# hpgltools usage

*atb abelew@gmail.com*

*2016-05-18*

## Using hpgltools for fun and profit!

hpgltools was written to make working with high-throughput data analyses easier. These analyses generally fall into a few stages:

1. Data visualization and outlier/batch evaluation
2. Differential expression analyses

a. Visualization and export of these results

3. Gene ontology/KEGG analyses

a. Visualization and export of these results

Before any of these tasks may be performed, the data must be loaded into memory. hpgltools attempts to make this easier with create_expt() and subset_expt().

## Loading (meta)data and annotations

The following examples will use a real data set from a recent experiment in our lab. The raw data was processed using a mix of trimmomatic, biopieces, bowtie, samtools, and htseq. The final count tables were deposited into the 'preprocessing/count_tables/' tree. The resulting data structure was named 'most_v0M1,' named because it is comprised of count tables with 0 mismatches and 1 randomly-placed multi-match.

The annotation file was mgas_5005.gff.xz residing in 'reference/gff/'.

The count tables and meta-data were loaded through the create_expt() function and the genome annotations were loaded with gff2df().

```
library(hpgltools)
data_file <- system.file("hpgltools.rda", package="hpgltools")
load(data_file, envir=globalenv())

ls()
```

```
##  [1] "all_combined"              "all_comparisons"
##  [3] "annotations"               "basic_comparison"
##  [5] "batchnorm_expt"            "color_hash"
##  [7] "colors"                    "compare_12"
##  [9] "condition_list"            "condition_names"
## [11] "count_dataframe"           "counts"
## [13] "data_file"                 "deseq_comparison"
## [15] "design_colors_list"        "early_late"
## [17] "early_late_thy"            "edger_comparison"
```

```
## [19] "elt"                          "elt_metrics"
## [21] "elt_norm"                     "empty_samples"
## [23] "ensembl_pombe"                "expt"
## [25] "fis_batchnormpca"             "fis_info"
## [27] "fis_libsize"                  "fis_nonzero"
## [29] "fis_normbatchpca"             "fis_normpca"
## [31] "fis_rawpca"                   "fission"
## [33] "fission_data"                 "fission_expt"
## [35] "fun_data"                     "fun_norm"
## [37] "gene_names"                   "gff_from_txdb"
## [39] "goseq_search"                 "hs_annotations"
## [41] "hs_gff"                       "hs_keepers"
## [43] "lengths"                      "limma_comparison"
## [45] "limma_results"                "lp_gff"
## [47] "meta"                         "meta_dataframe"
## [49] "most_v0M1"                    "mut_120"
## [51] "norm_expt"                    "norm_graphs"
## [53] "norm_test"                    "normbatch_expt"
## [55] "num_colors"                   "pca_test"
## [57] "pombe"                        "pombe_filters"
## [59] "pombe_goids"                  "pombe_transcripts"
## [61] "possible_pombe_attributes"    "raw_metrics"
## [63] "rle_expt"                     "sample_definitions"
## [65] "scatter_wt_mut"               "sf_expt"
## [67] "sig_genes"                    "sp_keepers"
## [69] "speciesMap"                   "table"
## [71] "test_pca"                     "tm_expt"
## [73] "tmp_definitions"              "tt"
## [75] "up_expt"                      "updown_genes"
## [77] "written_gff"                  "wt_120"
```

```
## The gff information is in 'annotations'
## The experiment is in most_v0M1
## Here is the meta-data! (well, the first 6 lines anyway).
knitr::kable(head(most_v0M1$definitions))
```

| | sample.id | type | stage | replicate | mutantname | media | exptdate | libdate | batch | condition |
|---|---|---|---|---|---|---|---|---|---|---|
| HPGL0406 | HPGL0406 | WT | EL | 1 | 5448/01/01 | THY | 20130430 | 20140402 | a | wt_el_thy |
| HPGL0407 | HPGL0407 | WT | EL | 2 | 5448/02/01 | THY | 20130430 | 20140402 | a | wt_el_thy |
| HPGL0408 | HPGL0408 | mga | EL | 1 | mga-1 | THY | 20130430 | 20140402 | a | mga1_el_thy |
| HPGL0409 | HPGL0409 | mga | EL | 2 | mga-2 | THY | 20130430 | 20140402 | a | mga1_el_thy |
| HPGL0149 | HPGL0149 | WT | LL | 1 | WT1-1 | THY | 20120924 | 20120926 | b | wt_ll_thy |
| HPGL0150 | HPGL0150 | WT | LL | 2 | WT1-2 | THY | 20120924 | 20120926 | b | wt_ll_thy |

```
## We can re-create the experiment using it:
meta <- most_v0M1$definitions
## This is a terrible thing to do, never do it outside of a demonstration!
meta$batch <- gsub(pattern="b", replacement="a", x=meta$batch)
counts <- Biobase::exprs(most_v0M1$expressionset)
## Originally the meta-data was kept in a file: all_samples.csv
expt <- create_expt(meta_dataframe=meta, count_dataframe=counts)
summary(expt)
```

```
##                         Length Class         Mode
## initial_metadata         5     data.frame    list
## original_expressionset   1     ExpressionSet S4
## expressionset            1     ExpressionSet S4
## design                  23     data.frame    list
## conditions              48     factor        numeric
## batches                 48     factor        numeric
## samplenames             48     -none-        character
## colors                  48     -none-        character
## state                    5     -none-        list
## original_libsize        48     -none-        numeric
## libsize                 48     -none-        numeric
```

The data structure generated by create_expt() is a list containing the following slots:

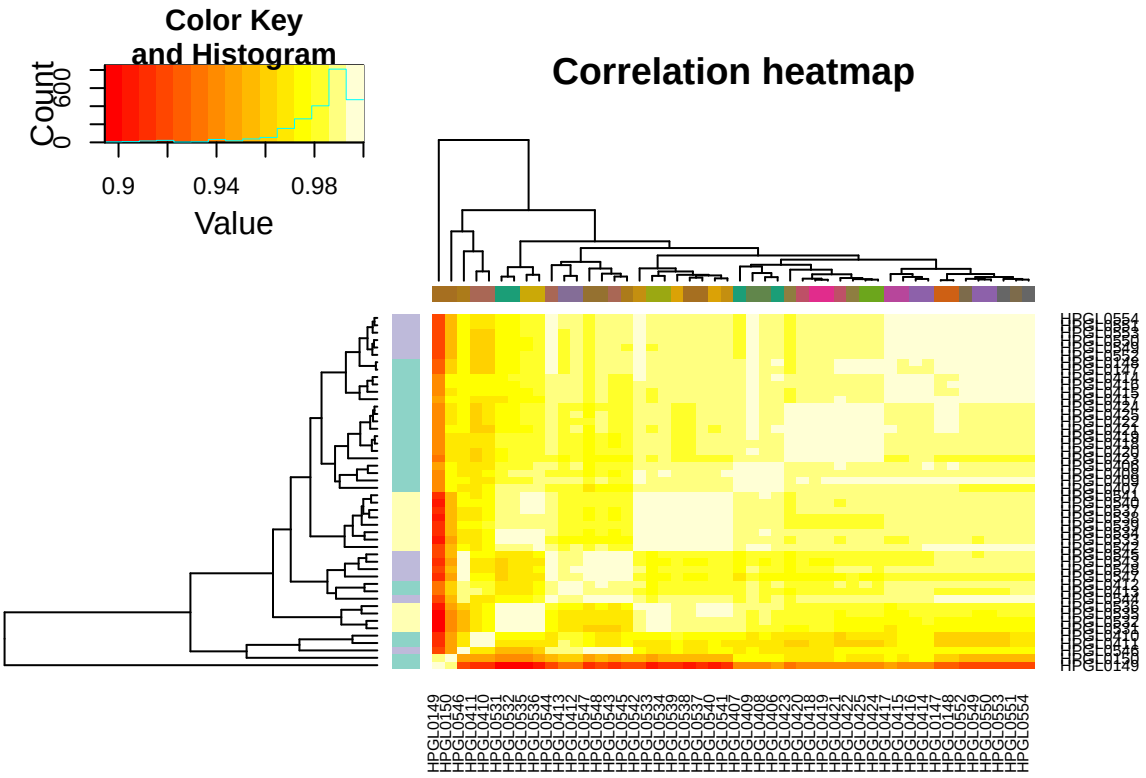
- initial_metadata: A backup of the metadata
- original_expressionset: A backup of the raw counts
- expressionset: The current count data
- samples: A data frame of metadata used for subsets
- design: The design of the experiment
- definitions: Extended design information, these are probably redundant and should be pruned.
- stages: The experimental stage
- types: Cell types
- conditions: Experimental condition
- batches: Experimental batch
- samplenames: Names of the samples
- colors: Colors chosen for graphs and such
- names: Bringing together the condition/batch
- filtered: low-count filtering status of the counts
- transform: transformation applied to the counts
- norm: normalization applied to the counts
- convert: cpm/rpkm/etc applied to the data
- original_libsize: the library sizes before normalization
- columns: A backup of the sample names

One possibility would be to examine the data in its unmolested state:

```
raw_metrics <- s_p(graph_metrics(expt, qq=TRUE))$result
```

```
## Graphing number of non-zero genes with respect to CPM by library.

## Graphing library sizes.

## Graphing a boxplot.

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

## Graphing a correlation heatmap.
```

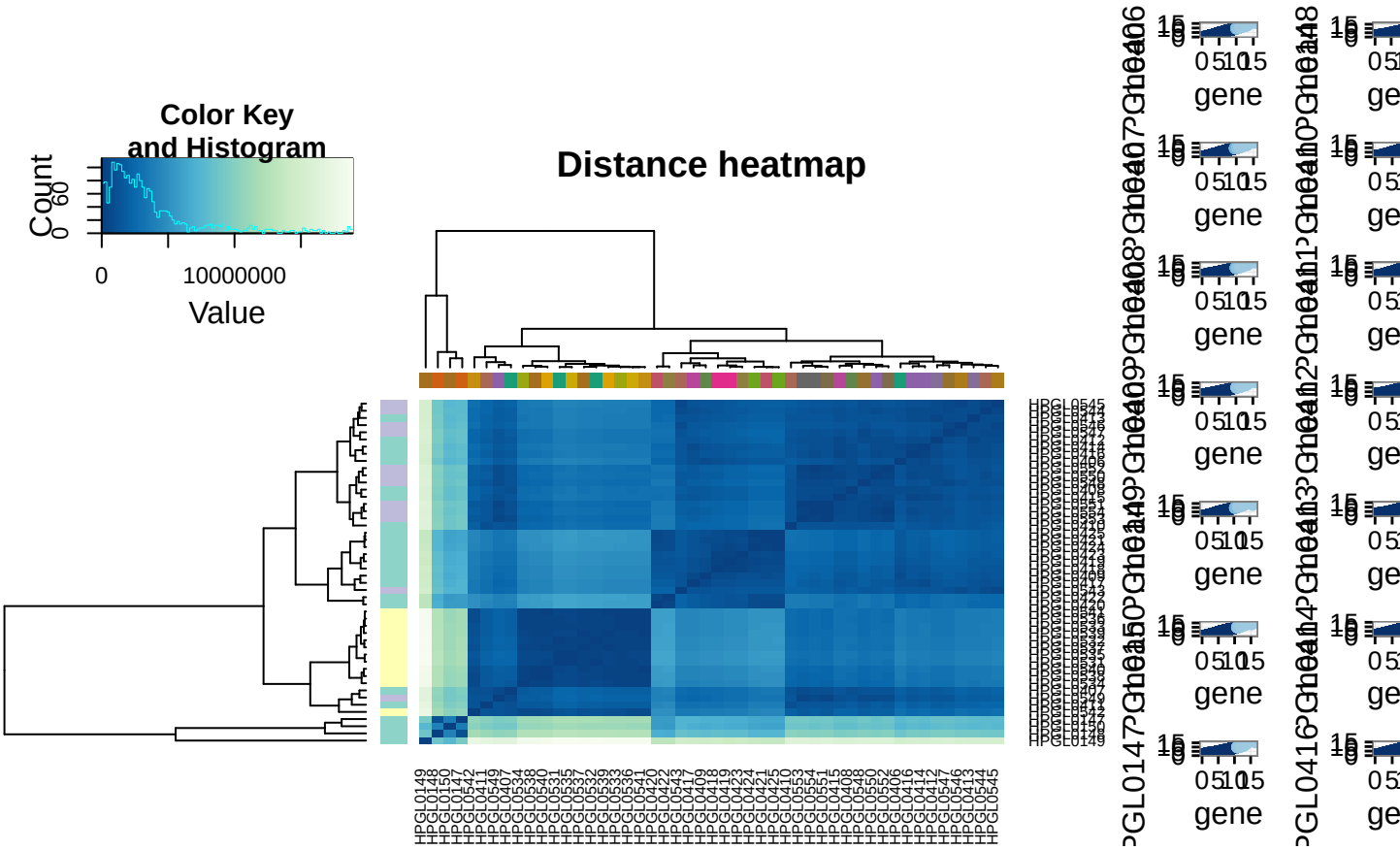
```
## Graphing a standard median correlation.

## Performing correlation.

## Graphing a distance heatmap.
```

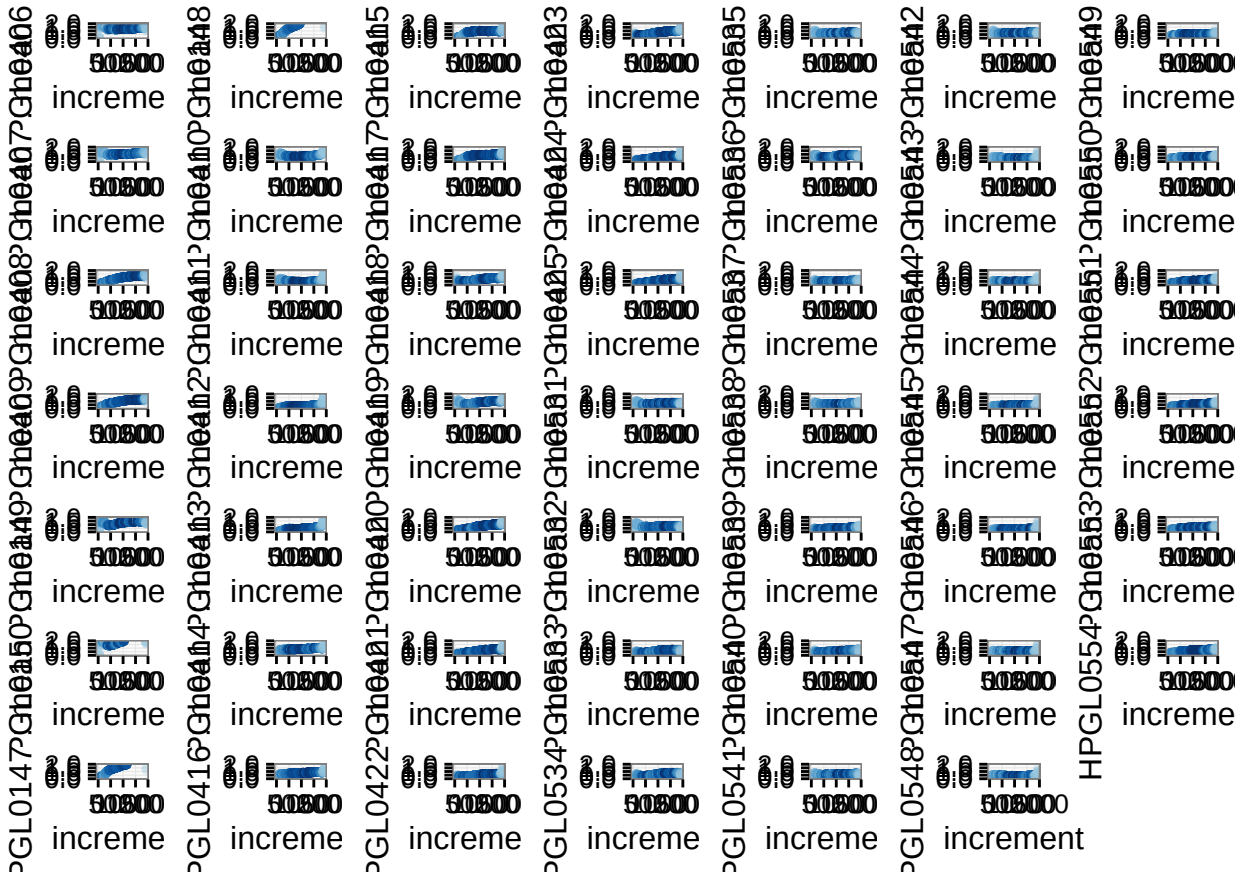



```
## Graphing a standard median distance.

## Performing distance.

## Graphing a PCA plot.

## Plotting a density plot.

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

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

## QQ plotting!

## Making plot of HPGL0406(1) vs. a sample distribution.

## Making plot of HPGL0407(2) vs. a sample distribution.

## Making plot of HPGL0408(3) vs. a sample distribution.
```

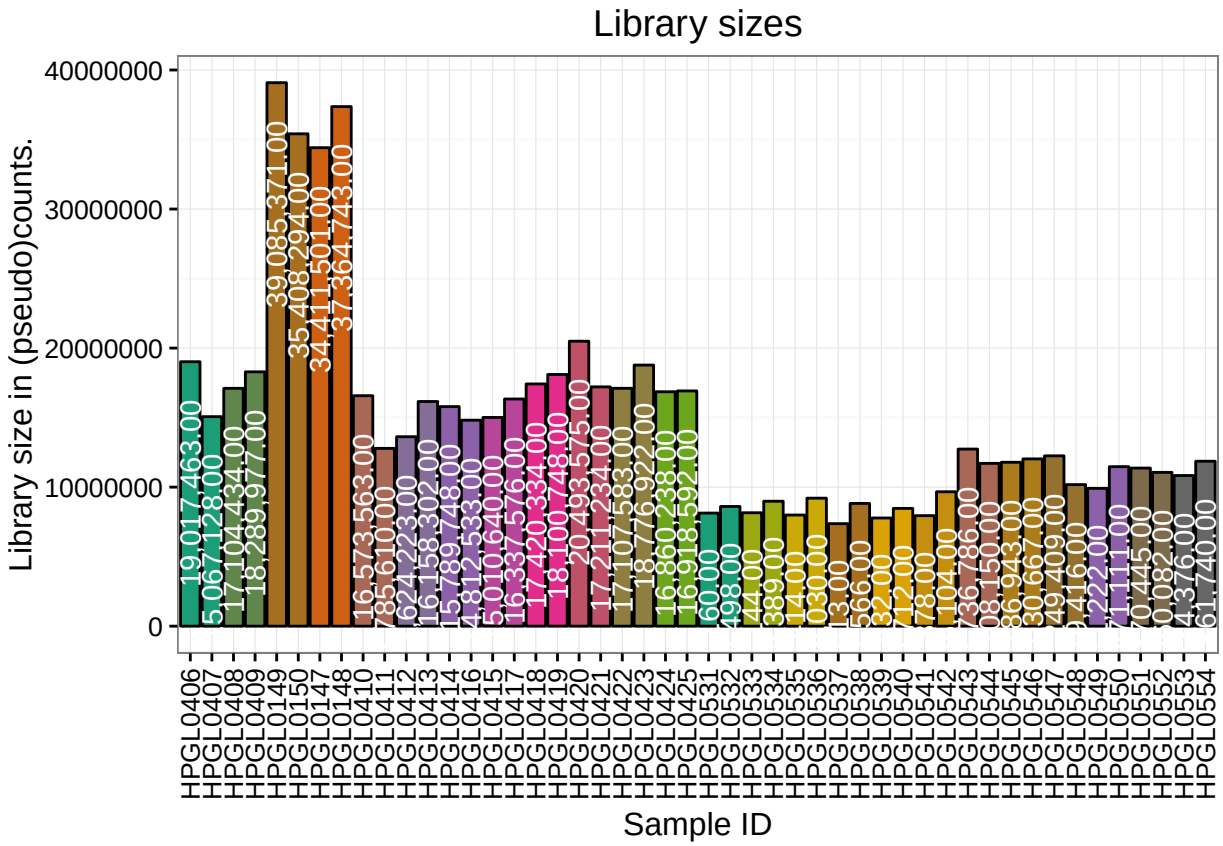
```
## Making plot of HPGL0409(4) vs. a sample distribution.

## Making plot of HPGL0149(5) vs. a sample distribution.

## Making plot of HPGL0150(6) vs. a sample distribution.

## Making plot of HPGL0147(7) vs. a sample distribution.

## Making plot of HPGL0148(8) vs. a sample distribution.

## Making plot of HPGL0410(9) vs. a sample distribution.

## Making plot of HPGL0411(10) vs. a sample distribution.

## Making plot of HPGL0412(11) vs. a sample distribution.

## Making plot of HPGL0413(12) vs. a sample distribution.

## Making plot of HPGL0414(13) vs. a sample distribution.

## Making plot of HPGL0416(14) vs. a sample distribution.

## Making plot of HPGL0415(15) vs. a sample distribution.

## Making plot of HPGL0417(16) vs. a sample distribution.

## Making plot of HPGL0418(17) vs. a sample distribution.

## Making plot of HPGL0419(18) vs. a sample distribution.

## Making plot of HPGL0420(19) vs. a sample distribution.

## Making plot of HPGL0421(20) vs. a sample distribution.

## Making plot of HPGL0422(21) vs. a sample distribution.

## Making plot of HPGL0423(22) vs. a sample distribution.

## Making plot of HPGL0424(23) vs. a sample distribution.

## Making plot of HPGL0425(24) vs. a sample distribution.

## Making plot of HPGL0531(25) vs. a sample distribution.

## Making plot of HPGL0532(26) vs. a sample distribution.

## Making plot of HPGL0533(27) vs. a sample distribution.
```

```
## Making plot of HPGL0534(28) vs. a sample distribution.

## Making plot of HPGL0535(29) vs. a sample distribution.

## Making plot of HPGL0536(30) vs. a sample distribution.

## Making plot of HPGL0537(31) vs. a sample distribution.

## Making plot of HPGL0538(32) vs. a sample distribution.

## Making plot of HPGL0539(33) vs. a sample distribution.

## Making plot of HPGL0540(34) vs. a sample distribution.

## Making plot of HPGL0541(35) vs. a sample distribution.

## Making plot of HPGL0542(36) vs. a sample distribution.

## Making plot of HPGL0543(37) vs. a sample distribution.

## Making plot of HPGL0544(38) vs. a sample distribution.

## Making plot of HPGL0545(39) vs. a sample distribution.

## Making plot of HPGL0546(40) vs. a sample distribution.

## Making plot of HPGL0547(41) vs. a sample distribution.

## Making plot of HPGL0548(42) vs. a sample distribution.

## Making plot of HPGL0549(43) vs. a sample distribution.

## Making plot of HPGL0550(44) vs. a sample distribution.

## Making plot of HPGL0551(45) vs. a sample distribution.

## Making plot of HPGL0552(46) vs. a sample distribution.

## Making plot of HPGL0553(47) vs. a sample distribution.

## Making plot of HPGL0554(48) vs. a sample distribution.
```

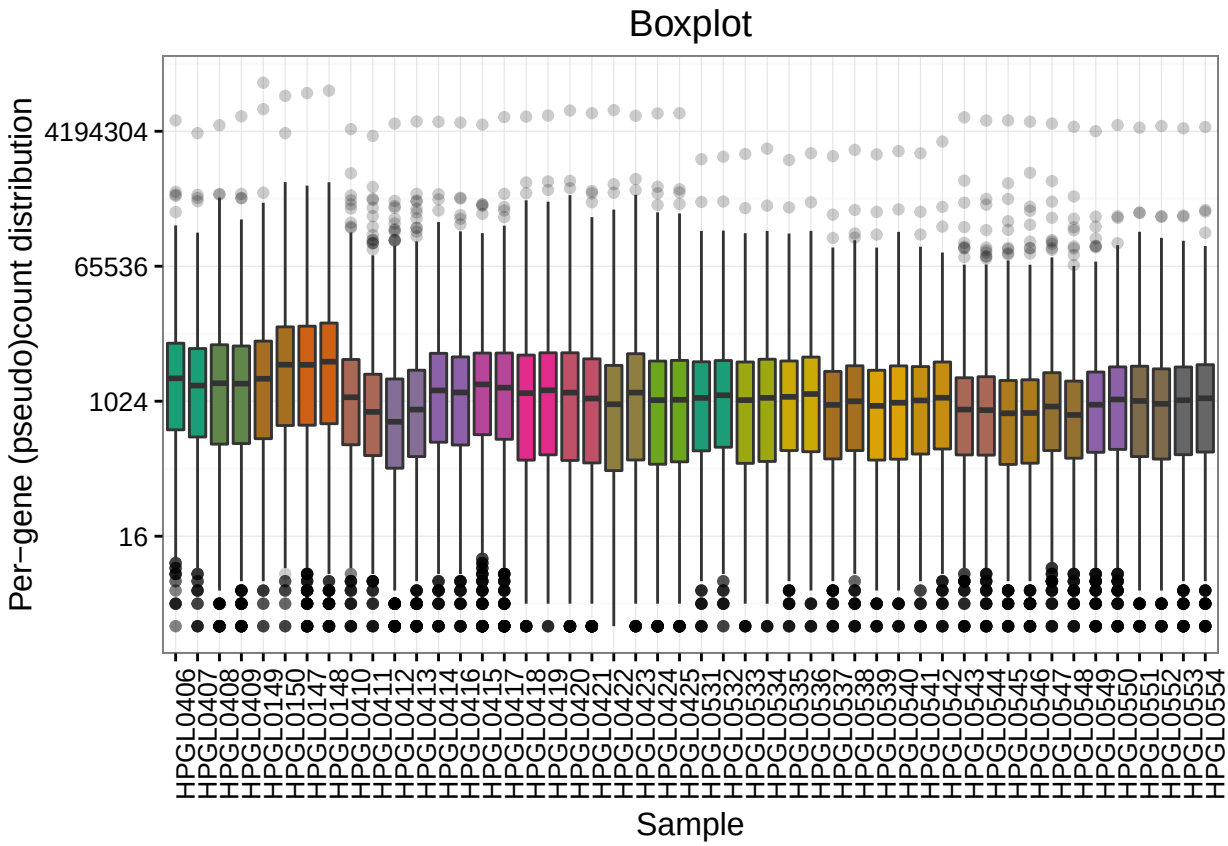

Color Key
and Histogram
Count
0
60
0
10000000
Value
Distance heatmap
gene



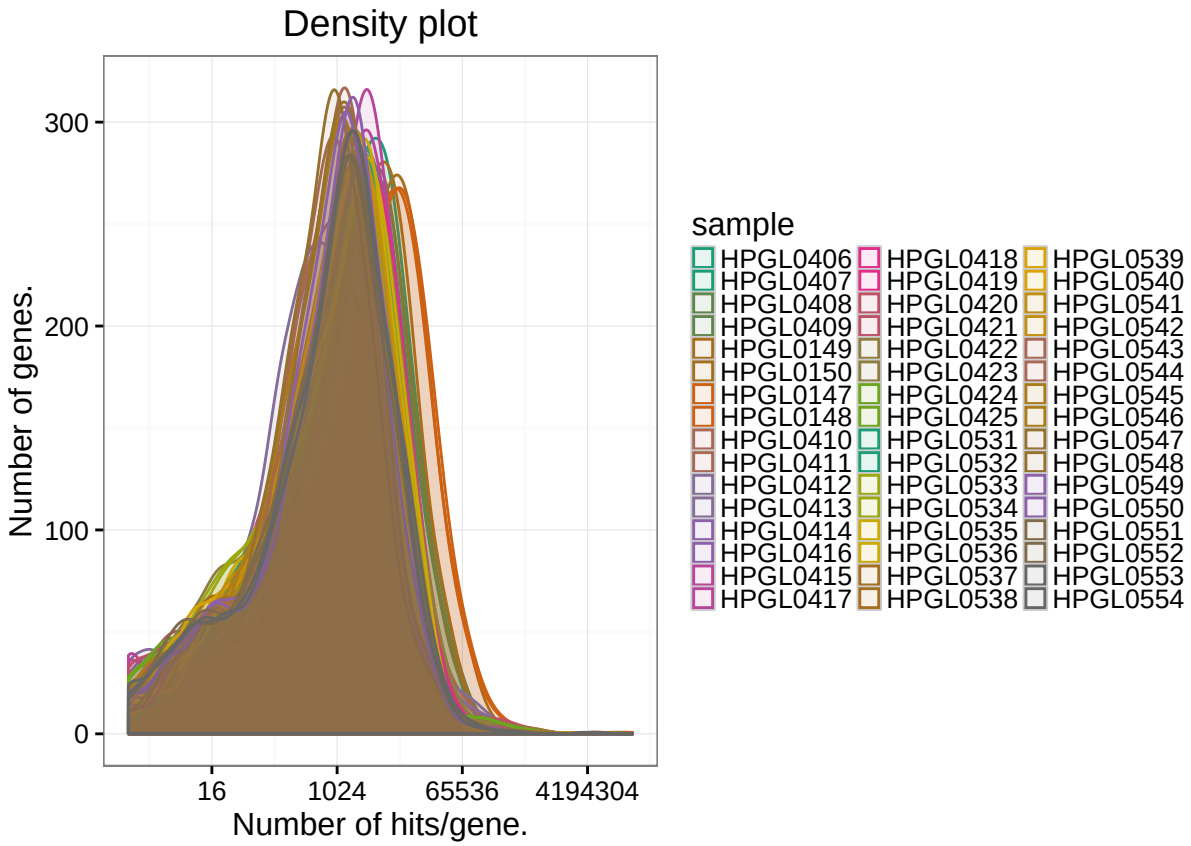
```
## View a raw library size plot
raw_metrics$libsize
```



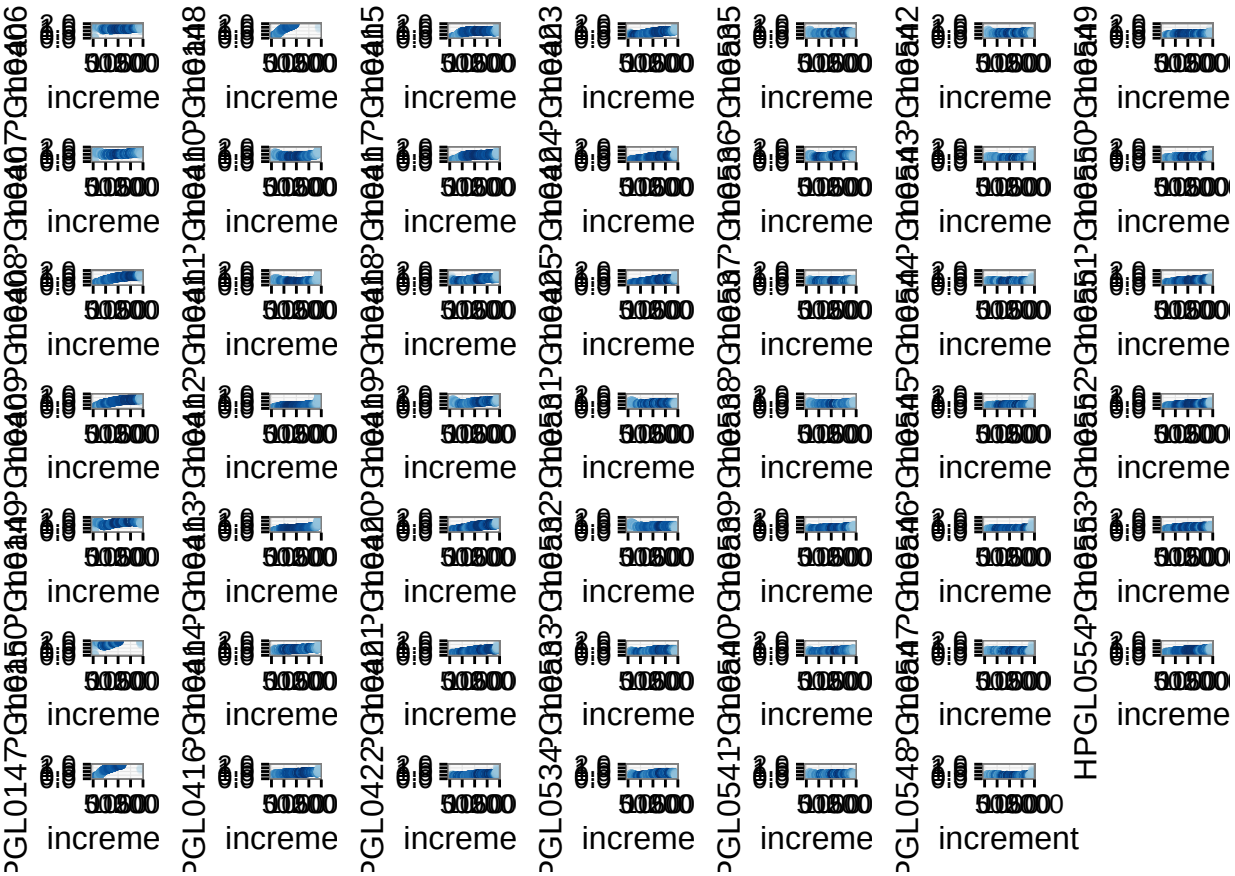

```
## Or boxplot to see the data distribution
raw_metrics$boxplot
```




```
## The warning is because it automatically uses a log scale and there are some 0 count genes.
## Perhaps you prefer density plots
raw_metrics$density
```




```
## quantile/quantile plots compared to the median of all samples
raw_metrics$qqrat
```



```
## Here we can see some samples are differently 'shaped' compared to the median than others
## There are other plots one may view, but this data set is a bit too crowded as is.
## The following summary shows the other available plots:
summary(raw_metrics)
```

```
##          Length Class        Mode
## nonzero   9     gg           list
## libsize   9     gg           list
## boxplot   9     gg           list
## corheat   3     recordedplot list
## smc       9     gg           list
## disheat   3     recordedplot list
## smd       9     gg           list
## pcaplot   9     gg           list
## pcatable  8     data.frame   list
## pcares    4     data.frame   list
## pcavar   47     -none-       numeric
## density   9     gg           list
## qqlog     3     recordedplot list
## qqrat     3     recordedplot list
## ma        0     -none-       NULL
```

On the other hand, we might take a subset of the data to focus on the late-log vs. early-log samples.

The expt_subset() function allows one to pull material from the experimental design.

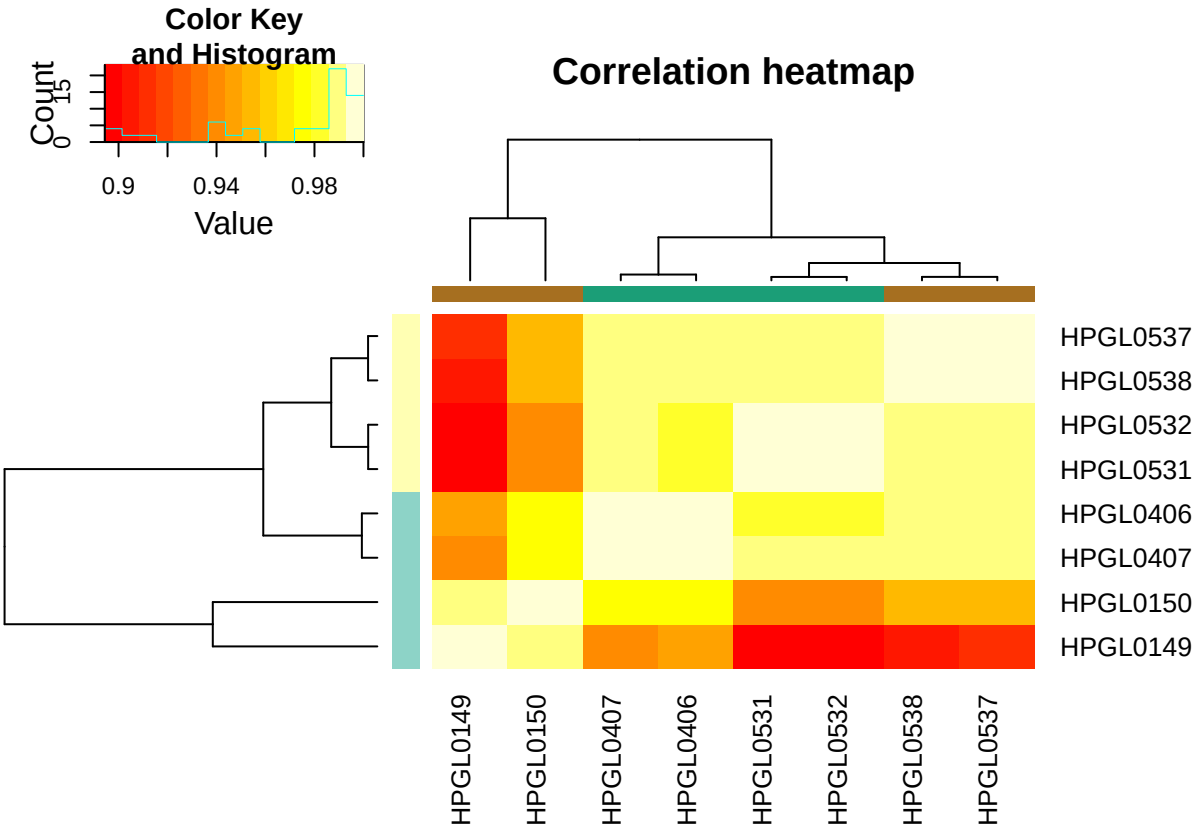
Once we have a smaller data set, we can more easily use PCA to see how the sample separate.

```
head(expt$definition)
```

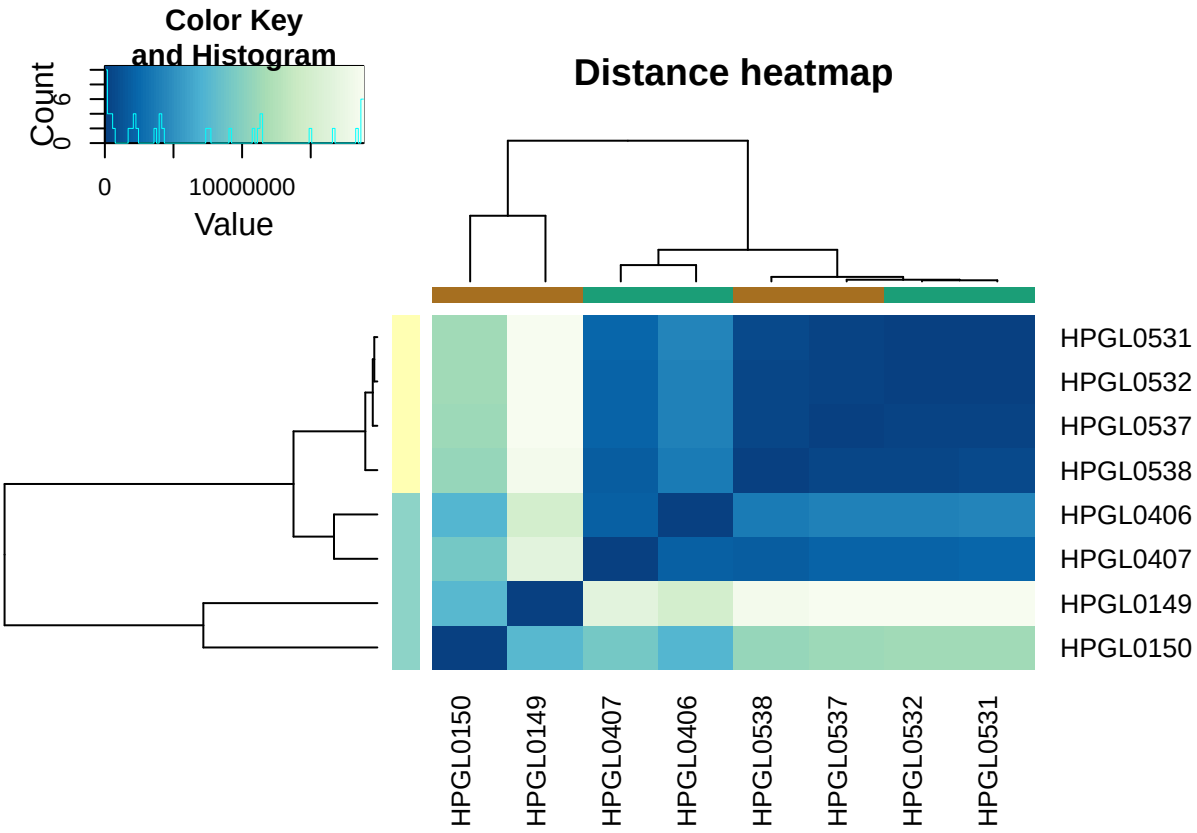
```
## NULL
```

```
## elt stands for: "early/late in thy"
elt <- expt_subset(expt, subset="(stage=='EL'|stage=='LL')&type=='WT'&grepl(pattern='_thy', x=condition)

elt_metrics <- graph_metrics(elt)
```

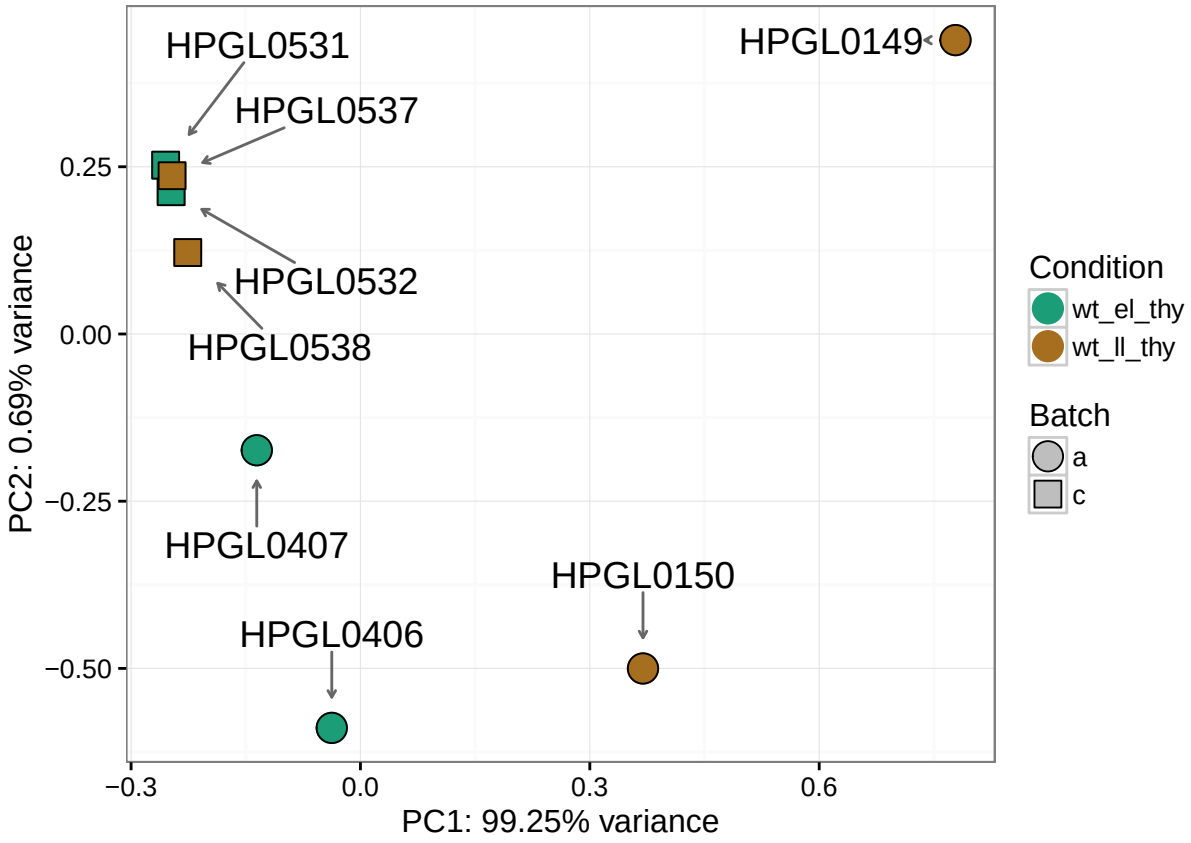
```
## Graphing number of non-zero genes with respect to CPM by library.

## Graphing library sizes.

## Graphing a boxplot.

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

## Graphing a correlation heatmap.

## Graphing a standard median correlation.

## Performing correlation.

## Graphing a distance heatmap.
```



```
## Graphing a standard median distance.

## Performing distance.

## Graphing a PCA plot.

## Plotting a density plot.

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

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



```
elt_metrics$pcaplot
```




```
head(elt_metrics$pcares)
```

```
##   propVar cumPropVar cond.R2 batch.R2
## 1   99.25      99.25   22.81    47.50
## 2    0.69      99.94    4.43    33.91
## 3    0.03      99.97   54.89     7.96
## 4    0.03     100.00    7.51     5.61
## 5    0.01     100.01    8.78     5.01
## 6    0.00     100.01    1.40     0.00
```

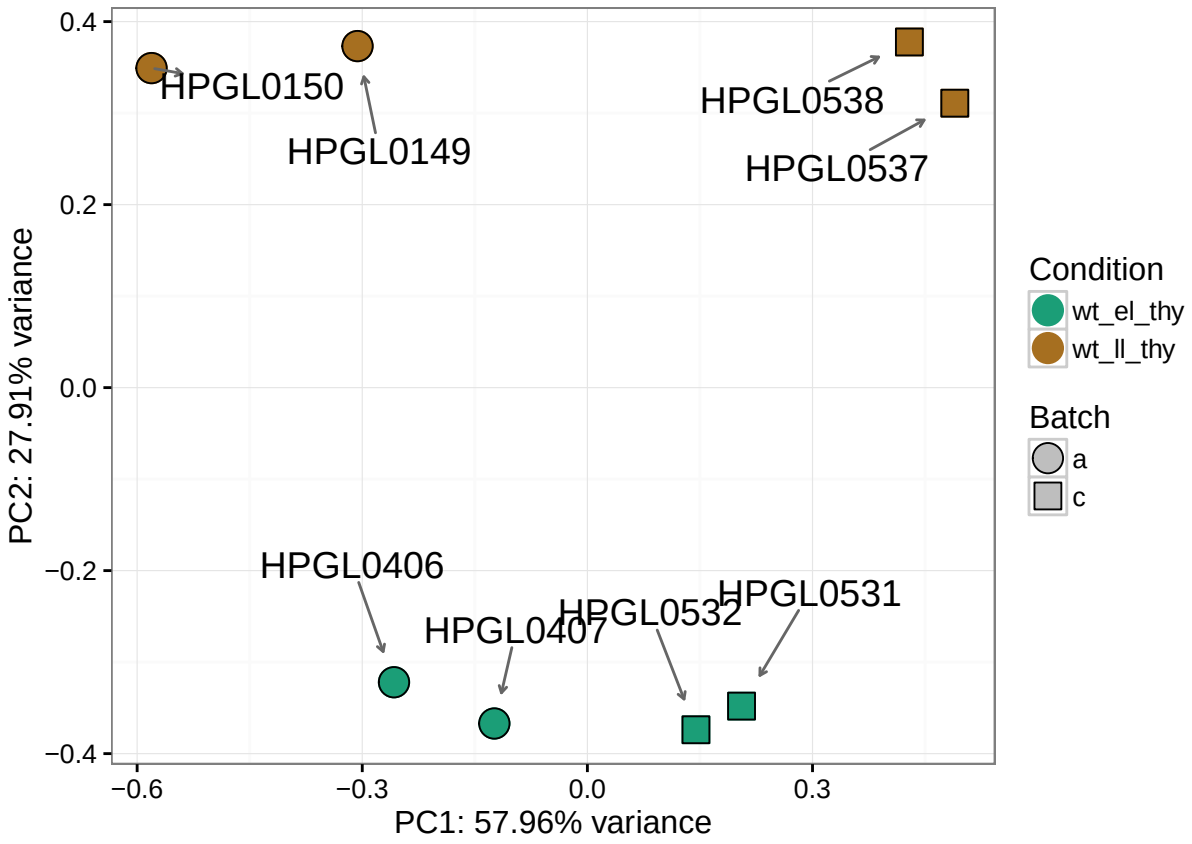
It is pretty obvious that the raw data is a bit jumbled according to PCA. This is not paricularly suprising since we didn't normalize it at all.

```
## doing nothing to the data except log2 transforming it has a surprisingly large effect
norm_test <- normalize_expt(elt, transform="log2")
```

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

## log2(data)

## It backs up the current data into a slot named:
##  expt$backup_expressionset. It will also save copies of each step along the way
```

```
##  in expt$normalized with the corresponding libsizes. Keep the libsizes in mind
##  when invoking limma.  The appropriate libsize is the non-log(cpm(normalized)).
##  This is most likely kept at:
##  'new_expt$normalized$intermediate_counts$normalization$libsizes'
##  A copy of this may also be found at:
##  new_expt$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 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).

## Leaving the data unnormalized.  This is necessary for DESeq, but
##  EdgeR/limma might benefit from normalization.  Good choices include quantile,
##  size-factor, tmm, etc.

## 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.

## Applying: log2 transformation.

## transform_counts: Found 531 values equal to 0, adding 1 to the matrix.
```

```
plot_pca(norm_test)$plot
```

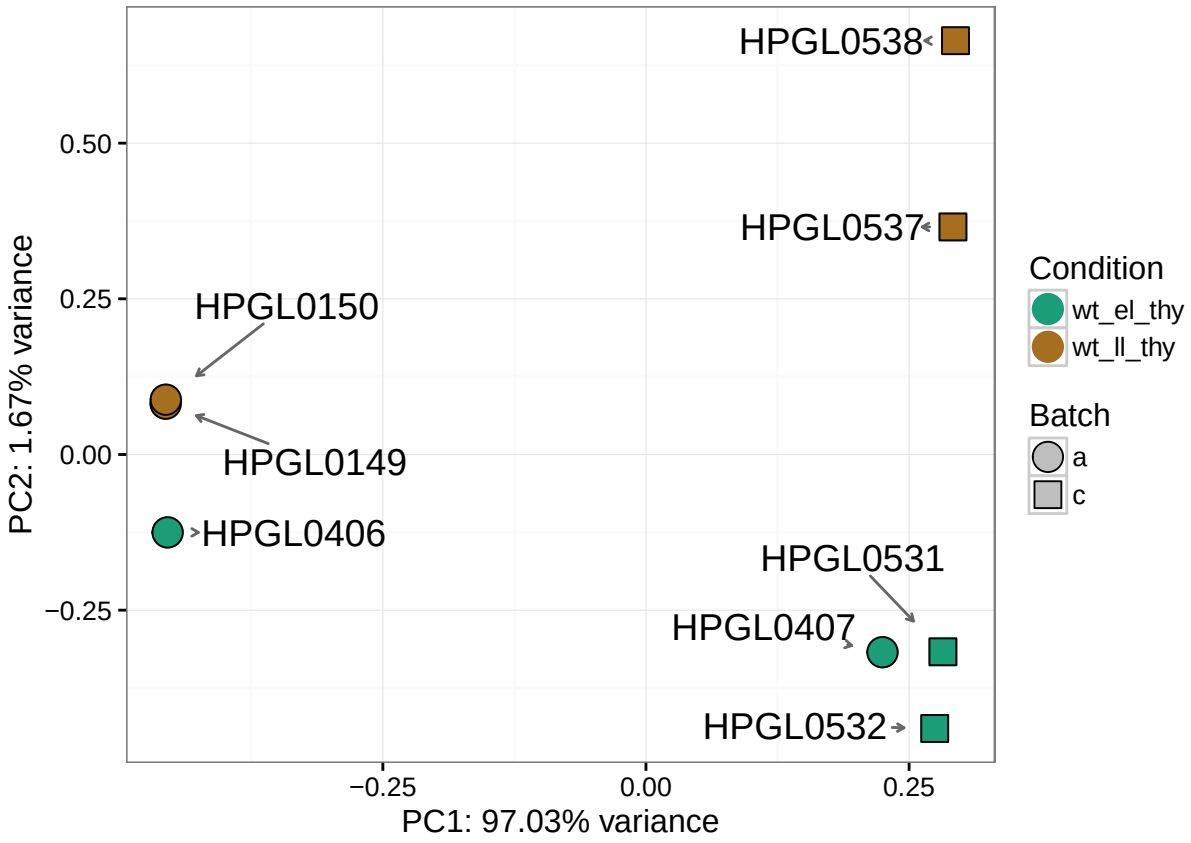



```
## a quantile normalization alone affect some, but not all of the data
norm_test <- normalize_expt(elt, norm="quant")
```

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

## quant(data)

## It backs up 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 at:
##  'new_expt$normalized$intermediate_counts$normalization$libsizes'
##  A copy of this may also be found at:
##  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 do not accept 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.
```

```
plot_pca(norm_test)$plot
```

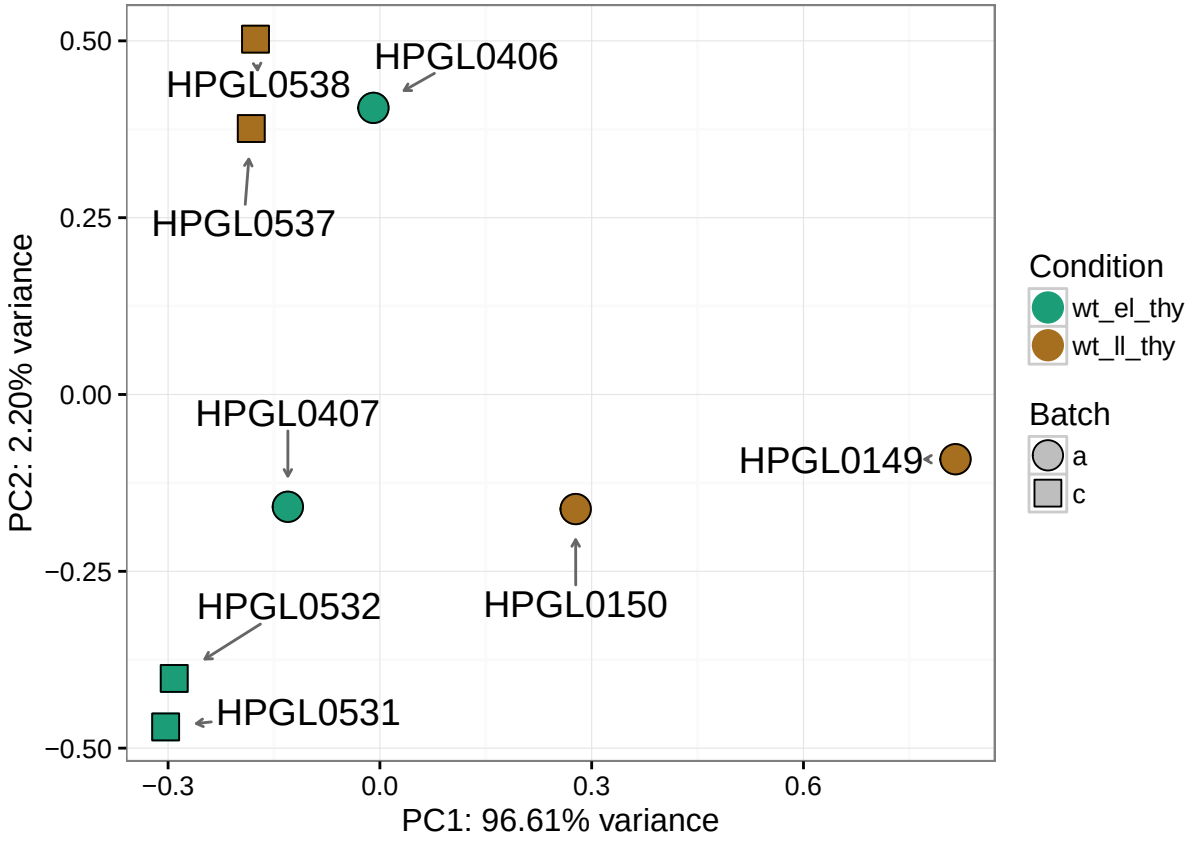



```
## cpm alone brings out some samples, too
norm_test <- normalize_expt(elt, convert="cpm")
```

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

## cpm(data)

## It backs up 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 at:
##  'new_expt$normalized$intermediate_counts$normalization$libsizes'
##  A copy of this may also be found at:
##  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 do not accept transformed data.

## Leaving the data unnormalized.  This is necessary for DESeq, but
##  EdgeR/limma might benefit from normalization.  Good choices include quantile,
##  size-factor, tmm, etc.

## 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.
```

```
plot_pca(norm_test)$plot
```

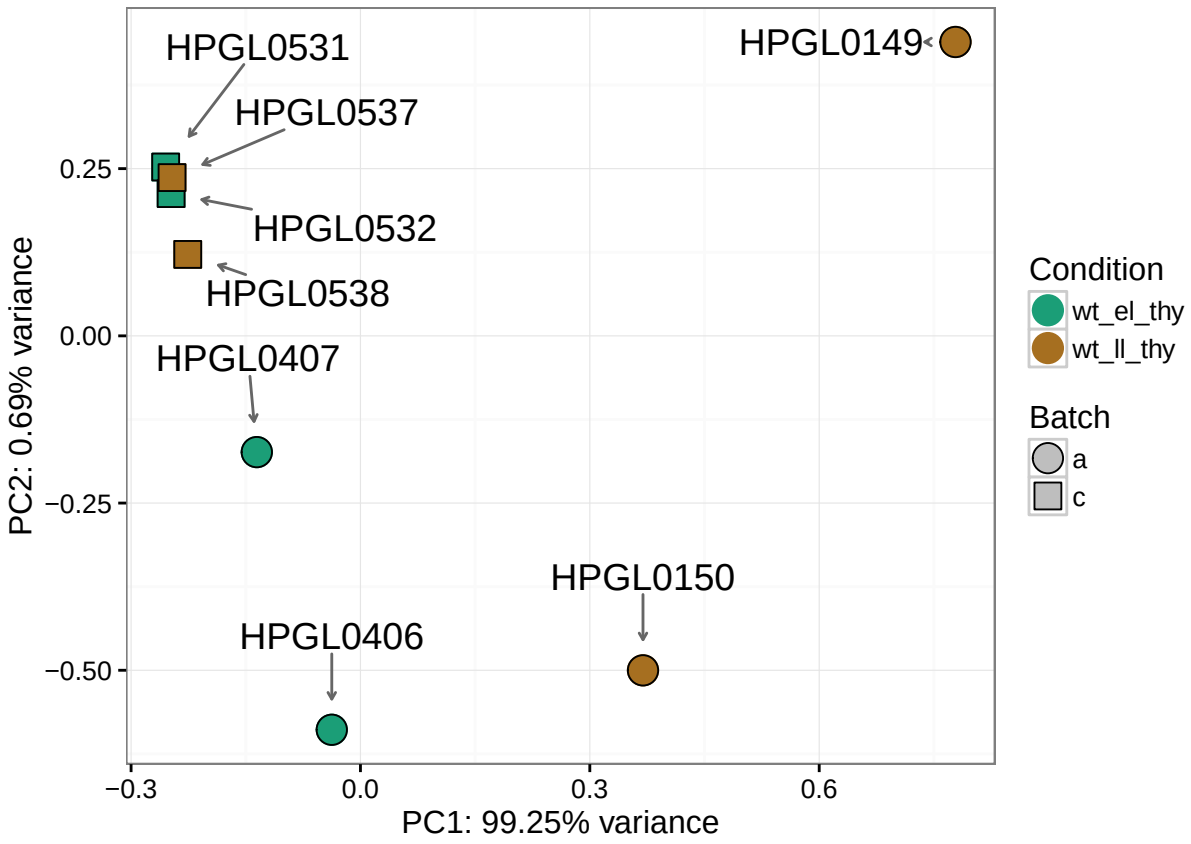



```
## low count filtering has some effect, too
norm_test <- normalize_expt(elt, filter_low="pofa")
```

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

## low-filter(data)
```

```
## It backs up 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 at:
##  'new_expt$normalized$intermediate_counts$normalization$libsizes'
##  A copy of this may also be found at:
##  new_expt$best_libsize

## 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 do not accept 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).

## Leaving the data unnormalized.  This is necessary for DESeq, but
##  EdgeR/limma might benefit from normalization.  Good choices include quantile,
##  size-factor, tmm, etc.

## 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.

## Performing low-count filter with option: pofa

## Removing 56 low-count genes (1875 remaining).
```

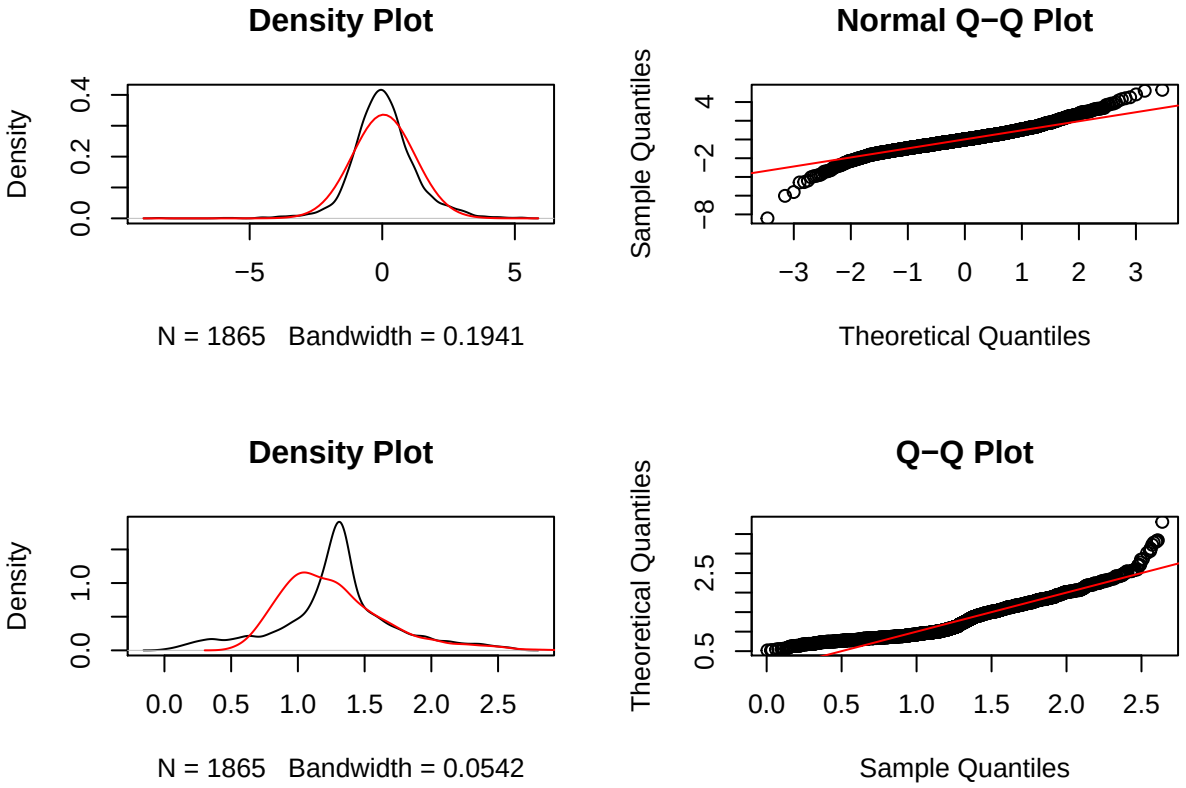
```
plot_pca(norm_test)$plot
```



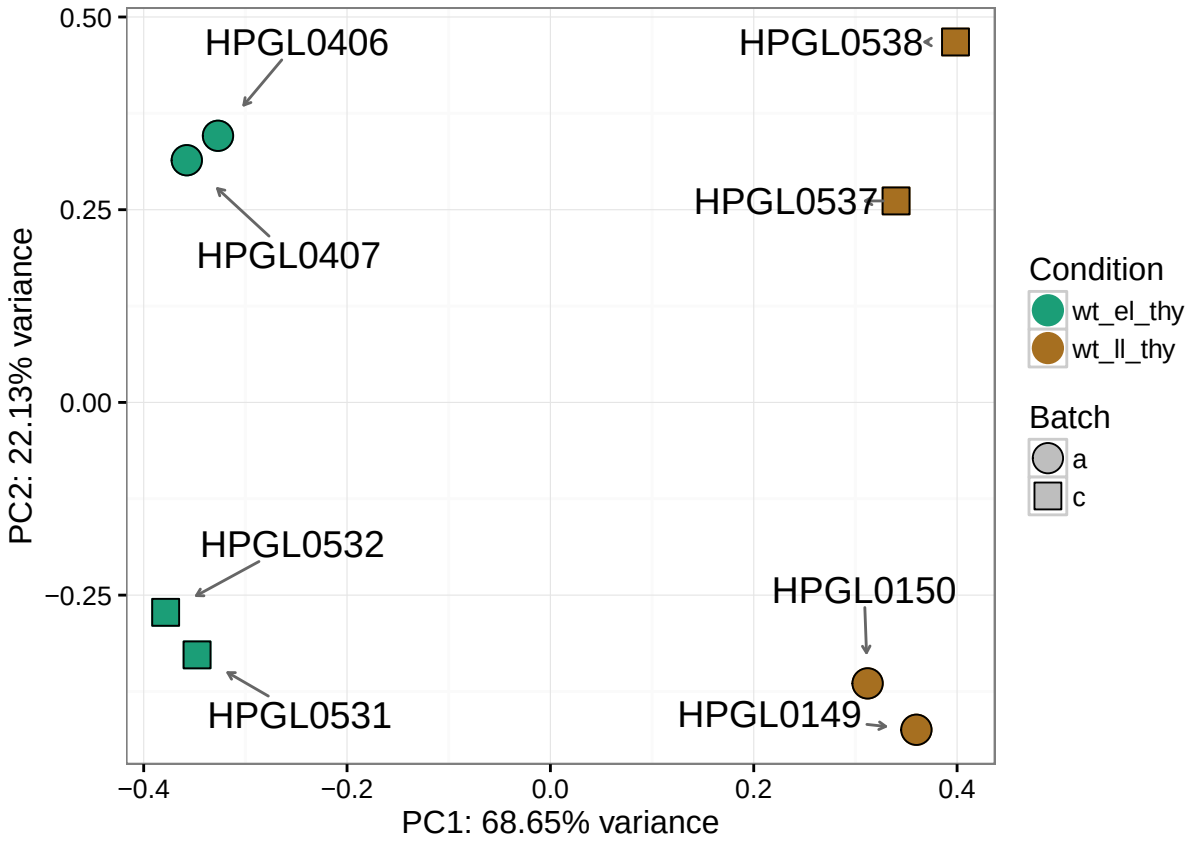

```
## how about if we mix and match methods?
norm_test <- normalize_expt(elt, transform="log2", convert="cpm", norm="quant", batch="combat_scale", fi
```

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

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

## It backs up 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 at:
##  'new_expt$normalized$intermediate_counts$normalization$libsizes'
##  A copy of this may also be found at:
##  new_expt$best_libsize

## Performing low-count filter with option: TRUE

## Removing 66 low-count genes (1865 remaining).

## Applying: log2 transformation.

## batch_counts: Before batch correction, 499 entries 0<x<1.

## batch_counts: Before batch correction, 815 entries are >= 0.
```

```
## batch_counts: Using sva::combat with a prior for batch correction and with scaling.

## Found 2 batches
## Adjusting for 1 covariate(s) or covariate level(s)
## Standardizing Data across genes
## Fitting L/S model and finding priors

## Finding parametric adjustments
## Adjusting the Data

## The number of elements which are < 0 after batch correction is: 814

## The variable low_to_zero sets whether to change <0 values to 0 and is: FALSE
```




```
plot_pca(norm_test)$plot
```



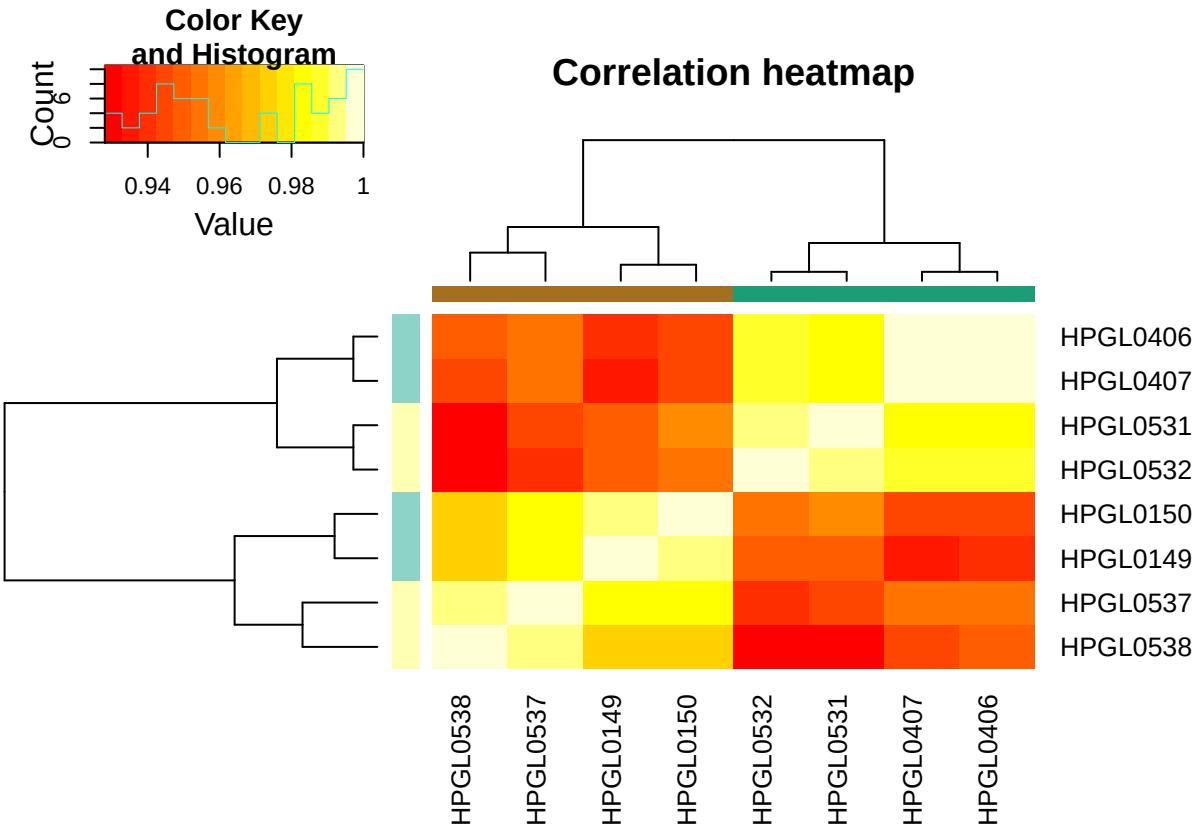

```
## The different batch effect testing methods have a pretty widely ranging effect on the clustering
## play with them by changing the batch= parameter to:
## "limma", "sva", "svaseq", "limmaresid", "ruvg", "combat", combatmod"
pca_test <- plot_pca(norm_test)
head(pca_test$res)
```

```
##   propVar cumPropVar cond.R2 batch.R2
## 1   68.65      68.65   99.47     0.01
## 2   22.13      90.78    0.18     0.83
## 3    2.79      93.57    0.14     0.73
## 4    2.23      95.80    0.11     0.02
## 5    2.06      97.86    0.03     2.92
## 6    1.57      99.43    0.07     0.12
```

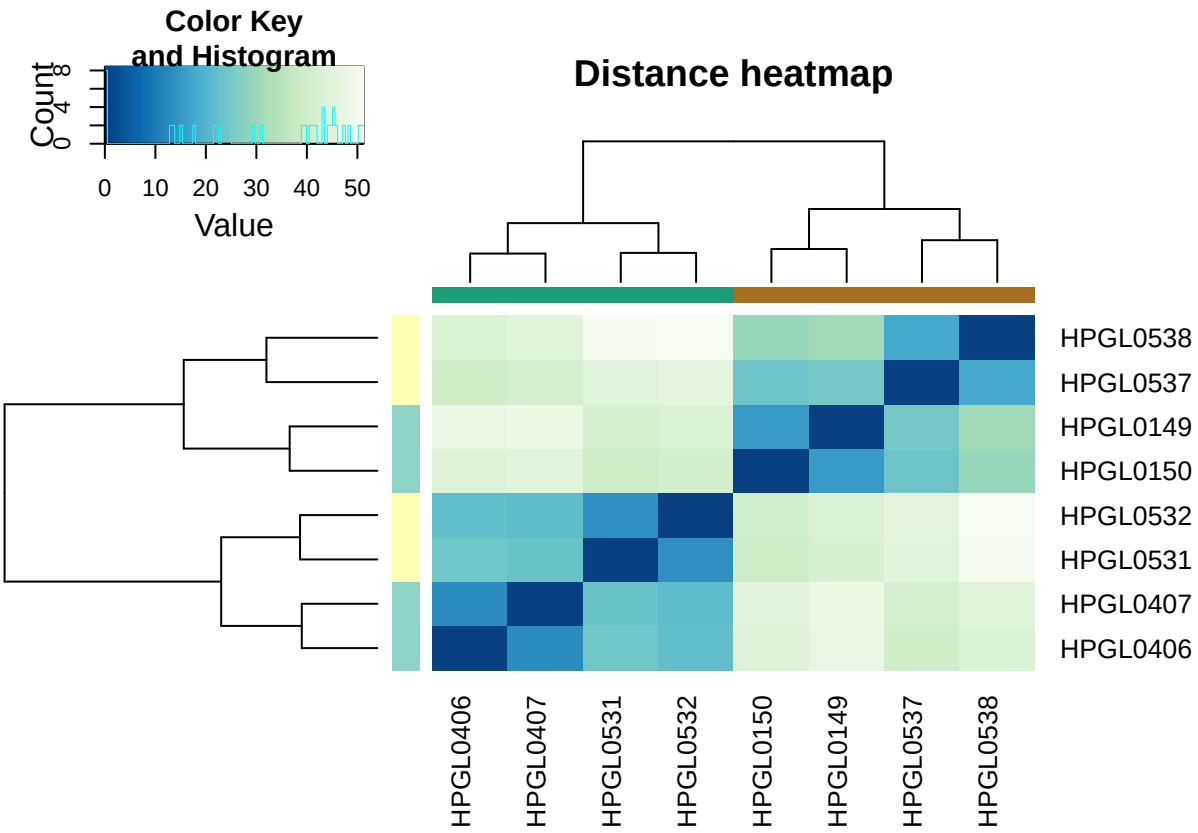
```
## Thus we see a dramatic decrease in variance accounted for
## by batch after applying limma's 'removebatcheffect'
## (see batch.R2 here vs. above)

## Some metrics are not very useful on (especially quantile) normalized data
norm_graphs <- graph_metrics(norm_test)
```

```
## Graphing number of non-zero genes with respect to CPM by library.

## Graphing library sizes.

## Graphing a boxplot.
```

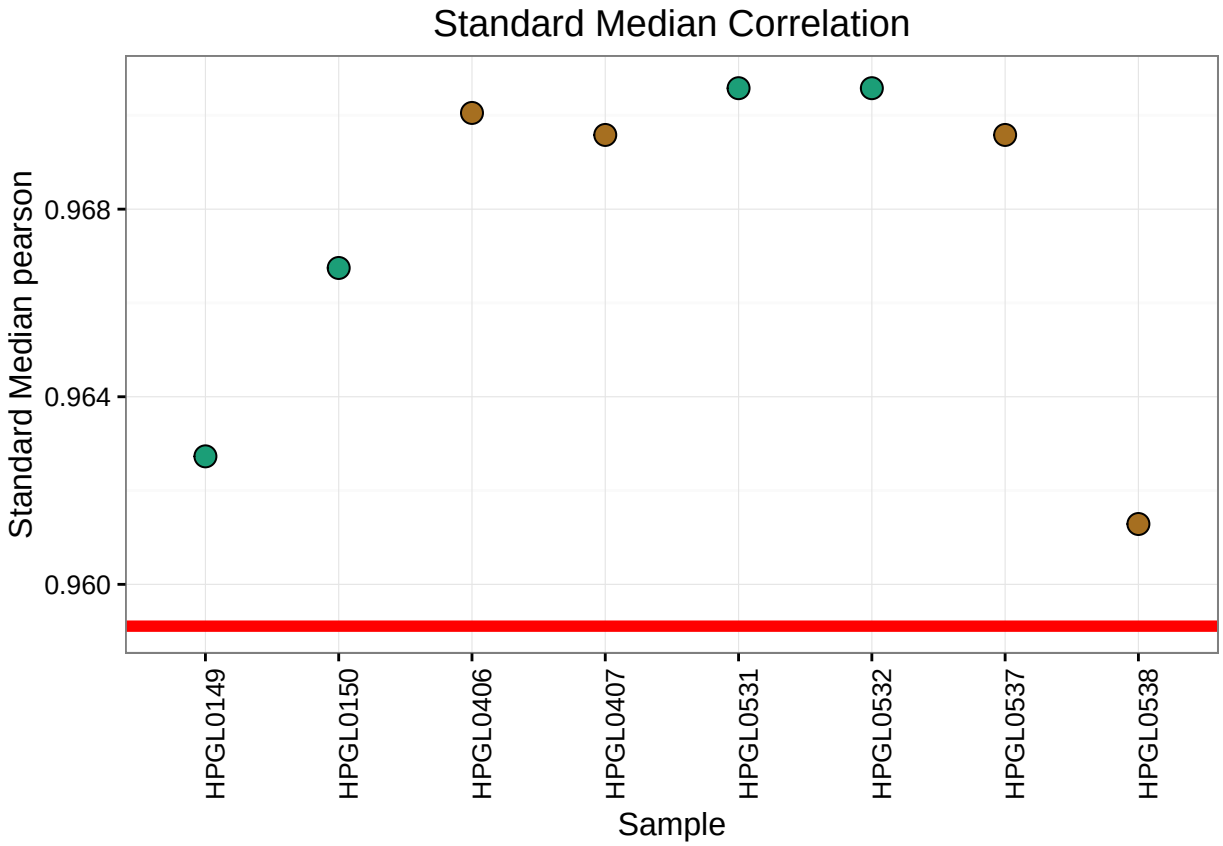
```
## Graphing a correlation heatmap.

## Graphing a standard median correlation.

## Performing correlation.

## Graphing a distance heatmap.
```



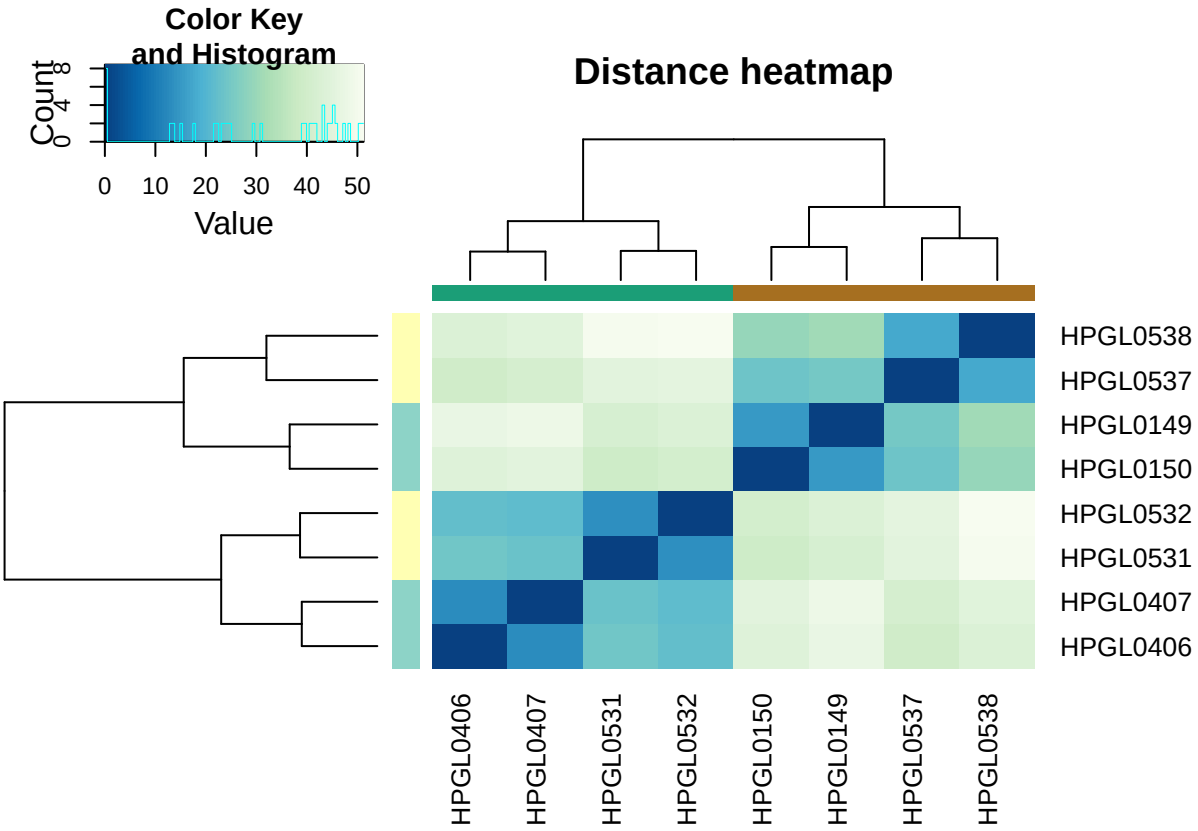

```
## Graphing a standard median distance.

## Performing distance.

## Graphing a PCA plot.

## Plotting a density plot.
```




```
norm_graphs$smc
```




```
norm_graphs$disheat  ## svaseq's batch correction seems to draw out the signal quite nicely.
```




```
## It is worth noting that the wt, early log, thy, replicate c samples are still a bit weird.
```