

# hpgltools usage

*atb abelew@gmail.com*

*2016-03-13*

## 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"      "ensembl_pombe"      "expt"
## [25] "fis_batchnormpca"   "fis_info"           "fis_libsize"
```

```

## [28] "fis_nonzero"          "fis_normbatchpca"    "fis_normpca"
## [31] "fis_rawpca"           "fission"            "fission_data"
## [34] "fission_expt"         "fun_data"           "fun_norm"
## [37] "gene_names"            "gff_from_txdb"     "goseq_search"
## [40] "gostats_search"        "lengths"            "limma_comparison"
## [43] "limma_results"         "meta"               "meta_dataframe"
## [46] "most_vOM1"             "mut_120"            "normbatch_expt"
## [49] "norm_expt"              "norm_graphs"       "norm_test"
## [52] "num_colors"             "pca_test"           "pombe"
## [55] "pombe_filters"          "pombe_goids"        "pombe_transcripts"
## [58] "raw_metrics"            "rle_expt"           "sample_definitions"
## [61] "scatter_wt_mut"         "sf_expt"             "sig_genes"
## [64] "table"                  "test_pca"            "tm_expt"
## [67] "tmp_definitions"        "topgo_search"       "tt"
## [70] "updown_genes"           "up_expt"             "written_gff"
## [73] "wt_120"

## The gff information is in 'annotations'
## The experiment is in most_vOM1
## Here is the meta-data! (well, the first 6 lines anyway).
knitr::kable(head(most_vOM1$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_vOM1$definitions
counts <- Biobase::exprs(most_vOM1$expressionset)
## Originally the meta-data was kept in a file: all_samples.csv
expt <- create_expt(meta_dataframe=meta, count_dataframe=counts)

```

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

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

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

```
summary(expt)
```

```

##                                     Length Class      Mode
## initial_metadata            8   data.frame   list
## original_expressionset      1 ExpressionSet S4
## expressionset                1 ExpressionSet S4
## samples                      8   data.frame   list
## design                       8   data.frame   list

```

```

## definitions           23   data.frame    list
## stages               48   factor        numeric
## types                48   factor        numeric
## conditions           48   factor        numeric
## batches              48   factor        numeric
## 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/rpkpm/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.

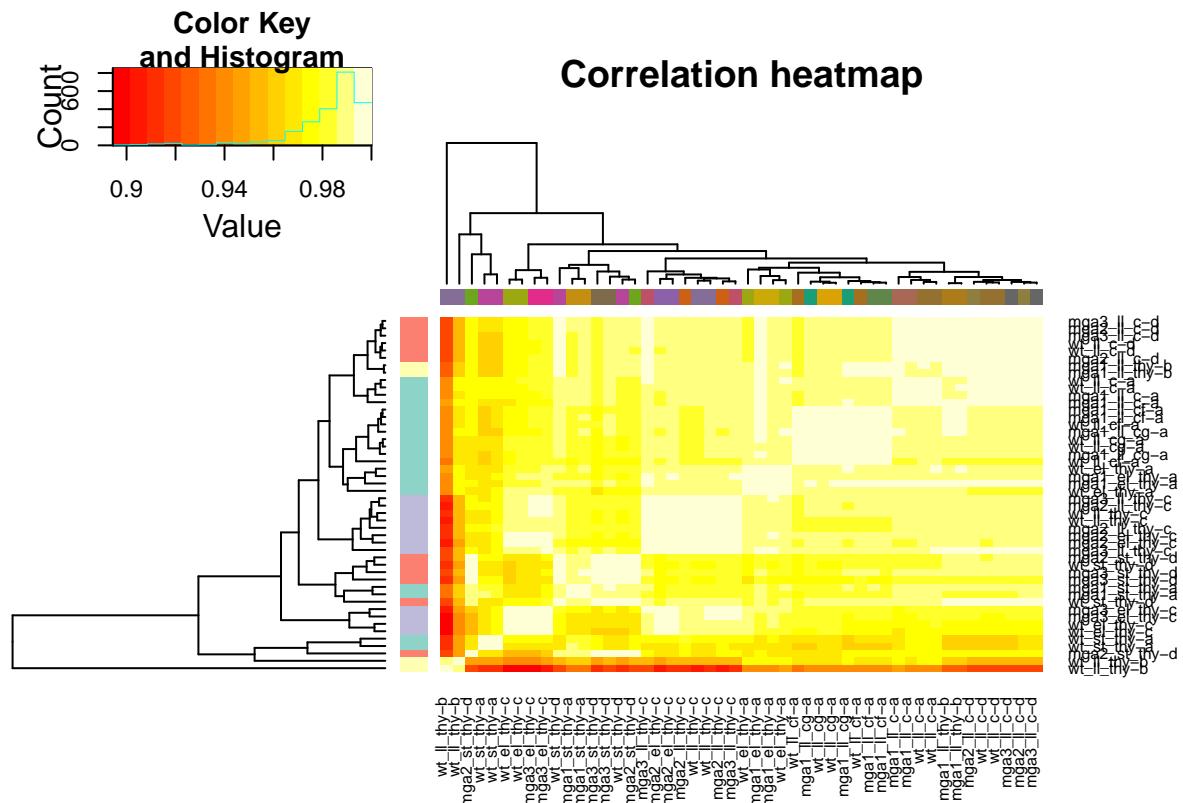
## Graphing a boxplot.

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

## Graphing a correlation heatmap.

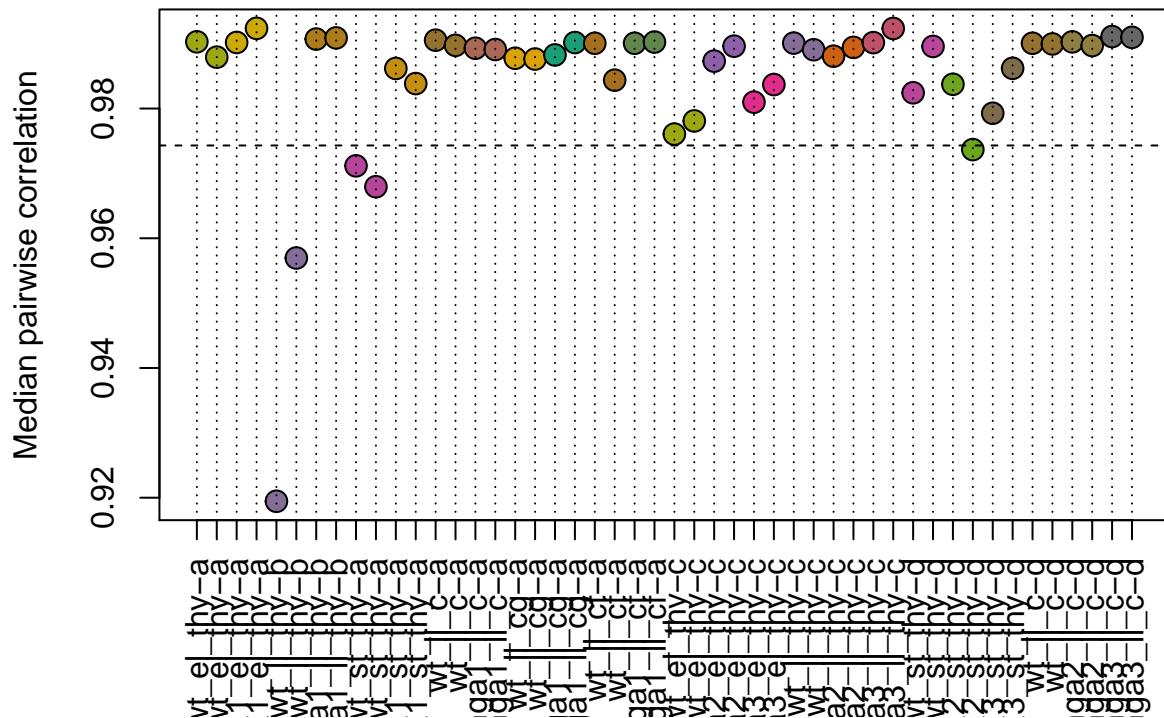
```

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

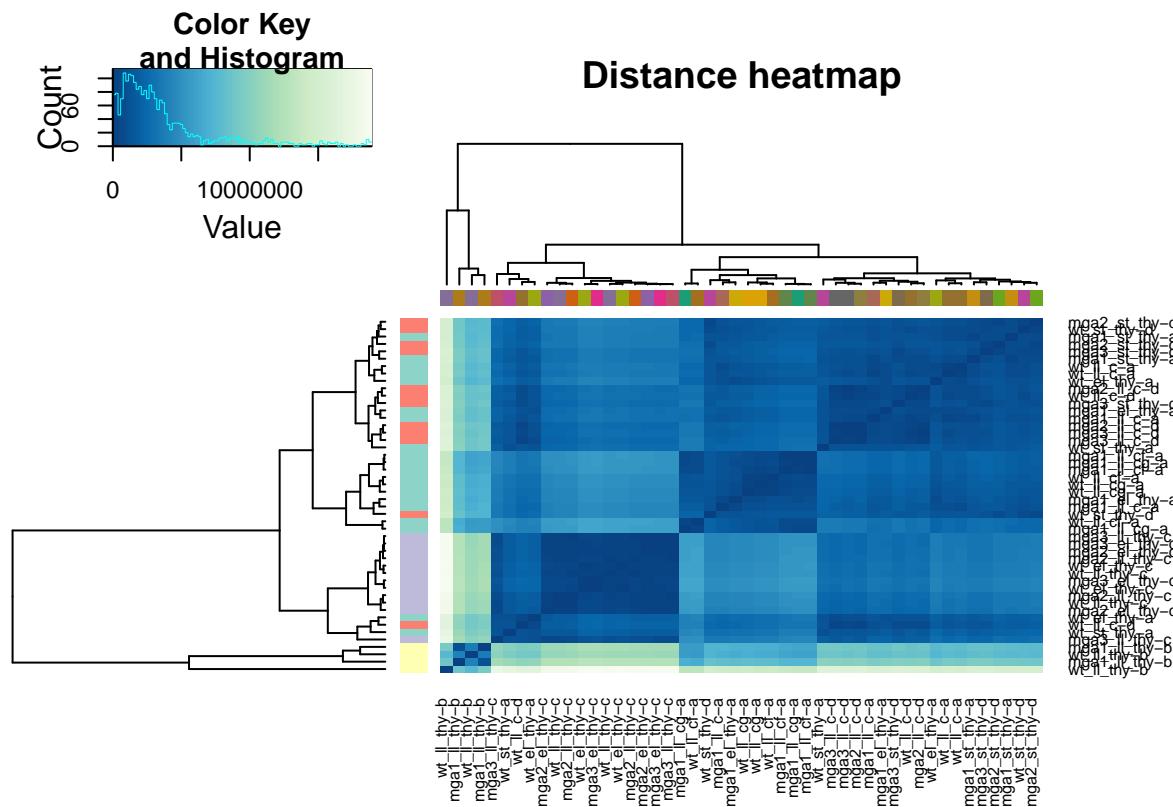


```
## Graphing a distance heatmap.
```

## 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.

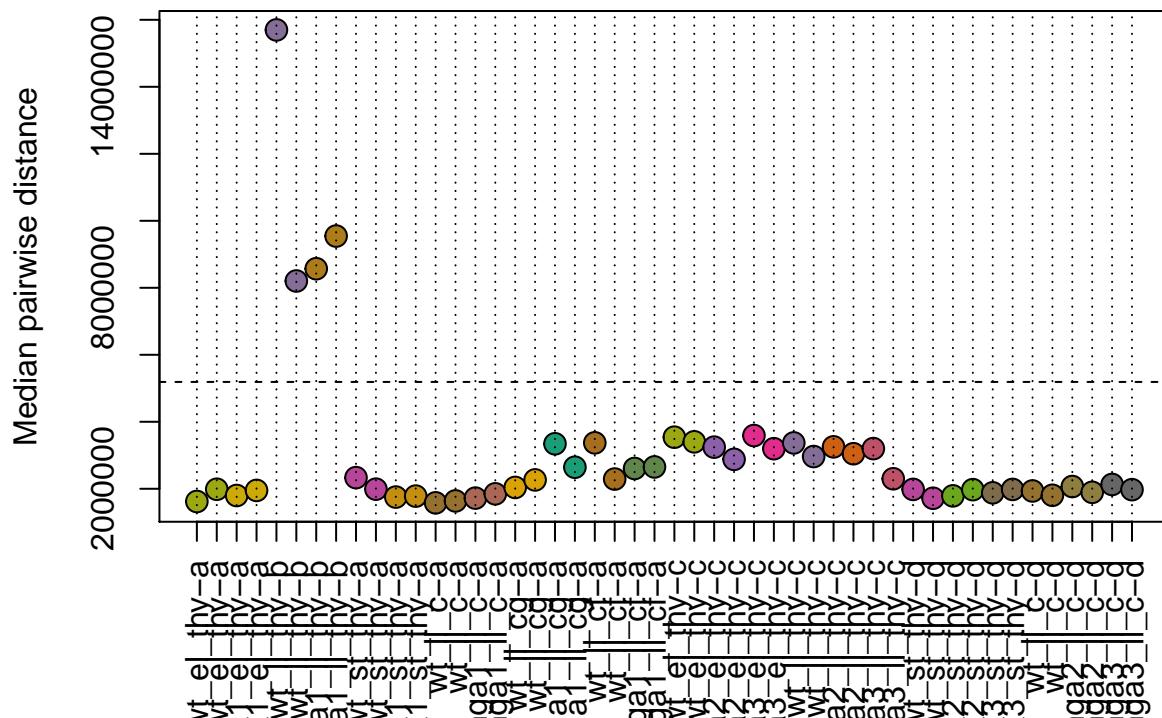
## If this is not desired, set scale='raw'

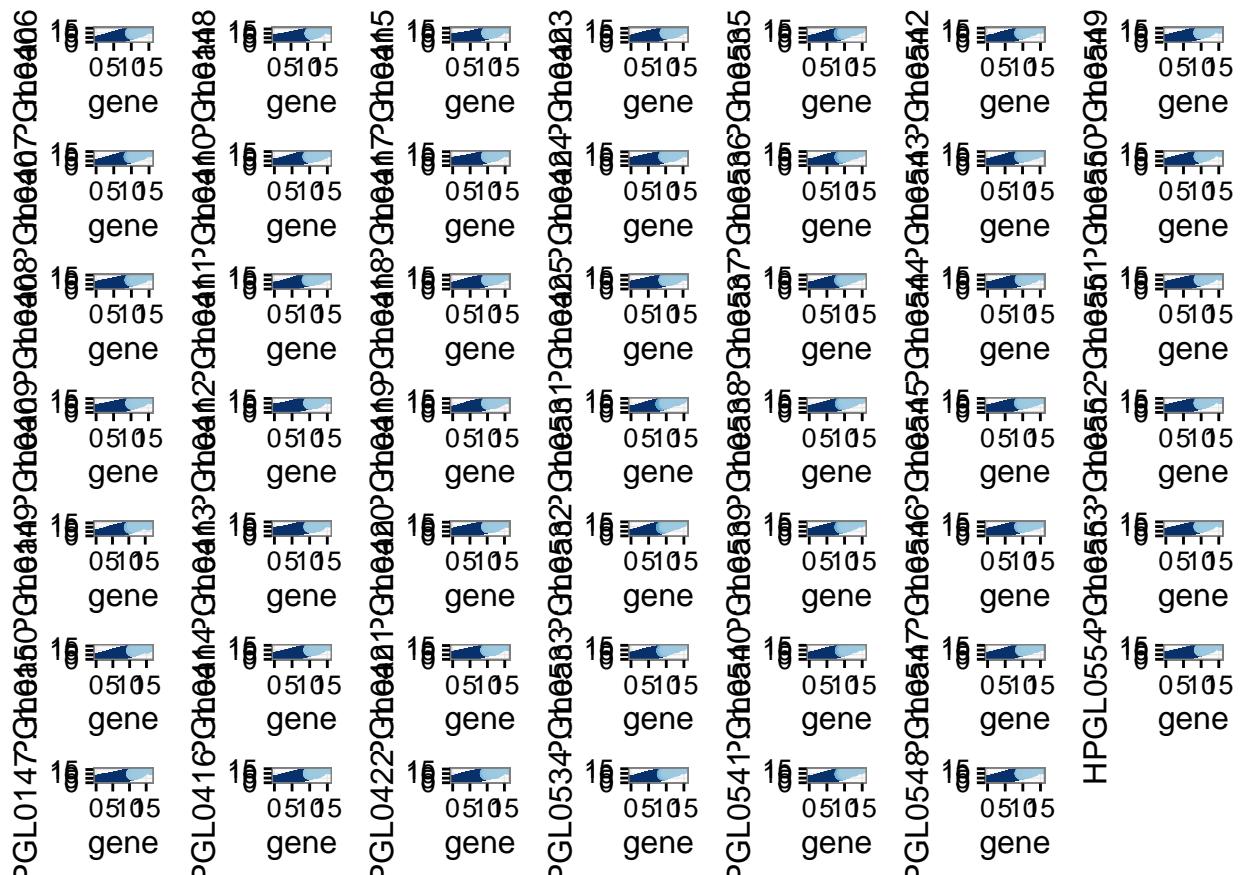
## No id variables; using all as measure variables

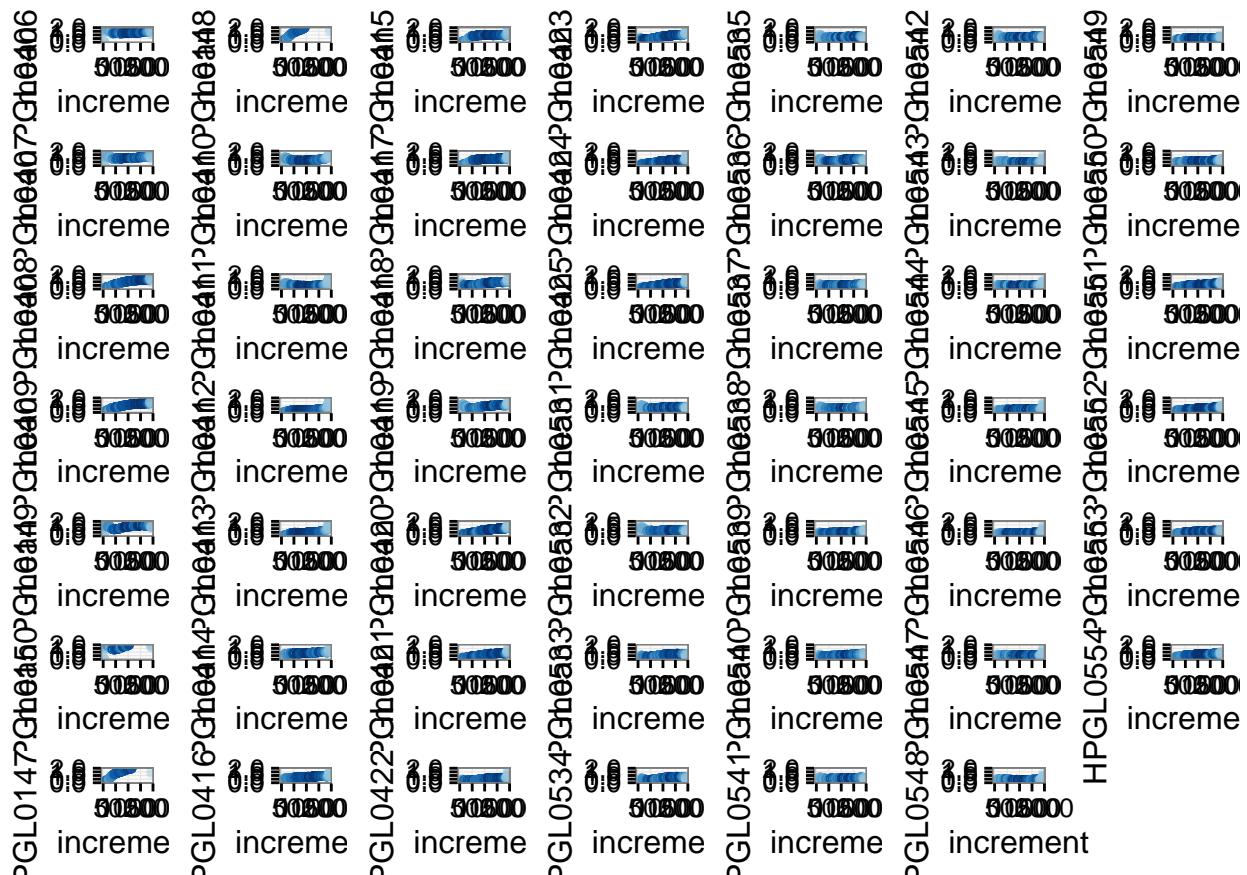
## QQ plotting!.

```

## Standard Median Distance

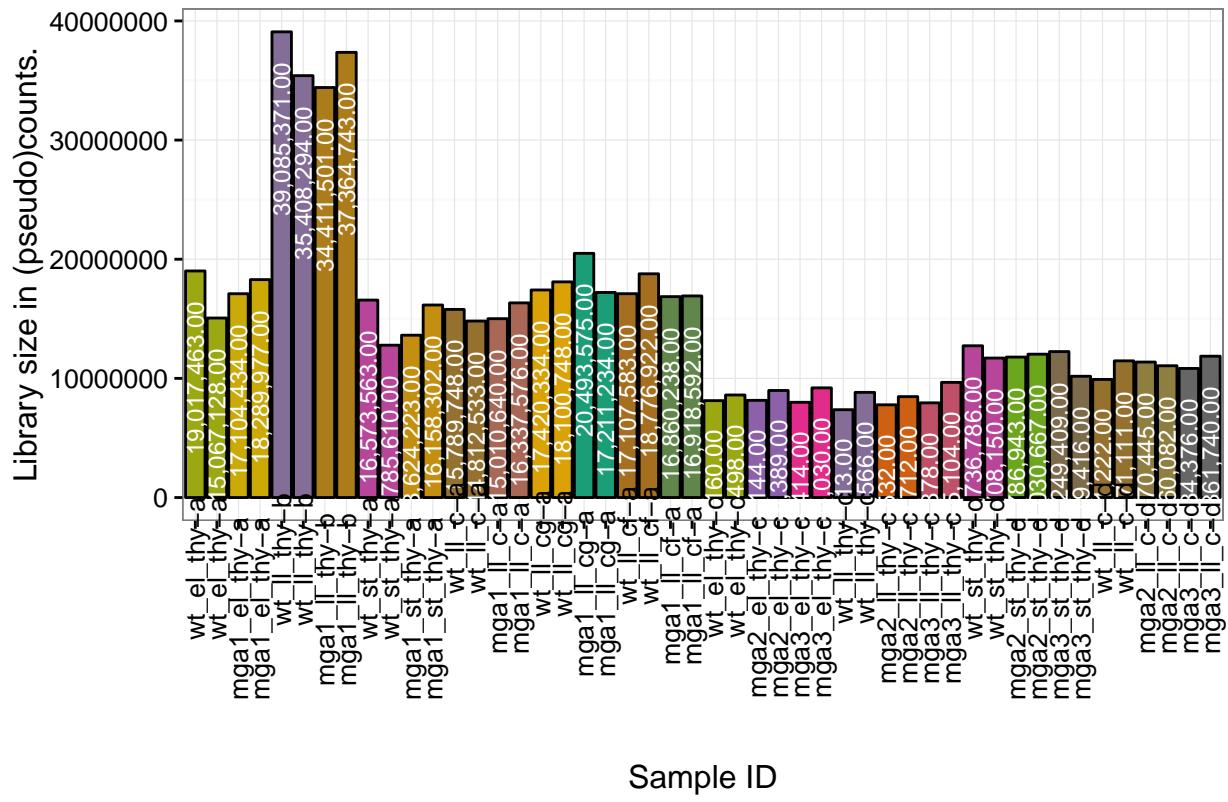






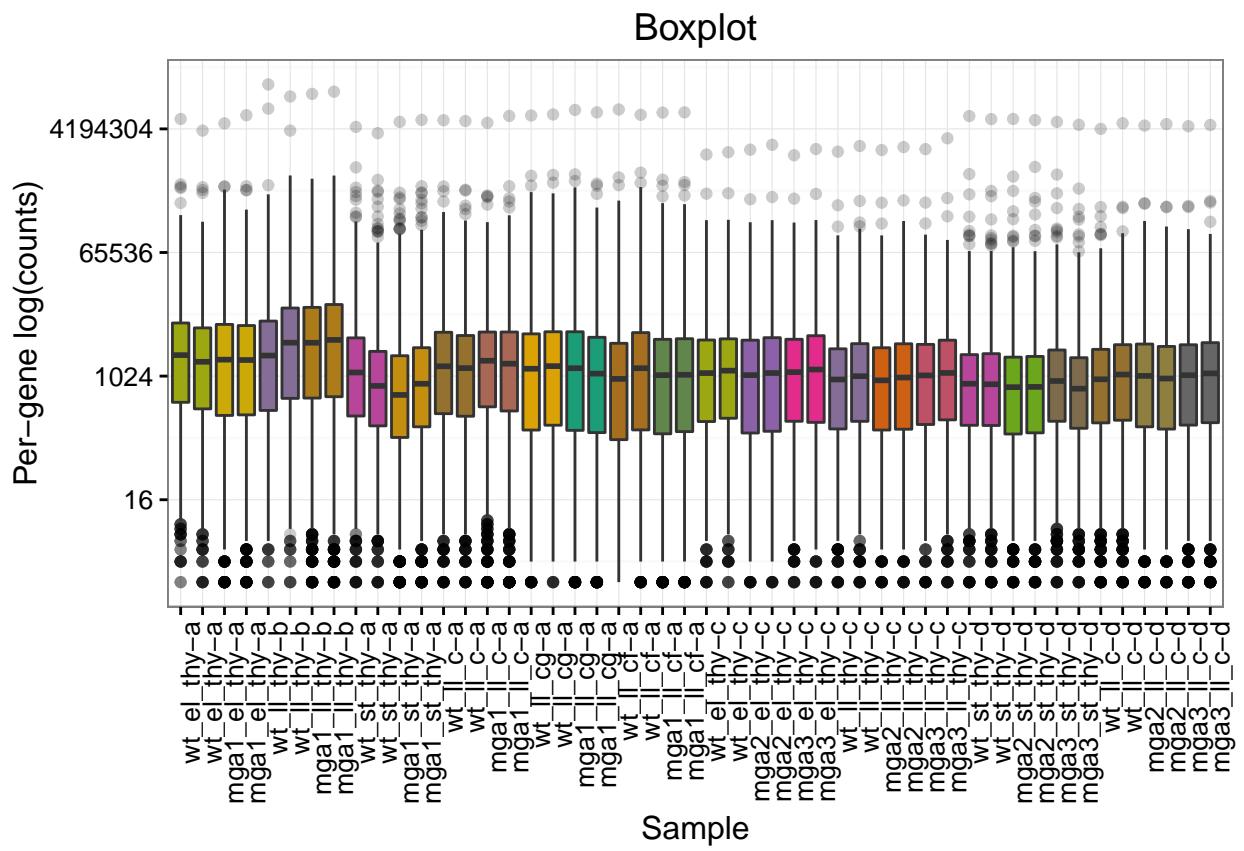
```
## View a raw library size plot
raw_metrics$libsize
```

## Library sizes



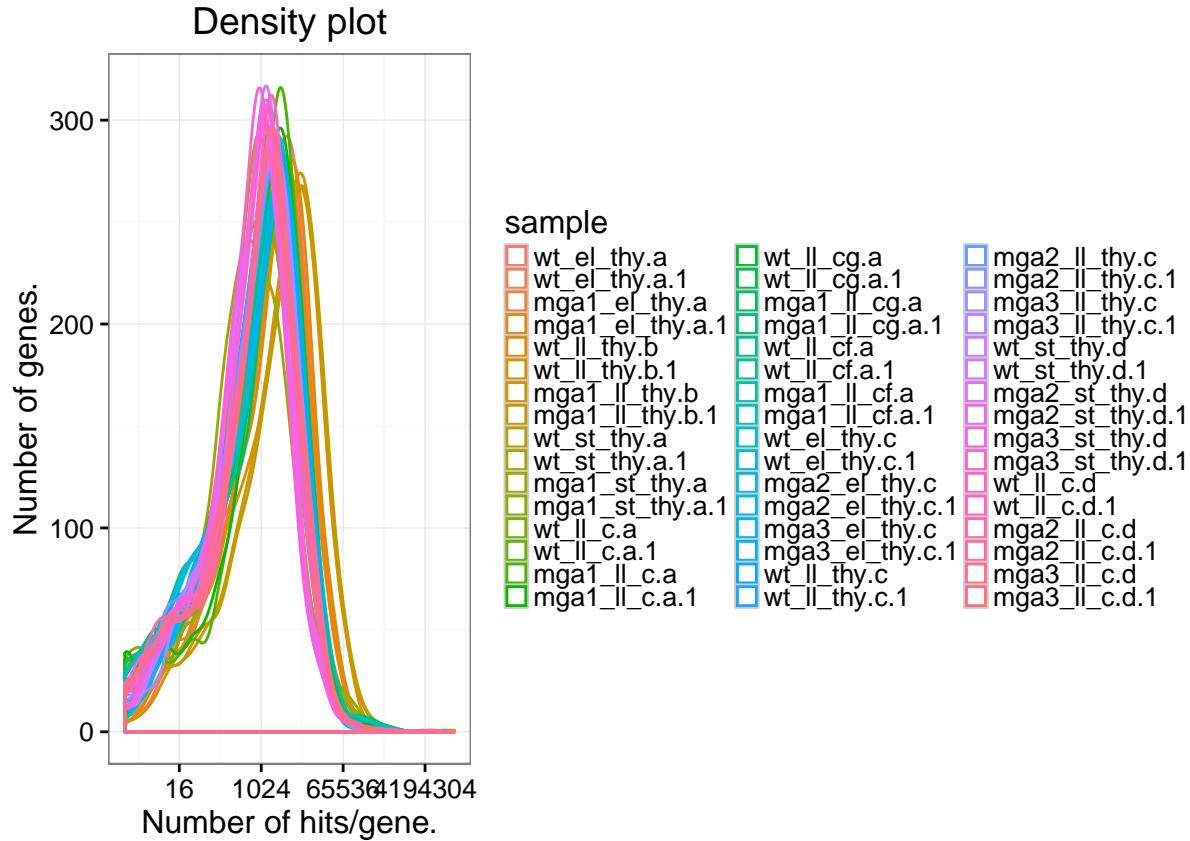
```
## Or boxplot to see the data distribution
raw_metrics$boxplot
```

```
## 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
```

```
## 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
```

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)

##           sample.id type stage replicate mutantname media exptdate libdate
## HPGL0406  HPGL0406  WT   EL        1 5448/01/01    THY 20130430 20140402
## HPGL0407  HPGL0407  WT   EL        2 5448/02/01    THY 20130430 20140402
## HPGL0408  HPGL0408  mga  EL        1      mga-1    THY 20130430 20140402
## HPGL0409  HPGL0409  mga  EL        2      mga-2    THY 20130430 20140402
## HPGL0149  HPGL0149  WT   LL        1     WT1-1    THY 20120924 20120926
## HPGL0150  HPGL0150  WT   LL        2     WT1-2    THY 20120924 20120926
##           batch condition sra reads.passed ncrna x..ncrna remaining
## HPGL0406    a  wt_el_thy  NA  19026277 353992  0.0186 18672285
## HPGL0407    a  wt_el_thy  NA  15074073 259613  0.0172 14814460
## HPGL0408    a  mga1_el_thy  NA 17112233 293752  0.0172 16818481
## HPGL0409    a  mga1_el_thy  NA 18298278 339862  0.0186 17958416
## HPGL0149    b  wt_ll_thy  NA 39107368 8055417  0.2060 31051951
## HPGL0150    b  wt_ll_thy  NA 35429033 3705275  0.1046 31723758
##           genome x..genome other.bacterial x..other
## HPGL0406 17810587  0.9539      366984  0.0197
## HPGL0407 14334043  0.9676      269649  0.0182
## HPGL0408 15769581  0.9376      318127  0.0189
## HPGL0409 16553148  0.9217      349869  0.0195
## HPGL0149 26285560  0.8465      324903  0.0105
## HPGL0150 30012962  0.9461      310449  0.0098
##           file
## HPGL0406 preprocessing/HPGL0406/bowtie_out/hpgl0406_forward-v0M1.count.xz
## HPGL0407 preprocessing/HPGL0407/bowtie_out/hpgl0407_forward-v0M1.count.xz
## HPGL0408 preprocessing/HPGL0408/bowtie_out/hpgl0408_forward-v0M1.count.xz
## HPGL0409 preprocessing/HPGL0409/bowtie_out/hpgl0409_forward-v0M1.count.xz
## HPGL0149 preprocessing/HPGL0149/bowtie_out/hpgl0149_forward-v0M1.count.xz
## HPGL0150 preprocessing/HPGL0150/bowtie_out/hpgl0150_forward-v0M1.count.xz
##           colors counts intercounts
## HPGL0406 #9BA812 unknown      unknown
## HPGL0407 #9BA812 unknown      unknown
## HPGL0408 #CBA907 unknown      unknown
## HPGL0409 #CBA907 unknown      unknown
## HPGL0149 #846D97 unknown      unknown
## HPGL0150 #846D97 unknown      unknown

## 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.

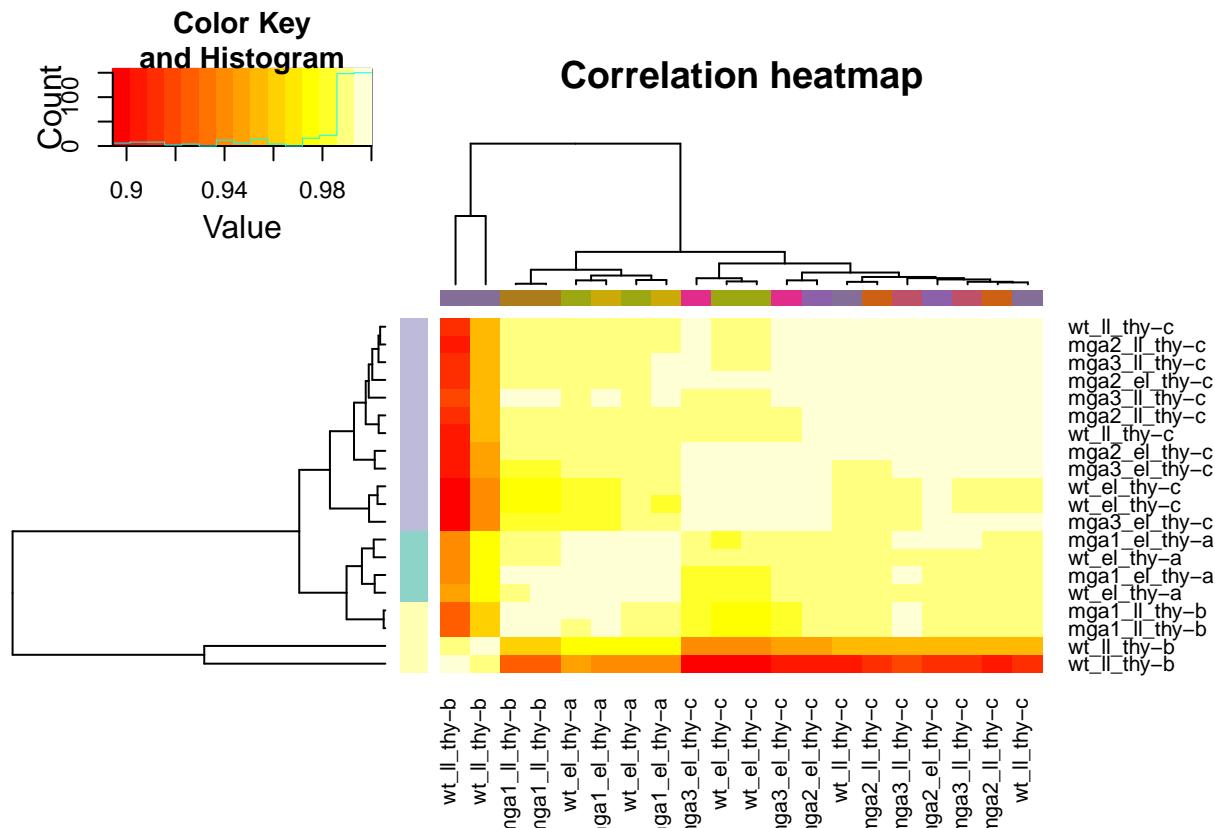
## Graphing library sizes.

## Graphing a boxplot.
```

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

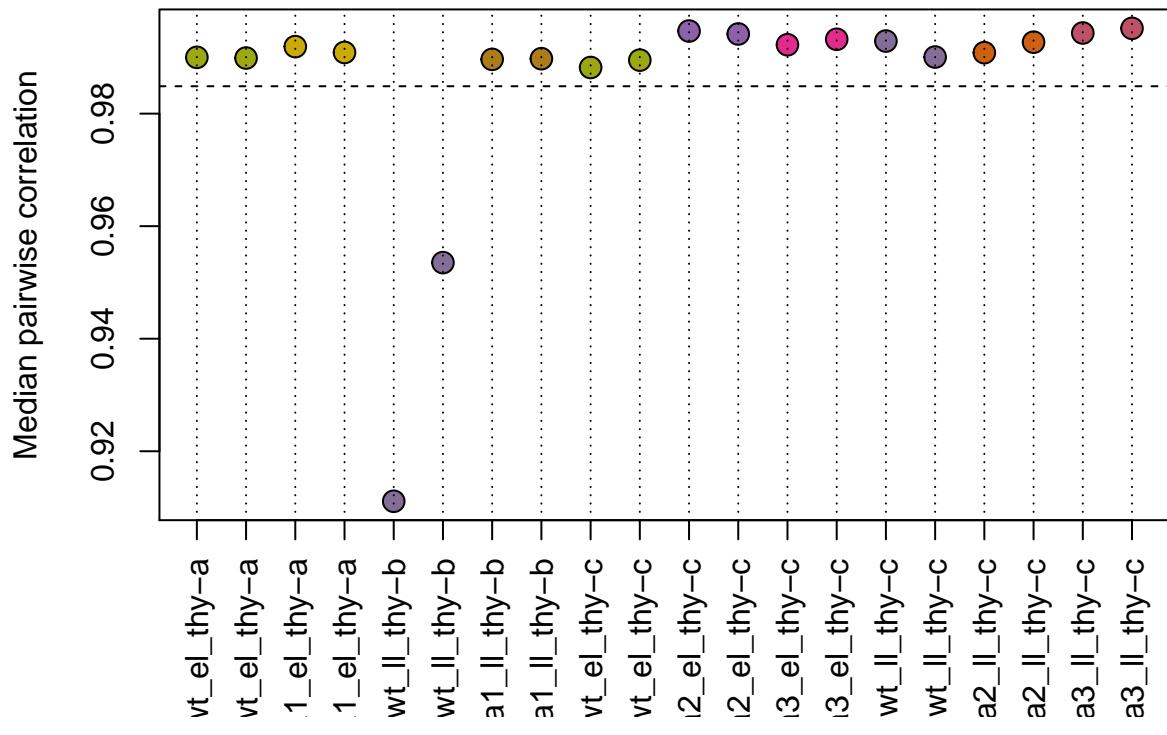
```
## Graphing a correlation heatmap.
```

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

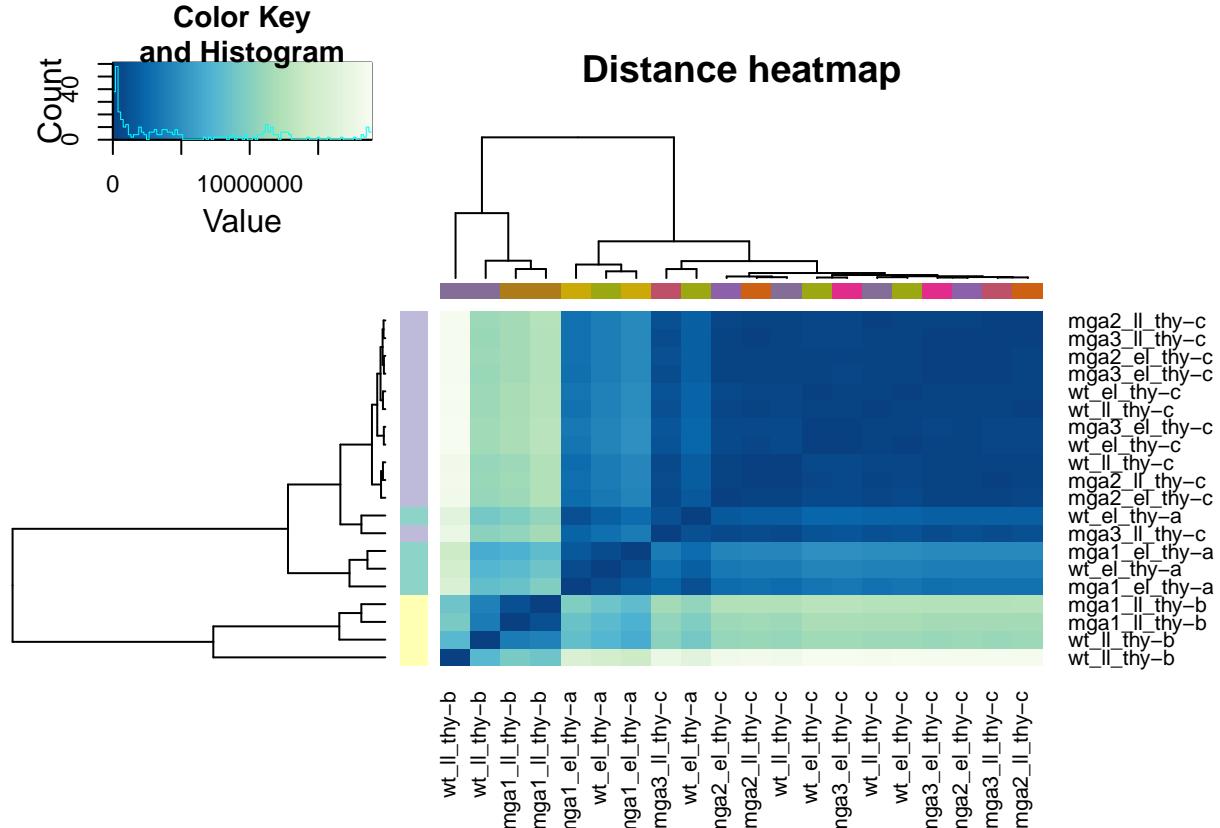


```
## Graphing a distance heatmap.
```

## Standard Median Correlation



## Graphing a standard median distance.



```

## Graphing a PCA plot.

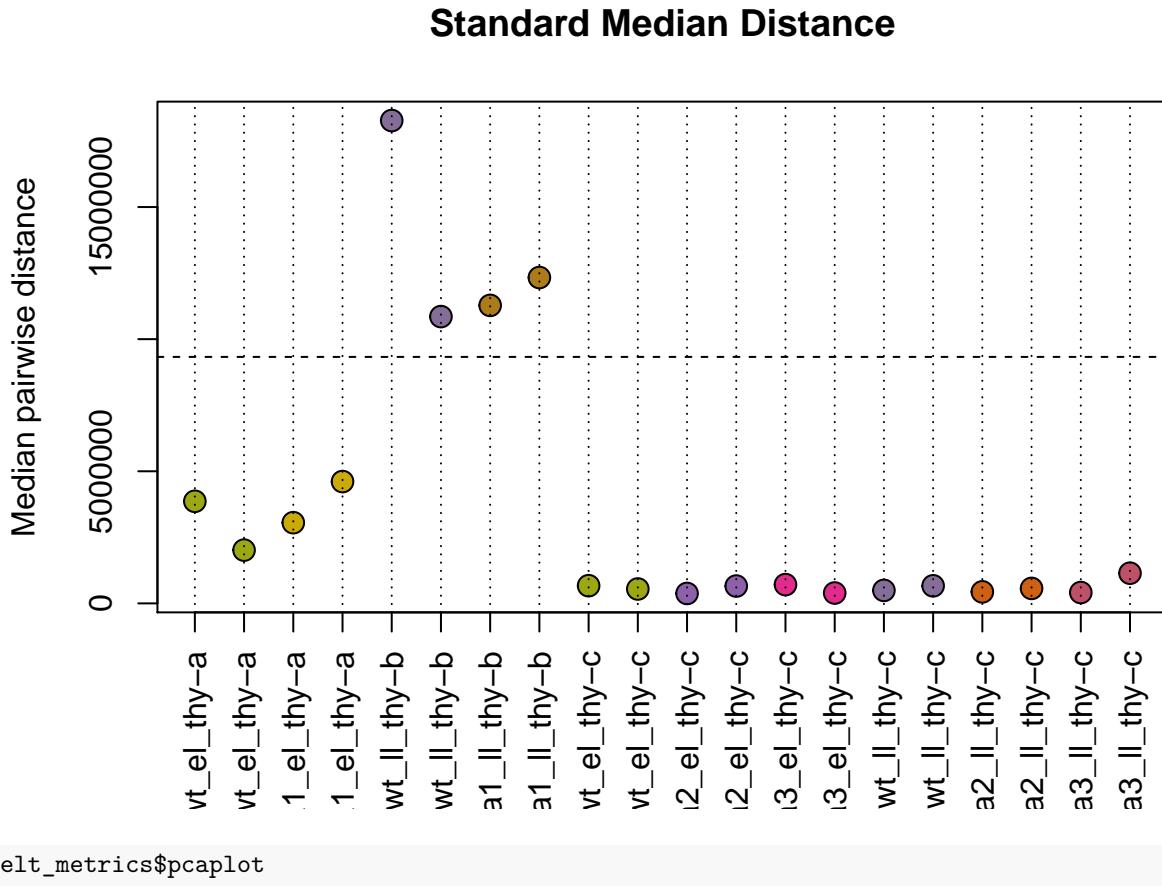
## Plotting a density plot.

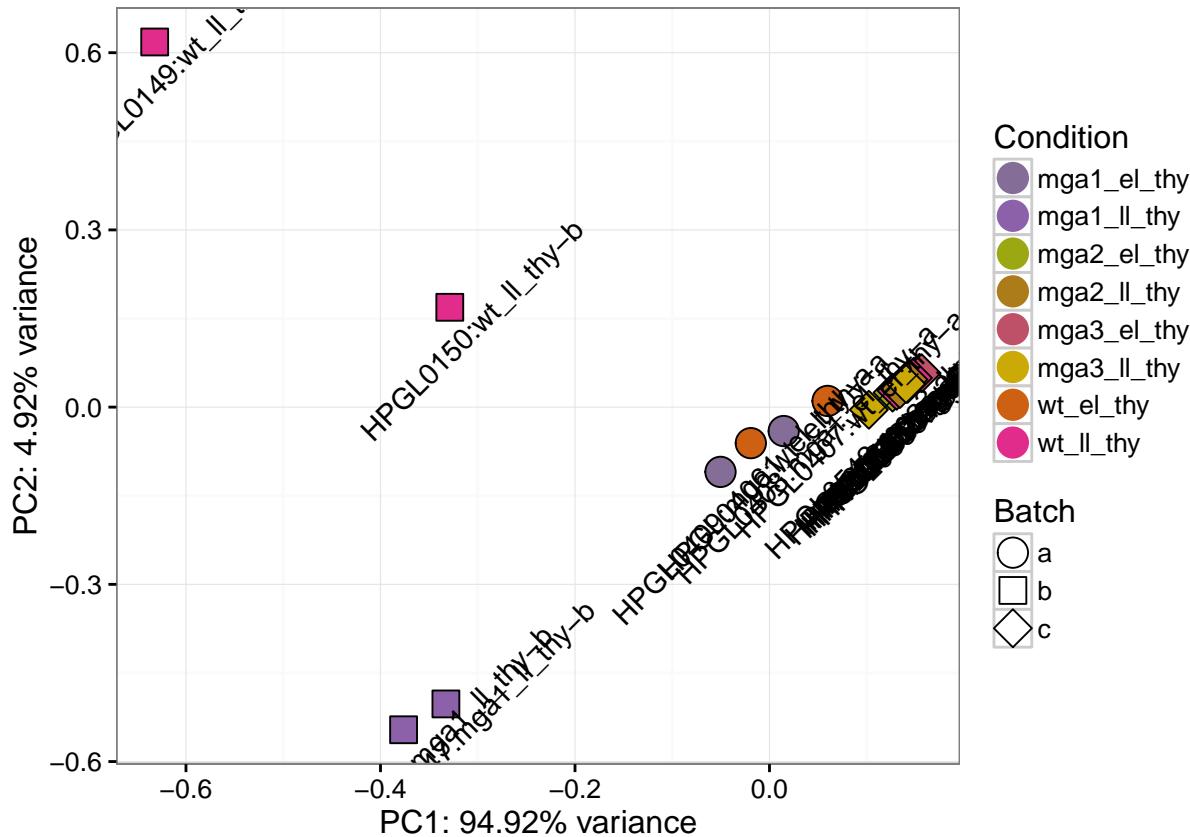
## This data will benefit from being displayed on the log scale.

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

## No id variables; using all as measure variables

```





```
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/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).

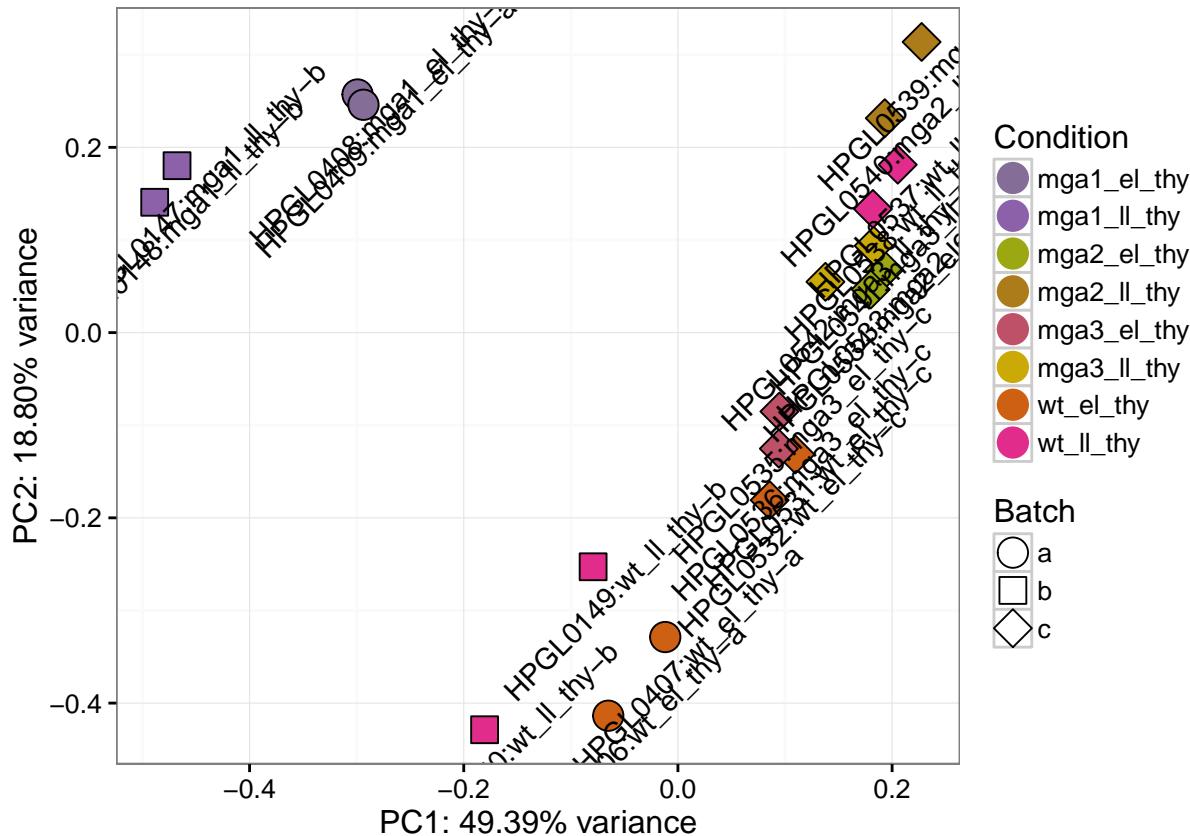
## Leaving the data unnormalized. This is necessary for DESeq, but
## EdgeR/limma might benefit from normalization. Good choices include quantile,
## size-factor, tmm, etc.

## Not correcting the count-data for batch effects. If batch is
## included in EdgerR/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!)

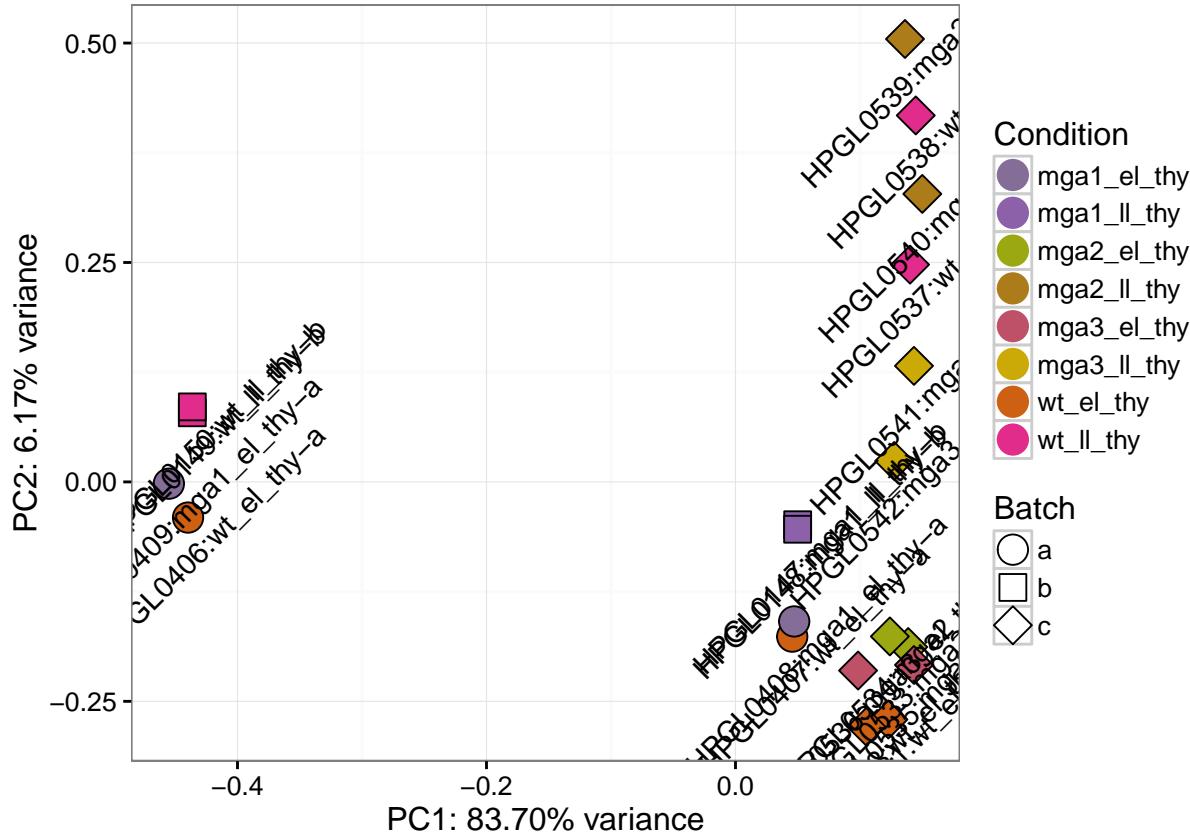
## Leaving the data in its current base format, keep in mind that
##   some metrics are easier to see when the data is log2 transformed, but
##   EdgeR/DESeq don't like transformed data.

## 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!)

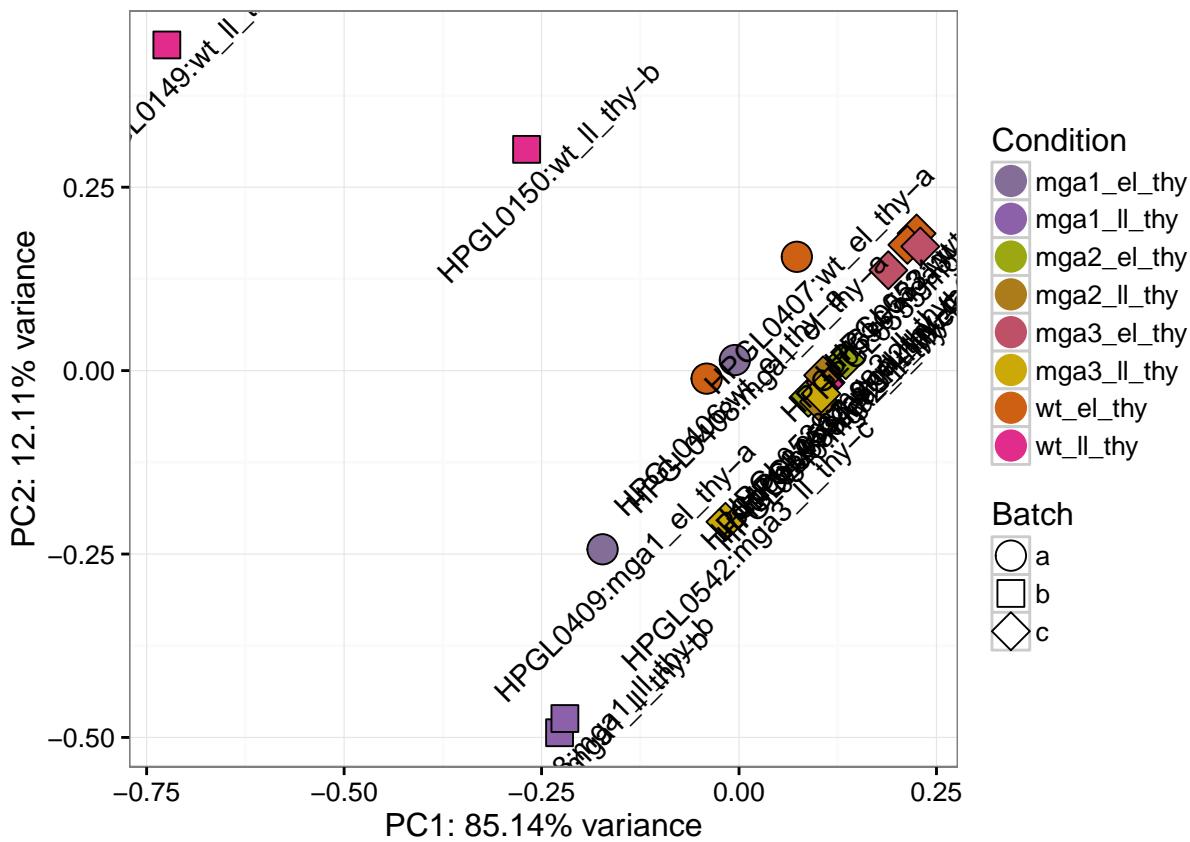
## Leaving the data in its current base format, keep in mind that
##   some metrics are easier to see when the data is log2 transformed, but
##   EdgeR/DESeq don't like transformed data.

## 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.
```

```
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 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).

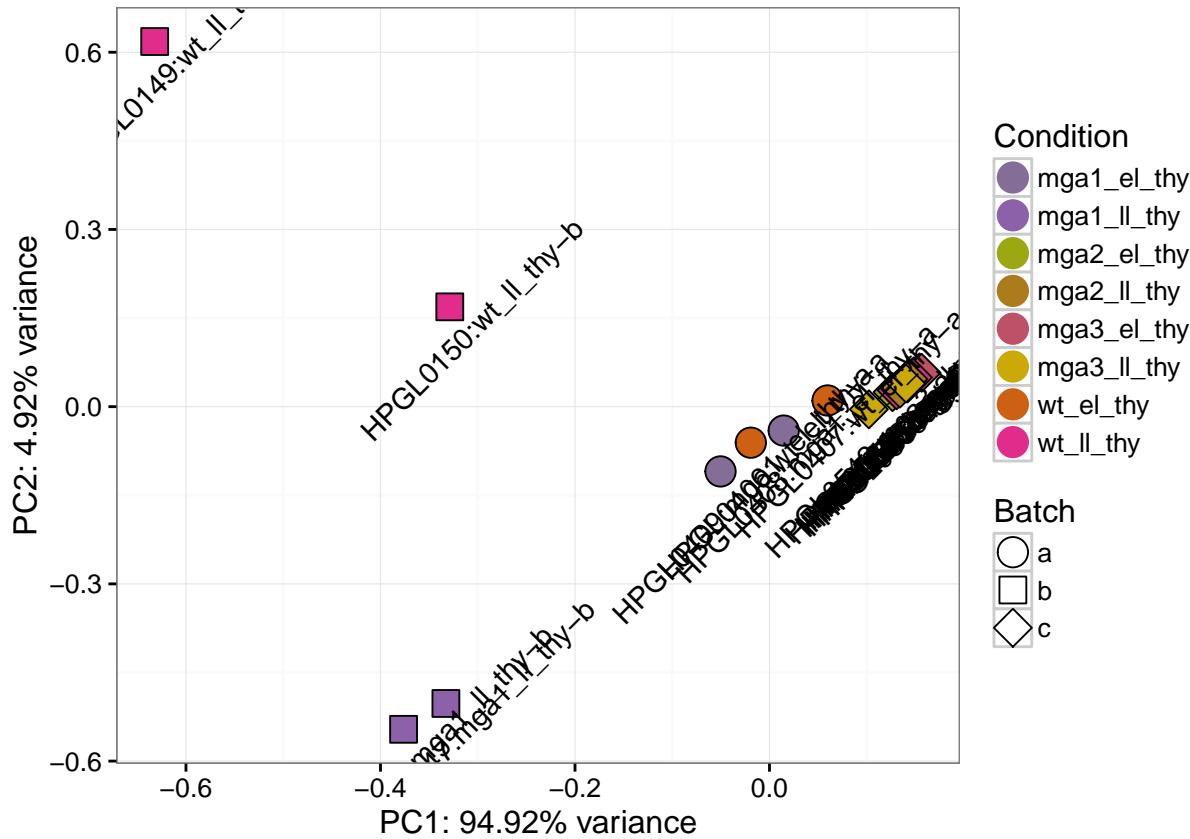
## Leaving the data unnormalized. This is necessary for DESeq, but
## EdgeR/limma might benefit from normalization. Good choices include quantile,
## size-factor, tmm, etc.

## 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.

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

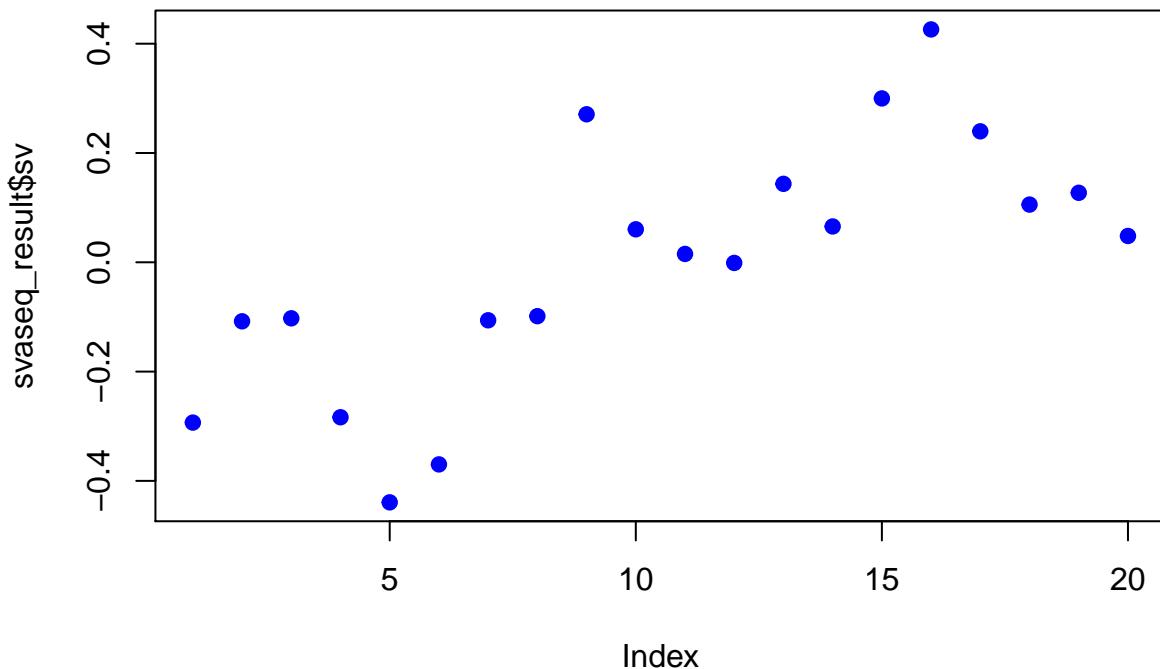
## data %*% (Id - mod %*% blah blah requires numeric/complex arguments.

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

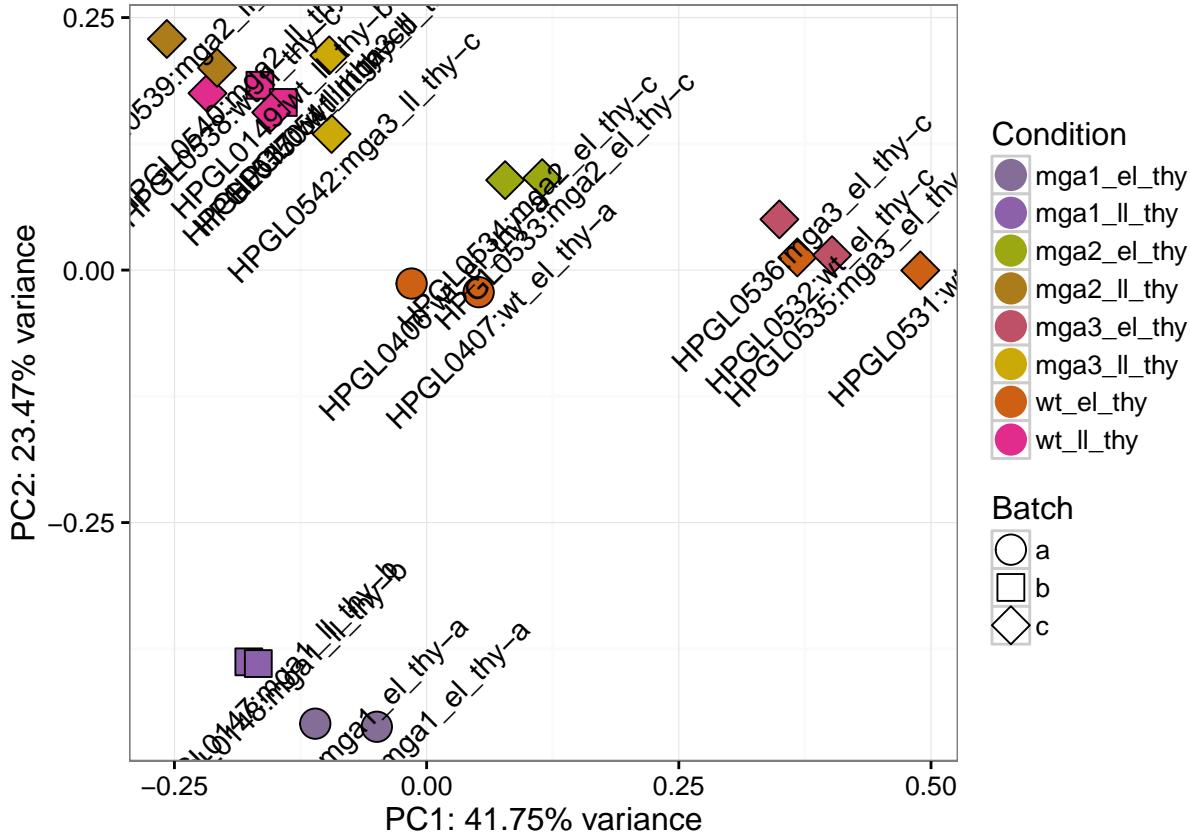
## The number of elements which are < 0 after batch correction is: 276

## 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.

```

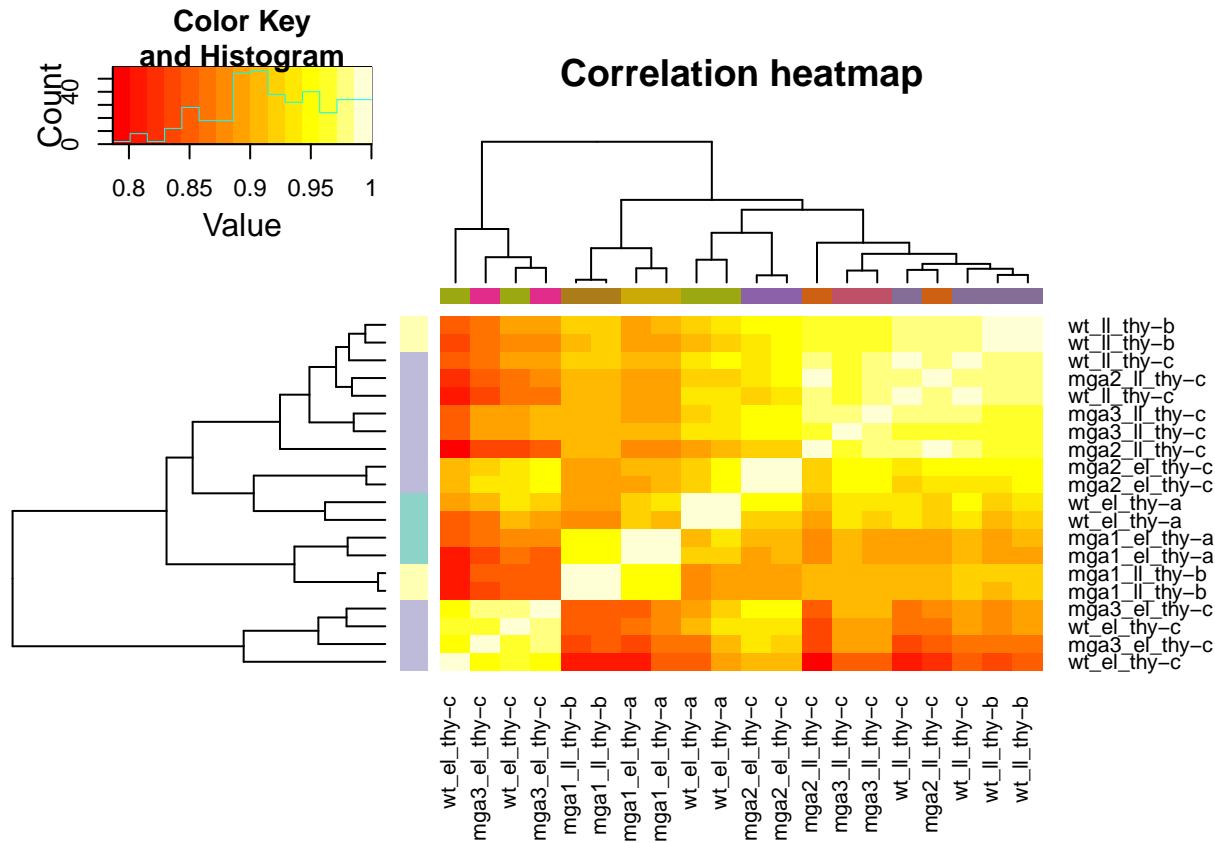
```

## Graphing a boxplot.

## Graphing a correlation heatmap.

## Graphing a standard median correlation.

```

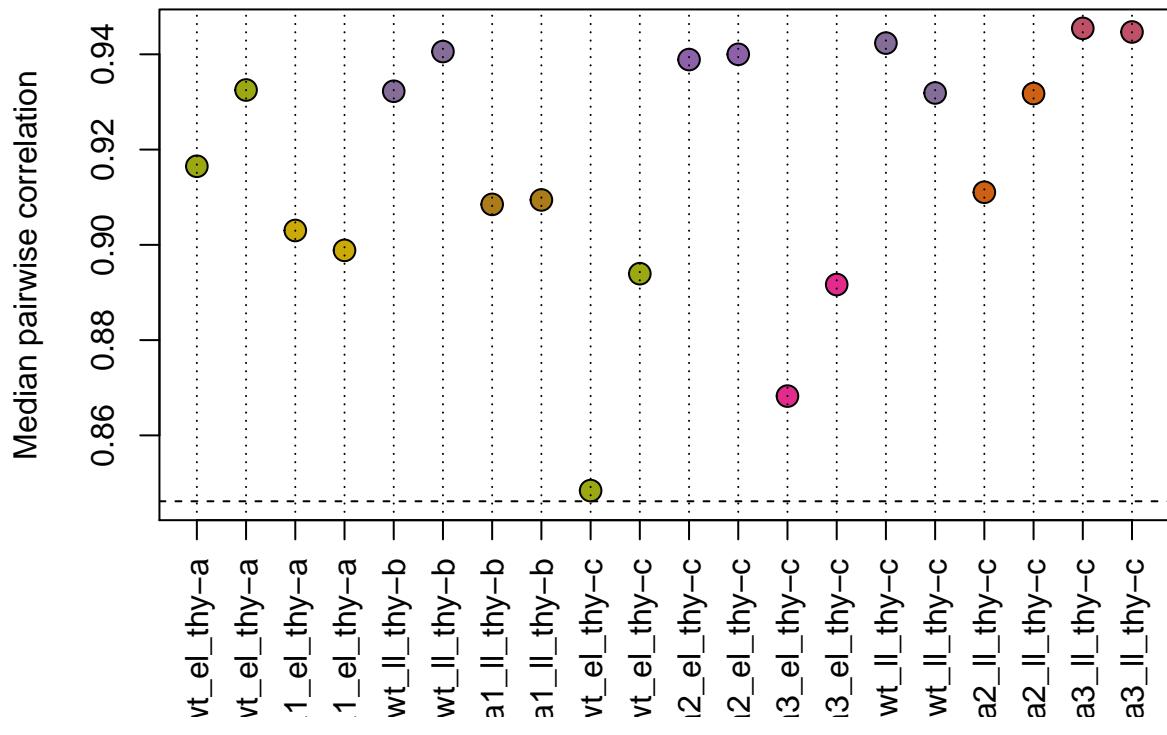


```

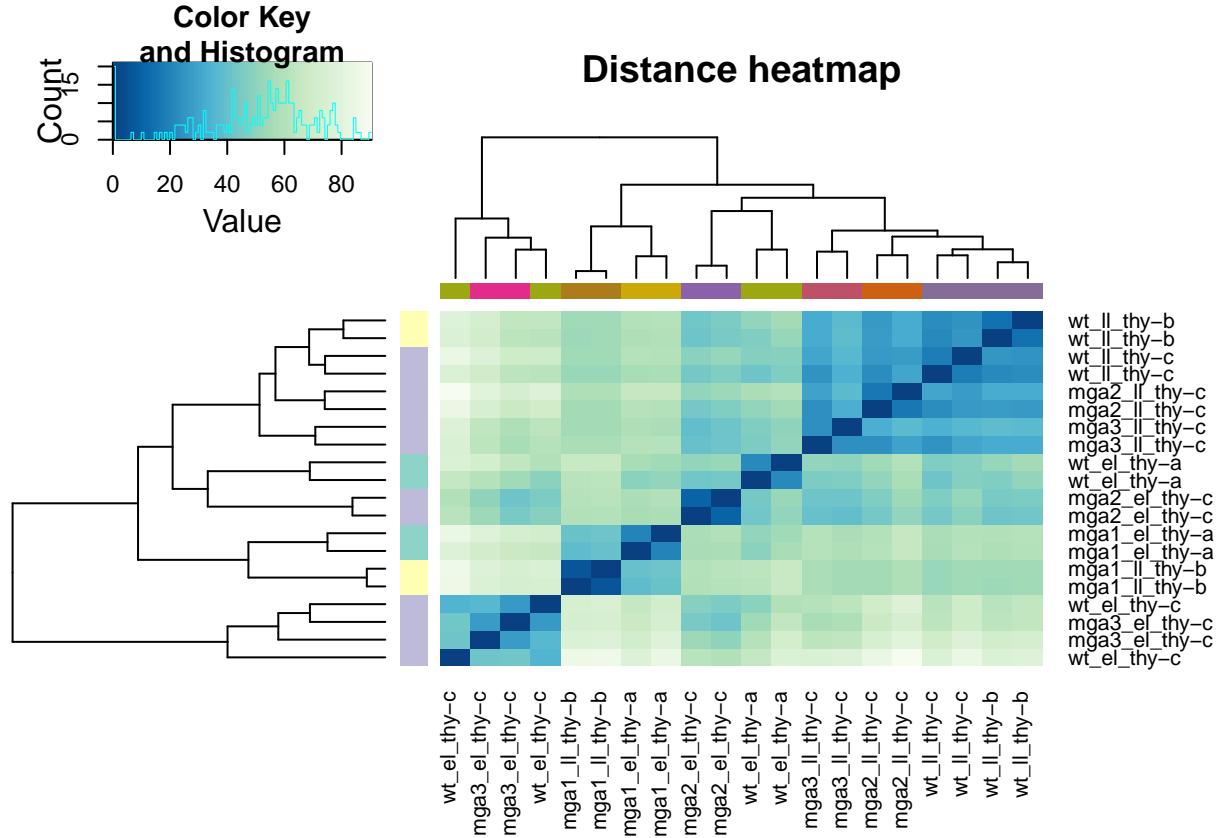
## Graphing a distance heatmap.

```

## Standard Median Correlation

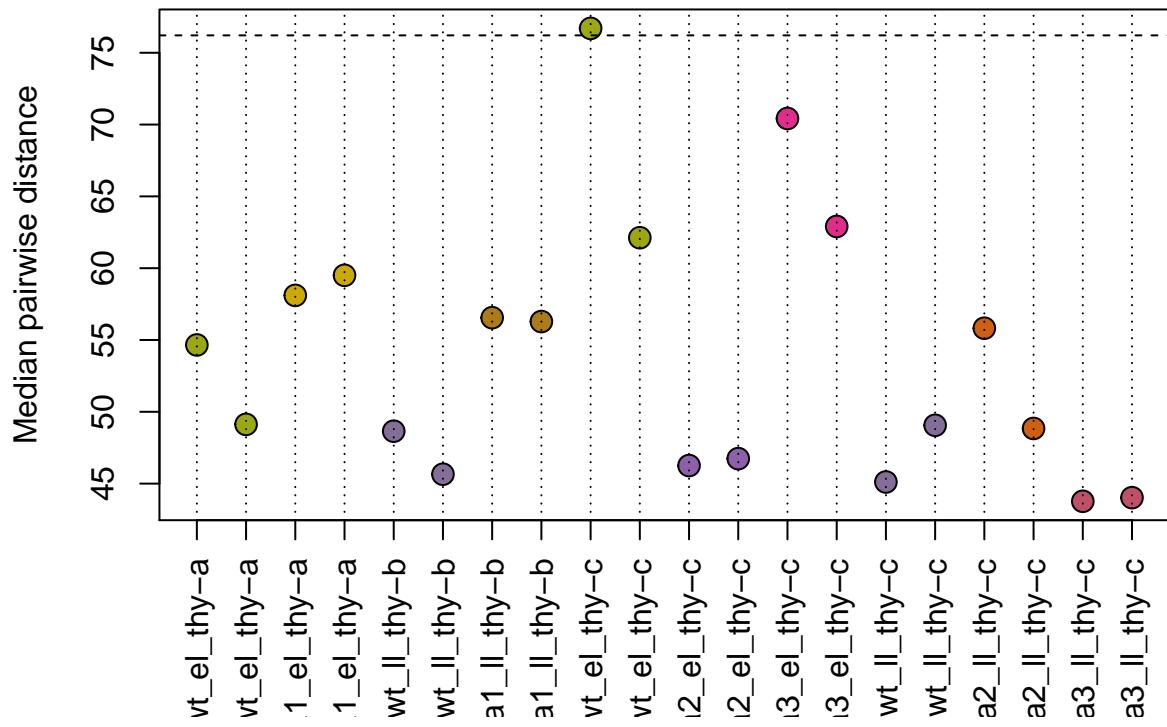


## Graphing a standard median distance.



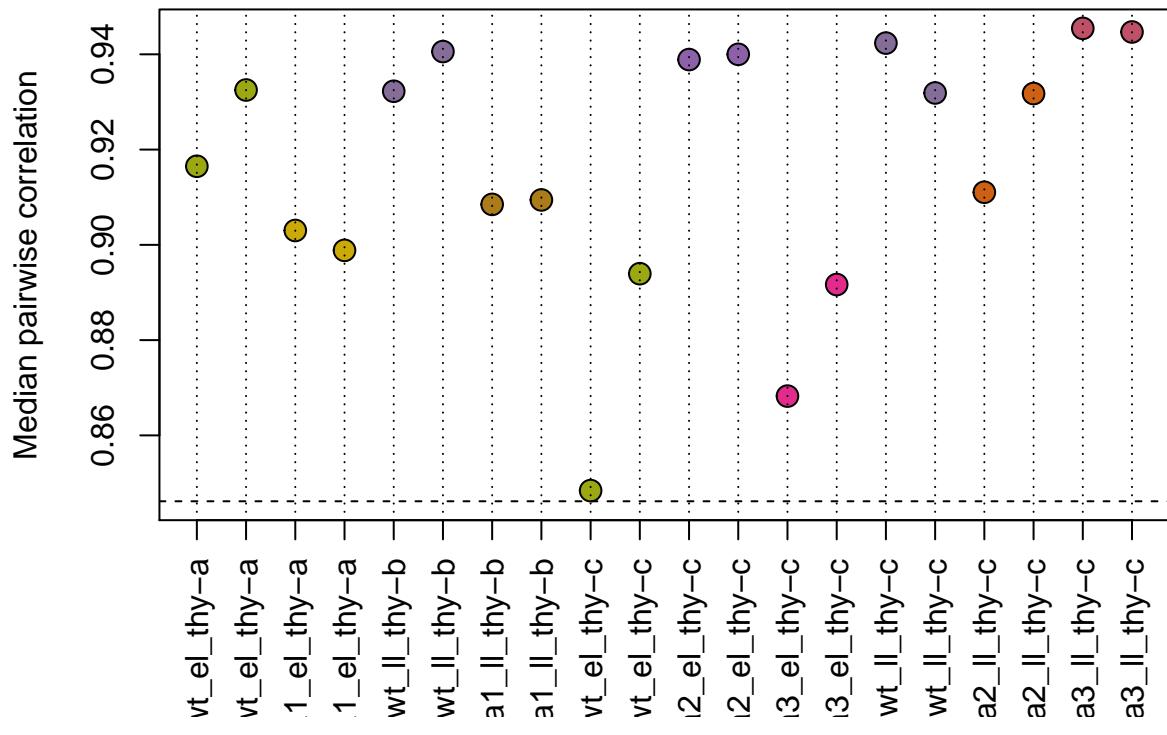
```
## Graphing a PCA plot.  
## Plotting a density plot.
```

### Standard Median Distance

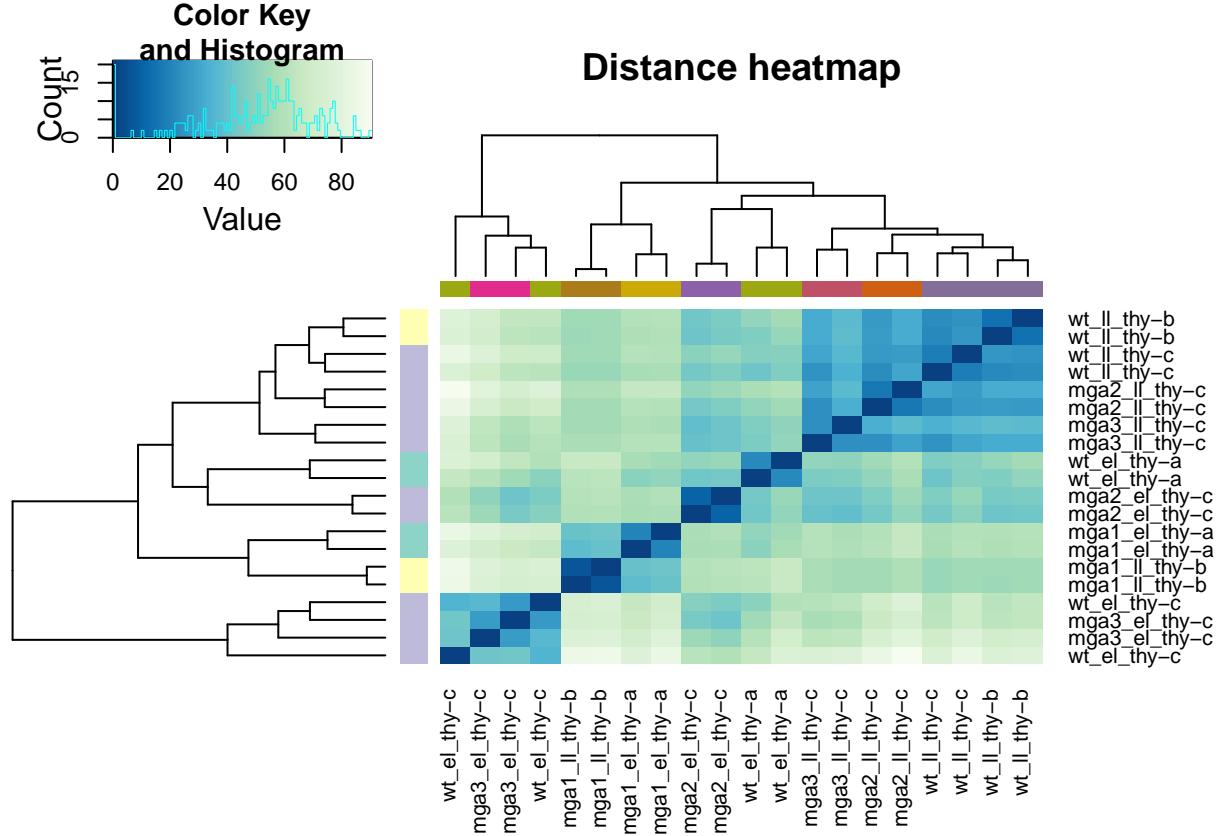


```
norm_graphs$smc
```

## 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.
```