

# Using hpgltools for fun and profit

Ashton Trey Belew [abelew@gmail.com](mailto:abelew@gmail.com)

2015-04-27

## Hpgltools: Stupid R tricks.

The following block shows how I handle autoloading requisite libraries for my code. This makes it easier for me to download/install the R requirements on a new computer, something which I have found myself needing to do more than I would have guessed.

```
## This block serves to load requisite libraries and set some options.
library("hpgltools")
## To set up an initial vignette, use the following line:
## devtools::use_vignette("hpgltools")
 autoloads_all()

##
## groupGOTerms:    GOBPTerm, GOMFTerm, GOCTerm environments built.

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

opts_knit$set(progress=TRUE, verbose=TRUE, stop_on_error=FALSE, purl=FALSE, error=TRUE, fig.width=7, fig.height=5)
options(java.parameters="-Xmx8g") ## used for xlconnect -- damn 4g wasn't enough
theme_set(theme_bw(base_size=10))
set.seed(1)
```

## Rendering the vignette

The following block has a few lines I use to load data, save it, and render pdf/html reports. I do this under the veritable editor, ‘emacs,’ with the key combination “Control-c, Control-n” for each line I want to evaluate in R, or “Control-c, Control-c” for a paragraph.

```
load("RData")
rm(list=ls())
save(list=ls(all=TRUE), file="RData")
render("hpgltools.Rmd", output_format="pdf_document")
render("hpgltools.Rmd", output_format="html_document")
```

## Tasks that hpgltools helps me perform

This code was written to speed up and simplify a few specific tasks:

- Reading RNA sequencing count tables (in R/count\_tables.R)
- Normalization of data (R/normalization.R)
- Graphing metrics of data to check and evaluate batch effects (R/plots.R)

- Performing contrasts of the data using voom/limma (R/misc\_functions.R)
- Plotting RNA abundances by condition/batch (R/plots.R)
- Simplifying ontology/KEGG searches (R/ontology.R)

The following paragraphs will attempt to show how I use it.

## Annotation information

Every RNA sequencing experiment I have played with has required a different handling of the genome's annotation. Most, but not all, have kept the data of interest in a gff file. Here is an example of how I process one of those files and make a data frame of genes as well as tooltips, which will be used for googleVis graphs later. In every experiment I have played with, I make a 'reference' directory into which I copy the current annotation data, this way I have a consistent and known version of the annotation. In the example below, this is the TriTrypDB version 8.1 of the *T. cruzi* genome.

```
tcruzi_annotations = import.gff3("reference/gff/clbrener_8.1_complete.gff.gz")
annotation_info = as.data.frame(tcruzi_annotations)

genes = annotation_info[annotation_info$type=="gene",]
gene_annotations = genes
rownames(genes) = genes$name
tooltip_data = genes
tooltip_data = tooltip_data[,c(11,12)]
tooltip_data$tooltip = paste(tooltip_data$name, tooltip_data$description, sep=": ")
tooltip_data$tooltip = gsub("\\\\+", " ", tooltip_data$tooltip)
rownames(tooltip_data) = tooltip_data$name
tooltip_data = tooltip_data[-1]
tooltip_data = tooltip_data[-1]
colnames(tooltip_data) = c("name.tooltip")
head(tooltip_data)
```

## Reading count tables

In Dr. El-Sayed's lab, there is a very specific naming convention for RNA sequencing experiments. Every sequencing run has an 'HPGL' (host pathogen genomics lab) identifier. All experiments have associated metadata, including the condition in the experiment, the batch, bioanalyzer reports, etc. When I play with data, I keep all this information in a csv file 'samples.csv' and the processed count-tables for the experiment in a specific directory: processed\_data/. Therefore, I have a couple functions which automate the import of data into R in the hopes that no mistakes are made.

Here is an example from a recent experiment.

```
samples = read.csv("data/all_samples.csv")
knitr::kable(head(samples))
```

Sample.ID	Type	Stage	batch	Media	SRA	Reads.Passed	ncRNA	X..ncRNA	Remaining	Genome	X..G
HPGL0406	WT	EL	1	THY	NA	19026277	353992	1.86%	18672285	17810587	95.3
HPGL0407	WT	EL	2	THY	NA	15074073	259613	1.72%	14814460	14334043	96.7
HPGL0408	mga	EL	1	THY	NA	17112233	293752	1.72%	16818481	15769581	93.7

Sample.ID	Type	Stage	batch	Media	SRA	Reads.Passed	ncRNA	X..ncRNA	Remaining	Genome	X..C
HPGL0409	mga	EL	2	THY	NA	18298278	339862	1.86%	17958416	16553148	92.1
HPGL0149	WT	LL	1	THY	NA	39107368	8055417	20.60%	31051951	26285560	84.6
HPGL0150	WT	LL	2	THY	NA	35429033	3705275	10.46%	31723758	30012962	94.6

Since I didn't want to copy over all my count tables, you, dear reader, will have to trust that there is a file for each entry in the above table which corresponds to the Sample.ID. These may be organized by sample name or condition. The following code shows how I create an expressionset and fill it with the count data.

```
example_data = counts(make_exampledatalog(ngenes=10000, columns=24))
## create_expt() usually expects that there are a bunch of count tables
## from htseq in the directory: processed_data/count_tables/
## These may be organised in separate directories by condition(type)
## in one directory each by sample. By default, this assumes they will be
## named sample_id.count.gz, but this may be changed with the suffix argument.
all_expt = create_expt("data/all_samples.csv", count_dataframe=example_data)

## [1] "This function needs the conditions and batches to be an explicit column in the sample sheet."
## [1] "Please note that thus function assumes a specific set of columns in the sample sheet:"
## [1] "The most important ones are: Sample.ID, Stage, Type."
## [1] "Other columns it will attempt to create by itself, but if"
## [1] "batch and condition are provided, that is a nice help."
```

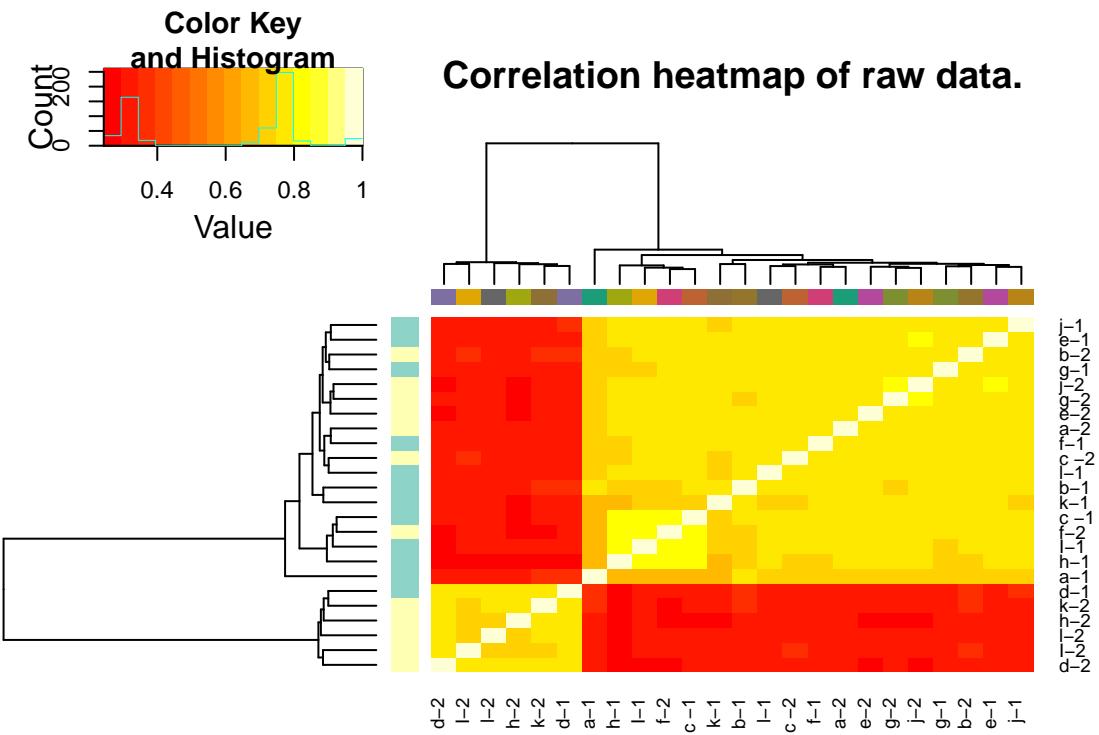
## Examining data

Once the data is read in, the first task is always to look at it and evaluate for batch effects and thus decide what to do about them. However, different normalization methods are appropriate in different data sets, therefore I have some functions which attempt to make this easier. For this, I will make a dummy data set using limma's makeExampleData()

```
## graph_metrics() performs the following:
## runs a libsize plot, non-zero genes plot, boxplot, correlation/distance heatmaps, and pca plots
## It performs a normalization of the data (log2(quantile(cpm)) by default), and does it again
## It then uses limma's removeBatchEffect() to make a stab at removing batch effect, and does it again.

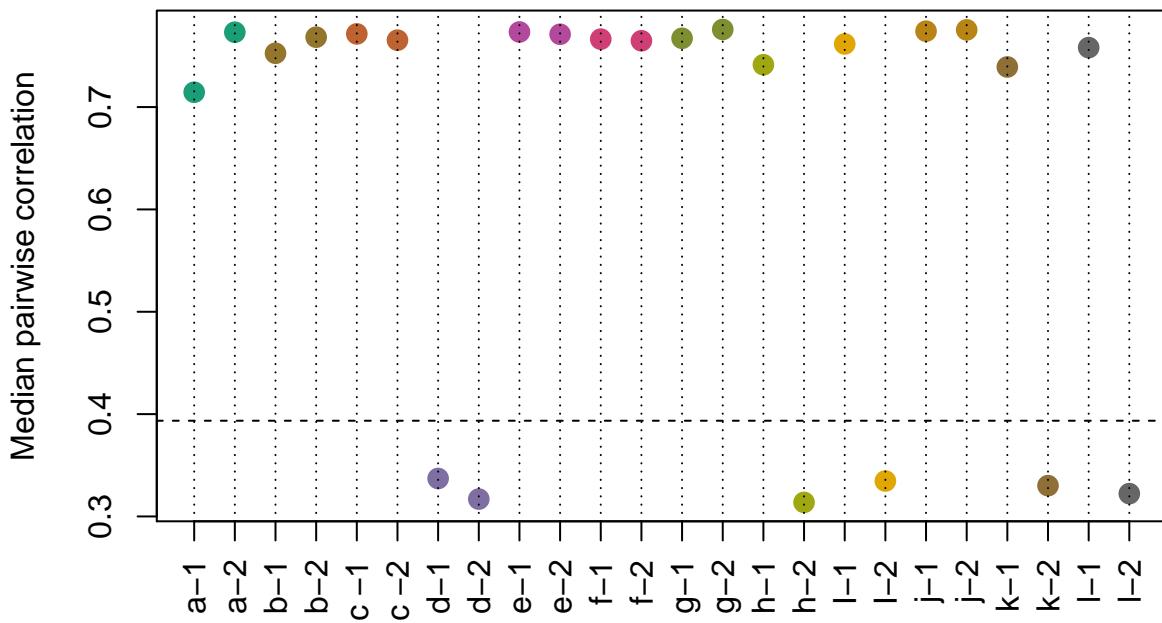
## An important thing to remember: the data from makeExampleData() is not very interesting, so the results
## plots are also not interesting...
fun = graph_metrics(expt=all_expt)

## Graphing number of non-zero genes with respect to CPM by library.
## Graphing library sizes.
## Adding log10
## Graphing a boxplot on log scale.
## Graphing a correlation heatmap.
## Graphing a standard median correlation.
```

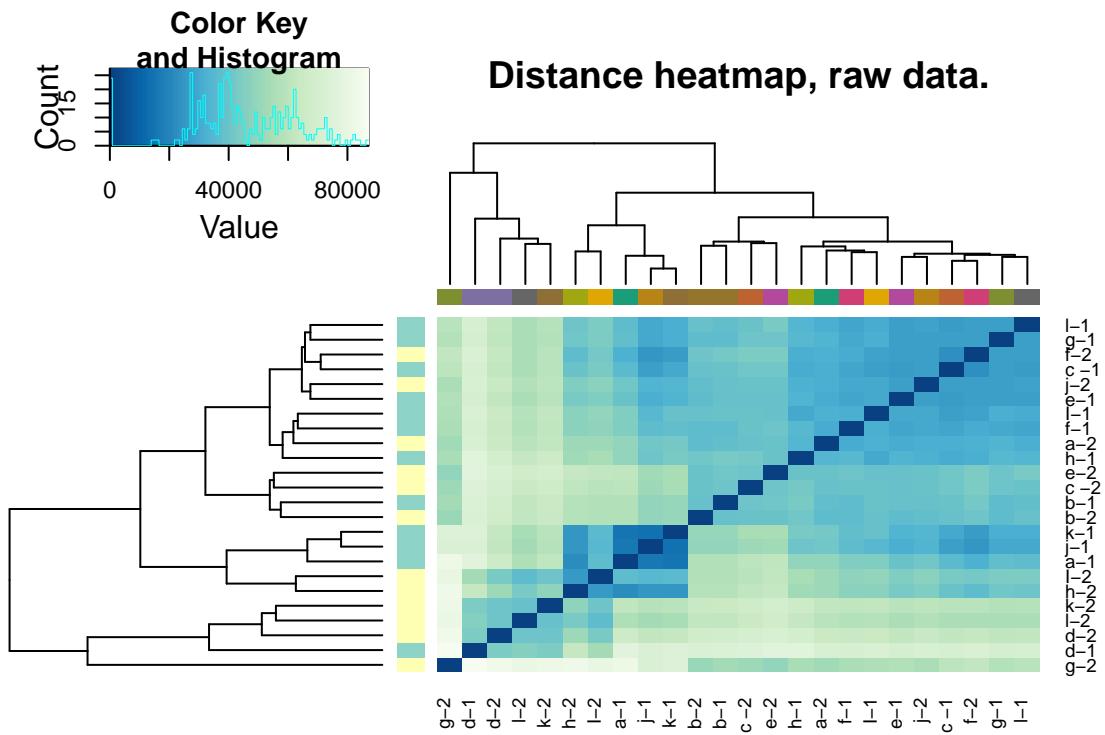


```
## Graphing a distance heatmap.
```

**Standard Median Correlation, raw data.**

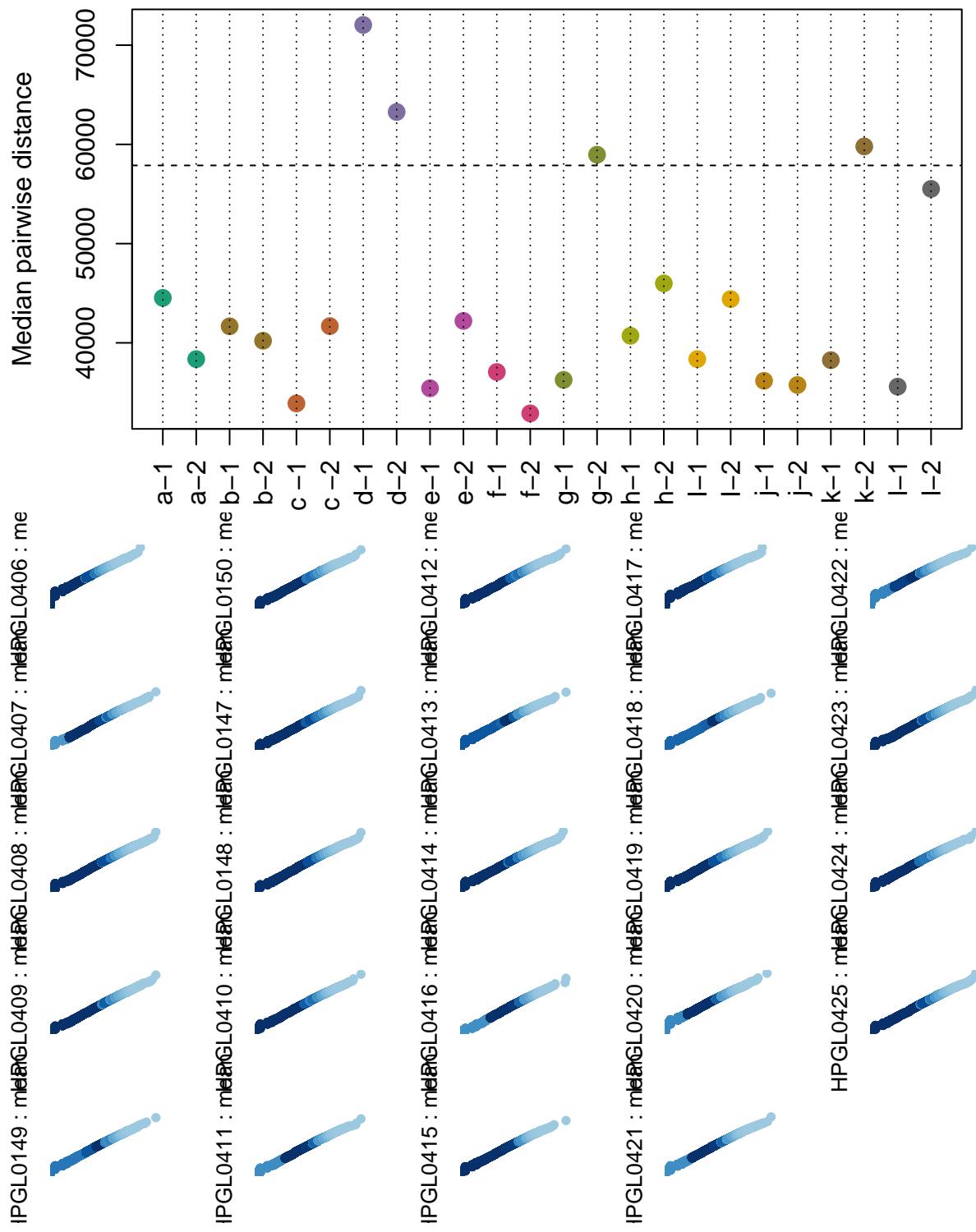


```
## Graphing a standard median distance.
```



```
## Graphing a PCA plot.
## Plotting a density plot.
## This plot looks neat if you do position='fill' or position='stack'
## QQ plotting!.
```

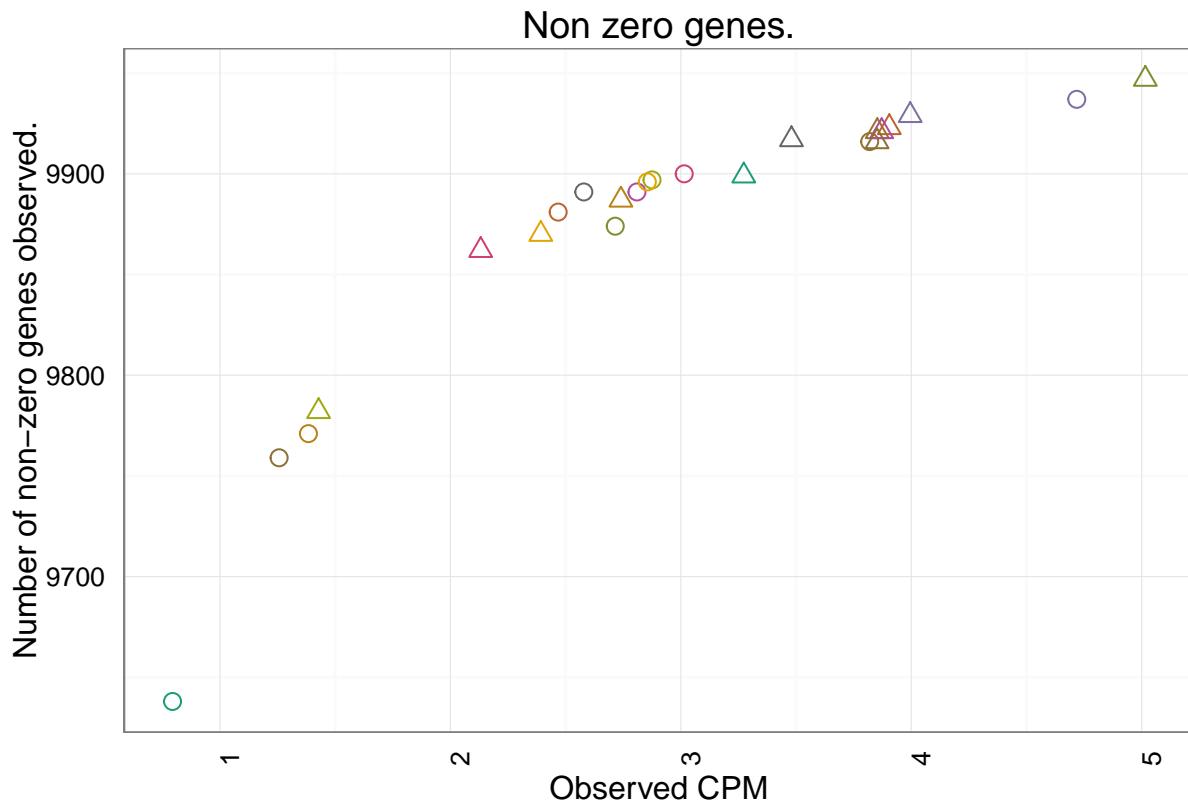
## Standard Median Distance, raw data.



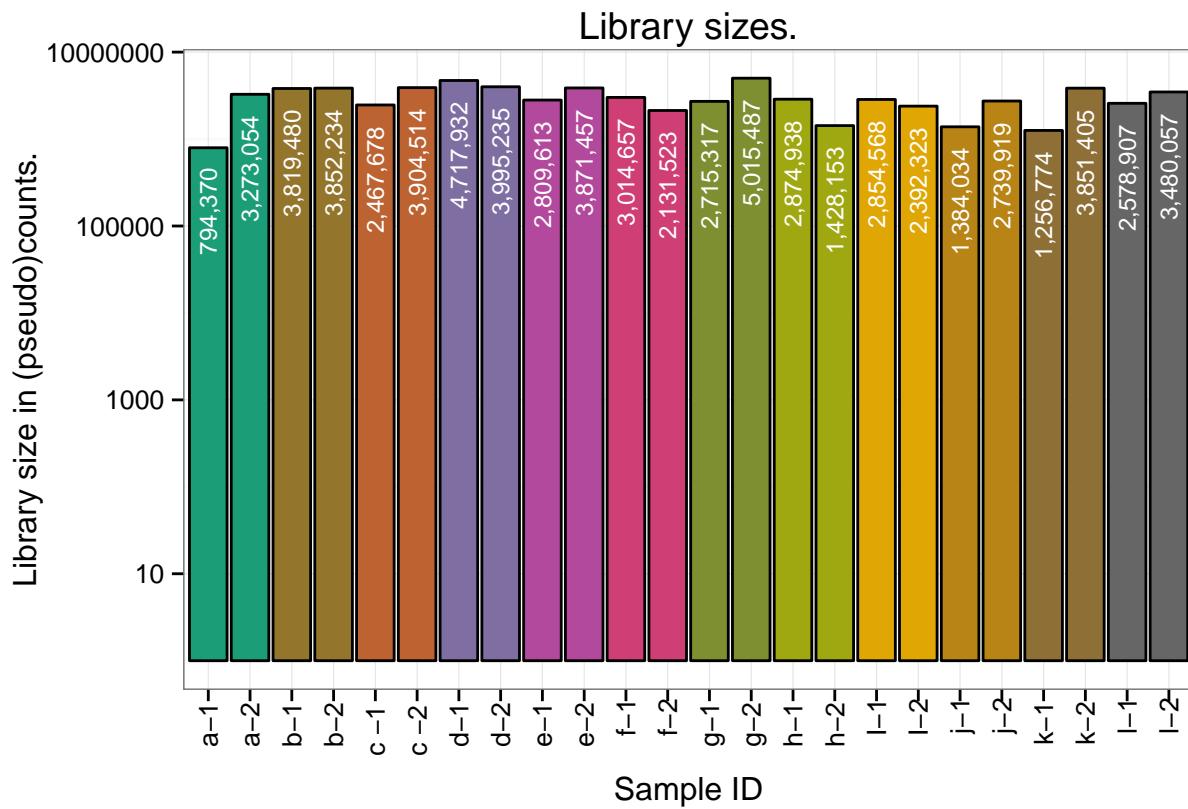
```
GL0149 :HPGL0409 :HPGL0408 :HPGL0407 :HPGL0406 : m  
GL0411 :HPGL0410 :HPGL0418 :HPGL0417 :HPGL04150 : m  
GL0415 :HPGL0416 :HPGL0414 :HPGL0413 :HPGL0412 : m  
GL0421 :HPGL0420 :HPGL0419 :HPGL0418 :HPGL0417 : m  
HPGL0425 :HPGL0424 :HPGL0423 :HPGL0422 : m
```

fun

```
## $nonzero
```

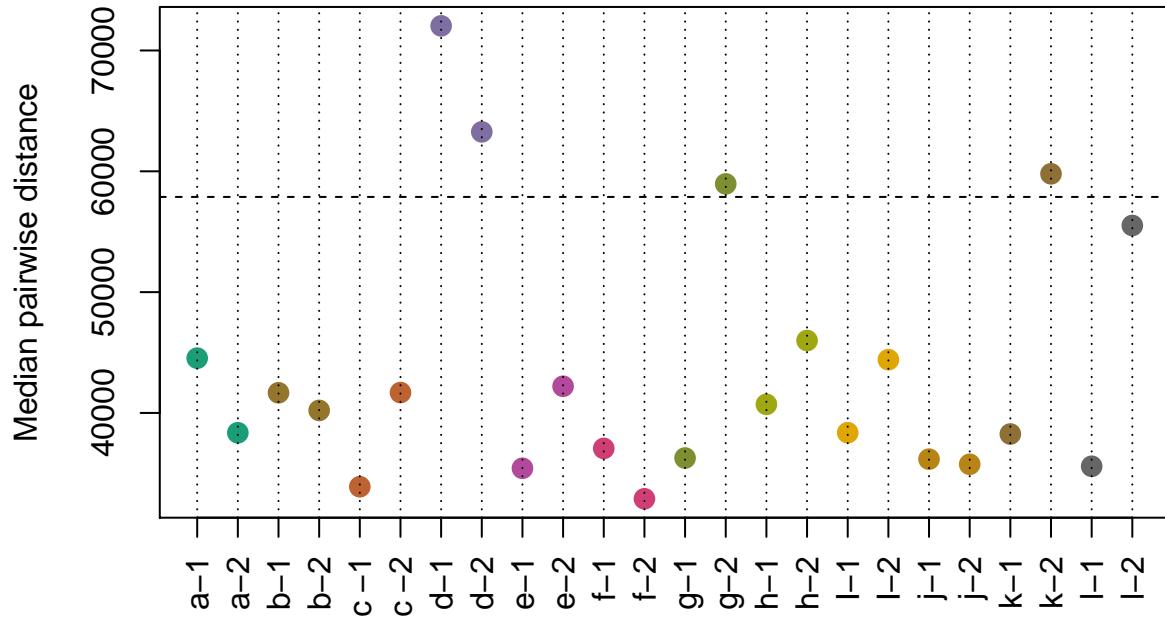


```
##  
## $libsize
```



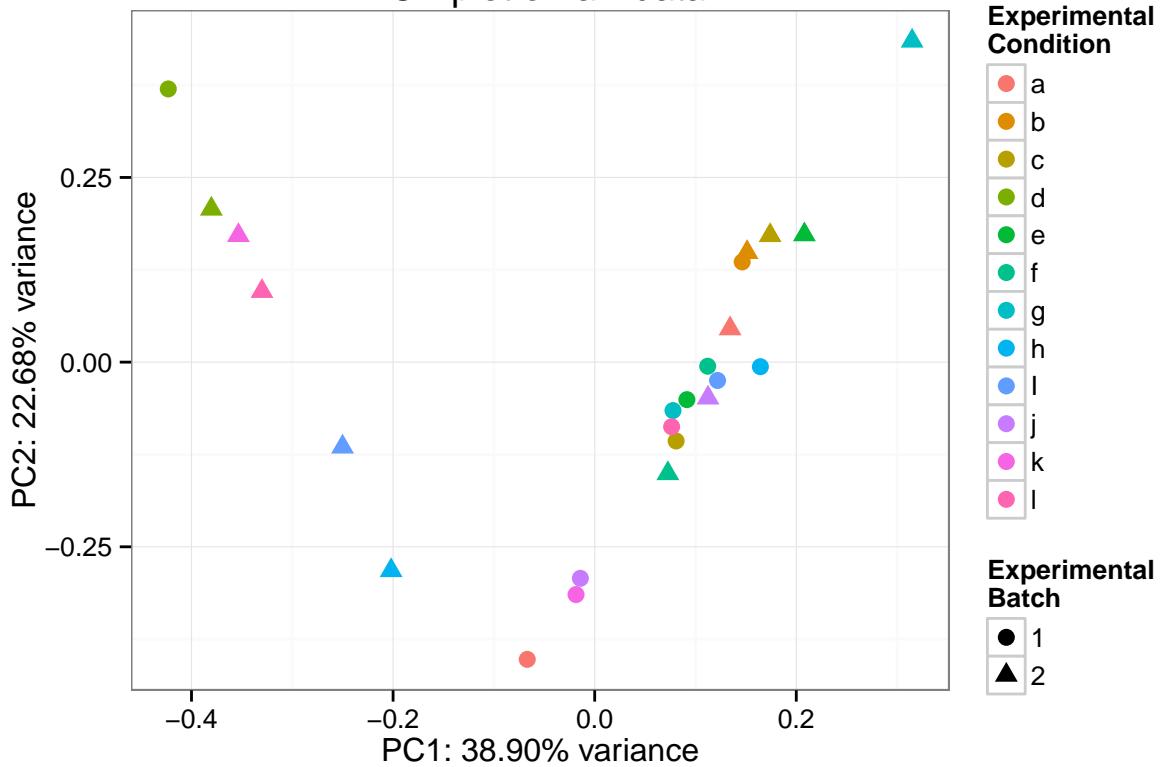
```
##  
## $boxplot
```

### Standard Median Distance, raw data.



```
##  
## $corheat  
##  
## $smc  
##  
## $disheat  
##  
## $smd  
##  
## $pcaplot
```

PCA plot of raw data.



```
##
## $pcatable
##           SampleID condition batch batch_int      PC1      PC2
## HPGL0406     a-1       a     1  1 -0.06691092 -0.402355331
## HPGL0407     a-2       a     2  2  0.13431332  0.045259223
## HPGL0408     b-1       b     1  1  0.14634248  0.135682093
## HPGL0409     b-2       b     2  2  0.15119716  0.148552727
## HPGL0149     c -1      c     1  1  0.08078548 -0.106755934
## HPGL0150     c -2      c     2  2  0.17403628  0.171498664
## HPGL0147     d -1      d     1  1 -0.42304652  0.369802408
## HPGL0148     d -2      d     2  2 -0.38027775  0.207290683
## HPGL0410     e -1      e     1  1  0.09148588 -0.050776835
## HPGL0411     e -2      e     2  2  0.20806087  0.172566966
## HPGL0412     f -1      f     1  1  0.11211147 -0.005541911
## HPGL0413     f -2      f     2  2  0.07258768 -0.150795269
## HPGL0414     g -1      g     1  1  0.07767734 -0.065569614
## HPGL0416     g -2      g     2  2  0.31491903  0.434733324
## HPGL0415     h -1      h     1  1  0.16438931 -0.006171446
## HPGL0417     h -2      h     2  2 -0.20194956 -0.282064901
## HPGL0418     I -1      I     1  1  0.12188367 -0.024827186
## HPGL0419     I -2      I     2  2 -0.25012222 -0.114756653
## HPGL0420     j -1      j     1  1  -0.01422104 -0.292626128
## HPGL0421     j -2      j     2  2  0.11244544 -0.048430048
## HPGL0422     k -1      k     1  1  -0.01852333 -0.314655028
## HPGL0423     k -2      k     2  2 -0.35343798  0.171561901
## HPGL0424     l -1      l     1  1  0.07632821 -0.087452287
## HPGL0425     l -2      l     2  2 -0.33007430  0.095830582
##
```

```

## $pcares
##   propVar cumPropVar cond.R2 batch.R2
## 1    38.90     38.90   65.58    2.02
## 2    22.68     61.58   48.03   12.08
## 3     4.18     65.76   55.85    2.90
## 4     3.47     69.23   60.55    1.22
## 5     3.06     72.29   13.77   16.57
## 6     2.91     75.20   51.94    0.01
## 7     2.83     78.03   58.16    2.50
## 8     2.60     80.63   40.48    3.92
## 9     2.57     83.20   29.00    2.97
## 10    2.50     85.70   31.46    6.33
## 11    1.99     87.69   55.96    4.23
## 12    1.64     89.33   50.20    9.96
## 13    1.56     90.89   52.36    0.16
## 14    1.43     92.32   33.43    0.01
## 15    1.27     93.59   45.31    0.17
## 16    1.25     94.84   50.07   16.02
## 17    1.18     96.02   46.71    3.30
## 18    1.12     97.14   60.63    0.07
## 19    1.00     98.14   40.19    0.77
## 20    0.84     98.98   54.10    6.48
## 21    0.45     99.43   53.26    8.16
## 22    0.34     99.77   54.93    0.10
## 23    0.22     99.99   48.03    0.06
##
## $pcavar
## [1] 38.90 22.68  4.18  3.47  3.06  2.91  2.83  2.60  2.57  2.50  1.99
## [12]  1.64  1.56  1.43  1.27  1.25  1.18  1.12  1.00  0.84  0.45  0.34
## [23]  0.22
##
## $density

```

```

>GL0149 :HPGL0409 :HPGL0408 :HPGL0407 :HPGL0406 : m
[1] 0.09966

>GL0411 :HPGL0410 :HPGL0418 :HPGL0417 :HPGL04150 : m
[1] 0.09966

>GL0415 :HPGL0416 :HPGL0414 :HPGL0413 :HPGL0412 : m
[1] 0.09966

>GL0421 :HPGL0420 :HPGL0419 :HPGL0418 :HPGL0417 : m
[1] 0.09966

HPGL0425 :HPGL0424 :HPGL0423 :HPGL0422 : m
[1] 0.09966

```

##  
## \$qq  
## \$qq\$logs  
##  
## \$qq\$ratios  
##  
## \$qq\$medians  
## \$qq\$medians[[1]]  
## [1] -1.485  
##  
## \$qq\$medians[[2]]  
## [1] -0.08172  
##  
## \$qq\$medians[[3]]  
## [1] 0.0755  
##  
## \$qq\$medians[[4]]  
## [1] 0.0855  
##  
## \$qq\$medians[[5]]  
## [1] -0.3545  
##  
## \$qq\$medians[[6]]  
## [1] 0.09966  
##  
## \$qq\$medians[[7]]

```

## [1] 0.2966
##
## $qq$medians[[8]]
## [1] 0.1217
##
## $qq$medians[[9]]
## [1] -0.2263
##
## $qq$medians[[10]]
## [1] 0.1006
##
## $qq$medians[[11]]
## [1] -0.1516
##
## $qq$medians[[12]]
## [1] -0.4996
##
## $qq$medians[[13]]
## [1] -0.2793
##
## $qq$medians[[14]]
## [1] 0.338
##
## $qq$medians[[15]]
## [1] -0.2126
##
## $qq$medians[[16]]
## [1] -0.914
##
## $qq$medians[[17]]
## [1] -0.2255
##
## $qq$medians[[18]]
## [1] -0.3956
##
## $qq$medians[[19]]
## [1] -0.9252
##
## $qq$medians[[20]]
## [1] -0.2499
##
## $qq$medians[[21]]
## [1] -1.044
##
## $qq$medians[[22]]
## [1] 0.1029
##
## $qq$medians[[23]]
## [1] -0.342
##
## $qq$medians[[24]]
## [1] -0.007835

```

```
## The following are some examples of other ways to make use of these plots:
```

```
##fun_boxplot = hpgl_boxplot(df=fun)
##print(fun_boxplot)
##log_boxplot = hpgl_boxplot(df=fun, scale="log")
##print(log_boxplot)
##hpgl_corheat(df=fun, colors=hpgl_colors)
##hpgl_disheat(df=fun, colors=hpgl_colors)
##hpgl_smc(df=fun, colors=hpgl_colors)
##hpgl_libsize(df=fun)
##hpgl_qq_all(df=fun)
```

## Normalizing data

RNAseq data must be normalized. Here is one easy method:

```
## normalize_expt will do this on the expt class, replace the expressionset therein, and
## make a backup of the data inside the expt class.
norm_expt = normalize_expt(all_expt)
```

```
## [1] "This function will replace the expt$expressionset slot with the raw(raw(raw))'d data."
## [1] "It saves the current data into a slot named: expt$backup_expressionset"
## [1] "It will also save copies of each step along the way in expt$normalized with the corresponding lib
## [1] "Keep the libsizes in mind when invoking limma. The appropriate libsize is the non-log(cpm(normalized
## [1] "This is most likely kept in the slot called: 'new_expt$normalized$normalized_counts$libsize' whi
## [1] "Filter low is false, this should likely be set to something, good choices include ccbc, kofa, p
## [1] "Leaving the data in its current base format, keep in mind that some metrics are easier to see w
## [1] "Leaving the data unconverted. It is often advisable to cpm/rpkm the data to normalize for samp
## [1] "Leaving the data unnormalized. This is necessary for DESeq, but EdgeR/limma might benefit from
## [1] "Not correcting the count-data for batch effects. If batch is included in EdgerR/limma's model,
```

```
head(exprs(norm_expt$expressionset))
```

	HPGL0406	HPGL0407	HPGL0408	HPGL0409	HPGL0149	HPGL0150	HPGL0147
## gene_1_F	10	155	326	552	248	92	206
## gene_2_T	127	1375	923	405	96	614	325
## gene_3_F	5	45	18	63	38	38	49
## gene_4_F	6	62	70	33	29	15	38
## gene_5_F	13	149	134	133	153	90	273
## gene_6_F	441	960	1884	650	604	817	1096
	HPGL0148	HPGL0410	HPGL0411	HPGL0412	HPGL0413	HPGL0414	HPGL0416
## gene_1_F	218	81	323	243	72	185	576
## gene_2_T	304	422	722	698	367	956	1380
## gene_3_F	52	47	35	76	28	34	86
## gene_4_F	37	30	26	24	10	22	34
## gene_5_F	87	187	228	132	185	107	138
## gene_6_F	1450	781	1252	1101	1017	706	833
	HPGL0415	HPGL0417	HPGL0418	HPGL0419	HPGL0420	HPGL0421	HPGL0422
## gene_1_F	374	92	144	163	121	78	57
## gene_2_T	260	121	769	175	285	199	182
## gene_3_F	45	19	38	27	22	7	47

```

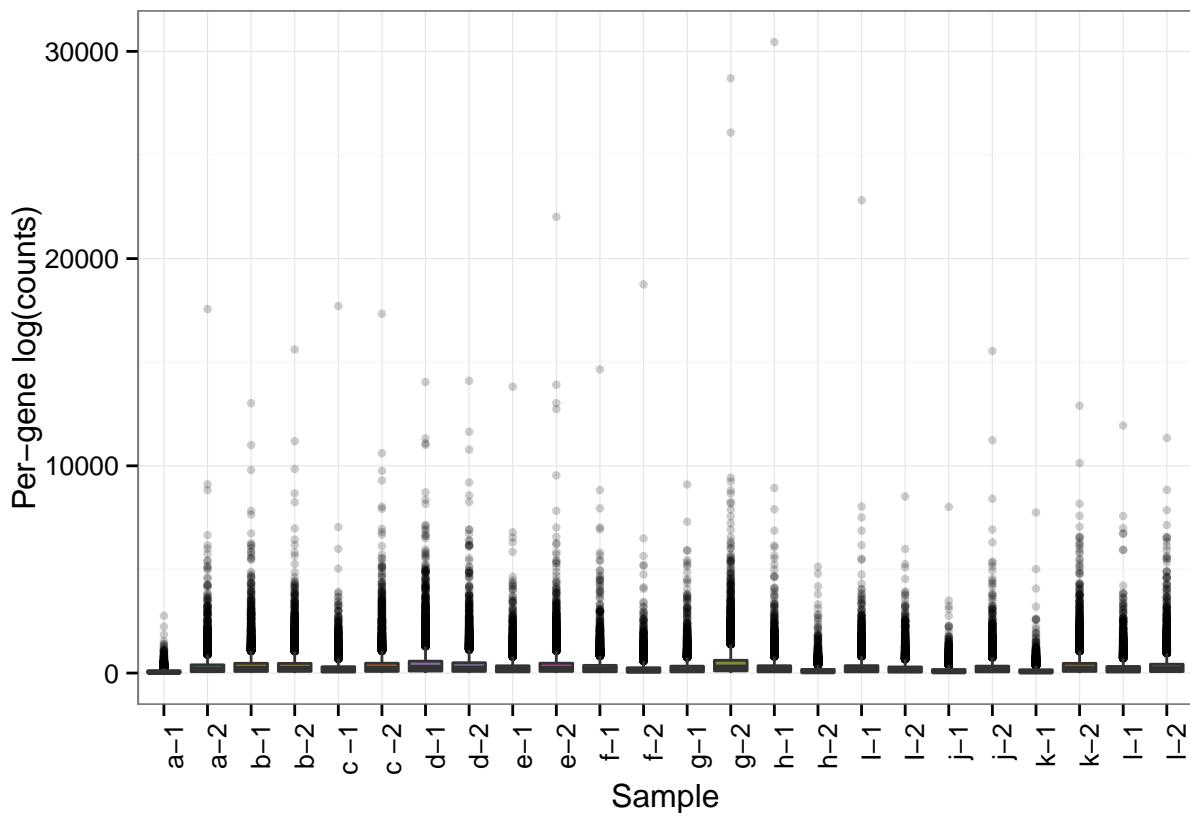
## gene_4_F      29      17      27      16      21      55      6
## gene_5_F     168      49     134     117      92      78     99
## gene_6_F     313     254     590     485     522     847    300
##          HPGL0423 HPGL0424 HPGL0425
## gene_1_F     273     243     192
## gene_2_T     219     310     287
## gene_3_F      15      38      47
## gene_4_F      92      16      70
## gene_5_F     139      63     136
## gene_6_F    1064     241    1083

```

```

## size factor, tmm, rle, upperQuartile all require a design matrix.
norm_boxplot = hpgl_boxplot(expt=norm_expt)
print(norm_boxplot)

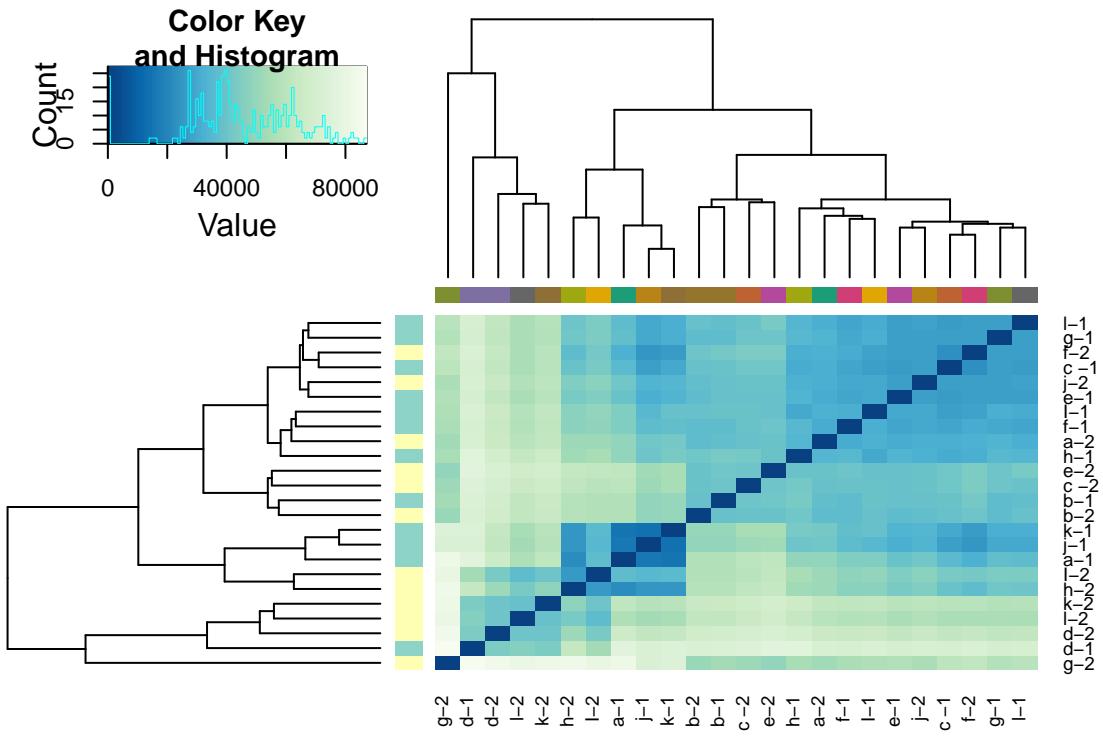
```



```

norm_disheat = hpgl_disheat(expt=norm_expt)

```

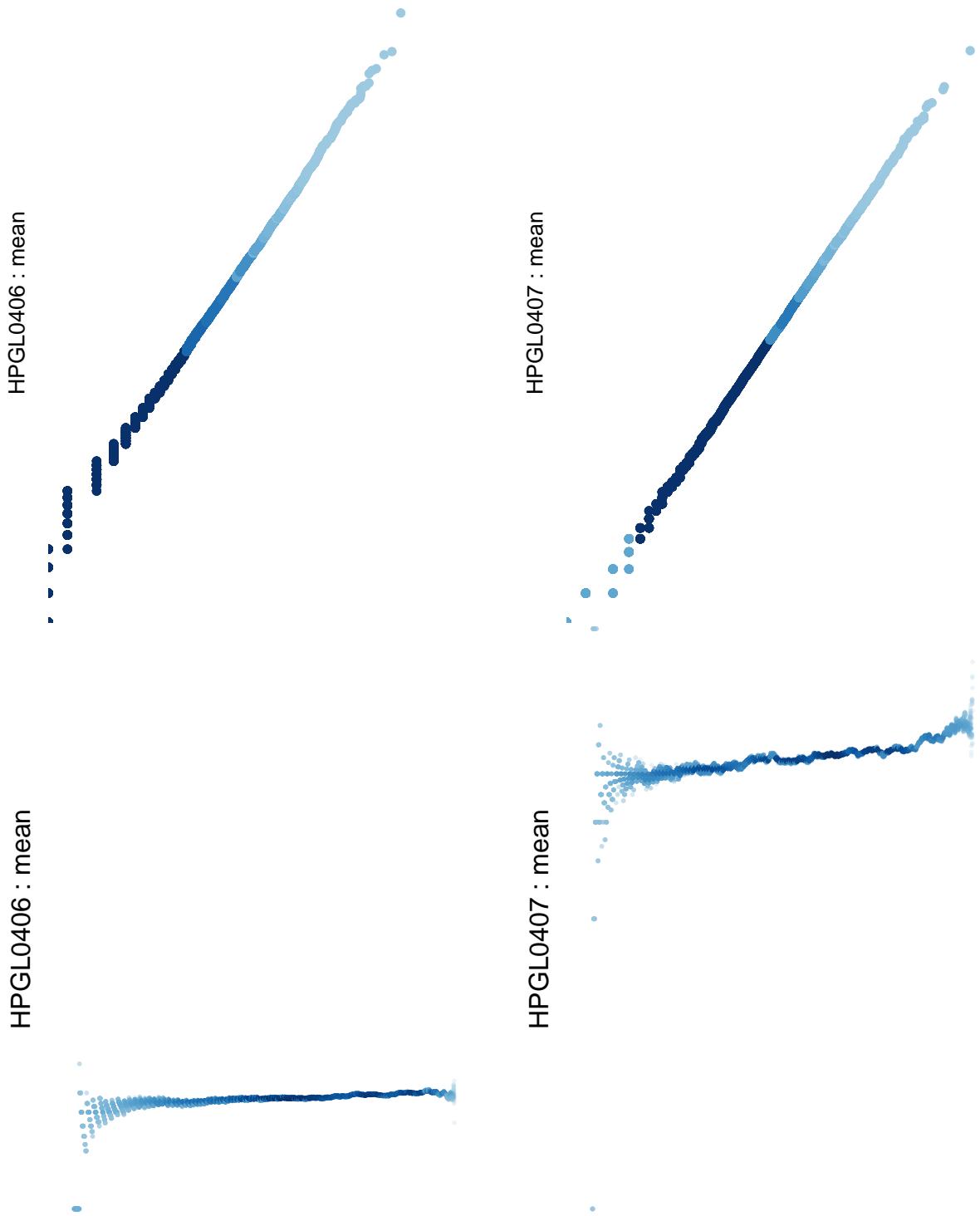


```
print(norm_disheat)
```

### Voom/limma etc

There are a couple ways to call limma using the expt class. In some cases, it might be useful to pull out a subset of the data and only compare the samples of specific conditions/batches/etc.

```
## el_subset means to pull out only those samples which represent 'Early Log' growth.
el_subset = expt_subset(norm_expt, "stage=='EL'")
## Conversely, one may pull samples which are early log and also wild type
elwt_subset = expt_subset(norm_expt, "stage=='EL'&type=='WT'")
## These subsets may be characterized with the plots as above
## Here is a qq plot as an example.
elwt_qqs = hpgl_qq_all(expt=elwt_subset)
```



```
## Simple comparison will take the first condition as control and the second
## as experimental, if we look at el_subset, we will see that means conditions
## 'a' and 'b'. Thus performing simple_comparison will look for differentially
## expressed genes between them.
head(el_subset$design)
```

#	sample	stage	type	condition	batch	color	counts	intercounts
---	--------	-------	------	-----------	-------	-------	--------	-------------

```

## HPGL0406 HPGL0406    EL    WT      a      1 #1B9E77 unknown    unknown
## HPGL0407 HPGL0407    EL    WT      a      2 #1B9E77 unknown    unknown
## HPGL0408 HPGL0408    EL    mga     b      1 #93752C unknown    unknown
## HPGL0409 HPGL0409    EL    mga     b      2 #93752C unknown    unknown

ab_comparison = simple_comparison(el_subset)

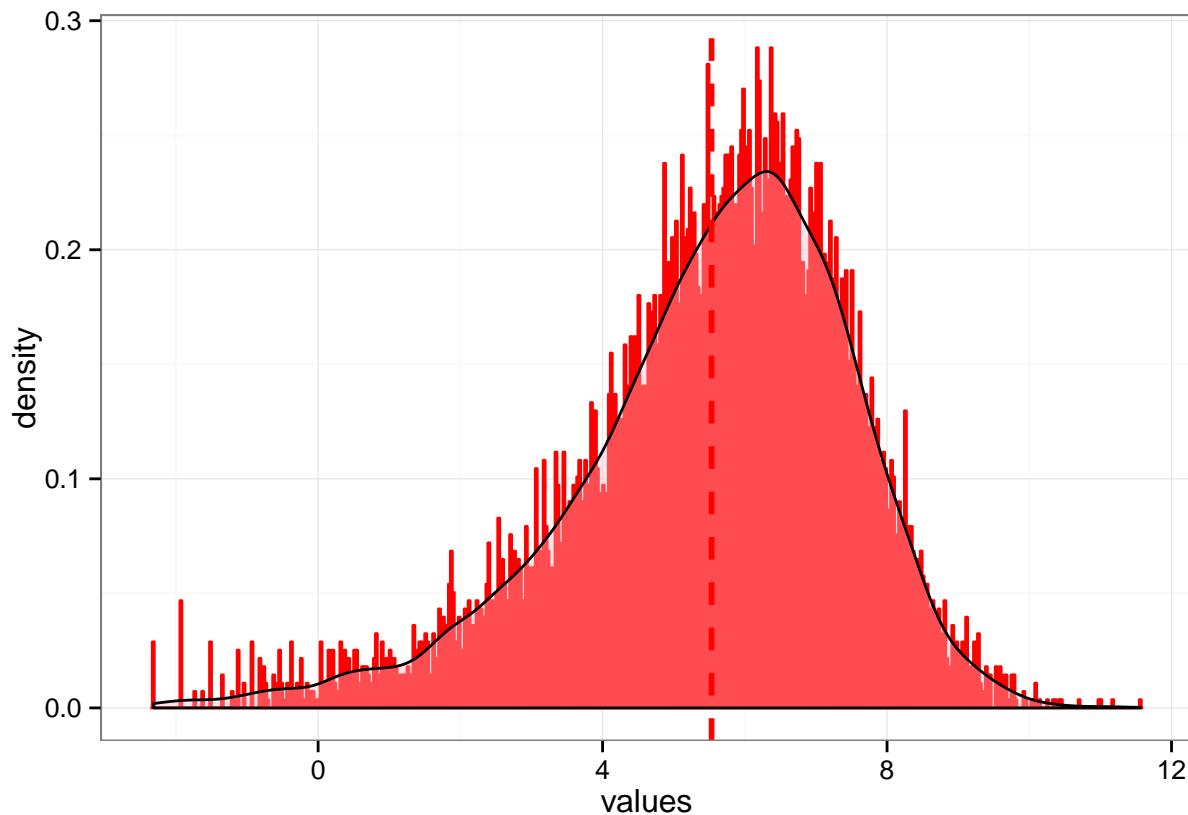
## The voom input was not cpm, converting now.
## The voom input was not log2, transforming now.
## No binwidth nor bins provided, setting it to 0.0299012603878232 in order to have 500 bins.

## A summary of the data will show the data provided:
## The following plots and pieces of data show the output provided by simple_comparison()
## This function isn't really intended to be used, but provides a reference point for performing other a
summary(ab_comparison)

##                                     Length Class      Mode
## amean_histogram                 9   gg      list
## coef_amean_cor                  9  htest     list
## coefficient_scatter              9   gg      list
## coefficient_x                   9   gg      list
## coefficient_y                   9   gg      list
## coefficient_both                 4 -none-    list
## coefficient_lm                  22 lmrob    list
## coefficient_lmsummary            15 summary.lmrob list
## coefficient_weights             10000 -none-   numeric
## comparisons                      10000 MArrayLM list
## contrasts                        10000 MArrayLM list
## contrast_histogram                9   gg      list
## downsignificant                   6  data.frame list
## fit                             30000 MArrayLM list
## ma_plot                          9   gg      list
## psignificant                     6  data.frame list
## pvalue_histogram                 9   gg      list
## table                           6  data.frame list
## upsignificant                    6  data.frame list
## volcano_plot                     9   gg      list
## voom_data                         40000 EList    list
## voom_plot                         9   gg      list

print(ab_comparison$amean_histogram) ## A histogram of the per-gene mean values

```



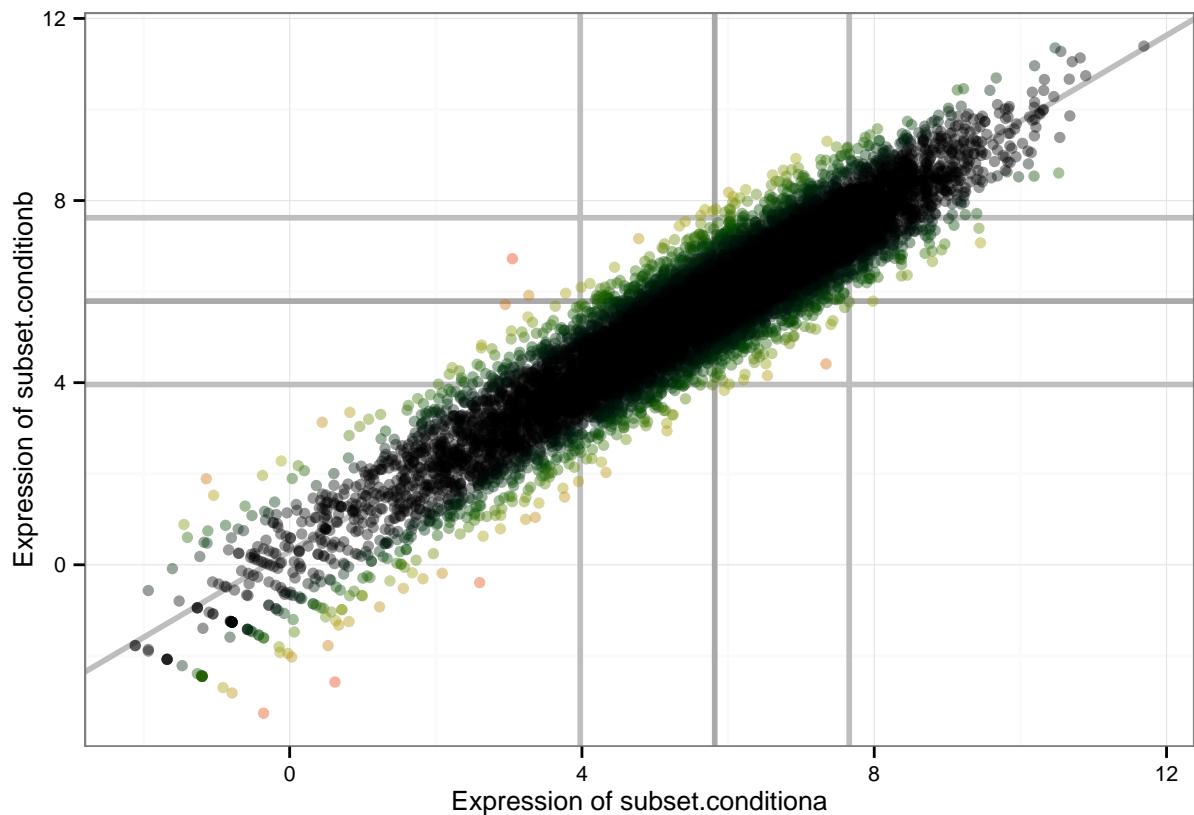
```

print(ab_comparison$coef_amean_cor)    ## The correlation of the means (should not be significant)

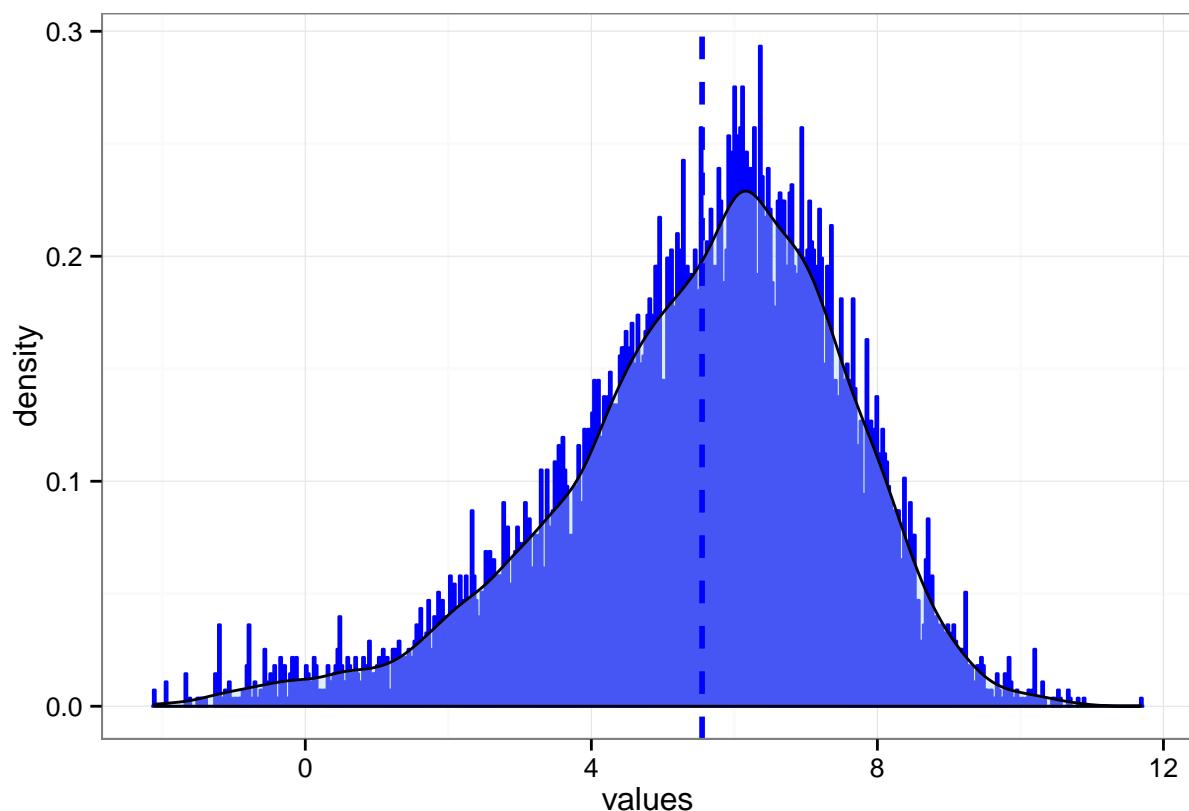
## 
## Pearson's product-moment correlation
## 
## data: cond_contrasts$coefficients and cond_contrasts$Ammean
## t = -4.3772, df = 9998, p-value = 0.00001214
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.06328073 -0.02415554
## sample estimates:
## cor
## -0.0437349

print(ab_comparison$coefficient_scatter) ## A scatter plot of condition b with respect to a

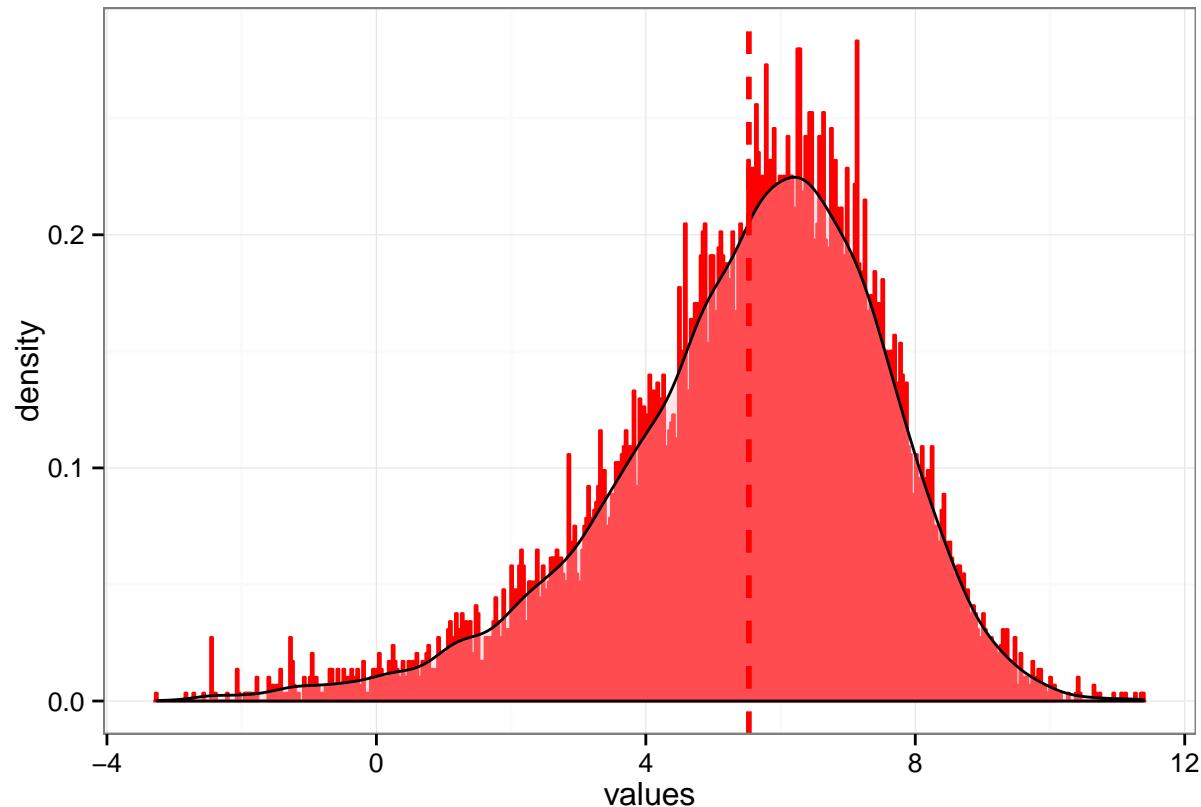
```



```
print(ab_comparison$coefficient_x) ## A histogram of the gene abundances of a
```

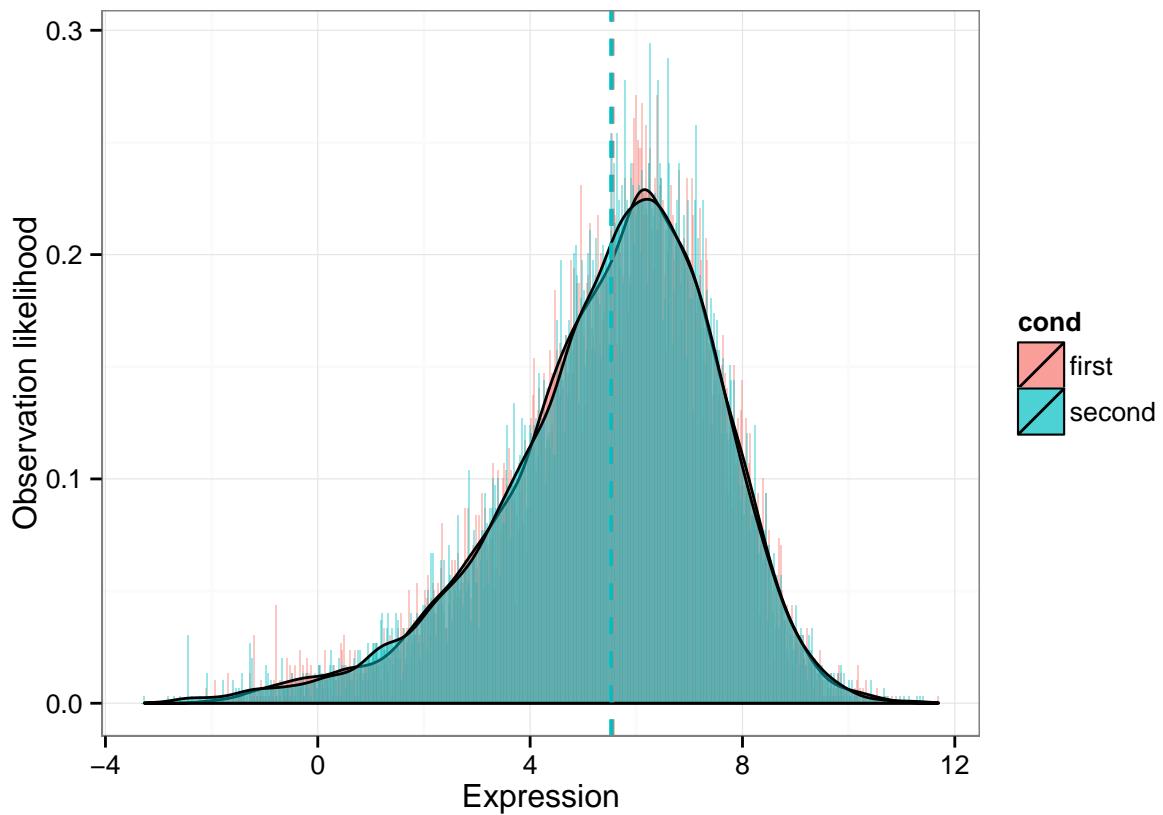


```
print(ab_comparison$coefficient_y) ## A histogram of the gene abundances of b
```



```
print(ab_comparison$coefficient_both) ## A histogram of the gene abundances of a and b
```

```
## $plot
```



```

## 
## $data_summary
##      first          second
##  Min.   :-2.113   Min.   :-3.261
##  1st Qu.: 4.421   1st Qu.: 4.411
##  Median : 5.817   Median : 5.791
##  Mean   : 5.548   Mean   : 5.530
##  3rd Qu.: 6.942   3rd Qu.: 6.925
##  Max.   :11.690   Max.   :11.393
##
## $uncor_t
##
##  Pairwise comparisons using t tests with pooled SD
##
## data: play_all$expression and play_all$cond
##
##      first
## second 0.51
##
## P value adjustment method: none
##
## $bon_t
##
##  Pairwise comparisons using t tests with pooled SD
##
## data: play_all$expression and play_all$cond
##
##      first

```

```

## second 0.51
##
## P value adjustment method: bonferroni

## Note to self, I keep meaning to change the colors of that to match the others
print(ab_comparison$coefficient_lm) ## The description of the line which describes the relationship

##
## Call:
## robustbase::lmrob(formula = second ~ first, data = df, method = "SMDM")
##
## Coefficients:
## (Intercept)      first
##       0.2936      0.9446

## of all of the genes in a to those in b
print(ab_comparison$coefficient_lmsummary) ## A summary of the robust linear model in coefficient_lm

##
## Call:
## robustbase::lmrob(formula = second ~ first, data = df, method = "SMDM")
## \--> method = "SMDM"
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.454612 -0.462071  0.005723  0.466988  3.549615
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 0.293646  0.021566 13.62 <0.000000000000002 ***
## first       0.944617  0.003646 259.09 <0.000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Robust residual standard error: 0.699
## Multiple R-squared:  0.8757, Adjusted R-squared:  0.8757
## Convergence in 7 IRWLS iterations
##
## Robustness weights:
## 6951 weights are ~= 1. The remaining 3049 ones are summarized as
##      Min. 1st Qu. Median Mean 3rd Qu. Max.
## 0.07042 0.81910 0.92800 0.86890 0.97920 0.99900
## Algorithmic parameters:
##      tuning.chi1      tuning.chi2      tuning.chi3      tuning.chi4
## -0.500000000000000  1.500000000000000          NA  0.500000000000000
##           bb      tuning.psi1      tuning.psi2      tuning.psi3
##  0.500000000000000 -0.500000000000000  1.500000000000000  0.950000000000000
##      tuning.psi4      refine.tol      rel.tol      solve.tol
##           NA  0.000000100000000  0.000000100000000  0.000000100000000
##      eps.outlier      eps.x warn.limit.reject warn.limit.meanrw
##  0.000010000000000  0.00000000002126  0.500000000000000  0.500000000000000
##      nResample      max.it      best.r.s      k.fast.s      k.max
##           500          50            2            1          200
##      maxit.scale      trace.lev      mts      compute.rd      numpoints

```

```

##          200      0     1000      0      10
## fast.s.large.n
##          2000
##           psi      subsampling      cov
##           "lqq"    "nonsingular" ".vcov.w"
## compute.outlier.stats
##           "SMDM"
## seed : int(0)

## This has some neat things like the R-squared value and the parameters used to arrive at the linear model
## ab_comparision$coefficient_weights ## a list of weights by gene, bigger weights mean closer to the linear model
## ab_comparision$comparisons ## the raw output from limma
print(ab_comparision$contrasts) ## The output from limma's makeContrasts()

## An object of class "MArrayLM"
## $coefficients
##   gene_1_F   gene_2_T   gene_3_F   gene_4_F   gene_5_F
## -2.06830281  0.73688637  0.05179155  0.10814455 -0.25655242
## 9995 more rows ...
##
## $stdev.unscaled
##   gene_1_F   gene_2_T   gene_3_F   gene_4_F   gene_5_F
## 0.4937486  0.4386966  0.5666225  0.5255633  0.4906638
## 9995 more rows ...
##
## $sigma
## [1] 1.1499226 2.9749817 0.7076561 2.2577474 1.5163376
## 9995 more elements ...
##
## $df.residual
## [1] 1 1 1 1 1
## 9995 more elements ...
##
## $cov.coefficients
##           Contrasts
## Contrasts      changed_v_control
##   changed_v_control      1
##
## $Amean
##   gene_1_F   gene_2_T   gene_3_F   gene_4_F   gene_5_F
## 5.719066 7.669256 3.226942 3.653567 4.963363
## 9995 more elements ...
##
## $method
## [1] "ls"
##
## $design
##   changed control subset_batch2
## 1       1      0      0
## 2       1      0      1
## 3       0      1      0
## 4       0      1      1
## attr(,"assign")
## [1] 1 1 2

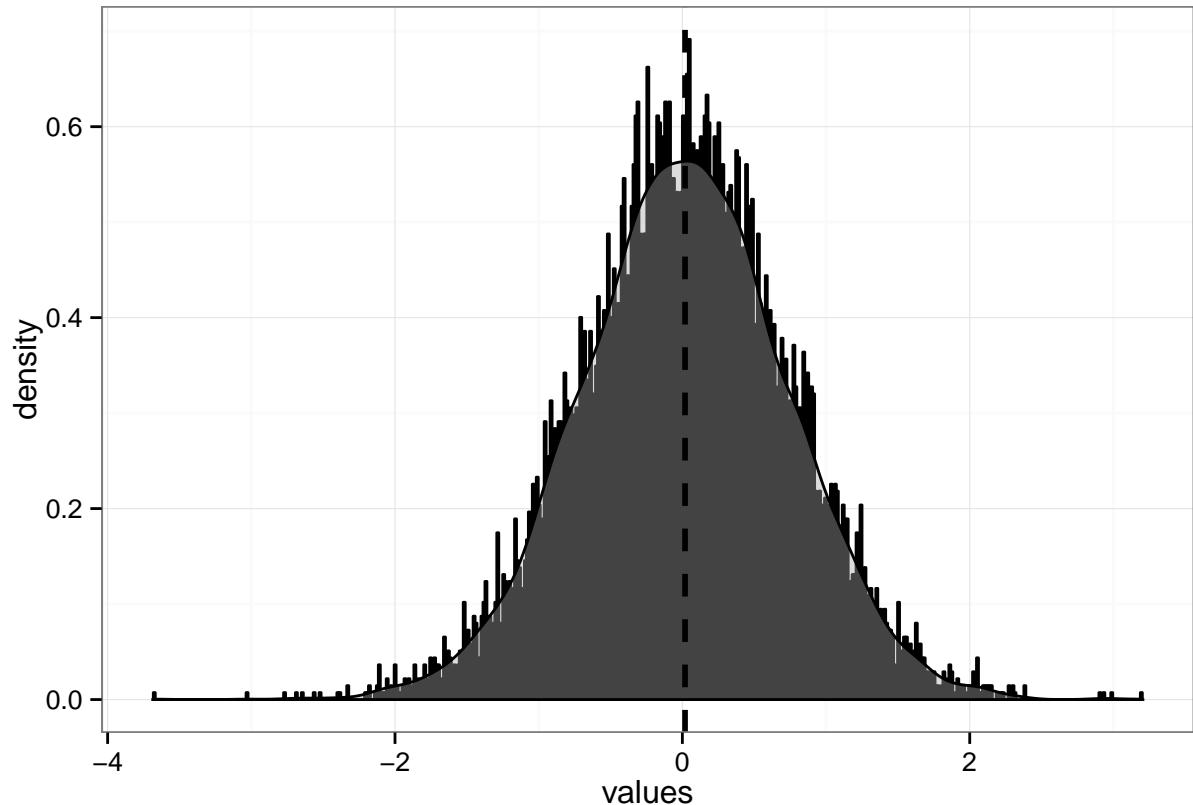
```

```

## attr(,"contrasts")
## attr(,"contrasts")$`subset$condition`
## [1] "contr.treatment"
##
## attr(,"contrasts")$`subset$batch`
## [1] "contr.treatment"
##
##
## $contrasts
##          Contrasts
## Levels      changed_v_control
##   changed                  1
##   control                 -1
## subset_batch2                0

print(ab_comparison$contrast_histogram) ## A histogram of the values of b-a for each gene

```



```

head(ab_comparison$downsignificant) ## The list of genes which are significantly down in b vs a

##           logFC    AveExpr         t     P.Value adj.P.Val
## gene_9453_F -3.674407 4.6687613 -4.751151 0.00000205079 0.0205079
## gene_6962_F -3.030009 0.5821939 -2.894170 0.00380984540 0.8466323
## gene_5901_F -2.766474 4.4491470 -3.583991 0.00033999470 0.8440245
## gene_2815_F -2.684763 2.1287330 -2.908773 0.00363649503 0.8440245
## gene_2096_F -2.642624 4.8614813 -3.465371 0.00053172354 0.8440245
## gene_2865_F -2.565979 0.3463398 -2.441604 0.01463940433 0.9979641
##           B

```

```

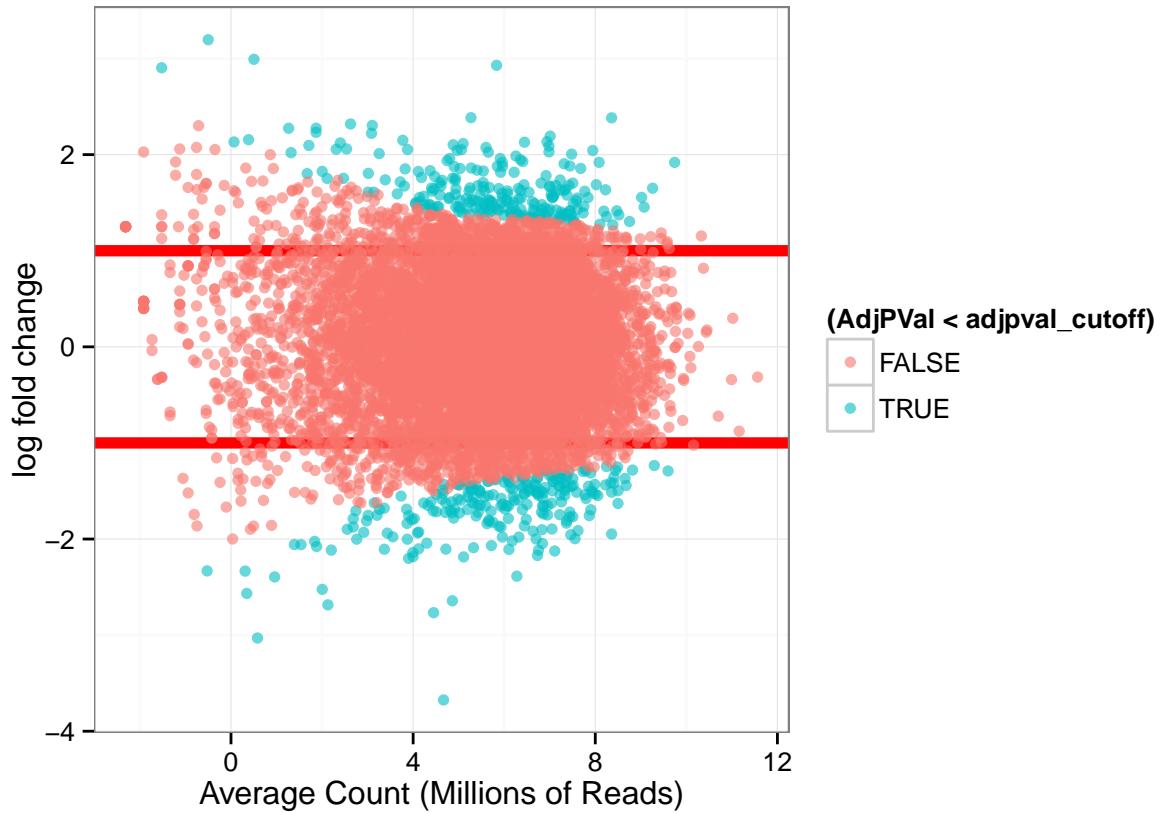
## gene_9453_F -1.244491
## gene_6962_F -3.870834
## gene_5901_F -2.764891
## gene_2815_F -3.707017
## gene_2096_F -2.868598
## gene_2865_F -4.114582

dim(ab_comparison$downsignificant)

## [1] 1903      6

## ab_comparison$fit ## the result from lmFit()
print(ab_comparison$ma_plot)  ## An ma plot of b vs a

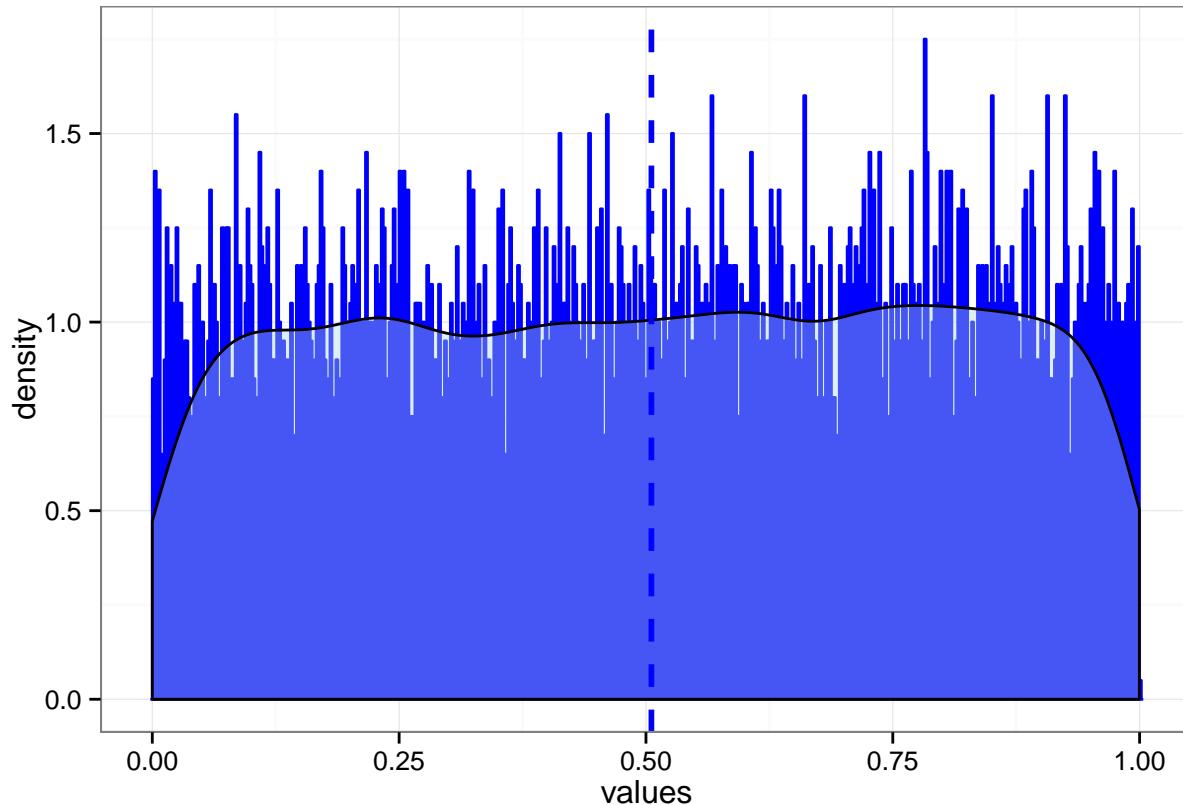
```



```

print(ab_comparison$pvalue_histogram) ## A histogram of the p-values, one would hope to see a spike in ...

```



```
head(ab_comparison$table) ## The full contrast table
```

```
##          logFC      AveExpr       t     P.Value adj.P.Val
## gene_9453_F -3.674407  4.6687613 -4.751151 0.00000205079 0.02050790
## gene_1147_F  3.195330 -0.5000518  2.662580 0.00776681755 0.88924321
## gene_6962_F -3.030009  0.5821939 -2.894170 0.00380984540 0.84663231
## gene_7746_F  2.991793  0.5039044  2.940911 0.00327998028 0.84402453
## gene_5460_F  2.929328  5.8334237  4.295525 0.00001759273 0.08796365
## gene_745_T   2.903883 -1.5219175  2.258808 0.02391675544 0.99796408
##          B
## gene_9453_F -1.244491
## gene_1147_F -4.116820
## gene_6962_F -3.870834
## gene_7746_F -3.809143
## gene_5460_F -1.412287
## gene_745_T  -4.309751
```

```
head(ab_comparison$upsignificant) ## The list of genes which are significantly up in b vs a
```

```
##          logFC      AveExpr       t     P.Value adj.P.Val
## gene_1147_F 3.195330 -0.5000518  2.662580 0.00776681755 0.88924321
## gene_7746_F  2.991793  0.5039044  2.940911 0.00327998028 0.84402453
## gene_5460_F  2.929328  5.8334237  4.295525 0.00001759273 0.08796365
## gene_745_T   2.903883 -1.5219175  2.258808 0.02391675544 0.99796408
## gene_6055_F  2.384475  5.2700388  3.446193 0.00057088066 0.84402453
## gene_5451_T  2.383115  8.3595141  3.708323 0.00020976507 0.69921690
##          B
```

```

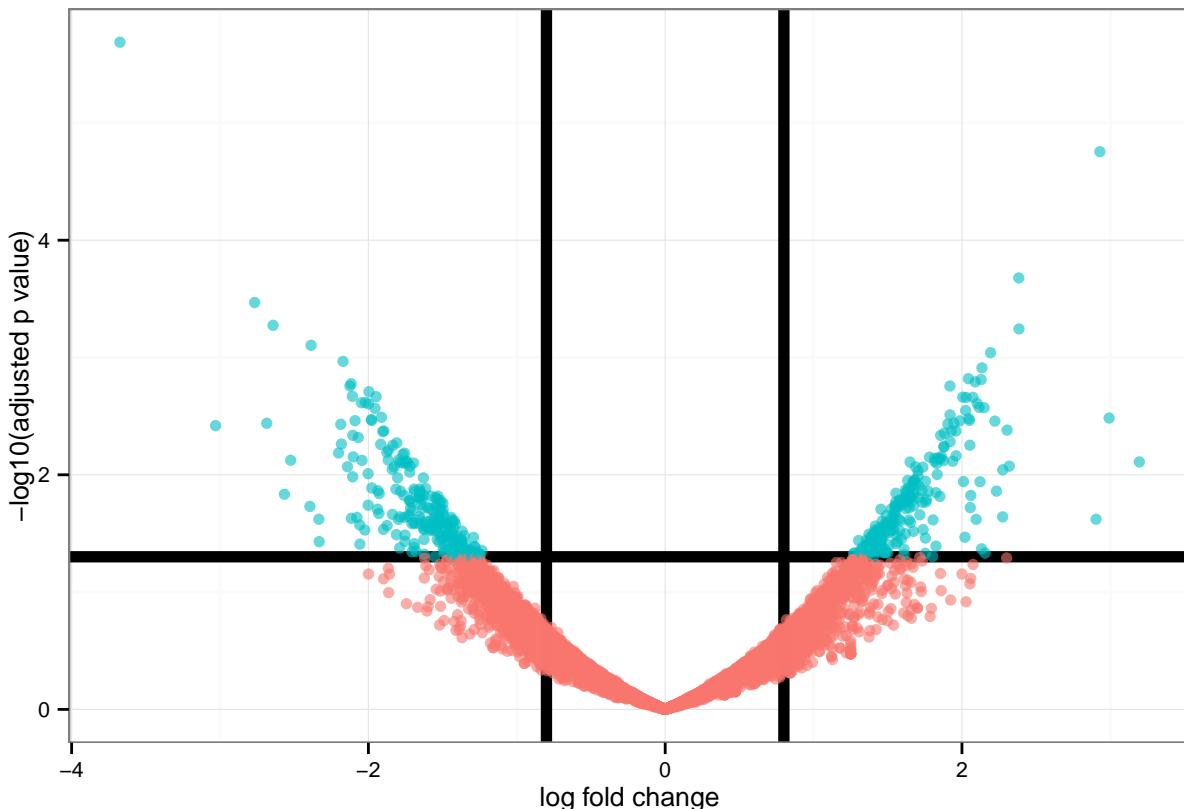
## gene_1147_F -4.116820
## gene_7746_F -3.809143
## gene_5460_F -1.412287
## gene_745_T -4.309751
## gene_6055_F -2.663534
## gene_5451_T -2.111578

dim(ab_comparison$upsignificant)

## [1] 2029      6

print(ab_comparison$volcano_plot) ## A Volcano plot of b vs a

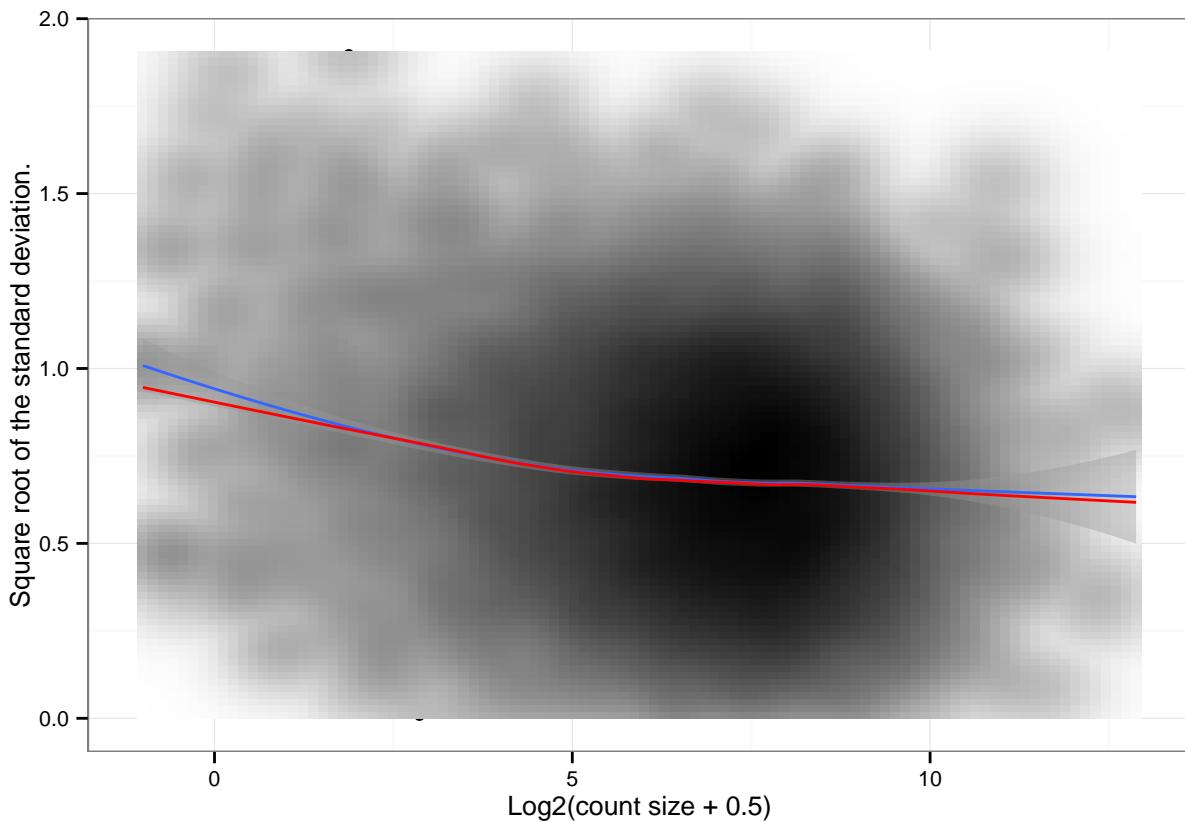
```



```

## ab_comparison$voom_data ## The output from voom()
print(ab_comparison$voom_plot) ## A ggplot2 version of the mean/variance trend provided by voom()

```



```

## The data structure ab_comparison$comparisons contains the output from eBayes() which comprises the 1
## limma step...
funkytown = write_limma(data=ab_comparison$comparisons, excel=FALSE, csv=FALSE)

## 1: Printing table: changed_v_control

## Lets make up some gene lengths
gene_lengths = funktown[[1]]
gene_lengths$width = sample(nrow(gene_lengths))
gene_lengths>ID = rownames(gene_lengths)
gene_lengths = gene_lengths[,c("ID","width")]

## And some GO categories
goids=funkytown[[1]]
all_go_categories = AnnotationDbi::keys(GO.db)
goids$GO = sample(all_go_categories, nrow(gene_lengths))
goids>ID = rownames(goids)
goids = goids[,c("ID","GO")]

ontology_fun = limma_ontology(funkytown, gene_lengths=gene_lengths, goids=goids, n=100, overwrite=TRUE)

## This function expects a list of limma contrast tables and some annotation information.
## The annotation information would be gene lengths and ontology ids
## Creating directory: excelfor writing excel/csv data.
## Performing ontology search of:changed_v_control
## simple_goseq() makes some pretty hard assumptions about the data it is fed:

```

```

## It requires 2 tables, one of GOids which must have columns (gene)ID and GO(category)
## The other table is of gene lengths with columns (gene)ID and (gene)width.
## Other columns are fine, but ignored.
## Using the length data to fill in the de vector.

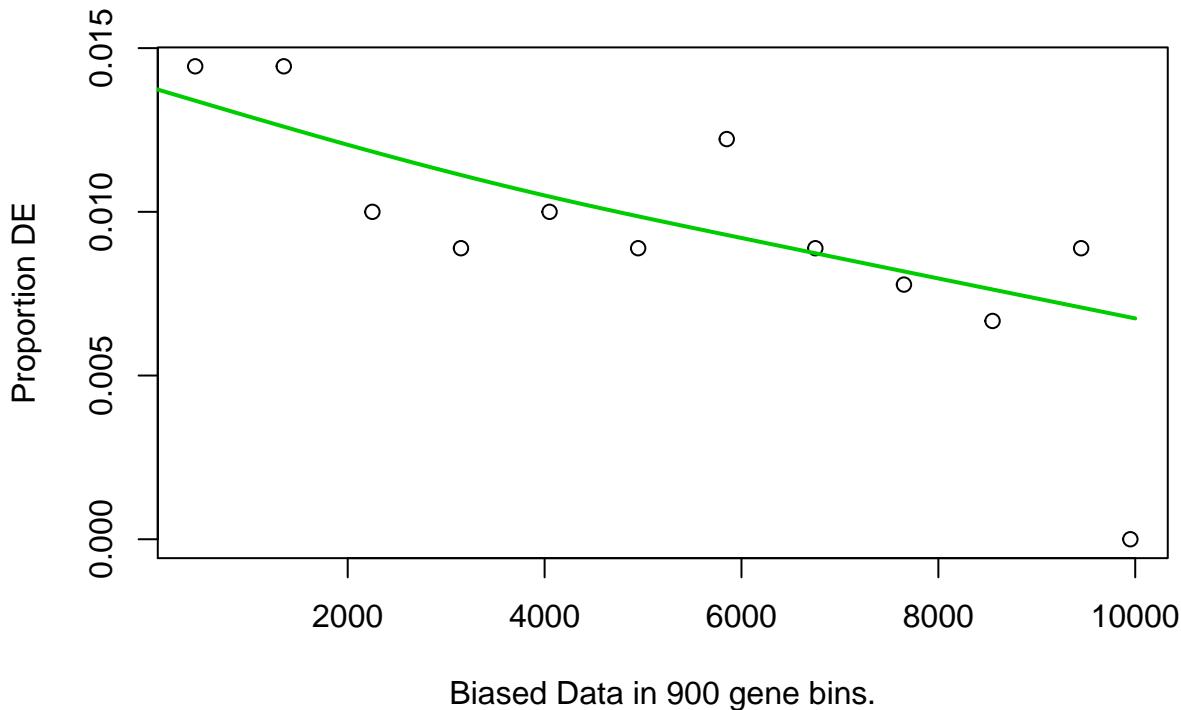
## [1] "Sometimes nullp throws a warning and some stuff about site-packages, I haven't figured out why,"

## Using manually entered categories.
## Calculating the p-values...

## Warning in goseq:::reversemapping(gene2cat): NAs introduced by coercion

## Error in rep(names(map), times = as.numeric(summary(map)[, 1])): invalid 'times' argument

```



```

testme = head(funktown[[1]], n=40)
tt = simple_clusterprofiler(testme, goids=goids, gff=goids)

## Using GO mapping data located in GO2EG.rda
## Starting MF(molecular function) analysis

## The minimum observed adjusted pvalue is: 0.194053

## Warning in if (class(all_mf_phist) != "try-error") {: the condition has
## length > 1 and only the first element will be used

## The minimum observed adjusted pvalue is: 0.582160

## Starting BP(biological process) analysis

```

```

## The minimum observed adjusted pvalue is: 0.015370

## Warning in if (class(all_bp_phist) != "try-error") {: the condition has
## length > 1 and only the first element will be used

## The minimum observed adjusted pvalue is: 0.220300

## Starting CC(cellular component) analysis

## The minimum observed adjusted pvalue is: 0.218434
## The minimum observed adjusted pvalue is: 0.397286

## Warning in if (class(all_cc_phist) != "try-error") {: the condition has
## length > 1 and only the first element will be used

## Attempting to include the cnetplots from clusterProfiler.
## They fail often, if this is causing errors, set:
## include_cnetplots to FALSE
## cnetplot just failed for the MF ontology. Do not be concerned with the previous error.
## cnetplot just failed for the BP ontology. Do not be concerned with the previous error.
## cnetplot just failed for the CC ontology. Do not be concerned with the previous error.

ttt = cluster_trees(testme, tt)

## Error in make_id2gomap(goid_map = goid_map, goids_df = goids_df, overwrite = overwrite): There is ne

tttt = simple_topgo(testme)

## Error in make_id2gomap(goid_map = goid_map, goids_df = goids_df, overwrite = overwrite): There is ne

```

### A cell-means model using all conditions and batches

```

## acb stands for "kept_conditions_batches" which takes too long to
## type when setting up the contrasts.
acb = paste0(kept_qcpml2$conditions, kept_qcpml2$batches)
kept_data = exprs(kept_qcpml2$expressionset)
table(acb)
## The invocation of table() keptows me to count up the contribution of
## each condition/batch combination to the whole data set.

## Doing this (as I understand it) means I do nothave to worry about
## balanced samples so much, but must be more careful to understand
## the relative contribution of each sample type to the entire data
## set.

complete_model = model.matrix(~0 + acb)
complete_fit = lmFit(kept_data, complete_model)
complete_voom = hpgl_voom(kept_data, complete_model)
complete_voom$plot

```

```

complete_model
## This is an example of what happens when I have heterogenous numbers of samples
## on each side of a contrast, so that a normal design matrix of conditions + batches
## would not work, so instead I add up the contributions of each batch (capital letters)
## and average them out, then use the resulting terms in the various contrasts below.
epi_cl14 = "acbcl14_epiF"
epi_clbr = "acbclbr_epiE"
tryp_cl14 = "(acbcl14_trypB + acbcl14_trypD + acbcl14_trypG) / 3"
tryp_clbr = "acbclbr_trypG"
a60_cl14 = "(acbcl14_a60A * 2/3) + (acbcl14_a60B * 1/3)"
a60_clbr = "acbclbr_a60A"
a96_cl14 = "acbcl14_a96C"
a96_clbr = "acbclbr_a96C"
epi_cl14clbr = paste0("(", epi_cl14, ") ", " - ", "(", epi_clbr, ")")
tryp_cl14clbr = paste0("(", tryp_cl14, ") ", " - ", "(", tryp_clbr, ")")
a60_cl14clbr = paste0("(", a60_cl14, ") ", " - ", "(", a60_clbr, ")")
a96_cl14clbr = paste0("(", a96_cl14, ") ", " - ", "(", a96_clbr, ")")
epitryp_cl14 = paste0("(", tryp_cl14, ") ", " - ", "(", epi_cl14, ")")
epitryp_clbr = paste0("(", tryp_clbr, ") ", " - ", "(", epi_clbr, ")")
epia60_cl14 = paste0("(", a60_cl14, ") ", " - ", "(", epi_cl14, ")")
epia60_clbr = paste0("(", a60_clbr, ") ", " - ", "(", epi_clbr, ")")
a60a96_cl14 = paste0("(", a96_cl14, ") ", " - ", "(", a60_cl14, ")")
a60a96_clbr = paste0("(", a96_clbr, ") ", " - ", "(", a60_clbr, ")")
a60tryp_cl14 = paste0("(", tryp_cl14, ") ", " - ", "(", a60_cl14, ")")
a60tryp_clbr = paste0("(", tryp_clbr, ") ", " - ", "(", a60_clbr, ")")
## The following contrast is messed up in some as of yet unknown way.
epitryp_cl14clbr = paste0("(", epitryp_cl14, ") ", " - ", "(", epitryp_clbr, ")")
## So I will add some more contrasts using data which doesn't get screwed up
epia60_cl14clbr = paste0("(", epia60_cl14, ") ", " - ", "(", epia60_clbr, ")")
a60tryp_cl14clbr = paste0("(", a60tryp_cl14, ") ", " - ", "(", a60tryp_clbr, ")")
a60a96_cl14clbr = paste0("(", a60a96_cl14, ") ", " - ", "(", a60a96_clbr, ")")

complete_contrasts_v2 = makeContrasts(
    epi_cl14=epi_cl14,
    epi_clbr=epi_clbr,
    tryp_cl14=tryp_cl14,
    tryp_clbr=tryp_clbr,
    a60_cl14=a60_cl14,
    a60_clbr=a60_clbr,
    a96_cl14=a96_cl14,
    a96_clbr=a96_clbr,
    epi_cl14clbr=epi_cl14clbr,
    tryp_cl14clbr=tryp_cl14clbr,
    a60_cl14clbr=a60_cl14clbr,
    a96_cl14clbr=a96_cl14clbr,
    epitryp_cl14=epitryp_cl14,
    epitryp_clbr=epitryp_clbr,
    epia60_cl14=epia60_cl14,
    epia60_clbr=epia60_clbr,
    a60a96_cl14=a60a96_cl14,
    a60a96_clbr=a60a96_clbr,
    a60tryp_cl14=a60tryp_cl14,
    a60tryp_clbr=a60tryp_clbr,

```

```

epitryp_cl14clbr=epitryp_cl14clbr,
epia60_cl14clbr=epia60_cl14clbr,
a60tryp_cl14clbr=a60tryp_cl14clbr,
a60a96_cl14clbr=a60a96_cl14clbr,
levels=complete_voom$design)
## This colnames() is annoyingly necessary to avoid really obnoxious contrast names.
colnames(complete_contrasts_v2) = c("epi_cl14","epi_clbr","tryp_cl14","tryp_clbr","a60_cl14","a60_clbr")
kept_fits = contrasts.fit(complete_fit, complete_contrasts_v2)
kept_comparisons = eBayes(kept_fits)

```

### Clean conditions, batches

On the other hand, I would like to perform arbitrary comparisons among my data even when the batches and conditions look good, so I set up my model/contrast matrices a little strangely even then:

```

all_data = exprs(norm_expt$expressionset)
complete_model = model.matrix(~0 + all_human_expt$conditions + all_human_expt$batches)
## Shorten the column names of the model so I don't have to type so much later...
tmpnames = colnames(complete_model)
tmpnames = gsub("all_human_expt[[:punct:]]","", tmpnames)
tmpnames = gsub("conditions","", tmpnames)
colnames(complete_model) = tmpnames
rm(tmpnames)

complete_voom = hpgl_voom(all_data, complete_model)
complete_voom$plot
complete_fit = lmFit(complete_voom, complete_model)

all_contrasts = makeContrasts(
    ## Start with the simple coefficient groupings for each condition
    none4=none4,
    none24=none24,
    none48=none48,
    none72=none72,
    bead4=bead4,
    bead24=bead24,
    bead48=bead48,
    bead72=bead72,
    maj4=maj4,
    maj24=maj24,
    maj48=maj48,
    maj72=maj72,
    ama4=ama4,
    ama24=ama24,
    ama48=ama48,
    ama72=ama72,
    ## Now do a few simple comparisons
    ## compare beads to uninfected
    beadnone_4=bead4-none4,
    beadnone_24=bead24-none24,
    beadnone_48=bead48-none48,
    beadnone_72=bead72-none72,
    majnone_4=maj4-none4,

```

```

majnone_24=maj24-none24,
majnone_48=maj48-none48,
majnone_72=maj72-none72,
amanone_4=ama4-none4,
amanone_24=ama24-none24,
amanone_48=ama48-none48,
amanone_72=ama72-none72,
## compare samples to beads
majbead_4=maj4-bead4,
majbead_24=maj24-bead24,
majbead_48=maj48-bead48,
majbead_72=maj72-bead72,
amabead_4=ama4-bead4,
amabead_24=ama24-bead24,
amabead_48=ama48-bead48,
amabead_72=ama72-bead72,
## (x-z)-(a-b)
## Use this to compare major and amazonensis
amamaj_bead_4=(ama4-bead4)-(maj4-bead4),
amamaj_bead_24=(ama24-bead24)-(maj24-bead24),
amamaj_bead_48=(ama48-bead48)-(maj48-bead48),
amamaj_bead_72=(ama72-bead72)-(maj72-bead72),
## (c-d)-(e-f) where c/d are: (amazon|major/none)/(beads/none)
majbead_none_4=(maj4-none4)-(bead4-none4),
majbead_none_24=(maj24-none24)-(bead24-none24),
majbead_none_48=(maj48-none48)-(bead48-none48),
majbead_none_72=(maj72-none72)-(bead72-none72),
amabead_none_4=(ama4-none4)-(bead4-none4),
amabead_none_24=(ama24-none24)-(bead24-none24),
amabead_none_48=(ama48-none48)-(bead48-none48),
amabead_none_72=(ama72-none72)-(bead72-none72),
levels=complete_voom$design)
all_fits = contrasts.fit(complete_fit, all_contrasts)
all_comparisons = eBayes(all_fits)
limma_list = write_limma(data=all_comparisons)

all_table = topTable(all_comparisons, adjust="fdr", n=nrow(all_data))
write.csv(all_comparisons, file="excel/all_tables.csv")
## write_limma() is a shortcut for writing out all the data structures
all_comparison_tables = write_limma(all_comparisons, excel=FALSE)

```

## Ontology searches

The following is an example of a simplified GO search given 20 groups of genes which are from an unannotated organism, but for which blast2GO was performed.

```

ontology_info = read.csv(file="data/trinotate_go_trimmed.csv.gz", header=FALSE, sep="\t")

## Warning in file(file, "rt"): cannot open file
## 'data/trinotate_go_trimmed.csv.gz': No such file or directory

## Error in file(file, "rt"): cannot open the connection

```

```

colnames(ontology_info) = c("gene_id", "transcript_id", "group", "startend", "blast_go", "pfam_go")

## Error in colnames(ontology_info) = c("gene_id", "transcript_id", "group", : object 'ontology_info' no

## Drop any entries which don't have a putative length
ontology_info = subset(ontology_info, startend != 0)

## Error in subset(ontology_info, startend != 0): error in evaluating the argument 'x' in selecting a m

## Split the column 'startend' into two columns by the '-' sign
ontology_info = as.data.frame(transform(ontology_info, startend=colsplit(startend, split="\\"-", names=c

## Error in as.data.frame(transform(ontology_info, startend = colsplit(startend, : error in evaluating :
##   error in evaluating the argument '_data' in selecting a method for function 'transform': Error: ob

## Make the resulting pieces into two separate columns, start and end.
ontology_info$start = ontology_info$startend$start

## Error in eval(expr, envir, enclos): object 'ontology_info' not found

ontology_info$end = ontology_info$startend$end

## Error in eval(expr, envir, enclos): object 'ontology_info' not found

## Use start and end to make length
ontology_info$length = abs(ontology_info$start - ontology_info$end)

## Error in eval(expr, envir, enclos): object 'ontology_info' not found

## Drop the unneeded columns
ontology_info = ontology_info[,c("gene_id", "transcript_id", "group", "start", "end", "length", "blast_go", "p

## Error in eval(expr, envir, enclos): object 'ontology_info' not found

head(ontology_info)

## Error in head(ontology_info): error in evaluating the argument 'x' in selecting a method for function

## goseq() requires mappings between ID/length and ID/GO category
## Currently I have my toy set to assume column names, which is admittedly stupid.
gene_lengths = ontology_info[,c("transcript_id", "length")]

## Error in eval(expr, envir, enclos): object 'ontology_info' not found

```

```

colnames(gene_lengths) = c("ID","width")
split_go = ontology_info[,c("transcript_id","blast_go")]

## Error in eval(expr, envir, enclos): object 'ontology_info' not found

split_go$blast_go = as.character(split_go$blast_go)

## Error in eval(expr, envir, enclos): object 'split_go' not found

## The following few lines were pulled from the internet
## they serve to generate a data structure in the format expected by goseq()
## It simply splits all space separated GO categories into separate rows
## with the same ID
require.auto("splitstackshape")
id_go = concat.split.multiple(split_go, "blast_go", seps=" ", "long")

## This function is deprecated. Use `cSplit` instead.

## Error in is.data.table(indt): object 'split_go' not found

id_go = as.data.frame(id_go)

## Error in as.data.frame(id_go): error in evaluating the argument 'x' in selecting a method for function

colnames(id_go) = c("ID","GO")

## Error in colnames(id_go) = c("ID", "GO"): object 'id_go' not found

go_ids = subset(id_go, GO != 0)

## Error in subset(id_go, GO != 0): error in evaluating the argument 'x' in selecting a method for function

## Pull out all entries from group 1
group_one = subset(ontology_info, group == "1")

## Error in subset(ontology_info, group == "1"): error in evaluating the argument 'x' in selecting a method for function

group_one = group_one[,c("transcript_id","start","end")]

## Error in eval(expr, envir, enclos): object 'group_one' not found

colnames(group_one) = c("ID","start","end")

## Error in colnames(group_one) = c("ID", "start", "end"): object 'group_one' not found

```

```

## Perform the goseq() analysis
group_one_go = simple_goseq(group_one, lengths=gene_lengths, goids=go_ids)

## simple_goseq() makes some pretty hard assumptions about the data it is fed:
## It requires 2 tables, one of GOids which must have columns (gene)ID and GO(category)
## The other table is of gene lengths with columns (gene)ID and (gene)width.
## Other columns are fine, but ignored.

## Error in simple_goseq(group_one, lengths = gene_lengths, goids = go_ids): object 'group_one' not found

group_one_go$pvalue_histogram

## Error in eval(expr, envir, enclos): object 'group_one_go' not found

head(group_one_go$godata_interesting)

## Error in head(group_one_go$godata_interesting): error in evaluating the argument 'x' in selecting a method for

head(group_one_go$mf_subset)

## Error in head(group_one_go$mf_subset): error in evaluating the argument 'x' in selecting a method for

group_one_go$mfp_plot

## Error in eval(expr, envir, enclos): object 'group_one_go' not found

group_one_go$bpp_plot

## Error in eval(expr, envir, enclos): object 'group_one_go' not found

group_one_go$ccp_plot

## Error in eval(expr, envir, enclos): object 'group_one_go' not found

## Print trees of the goseq() data
initial_trees = goseq_trees(group_one, group_one_go, goids_df=go_ids)

## Error in make_id2gomap(goid_map = goid_map, goids_df = goids_df, overwrite = overwrite): object 'go

initial_trees$MF

## Error in eval(expr, envir, enclos): object 'initial_trees' not found

initial_trees$BP

## Error in eval(expr, envir, enclos): object 'initial_trees' not found

```

```
initial_trees$CC

## Error in eval(expr, envir, enclos): object 'initial_trees' not found
```

## Vignette Info

Note the various macros within the `vignette` section of the metadata block above. These are required in order to instruct R how to build the vignette. Note that you should change the `title` field and the `\VignetteIndexEntry` to match the title of your vignette.

## Styles

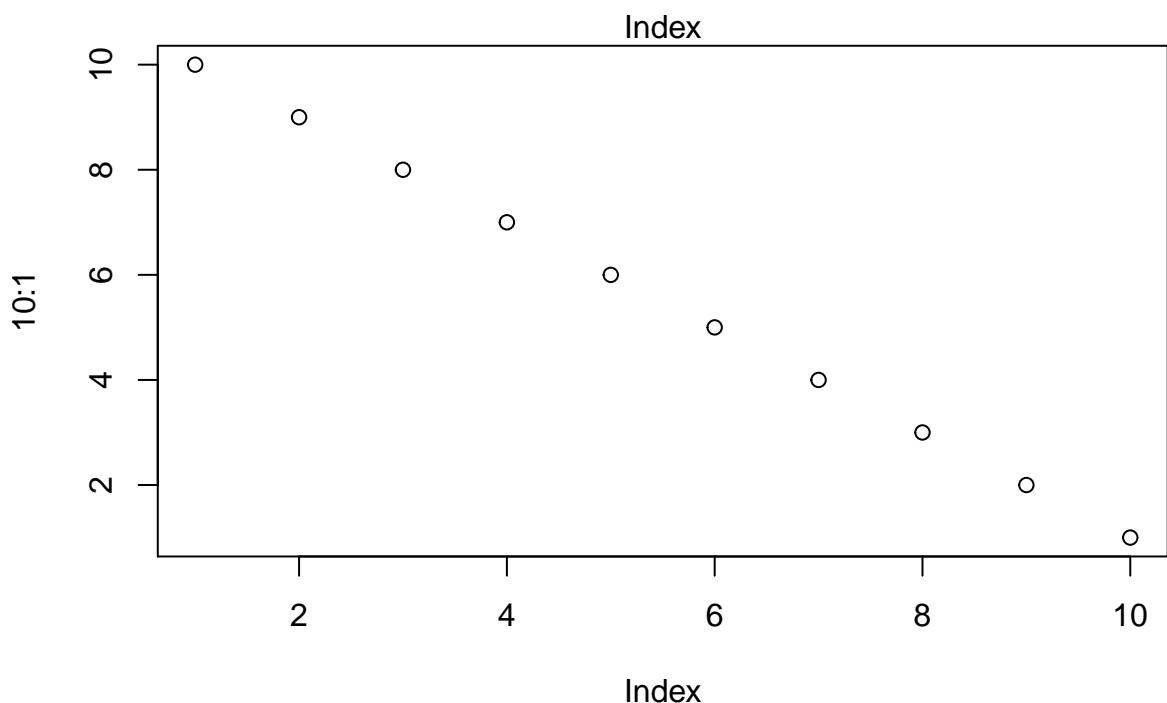
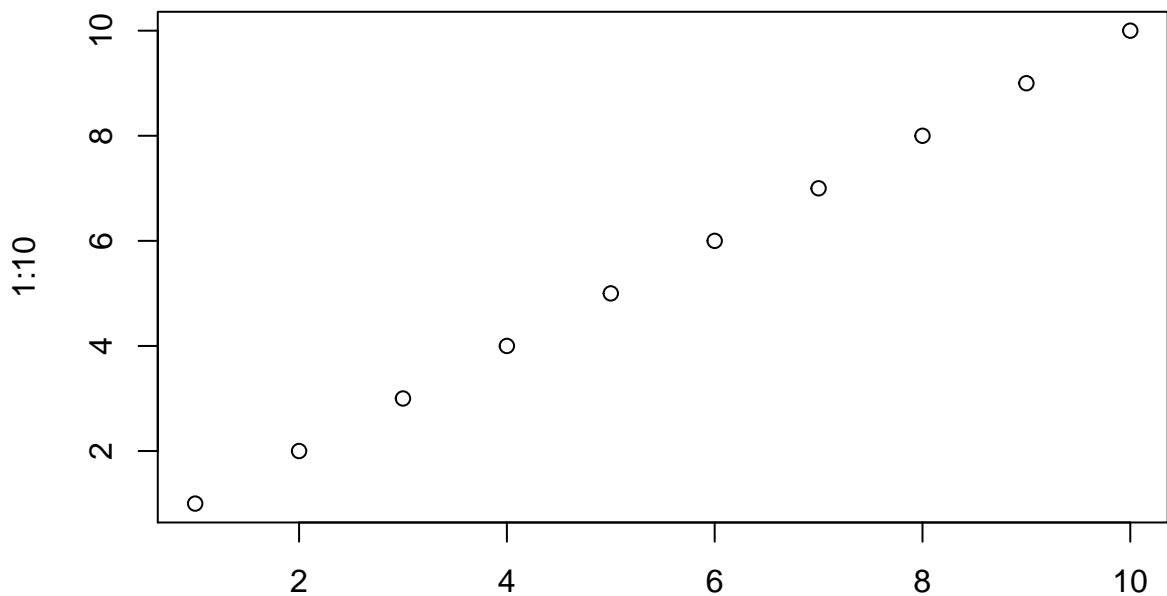
The `html_vignette` template includes a basic CSS theme. To override this theme you can specify your own CSS in the document metadata as follows:

```
output:
  rmarkdown::html_vignette:
    css: mystyles.css
```

## Figures

The figure sizes have been customised so that you can easily put two images side-by-side.

```
plot(1:10)
plot(10:1)
```



You can enable figure captions by `fig_caption: yes` in YAML:

```
output:  
  rmarkdown::html_vignette:  
    fig_caption: yes
```

Then you can use the chunk option `fig.cap = "Your figure caption."` in knitr.

## More Examples

You can write math expressions, e.g.  $Y = X\beta + \epsilon$ , footnotes<sup>1</sup>, and tables, e.g. using `knitr::kable()`.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

Also a quote using >:

“He who gives up [code] safety for [code] speed deserves neither.” ([via](#))

---

<sup>1</sup>A footnote here.