Using hpgltools for fun and profit

Ashton Trey Belew abelew@gmail.com

2015-01-27

Hpgltools: Stupid R tricks.

The following block shows how I handle autloading 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()
```

```
## [1] "Loading devtools"
## [1] "Loading data.table"
## [1] "Loading gtools"
## [1] "Loading hash"
## [1] "Loading knitcitations"
## [1] "Loading knitr"
## [1] "Loading knitrBootstrap"
## [1] "Loading methods"
## [1] "Loading plyr"
## [1] "Loading reshape"
## [1] "Loading rjson"
## [1] "Loading rmarkdown"
## [1] "Loading roxygen2"
## [1] "Loading testthat"
## [1] "Loading XLConnect"
## [1] "Loading xtable"
## [1] "Loading clusterProfiler"
## [1] "Loading GO.db"
## [1] "Loading DOSE"
## [1] "Loading goseq"
## [1] "Loading KEGGREST"
## [1] "Loading pathview"
## [1] "Loading RamiGO"
## [1] "Loading topGO"
## [1] "Loading biomaRt"
## [1] "Loading BSgenome"
## [1] "Loading BSgenome.Lmajor.friedlin"
## [1] "Loading genomeIntervals"
## [1] "Loading rtracklayer"
## [1] "Loading cbcbSEQ"
## [1] "Loading DESeq2"
## [1] "Loading DESeq"
## [1] "Loading edgeR"
## [1] "Loading sva"
## [1] "Loading Cairo"
## [1] "Loading directlabels"
## [1] "Loading ggplot2"
## [1] "Loading googleVis"
## [1] "Loading gplots"
## [1] "Loading gridExtra"
## [1] "Loading RColorBrewer"
## [1] "Loading Rgraphviz"
## [1] "Loading multtest"
## [1] "Loading qvalue"
## [1] "Loading robust"
## [1] "Loading motifRG"
## [1] "Loading Rsamtools"
## [1] "Loading scales"
## [1] "Loading seqinr"
```

```r
opts_knit$set(progress=TRUE, verbose=TRUE, stop_on_error=FALSE, error=TRUE, fig.width=7, fig.height=7)
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.

```r
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 evaludate 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.

```r
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.

```r
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
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_exampledata(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."
```

```
## Warning in brewer.pal(num_colors, "Dark2"): n too large, allowed maximum for palette Dark2 is 8
## Returning the palette you asked for with that many colors
```

```
## [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 resul
## plots are also not interesting...
fun = graph_metrics(expt=all_expt)
```

```
## Error: 'hpgl_norm' is not an exported object from 'namespace:myr'
```

```
fun
```

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

```
## 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 defaults to using the original expressionset for normalization."
```

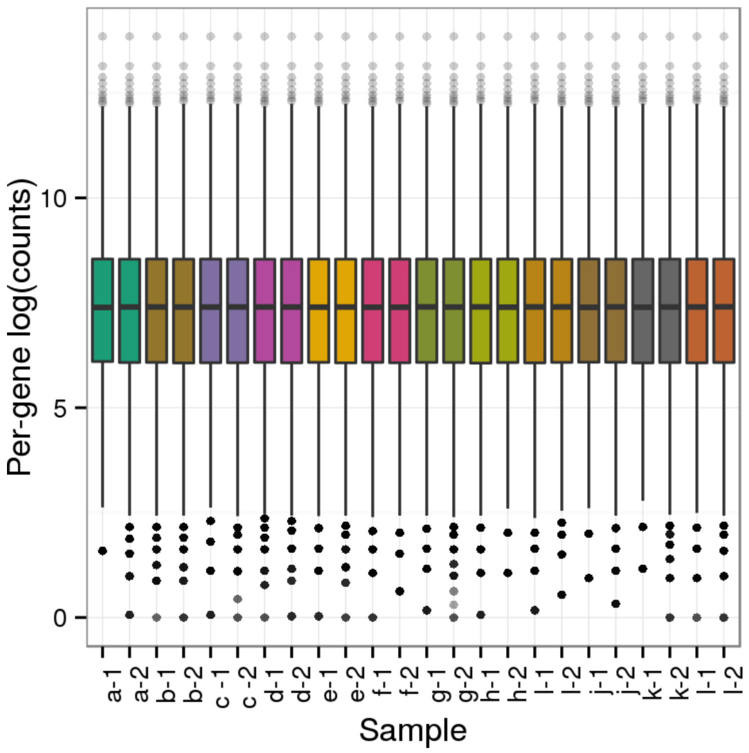
```
head(exprs(norm_expt$expressionset))
```

```
##           HPGL0406  HPGL0407  HPGL0408 HPGL0409 HPGL0149 HPGL0150 HPGL0147
## gene_1_F  5.277675  7.136137  7.984537 8.745534 8.222795 6.158610 7.002112
## gene_2_T  8.882579 10.240344  9.477674 8.290211 6.854609 8.886459 7.682117
## gene_3_F  4.333901  5.402302  3.894818 5.613483 5.571121 4.909893 4.996238
## gene_4_F  4.579944  5.842350  5.815383 4.698126 5.171594 3.662965 4.634206
## gene_5_F  5.630570  7.080373  6.719389 6.703904 7.530081 6.121534 7.421890
## gene_6_F 10.772143  9.766805 10.522132 8.964822 9.490308 9.277675 9.431367
##           HPGL0148 HPGL0410 HPGL0411  HPGL0412  HPGL0413  HPGL0414
## gene_1_F  7.349834 6.422766 7.955892  7.903882  6.677132  7.662075
## gene_2_T  7.823633 8.765010 9.096825  9.443980  9.001877 10.013904
## gene_3_F  5.321928 5.637832 4.809500  6.203756  5.370687  5.241586
## gene_4_F  4.853829 5.014950 4.357552  4.579944  3.954196  4.641450
## gene_5_F  6.048487 7.634206 7.442598  7.005157  8.040290  6.893302
## gene_6_F 10.088070 9.692107 9.934213 10.129337 10.482724  9.604476
##          HPGL0416 HPGL0415 HPGL0417 HPGL0418 HPGL0419  HPGL0420 HPGL0421
## gene_1_F 8.421364 8.599215 7.591211 7.252271 7.667407  8.006560 6.399456
## gene_2_T 9.660033 8.061371 7.989986 9.630950 7.773139  9.268152 7.742590
## gene_3_F 5.685333 5.562242 5.353147 5.367779 5.127564  5.606097 3.122397
## gene_4_F 4.377934 4.952256 5.191471 4.849666 4.412217  5.524215 5.931230
## gene_5_F 6.352411 7.454642 6.717676 7.140830 7.208234  7.609025 6.399456
## gene_6_F 8.941048 8.323336 9.055508 9.260233 9.232621 10.137925 9.850708
##          HPGL0422 HPGL0423 HPGL0424 HPGL0425
## gene_1_F 7.097593 7.684895 8.136350 7.349466
## gene_2_T 8.747564 7.378656 8.468963 7.948489
## gene_3_F 6.815917 3.646259 5.474382 5.366322
## gene_4_F 4.033423 6.162391 4.288482 5.929752
## gene_5_F 7.870878 6.767633 6.207014 6.863670
## gene_6_F 9.463780 9.681897 8.124660 9.840286
```

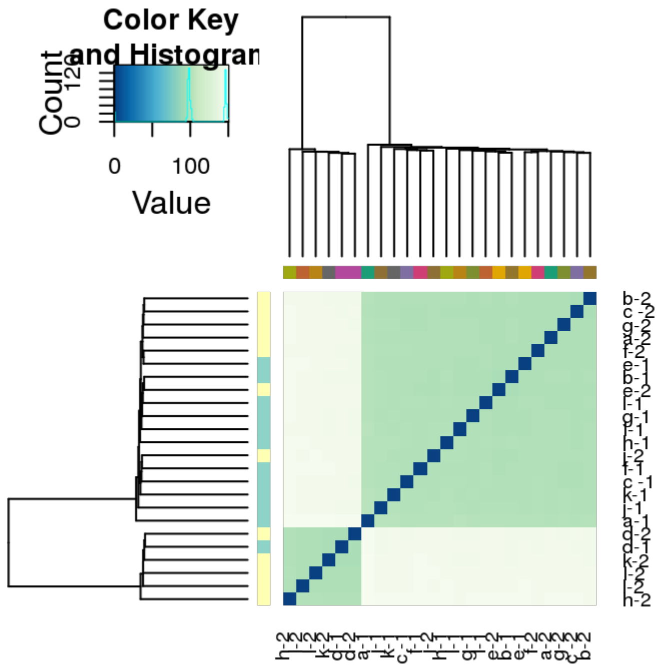
```
## size factor, tmm, rle, upperQuartile all require a design matrix.
norm_boxplot = hpgl_boxplot(expt=norm_expt)
```

```
## Using id as id variables
```

```
print(norm_boxplot)
```



```r
norm_disheat = hpgl_disheat(expt=norm_expt)
```



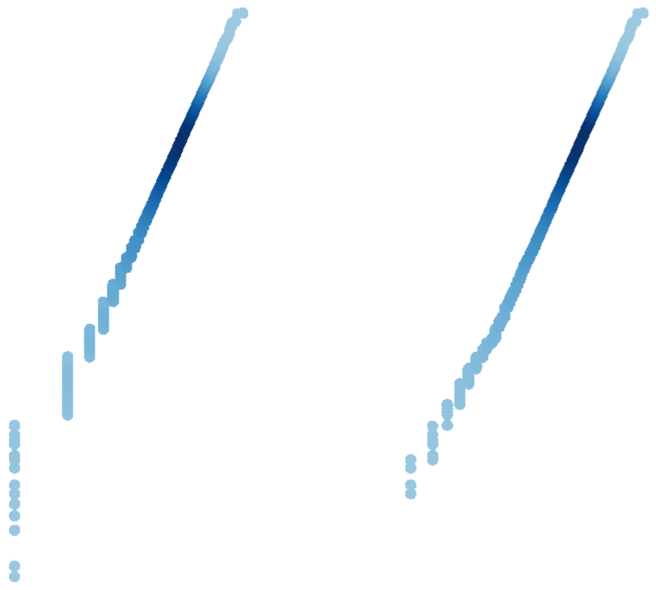
```r
print(norm_disheat)
```

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)
```

```
## [1] "Loading grid"
```



```
## [1] "Loading grid"
```

```
## 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
## HPGL0406 HPGL0406    EL   WT         a     1 #1B9E77
## HPGL0407 HPGL0407    EL   WT         a     2 #1B9E77
## HPGL0408 HPGL0408    EL  mga         b     1 #93752C
## HPGL0409 HPGL0409    EL  mga         b     2 #93752C
##                                                      counts
## HPGL0406  processed_data/count_tables/wt/el/HPGL0406.count.gz
## HPGL0407  processed_data/count_tables/wt/el/HPGL0407.count.gz
## HPGL0408 processed_data/count_tables/mga/el/HPGL0408.count.gz
## HPGL0409 processed_data/count_tables/mga/el/HPGL0409.count.gz
##                                               intercounts
## HPGL0406  data/count_tables/wt/el/HPGL0406_inter.count.gz
## HPGL0407  data/count_tables/wt/el/HPGL0407_inter.count.gz
## HPGL0408 data/count_tables/mga/el/HPGL0408_inter.count.gz
## HPGL0409 data/count_tables/mga/el/HPGL0409_inter.count.gz
```

```
ab_comparison = simple_comparison(el_subset)
```

```
## Error: 'hpgl_voom' is not an exported object from 'namespace:myr'
```

```
## 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)
```

```
## Error in summary(ab_comparison): error in evaluating the argument 'object' in selecting a method for
```

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

```
## Error in print(ab_comparison$amean_histogram): error in evaluating the argument 'x' in selecting a me
```

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

```
## Error in print(ab_comparison$coef_amean_cor): error in evaluating the argument 'x' in selecting a met
```

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

```
## Error in print(ab_comparison$coefficient_scatter): error in evaluating the argument 'x' in selecting
```

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

```
## Error in print(ab_comparison$coefficient_x): error in evaluating the argument 'x' in selecting a meth
```

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

```
## Error in print(ab_comparison$coefficient_y): error in evaluating the argument 'x' in selecting a meth
```

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

```
## Error in print(ab_comparison$coefficient_both): error in evaluating the argument 'x' in selecting a m
```

```r
## 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
```

```
## Error in print(ab_comparison$coefficient_lm): error in evaluating the argument 'x' in selecting a met
```

```r
## 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
```

```
## Error in print(ab_comparison$coefficient_lmsummary): error in evaluating the argument 'x' in selecti
```

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

```
## Error in print(ab_comparison$contrasts): error in evaluating the argument 'x' in selecting a method f
```

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

```
## Error in print(ab_comparison$contrast_histogram): error in evaluating the argument 'x' in selecting a
```

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

```
## Error in head(ab_comparison$downsignificant): error in evaluating the argument 'x' in selecting a me
```

```
dim(ab_comparison$downsignificant)
```

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

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

```
## Error in print(ab_comparison$ma_plot): error in evaluating the argument 'x' in selecting a method fo
```

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

```
## Error in print(ab_comparison$pvalue_histogram): error in evaluating the argument 'x' in selecting a
```

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

```
## Error in head(ab_comparison$table): error in evaluating the argument 'x' in selecting a method for f
```

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

```
## Error in head(ab_comparison$upsignificant): error in evaluating the argument 'x' in selecting a meth
```

```
dim(ab_comparison$upsignificant)
```

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

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

```
## Error in print(ab_comparison$volcano_plot): error in evaluating the argument 'x' in selecting a meth
```

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

```
## Error in print(ab_comparison$voom_plot): error in evaluating the argument 'x' in selecting a method
```

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

```
## Error in write_limma(data = ab_comparison$comparisons, excel = FALSE, : object 'ab_comparison' not f
```

```r
## Lets make up some gene lengths
gene_lengths = funkytown[[1]]
```

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

```r
gene_lengths$width = sample(nrow(gene_lengths))
```

```
## Error in sample(nrow(gene_lengths)): error in evaluating the argument 'x' in selecting a method for f
##   error in evaluating the argument 'x' in selecting a method for function 'nrow': Error: object 'gene
```

```r
gene_lengths$ID = rownames(gene_lengths)
```

```
## Error in rownames(gene_lengths): error in evaluating the argument 'x' in selecting a method for funct
```

```r
gene_lengths = gene_lengths[,c("ID","width")]
```

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

```r
## And some GO categories
goids=funkytown[[1]]
```

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

```r
all_go_categories = AnnotationDbi::keys(GO.db)
goids$GO = sample(all_go_categories, nrow(gene_lengths))
```

```
## Error in nrow(gene_lengths): error in evaluating the argument 'x' in selecting a method for function
```

```r
goids$ID = rownames(goids)
```

```
## Error in rownames(goids): error in evaluating the argument 'x' in selecting a method for function 'ro
```

```r
goids = goids[,c("ID","GO")]
```

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

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

```
## [1] "This function expects a list of limma contrast tables and some annotation information."
## [1] "The annotation information would be gene lengths and ontology ids"
```

```
## Error in limma_ontology(funkytown, gene_lengths = gene_lengths, goids = goids, : object 'funkytown' n
```

```r
testme = head(funkytown[[1]], n=40)
```

```
## Error in head(funkytown[[1]], n = 40): error in evaluating the argument 'x' in selecting a method for
```

```
tt = simple_clusterprofiler(testme, goids=goids, gff=goids)
```

```
## Warning in readChar(con, 5L, useBytes = TRUE): cannot open compressed file
## 'geneTable.rda', probable reason 'No such file or directory'
```

```
## Error in simple_clusterprofiler(testme, goids = goids, gff = goids): object 'goids' not found
```

```
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:

```r
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 me
```

```
## 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 t
##   object 'ontology_info' not found
```

```
## 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 functio
```

```
## 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")
```

```
## Error in colnames(gene_lengths) = c("ID", "width"): object 'gene_lengths' not found
```

```
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")
```

```
## [1] "Loading 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 functi
```

```
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 func
```

```
## 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 me
```

```
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)
```

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

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

```r
group_one_go$pvalue_histogram
```

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

```r
head(group_one_go$godata_interesting)
```

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

```r
head(group_one_go$mf_subset)
```

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

```r
group_one_go$mfp_plot
```

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

```r
group_one_go$bpp_plot
```

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

```r
group_one_go$ccp_plot
```

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

```r
## 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_i
```

```r
initial_trees$MF
```

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

```r
initial_trees$BP
```

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

```r
initial_trees$CC
```

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

Vignette Info

Note the various macros within the `vignette` setion 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

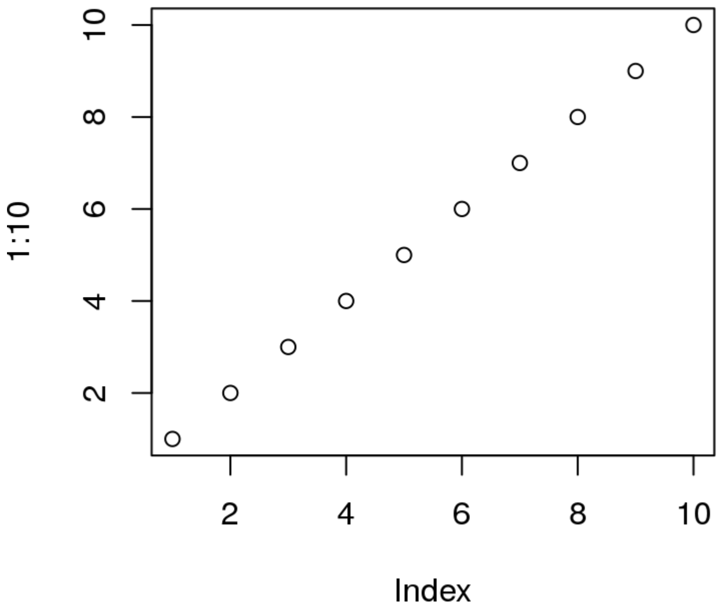
The `html_vignette` template includes a basic CSS theme. To override this theme you can specify your own CSS in the document metadata as follows:

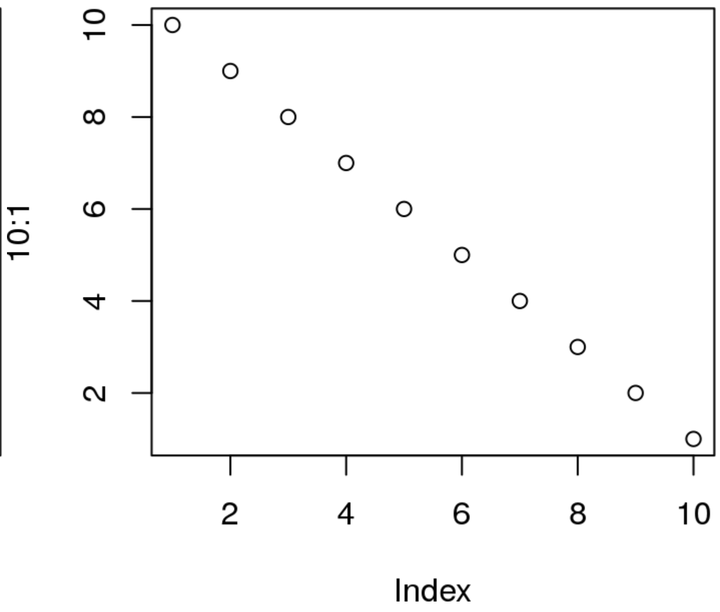
```
output:
  rmarkdown::html_vignette:
    css: mystyles.css
```

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[1], 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

[1] A footnote here.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
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)