Package 'myr'

December 23, 2014

Type Package

Title A pile of (hopefully) useful R functions

Version 0.1

Date 2014-05-23

Author Ashton Trey Belew

Maintainer Ashton Trey Belew <abelew@gmail.com>

Description This is a set of functions I have been using in my various
analyses in the El-Sayed laboratory. They are intended to
be useful for anyone, but primarily attempt to make some
graphs easier to create, some data normalizations easier,and as reminders about what to (not) do.

License GPL

Suggests
biomaRt,BSgenome,BSgenome.Lmajor.friedlin,Cairo,cbcbSEQ,clusterProfiler,data.table,DESeq2,DESeq,devtools,directla

VignetteBuilder knitr

R topics documented:

`create_experiment` *Wrap bioconductor's expressionset to include some other extraneous information.*

Description

Wrap bioconductor's expressionset to include some other extraneous information.

Usage

```
create_experiment(file, color_hash)
```

Arguments

`file`	a comma separated file describing the samples with information like condition,batch,count_filename,etc
`color_hash`	a hash which describes how to color the samples

Value

experiment an expressionset

See Also

`pData`, `fData`, `exprs`, `my_read_files`, `as.list.hash`

Examples

```
## new_experiment = create_experiment("some_csv_file.csv", color_hash)
```

dirty_go — *Perform a quick and dirty gene ontology enrichment search*

Description

Perform a quick and dirty gene ontology enrichment search

Usage

```
dirty_go(de_genes, lengths = NULL, goids = NULL, adjust = 0.05)
```

Arguments

de_genes	a dataframe of gene IDs which have been deemed 'differential' by whatever metric you choose.
lengths	a dataframe of gene id and lengths, null by default.
goids	a dataframe of gene to goid mappings
adjust	minimum p value for (bh by default) adjustment

Value

a list including all the goseq data, subsets deemed interesting by the pvalue, and associated goterms.

See Also

nullp, goseq,

Examples

```
## dirty_godata = dirty_go(de_genes, lengths=length_df, goids=genetogo_df)
```

expt_subset — *Extract a subset of samples following some rule(s) from an experiment class*

Description

Extract a subset of samples following some rule(s) from an experiment class

Usage

```
expt_subset(expt, subset)
```

Arguments

expt	an expt which is a home-grown class containing an expressionSet, design, colors, etc.
subset	a valid R expression which defines a subset of the design to keep.

Value

metadata an expt class which contains the smaller set of data

See Also

ExpressionSet, pData, exprs, and fData

Examples

```
## smaller_expt = expt_subset(big_expt, "condition=='control'")
## all_expt = expt_subset(expressionset, "")  ## extracts everything
```

---

extract_go *Make a table of gene ontology information*

Description

Make a table of gene ontology information

Usage

```
extract_go(df, file = NULL)
```

Arguments

df	a dataframe of ontology information. This is intended to be the output from goseq including information like numbers/category, GOids, etc.
file	a csv file to which to write the table

Value

the ontology table with annotation information included

See Also

GOTERM, GO.db,

Examples

```
## annotated_go = extract_go(go_ids)
```

`filter_counts` *Filter low-count genes from a data set.*

Description

Filter low-count genes from a data set.

Usage

```
filter_counts(counts, lib.size = NULL, thresh = 4, minSamples = 2)
```

Arguments

`df`	input data frame of counts by sample
`lib.size`	optional list of library sizes
`thresh`	lower threshold of counts (4 by default)
`minSamples`	minimum number of samples

Value

dataframe of counts without the low-count genes

See Also

`log2CPM` which this uses to decide what to keep

Examples

```
## filtered_table = filter_counts(count_table)
```

`graph_metrics` *Make a bunch of graphs describing the state of an experiment before/after normalization.*

Description

Make a bunch of graphs describing the state of an experiment before/after normalization.

Usage

```
graph_metrics(expt, norm_type = "quant", filter = "log2",
  out_type = "cpm", filter_low = TRUE, cormethod = "pearson",
  distmethod = "euclidean")
```

Arguments

expt	an expt experiment containing the data, design, and colors
norm_type	normalization strategy for the data. Defaults to quantile.
filter	whether to log2/10 filter the data. Defaults to log2.
out_type	whether to cpm/rpkm the data. Defaults to cpm.
filter_low	whether to low-count filter the data. Defaults to TRUE.
cormethod	define the correlation test for heatmaps. Defaults to pearson (Available: pearson, spearman, kendal, robust)
distmethod	define the distance metric for heatmaps. Defaults to euclidean (Lots are available, I don't understand them.)

Value

a list of plots. This is a mix of ggplots and replayed built-ins.

See Also

exprs, my_norm, graph_nonzero, my_libsize, my_boxplot, my_corheat, my_smc, my_disheat, my_smd, my_pca, replayPlot, recordPlot

Examples

```
## toomany_plots = graph_metrics(expt)
## testnorm = graph_metrics(expt, norm_type="tmm", filter="log2", out_type="rpkm", cormethod="robust")
```

---

graph_nonzero *Make a ggplot graph of the number of non-zero genes by sample.*

Description

Make a ggplot graph of the number of non-zero genes by sample.

Usage

```
graph_nonzero(df = NULL, design = NULL, colors = NULL, expt = NULL)
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme

Value

nonzero_plot a ggplot2 plot of the number of non-zero genes with respect to each library's CPM

See Also

geom_point, geom_dl

Examples

```
## nonzero_plot = graph_nonzero(expt=expt)
## nonzero_plot  ## ooo pretty
```

---

my_boxplot *Make a ggplot boxplot of a set of samples.*

Description

Make a ggplot boxplot of a set of samples.

Usage

```
my_boxplot(df = NULL, colors = NULL, expt = NULL, names = NULL,
  scale = "raw")
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme
names	a nicer version of the sample names
scale	whether to log scale the y-axis

Value

box_plot a ggplot2 boxplot of the samples. Each boxplot contains the following information: a centered line describing the median value of counts of all genes in the sample, a box around the line describing the inner-quartiles around the median (quartiles 2 and 3 for those who are counting), a vertical line above/below the box which shows 1.5x the inner quartile range (a common metric of the non-outliers), and single dots for each gene which is outside that range. A single dot is transparent.

See Also

geom_boxplot, melt, scale_x_discrete

Examples

```
## a_boxplot = my_boxplot(expt=expt)
## a_boxplot  ## ooo pretty boxplot look at the lines
```

`my_cor` *Wrap cor() to include robust correlations*

Description

Wrap cor() to include robust correlations

Usage

```
my_cor(df = NULL, method = "pearson")
```

Arguments

`df`	a data frame to test
`method`	Correlation method to use. Defaults to pearson. Includes pearson, spearman, kendal, robust.

Value

correlation some fun correlation statistics

See Also

cor, cov, covRob

Examples

```
## my_cor(df=df)
## my_cor(df=df, method="robust")
```

`my_corheat` *Make a heatmap2 description of the correlation between samples.*

Description

Make a heatmap2 description of the correlation between samples.

Usage

```
my_corheat(df = NULL, colors = NULL, design = NULL, expt = NULL,
  method = "pearson")
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme
method	correlation statistic to use. Defaults to pearson.

Value

corheat_plot a gplots heatmap describing how the samples pairwise correlate with one another.

See Also

my_cor, brewer.pal, heatmap.2, recordPlot

Examples

```
## corheat_plot = my_corheat(expt=expt, method="robust")
## corheat_plot
```

---

my_disheat *Make a heatmap2 description of the distance between samples.*

Description

Make a heatmap2 description of the distance between samples.

Usage

```
my_disheat(df = NULL, colors = NULL, design = NULL, expt = NULL,
  method = "euclidean")
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme
method	distance metric to use. Defaults to euclidean. Available: euclidean, manhattan, maximum, canberra, binary, minkowski, I might add more

Value

disheat_plot a gplots heatmap describing the distance between each pair of samples.

See Also

dist, brewer.pal, heatmap.2, recordPlot

Examples

```
## disheat_plot = my_disheat(expt=expt)
## disheat_plot  ## ooo blue
```

---

my_dist_scatter *Make a pretty scatter plot between two sets of numbers with a cheesy distance metric and some statistics of the two sets.*

Description

Make a pretty scatter plot between two sets of numbers with a cheesy distance metric and some statistics of the two sets.

Usage

```
my_dist_scatter(df, tooltip_data = NULL, gvis_filename = NULL)
```

Arguments

df	a dataframe likely containing two columns
gvis_filename	a filename to write a fancy html graph. Defaults to NULL in which case the following parameter isn't needed.
tooltip_data	a df of tooltip information for gvis graphs. NULL by default.

Value

a ggplot2 scatter plot. This plot provides a "bird's eye" view of two data sets. This plot assumes the two data structures are not correlated, and so it calculates the median/mad of each axis and uses these to calculate a stupid, home-grown distance metric away from both medians. This distance metric is used to color dots which are presumed the therefore be interesting because they are far from 'normal.' This will make a fun clicky googleVis graph if requested.

The distance metric should be codified and made more intelligent. Currently it creates a dataframe of distances which are absolute distances from each axis, multiplied by each other, summed by axis, then normalized against the maximum.

See Also

my_gvis_scatter, geom_scatter, hsv, my_linear_scatter

Examples

```
## my_dist_scatter(lotsofnumbers_intwo_columns, tooltip_data=tooltip_dataframe, gvis_filename="html/fun_scatter
```

my_gvis_ma_plot *Make an html version of an MA plot.*

Description

Make an html version of an MA plot.

Usage

```
my_gvis_ma_plot(counts, degenes, tooltip_data = NULL,
  filename = "html/gvis_ma_plot.html", base_url = "")
```

Arguments

counts	df of linear-modelling, normalized counts by sample-type, which is to say the output from voom/voomMod/my_voom().
de_genes	df from toptable or its friends containing p-values.
adjpval_cutoff	a cutoff defining significant from not. Defaults to 0.05.
gvis_filename	a filename to write a fancy html graph. Defaults to NULL in which case the following parameter isn't needed.
tooltip_data	a df of tooltip information for gvis graphs. NULL by default.

Value

NULL, but along the way an html file is generated which contains a googleVis MA plot. See my_ma_plot() for details.

See Also

my_ma_plot

Examples

```
## my_gvis_ma_plot(voomed_data, toptable_data, filename="html/fun_ma_plot.html", base_url="http://yeastgenome.o
```

my_gvis_scatter *Make an html version of a scatter plot.*

Description

Make an html version of a scatter plot.

Usage

```
my_gvis_scatter(df, tooltip_data = NULL,
  filename = "html/gvis_scatter.html", base_url = "")
```

Arguments

counts	df of two columns to compare
filename	a filename to write a fancy html graph. Defaults to NULL in which case the following parameter isn't needed.
tooltip_data	a df of tooltip information for gvis graphs. NULL by default.
base_url	a url to send click events which will be suffixed with the gene name

Value

NULL, but along the way an html file is generated which contains a googleVis scatter plot. See my_scatter_plot() for details.

See Also

gvisScatterChart

Examples

```
## my_gvis_scatter(a_dataframe_twocolumns, filename="html/fun_scatter_plot.html", base_url="http://yeastgenome.
```

my_histogram *Make a pretty histogram of something*

Description

Make a pretty histogram of something

Usage

```
my_histogram(df)
```

Arguments

df	a dataframe of lots of pretty numbers

Value

a ggplot histogram

See Also

geom_histogram, geom_density,

Examples

```
## kittytime = my_histogram(df)
```

---

my_libsize *Make a ggplot graph of library sizes.*

Description

Make a ggplot graph of library sizes.

Usage

```
my_libsize(df = NULL, design = NULL, colors = NULL, expt = NULL,
  scale = TRUE, names = NULL)
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme
scale	whether or not to log10 the y-axis

Value

libsize_plot a ggplot2 plot of each library's size

See Also

geom_bar, geom_text, prettyNum, scale_y_log10

Examples

```
## libsize_plot = my_libsize(expt=expt)
## libsize_plot  ## ooo pretty bargraph
```

my_linear_scatter *Make a pretty scatter plot between two sets of numbers with a linear model superimposed and some supporting statistics.*

Description

Make a pretty scatter plot between two sets of numbers with a linear model superimposed and some supporting statistics.

Usage

```
my_linear_scatter(df, tooltip_data = NULL, gvis_filename = NULL,
  cormethod = "pearson")
```

Arguments

df	a dataframe likely containing two columns
gvis_filename	a filename to write a fancy html graph. Defaults to NULL in which case the following parameter isn't needed.
tooltip_data	a df of tooltip information for gvis graphs. NULL by default.
cormethod	what type of correlation to check? Defaults to 'pearson'

Value

a list including a ggplot2 scatter plot and some histograms. This plot provides a "bird's eye" view of two data sets. This plot assumes a (potential) linear correlation between the data, so it calculates the correlation between them. It then calculates and plots a robust linear model of the data using an 'SMDM' estimator (which I don't remember how to describe, just that the document I was reading said it is good). The median/mad of each axis is calculated and plotted as well. The distance from the linear model is finally used to color the dots on the plot. Histograms of each axis are plotted separately and then together under a single cdf to allow tests of distribution similarity. This will make a fun clicky googleVis graph if requested.

See Also

lmrob, weights, hsv, mad, my_histogram

Examples

```
## my_linear_scatter(lotsofnumbers_intwo_columns, tooltip_data=tooltip_dataframe, gvis_filename="html/fun_scatt
```

my_ma_plot *Make a pretty MA plot from the output of voom/limma/eBayes/toptable*

Description

Make a pretty MA plot from the output of voom/limma/eBayes/toptable

Usage

```
my_ma_plot(counts, de_genes, adjpval_cutoff = 0.05, alpha = 0.6, size = 2,
  tooltip_data = NULL, gvis_filename = NULL)
```

Arguments

counts	df of linear-modelling, normalized counts by sample-type, which is to say the output from voom/voomMod/my_voom().
de_genes	df from toptable or its friends containing p-values.
adjpval_cutoff	a cutoff defining significant from not. Defaults to 0.05.
alpha	how transparent to make the dots. Defaults to 0.6.
size	how big are the dots? Defaults to 2.
gvis_filename	a filename to write a fancy html graph. Defaults to NULL in which case the following parameter isn't needed.
tooltip_data	a df of tooltip information for gvis graphs. NULL by default.

Value

a ggplot2 MA scatter plot. This is defined as the rowmeans of the normalized counts by type across all sample types on the x-axis, and the log fold change between conditions on the y-axis. Dots are colored depending on if they are 'significant.' This will make a fun clicky googleVis graph if requested.

See Also

my_gvis_ma_plot, toptable, voom, voomMod, my_voom, lmFit, makeContrasts, contrasts.fit

Examples

```
## my_ma_plot(voomed_data, toptable_data, gvis_filename="html/fun_ma_plot.html")
## Currently this assumes that a variant of toptable was used which
## gives adjusted p-values.  This is not always the case and I should
## check for that, but I have not yet.
```

my_multihistogram *Make a pretty histogram of multiple datasets*

Description

Make a pretty histogram of multiple datasets

Usage

```
my_multihistogram(df)
```

Arguments

df	a dataframe of lots of pretty numbers

Value

a ggplot histogram comparing multiple data sets Along the way this generates pairwise t tests of the columns of data.

See Also

pairwise.t.test, ddply, rbind

Examples

```
## kittytime = my_multihistogram(df)
```

my_norm *Normalize a dataframe/expt, express it, and/or transform it*

Description

Normalize a dataframe/expt, express it, and/or transform it

Usage

```
my_norm(df = NULL, expt = NULL, design = NULL, out_type = "raw",
  norm_type = "raw", filter = "raw", filter_low = TRUE)
```

Arguments

expt=expt	an expt class containing all the necessary metadata
df=df	alternately a dataframe of counts may be used
design=design	but a design dataframe must come with it
out_type	defines the output type which may be raw, cpm, or rpkm. Defaults to raw.
filter	defines whether to log(2|10) transform the data. Defaults to raw.
norm_type	specify the normalization strategy. Defaults to raw. This makes use of DESeq/EdgeR/cbcbSEQ to provide: quantile, RLE, upperquartile, size-factor, or tmm normalization. I tend to like quantile, but there are definitely corner-case scenarios for all strategies.
filter_low	choose whether to low-count filter the data. Defaults to true.

Value

edgeR's DGEList expression of a count table. This seems to me to be the easiest to deal with.

See Also

cpm, rpkm, my_rpkm, filterCounts, DESeqDataSetFromMatrix, estimateSizeFactors, DGEList, qNorm, calcNormFactors

Examples

```
## df_raw = my_norm(expt=expt)  ## Only performs low-count filtering
## df_raw = my_norm(df=a_df, design=a_design) ## Same, but using a df
## df_ql2rpkm = my_norm(expt=expt, norm_type='quant', filter='log2', out_type='rpkm')  ## Quantile, log2, rpkm
## count_table = df_ql2rpkm$counts
## library_size = df_ql2rpkm$lib.size
```

---

my_pca *Make a ggplot PCA plot describing the samples' clustering*

Description

Make a ggplot PCA plot describing the samples' clustering

Usage

```
my_pca(df = NULL, colors = NULL, design = NULL, expt = NULL,
  names = NULL)
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme
method	a correlation method to use. Defaults to pearson.
names	use pretty names for the samples?

Value

pca_plot a ggplot2 describing the principle component analysis of the samples. This makes use of cbcbSEQ and prints the table of variance by component.

See Also

makeSVD, pcRes, geom_dl

Examples

```
## pca_plot = my_pca(expt=expt)
## pca_plot
```

---

my_read_files *Read a bunch of count tables and create a usable data frame from them.*

Description

Read a bunch of count tables and create a usable data frame from them.

Usage

```
my_read_files(ids, files)
```

Arguments

ids	a list of experimental ids
files	a list of files to read

Value

initial_count a data frame of count tables

See Also

create_experiment

Examples

```
## count_tables = my_read_files(as.character(sample_ids), as.character(count_filenames))
```

---

my_rpkm *Express a data frame of counts as reads per killobase(gene) per million(library).*

Description

Express a data frame of counts as reads per killobase(gene) per million(library).

Usage

```
my_rpkm(df, annotations = gene_annotations)
```

Arguments

df	a data frame of counts, alternately an edgeR DGEList
annotations	containing gene lengths, defaulting to 'gene_annotations'

Value

rpkm_df a data frame of counts expressed as rpkm

See Also

edgeR and cpm

Examples

```
## rpkm_df = my_rpkm(df, annotations=gene_annotations)
```

---

my_smc *Make an R plot of the standard median correlation among samples*

Description

Make an R plot of the standard median correlation among samples

Usage

```
my_smc(df = NULL, colors = NULL, expt = NULL, method = "pearson")
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme
method	a correlation method to use. Defaults to pearson.
names	use pretty names for the samples?

Value

smc_plot a recordPlot of plot. This will also write to an open device. The resulting plot measures the median correlation of each sample among its peers. It notes 1.5* the interquartile range among the samples and makes a horizontal line at that correlation coefficient. Any sample which falls below this line is considered for removal because it is much less similar to all of its peers.

See Also

my_cor, matrixStats::rowMedians, quantile, diff, recordPlot

Examples

```
## smc_plot = my_smc(expt=expt)
```

---

my_smd *Make an R plot of the standard median distance among samples*

Description

Make an R plot of the standard median distance among samples

Usage

```
my_smd(expt = NULL, df = NULL, colors = NULL, method = "euclidean")
```

Arguments

expt	an expt set of samples
df	alternately a data frame which must be accompanied by
design	a design matrix and
colors	a color scheme
method	a distance metric to use. Defaults to euclidean.
names	use pretty names for the samples?

Value

smd_plot a recordPlot of plot. This will also write to an open device. This plot takes the median distance of each sample with all of its peers. It then calculates 1.5* the interquartile range of distances. Any sample which has a median distance greater than this is considered for removal.

See Also

dist, quantile, diff, recordPlot

Examples

```
## smd_plot = my_smd(expt=expt)
```

`my_voom` *Estimate mean-variance relationship between samples and generate 'observational-level weights' in preparation for linear modelling RNAseq data. This particular implementation was primarily scabbed from cbcbSEQ, but changes the mean-variance plot slightly and attempts to handle corner cases where the sample design is confounded by setting the coefficient to 1 for those samples rather than throwing an unhelpful error. Also, the Elist output gets a 'plot' slot which contains the plot rather than just printing it.*

Description

Estimate mean-variance relationship between samples and generate 'observational-level weights' in preparation for linear modelling RNAseq data. This particular implementation was primarily scabbed from cbcbSEQ, but changes the mean-variance plot slightly and attempts to handle corner cases where the sample design is confounded by setting the coefficient to 1 for those samples rather than throwing an unhelpful error. Also, the Elist output gets a 'plot' slot which contains the plot rather than just printing it.

Usage

```
my_voom(dataframe, model, libsize = NULL)
```

Arguments

`dataframe`	a dataframe of sample counts which have been normalized and log transformed
`model`	an experimental model defining batches/conditions/etc
`libsize`	the size of the libraries (usually provided by edgeR). By default this is NULL.

Value

an EList containing the modified data, weights, design, libsize, and mean-variance plot.

See Also

voom, voomMod, lmFit

Examples

```
## funkytown = my_voom(samples, model)
```

`perform_comparison` *Perform a quick and dirty differential expression comparison*

Description

Perform a quick and dirty differential expression comparison

Usage

```
perform_comparison(expt_subset, model, time = "LL", media = "THY",
  new_media = NULL, mga = "wt", basename = "testing")
```

Arguments

`expt_subset`	an experiment subset including normalized samples and an experimental design
`model`	a model matrix defining what to compare

Value

a list including the voomed data, fitted data by lmFit, contrasts from limma, some histograms of the data, a table of the differentially expressed data from toptable. Along the way this attempts to print some hopefully useful diagnostic information and write the relevant tables to an excel file. In its current implementation it assumes a specific experiment, which is bad. In addition it must perforce assume that the first provided condition is a control and the second is an experimental. It changes the columns accordingly.

See Also

voom, lmFit, makeContrasts, contrasts.fit, eBayes, topTable, my_linear_scatter, write_xls, my_ma_plot

Examples

```
## de_information = perform_comparison(expt_subset, model)
```

require.auto *Automatic loading and/or installing of packages.*

Description

Automatic loading and/or installing of packages.

Usage

```
require.auto(lib)
```

Arguments

lib	string name of a library

Value

NULL currently

See Also

biocLite and install.packages

Examples

```
## require.auto("ggplot2")
```

write_xls *Write a dataframe to an excel spreadsheet sheet.*

Description

Write a dataframe to an excel spreadsheet sheet.

Usage

```
write_xls(data, sheet, file = "excel/workbook.xls", rowname = NA)
```

Arguments

data	a dataframe of information
sheet	the name of an excel sheet in a workbook.
file	an excel workbook to which to write. Defaults to "excel/workbook.xls"
rowname	include rownames? Defalts to no.

Value

NULL, on the say it creates a workbook if necessary, creates a sheet, and writes the data to it.

See Also

loadWorkbook, createSheet, writeWorksheet, saveWorkbook

Examples

```
## write_xls(dataframe, "my_data")
## re-create it if this is used heavily, because it will get crufty.
```

Index