

hpgltools usage

atb abelew@gmail.com

2016-03-09

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
3. Gene ontology/KEGG 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"    "annotations"
## [4] "basic_comparison" "batchnorm_expt"     "color_hash"
## [7] "colors"            "compare_12"         "condition_list"
## [10] "condition_names"   "count_dataframe"    "counts"
## [13] "data_file"          "deseq_comparison"   "design_colors_list"
## [16] "early_late"         "early_late_thy"      "edger_comparison"
## [19] "elt"                "elt_metrics"        "elt_norm"
## [22] "empty_samples"      "expt"                "fis_batchnormpca"
## [25] "fis_info"           "fis_libsize"         "fis_nonzero"
```



```

## samplenames      48    -none-     character
## colors          48    -none-     character
## names           48    -none-     character
## filtered        1     -none-     logical
## transform       1     -none-     character
## norm            1     -none-     character
## convert         1     -none-     character
## original_libsize 48    -none-     numeric
## columns          1     data.frame list

```

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.

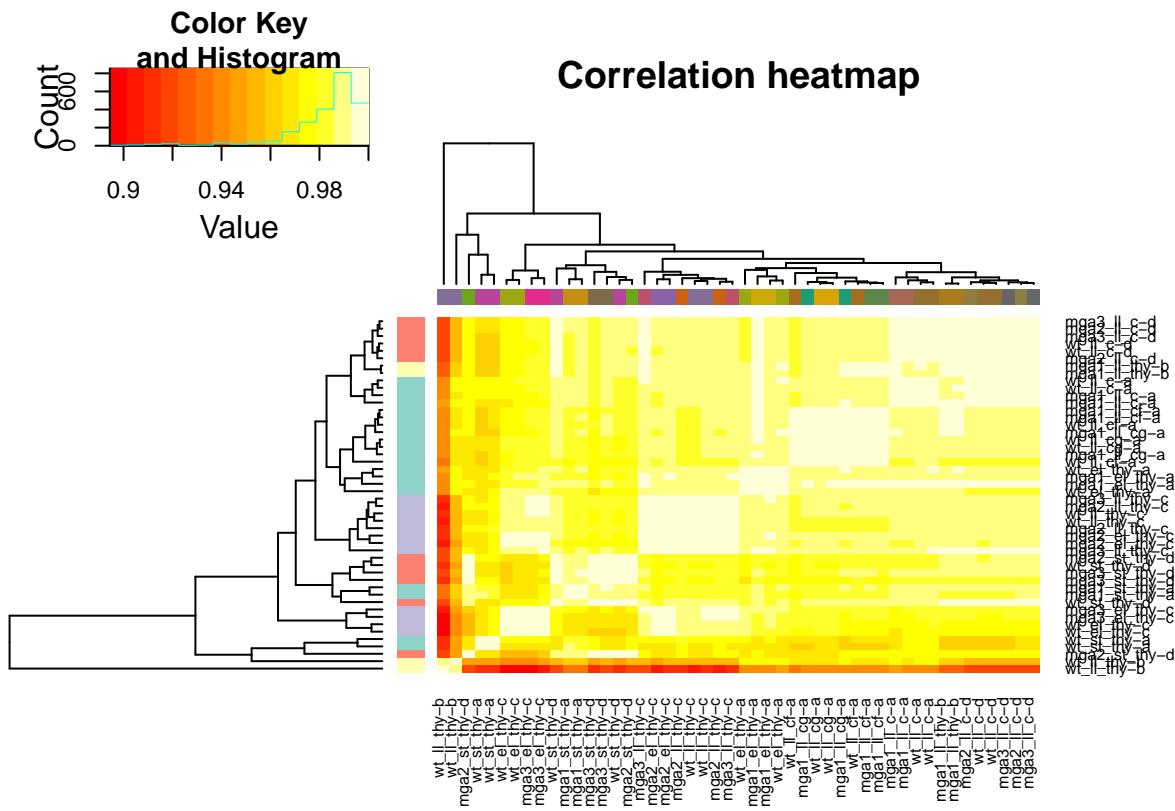
## Graphing a boxplot.

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

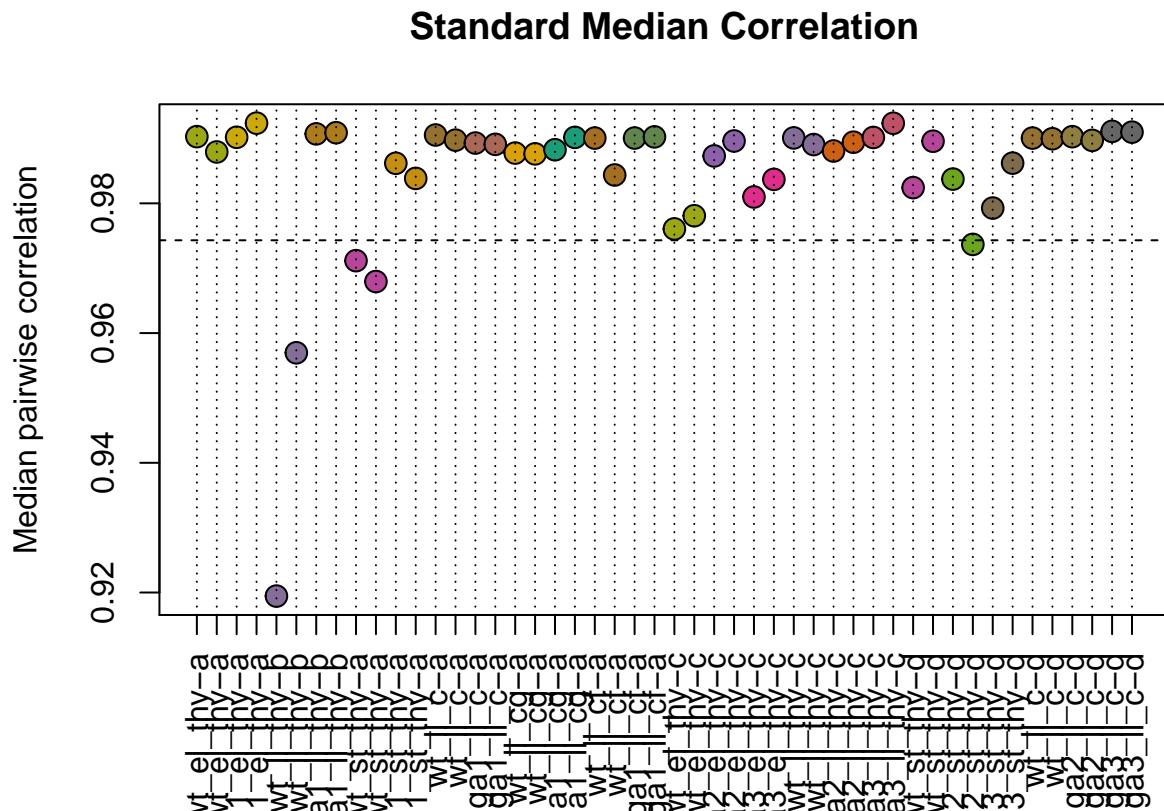
## Graphing a correlation heatmap.

## Graphing a standard median correlation.

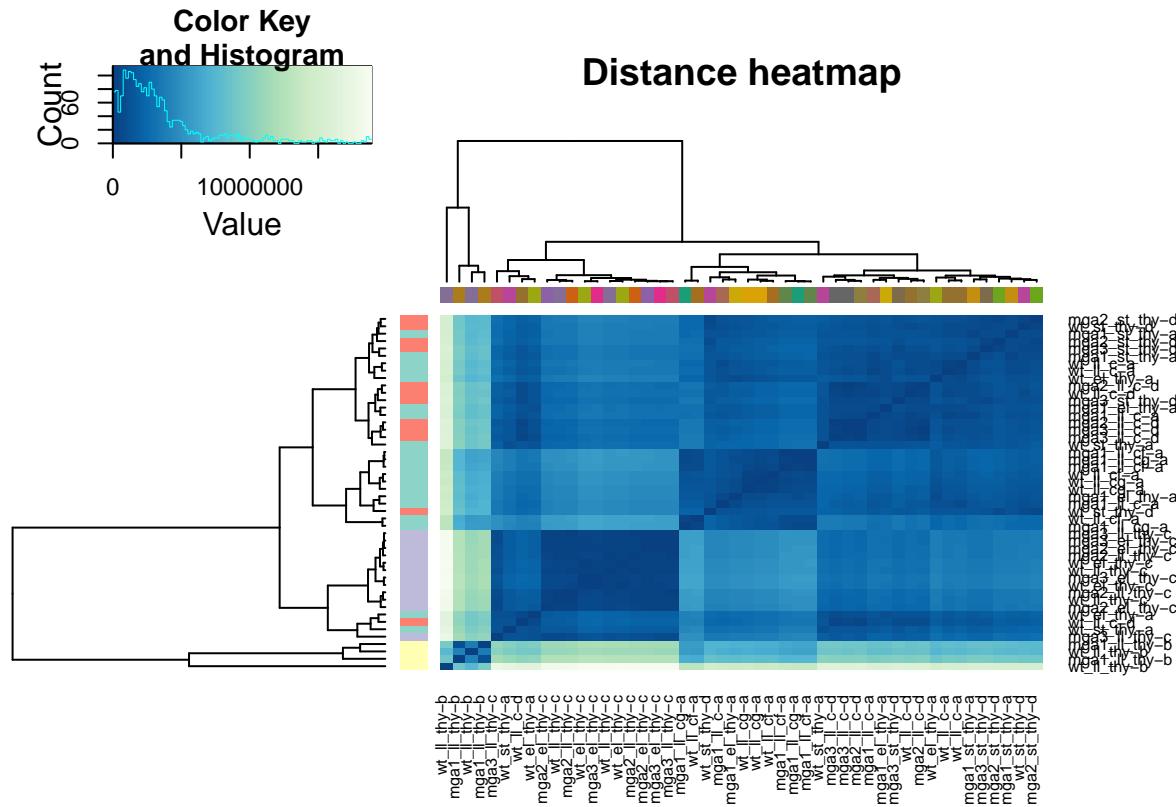
```



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



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



```
## Graphing a PCA plot.
```

```
## Plotting a density plot.
```

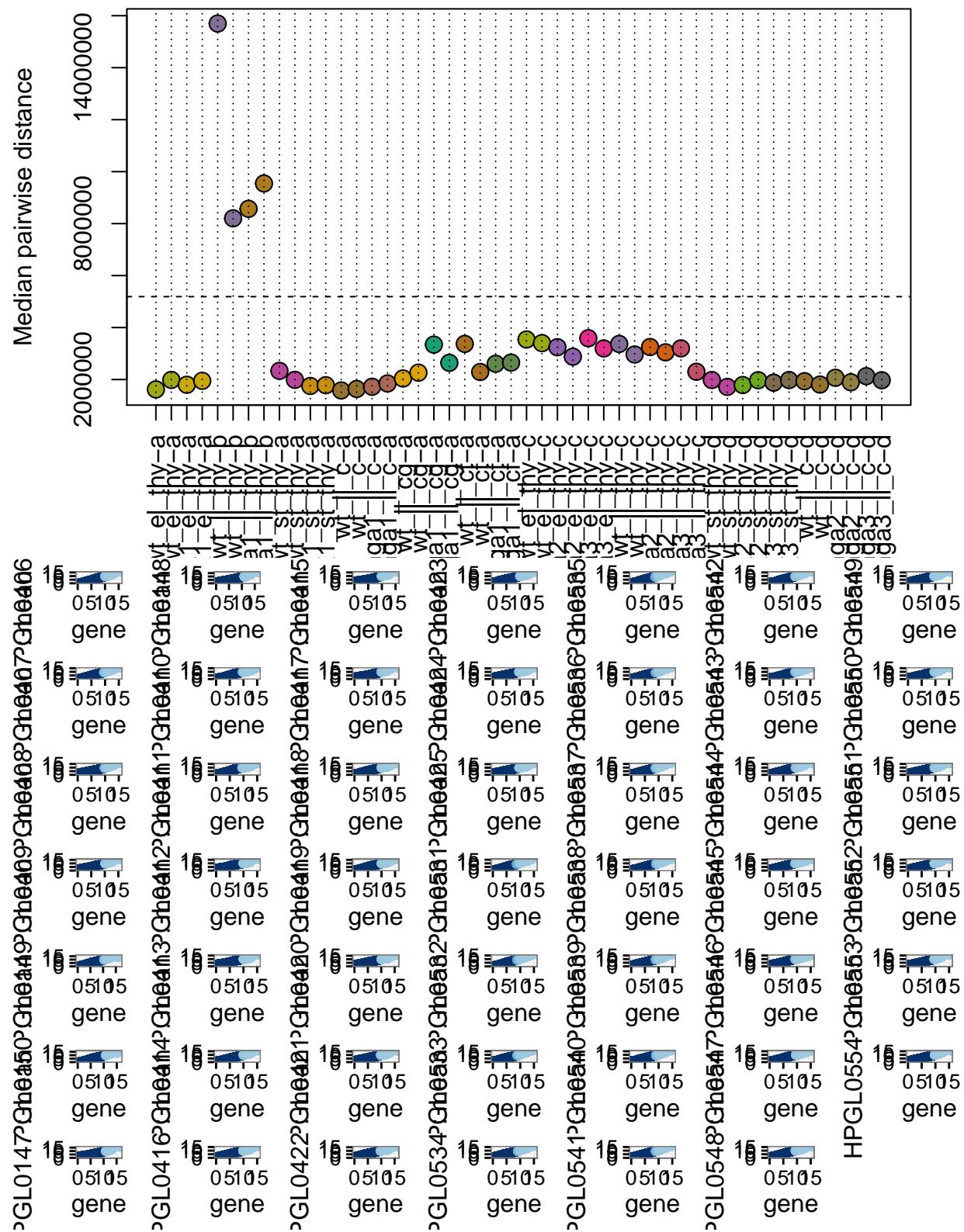
```
## This data will benefit from being displayed on the log scale.
```

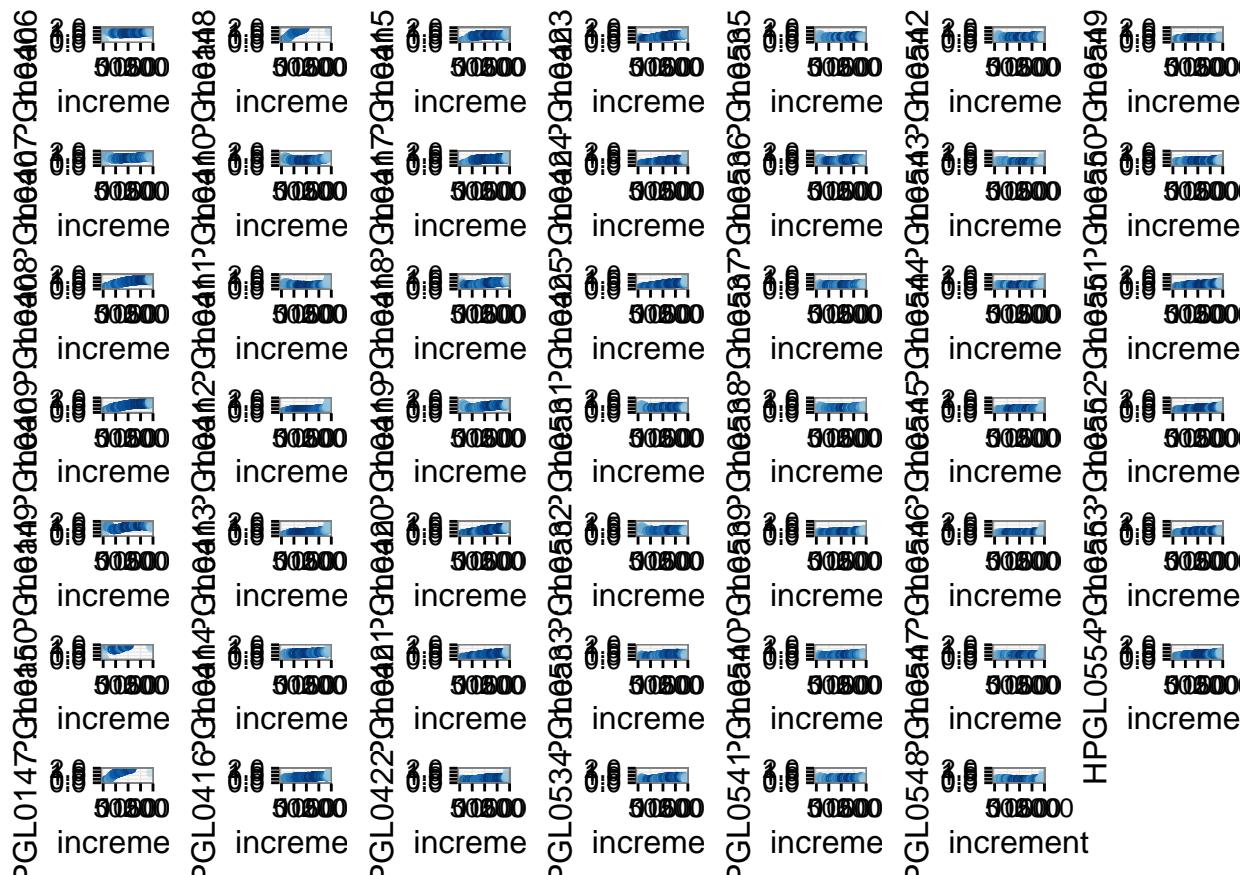
```
## If this is not desired, set scale='raw'
```

```
## No id variables; using all as measure variables
```

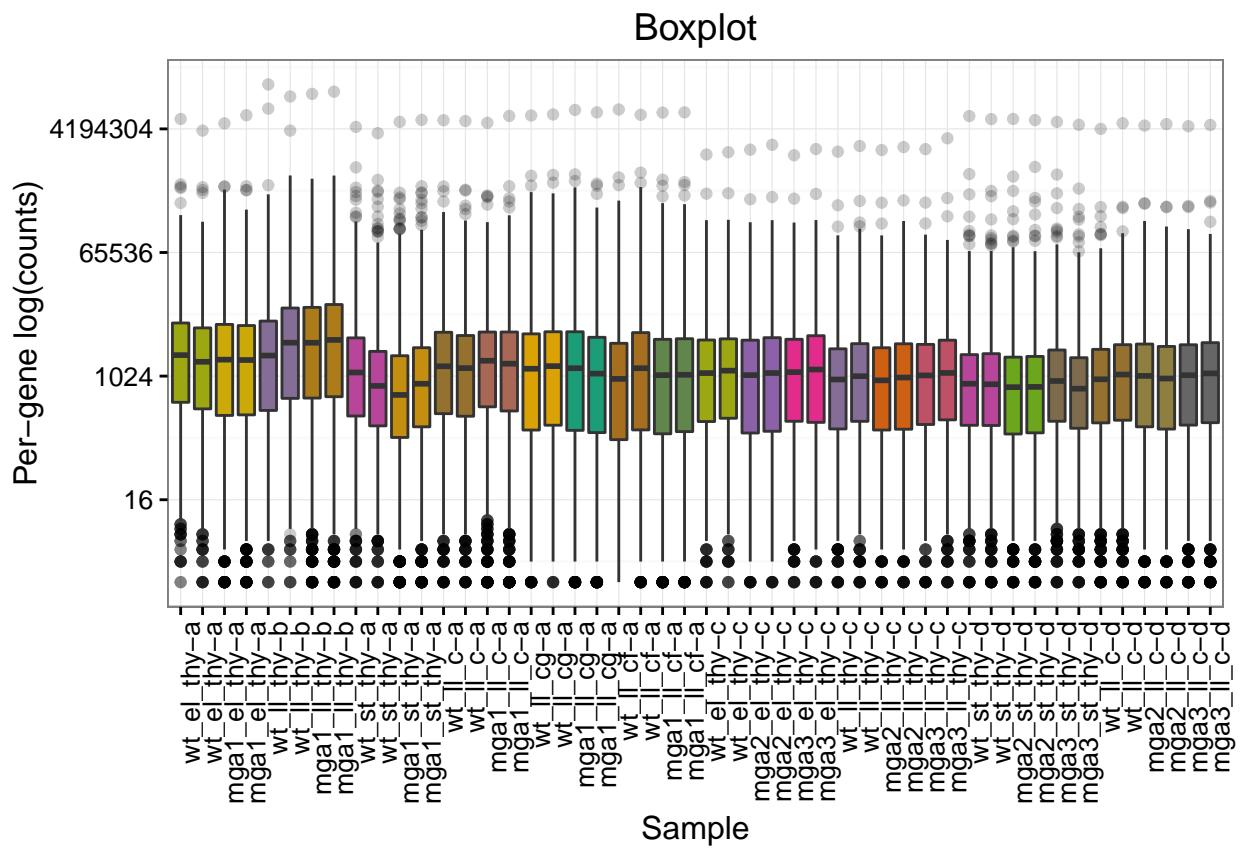
```
## QQ plotting!.
```

Standard Median Distance



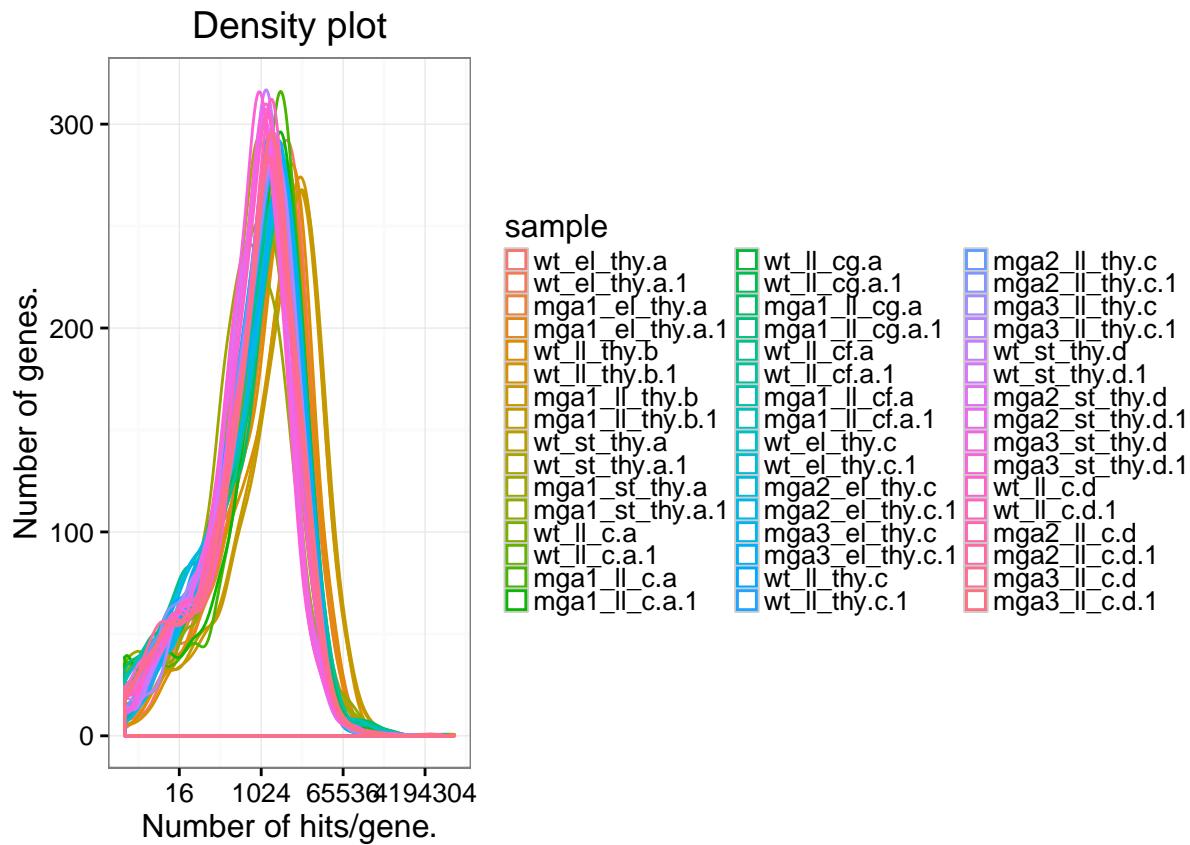


```
## View a raw library size plot
raw_metrics$libsize
```

```
## 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
```

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

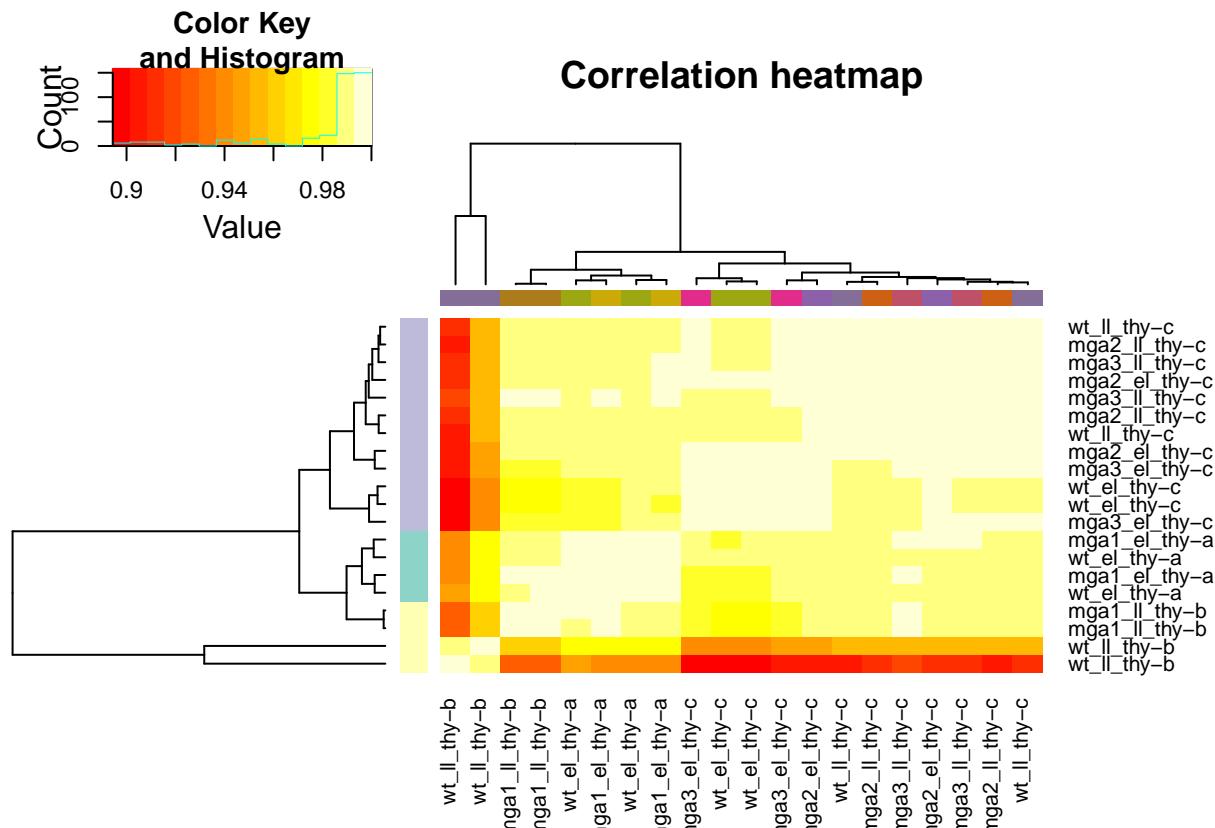
```
##          Length Class      Mode
## nonzero     9    gg     list
## libsize     9    gg     list
## boxplot     9    gg     list
## corheat     3 recordedplot list
## smc         3 recordedplot list
## disheat     3 recordedplot list
## smd         3 recordedplot list
## pcaplot    10    gg     list
## pcatable    8   data.frame  list
## pcares      4   data.frame  list
## pcavar     47   -none-    numeric
## density     9    gg     list
## qqlog       3 recordedplot list
## qrat        3 recordedplot list
## ma          0   -none-    NULL
```



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

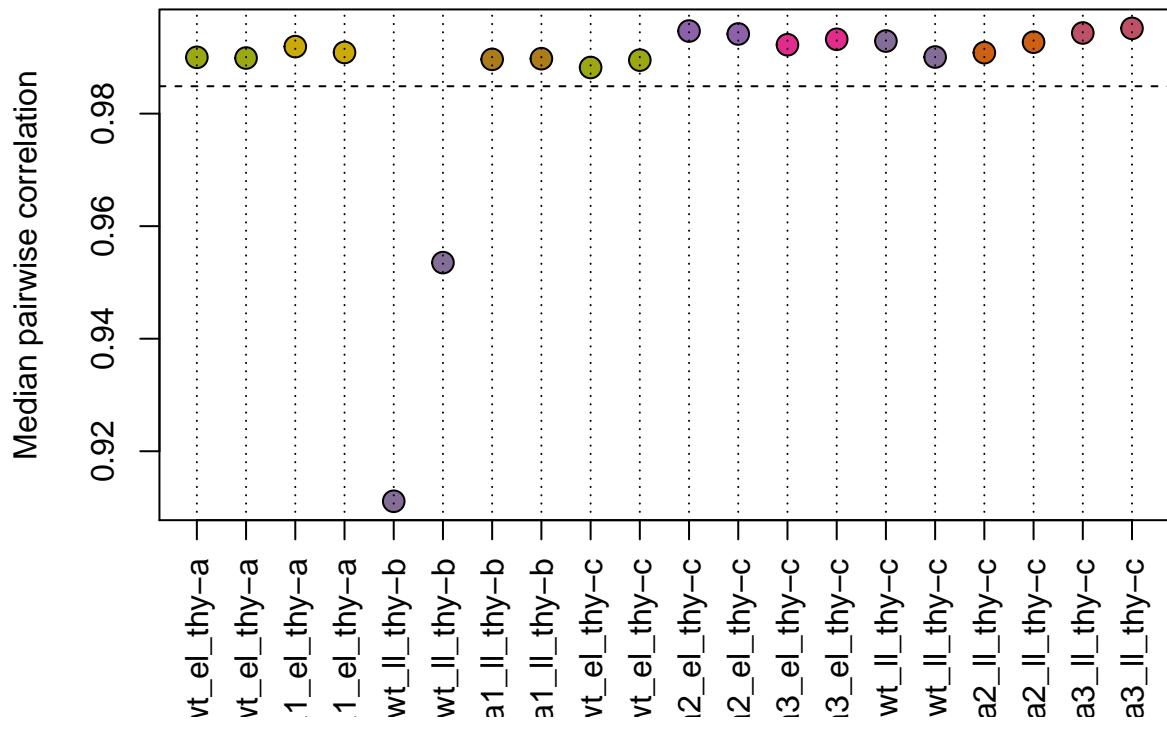
```
## Graphing a correlation heatmap.
```

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

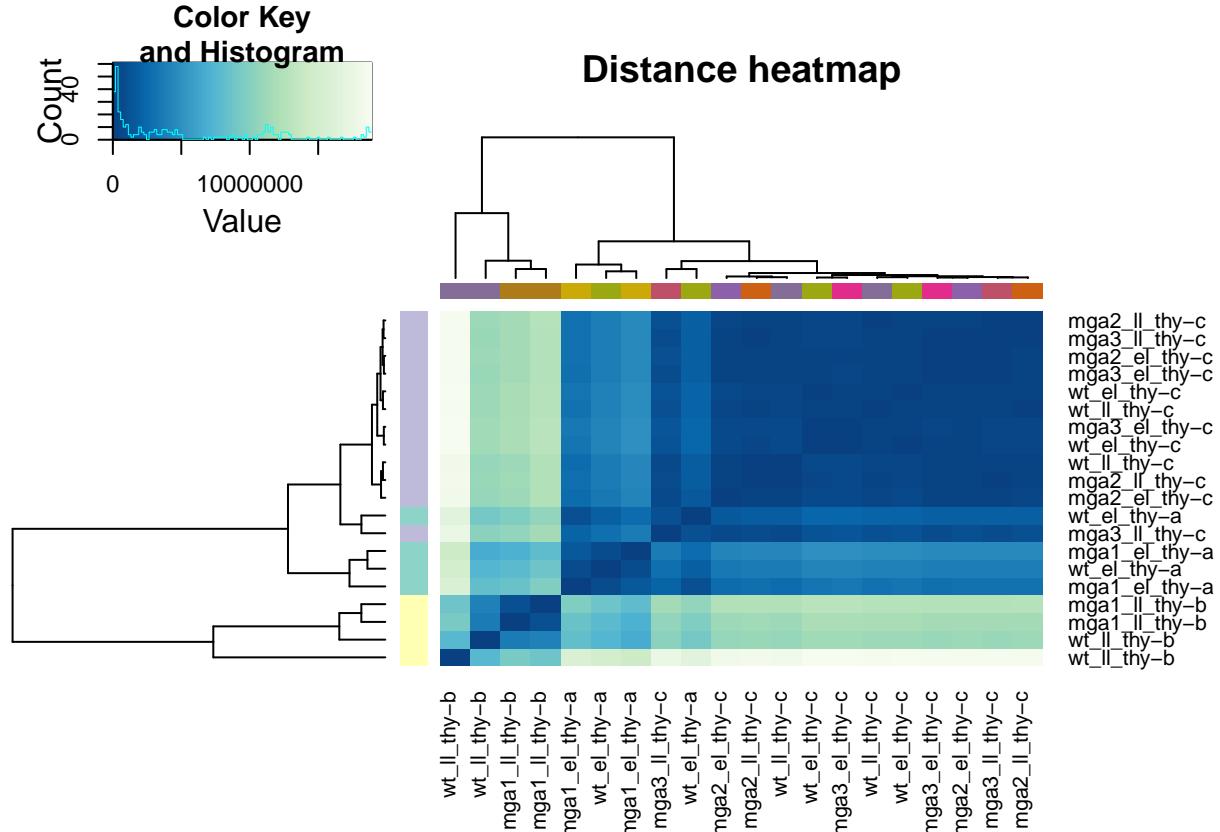


```
## Graphing a distance heatmap.
```

Standard Median Correlation



Graphing a standard median distance.



```

## Graphing a PCA plot.

## Plotting a density plot.

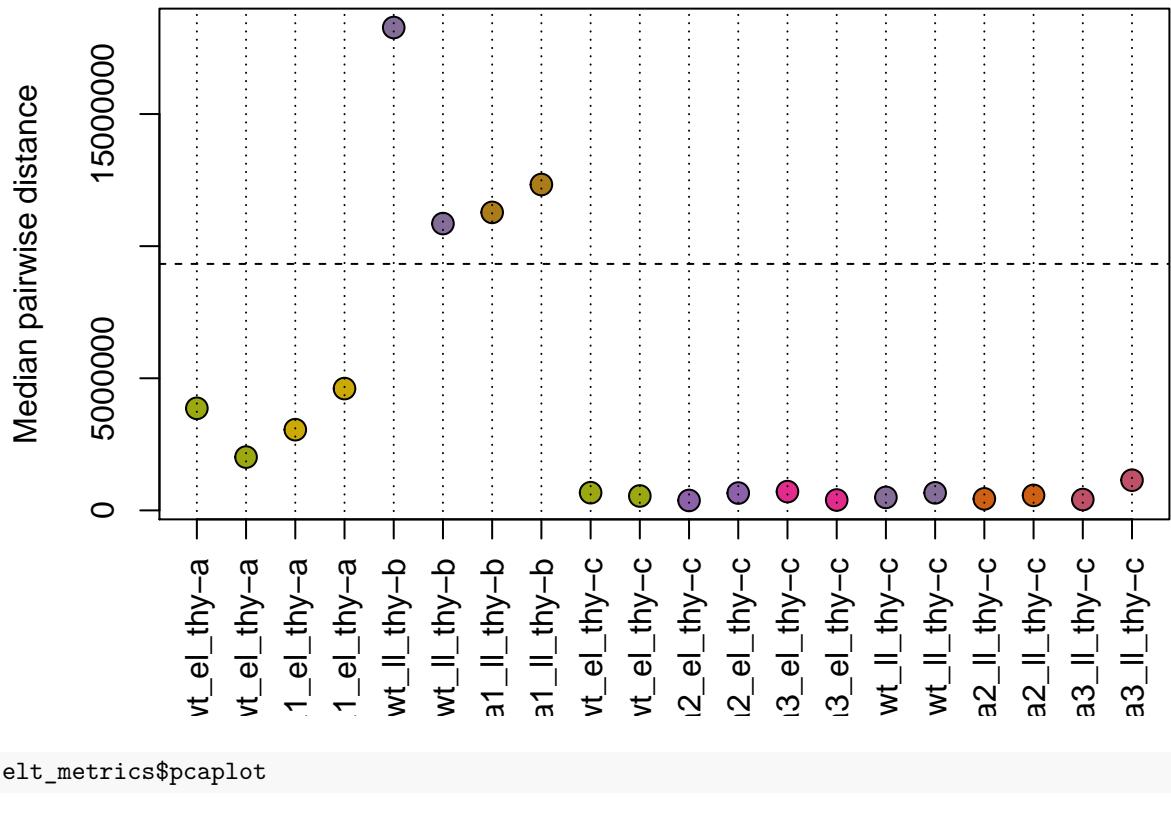
## This data will benefit from being displayed on the log scale.

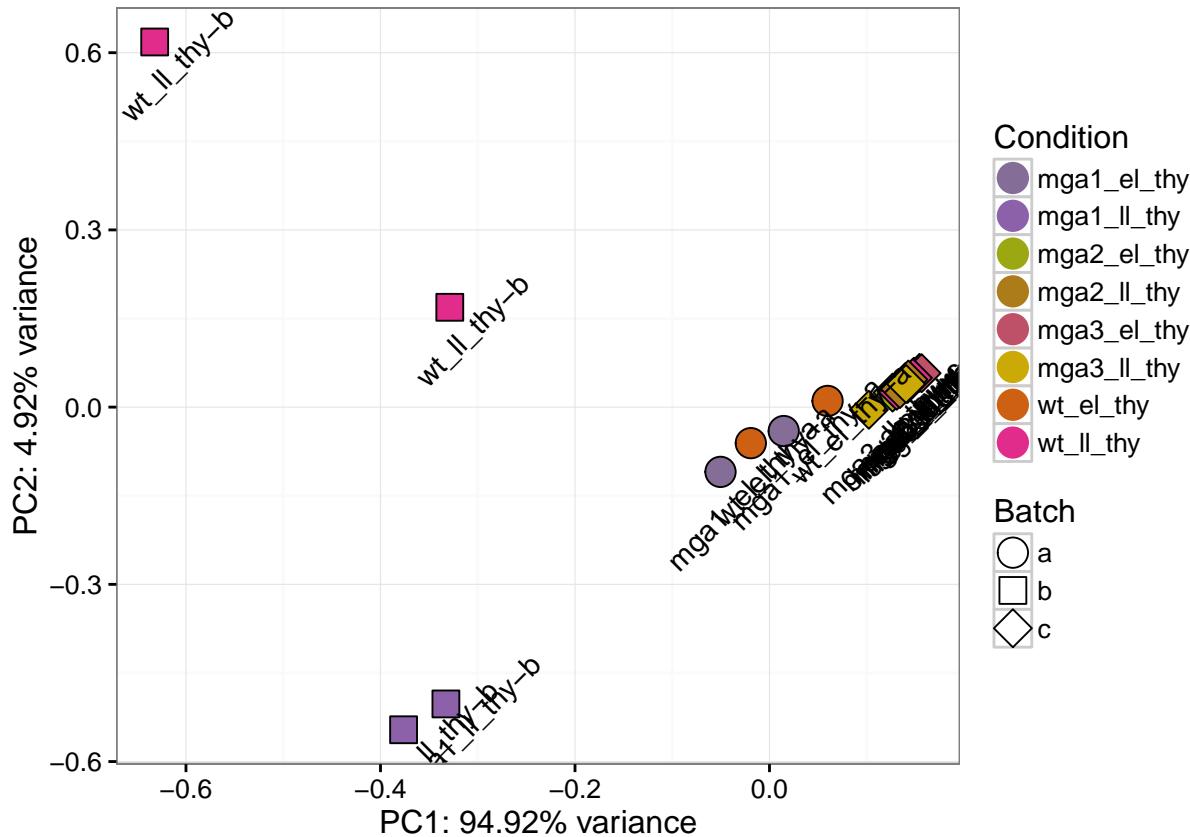
## If this is not desired, set scale='raw'

## No id variables; using all as measure variables

```

Standard Median Distance





```
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 particularly surprising 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 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 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).

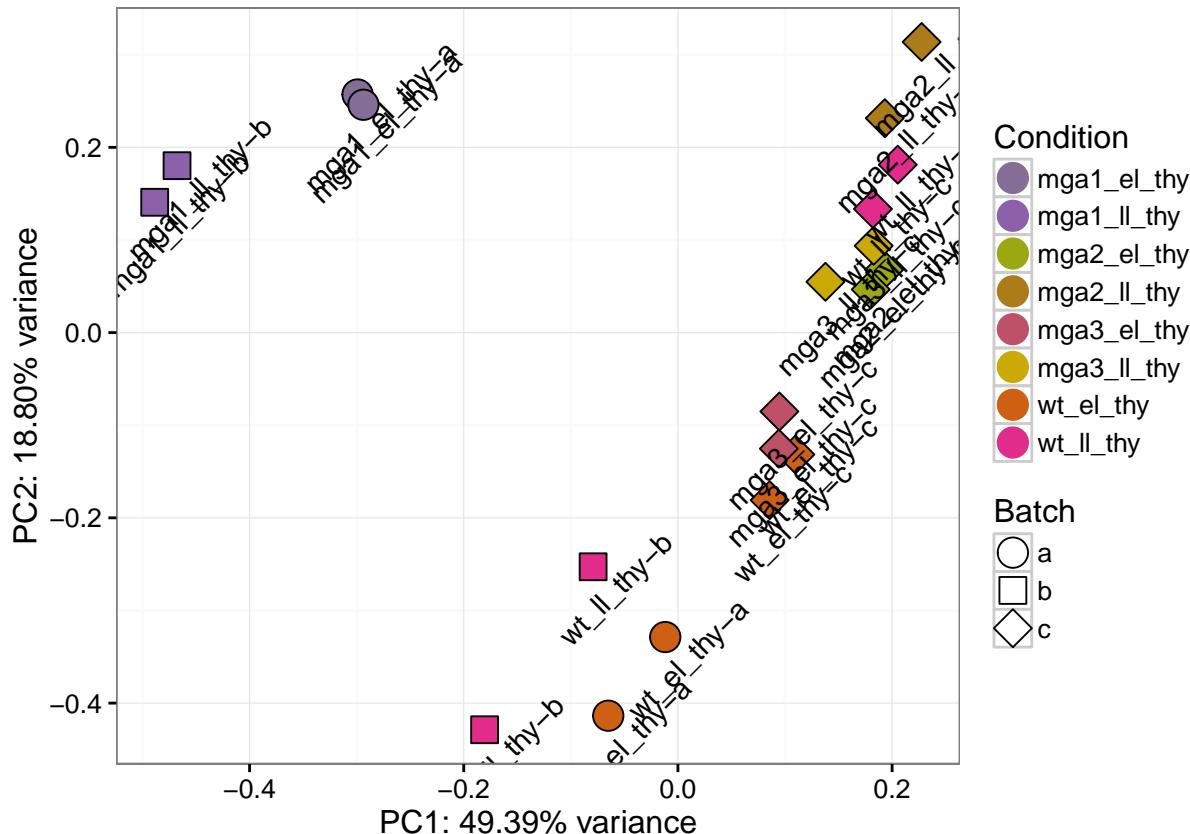
## Leaving the data unnormalized. This is necessary for DESeq, but
## EdgeR/limma might benefit from normalization. Good choices include quantile,
## size-factor, tmm, etc.

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

## transform_counts: Found 1519 values equal to 0, adding 0.5
## 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 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 ccbc, 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!)

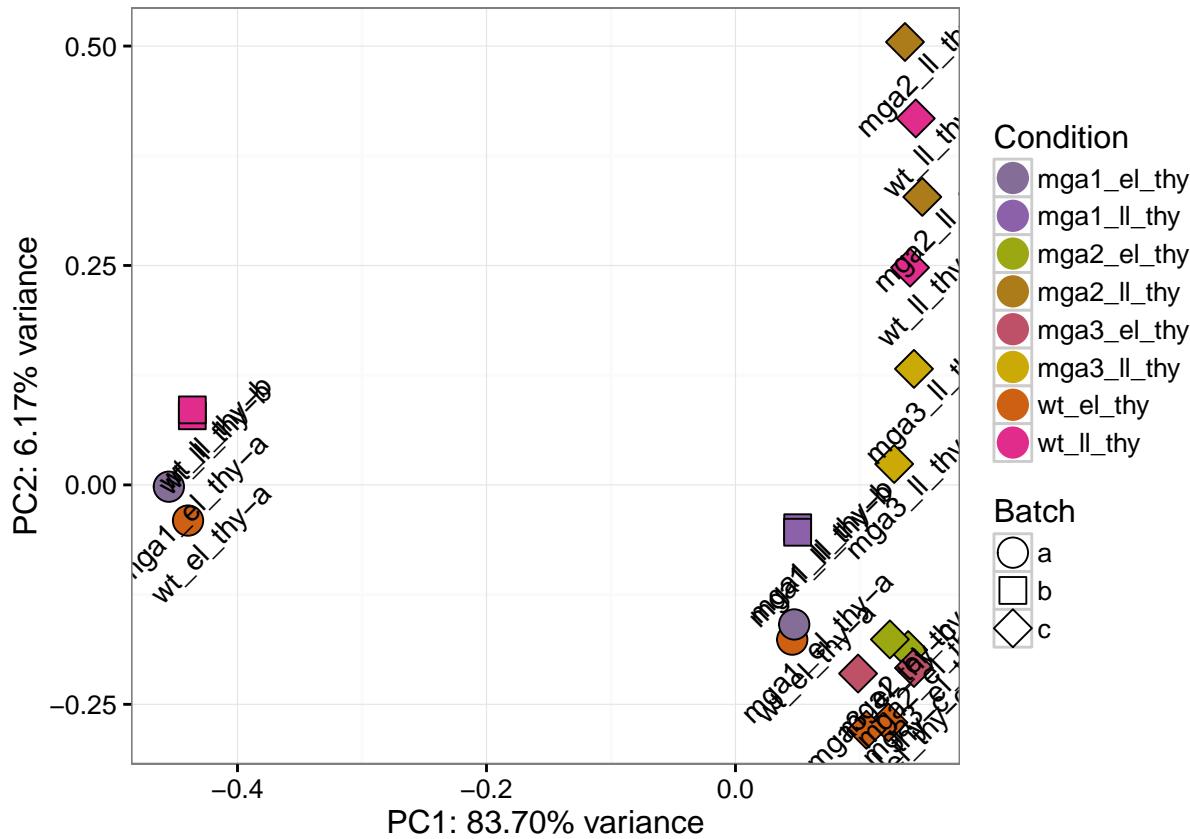
## Leaving the data in its current base format, keep in mind that
##   some metrics are easier to see when the data is log2 transformed, but
##   EdgeR/DESeq don't like transformed data.

## Leaving the data unconverted. It is often advisable to cpm/rpk
##   the data to normalize for sampling differences, keep in mind though that rpk
##   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 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 ccbc, 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!)

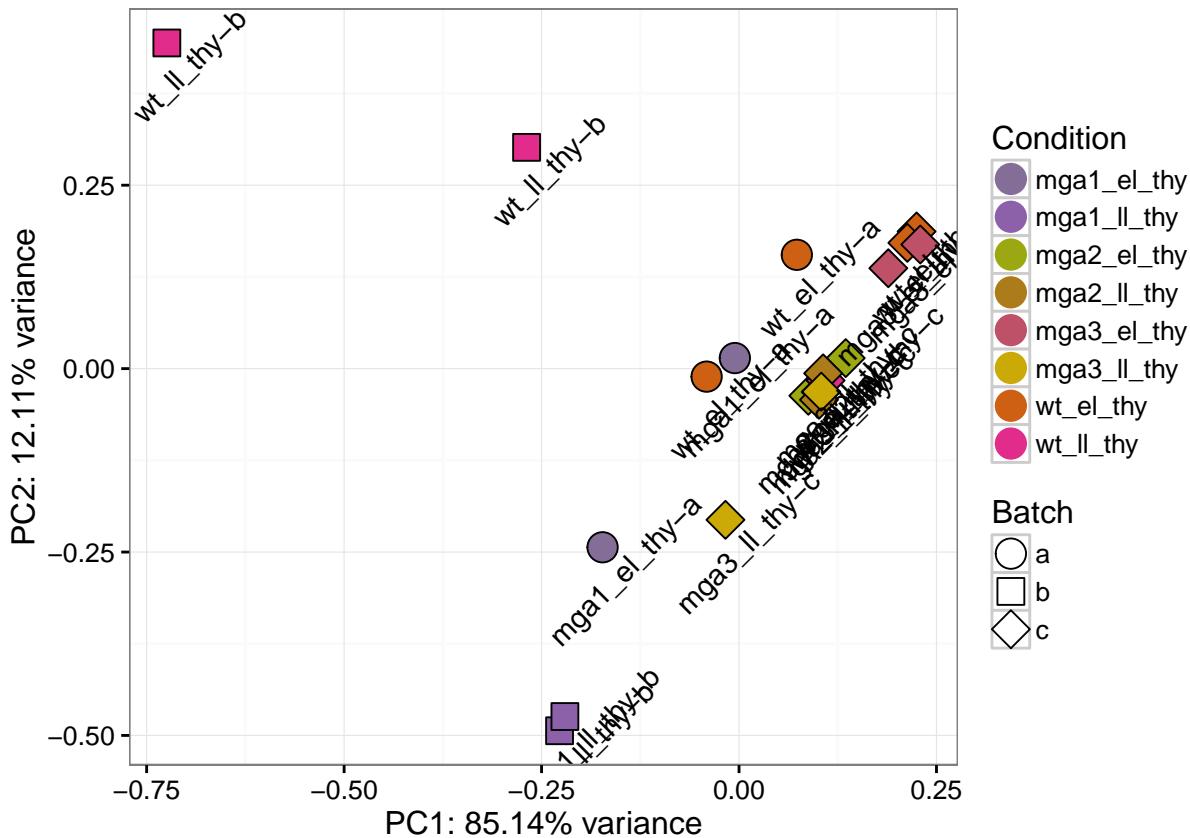
## Leaving the data in its current base format, keep in mind that
##   some metrics are easier to see when the data is log2 transformed, but
##   EdgeR/DESeq don't like transformed data.

## 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.
```

```
hpg1_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 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

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

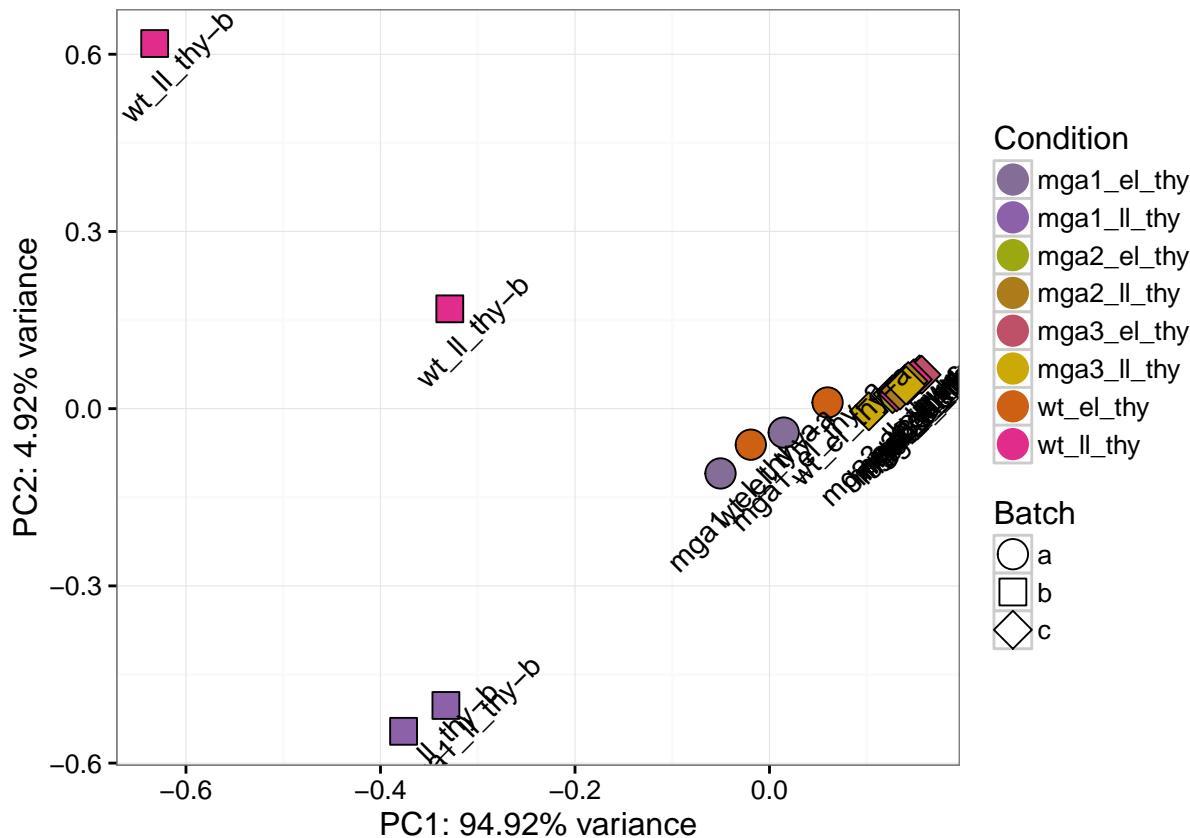
## Leaving the data unnormalized. This is necessary for DESeq, but
## EdgeR/limma might benefit from normalization. Good choices include quantile,
## size-factor, tmm, etc.

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

## 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 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

## Did not recognize the filtering argument, defaulting to ccbc's.
## Recognized filters are: 'cv', 'kofa', 'pofa', 'ccbc'

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

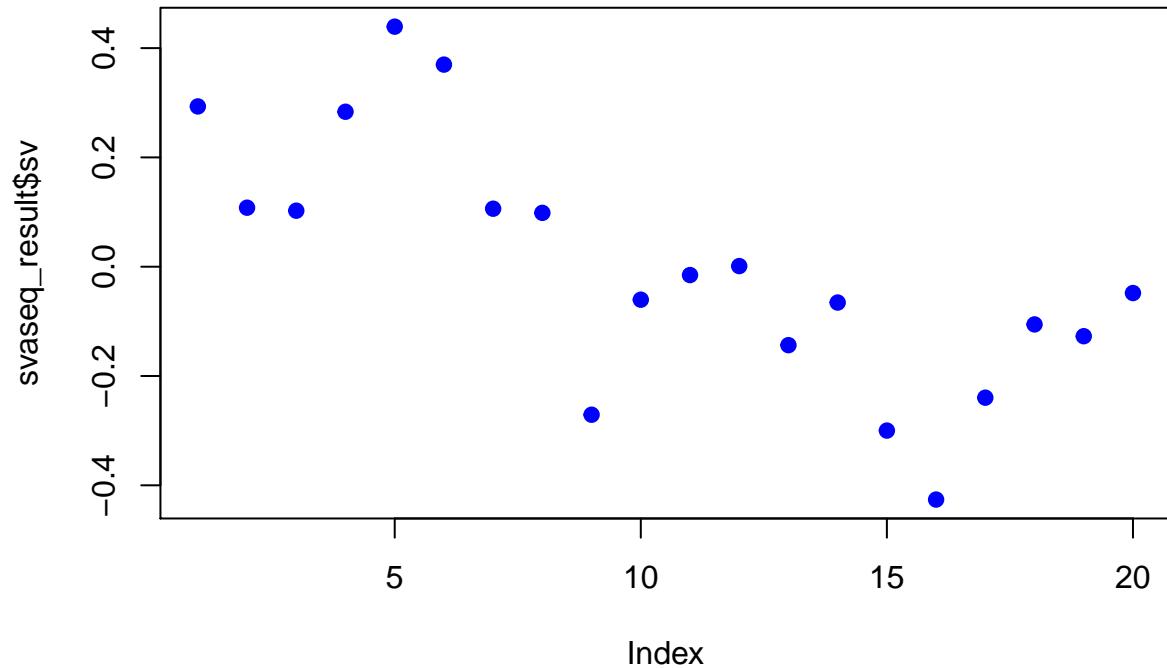
## batch_counts: Using svaseq for batch correction.

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

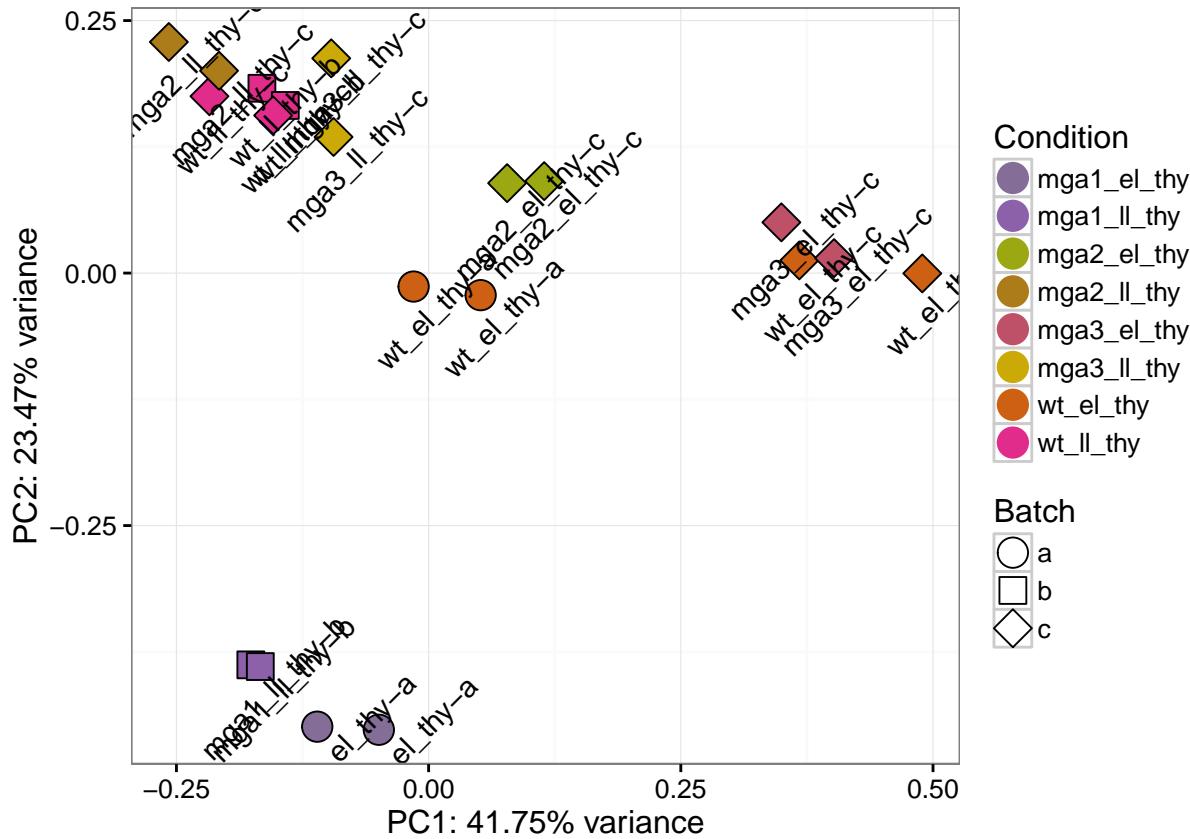
## The number of elements which are < 0 after batch correction is: 276

## transform_counts: Found 276 values equal to 0, adding 0.5
## to the matrix.

```



```
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", "limmarestid", "ruvg", "combat", "combatmod"

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

##   propVar cumPropVar cond.R2 batch.R2
## 1   41.75      41.75   81.37    15.94
## 2   23.47      65.22   99.48    42.04
## 3   11.58      76.80   61.57    67.21
## 4    5.99      82.79   88.18    12.49
## 5    4.59      87.38   77.78     5.84
## 6    3.04      90.42   87.63    16.47

## 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.
```

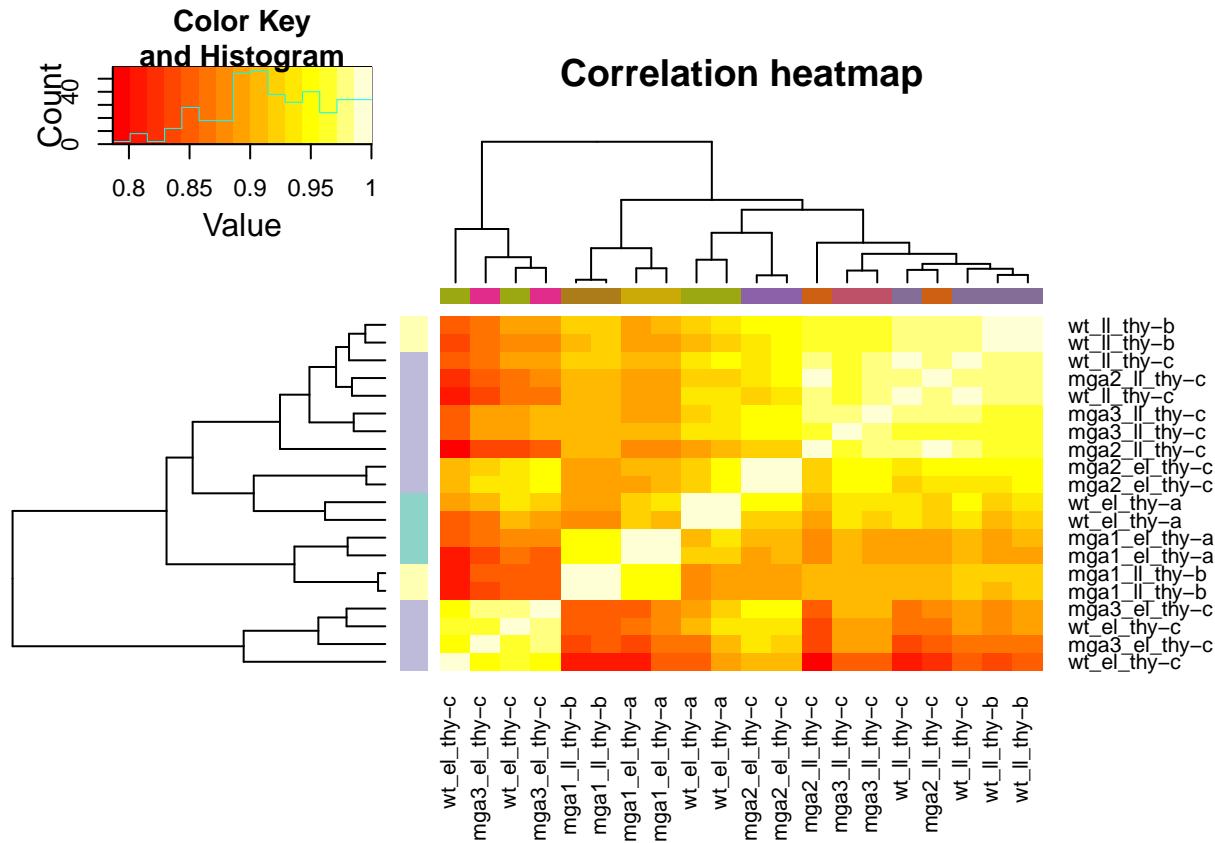
```

## Graphing a boxplot.

## Graphing a correlation heatmap.

## Graphing a standard median correlation.

```

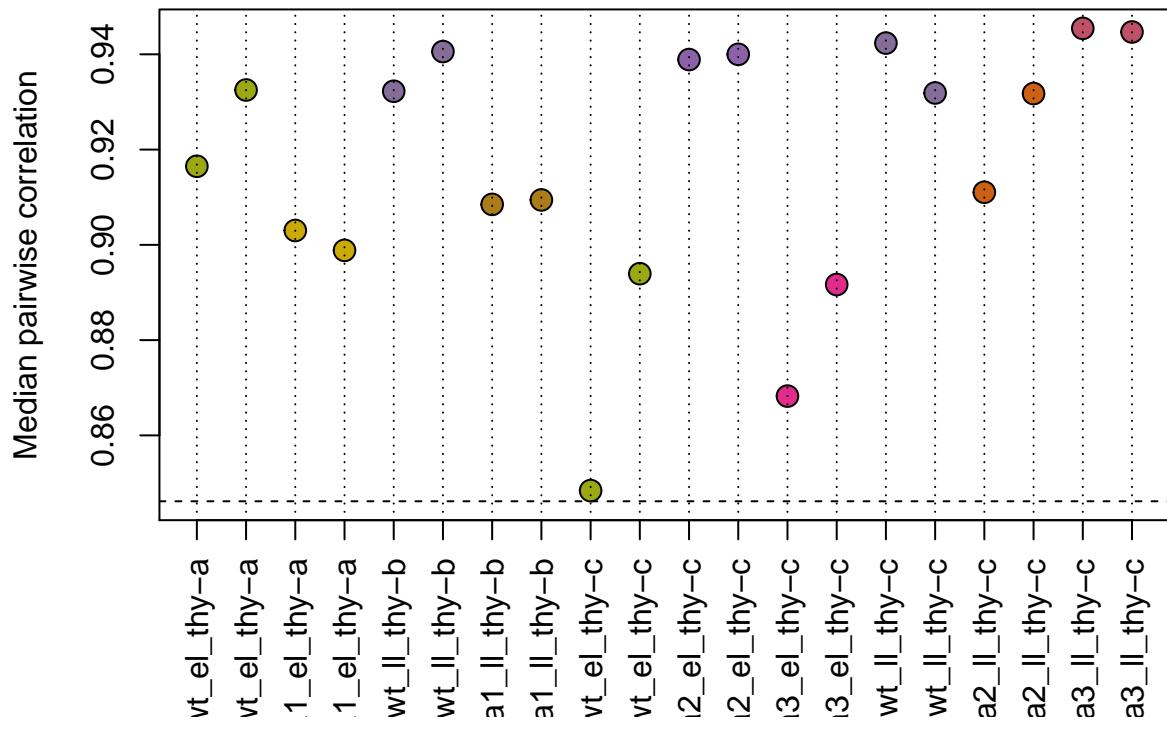


```

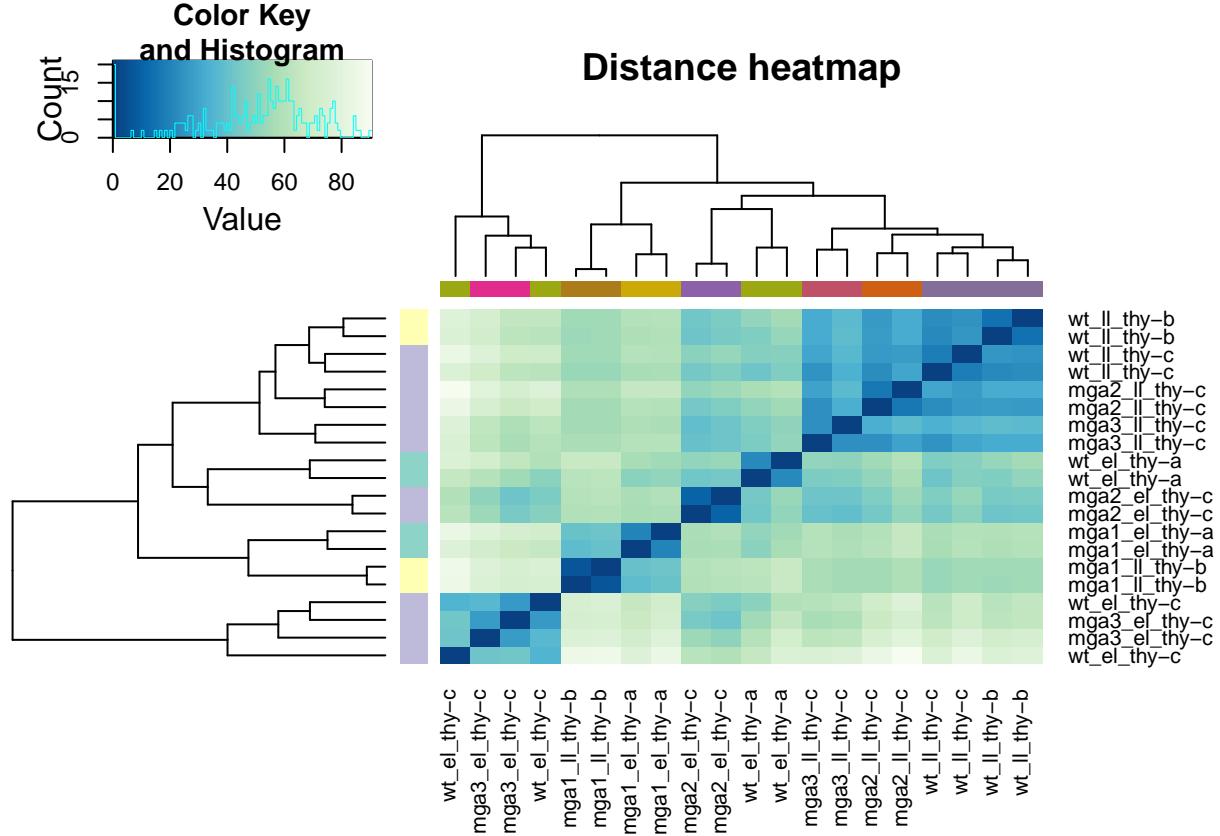
## Graphing a distance heatmap.

```

Standard Median Correlation

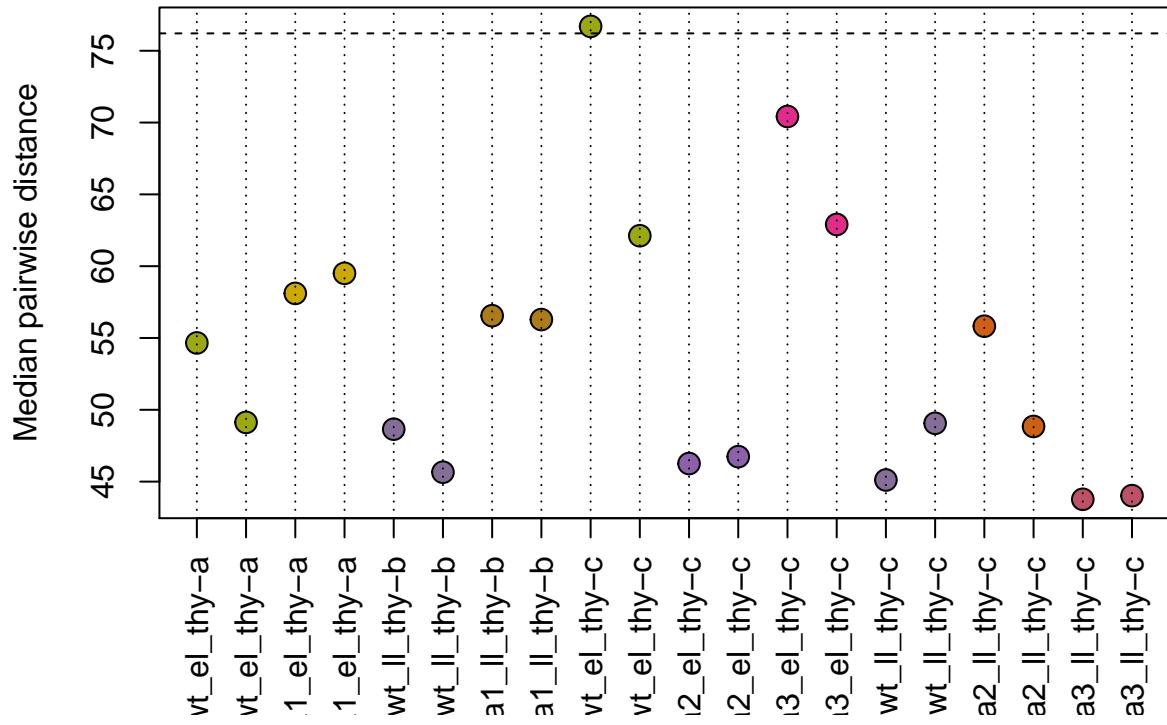


Graphing a standard median distance.



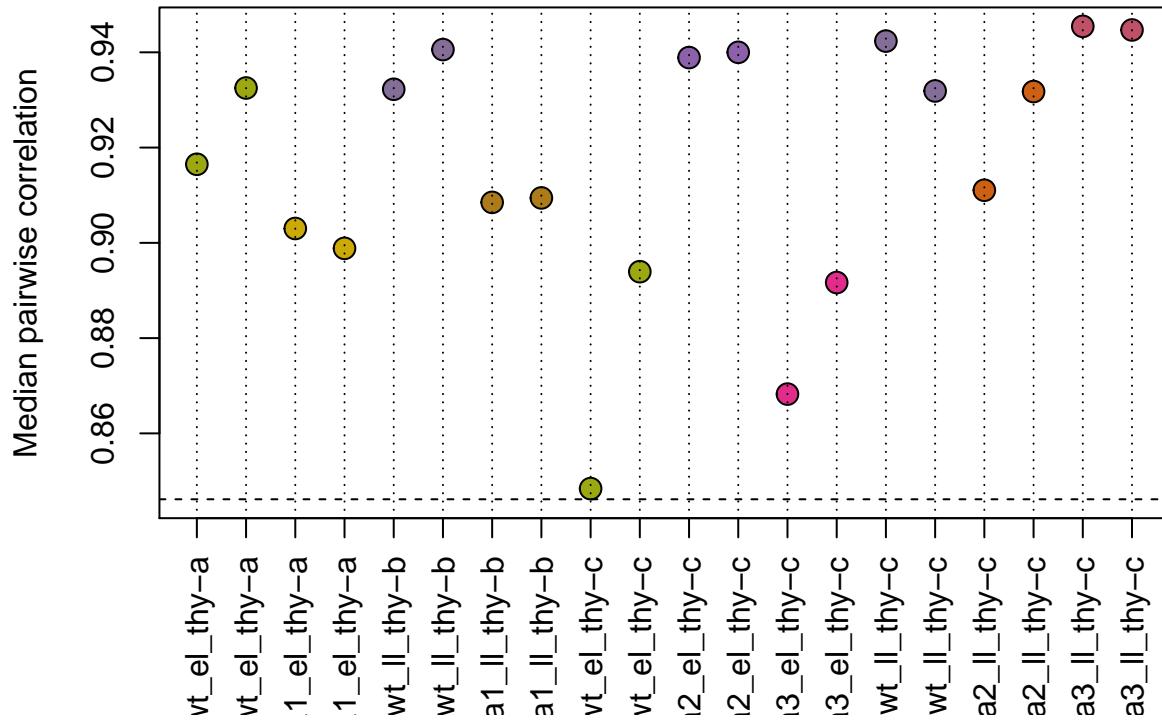
```
## Graphing a PCA plot.  
## Plotting a density plot.
```

Standard Median Distance

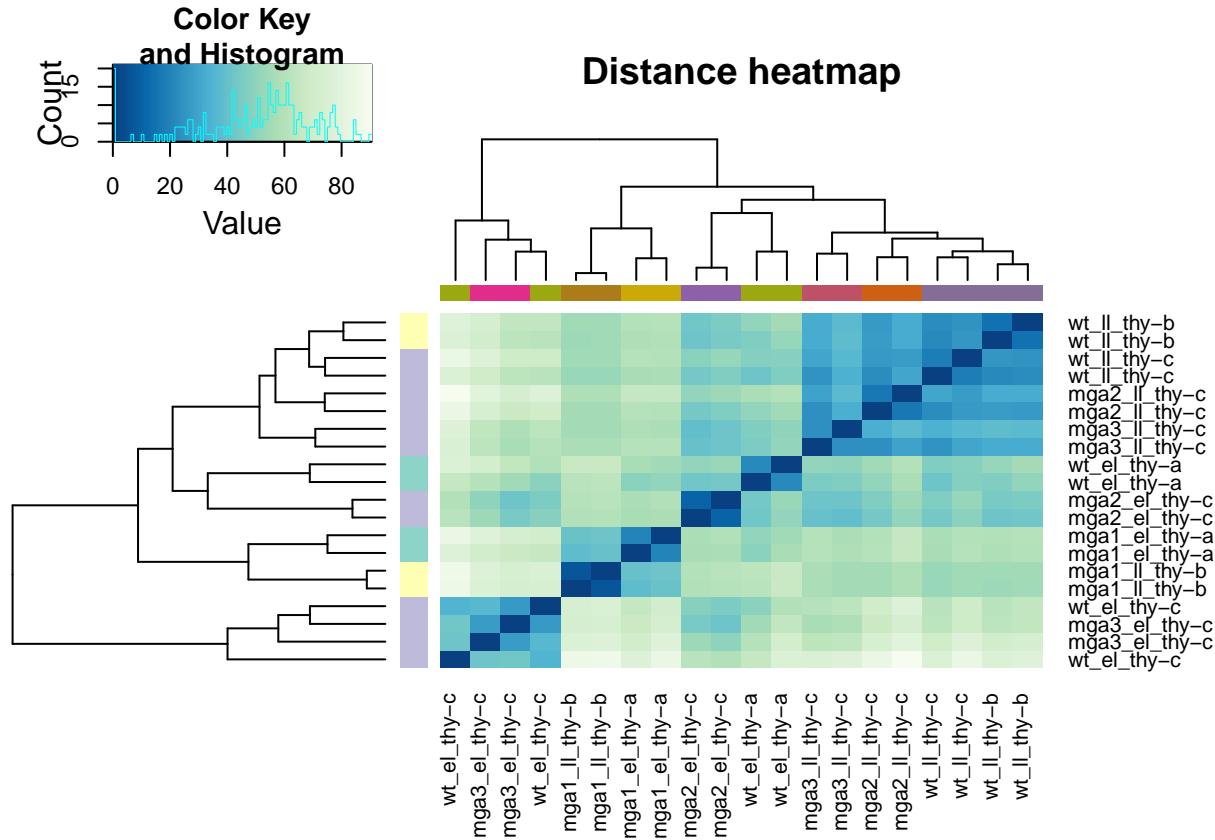


```
norm_graphs$smc
```

Standard Median Correlation



```
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.
```