hpgltools usage

atb abelew@gmail.com

2016-04-14

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"                 "lengths"
## [41] "limma_comparison"             "limma_results"
## [43] "meta"                         "meta_dataframe"
## [45] "most_v0M1"                    "mut_120"
## [47] "normbatch_expt"               "norm_expt"
## [49] "norm_graphs"                  "norm_test"
## [51] "num_colors"                   "pca_test"
## [53] "pombe"                        "pombe_filters"
## [55] "pombe_goids"                  "pombe_transcripts"
## [57] "possible_pombe_attributes"    "raw_metrics"
## [59] "rle_expt"                     "sample_definitions"
## [61] "scatter_wt_mut"               "sf_expt"
## [63] "sig_genes"                    "table"
## [65] "test_pca"                     "tm_expt"
## [67] "tmp_definitions"              "tt"
## [69] "updown_genes"                 "up_expt"
## [71] "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$definitions
counts <- Biobase::exprs(most_v0M1$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                 0     -none-        NULL
```

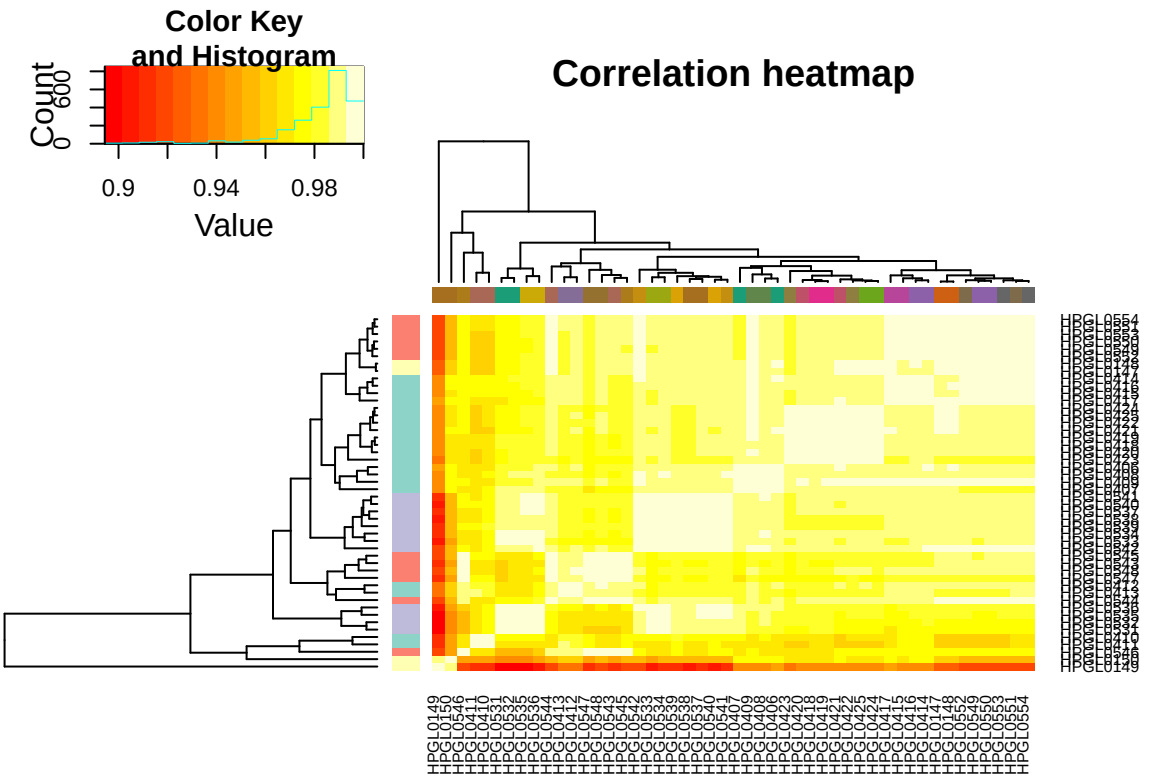
The data structure generated by create_expt() is a list containing the following slots:

- 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

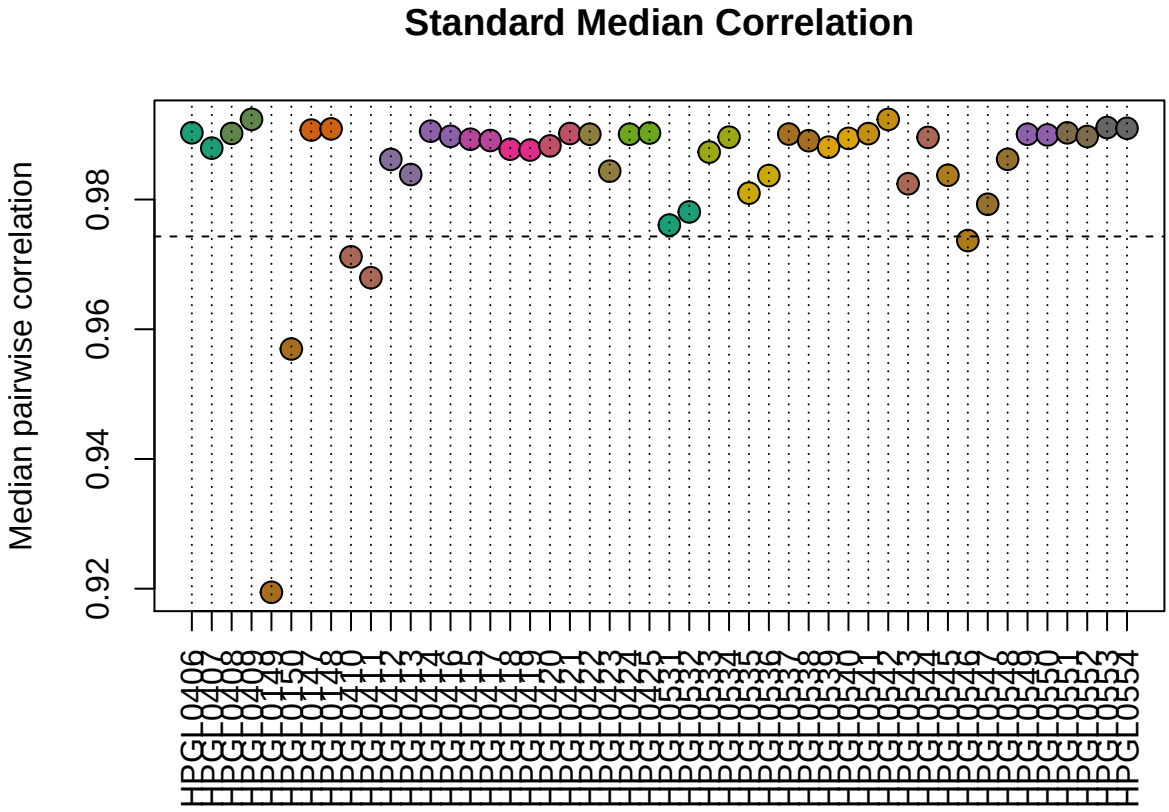
One possibility would be to examine the data in its unmolested state:

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

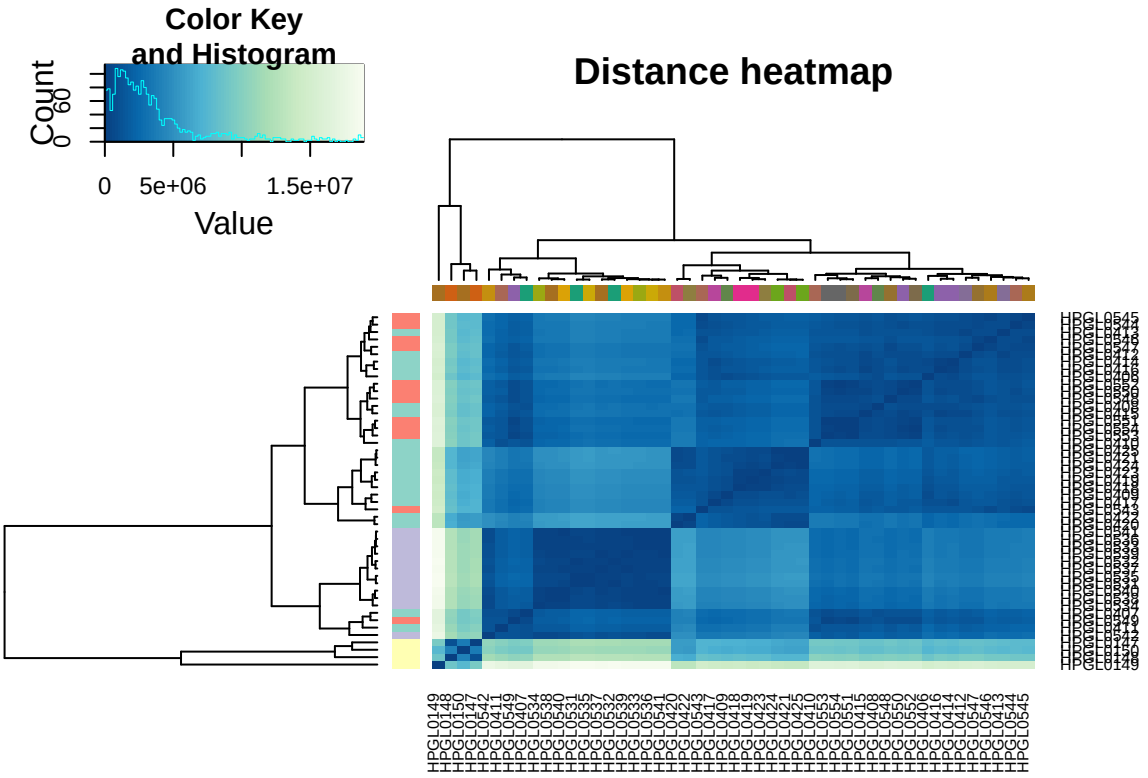
```
## Graphing number of non-zero genes with respect to CPM by library.

## 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.
```



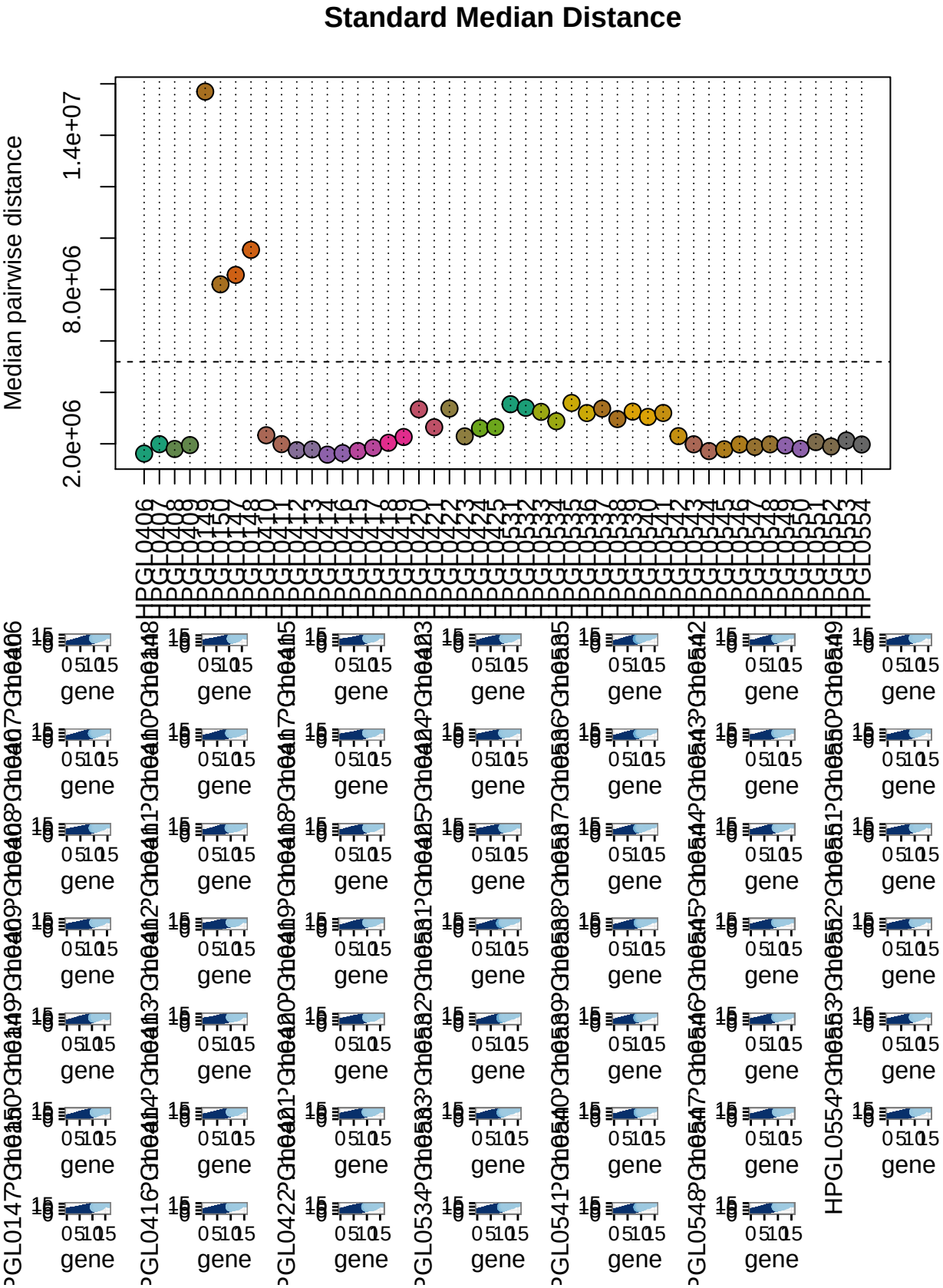

```
## Graphing a distance heatmap.
```

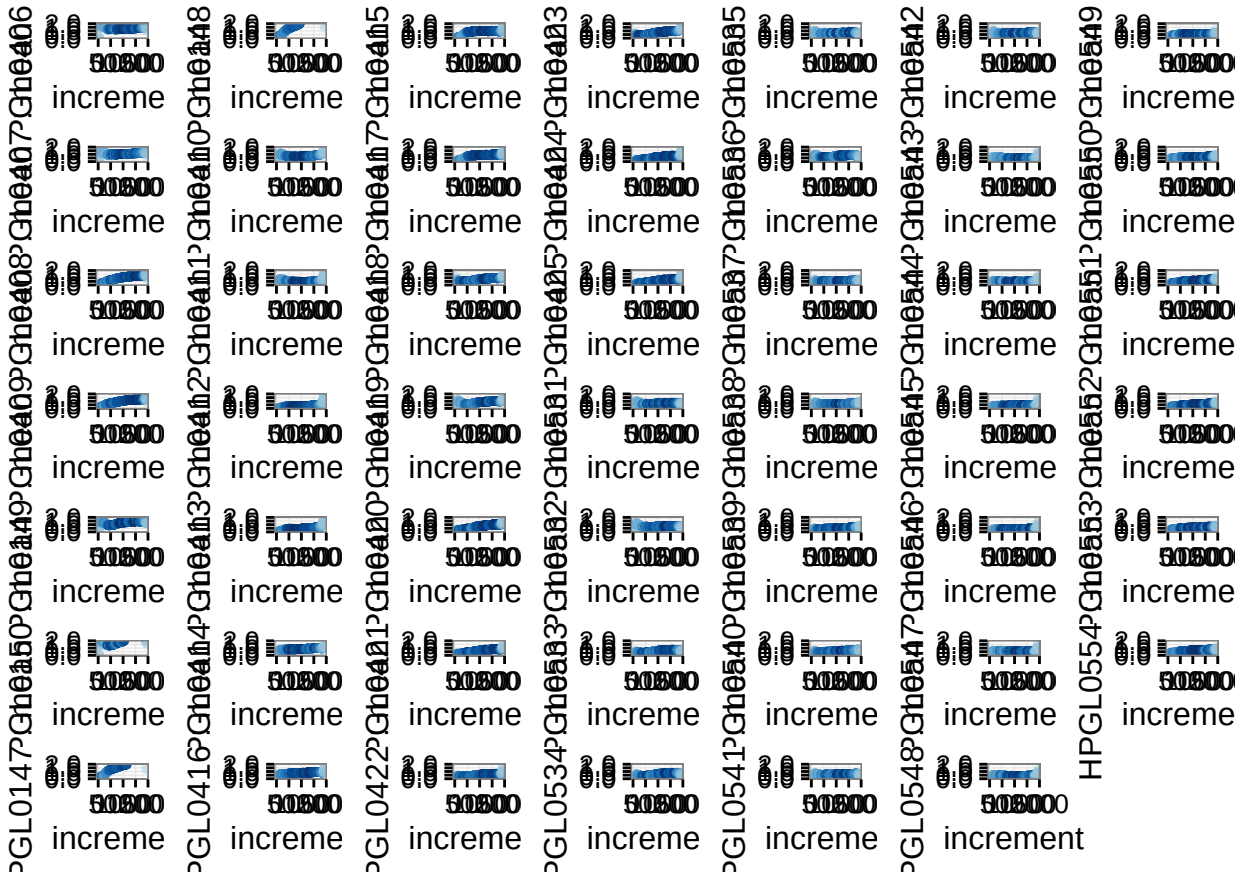



```
## Graphing a standard median distance.
```

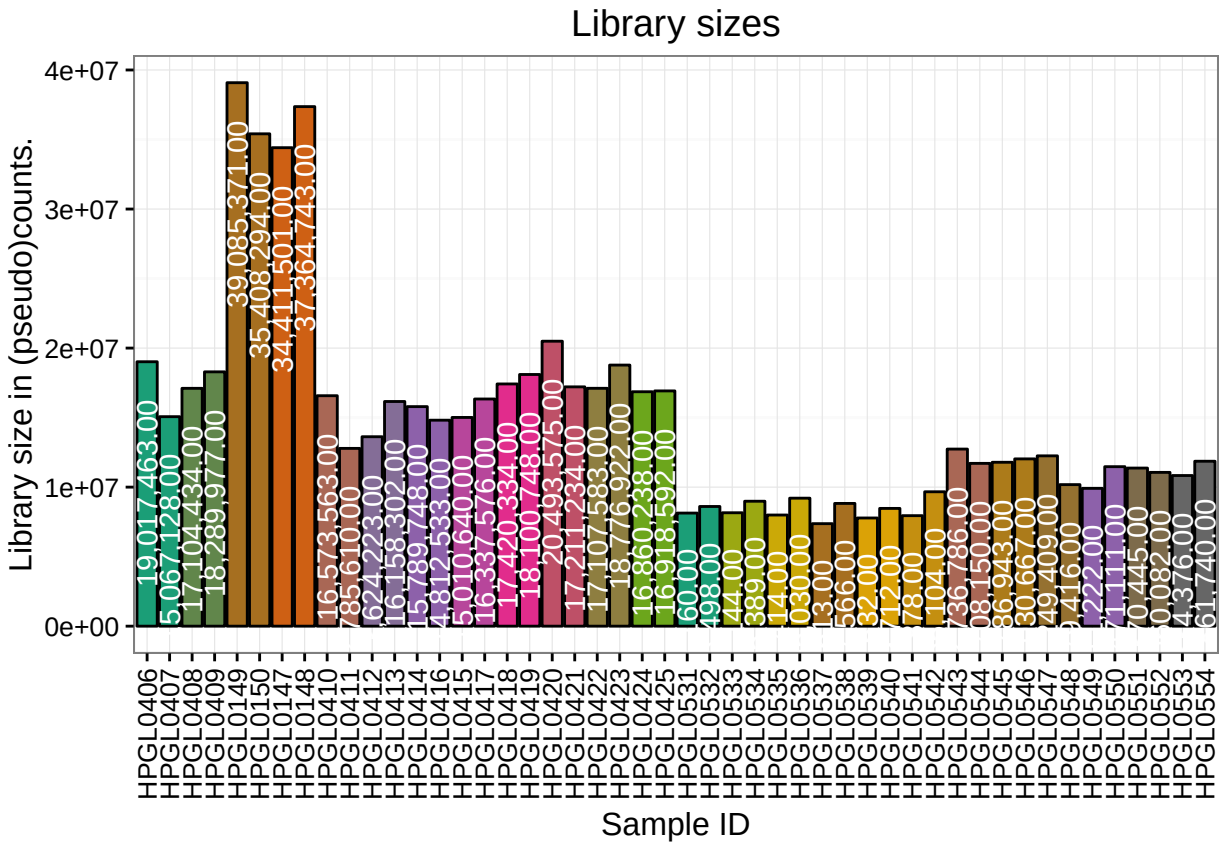



```
## Graphing a PCA plot.

## 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!.
```


Standard Median Distance
Median pairwise distance
2.0e+06
8.0e+06
1.4e+07
gene
0
5
10
15

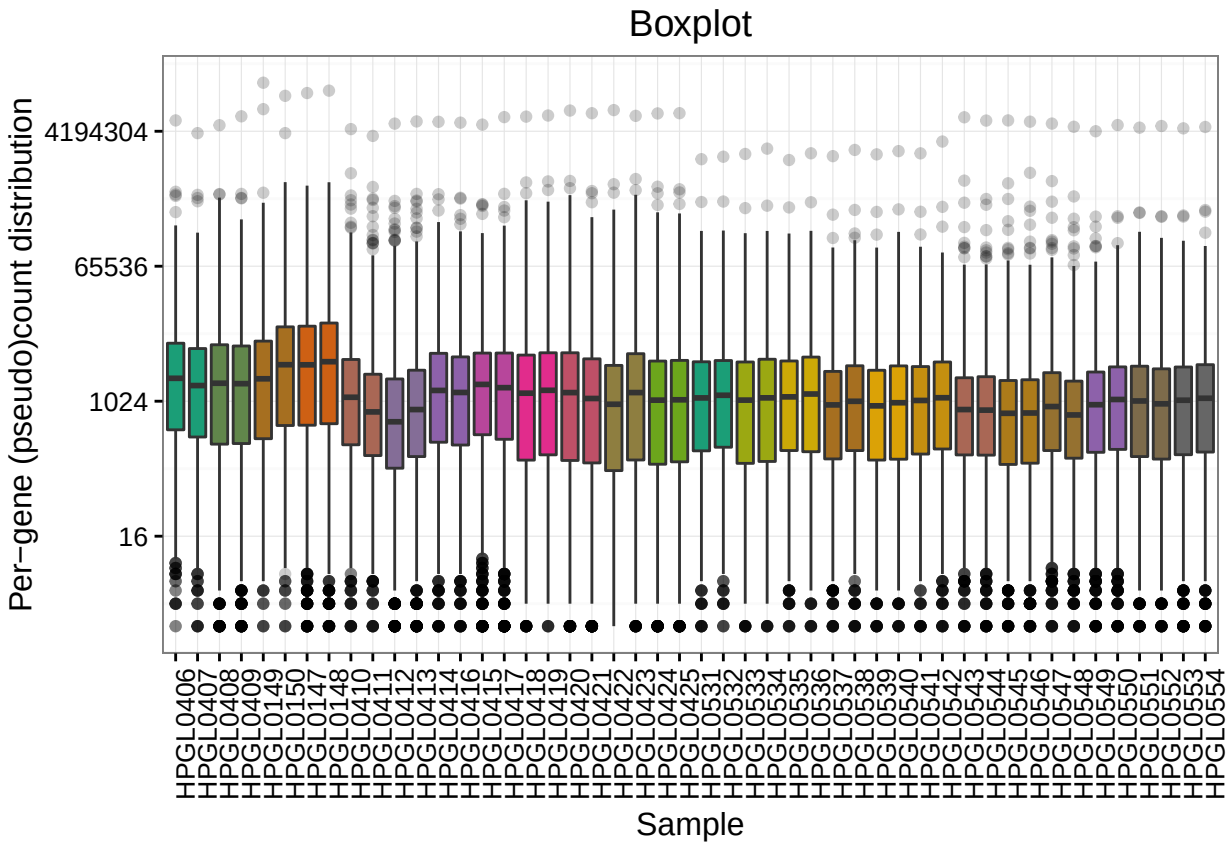


```
## View a raw library size plot
raw_metrics$libsize
```

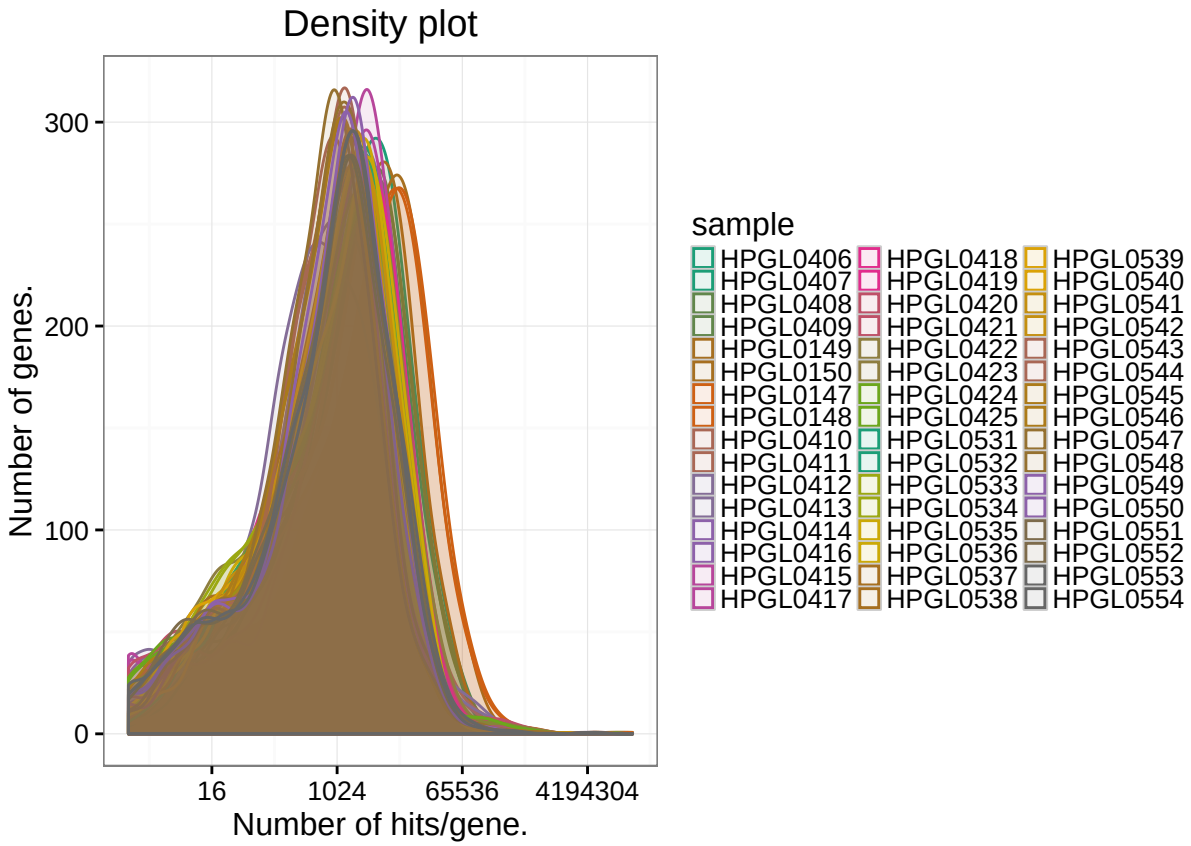



```
## Or boxplot to see the data distribution
raw_metrics$boxplot
```

```
## Warning: Removed 4035 rows containing non-finite values (stat_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$qq
```

```
## NULL
```

```
## 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       3     recordedplot list
## disheat   3     recordedplot list
## smd       3     recordedplot list
## pcaplot  10     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.

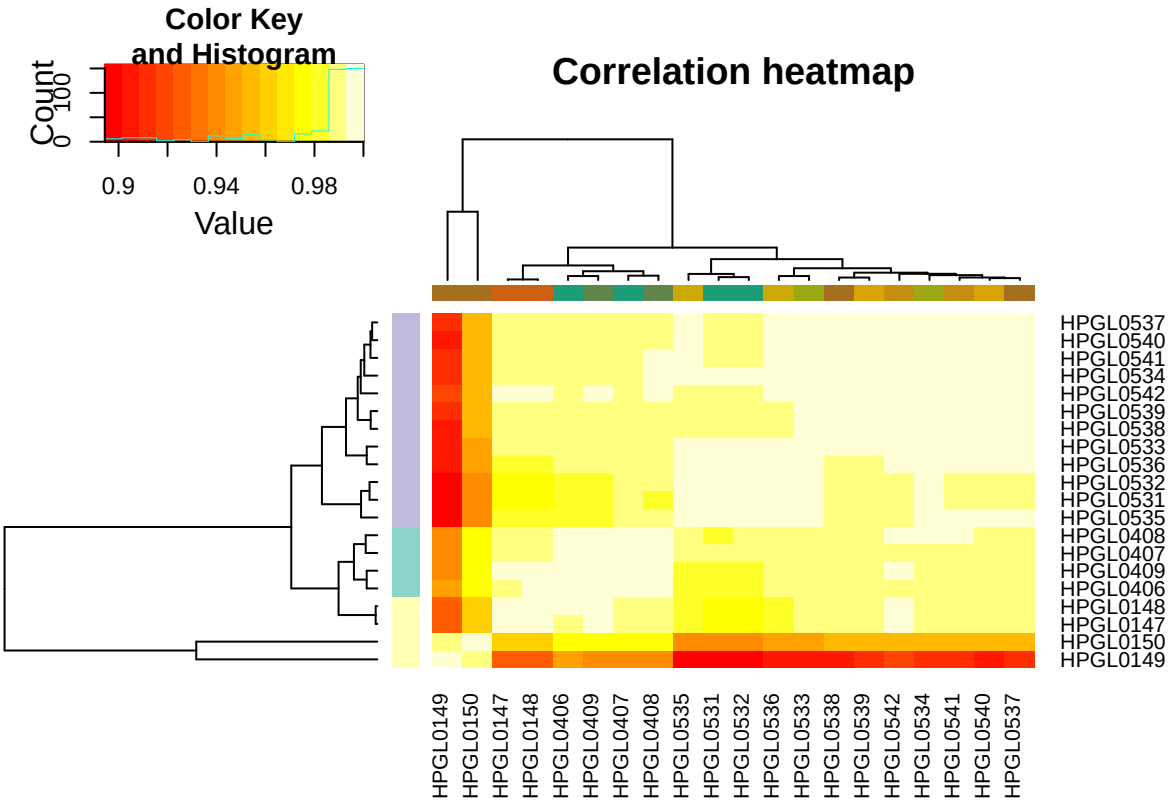
Once we have a smaller data set, we can more easily use PCA to see how the sample separate.

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

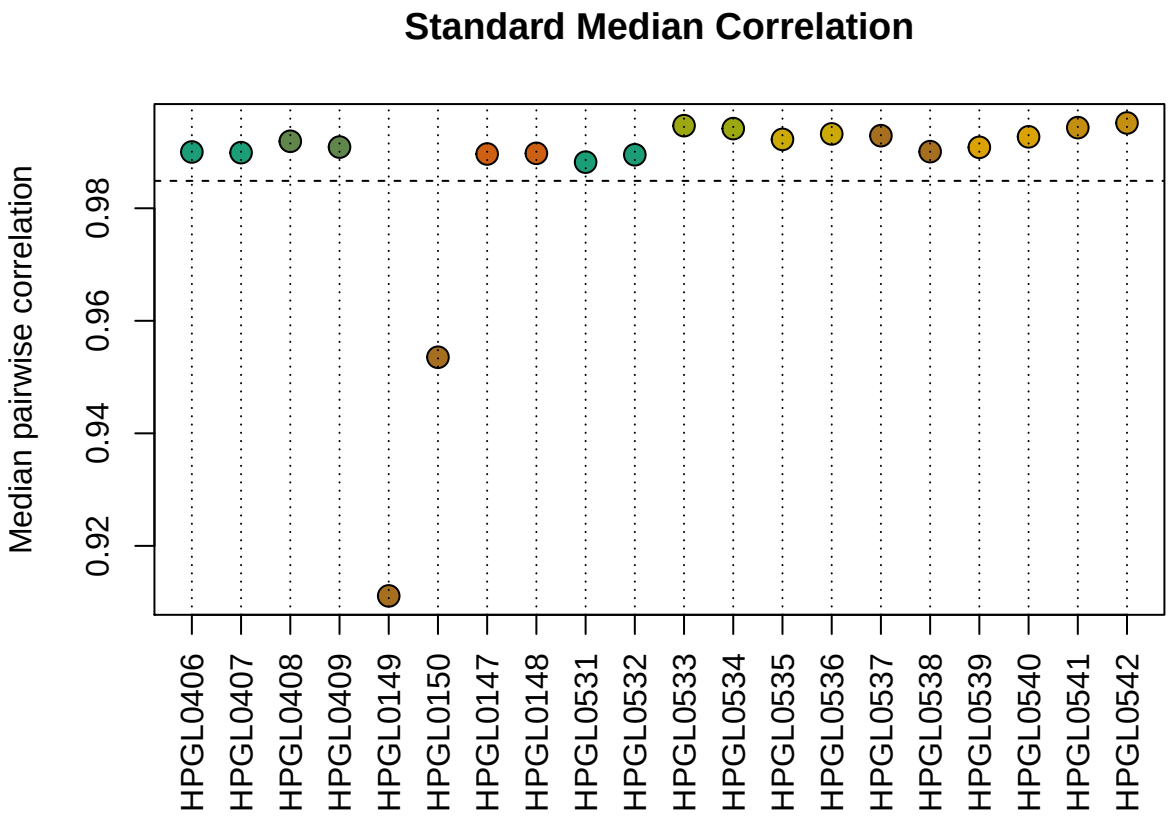
```
## NULL
```

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

elt_metrics <- graph_metrics(elt)
```

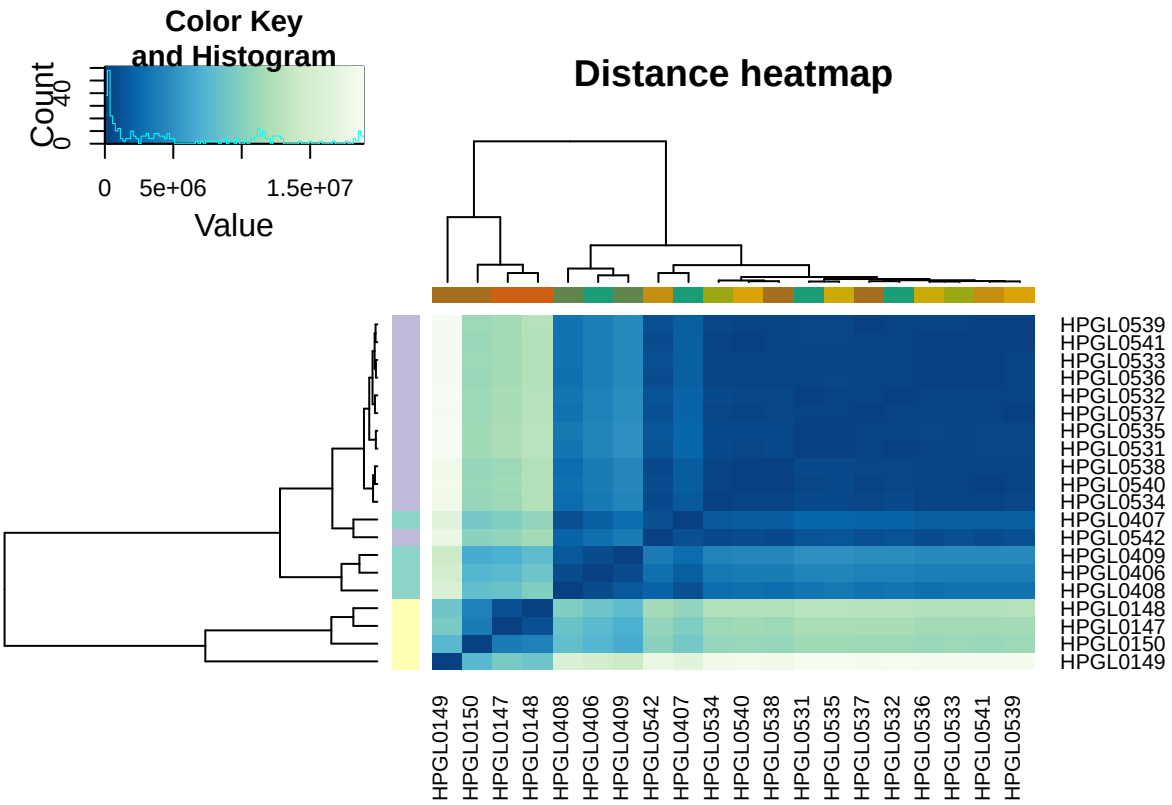
```
## Graphing number of non-zero genes with respect to CPM by library.

## 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.
```



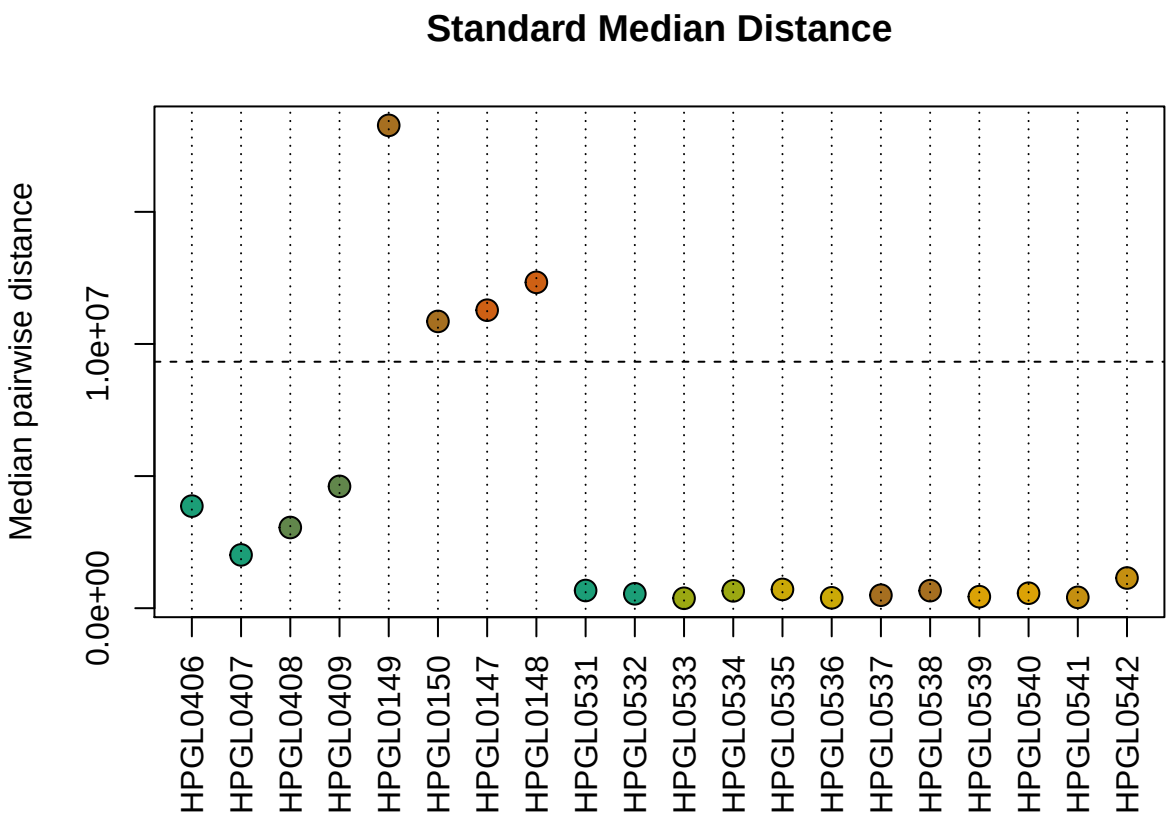
```
## Graphing a distance heatmap.
```



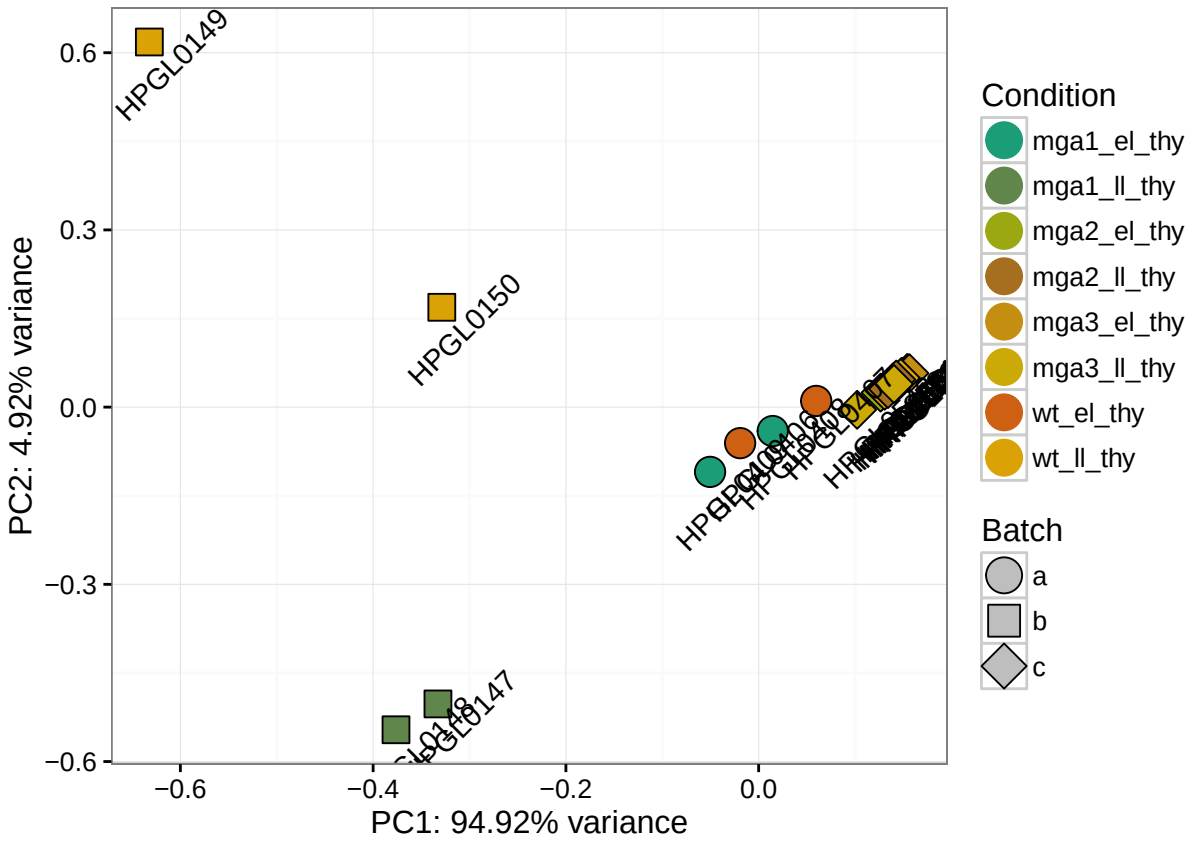

```
## Graphing a standard median distance.
```




```
## 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   94.92      94.92   54.56    92.82
## 2    4.92      99.84   75.93     4.50
## 3    0.09      99.93   57.71    62.89
## 4    0.02      99.95   28.87    35.35
## 5    0.02      99.97   66.35     0.71
## 6    0.01      99.98   61.77     2.25
```

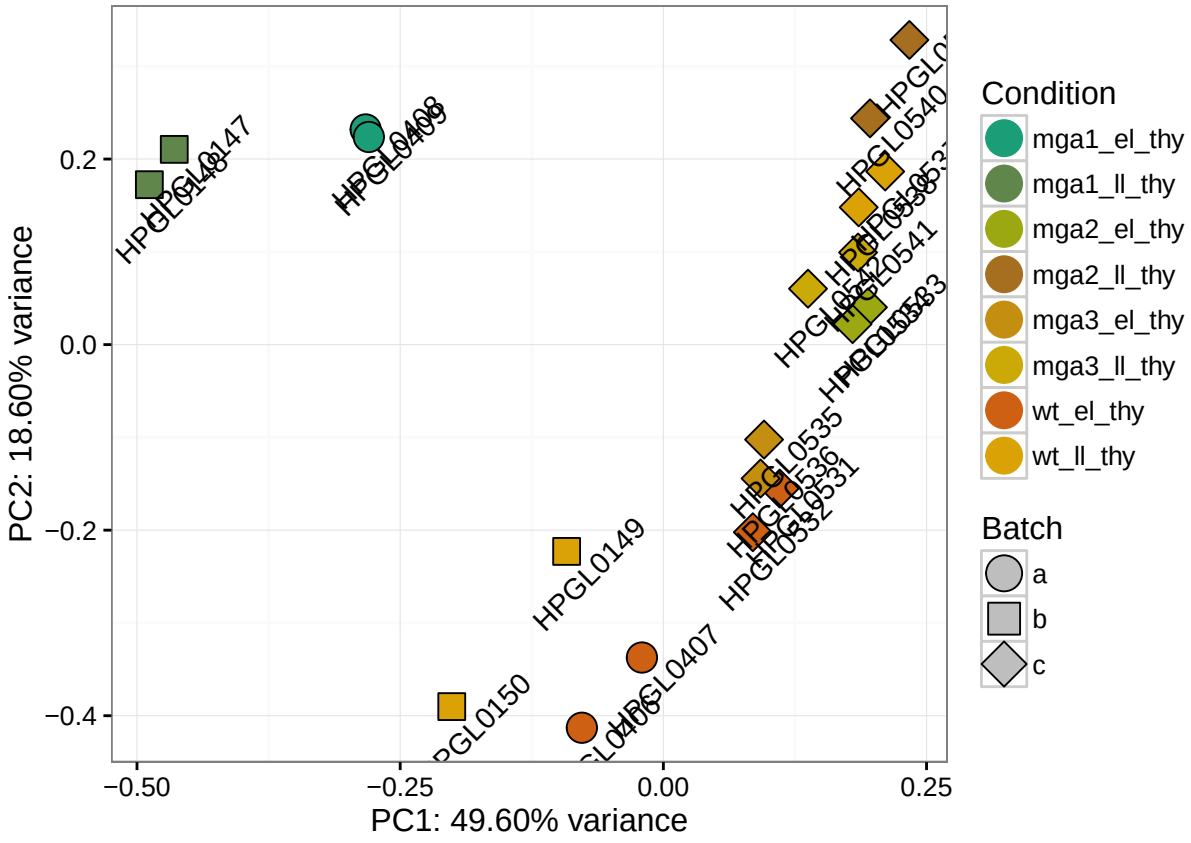
It is pretty obvious that the raw data is a bit jumbled according to PCA. This is not paricularly suprising since we didn't normalize it at all.

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

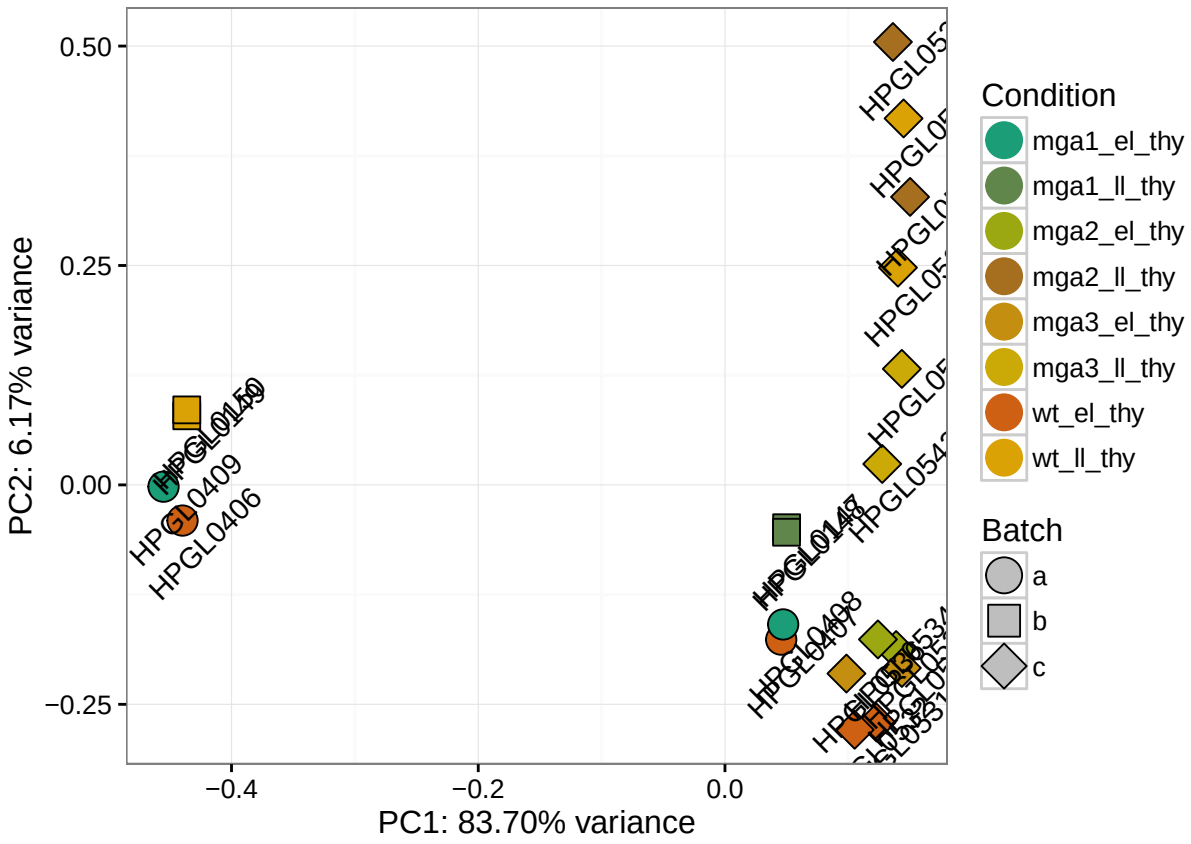
```
## This function will replace the expt$expressionset slot with:

## 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 1519 values equal to 0, adding 1 to the matrix.
```

```
hpgl_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 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_pca(norm_test)$plot
```

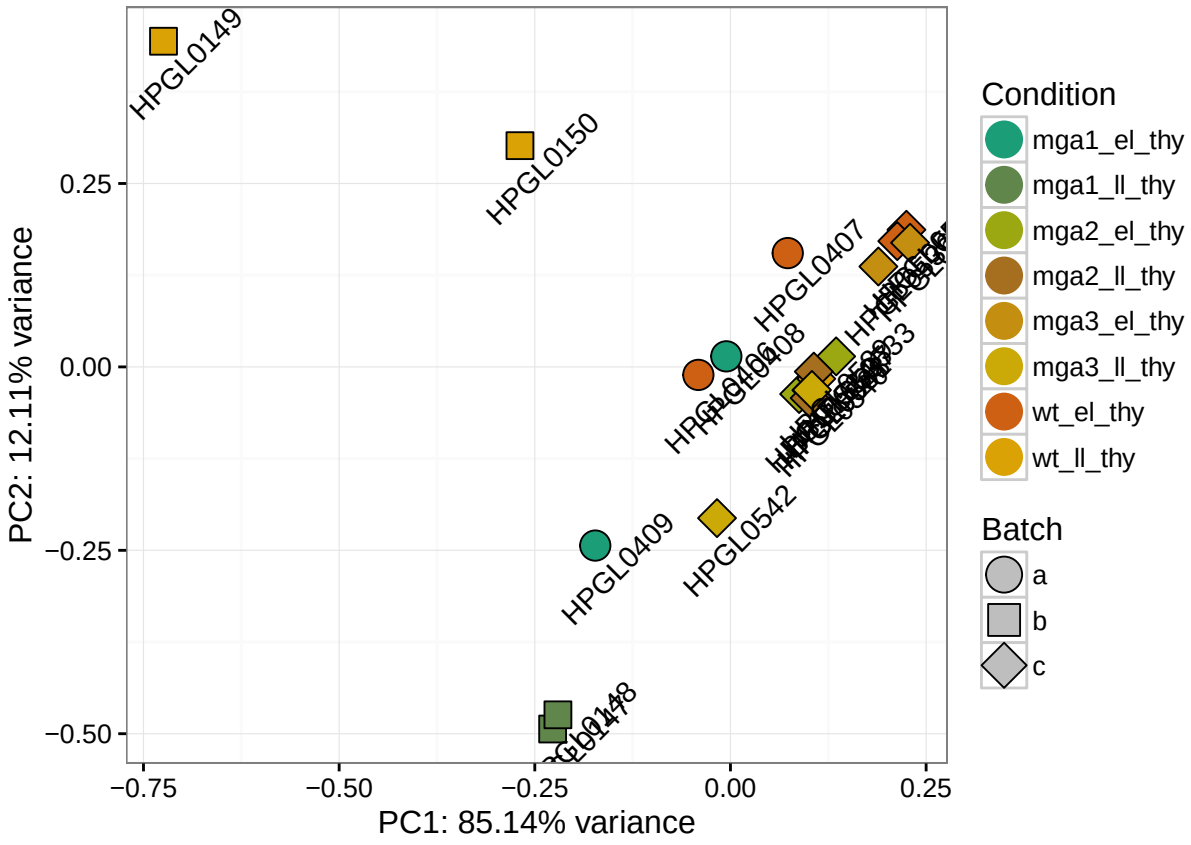



```
## cpm alone brings out some samples, too
norm_test <- normalize_expt(elt, convert="cpm")
```

```
## 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 don't like 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.
```

```
hpgl_pca(norm_test)$plot
```

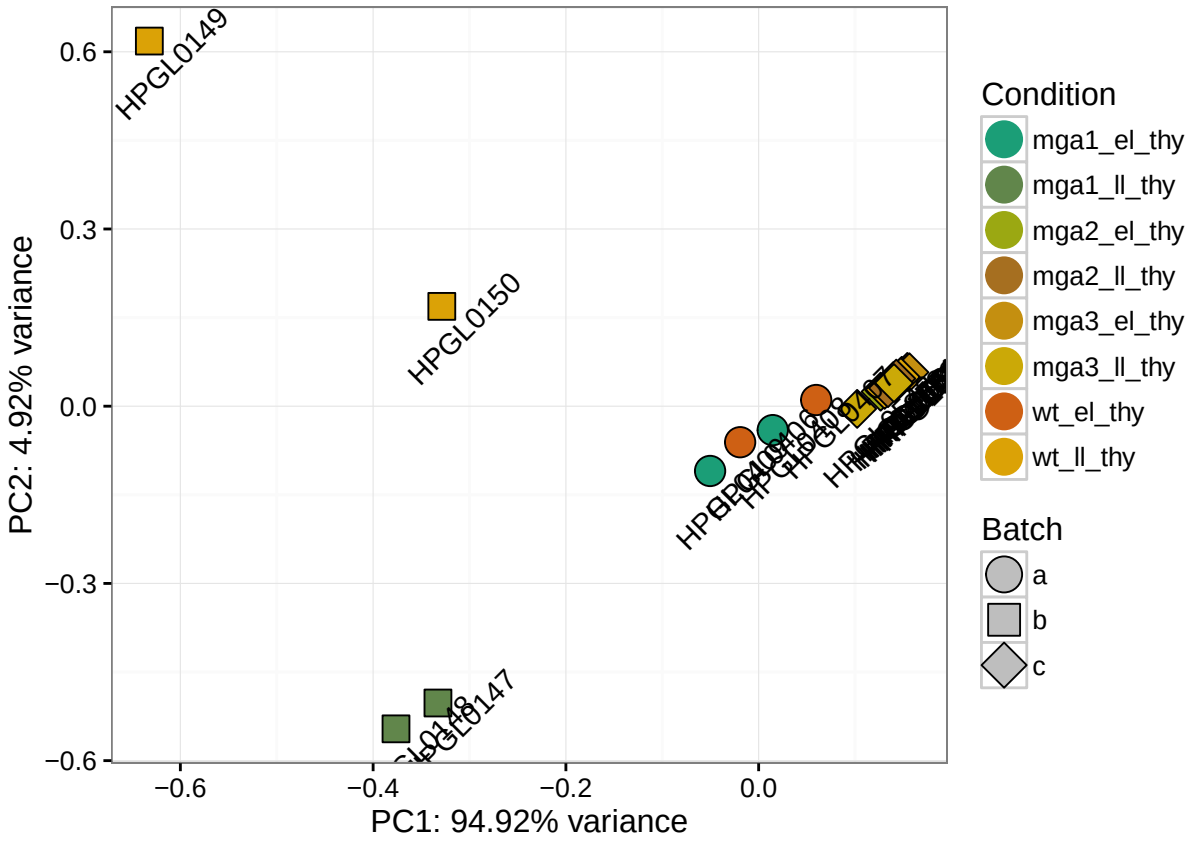



```
## low count filtering has some effect, too
norm_test <- normalize_expt(elt, filter_low="pofa")
```

```
## 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 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).

## 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).
```

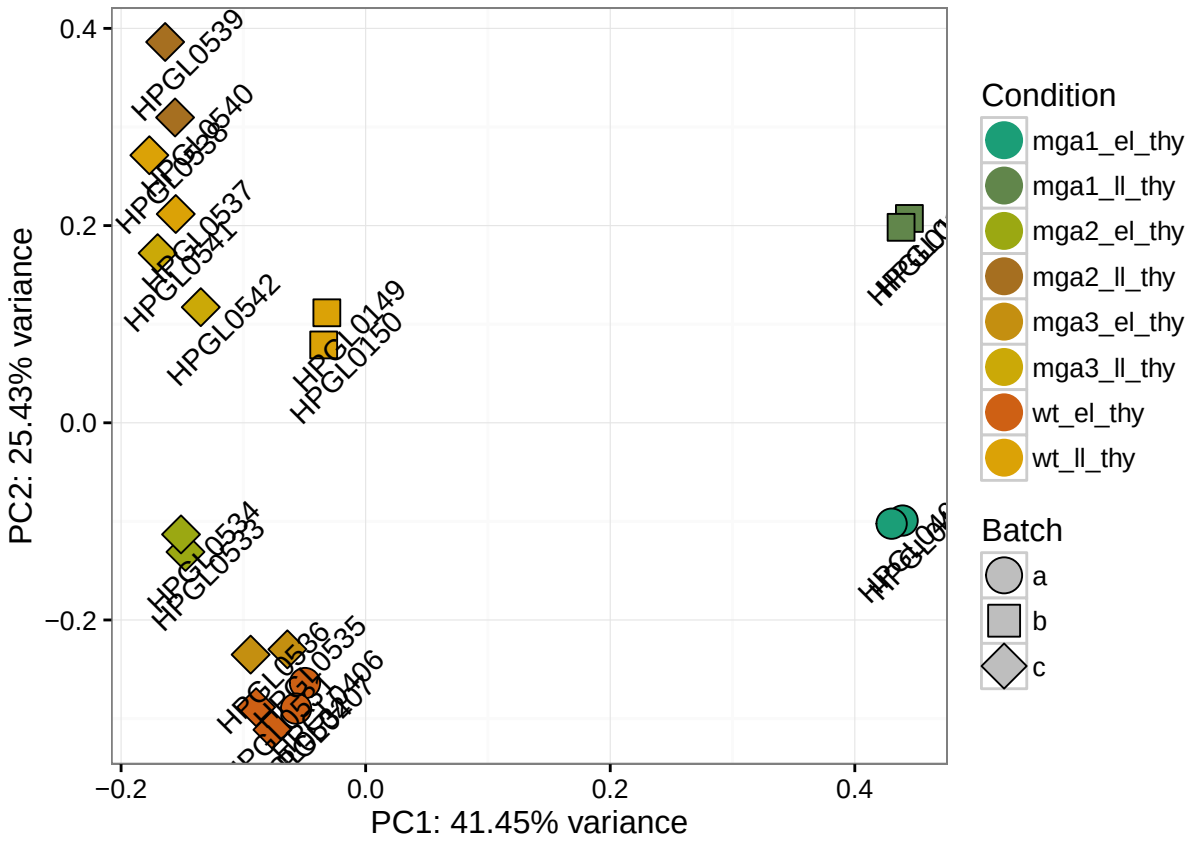
```
hpgl_pca(norm_test)$plot
```



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

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

## log2(quant(cpm(low-filter(batch-correct(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 61 low-count genes (1870 remaining).

## batch_counts: Before batch correction, 2595 entries 0<x<1.

## batch_counts: Using sva::svaseq for batch correction.

## Note to self:  If you feed svaseq a data frame you will get an error like:

## data %*% (Id - mod %*% blah blah requires numeric/complex arguments.

## Warning in hpgl_norm(current_data, design = design, transform =
## transform, : The batch_counts called failed. Returning non-batch reduced
## data.

## Applying: log2 transformation.
```

```
hpgl_pca(norm_test)$plot
```



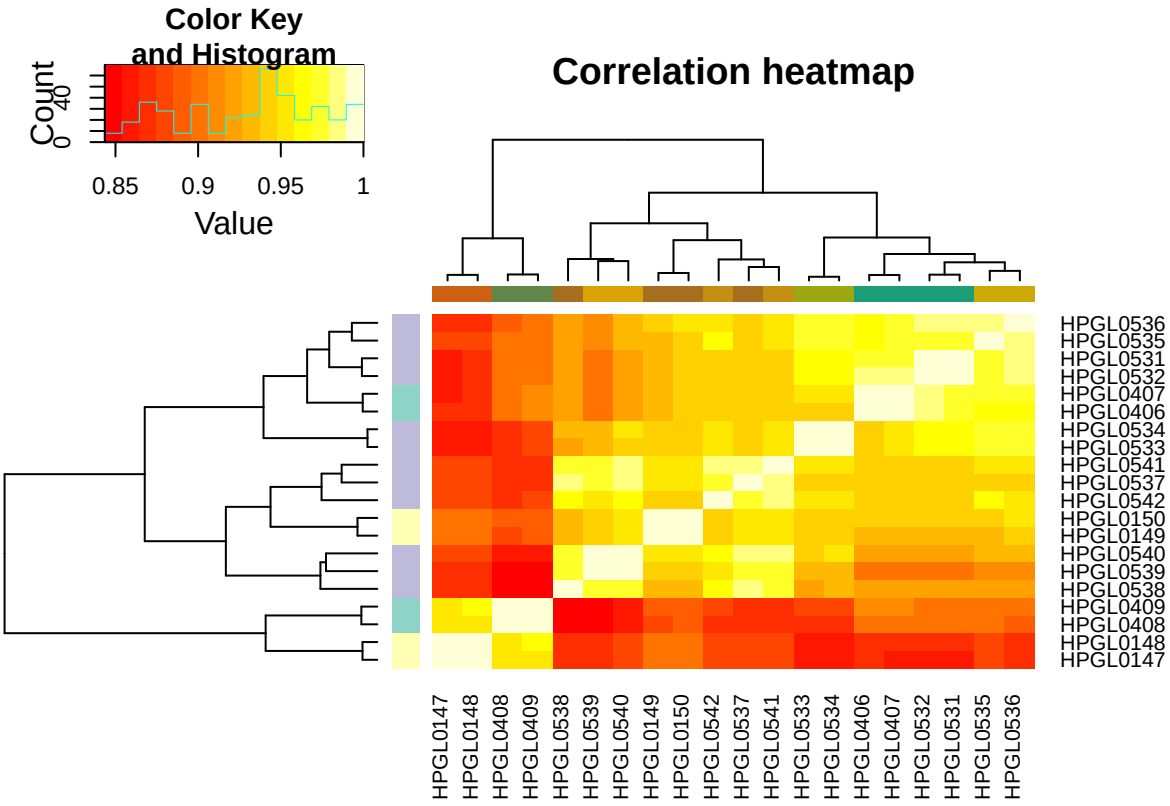

```
## The different batch effect testing methods have a pretty widely ranging effect on the clustering
## play with them by changing the batch= parameter to:
## "limma", "sva", "svaseq", "limmaresid", "ruvg", "combat", combatmod"

pca_test <- hpgl_pca(norm_test)
head(pca_test$res)
```

```
##   propVar cumPropVar cond.R2 batch.R2
## 1   41.45      41.45   97.99    51.98
## 2   25.43      66.88   97.05    23.36
## 3    8.02      74.90   59.13    35.72
## 4    7.95      82.85   56.72    43.74
## 5    3.87      86.72   94.87    19.40
## 6    3.09      89.81   62.67    11.32
```

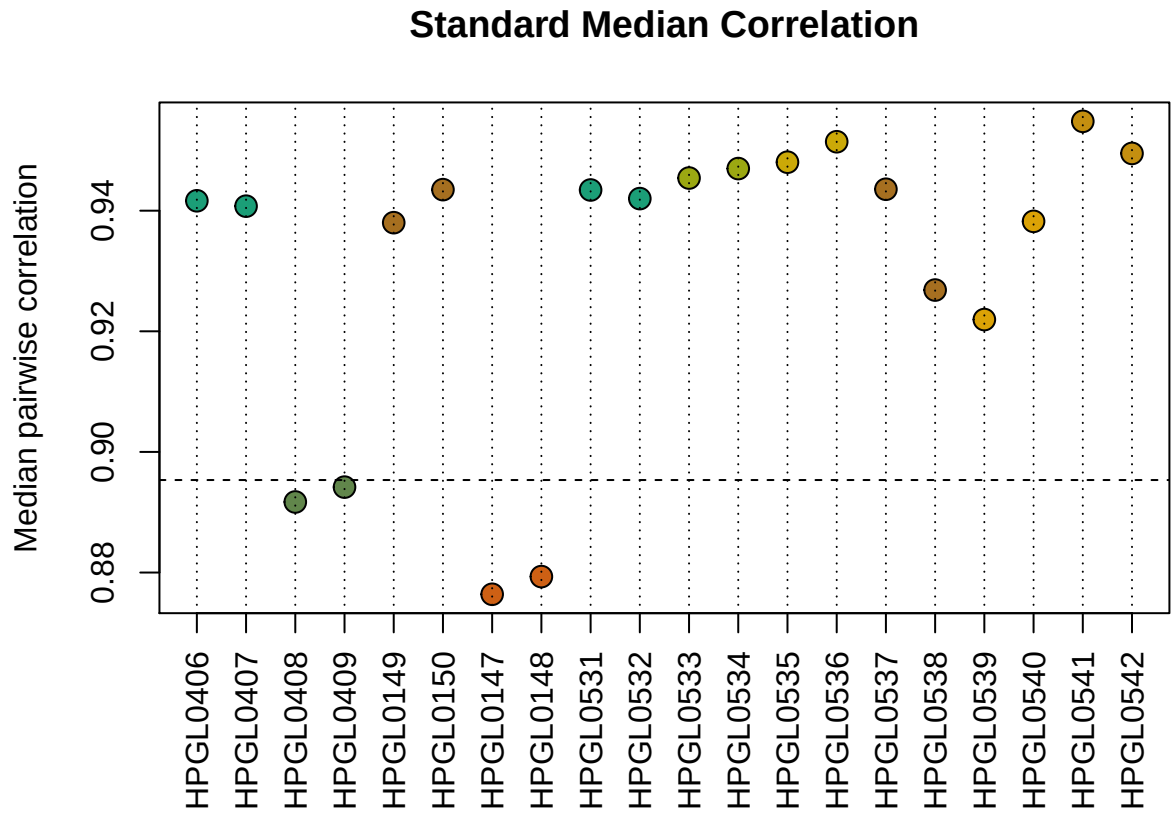
```
## Thus we see a dramatic decrease in variance accounted for
## by batch after applying limma's 'removebatcheffect'
## (see batch.R2 here vs. above)

## 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.
```

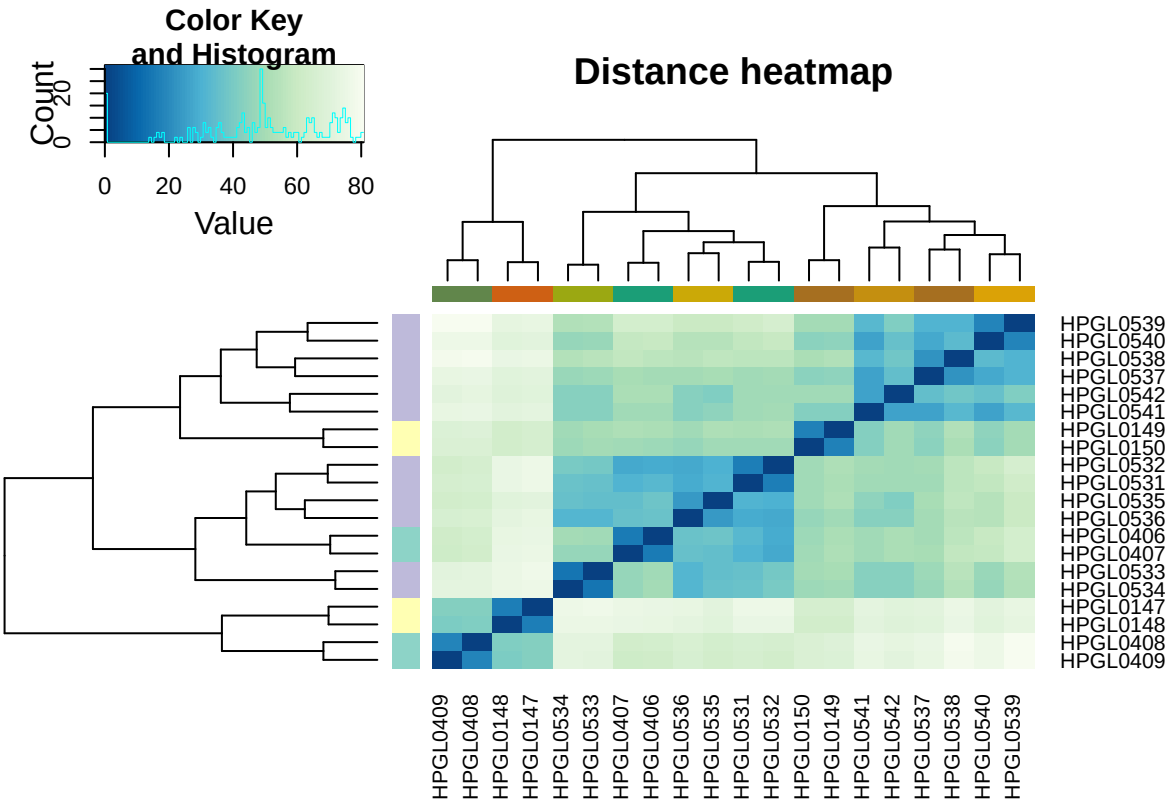
```
## Graphing a boxplot.

## Graphing a correlation heatmap.

## Graphing a standard median correlation.
```



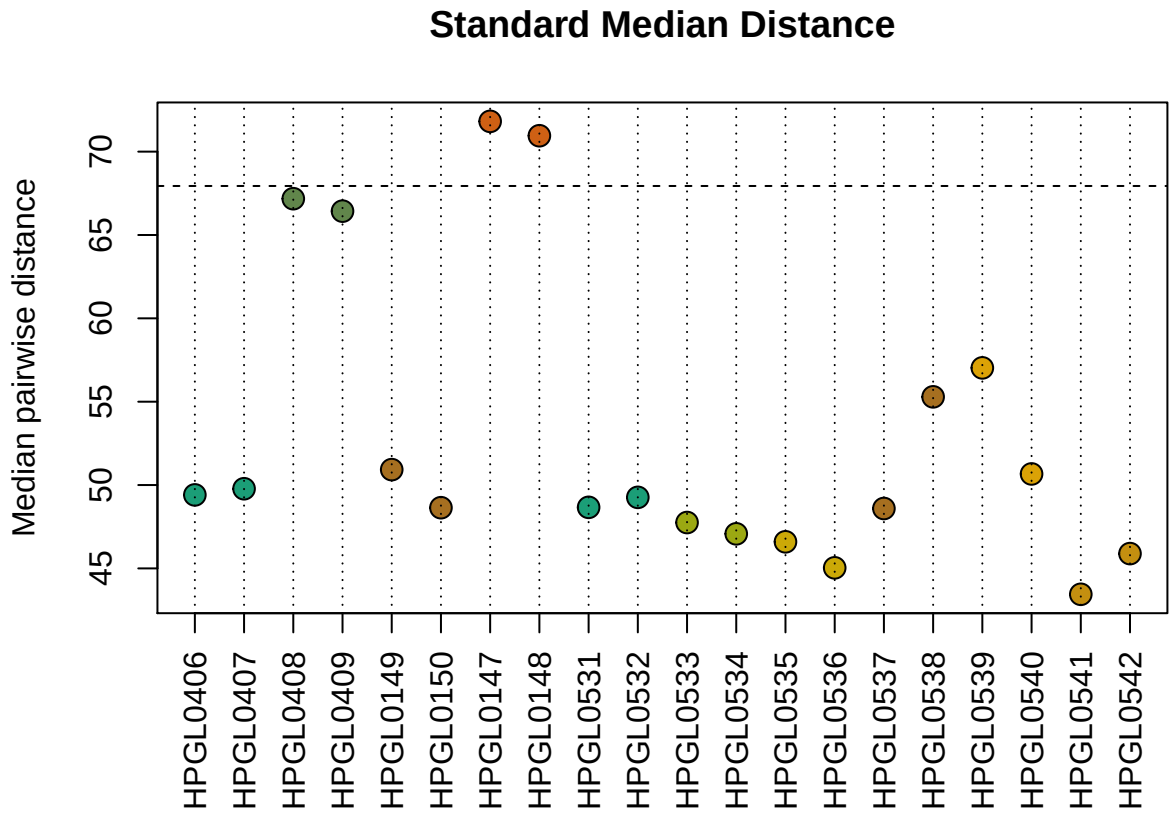

```
## Graphing a distance heatmap.
```



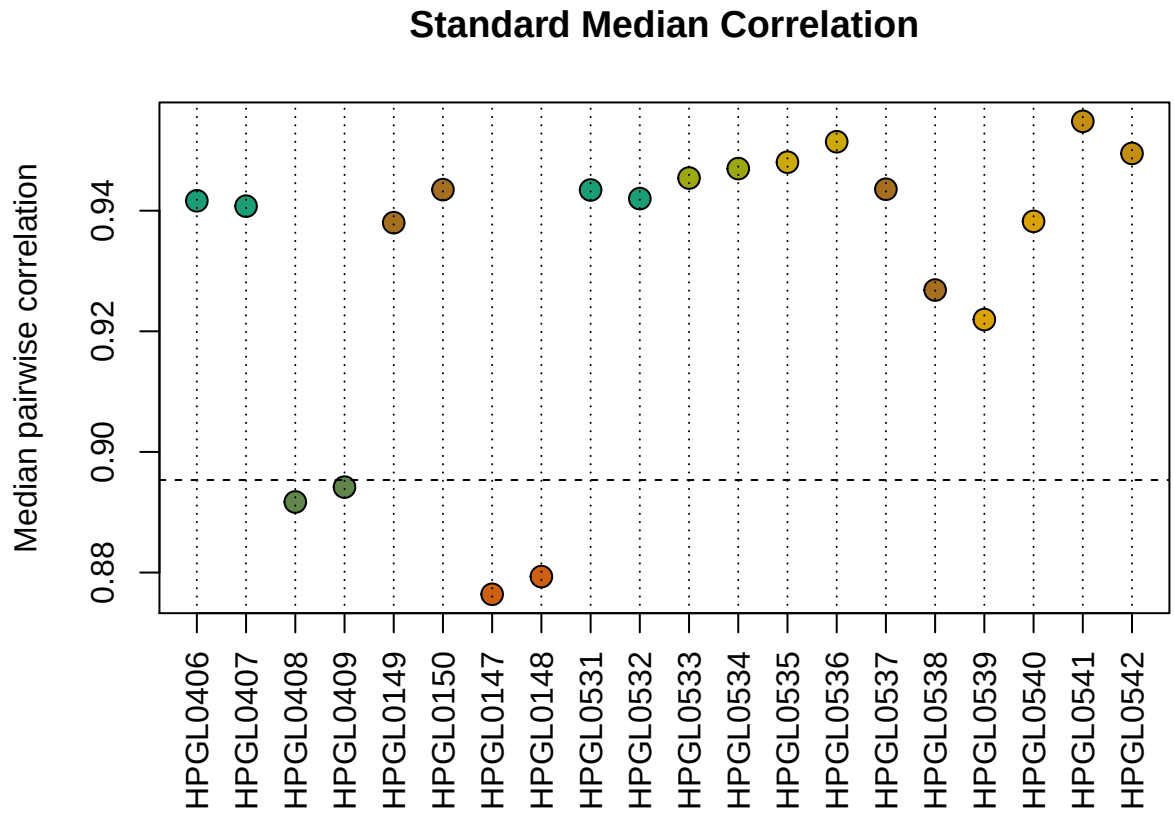

```
## Graphing a standard median distance.
```



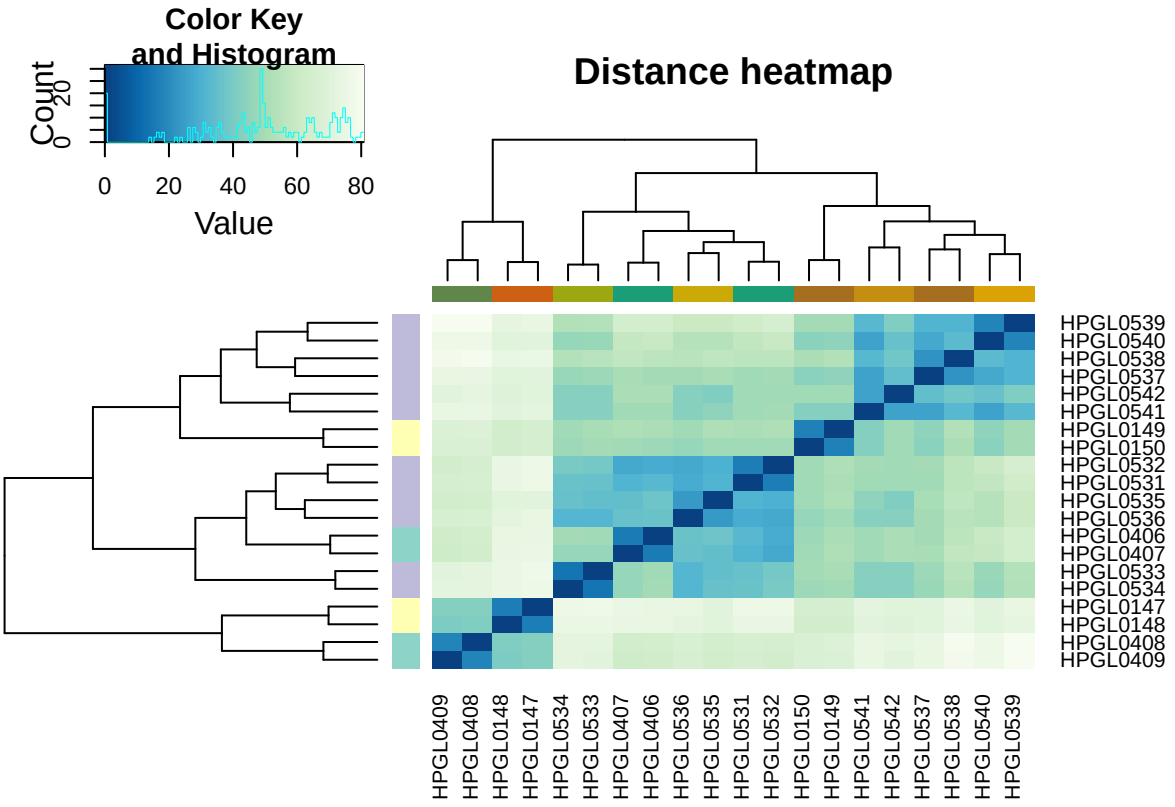
```
## Graphing a PCA plot.

## 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.
```