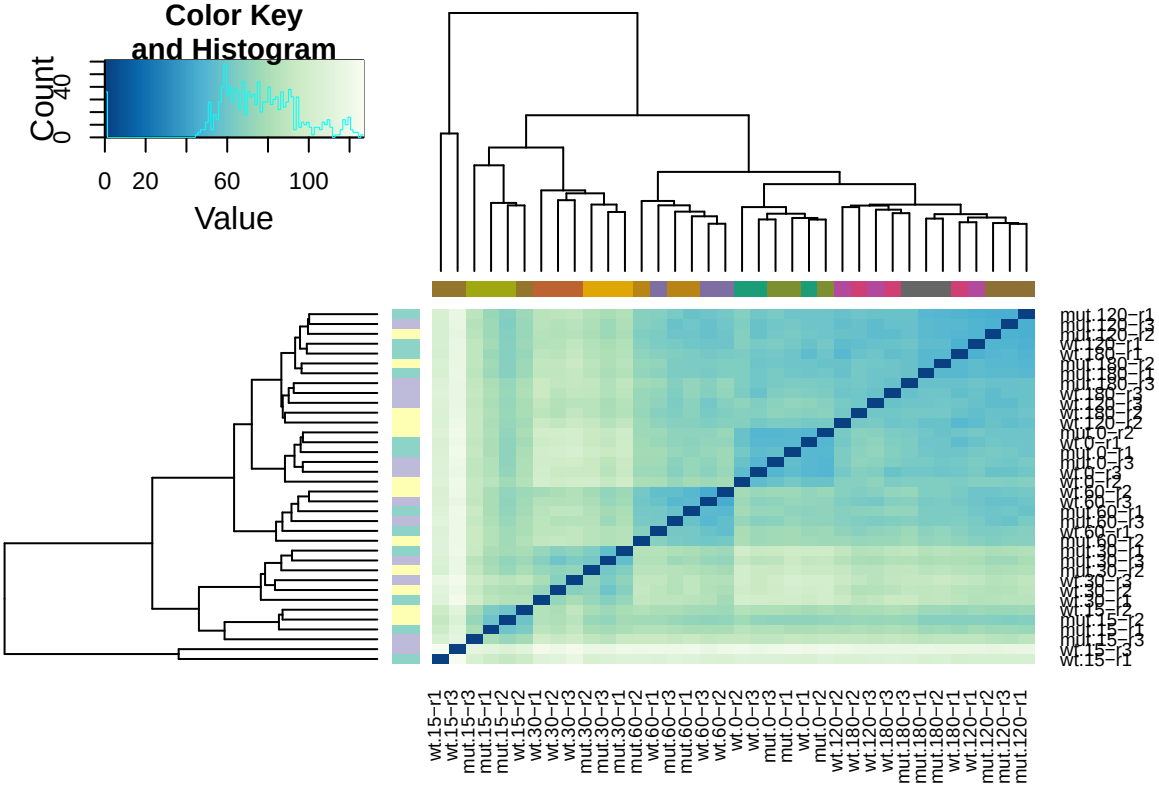
```
hpgl_corheat(expt=norm_expt)
```



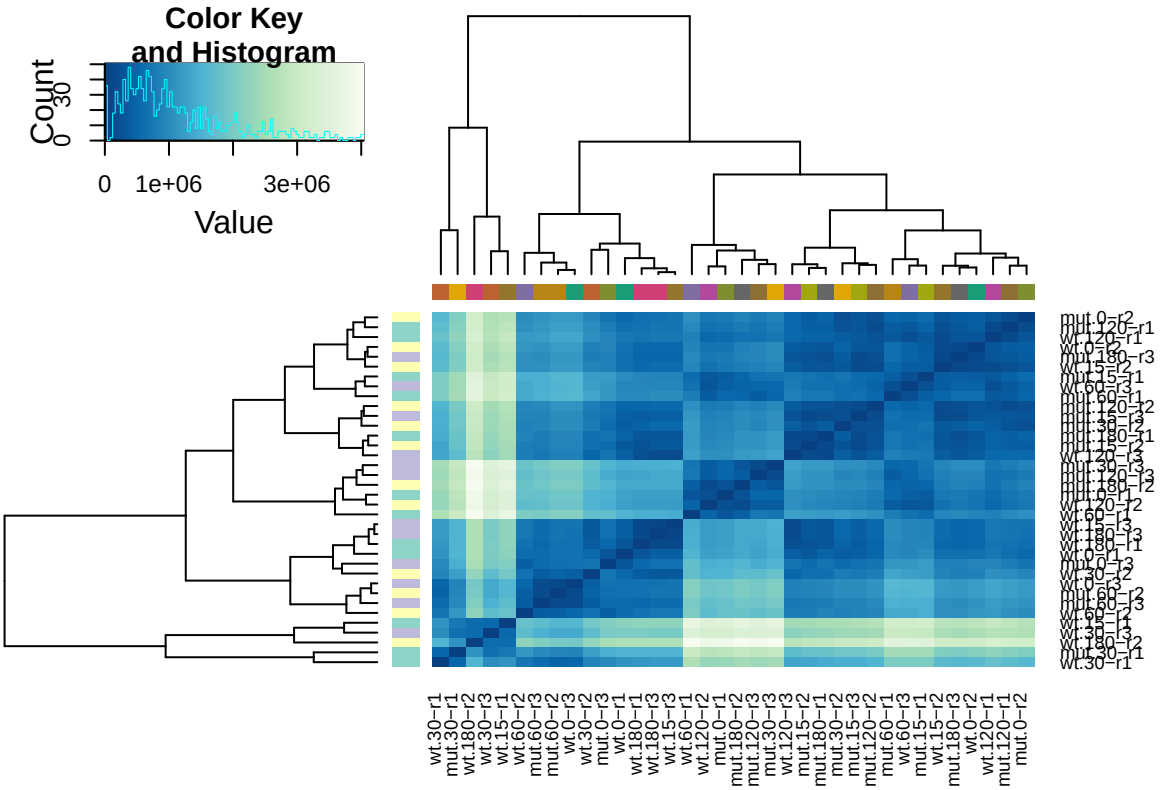

```
hpgl_corheat(expt=batchnorm_expt)
```




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



```
hpgl_disheat(expt=batchnorm_expt)
```



Some simple differential expression analyses

```
all_pairwise = limma_pairwise(expt=norm_expt, conditions=norm_expt$conditions, batches=norm_expt$batches
```

```
## libsize was not specified, this parameter has profound effects on limma's result.
## Using the libsize from expt$best_libsize.

## [1] "As a reference, the identity is: mut.0 = mut.0,"
## [1] "As a reference, the identity is: mut.120 = mut.120,"
## [1] "As a reference, the identity is: mut.15 = mut.15,"
## [1] "As a reference, the identity is: mut.180 = mut.180,"
## [1] "As a reference, the identity is: mut.30 = mut.30,"
## [1] "As a reference, the identity is: mut.60 = mut.60,"
## [1] "As a reference, the identity is: wt.0 = wt.0,"
## [1] "As a reference, the identity is: wt.120 = wt.120,"
## [1] "As a reference, the identity is: wt.15 = wt.15,"
## [1] "As a reference, the identity is: wt.180 = wt.180,"
## [1] "As a reference, the identity is: wt.30 = wt.30,"
## [1] "As a reference, the identity is: wt.60 = wt.60,"

## 1: Printing table: mut.0
## 2: Printing table: mut.120
```

```
## 3: Printing table: mut.15
## 4: Printing table: mut.180
## 5: Printing table: mut.30
## 6: Printing table: mut.60
## 7: Printing table: wt.0
## 8: Printing table: wt.120
## 9: Printing table: wt.15
## 10: Printing table: wt.180
## 11: Printing table: wt.30
## 12: Printing table: wt.60
## 13: Printing table: mut.120_minus_mut.0
## 14: Printing table: mut.15_minus_mut.0
## 15: Printing table: mut.180_minus_mut.0
## 16: Printing table: mut.30_minus_mut.0
## 17: Printing table: mut.60_minus_mut.0
## 18: Printing table: wt.0_minus_mut.0
## 19: Printing table: wt.120_minus_mut.0
## 20: Printing table: wt.15_minus_mut.0
## 21: Printing table: wt.180_minus_mut.0
## 22: Printing table: wt.30_minus_mut.0
## 23: Printing table: wt.60_minus_mut.0
## 24: Printing table: mut.15_minus_mut.120
## 25: Printing table: mut.180_minus_mut.120
## 26: Printing table: mut.30_minus_mut.120
## 27: Printing table: mut.60_minus_mut.120
## 28: Printing table: wt.0_minus_mut.120
## 29: Printing table: wt.120_minus_mut.120
## 30: Printing table: wt.15_minus_mut.120
## 31: Printing table: wt.180_minus_mut.120
## 32: Printing table: wt.30_minus_mut.120
## 33: Printing table: wt.60_minus_mut.120
## 34: Printing table: mut.180_minus_mut.15
## 35: Printing table: mut.30_minus_mut.15
## 36: Printing table: mut.60_minus_mut.15
## 37: Printing table: wt.0_minus_mut.15
## 38: Printing table: wt.120_minus_mut.15
## 39: Printing table: wt.15_minus_mut.15
## 40: Printing table: wt.180_minus_mut.15
## 41: Printing table: wt.30_minus_mut.15
## 42: Printing table: wt.60_minus_mut.15
## 43: Printing table: mut.30_minus_mut.180
## 44: Printing table: mut.60_minus_mut.180
## 45: Printing table: wt.0_minus_mut.180
## 46: Printing table: wt.120_minus_mut.180
## 47: Printing table: wt.15_minus_mut.180
## 48: Printing table: wt.180_minus_mut.180
## 49: Printing table: wt.30_minus_mut.180
## 50: Printing table: wt.60_minus_mut.180
## 51: Printing table: mut.60_minus_mut.30
## 52: Printing table: wt.0_minus_mut.30
## 53: Printing table: wt.120_minus_mut.30
## 54: Printing table: wt.15_minus_mut.30
## 55: Printing table: wt.180_minus_mut.30
## 56: Printing table: wt.30_minus_mut.30
```

```
## 57: Printing table: wt.60_minus_mut.30
## 58: Printing table: wt.0_minus_mut.60
## 59: Printing table: wt.120_minus_mut.60
## 60: Printing table: wt.15_minus_mut.60
## 61: Printing table: wt.180_minus_mut.60
## 62: Printing table: wt.30_minus_mut.60
## 63: Printing table: wt.60_minus_mut.60
## 64: Printing table: wt.120_minus_wt.0
## 65: Printing table: wt.15_minus_wt.0
## 66: Printing table: wt.180_minus_wt.0
## 67: Printing table: wt.30_minus_wt.0
## 68: Printing table: wt.60_minus_wt.0
## 69: Printing table: wt.15_minus_wt.120
## 70: Printing table: wt.180_minus_wt.120
## 71: Printing table: wt.30_minus_wt.120
## 72: Printing table: wt.60_minus_wt.120
## 73: Printing table: wt.180_minus_wt.15
## 74: Printing table: wt.30_minus_wt.15
## 75: Printing table: wt.60_minus_wt.15
## 76: Printing table: wt.30_minus_wt.180
## 77: Printing table: wt.60_minus_wt.180
## 78: Printing table: wt.60_minus_wt.30
```

```
## I have the following elements in this list:
### conditions_table     :  how many replicates are there of each condition
### batches_table        :  how many replicates are there of each batch
### conditions           :  a factor of all the conditions
### batches              :  a factor of all the batches
### model                :  the model of the data used for voom etc.
### fit                  :  the result of lmfit()
### voom_result          :  the result of voom()
### voom_design          :  the design matrix fed to voom()
### identities           :  the strings fed to makeContrasts() which describe each sample alone
### all_pairwise         :  the strings describing all the subtractions fed to makeContrasts()
### contrast_string      :  the entire string fed to makeContrasts() including the design etc.
### pairwise_fits        :  the result of contrasts.fit()
### pairwise_comparisons :  the result from eBayes()
### limma_result         :  a list of toptable()s corresponding to each pairwise comparison

names(all_pairwise$limma_result)
```

```
## NULL
```

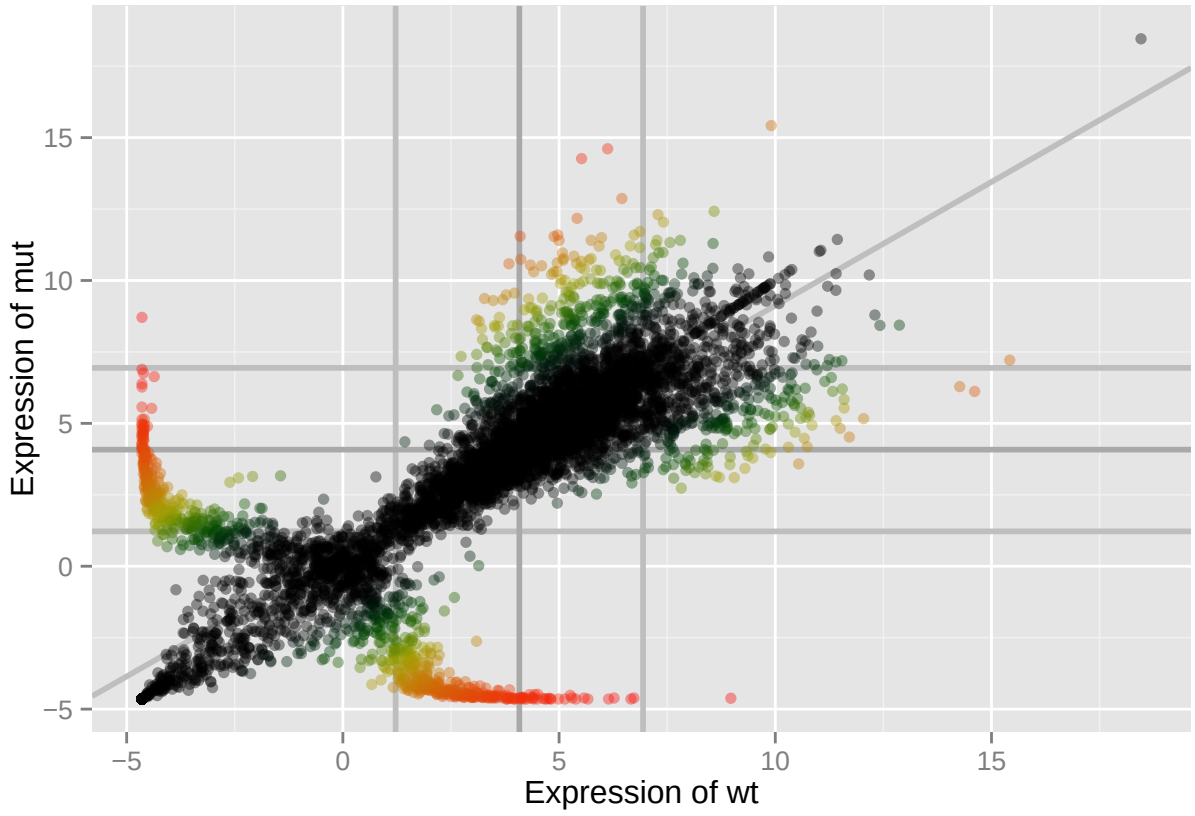
```
summary(all_pairwise$limma_result$wt.120_minus_mut.120)
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

```
wt_120 = all_pairwise$all_tables$wt.120
mut_120 = all_pairwise$all_tables$mut.120

scatter_wt_mut = hpgl_linear_scatter(df=data.frame(wt=wt_120$AveExpr, mut=mut_120$AveExpr))
```

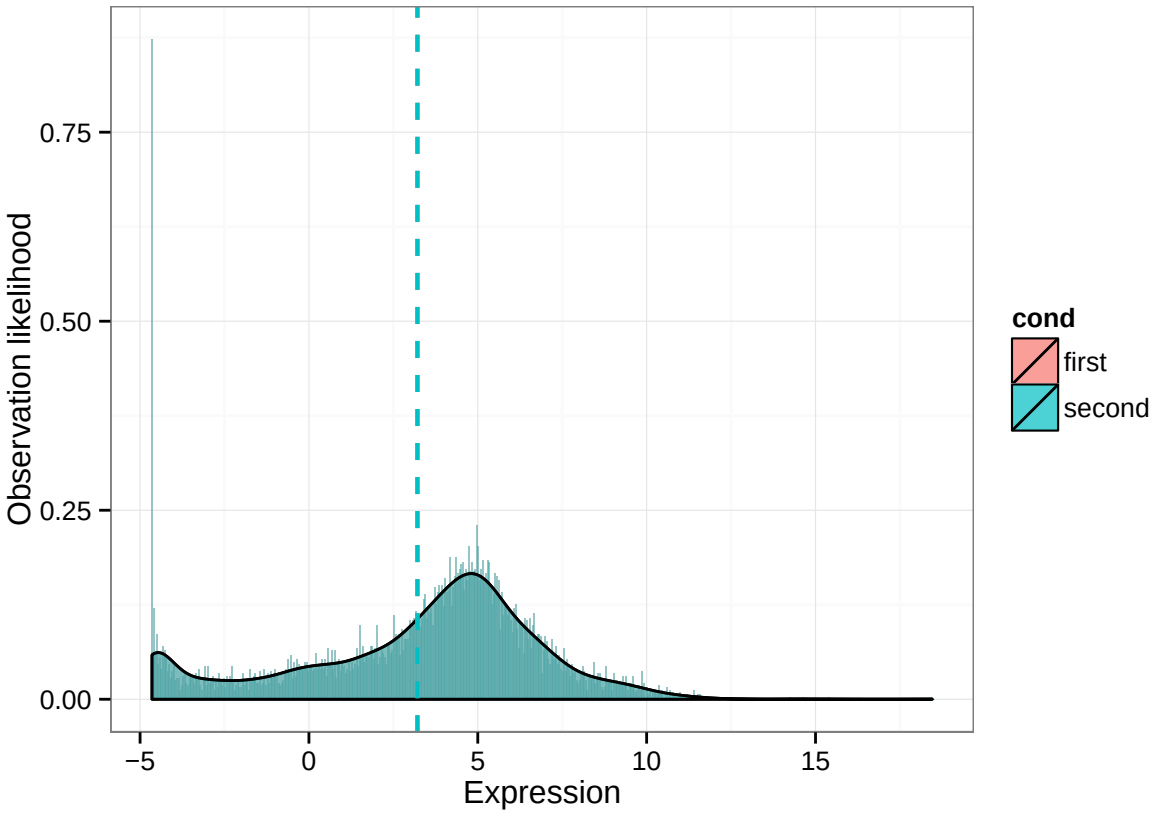
```
## No binwidth nor bins provided, setting it to 0.0462073143417922 in order to have 500 bins.
```

```
scatter_wt_mut$scatter
```




```
scatter_wt_mut$both_histogram
```

```
## $plot
```




```
##
## $data_summary
##      first             second
##  Min.   :-4.646   Min.   :-4.646
##  1st Qu.: 1.259   1st Qu.: 1.259
##  Median : 4.081   Median : 4.081
##  Mean   : 3.213   Mean   : 3.213
##  3rd Qu.: 5.571   3rd Qu.: 5.571
##  Max.   :18.458   Max.   :18.458
##
## $uncor_t
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  play_all$expression and play_all$cond
##
##        first
## second 1
##
## 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 1
##
## P value adjustment method: bonferroni
```