Package 'hpgltools'

February 4, 2016

Type Package

Title A pile of (hopefully) useful R functions

Version 2016.02

Date 2016-02-01

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-2 | file LICENSE

Suggests affy, AnnotationDbi, Biobase, BiocGenerics, Biostrings, biomaRt, Category, cbcbSEQ, clusterProfiler, corpcor, data.table, DESeq2, DESeq, devtools, directlabels, DOSE, edgeR, genefilter, genomeIntervals, GenomicRanges, ggplot2, GO.db, googleVis, goseq, GOstats, gplots, graph, GSEABase, gtools, gridExtra, hash, Hmisc, igraph, IRanges, KEGGREST, knitcitations, knitr, knitrBootstrap, lattice, limma, matrixStats, methods, motifRG, motifStack, multtest, openxlsx, pathview, plyr, preprocessCore, qsmooth, qvalue, RamiGO, RColorBrewer, ReactomePA, reshape2, RCurl, rGADEM, Rgraphviz, robustbase, RUVSeq, reshape, rjson, rmarkdown, robust, roxygen2, Rsamtools, rtracklayer, S4Vectors, scales, seqinr, seqLogo, SeqTools, stringi, stringr, survJamda, sva, testthat, topGO, xtable, XVector

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

R topics documented:

`all_ontology_searches` *Perform ontology searches of the output from limma.*

Description

This passes a set of limma results to (optionally) goseq, clusterprofiler, topgo, and gostats, collects the outputs, and provides them as a list. This function needs a species argument, as I recently made the simple_() functions able to automatically use the various supported organisms.

Usage

```
all_ontology_searches(de_out, gene_lengths = NULL, goids = NULL, n = NULL,
  z = NULL, fc = NULL, p = NULL, overwrite = FALSE,
  goid_map = "reference/go/id2go.map", gff_file = NULL, gff_type = "gene",
  goids_df = NULL, do_goseq = TRUE, do_cluster = TRUE, do_topgo = TRUE,
  do_gostats = TRUE, do_trees = FALSE)
```

Arguments

`de_out`	a list of topTables comprising limma/deseq/edger outputs.
`gene_lengths`	a data frame of gene lengths for goseq.
`goids`	a data frame of goids and genes.
`n`	a number of genes at the top/bottom to search.
`z`	a number of standard deviations to search. (if this and n are null, it assumes 1z)
`fc`	a number of standard deviations to search. (if this and n are null, it assumes 1z)
`p`	a maximum pvalue
`overwrite`	overwrite the excel file
`goid_map`	a map file used by topGO, if it does not exist then provide goids_df to make it.

gff_file	a gff file containing the annotations used by gff2genetable from clusterprofiler, which I hacked to make faster.
gff_type	column to use from the gff file
goids_df	FIXME! a dataframe of genes and goids which I am relatively certain is no longer needed and superseded by goids.
do_goseq	perform simple_goseq()?
do_cluster	perform simple_clusterprofiler()?
do_topgo	perform simple_topgo()?
do_gostats	perform simple_gostats()?
do_trees	make topGO trees from the data?

Value

a list of up/down ontology results from goseq/clusterprofiler/topgo/gostats, and associated trees, all optionally.

Examples

```
## Not run:
 many_comparisons = limma_pairwise(expt=an_expt)
 tables = many_comparisons$limma
 this_takes_forever = limma_ontology(tables, gene_lengths=lengthdb, goids=goids_df, z=1.5, gff_file='length_db.g

## End(Not run)
```

---

all_pairwise *Wrap up limma/DESeq2/EdgeR pairwise analyses in one call.*

Description

Wrap up limma/DESeq2/EdgeR pairwise analyses in one call.

Usage

```
all_pairwise(input, conditions = NULL, batches = NULL, model_cond = TRUE,
  model_batch = TRUE, model_intercept = FALSE, extra_contrasts = NULL,
  alt_model = NULL, libsize = NULL, annot_df = NULL, ...)
```

Arguments

input	a dataframe/vector or expt class containing count tables, normalization state, etc.
conditions	a factor of conditions in the experiment
batches	a factor of batches in the experiment
model_cond	include condition in the model? This is likely always true.
model_batch	include batch in the model?

model_intercept	use an intercept model instead of cell means?
extra_contrasts	some extra contrasts to add to the list This can be pretty neat, lets say one has conditions A,B,C,D,E and wants to do (C/B)/A and (E/D)/A or (E/D)/(C/B) then use this with a string like: "c_vs_b_ctrla = (C-B)-A, e_vs_d_ctrla = (E-D)-A, de_vs_cb = (E-D)-(C-B),"
alt_model	an optional alternate model to use rather than just condition/batch
libsize	the library size of the original data to help voom()
annot_df	annotations to add to the tables
...	The elipsis parameter is fed to write_limma() at the end.

Value

A list of limma, deseq, edger results.

Examples

```
## Not run:
 finished_comparison = eBayes(limma_output)
 data_list = write_limma(finished_comparison, workbook="excel/limma_output.xls")

## End(Not run)
```

---

autoloads_all *Automatic loading of stuff I use, I am deprecating this now.*

Description

Automatic loading of stuff I use, I am deprecating this now.

Usage

```
autoloads_all(update = FALSE)
```

Arguments

update	packages?

Value

NULL currently

See Also

biocLite install.packages

`backup_file` *Make a backup of an existing file with n revisions, like VMS!*

Description

Make a backup of an existing file with n revisions, like VMS!

Usage

```
backup_file(backup_file, backups = 4)
```

Arguments

`backup_file`	the file to backup.
`backups`	how many revisions?

`basic_pairwise` *Perform a pairwise comparison among conditions which takes nothing into account. It _only_ takes the conditions, a mean value/variance among them, divides by condition, and returns the result. No fancy nomalizations, no statistical models, no nothing. It should be the very worst method possible. But, it should also provide a baseline to compare the other tools against, they should all do better than this, always.*

Description

Perform a pairwise comparison among conditions which takes nothing into account. It _only_ takes the conditions, a mean value/variance among them, divides by condition, and returns the result. No fancy nomalizations, no statistical models, no nothing. It should be the very worst method possible. But, it should also provide a baseline to compare the other tools against, they should all do better than this, always.

Usage

```
basic_pairwise(input, design = NULL)
```

Arguments

`input`	a count table by sample
`design`	a data frame of samples and conditions

Value

I am not sure yet

See Also

limma DESeq2 edgeR

Examples

```
## Not run:
stupid_de <- basic_pairwise(expt)

## End(Not run)
```

`batch_counts` *Perform different batch corrections using limma, sva, ruvg, and cbcbSEQ.*

Description

Perform different batch corrections using limma, sva, ruvg, and cbcbSEQ.

Usage

```
batch_counts(count_table, design, batch = TRUE, batch1 = "batch",
  batch2 = NULL, noscale = TRUE, ...)
```

Arguments

`count_table`	a matrix of (pseudo)counts.
`design`	a model matrix defining the experimental conditions/batches/etc
`batch`	a string describing the method to try to remove the batch effect (or FALSE to leave it alone, TRUE uses limma)
`batch1`	the column in the design table describing the presumed covariant to remove.
`batch2`	the column in the design table describing the second covariant to remove (only used by limma at the moment).
`noscale`	used for combatmod, when true it removes the scaling parameter from the invocation of the modified combat.
`...`	more options for you!

Value

The 'batch corrected' count table and new library size. Please remember that the library size which comes out of this may not be what you want for voom/limma and would therefore lead to spurious differential expression values.

See Also

limma edgeR RUVSeq sva cbcbSEQ

Examples

```
## Not run:
limma_batch <- batch_counts(table, design, batch1='batch', batch2='strain')
sva_batch <- batch_counts(table, design, batch='sva')

## End(Not run)
```

`Beta.NA` *Beta.NA: Perform a quick solve to gather residuals etc This was provided by Kwame for something which I don't remember a loong time ago.*

Description

Beta.NA: Perform a quick solve to gather residuals etc This was provided by Kwame for something which I don't remember a loong time ago.

Usage

```
Beta.NA(y, X)
```

Arguments

y	a y
X	a x

`cbcb_batch_effect` *A function suggested by Hector Corrada Bravo and Kwame Okrah for batch removal*

Description

During a lab meeting, the following function was suggested as a quick and dirty batch removal tool

Usage

```
cbcb_batch_effect(normalized_counts, model)
```

Arguments

normalized_counts	a data frame of log2cpm counts
model	a balanced experimental model containing condition and batch factors

Value

a dataframe of residuals after subtracting batch from the model

See Also

voom lmFit

Examples

```
## Not run:
newdata <- cbcb_batch_effect(counts, expt_model)

## End(Not run)
```

---

cbcb_filter_counts *Filter low-count genes from a data set.*

Description

This was a function written by Kwame Okrah and perhaps also Laura Dillon to remove low-count genes. It drops genes based on a threshold and number of samples.

Usage

```
cbcb_filter_counts(count_table, threshold = 2, min_samples = 2,
  verbose = FALSE)
```

Arguments

count_table	a data frame of (pseudo)counts by sample.
threshold	lower threshold of counts for each gene.
min_samples	minimum number of samples
verbose	if set to true, prints number of genes removed and remaining.

Value

dataframe of counts without the low-count genes

See Also

log2CPM which this uses to decide what to keep

Examples

```
## Not run:
filtered_table <- cbcb_filter_counts(count_table)

## End(Not run)
```

cbcb_lowfilter_counts *Filter low-count genes from a data set using cbcbSEQ::filterCounts()*

Description

Filter low-count genes from a data set using cbcbSEQ::filterCounts()

Usage

```
cbcb_lowfilter_counts(count_table, thresh = 2, min_samples = 2,
  verbose = FALSE)
```

Arguments

count_table	input data frame of counts by sample
thresh	lower threshold of counts (default: 4)
min_samples	minimum number of samples (default: 2)
verbose	If set to true, prints number of genes removed / remaining

Value

dataframe of counts without the low-count genes

See Also

log2CPM which this uses to decide what to keep

Examples

```
## Not run:
 filtered_table = cbcb_lowfilter_counts(count_table)

## End(Not run)
```

check_clusterprofiler *Make sure that clusterProfiler is ready to run*

Description

Make sure that clusterProfiler is ready to run

Usage

```
check_clusterprofiler(gff = "test.gff", gomap = NULL)
```

Arguments

gff	The gff file containing annotation data (gene lengths)
gomap	a data frame of gene IDs and GO ontologies 1:1, other columns are ignored.

Value

the GO2EG data structure created, probably don't save this, its big

Examples

```
## Not run:
 go2eg <- check_clusterprofiler(gff, goids)
 rm(go2eg)

## End(Not run)
```

---

circos_arc *Write arcs between chromosomes in circos.*

Description

Ok, so when I said I only do 1 chromosome images, I lied. This function tries to make writing arcs between chromosomes easier. It too works in 3 stages, It writes out a data file using cfgout as a basename and the data from df in the circos arc format into circos/data/bob_arc.txt It then writes out a configuration plot stanza in circos/conf/bob_arc.conf and finally adds an include to circos/bob.conf

Usage

```
circos_arc(df, cfgout = "circos/conf/default.conf", first_col = "chr1",
  second_col = "chr2", color = "blue", radius = 0.75, thickness = 3)
```

Arguments

df	a dataframe with starts/ends and the floating point information
cfgout	The master configuration file to write.
first_col	The name of the first chromosome
second_col	The name of the second chromosome
color	the color of the histogram
radius	the radius at which to add the arcs
thickness	integer thickness of the arcs

Details

In its current implementation, this only understands two chromosomes. A minimal amount of logic and data organization will address this weakness.

Value

undef

`circos_heatmap` *Write tiles of arbitrary heat-mappable data in circos.*

Description

This function tries to make the writing circos heatmaps easier. Like circos_plus_minus() and circos_hist() it works in 3 stages, It writes out a data file using cfgout as a basename and the data from df in the circos histogram format into circos/data/bob_heatmap.txt It then writes out a configuration plot stanza in circos/conf/bob_heatmap.conf and finally adds an include to circos/bob.conf

Usage

```
circos_heatmap(df, cfgout = "circos/conf/default.conf", colname = "datum",
  chr = "chr1", colors = NULL, outer = 0.9, width = 0.08, spacing = 0)
```

Arguments

`df`	a dataframe with starts/ends and the floating point information
`cfgout`	The master configuration file to write.
`colname`	The name of the column with the data of interest.
`chr`	the name of the chromosome (This currently assumes a bacterial chromosome)
`colors`	the color of the histogram
`outer`	the floating point radius of the circle into which to place the plus-strand data
`width`	the radial width of each tile
`spacing`	the radial distance between outer,inner and inner,whatever follows.

Value

the radius after adding the histogram and the spacing.

circos_hist *Write histograms of arbitrary floating point data in circos.*

Description

This function tries to make the writing of histogram data in circos easier. Like circos_plus_minus() it works in 3 stages, It writes out a data file using cfgout as a basename and the data from df in the circos histogram format into circos/data/bob_hist.txt It then writes out a configuration plot stanza in circos/conf/bob_hist.conf and finally adds an include to circos/bob.conf

Usage

```
circos_hist(df, cfgout = "circos/conf/default.conf", colname = "datum",
  chr = "chr1", color = "blue", fill_color = "blue", outer = 0.9,
  width = 0.08, spacing = 0)
```

Arguments

df	a dataframe with starts/ends and the floating point information
cfgout	The master configuration file to write.
colname	The name of the column with the data of interest.
chr	The name of the chromosome (This currently assumes a bacterial chromosome)
color	the color of the histogram
fill_color	guess
outer	the floating point radius of the circle into which to place the plus-strand data
width	the radial width of each tile
spacing	the radial distance between outer,inner and inner,whatever follows.

Value

the radius after adding the histogram and the spacing.

circos_ideogram *Create the description of chromosome markings*

Description

This function writes ideogram files for circos. Currently it only has a single format.

Usage

```
circos_ideogram(name = "default", conf_dir = "circos/conf",
  band_url = NULL)
```

Arguments

name	the name of the configuration
conf_dir	where does the configuration live?
band_url	provide a url for making these imagemaps.

Value

undef

---

circos_karyotype *Create the description of (a)chromosome(s) for circos.*

Description

This function tries to save me from having to get the lengths of arcs for bacterial chromosomes manually correct, and writes them as a circos compatible karyotype file. The outfile parameter was chosen to match the configuration directive outlined in circos_prefix(), however that will need to be changed in order for this to work in variable conditions. Next time I make one of these graphs I will do that I suspect. In addition, this currently only understands how to write bacterial chromosomes, that will likely be fixed when I am asked to write out a L.major karyotype.

Usage

```
circos_karyotype(name = "default", conf_dir = "circos/conf",
  length = NULL, chr_name = "chr1", segments = 6, color = "white",
  chr_num = 1, fasta = NULL)
```

Arguments

name	the name of the chromosome (This currently assumes a bacterial chromosome)
conf_dir	where to put the circos configuration
length	the default length of the chromosome (That is mgas5005)
chr_name	the name of the chromosome
segments	how many segments to cut it into
color	how to colors the chromosomal arc. (circos images are cluttered enough)
chr_num	the number to record (This and name above should change for multi-chromosomal species)
fasta	fasta file to use to create the karyotype

Details

These defaults were chosen because I have a chromosome of this length that is correct.

Value

undef

circos_make — *Write a simple makefile for circos.*

Description

I regenerate all my circos pictures with make(1). This is my makefile.

Usage

```
circos_make(target = "", output = "circos/Makefile",
  circos = "/usr/bin/circos")
```

Arguments

target	the make target
output	the makefile
circos	the location of circos. I have a copy in home/bin/circos and use that sometimes.

Value

a kitten

circos_plus_minus — *Write tiles of bacterial ontology groups using the categories from microbesonline.org*

Description

This function tries to save me from writing out ontology definitions and likely making mistakes. It uses the start/ends from the gff annotation along with the 1 letter GO-like categories from microbesonline.org. It then writes two data files circos/data/bob_plus_go.txt, circos/data/bob_minus_go.txt along with two configuration files circos/conf/bob_minus_go.conf and circos/conf/bob_plus_go.conf and finally adds an include to circos/bob.conf

Usage

```
circos_plus_minus(go_table, cfgout = "circos/conf/default.conf",
  chr = "chr1", outer = 1, width = 0.08, spacing = 0)
```

Arguments

go_table	a dataframe with starts/ends and categories
cfgout	The master configuration file to write.
chr	The name of the chromosome (This currently assumes a bacterial chromosome)
outer	the floating point radius of the circle into which to place the plus-strand data
width	the radial width of each tile
spacing	the radial distance between outer,inner and inner,whatever follows.

Value

the radius after adding the plus/minus information and the spacing between them.

`circos_prefix` *Write the beginning of a circos configuration file.*

Description

A few parameters need to be set when starting circos. This sets some of them and gets ready for plot stanzas.

Usage

```
circos_prefix(name = "mgas", conf_dir = "circos/conf", radius = 1800,
  band_url = NULL)
```

Arguments

`name`	The name of the map, called with 'make name'
`conf_dir`	The directory containing the circos configuration data.
`radius`	The size of the image.
`band_url`	a place to imagemap link

Details

In its current implementation, this really assumes that there will be no highlight stanzas and at most 1 link stanza. chromosomes. A minimal amount of logic and data organization will address these weaknesses.

Value

undef

`circos_suffix` *Write the end of a circos master configuration.*

Description

circos configuration files need an ending. This writes it.

Usage

```
circos_suffix(cfgout = "circos/conf/default.conf")
```

Arguments

cfgout	The master configuration file to write.

Value

undef

`circos_tile` *Write tiles of arbitrary categorical point data in circos.*

Description

This function tries to make the writing circos tiles easier. Like circos_plus_minus() and circos_hist() it works in 3 stages, It writes out a data file using cfgout as a basename and the data from df in the circos histogram format into circos/data/bob_tile.txt It then writes out a configuration plot stanza in circos/conf/bob_tile.conf and finally adds an include to circos/bob.conf

Usage

```
circos_tile(df, cfgout = "circos/conf/default.conf", colname = "datum",
  chr = "chr1", colors = NULL, outer = 0.9, width = 0.08, spacing = 0)
```

Arguments

df	a dataframe with starts/ends and the floating point information
cfgout	The master configuration file to write.
colname	The name of the column with the data of interest.
chr	the name of the chromosome (This currently assumes a bacterial chromosome)
colors	the color of the histogram
outer	the floating point radius of the circle into which to place the plus-strand data
width	the radial width of each tile
spacing	the radial distance between outer,inner and inner,whatever follows.

Value

the radius after adding the histogram and the spacing.

`cluster_trees` *Take clusterprofile group data and print it on a tree as topGO does Make fun trees a la topgo from goseq data.*

Description

Take clusterprofile group data and print it on a tree as topGO does Make fun trees a la topgo from goseq data.

Usage

```
cluster_trees(de_genes, cpdata, goid_map = "reference/go/id2go.map",
  goids_df = NULL, score_limit = 0.2, overwrite = FALSE,
  selector = "topDiffGenes", pval_column = "adj.P.Val")
```

Arguments

`de_genes`	A list of genes deemed 'interesting'
`cpdata`	data from simple_clusterprofiler()
`goid_map`	A mapping file of IDs to GO ontologies
`goids_df`	A dataframe of mappings used to build goid_map
`score_limit`	A scoring limit above which to ignore genes
`overwrite`	Overwrite an existing goid mapping file?
`selector`	The name of a function for applying scores to the trees
`pval_column`	The name of the column in the table from which to extract scores

Value

plots! Trees! oh my!

See Also

Ramigo `showSigOfNodes`

Examples

```
## Not run:
cluster_data <- simple_clusterprofiler(genes, stuff)
ctrees <- cluster_trees(genes, cluster_data)

## End(Not run)
```

combine_de_table *Given a limma, edger, and deseq table, combine them*

Description

Given a limma, edger, and deseq table, combine them

Usage

```
combine_de_table(li, ed, de, ba, table, annot_df = NULL, inverse = FALSE,
  include_basic = TRUE)
```

Arguments

li	a limma output
ed	a edger output
de	a deseq output
ba	a basic output
table	name of the table to merge
annot_df	add some annotation information
inverse	invert the fold changes
include_basic	include the basic table?

combine_de_tables *Combine portions of deseq/limma/edger table output*

Description

This hopefully makes it easy to compare the outputs from limma/DESeq2/EdgeR on a table-by-table basis.

Usage

```
combine_de_tables(all_pairwise_result, annot_df = NULL, excel = NULL,
  excel_title = "Table SXXX: Combined Differential Expression of YYY",
  excel_sheet = "combined_DE", keepers = "all", include_basic = TRUE,
  add_plots = TRUE, plot_dim = 3)
```

Arguments

all_pairwise_result	the output from all_pairwise()
annot_df	add some annotation information
excel	print the excel workbook
excel_title	a title, if it has YYY in it, that will be replaced by the contrast name
excel_sheet	name the sheet
keepers	a list of reformatted table names to explicitly keep certain contrasts in specific orders
include_basic	Include my stupid basic logFC tables
add_plots	add plots to the end of the sheets
plot_dim	number of inches squared for the plot if added

Value

a table combinine limma/edger/deseq outputs.

See Also

all_pairwise

Examples

```
## Not run:
pretty = combine_de_tables(big_result, table='t12_vs_t0')

## End(Not run)
```

---

compare_go_searches *Compare the results from different ontology tools*

Description

Combine the results from goseq, cluster profiler, topgo, and gostats; poke at them with a stick and see what happens. The general idea is to pull the p-value data from each tool and contrast that to the set of all possibile ontologies. This allows one to do a correlation coefficient between them. In addition, take the 1-pvalue for each ontology for each tool. Thus for strong p-values the score will be near 1 and so we can sum the scores for all the tools. Since topgo has 4 tools, the total possible is 7 if everything has a p-value equal to 0.

Usage

```
compare_go_searches(goseq = NULL, cluster = NULL, topgo = NULL,
  gostats = NULL)
```

Arguments

goseq	The goseq result from simple_goseq()
cluster	The result from simple_clusterprofiler()
topgo	Guess
gostats	Yep, ditto

Value

a summary of the similarities of ontology searches

---

compare_tables *See how similar are results from limma/deseq/edger.*

Description

limma, DEseq2, and EdgeR all make somewhat different assumptions and choices about what makes a meaningful set of differentially expressed genes. This seeks to provide a quick and dirty metric describing the degree to which they (dis)agree.

Usage

```
compare_tables(limma = NULL, deseq = NULL, edger = NULL, basic = NULL,
  include_basic = TRUE, annot_df = NULL, ...)
```

Arguments

limma	limma data from limma_pairwise()
deseq	deseq data from deseq2_pairwise()
edger	edger data from edger_pairwise()
basic	basic data from basic_pairwise()
include_basic	include the basic data?
annot_df	include annotation data
...	more options!

Value

a heatmap showing how similar they are along with some correlations betwee the three players.

See Also

limma_pairwise edger_pairwise deseq2_pairwise

Examples

```
## Not run:
 l = limma_pairwise(expt)
 d = deseq_pairwise(expt)
 e = edger_pairwise(expt)
 fun = compare_tables(limma=l, deseq=d, edger=e)

## End(Not run)
```

---

concatenate_runs — concatenate_runs() *Sum the reads/gene for multiple sequencing runs of a single condition/batch*

Description

concatenate_runs() Sum the reads/gene for multiple sequencing runs of a single condition/batch

Usage

```
concatenate_runs(expt, column = "replicate")
```

Arguments

expt	an experiment class containing the requisite metadata and count tables
column	a column of the design matrix used to specify which samples are replicates

Value

the input expt with the new design matrix, batches, conditions, colors, and count tables.

See Also

Biobase

Examples

```
## Not run:
 compressed = concatenate_runs(expt)

## End(Not run)
```

convert_counts *Perform a cpm/rpkm/whatever transformation of a count table.*

Description

I should probably tell it to also handle a simple df/vector/list of gene lengths, but I haven't. cp_seq_m is a cpm conversion of the data followed by a rp-ish conversion which normalizes by the number of the given oligo. By default this oligo is 'TA' because it was used for tnseq which should be normalized by the number of possible transposition sites by mariner. It could, however, be used to normalize by the number of methionines, for example – if one wanted to do such a thing.

Usage

```
convert_counts(data, convert = "raw", annotations = NULL, fasta = NULL,
  pattern = "TA", entry_type = "gene", ...)
```

Arguments

data	A matrix of count data
convert	A type of conversion to perform: edgecpm/cpm/rpkm/cp_seq_m
annotations	a set of gff annotations are needed if using rpkm so we can get gene lengths.
fasta	a fasta for rpkmish
pattern	for cp_seq_m counts
entry_type	used to acquire gene lengths
...	more options

Value

dataframe of cpm/rpkm/whatever(counts)

See Also

edgeR Biobase cpm

Examples

```
## Not run:
 converted_table = convert_counts(count_table, convert='edgecpm')

## End(Not run)
```

`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 = NULL, color_hash, suffix = ".count.gz",
  header = FALSE, gene_info = NULL, by_type = FALSE, by_sample = FALSE,
  include_type = "all", include_gff = NULL, count_dataframe = NULL,
  meta_dataframe = NULL, sep = ",", ...)
```

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
`suffix`	when looking for the count tables in processed_data look for this suffix on the end of the files.
`header`	Does the csv metadata file have a header?
`gene_info`	annotation information describing the rows of the data set, usually this comes from a call to import.gff()
`by_type`	when looking for count tables, are they organized by type?
`by_sample`	or by sample? I do all mine by sample, but others do by type...
`include_type`	I have usually assumed that all gff annotations should be used, but that is not always true, this allows one to limit.
`include_gff`	A gff file to help in sorting which features to keep
`count_dataframe`	If one does not wish to read the count tables from processed_data/ they may instead be fed here
`meta_dataframe`	an optional dataframe containing the metadata rather than a file
`sep`	some people prefer their csv files as tab or semicolon separated.
`...`	more parameters

Value

experiment an expressionset

See Also

Biobase pData fData exprs hpgl_read_files as.list.hash

Examples

```
## Not run:
new_experiment = create_experiment("some_csv_file.csv", color_hash)

## End(Not run)
```

`create_expt` *Wrap bioconductor's expressionset to include some other extraneous information. This simply calls create_experiment and then does expt_subset for everything*

Description

this is relevant because the ceph object storage by default lowercases filenames.

Usage

```
create_expt(file = NULL, color_hash = NULL, suffix = ".count.gz",
  header = FALSE, gene_info = NULL, by_type = FALSE, by_sample = FALSE,
  sep = ",", include_type = "all", include_gff = NULL,
  count_dataframe = NULL, meta_dataframe = NULL, savefile = "expt",
  low_files = FALSE, ...)
```

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, it will generate its own colors using colorBrewer
`suffix`	when looking for the count tables in processed_data look for this suffix on the end of the files.
`header`	Does the csv metadata file have a header?
`gene_info`	annotation information describing the rows of the data set, usually this comes from a call to import.gff()
`by_type`	when looking for count tables, are they organized by type?
`by_sample`	or by sample? I do all mine by sample, but others do by type...
`sep`	some people prefer their csv files as tab or semicolon separated.
`include_type`	I have usually assumed that all gff annotations should be used, but that is not always true, this allows one to limit.
`include_gff`	A gff file to help in sorting which features to keep
`count_dataframe`	If one does not wish to read the count tables from processed_data/ they may instead be fed here
`meta_dataframe`	an optional dataframe containing the metadata rather than a file

savefile	an Rdata filename prefix for saving the data of the resulting expt.
low_files	whether or not to explicitly lowercase the filenames when searching in processed_data/
...	more parameters are fun

Details

It is worth noting that this function has a lot of logic used to find the count tables in the local filesystem. This logic has been superceded by simply adding a field to the .csv file called 'file'. create_expt() will then just read that filename, it may be a full pathname or local to the cwd of the project.

Value

experiment an expressionset

See Also

Biobase pData fData exprs hpgl_read_files as.list.hash

Examples

```
## Not run:
new_experiment = create_experiment("some_csv_file.csv", color_hash)
## Remember that this depends on an existing data structure of gene annotations.

## End(Not run)
```

---

deparse_go_value *Extract more easily readable information from a GOTERM datum.*

Description

The output from the GOTERM/GO.db functions is inconsistent, to put it nicely. This attempts to extract from that heterogeneous datatype something easily readable. Example: Synonym() might return any of the following: NA, NULL, "NA", "NULL", c("NA",NA,"GO:00001"), "GO:00002", c("Some text",NA, NULL, "GO:00003") This function will boil that down to 'not found', '', 'GO:00004', or "GO:0001, some text, GO:00004"

Usage

```
deparse_go_value(value)
```

Arguments

value	the result of try(as.character(somefunction(GOTERM[id])), silent=TRUE) somefunction would be 'Synonym' 'Secondary' 'Ontology', etc...

Value

something more sane (hopefully)

Examples

```
## Not run:
 goterms = GOTERM[ids]
 sane_goterms = deparse_go_value(goterms)

## End(Not run)
```

`deseq2_pairwise` *Set up a model matrix and set of contrasts to do a pairwise comparison of all conditions using DESeq2.*

Description

Set up a model matrix and set of contrasts to do a pairwise comparison of all conditions using DESeq2.

Usage

```
deseq2_pairwise(input, conditions = NULL, batches = NULL,
  model_cond = TRUE, model_batch = FALSE, annot_df = NULL, ...)
```

Arguments

`input`	A dataframe/vector or expt class containing data, normalization state, etc.
`conditions`	A factor of conditions in the experiment
`batches`	A factor of batches in the experiment
`model_cond`	Have condition in the experimental model?
`model_batch`	Have batch in the experimental model?
`annot_df`	Include some annotation information in the results?
`...`	triple dots!

Value

A list including the following information: run = the return from calling DESeq() denominators = list of denominators in the contrasts numerators = list of the numerators in the contrasts conditions = the list of conditions in the experiment coefficients = list of coefficients making the contrasts all_tables = list of DE tables

See Also

DESeq2 `results` `estimateSizeFactors` `estimateDispersions` `nbinomWaldTest`

Examples

```
## Not run:
pretend = deseq2_pairwise(data, conditions, batches)

## End(Not run)
```

`deseq_coefficient_scatter` *Plot out 2 coefficients with respect to one another from limma*

Description

It can be nice to see a plot of two coefficients from a limma comparison with respect to one another This hopefully makes that easy.

Usage

```
deseq_coefficient_scatter(output, x = 1, y = 2, gvis_filename = NULL,
  gvis_trendline = TRUE, tooltip_data = NULL, flip = FALSE,
  base_url = NULL)
```

Arguments

`output`	the set of pairwise comparisons provided by limma_pairwise()
`x`	the name or number of the first coefficient column to extract, this will be the x-axis of the plot
`y`	the name or number of the second coefficient column to extract, this will be the y-axis of the plot
`gvis_filename`	A filename for plotting gvis interactive graphs of the data.
`gvis_trendline`	add a trendline to the gvis plot?
`tooltip_data`	a dataframe of gene annotations to be used in the gvis plot
`flip`	flip the axes
`base_url`	for gvis plots

Value

a ggplot2 plot showing the relationship between the two coefficients

See Also

hpgl_linear_scatter limma_pairwise

Examples

```
## Not run:
 pretty = coefficient_scatter(limma_data, x="wt", y="mut")

## End(Not run)
```

deseq_pairwise *deseq_pairwise() Because I can't be trusted to remember '2'*

Description

This calls deseq2_pairwise(...) because I am determined to forget typing deseq2

Usage

```
deseq_pairwise(...)
```

Arguments

...	I like cats

Value

stuff from deseq2_pairwise

See Also

deseq2_pairwise

divide_seq *Express a data frame of counts as reads per pattern per million(library).*

Description

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

Usage

```
divide_seq(counts, pattern = "TA", fasta = "testme.fasta",
  gff = "testme.gff", entry_type = "gene")
```

Arguments

counts	read count matrix
pattern	pattern to search against. Defaults to 'TA'
fasta	a fasta genome to search
gff	the gff set of annotations to define start/ends of genes.
entry_type	which type of gff entry to search against. Defaults to 'gene'.

Value

The 'RPseqM' counts

See Also

FaFile rpkm

Examples

```
## Not run:
cptam <- divide_seq(cont_table, fasta="mgas_5005.fasta.xz", gff="mgas_5005.gff.xz")

## End(Not run)
```

---

edger_pairwise *Set up a model matrix and set of contrasts to do a pairwise comparison of all conditions using EdgeR.*

Description

Set up a model matrix and set of contrasts to do a pairwise comparison of all conditions using EdgeR.

Usage

```
edger_pairwise(input, conditions = NULL, batches = NULL,
  model_cond = TRUE, model_batch = FALSE, model_intercept = FALSE,
  alt_model = NULL, extra_contrasts = NULL, annot_df = NULL, ...)
```

Arguments

input	a dataframe/vector or expt class containing data, normalization state, etc.
conditions	a factor of conditions in the experiment
batches	a factor of batches in the experiment
model_cond	Include condition in the experimental model? This is pretty much always true.
model_batch	Include batch in the model? In most cases this is a good thing(tm).
model_intercept	Use cell means or intercept? (I default to the former, but they work out the same)
alt_model	An alternate experimental model to use
extra_contrasts	some extra contrasts to add to the list This can be pretty neat, lets say one has conditions A,B,C,D,E and wants to do (C/B)/A and (E/D)/A or (E/D)/(C/B) then use this with a string like: "c_vs_b_ctrla = (C-B)-A, e_vs_d_ctrla = (E-D)-A, de_vs_cb = (E-D)-(C-B),"
annot_df	Add some annotation information to the data tables?
...	The elipsis parameter is fed to write_edger() at the end.

Value

A list including the following information: contrasts = The string representation of the contrasts performed. lrt = A list of the results from calling glmLRT(), one for each contrast. contrast_list = The list of each call to makeContrasts() I do this to avoid running into the limit on # of contrasts addressable by topTags() all_tables = a list of tables for the contrasts performed.

See Also

edgeR topTags glmLRT make_pairwise_contrasts DGEList calcNormFactors estimateTagwiseDisp estimateCommonDisp estimateGLMCommonDisp estimateGLMTrendedDisp glmFit

Examples

```
## Not run:
 pretend = edger_pairwise(data, conditions, batches)

## End(Not run)
```

---

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 = NULL)
```

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

Biobase pData exprs fData

Examples

```
## Not run:
 smaller_expt = expt_subset(big_expt, "condition=='control'")
 all_expt = expt_subset(expressionset, "")  ## extracts everything

## End(Not run)
```

`extract_significant_genes` *Pull the highly up/down genes in combined tables*

Description

Given the output from combine_de_tables(), extract the fun genes.

Usage

```
extract_significant_genes(combined, according_to = "limma", fc = 1,
  p = 0.05, z = NULL, n = NULL,
  sig_table = "excel/significant_genes.xlsx")
```

Arguments

`combined`	the output from combine_de_tables()
`according_to`	one may use the deseq, edger, limma, or meta data.
`fc`	a log fold change to define 'significant'
`p`	a (adjusted)p-value to define 'significant'
`z`	a z-score to define 'significant'
`n`	a set of top/bottom-n genes
`sig_table`	an excel file to write

Value

a set of up-genes, down-genes, and numbers therein

See Also

`combine_de_tables`

`factor_rsquared` *Collect the r^2 values from a linear model fitting between a singular value decomposition and factor.*

Description

Collect the r^2 values from a linear model fitting between a singular value decomposition and factor.

Usage

```
factor_rsquared(svd_v, factor)
```

Arguments

`svd_v`	the V' V = I portion of a fast.svd call.
`factor`	an experimental factor from the original data.

Value

The r^2 values of the linear model as a percentage.

See Also

`fast.svd`

`gather_genes` *Given a set of goseq data from simple_goseq(), make a list of genes represented in each ontology.*

Description

This function uses the GO2ALLEG data structure to reverse map ontology categories to a list of genes represented. It therefore assumes that the GO2ALLEG.rda data structure has been deposited in pwd(). This in turn may be generated by clusterProfilers buildGOmap() function if it doesn't exist. For some species it may also be auto-generated. With little work this can be made much more generic, and it probably should.

Usage

```
gather_genes(goseq_data, ontology = "MF", pval = 0.05,
  include_all = FALSE)
```

Arguments

goseq_data	a list of goseq specific results as generated by simple_goseq()
ontology	an ontology to search
pval	a maximum accepted pvalue to include in the list of categories to cross reference.
include_all	include all genes in the ontology search

Value

a data frame of categories/genes.

See Also

simple_goseq buildGOmap,

Examples

```
## Not run:
 data = simple_goseq(de_genes=limma_output, lengths=annotation_df, goids=goids_df)
 genes_in_cats = gather_genes(data, ont='BP')

## End(Not run)
```

---

genefilter_cv_counts *Filter genes from a dataset outside a range of variance*

Description

Filter genes from a dataset outside a range of variance

Usage

```
genefilter_cv_counts(count_table, cv_min = 0.01, cv_max = 1000,
  verbose = FALSE)
```

Arguments

count_table	input data frame of counts by sample
cv_min	a minimum coefficient of variance
cv_max	guess
verbose	If set to true, prints number of genes removed / remaining

Value

dataframe of counts without the low-count genes

See Also

genefilter kOverA which this uses to decide what to keep

Examples

```
## Not run:
filtered_table = genefilter_kofa_counts(count_table)

## End(Not run)
```

---

genefilter_kofa_counts *Filter low-count genes from a data set using genefilter's kOverA()*

Description

Filter low-count genes from a data set using genefilter's kOverA()

Usage

```
genefilter_kofa_counts(count_table, k = 1, A = 1, verbose = FALSE)
```

Arguments

count_table	input data frame of counts by sample
k	a minimum number of samples to have >A counts
A	the minimum number of counts for each gene's sample in kOverA()
verbose	If set to true, prints number of genes removed / remaining

Value

dataframe of counts without the low-count genes

See Also

genefilter kOverA which this uses to decide what to keep

Examples

```
## Not run:
 filtered_table = genefilter_kofa_counts(count_table)

## End(Not run)
```

genefilter_pofa_counts *Filter low-count genes from a data set using genefilter's pOverA()*

Description

I keep thinking this function is pofa... oh well.

Usage

```
genefilter_pofa_counts(count_table, p = 0.01, A = 100, verbose = TRUE)
```

Arguments

count_table	input data frame of counts by sample
p	a minimum proportion of each gene's counts/sample to be greater than a minimum(A)
A	the minimum number of counts in the above proportion
verbose	If set to true, prints number of genes removed / remaining

Value

dataframe of counts without the low-count genes

See Also

genefilter pOverA which this uses to decide what to keep

Examples

```
## Not run:
 filtered_table = genefilter_pofa_counts(count_table)

## End(Not run)
```

getEdgeWeights *Plot the ontology DAG*

Description

This function was stolen from topgo in order to figure out where it was failing

Usage

```
getEdgeWeights(graph)
```

Arguments

graph	A graph from topGO

Value

weights

`get_genelengths` *Grab gene lengths from a gff file.*

Description

This function attempts to be robust to the differences in output from importing gff2/gff3 files. But it certainly isn't perfect.

Usage

```
get_genelengths(gff, type = "gene", key = "ID")
```

Arguments

gff	a gff file with (hopefully) IDs and widths
type	the annotation type to use.
key	the identifier in the 10th column of the gff file to use.

Value

a data frame of gene IDs and widths.

See Also

rtracklayer import.gff

Examples

```
## Not run:
 tt = get_genelengths('reference/fun.gff.gz')
 head(tt)
#          ID width
#1   YAL069W   312
#2   YAL069W   315
#3   YAL069W     3
#4 YAL068W-A   252
#5 YAL068W-A   255
#6 YAL068W-A     3

## End(Not run)
```

`get_sig_genes` *Get a set of up/down genes using the top/bottom n or >/< z scores away from the median.*

Description

Get a set of up/down genes using the top/bottom n or >/< z scores away from the median.

Usage

```
get_sig_genes(table, n = NULL, z = NULL, fc = NULL, p = NULL,
  column = "logFC", fold = "plusminus", p_column = "adj.P.Val")
```

Arguments

`table`	a table from limma/edger/deseq.
`n`	a rank-order top/bottom number of genes to take.
`z`	a number of z-scores >/< the median to take.
`fc`	a number of fold-changes to take
`p`	a p-value cutoff
`column`	a column to use to distinguish top/bottom
`fold`	an identifier reminding how to get the bottom portion of a fold-change (plusminus says to get the negative of the positive, otherwise 1/positive is taken).
`p_column`	a column containing (adjusted or not)p-values

Value

a list of up/down genes

`gff2df` *Try to make import.gff a little more robust I acquire (hopefully) valid gff3 files from various sources: yeastgenome.org, microbesonline, tritrypdb, ucsc, ncbi. To my eyes, they all look like reasonably good gff3 files, but some of them must be loaded with import.gff2, import.gff3, etc. That is super annoying. Also, I pretty much always just do as.data.frame() when I get something valid from rtracklayer, so this does that for me, I have another function which returns the iranges etc.*

Description

This function wraps import.gff/import.gff3/import.gff2 calls in try() Because sometimes those functions fail in unpredictable ways.

Usage

```
gff2df(gff, type = NULL)
```

Arguments

gff	a gff filename
type	subset the gff file for entries of a specific type

Value

a df!

See Also

rtracklayer import.gff import.gff2 import.gff3

Examples

```
## Not run:
funkytown <- gff2df('reference/gff/saccharomyces_cerevsiae.gff.xz')

## End(Not run)
```

---

gff2irange *Try to make import.gff a little more robust*

Description

Try to make import.gff a little more robust

Usage

```
gff2irange(gff, type = NULL)
```

Arguments

gff	a gff filename
type	a subset to extract Essentially gff2df() above, but returns data suitable for getSet()

Value

an iranges! (useful for getSeq())

See Also

rtracklayer gff2df getSeq

Examples

```
## Not run:
library(BSgenome.Tcruzi.clbrener.all)
tc_clb_all <- BSgenome.Tcruzi.clbrener.all
cds_ranges <- gff2irange('reference/gff/tcruzi_clbrener.gff.xz', type='CDS')
cds_sequences <- Biostrings::getSeq(tc_clb_all, cds_ranges)

## End(Not run)
```

---

godef *Get a go long-form definition from an id.*

Description

Get a go long-form definition from an id.

Usage

```
godef(go)
```

Arguments

go	a go ID, this may be a character or list (assuming the elements are goids).

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 godef("GO:0032432")
## > GO:0032432
## > "An assembly of actin filaments that are on the same axis but may be oriented with the same or opposite polari

## End(Not run)
```

golev *Get a go level approximation from an ID.*

Description

Get a go level approximation from an ID.

Usage

```
golev(go, verbose = FALSE)
```

Arguments

go	a go ID, this may be a character or list (assuming the elements are goids).
verbose	print some information as it recurses.

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 golev("GO:0032559")
 ## > 3

## End(Not run)
```

golevel *Get a go level approximation from a set of IDs. This just wraps golev() in mapply.*

Description

Get a go level approximation from a set of IDs. This just wraps golev() in mapply.

Usage

```
golevel(go)
```

Arguments

go	a character list of IDs.

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 golevel(c("GO:0032559", "GO:0000001")
## > 3 4

## End(Not run)
```

`golevel_df` *Extract a dataframe of golevels using getGOLevel() from clusterProfiler.*

Description

This function is way faster than my previous iterative golevel function. That is not to say it is very fast, so it saves the result to ontlevel.rda for future lookups.

Usage

```
golevel_df(ont = "MF", savefile = "ontlevel.rda")
```

Arguments

`ont`	the ontology to recurse.
`savefile`	a file to save the results for future lookups.

Value

golevels a dataframe of goids<->highest level

goont *Get a go ontology name from an ID.*

Description

Get a go ontology name from an ID.

Usage

```
goont(go)
```

Arguments

go	a go ID, this may be a character or list (assuming the elements are goids).

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 goont(c("GO:0032432", "GO:0032433"))
## > GO:0032432 GO:0032433
## > "CC" "CC"

## End(Not run)
```

gosec *Get a go secondary ID from an id*

Description

Unfortunately, GOTERM's returns for secondary IDs are not consistent, so this function has to have a whole bunch of logic to handle the various outputs.

Usage

```
gosec(go)
```

Arguments

go	A go ID – this may be a character or list(assuming the elements, not names, are goids)

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 gosec("GO:0032432")
## > GO:0032432
## > "GO:0000141" "GO:0030482"

## End(Not run)
```

`goseq_pval_plots` *Make a pvalue plot from goseq data*

Description

Make a pvalue plot from goseq data

Usage

```
goseq_pval_plots(goterms, wrapped_width = 20, cutoff = 0.1, n = 10,
  mincat = 10, level = NULL)
```

Arguments

`goterms`	some data from goseq!
`wrapped_width`	the number of characters before wrapping to help legibility
`cutoff`	pvalue cutoff for the plot
`n`	how many groups to include
`mincat`	minimum size of the category
`level`	levels of the ontology tree to use

Value

plots!

See Also

goseq **clusterProfiler** `pval_plot`

goseq_table *Enhance the goseq table of gene ontology information.*

Description

Enhance the goseq table of gene ontology information.

Usage

```
goseq_table(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. It requires a column 'category' which contains: GO:000001 and such.
file	a csv file to which to write the table

Value

the ontology table with annotation information included

See Also

goseq

Examples

```
## Not run:
 annotated_go = goseq_table(go_ids)
 head(annotated_go, n=1)
## >        category numDEInCat numInCat over_represented_pvalue
## > 571  GO:0006364          9       26            4.655108e-08
## >      under_represented_pvalue       qvalue ontology
## > 571                 1.0000000 6.731286e-05       BP
## >                               term
## > 571                 rRNA processing
## >                             synonym
## > 571        "35S primary transcript processing, GO:0006365"
## >        secondary    definition
## > 571   GO:0006365  Any process involved in the conversion of a primary ribosomal RNA (rRNA) transcript into c

## End(Not run)
```

goseq_trees *Make fun trees a la topgo from goseq data.*

Description

Make fun trees a la topgo from goseq data.

Usage

```
goseq_trees(de_genes, godata, goid_map = "reference/go/id2go.map",
  score_limit = 0.01, goids_df = NULL, overwrite = FALSE,
  selector = "topDiffGenes", pval_column = "adj.P.Val")
```

Arguments

de_genes	some differentially expressed genes
godata	data from goseq
goid_map	file to save go id mapping
score_limit	score limit for the coloring
goids_df	a mapping of IDs to GO in the Ramigo expected format
overwrite	overwrite the trees
selector	a function for choosing genes
pval_column	column to acquire pvalues

Value

a plot!

See Also

Ramigo

---

gostats_kegg *Use gostats() against kegg pathways*

Description

Does this even work? I don't think I have ever tested it yet. oh, it sort of does, maybe if I export it I will rembmer it

Usage

```
gostats_kegg()
```

`gostats_pval_plots` *Make a pvalue plot similar to that from clusterprofiler from gostats data*

Description

clusterprofiler provides beautiful plots describing significantly overrepresented categories. This function attempts to expand the repetoire of data available to them to include data from gostats.

Usage

```
gostats_pval_plots(gs_result, wrapped_width = 20, cutoff = 0.1, n = 12,
  group_minsize = 5)
```

Arguments

`gs_result`	ontology search results
`wrapped_width`	how big to make the text so that it is legible
`cutoff`	what is the maximum pvalue allowed
`n`	how many groups to include in the plot
`group_minsize`	default=5 minimum group size before inclusion

Details

The pval_plot function upon which this is based now has a bunch of new helpers now that I understand how the ontology trees work better, this should take advantage of that, but currently does not.

Value

plots!

See Also

clusterProfiler pval_plot

gostats_trees *Make fun trees a la topgo from goseq data.*

Description

Make fun trees a la topgo from goseq data.

Usage

```
gostats_trees(de_genes, mf_over, bp_over, cc_over, mf_under, bp_under, cc_under,
  goid_map = "reference/go/id2go.map", score_limit = 0.01,
  goids_df = NULL, overwrite = FALSE, selector = "topDiffGenes",
  pval_column = "adj.P.Val")
```

Arguments

de_genes	some differentially expressed genes
mf_over	mfover data
bp_over	bpover data
cc_over	ccover data
mf_under	mfunder data
bp_under	bpunder data
cc_under	ccunder expression data
goid_map	a mapping of IDs to GO in the Ramigo expected format
score_limit	maximum score to include as 'significant'
goids_df	a dataframe of available goids (used to generate goid_map)
overwrite	overwrite the goid_map?
selector	a function to choose differentially expressed genes in the data
pval_column	a column in the data to be used to extract pvalue scores

Value

plots! Trees! oh my!

See Also

topGO

gosyn *Get a go synonym from an ID.*

Description

I think I will need to do similar parsing of the output for this function as per gosec() In some cases this also returns stuff like c("some text", "GO:someID") versus "some other text" versus NULL versus NA

Usage

```
gosyn(go)
```

Arguments

go	a go ID, this may be a character or list(assuming the elements are goids).

Details

This function just goes a mapply(gosn, go).

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 text =  gosyn("GO:0000001")
 text
## > GO:000001
## > "mitochondrial inheritance"

## End(Not run)
```

goterm *Get a go term from ID.*

Description

Get a go term from ID.

Usage

```
goterm(go = "GO:0032559")
```

Arguments

go	a go ID or list thereof this may be a character or list(assuming the elements, not names, are goids)

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 goterm("GO:0032559")
## > GO:0032559
## > "adenyl ribonucleotide binding"

## End(Not run)
```

gotest *Test GO ids to see if they are useful. This just wraps gotst in mapply.*

Description

Test GO ids to see if they are useful. This just wraps gotst in mapply.

Usage

```
gotest(go)
```

Arguments

go	go IDs as characters.

Value

Some text

See Also

GOTermsAnnDbBimap

Examples

```
## Not run:
 gotest("GO:0032559")
## > 1
 gotest("GO:0923429034823904")
## > 0

## End(Not run)
```

---

graph_metrics *Make lots of graphs!*

Description

Plot out a set of metrics describing the state of an experiment including library sizes, # non-zero genes, heatmaps, boxplots, density plots, pca plots, standard median distance/correlation, and qq plots.

Usage

```
graph_metrics(expt, cormethod = "pearson", distmethod = "euclidean",
  title_suffix = NULL, qq = NULL, ma = NULL, ...)
```

Arguments

expt	an expt to process
cormethod	the correlation test for heatmaps.
distmethod	define the distance metric for heatmaps.
title_suffix	text to add to the titles of the plots.
qq	include qq plots
ma	include pairwise ma plots
...	extra parameters optionally fed to the various plots

Value

a loooong list of plots including the following: nonzero = a ggplot2 plot of the non-zero genes vs library size libsize = a ggplot2 bar plot of the library sizes boxplot = a ggplot2 boxplot of the raw data corheat = a recordPlot()ed pairwise correlation heatmap of the raw data smc = a recordPlot()ed view of the standard median pairwise correlation of the raw data disheat = a recordPlot()ed pairwise euclidean distance heatmap of the raw data smd = a recordPlot()ed view of the standard median pairwise distance of the raw data pcaplot = a recordPlot()ed PCA plot of the raw samples pcatable = a table describing the relative contribution of condition/batch of the raw data pcares = a table describing the relative contribution of condition/batch of the raw data pcavar = a table describing the variance of the raw data qq = a recordPlotted() view comparing the quantile/quantiles between the mean of all data and every raw sample density = a ggplot2 view of the density of each raw sample (this is complementary but more fun than a boxplot)

See Also

Biobase ggplot2 grDevices gplots exprs hpgl_norm hpgl_nonzero hpgl_libsize hpgl_boxplot hpgl_corheat hpgl_smc hpgl_disheat hpgl_smd hpgl_pca hpgl_qq_all hpgl_pairwise_ma

Examples

```
## Not run:
toomany_plots <- graph_metrics(expt)
toomany_plots$pcaplot
norm <- normalize_expt(expt, convert="cpm", batch=TRUE, filter_low=TRUE, transform="log2", norm="rle")
holy_asscrackers <- graph_metrics(norm, qq=TRUE, ma=TRUE)
## good luck, you are going to be waiting a while for the ma plots to print!

## End(Not run)
```

---

hpgl_arescore *Implement the arescan function in R*

Description

This function was taken almost verbatim from AREScore() in SeqTools Available at: https://github.com/lianos/seqtools.git At least on my computer I could not make that implementation work So I rewrapped its apply() calls and am now hoping to extend its logic a little to make it more sensitive and get rid of some of the spurious parameters or at least make them more transparent.

Usage

```
hpgl_arescore(x, basal = 1, overlapping = 1.5, d1.3 = 0.75, d4.6 = 0.4,
  d7.9 = 0.2, within.AU = 0.3, aub.min.length = 10,
  aub.p.to.start = 0.8, aub.p.to.end = 0.55)
```

Arguments

x	A DNA/RNA StringSet containing the UTR sequences of interest
basal	I dunno.
overlapping	default=1.5
d1.3	default=0.75 These parameter names are so stupid, lets be realistic
d4.6	default=0.4
d7.9	default=0.2
within.AU	default=0.3
aub.min.length	default=10
aub.p.to.start	default=0.8
aub.p.to.end	default=0.55

Details

Note that I did this two months ago and haven't touched it since...

Value

a DataFrame of scores

See Also

IRanges Biostrings

Examples

```
## Not run:
## Extract all the genes from my genome, pull a static region 120nt following the stop
## and test them for potential ARE sequences.
## FIXME: There may be an error in this example, another version I have handles the +/- strand
## genes separately, I need to return to this and check if it is providing the 5' UTR for 1/2
## the genome, which would be unfortunate -- but the logic for testing remains the same.
are_candidates <- hpgl_arescore(genome)
utr_genes <- subset(lmajor_annotations, type == 'gene')
threep <- GenomicRanges::GRanges(seqnames=Rle(utr_genes[,1]),
                         ranges=IRanges(utr_genes[,3], end=(utr_genes[,3] + 120)), strand=Rle(utr_genes[,5]),
                               name=Rle(utr_genes[,10]))
threep_seqstrings <- Biostrings::getSeq(lm, threep)
are_test <- hpgltools:::hpgl_arescore(x=threep_seqstrings)
are_genes <- rownames(are_test[ which(are_test$score > 0), ])

## End(Not run)
```

`hpgl_bcv_plot` *Steal edgeR's plotBCV() and make it a ggplot2 This was written primarily to understand what that function is doing in edgeR.*

Description

Steal edgeR's plotBCV() and make it a ggplot2 This was written primarily to understand what that function is doing in edgeR.

Usage

```
hpgl_bcv_plot(data)
```

Arguments

`data`	A dataframe/expt/exprs with count data

Value

a plot! of the BCV a la ggplot2.

See Also

edgeR plotBCV

Examples

```
## Not run:
bcv <- hpgl_bcv_plot(expt)
summary(bcv$data)
bcv$plot

## End(Not run)
```

`hpgl_boxplot` *Make a ggplot boxplot of a set of samples.*

Description

Make a ggplot boxplot of a set of samples.

Usage

```
hpgl_boxplot(data, colors = NULL, names = NULL, title = NULL,
  scale = NULL, ...)
```

Arguments

data	an expt or data frame set of samples.
colors	a color scheme, if not provided will make its own.
names	a nicer version of the sample names.
title	A title!
scale	whether to log scale the y-axis.
...	more parameters are fun

Value

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

ggplot2 reshape2 geom_boxplot melt scale_x_discrete

Examples

```
## Not run:
 a_boxplot <- hpgl_boxplot(expt)
 a_boxplot  ## ooo pretty boxplot look at the lines

## End(Not run)
```

---

hpgl_combatMod *Use a modified version of combat on some data This is a hack of Kwame's combatMod to make it not fail on corner-cases.*

Description

Use a modified version of combat on some data This is a hack of Kwame's combatMod to make it not fail on corner-cases.

Usage

```
hpgl_combatMod(dat, batch, mod, noScale = TRUE, prior.plots = FALSE)
```

Arguments

dat	a df to modify
batch	a factor of batches
mod	a factor of conditions
noScale	the normal 'scale' option squishes the data too much, so this defaults to TRUE
prior.plots	print out prior plots? FALSE

Value

a df of batch corrected data

See Also

sva ComBat

Examples

```
## Not run:
df_new = hpgl_combatMod(df, batches, model)

## End(Not run)
```

---

hpgl_cor *Wrap cor() to include robust correlations.*

Description

Wrap cor() to include robust correlations.

Usage

```
hpgl_cor(df, method = "pearson", ...)
```

Arguments

df	a data frame to test.
method	correlation method to use. Includes pearson, spearman, kendal, robust.
...	other options to pass to stats::cor()

Value

correlation some fun correlation statistics

See Also

robust cor cov covRob

Examples

```
## Not run:
hpgl_cor(df=df)
hpgl_cor(df=df, method="robust")

## End(Not run)
```

hpgl_corheat *Make a heatmap.3 description of the correlation between samples.*

Description

Make a heatmap.3 description of the correlation between samples.

Usage

```
hpgl_corheat(data, colors = NULL, design = NULL, method = "pearson",
  names = NULL, row = "batch", title = NULL, ...)
```

Arguments

data	a dataframe, expt, or expressionset to work with.
colors	a color scheme.
design	a design matrix.
method	correlation statistic to use.
names	alternate names to use.
row	what to place on the row of the map, batches or conditions?
title	a title for the plot.
...	more options are wonderful

Value

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

See Also

hpgl_cor brewer.pal recordPlot

Examples

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

hpgl_density *Density plots!*

Description

Density plots!

Usage

```
hpgl_density(data, colors = NULL, names = NULL, position = "identity",
  fill = NULL, title = NULL, scale = NULL)
```

Arguments

data	an expt, expressionset, or data frame.
colors	a color scheme to use.
names	names of the samples.
position	how to place the lines, either let them overlap (identity), or stack them.
fill	fill the distributions? This might make the plot unreasonably colorful.
title	a title for the plot.
scale	plot on the log scale?

Value

a density plot!

See Also

ggplot2 geom_density

Examples

```
## Not run:
funkytown <- hpgl_density(data)

## End(Not run)
```

`hpgl_disheat` *Make a heatmap.3 description of the similarity (euclildean distance) between samples.*

Description

Make a heatmap.3 description of the similarity (euclildean distance) between samples.

Usage

```
hpgl_disheat(data, colors = NULL, design = NULL, method = "euclidean",
  names = NULL, row = "batch", title = NULL, ...)
```

Arguments

`data`	a dataframe, expt, or expressionset to work with.
`colors`	a color scheme.
`design`	a design matrix.
`method`	distance metric to use.
`names`	alternate names to use.
`row`	what to place on the row of the map, batches or conditions?
`title`	a title for the plot.
`...`	more parameters

Value

a recordPlot() heatmap describing the distance between samples.

See Also

brewer.pal heatmap.2 recordPlot

Examples

```
## Not run:
 disheat_plot = hpgl_disheat(expt=expt, method="euclidean")
 disheat_plot

## End(Not run)
```

`hpgl_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

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.

Usage

```
hpgl_dist_scatter(df, tooltip_data = NULL, gvis_filename = NULL, size = 2)
```

Arguments

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

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.

See Also

ggplot2 hpgl_gvis_scatter geom_point hpgl_linear_scatter

Examples

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

hpgl_enrich.internal *A minor hack in the clusterProfiler function 'enrich.internal'*

Description

A minor hack in the clusterProfiler function 'enrich.internal'

Usage

```
hpgl_enrich.internal(gene, organism, pvalueCutoff = 1, pAdjustMethod = "BH",
  ont, minGSSize = 2, qvalueCutoff = 0.2, readable = FALSE,
  universe = NULL)
```

Arguments

gene	some differentially expressed genes
organism	again will cause this to pull ensG if recognized by clusteprofiler
pvalueCutoff	a pvalue cutoff
pAdjustMethod	p adjust method
ont	mf bp or cc
minGSSize	a minimum gs size
qvalueCutoff	maximum q value
readable	set the readable flag for dose
universe	a universe to use

Value

some clusterProfiler data

See Also

clusterProfiler

hpgl_enrichGO *A minor hack in the clusterProfiler function 'enrichGO'*

Description

A minor hack in the clusterProfiler function 'enrichGO'

Usage

```
hpgl_enrichGO(gene, organism = "human", ont = "MF", pvalueCutoff = 0.05,
  pAdjustMethod = "BH", universe, qvalueCutoff = 0.2, minGSSize = 2,
  readable = FALSE)
```

Arguments

gene	some differentially expressed genes
organism	if used will cause this to pull the ensG annotations
ont	mf bp or cc
pvalueCutoff	pvalue cutoff
pAdjustMethod	p-value adjustment
universe	the gene universe
qvalueCutoff	maximum qvalue before adding
minGSSize	smallest group size
readable	readable tag on the object

Value

some clusterProfiler data

See Also

clusterProfiler

---

hpgl_Gff2GeneTable *A copy and paste of clusterProfiler's readGff*

Description

A copy and paste of clusterProfiler's readGff

Usage

```
hpgl_Gff2GeneTable(gffFile, compress = TRUE, split = "=")
```

Arguments

gffFile	a gff file
compress	compress them
split	the splitter when reading gff files

hpgl_GOplot	*hpgl_GOplot() A minor hack of the topGO GOplot function This allows me to change the line widths from the default.*

Description

hpgl_GOplot() A minor hack of the topGO GOplot function This allows me to change the line widths from the default.

Usage

```
hpgl_GOplot(dag, sigNodes, dag.name = "GO terms", edgeTypes = TRUE,
  nodeShape.type = c("box", "circle", "ellipse", "plaintext")[3],
  genNodes = NULL, wantedNodes = NULL, showEdges = TRUE,
  useFullNames = TRUE, oldSigNodes = NULL, nodeInfo = NULL,
  maxchars = 30)
```

Arguments

dag	The DAG tree of ontologies
sigNodes	The set of significant ontologies (with p-values)
dag.name	A name for the graph
edgeTypes	Set the types of the edges for graphviz
nodeShape.type	The shapes on the tree
genNodes	Generate the nodes?
wantedNodes	A subset of the ontologies to plot
showEdges	Show the arrows?
useFullNames	Full names of the ontologies (they can get long)
oldSigNodes	I dunno
nodeInfo	Hmm
maxchars	Maximum characters per line inside the shapes

Value

a topgo plot

hpgl_GroupDensity *A hack of topGO's groupDensity()*

Description

This just adds a couple wrappers to avoid errors in groupDensity.

Usage

```
hpgl_GroupDensity(object, whichGO, ranks = TRUE, rm.one = FALSE)
```

Arguments

object	a topGO enrichment object
whichGO	an individual ontology group to compare with
ranks	rank order the set of ontologies
rm.one	remove pvalue=1 groups

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

Description

Make an html version of an MA plot.

Usage

```
hpgl_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/hpgl_voom().
degenes	df from toptable or its friends containing p-values.
tooltip_data	a df of tooltip information.
filename	a filename to write a fancy html graph.
base_url	a string with a basename used for generating URLs for clicking dots on the graph.
...	more options are more options

Value

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

See Also

hpgl_ma_plot

Examples

```
## Not run:
 hpgl_gvis_ma_plot(voomed_data, toptable_data, filename="html/fun_ma_plot.html", base_url="http://yeastgenome.o

## End(Not run)
```

`hpgl_gvis_scatter` *Make an html version of a scatter plot.*

Description

Make an html version of a scatter plot.

Usage

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

Arguments

`df`	df of two columns to compare
`tooltip_data`	a df of tooltip information for gvis graphs.
`filename`	a filename to write a fancy html graph.
`base_url`	a url to send click events which will be suffixed with the gene name
`trendline`	add a trendline?

Value

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

See Also

gvisScatterChart

Examples

```
## Not run:
 hpgl_gvis_scatter(a_dataframe_twocolumns, filename="html/fun_scatter_plot.html", base_url="http://yeastgenome.

## End(Not run)
```

`hpgl_gvis_volcano_plot` *Make an html version of an volcano plot.*

Description

Make an html version of an volcano plot.

Usage

```
hpgl_gvis_volcano_plot(toptable_data, fc_cutoff = 0.8, p_cutoff = 0.05,
  tooltip_data = NULL, filename = "html/gvis_vol_plot.html",
  base_url = "", ...)
```

Arguments

`toptable_data`	df of toptable() data
`fc_cutoff`	fold change cutoff.
`p_cutoff`	maximum p value to allow.
`tooltip_data`	a df of tooltip information.
`filename`	a filename to write a fancy html graph.
`base_url`	a string with a basename used for generating URLs for clicking dots on the graph.
`...`	more options

Value

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

See Also

hpgl_volcano_plot

Examples

```
## Not run:
 hpgl_gvis_ma_plot(voomed_data, toptable_data, filename="html/fun_ma_plot.html", base_url="http://yeastgenome.o

## End(Not run)
```

`hpgl_heatmap` *Make a heatmap.3 plots, does the work for hpgl_disheat and hpgl_corheat.*

Description

Make a heatmap.3 plots, does the work for hpgl_disheat and hpgl_corheat.

Usage

```
hpgl_heatmap(data, colors = NULL, design = NULL, method = "pearson",
  names = NULL, type = "correlation", row = "batch", title = NULL, ...)
```

Arguments

`data`	a dataframe, expt, or expressionset to work with.
`colors`	a color scheme.
`design`	a design matrix.
`method`	distance or correlation metric to use.
`names`	alternate names to use.
`type`	correlation or distance or sample
`row`	what to place on the row of the map, batches or conditions?
`title`	a title for the plot.
`...`	I like elipses

Value

a recordPlot() heatmap describing the distance between samples.

See Also

brewer.pal recordPlot

`hpgl_histogram` *Make a pretty histogram of something.*

Description

Make a pretty histogram of something.

Usage

```
hpgl_histogram(df, binwidth = NULL, log = FALSE, bins = 500,
  verbose = FALSE, fillcolor = "darkgrey", color = "black")
```

Arguments

df	a dataframe of lots of pretty numbers.
binwidth	width of the bins for the histogram.
log	replot on the log scale?
bins	bins for the histogram
verbose	be verbose?
fillcolor	change the fill colors of the plotted elements.
color	change the color of the lines of the plotted elements.

Value

a ggplot histogram

See Also

geom_histogram geom_density

Examples

```
## Not run:
 kittytime = hpgl_histogram(df)

## End(Not run)
```

---

hpgl_libsize *Make a ggplot graph of library sizes.*

Description

Make a ggplot graph of library sizes.

Usage

```
hpgl_libsize(data, colors = NULL, names = NULL, text = TRUE,
  title = NULL, yscale = NULL, ...)
```

Arguments

data	an expt, dataframe, or expressionset of samples.
colors	a color scheme.
names	alternate names for the x-axis.
text	add the numeric values inside the top of the bars of the plot?
title	a title for the plot.
yscale	whether or not to log10 the y-axis.
...	more parameters for your good time

Value

a ggplot2 bar plot of every sample's size

See Also

geom_bar geom_text prettyNum scale_y_log10

Examples

```
## Not run:
 libsize_plot = hpgl_libsize(expt=expt)
 libsize_plot  ## ooo pretty bargraph

## End(Not run)
```

---

hpgl_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

```
hpgl_linear_scatter(df, tooltip_data = NULL, gvis_filename = NULL,
  cormethod = "pearson", size = 2, verbose = FALSE, loess = FALSE,
  identity = FALSE, gvis_trendline = NULL, first = NULL, second = NULL,
  base_url = NULL, pretty_colors = TRUE)
```

Arguments

df	a dataframe likely containing two columns
tooltip_data	a df of tooltip information for gvis graphs.
gvis_filename	a filename to write a fancy html graph.
cormethod	what type of correlation to check?
size	size of the dots on the plot.
verbose	be verbose?
loess	add a loess estimation?
identity	add the identity line?
gvis_trendline	add a trendline to the gvis plot? There are a couple possible types, I think linear is the most common.
first	first column to plot
second	second column to plot
base_url	a base url to add to the plot
pretty_colors	colors!

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 hpgl_histogram

Examples

```
## Not run:
 hpgl_linear_scatter(lotsofnumbers_intwo_columns, tooltip_data=tooltip_dataframe, gvis_filename="html/fun_scatt

## End(Not run)
```

---

hpgl_log2cpm *Converts count matrix to log2 counts-per-million reads.*

Description

Based on the method used by limma as described in the Law et al. (2014) voom paper.

Usage

```
hpgl_log2cpm(counts, lib.size = NULL)
```

Arguments

counts	read count matrix
lib.size	library size

Value

log2-CPM read count matrix

See Also

cbcbSEQ edgeR

Examples

```
## Not run:
l2cpm <- hpgl_log2cpm(counts)

## End(Not run)
```

`hpgl_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

```
hpgl_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/hpgl_voom().
`de_genes`	df from toptable or its friends containing p-values.
`adjpval_cutoff`	a cutoff defining significant from not.
`alpha`	how transparent to make the dots.
`size`	how big are the dots?
`tooltip_data`	a df of tooltip information for gvis graphs.
`gvis_filename`	a filename to write a fancy html graph.
`...`	more poptions por pou

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

hpgl_gvis_ma_plot toptable voom voomMod hpgl_voom lmFit makeContrasts contrasts.fit

Examples

```
## Not run:
hpgl_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.

## End(Not run)
```

`hpgl_multihistogram` *Make a pretty histogram of multiple datasets.*

Description

Make a pretty histogram of multiple datasets.

Usage

```
hpgl_multihistogram(data, log = FALSE, binwidth = NULL, bins = NULL,
  verbose = FALSE)
```

Arguments

`data`	a dataframe of lots of pretty numbers, this also accepts lists.
`log`	plot the data on the log scale?
`binwidth`	set a static bin width with an unknown # of bins? If neither of these are provided, then bins is set to 500, if both are provided, then bins wins.
`bins`	set a static # of bins of an unknown width?
`verbose`	be verbose?

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

Examples

```
## Not run:
 kittytime = hpgl_multihistogram(df)

## End(Not run)
```

`hpgl_multiplot` *Make a grid of plots.*

Description

Make a grid of plots.

Usage

```
hpgl_multiplot(plots, file, cols = NULL, layout = NULL)
```

Arguments

`plots`	a list of plots
`file`	a file to write to
`cols`	the number of columns in the grid
`layout`	set the layout specifically

Value

a multiplot!

`hpgl_nonzero` *Make a ggplot graph of the number of non-zero genes by sample. Made by Ramzi Temanni <temanni at umd dot edu>*

Description

Make a ggplot graph of the number of non-zero genes by sample. Made by Ramzi Temanni <temanni at umd dot edu>

Usage

```
hpgl_nonzero(data, design = NULL, colors = NULL, labels = NULL,
  title = NULL, ...)
```

Arguments

`data`	an expt, expressionset, or dataframe.
`design`	a design matrix.
`colors`	a color scheme.
`labels`	how do you want to label the graph? 'fancy' will use directlabels() to try to match the labels with the positions without overlapping anything else will just stick them on a 45' offset next to the graphed point
`title`	add a title?
`...`	rawr

Value

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

See Also

geom_point geom_dl

Examples

```
## Not run:
 nonzero_plot = hpgl_nonzero(expt=expt)
 nonzero_plot  ## ooo pretty

## End(Not run)
```

---

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

Description

Sometime soon I am going to elipsis all these variables

Usage

```
hpgl_norm(data, design = NULL, transform = "raw", norm = "raw",
  convert = "raw", batch = "raw", batch1 = "batch", batch2 = NULL,
  filter_low = FALSE, annotations = NULL, entry_type = "gene",
  fasta = NULL, verbose = FALSE, thresh = 2, min_samples = 2,
  noscale = TRUE, p = 0.01, A = 1, k = 1, cv_min = 0.01,
  cv_max = 1000, ...)
```

Arguments

data	some data
design	design dataframe must come with it
transform	defines whether to log(2|10) transform the data. Defaults to raw.
norm	specify the normalization strategy. Defaults to raw. This makes use of DESeq/EdgeR to provide: RLE, upperquartile, size-factor, or tmm normalization. I tend to like quantile, but there are definitely corner-case scenarios for all strategies.
convert	defines the output type which may be raw, cpm, rpkm, or cp_seq_m. Defaults to raw.
batch	batch correction method to try out
batch1	column from design to get batch info
batch2	a second covariate to try

filter_low	choose whether to low-count filter the data.
annotations	is used for rpkm or sequence normalizations to extract the lengths of sequences for normalization
entry_type	default gff entry to cull from
fasta	fasta genome for rpkm
verbose	talk
thresh	threshold for low count filtering
min_samples	minimum samples for low count filtering
noscale	used by combatmod
p	for povera genefilter
A	for povera genefilter
k	for kovera genefilter
cv_min	for genefilter cv
cv_max	for genefilter cv
...	I should put all those other options here

Value

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

See Also

cpm rpkm hpgl_rpkm filterCounts DESeqDataSetFromMatrix estimateSizeFactors DGEList calcNormFactors

Examples

```
## Not run:
df_raw = hpgl_norm(expt=expt)  ## Only performs low-count filtering
df_raw = hpgl_norm(df=a_df, design=a_design) ## Same, but using a df
df_ql2rpkm = hpgl_norm(expt=expt, norm='quant', transform='log2', convert='rpkm')  ## Quantile, log2, rpkm
count_table = df_ql2rpkm$counts

## End(Not run)
```

---

hpgl_pairwise_ma *Plot all pairwise MA plots in an experiment.*

Description

Use affy's ma.plot() on every pair of columns in a data set to help diagnose problematic samples.

Usage

```
hpgl_pairwise_ma(data, log = NULL, ...)
```

Arguments

data	an expt expressionset or data frame
log	is the data in log format?
...	more options are good

Value

a list of affy::maplots

See Also

ma.plot

Examples

```
## Not run:
 ma_plots = hpgl_pairwise_ma(expt=some_expt)

## End(Not run)
```

---

hpgl_pathview *Print some data onto KEGG pathways*

Description

Print some data onto KEGG pathways

Usage

```
hpgl_pathview(path_data, indir = "pathview_in", outdir = "pathview",
  pathway = "all", species = "lma", string_from = "LmjF",
  string_to = "LMJF", suffix = "_colored", second_from = NULL,
  second_to = NULL, verbose = FALSE, filenames = "id")
```

Arguments

path_data	some differentially expressed genes
indir	A directory into which the unmodified kegg images will be downloaded (or already exist).
outdir	A directory which will contain the colored images.
pathway	Perform the coloring for a specific pathway?
species	The kegg identifier for the species of interest.
string_from	for renaming kegg categories
string_to	for renaming kegg categories
suffix	for renaming finished files

second_from	sometimes jsut one regex isnt enough
second_to	sometimes just one regex isnt enough
verbose	talk more
filenames	name the final files by id or name?

Value

A list of some information for every KEGG pathway downloaded/examined. This information includes: a. The filename of the final image for each pathway. b. The number of genes which were found in each pathway image. c. The number of genes in the 'up' category d. The number of genes in the 'down' category

See Also

Ramigo pathview

Examples

```
## Not run:
 thy_el_comp2_path = hpgl_pathview(thy_el_comp2_kegg, species="spz", indir="pathview_in", outdir="kegg_thy_el_co

## End(Not run)
```

---

hpgl_pca *Make a ggplot PCA plot describing the samples' clustering.*

Description

Make a ggplot PCA plot describing the samples' clustering.

Usage

```
hpgl_pca(data, design = NULL, plot_colors = NULL, plot_labels = NULL,
  plot_title = NULL, plot_size = 5, ...)
```

Arguments

data	an expt set of samples.
design	a design matrix and.
plot_colors	a color scheme.
plot_labels	add labels? Also, what type? FALSE, "default", or "fancy".
plot_title	a title for the plot.
plot_size	size for the glyphs on the plot.
...	arglist from elipsis!

Value

a list containing the following: pca = the result of fast.svd() plot = ggplot2 pca_plot describing the principle component analysis of the samples. table = a table of the PCA plot data res = a table of the PCA res data variance = a table of the PCA plot variance This makes use of cbcbSEQ and prints the table of variance by component.

See Also

makeSVD, pcRes, geom_dl pca_plot_smallbatch pca_plot_largebatch

Examples

```
## Not run:
 pca_plot = hpgl_pca(expt=expt)
 pca_plot

## End(Not run)
```

---

hpgl_qq_all *quantile/quantile comparison of all samples (in this case the mean of all samples, and each sample)*

Description

quantile/quantile comparison of all samples (in this case the mean of all samples, and each sample)

Usage

```
hpgl_qq_all(data, verbose = FALSE, labels = "short")
```

Arguments

data	an expressionset, expt, or dataframe of samples.
verbose	be chatty while running?
labels	what kind of labels to print?

Value

a list containing: logs = a recordPlot() of the pairwise log qq plots ratios = a recordPlot() of the pairwise ratio qq plots means = a table of the median values of all the summaries of the qq plots

hpgl_qq_all_pairwise *Perform qq plots of every column against every other column of a dataset. This function is stupid, don't use it.*

Description

Perform qq plots of every column against every other column of a dataset. This function is stupid, don't use it.

Usage

```
hpgl_qq_all_pairwise(data, verbose = FALSE)
```

Arguments

data	the data
verbose	talky talky

Value

a list containing the recordPlot() output of the ratios, logs, and means among samples

hpgl_qq_plot *Perform a qqplot between two columns of a matrix.*

Description

Perform a qqplot between two columns of a matrix.

Usage

```
hpgl_qq_plot(data, x = 1, y = 2, labels = TRUE)
```

Arguments

data	data frame/expt/expressionset.
x	the first column.
y	the second column.
labels	include the lables?

Value

a list of the logs, ratios, and mean between the plots as ggplots.

hpgl_qshrink *A hacked copy of Kwame's qsmooth/qstats code*

Description

I made a couple small changes to Kwame's qstats() function to make it not fail when on corner-cases. I sent him a diff, but haven't checked to see if it was useful yet.

Usage

```
hpgl_qshrink(data = NULL, groups = NULL, refType = "mean",
  groupLoc = "mean", window = 99, verbose = FALSE, groupCol = NULL,
  plot = TRUE, ...)
```

Arguments

data	count table to modify
groups	factor of the experimental conditions
refType	method for grouping conditions
groupLoc	method for grouping groups
window	a window, for looking!
verbose	talky talky
groupCol	column to define conditions
plot	plot the quantiles?
...	more options

Value

data a new data frame of normalized counts

See Also

qsmooth

Examples

```
## Not run:
df <- hpgl_qshrink(data)

## End(Not run)
```

hpgl_qstats *A hacked copy of Kwame's qsmooth/qstats code*

Description

I made a couple small changes to Kwame's qstats() function to make it not fail when on corner-cases. I sent him a diff, but haven't checked to see if it was useful yet.

Usage

```
hpgl_qstats(data, groups, refType = "mean", groupLoc = "mean",
  window = 99)
```

Arguments

data	the initial count data
groups	the experimental conditions as a factor
refType	(or median) the method to separate groups
groupLoc	I don't remember
window	window for basking

Value

new data

Examples

```
## Not run:
qstatted <- hpgl_qstats(data, conditions)

## End(Not run)
```

hpgl_read_files *Read a bunch of count tables and create a usable data frame from them. It is worth noting that this function has some logic intended for the elsayed lab's data storage structure. It shouldn't interfere with other usages, but it attempts to take into account different ways the data might be stored.*

Description

Read a bunch of count tables and create a usable data frame from them. It is worth noting that this function has some logic intended for the elsayed lab's data storage structure. It shouldn't interfere with other usages, but it attempts to take into account different ways the data might be stored.

Usage

```
hpgl_read_files(ids, files, header = FALSE, include_summary_rows = FALSE,
  suffix = NULL, ...)
```

Arguments

ids	a list of experimental ids
files	a list of files to read
header	whether or not the count tables include a header row.
include_summary_rows	whether HTSeq summary rows should be included.
suffix	an optional suffix to add to the filenames when reading them.
...	more options for happy time

Value

count_table a data frame of count tables

See Also

create_experiment

Examples

```
## Not run:
 count_tables = hpgl_read_files(as.character(sample_ids), as.character(count_filenames))

## End(Not run)
```

---

hpgl_rpkm *Reads/(kilobase(gene) * million reads)*

Description

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

Usage

```
hpgl_rpkm(df, annotations = get0("gene_annotations"))
```

Arguments

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

Details

This function wraps EdgeR's rpkm in an attempt to make sure that the required gene lengths get sent along.

Value

rpkm_df a data frame of counts expressed as rpkm

See Also

edgeR and cpm rpkm

Examples

```
## Not run:
rpkm_df = hpgl_rpkm(df, annotations=gene_annotations)

## End(Not run)
```

---

hpgl_sample_heatmap *Make a heatmap.3 description of the similarity of the genes among samples.*

Description

Make a heatmap.3 description of the similarity of the genes among samples.

Usage

```
hpgl_sample_heatmap(data, colors = NULL, design = NULL, names = NULL,
  title = NULL, Rowv = FALSE, ...)
```

Arguments

data	an expt/expressionset/dataframe set of samples
colors	a color scheme
design	a design matrix
names	add names?
title	title of the plot.
Rowv	include the row names
...	more parameters for a good time

Value

a recordPlot() heatmap describing the samples.

See Also

brewer.pal recordPlot

`hpgl_scatter` *Make a pretty scatter plot between two sets of numbers.*

Description

Make a pretty scatter plot between two sets of numbers.

Usage

```
hpgl_scatter(df, tooltip_data = NULL, color = "black",
  gvis_filename = NULL, size = 2)
```

Arguments

`df`	a dataframe likely containing two columns
`tooltip_data`	a df of tooltip information for gvis
`color`	color of the dots on the graph.
`gvis_filename`	a filename to write a fancy html graph.
`size`	the size of the dots on the graph.

Value

a ggplot2 scatter plot.

See Also

hpgl_gvis_scatter geom_point hpgl_linear_scatter

Examples

```
## hpgl_scatter(lotsofnumbers_intwo_columns, tooltip_data=tooltip_dataframe, gvis_filename="html/fun_scatterplo
```

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

Description

This was written by a mix of Kwame Okrah <kokrah at gmail dot com>, Laura Dillon <dillonl at umd dot edu>, and Hector Corrada Bravo <hcorrada at umd dot edu>

Usage

```
hpgl_smc(data, colors = NULL, method = "pearson", names = NULL,
  title = NULL, ...)
```

Arguments

data	an expt, expressionset, or data frame.
colors	a color scheme
method	a correlation method to use.
names	use pretty names for the samples?
title	title for the graph.
...	more parameters to make you happy

Value

a recordPlot() of the standard median pairwise correlation among the samples. 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

hpgl_cor rowMedians quantile diff recordPlot

Examples

```
## Not run:
 smc_plot = hpgl_smc(expt=expt)

## End(Not run)
```

hpgl_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

```
hpgl_smd(data, colors = NULL, names = NULL, method = "euclidean",
  title = NULL, ...)
```

Arguments

data	an expt/expressionset/data frame of samples.
colors	a color scheme
names	use pretty names for the samples?
method	a distance metric to use.
title	title for the graph.
...	parameters make me happy

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

```
## Not run:
 smd_plot = hpgl_smd(expt=expt)

## End(Not run)
```

`hpgl_volcano_plot` *Make a pretty Volcano plot!*

Description

Make a pretty Volcano plot!

Usage

```
hpgl_volcano_plot(toptable_data, tooltip_data = NULL, gvis_filename = NULL,
  fc_cutoff = 0.8, p_cutoff = 0.05, size = 2, alpha = 0.6, ...)
```

Arguments

`toptable_data`	a dataframe from limma's toptable which includes log(fold change) and an adjusted p-value.
`tooltip_data`	a df of tooltip information for gvis.
`gvis_filename`	a filename to write a fancy html graph.
`fc_cutoff`	a cutoff defining the minimum/maximum fold change for interesting. This is log, so I went with +/- 0.8 mostly arbitrarily as the default.
`p_cutoff`	a cutoff defining significant from not.
`size`	how big are the dots?
`alpha`	how transparent to make the dots.
`...`	I love parameters!

Value

a ggplot2 MA scatter plot. This is defined as the -log10(p-value) with respect to log(fold change). The cutoff values are delineated with lines and mark the boundaries between 'significant' and not. This will make a fun clicky googleVis graph if requested.

See Also

hpgl_gvis_ma_plot toptable voom hpgl_voom lmFit makeContrasts contrasts.fit

Examples

```
## Not run:
 hpgl_volcano_plot(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.

## End(Not run)
```

hpgl_voom	*A slight modification of limma's voom() function. 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

A slight modification of limma's voom() function. 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

```
hpgl_voom(dataframe, model = NULL, libsize = NULL, stupid = FALSE,
  logged = FALSE, converted = FALSE)
```

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).
stupid	whether or not to cheat when the resulting matrix is not solvable.
logged	whether the input data is known to be logged.
converted	whether the input data is known to be cpm converted.

Value

an EList containing the following information: E = The normalized data weights = The weights of said data design = The resulting design lib.size = The size in pseudocounts of the library plot = A ggplot of the mean/variance trend with a blue loess fit and red trend fit

See Also

voom voomMod lmFit

Examples

```
## Not run:
 funkytown = hpgl_voom(samples, model)

## End(Not run)
```

---

kegg_get_orgn *Search the kegg identifier for a given species*

Description

Search the kegg identifier for a given species

Usage

```
kegg_get_orgn(species = "Leishmania", short = TRUE)
```

Arguments

species	A search string (Something like 'Homo sapiens')
short	only pull the orgid

Value

a data frame of possible KEGG identifier codes, genome ID numbers, species, and phylogenetic classifications.

See Also

RCurl

Examples

```
## Not run:
 fun = kegg_get_orgn('Canis')
## >     Tid     orgid      species                   phylogeny
## >  17 T01007   cfa Canis familiaris (dog) Eukaryotes;Animals;Vertebrates;Mammals

## End(Not run)
```

`limma_coefficient_scatter` *Plot out 2 coefficients with respect to one another from limma*

Description

It can be nice to see a plot of two coefficients from a limma comparison with respect to one another This hopefully makes that easy.

Usage

```
limma_coefficient_scatter(output, toptable = NULL, x = 1, y = 2,
  gvis_filename = NULL, gvis_trendline = TRUE, z = 1.5,
  tooltip_data = NULL, flip = FALSE, base_url = NULL,
  up_color = "#7B9F35", down_color = "#DD0000", ...)
```

Arguments

`output`	the set of pairwise comparisons provided by limma_pairwise()
`toptable`	use this to get up/downs and color them on the scatter plot
`x`	the name or number of the first coefficient column to extract
`y`	the name or number of the second coefficient column to extract
`gvis_filename`	A filename for plotting gvis interactive graphs of the data.
`gvis_trendline`	add a trendline to the gvis plot?
`z`	how far from the median to color the plot red and green
`tooltip_data`	a dataframe of gene annotations to be used in the gvis plot
`flip`	flip the axes?
`base_url`	a basename for gvis plots
`up_color`	color for the ups
`down_color`	color for the downs
`...`	more parameters to make you happy

Value

a ggplot2 plot showing the relationship between the two coefficients

See Also

hpgl_linear_scatter limma_pairwise

Examples

```
## Not run:
 pretty = coefficient_scatter(limma_data, x="wt", y="mut")

## End(Not run)
```

limma_pairwise *Set up a model matrix and set of contrasts to do a pairwise comparison of all conditions using voom/limma.*

Description

Set up a model matrix and set of contrasts to do a pairwise comparison of all conditions using voom/limma.

Usage

```
limma_pairwise(input, conditions = NULL, batches = NULL,
  model_cond = TRUE, model_batch = FALSE, model_intercept = FALSE,
  extra_contrasts = NULL, alt_model = NULL, libsize = NULL,
  annot_df = NULL, ...)
```

Arguments

input	a dataframe/vector or expt class containing count tables, normalization state, etc.
conditions	a factor of conditions in the experiment
batches	a factor of batches in the experiment
model_cond	include condition in the model?
model_batch	include batch in the model? This is hopefully TRUE.
model_intercept	perform a cell-means or intercept model? A little more difficult for me to understand. I have tested and get the same answer either way.
extra_contrasts	some extra contrasts to add to the list This can be pretty neat, lets say one has conditions A,B,C,D,E and wants to do (C/B)/A and (E/D)/A or (E/D)/(C/B) then use this with a string like: "c_vs_b_ctrla = (C-B)-A, e_vs_d_ctrla = (E-D)-A, de_vs_cb = (E-D)-(C-B),"
alt_model	a separate model matrix instead of the normal condition/batch.
libsize	I've recently figured out that libsize is far more important than I previously realized. Play with it here.
annot_df	data frame for annotations
...	The elipsis parameter is fed to write_limma() at the end.

Value

A list including the following information: macb = the mashing together of condition/batch so you can look at it macb_model = The result of calling model.matrix(~0 + macb) macb_fit = The result of calling lmFit(data, macb_model) voom_result = The result from voom() voom_design = The design from voom (redundant from voom_result, but convenient) macb_table = A table of the number of times each condition/batch pairing happens cond_table = A table of the number of

times each condition appears (the denominator for the identities) batch_table = How many times each batch appears identities = The list of strings defining each condition by itself all_pairwise = The list of strings defining all the pairwise contrasts contrast_string = The string making up the makeContrasts() call pairwise_fits = The result from calling contrasts.fit() pairwise_comparisons = The result from eBayes() limma_result = The result from calling write_limma()

See Also

write_limma

Examples

```
## Not run:
pretend = balanced_pairwise(data, conditions, batches)

## End(Not run)
```

`limma_scatter` *Plot arbitrary data from limma*

Description

Plot arbitrary data from limma

Usage

```
limma_scatter(all_pairwise_result, first_table = 1, first_column = "logFC",
  second_table = 2, second_column = "logFC", type = "linear_scatter", ...)
```

Arguments

`all_pairwise_result`	the result from calling balanced_pairwise()
`first_table`	the first table from all_pairwise_result$limma_result to look at (may be a name or number)
`first_column`	the name of the column to plot from the first table
`second_table`	the second table inside all_pairwise_result$limma_result (name or number)
`second_column`	a column to compare against
`type`	A type of scatter plot (linear model, distance, vanilla)
`...`	so that you may feed it the gvis/tooltip information to make clicky graphs if so desired.

Value

a hpgl_linear_scatter() set of plots comparing the chosen columns If you forget to specify tables to compare, it will try the first vs the second.

See Also

hpgl_linear_scatter topTable

Examples

```
## Not run:
compare_logFC = limma_scatter(all_pairwise, first_table="wild_type", second_column="mutant", first_table="AveEx
compare_B = limma_scatter(all_pairwise, first_column="B", second_column="B")

## End(Not run)
```

---

limma_subset *A quick and dirty way to pull the top/bottom genes from toptable()*

Description

If neither n nor z is provided, it assumes you want 1.5 z-scores from the median.

Usage

```
limma_subset(table, n = NULL, z = NULL)
```

Arguments

table	the original data from limma
n	a number of genes to keep
z	a number of z-scores from the mean

Value

a dataframe subset from toptable

See Also

limma

Examples

```
## Not run:
 subset = limma_subset(df, n=400)
 subset = limma_subset(df, z=1.5)

## End(Not run)
```

loadme *Load a backup rdata file*

Description

I often use R over a sshfs connection, sometimes with significant latency, and I want to be able to save/load my R sessions relatively quickly. Thus this function uses my backup directory to load its R environment.

Usage

```
loadme(dir = "savefiles")
```

Arguments

dir	the directory containing the RData.rda.xz file.

Value

a bigger global environment

See Also

load save

Examples

```
## Not run:
loadme()
saveme()

## End(Not run)
```

lowfilter_counts *A caller for different low-count filters*

Description

A caller for different low-count filters

Usage

```
lowfilter_counts(count_table, type = "cbcb", p = 0.01, A = 1, k = 1,
  cv_min = 0.01, cv_max = 1000, thresh = 2, min_samples = 2)
```

Arguments

count_table	some counts to filter
type	Filtering method to apply (cbcb, pofa, kofa, cv right now)
p	For pofa()
A	For pofa()
k	For kofa()
cv_min	For cv()
cv_max	For cv()
thresh	Minimum threshold across samples for cbcb
min_samples	Minimum number of samples for cbcb

Value

a data frame of lowfiltered counts

See Also

genefilter

Examples

```
## Not run:
new <- lowfilter_counts(old)

## End(Not run)
```

---

makeSVD *this a function scabbed from Hector and Kwame's cbcbSEQ It just does fast.svd of a matrix against its rowMeans().*

Description

this a function scabbed from Hector and Kwame's cbcbSEQ It just does fast.svd of a matrix against its rowMeans().

Usage

```
makeSVD(data)
```

Arguments

data	A data frame to decompose

Value

a list containing the s,v,u from fast.svd

See Also

corpcor fast.svd

Examples

```
## Not run:
 svd = makeSVD(data)

## End(Not run)
```

`make_exampledata` *A small hack of limma's exampleData() function to allow for arbitrary data set sizes.*

Description

A small hack of limma's exampleData() function to allow for arbitrary data set sizes.

Usage

```
make_exampledata(ngenes = 1000, columns = 5)
```

Arguments

`ngenes`	how many genes in the fictional data set.
`columns`	how many samples in this data set.

Value

a matrix of pretend counts

See Also

limma

Examples

```
## Not run:
 pretend = make_exampledata()

## End(Not run)
```

make_id2gomap *Make a go mapping from IDs in a format suitable for topGO*

Description

Make a go mapping from IDs in a format suitable for topGO

Usage

```
make_id2gomap(goid_map = "reference/go/id2go.map", goids_df = NULL,
  overwrite = FALSE)
```

Arguments

goid_map	A topGO mapping file
goids_df	If there is no goid_map, create it with this
overwrite	A boolean, if it already exists, rewrite the mapping file?

Value

a summary of the new goid table

---

make_pairwise_contrasts *Run makeContrasts() with all pairwise comparisons.*

Description

Run makeContrasts() with all pairwise comparisons.

Usage

```
make_pairwise_contrasts(model, conditions, do_identities = TRUE,
  do_pairwise = TRUE, extra_contrasts = NULL)
```

Arguments

model	a model describing the conditions/batches/etc in the experiment
conditions	a factor of conditions in the experiment
do_identities	whether or not to include all the identity strings. Limma can handle this, edgeR cannot.
do_pairwise	whether or not to include all the pairwise strings. This shouldn't need to be set to FALSE, but just in case.
extra_contrasts	an optional string of extra contrasts to include.

Value

A list including the following information: all_pairwise_contrasts = the result from makeContrasts(...) identities = the string identifying each condition alone all_pairwise = the string identifying each pairwise comparison alone contrast_string = the string passed to R to call makeContrasts(...) names = the names given to the identities/contrasts

See Also

makeContrasts

Examples

```
## Not run:
 pretend = make_pairwise_contrasts(model, conditions)

## End(Not run)
```

`make_report` *Make a knitr report with some defaults set*

Description

Make a knitr report with some defaults set

Usage

```
make_report(name = "report", type = "pdf")
```

Arguments

`name`	Name the document!
`type`	html/pdf/fancy html reports?

Value

a dated report file

See Also

knitr rmarkdown knitrBootstrap

`make_tooltips` *Create a simple df from gff which contains tooltip usable information for gVis graphs. The tooltip column is also a handy proxy for anontations information when it would otherwise be too troublesome.*

Description

Create a simple df from gff which contains tooltip usable information for gVis graphs. The tooltip column is also a handy proxy for anontations information when it would otherwise be too troublesome.

Usage

```
make_tooltips(annotations, desc_col = "description")
```

Arguments

`annotations`	Either a gff file or annotation data frame (which likely came from a gff file.)
`desc_col`	a column from a gff file to grab the data from

Value

a df of tooltip information or name of a gff file

See Also

googleVis gff2df

Examples

```
## Not run:
tooltips <- make_tooltips('reference/gff/saccharomyces_cerevisiae.gff.gz')

## End(Not run)
```

`median_by_factor` *Create a data frame of the medians of rows by a given factor in the data*

Description

This assumes of course that (like expressionsets) there are separate columns for each replicate of the conditions. This will just iterate through the levels of a factor describing the columns, extract them, calculate the median, and add that as a new column in a separate data frame.

Usage

```
median_by_factor(data, fact)
```

Arguments

data	a data frame, presumably of counts.
fact	a factor describing the columns in the data.

Value

a data frame of the medians

Examples

```
## Not run:
 compressed = hpgltools:::median_by_factor(data, experiment$condition)

## End(Not run)
```

---

my_identifyAUBlocks *copy/paste the function from SeqTools and find where it falls on its ass.*

Description

Yeah, I do not remember what I changed in this function.

Usage

```
my_identifyAUBlocks(x, min.length = 20, p.to.start = 0.8, p.to.end = 0.55)
```

Arguments

x	A sequence object
min.length	I dunno.
p.to.start	the p to start of course
p.to.end	and the p to end

Value

a list of IRanges which contain a bunch of As and Us.

normalize_counts *Perform a simple normalization of a count table*

Description

Perform a simple normalization of a count table

Usage

```
normalize_counts(data, design = NULL, norm = "raw")
```

Arguments

data	A matrix of count data
design	A dataframe describing the experimental design (conditions/batches/etc)
norm	A normalization to perform: 'sf|quant|qsmooth|tmm|upperquartile|tmm|rle' I keep wishy-washing on whether design is a required argument.

Value

dataframe of normalized(counts)

See Also

edgeR limma DESeq2

Examples

```
## Not run:
norm_table = normalize_counts(count_table, design=design, norm='qsmooth')

## End(Not run)
```

normalize_expt *Replace the data of an expt with normalized data.*

Description

Replace the data of an expt with normalized data.

Usage

```
normalize_expt(expt, transform = "raw", norm = "raw", convert = "raw",
  batch = "raw", filter_low = FALSE, annotations = NULL, fasta = NULL,
  entry_type = "gene", verbose = FALSE, use_original = FALSE,
  batch1 = "batch", batch2 = NULL, thresh = 2, min_samples = 2,
  p = 0.01, A = 1, k = 1, cv_min = 0.01, cv_max = 1000, ...)
```

Arguments

expt	The original expt
transform	The transformation desired (raw, log2, log, log10)
norm	How to normalize the data (raw, quant, sf, upperquartile, tmm, rle)
convert	Conversion to perform (raw, cpm, rpkm, cp_seq_m)
batch	Batch effect removal tool to use (limma sva fsva ruv etc)
filter_low	Filter out low sequences (cbcb, pofa, kofa, others?)
annotations	used for rpkm, a df
fasta	fasta file for cp_seq_m counting of oligos
entry_type	for getting genelengths by feature type (rpkm or cp_seq_m)
verbose	talk?
use_original	whether to use the backup data in the expt class
batch1	experimental factor to extract first
batch2	a second factor to remove (only with limma's removebatcheffect())
thresh	for cbcb_lowfilter
min_samples	for cbcb_lowfilter
p	for genefilter's pofa
A	for genefilter's pofa
k	for genefilter's kofa
cv_min	for genefilter's cv()
cv_max	for genefilter's cv()
...	more options

Value

a new expt object with normalized data and the original data saved as 'original_expressionset'

See Also

genefilter cbcbSEQ limma sva edgeR DESeq2

Examples

```
## Not run:
normed <- normalize_expt(exp, transform='log2', norm='rle', convert='cpm', batch='raw', filter_low='pofa')
normed_batch <- normalize_expt(exp, transform='log2', norm='rle', convert='cpm', batch='sva', filter_low='pofa')

## End(Not run)
```

parse_gene_go_terms *TriTrypDB gene information table GO term parser*

Description

TriTrypDB gene information table GO term parser

Usage

```
parse_gene_go_terms(filepath, verbose = FALSE)
```

Arguments

filepath	Location of TriTrypDB gene information table.
verbose	Whether or not to enable verbose output.

Value

Returns a dataframe where each line includes a gene/GO terms pair along with some addition information about the GO term. Note that because each gene may have multiple GO terms, a single gene ID may appear on multiple lines.

Author(s)

Keith Hughitt

parse_gene_info_table *TriTrypDB gene information table parser*

Description

An example input file is the T. brucei Lister427 gene information table available at: http://tritrypdb.org/common/downloads/C
5.0_TbruceiLister427Gene.txt

Usage

```
parse_gene_info_table(filepath, verbose = FALSE)
```

Arguments

filepath	Location of TriTrypDB gene information table.
verbose	Whether or not to enable verbose output.

Value

Returns a dataframe of gene info.

Author(s)

Keith Hughitt

`pattern_count_genome` *Find how many times a given pattern occurs in every gene of a genome.*

Description

Find how many times a given pattern occurs in every gene of a genome.

Usage

```
pattern_count_genome(fasta, gff = NULL, pattern = "TA", type = "gene",
  key = "locus_tag")
```

Arguments

`fasta`	a fasta genome
`gff`	an optional gff of annotations (if not provided it will just ask the whole genome.
`pattern`	what pattern to search for? This was used for tnseq and TA is the mariner insertion point.
`type`	the column to get frmo the gff file
`key`	what type of entry of the gff file to key from?

Value

num_pattern a data frame of names and numbers.

See Also

Biostrings Rsamtools PDict FaFile

Examples

```
## Not run:
num_pattern = pattern_count_genome('mgas_5005.fasta', 'mgas_5005.gff')

## End(Not run)
```

pca_highscores *Get the highest/lowest scoring genes for every principle component.*

Description

This function uses princomp to acquire a principle component biplot for some data and extracts a dataframe of the top n genes for each component by score.

Usage

```
pca_highscores(df = NULL, conditions = NULL, batches = NULL, n = 20)
```

Arguments

df	a dataframe of (pseudo)counts
conditions	a factor or character of conditions in the experiment.
batches	a factor or character of batches in the experiment.
n	the number of genes to extract.

Value

a list including the princomp biplot, histogram, and tables of top/bottom n scored genes with their scores by component.

See Also

princomp

Examples

```
## Not run:
 information = pca_highscores(df=df, conditions=cond, batches=bat)
 information$pca_bitplot  ## oo pretty

## End(Not run)
```

`pca_information` *Gather information about principle components.*

Description

Calculate some information useful for generating PCA plots.

Usage

```
pca_information(expt_data, expt_design = NULL, expt_factors = c("condition",
  "batch"), num_components = NULL, plot_pcas = FALSE,
  plot_labels = "fancy")
```

Arguments

`expt_data`	the data to analyze (usually exprs(somedataset)).
`expt_design`	a dataframe describing the experimental design, containing columns with useful information like the conditions, batches, number of cells, whatever...
`expt_factors`	a character list of experimental conditions to query for R^2 against the fast.svd of the data.
`num_components`	a number of principle components to compare the design factors against. If left null, it will query the same number of components as factors asked for.
`plot_pcas`	plot the set of PCA plots for every pair of PCs queried.
`plot_labels`	how to label the glyphs on the plot.

Details

pca_information seeks to gather together interesting information to make principle component analyses easier, including: the results from (fast.)svd, a table of the r^2 values, a table of the variances in the data, coordinates used to make a pca plot for an arbitrarily large set of PCs, correlations and fstats between experimental factors and the PCs, and heatmaps describing these relationships. Finally, it will provide a plot showing how much of the variance is provided by the top-n genes and (optionally) the set of all PCA plots with respect to one another. (PCx vs. PCy)

Value

a list of fun pca information: svd_u/d/v: The u/d/v parameters from fast.svd rsquared_table: A table of the rsquared values between each factor and principle component pca_variance: A table of the pca variances pca_data: Coordinates for a pca plot pca_cor: A table of the correlations between the factors and principle components anova_fstats: the sum of the residuals with the factor vs without (manually calculated) anova_f: The result from performing anova(withfactor, withoutfactor), the F slot anova_p: The p-value calculated from the anova() call anova_sums: The RSS value from the above anova() call cor_heatmap: A heatmap from recordPlot() describing pca_cor.

Warning

This function has gotten too damn big and needs to be split up.

See Also

fast.svd, lm

Examples

```
## Not run:
 pca_info = pca_information(exprs(some_expt$expressionset), some_design, "all")
 pca_info

## End(Not run)
```

---

pca_plot_largebatch *ggplot2 plots of PCA data with >= 6 batches.*

Description

ggplot2 plots of PCA data with >= 6 batches.

Usage

```
pca_plot_largebatch(df, size = 5, first = "PC1", second = "PC2")
```

Arguments

df	A dataframe of PC1/PC2 and other arbitrary data.
size	The size of glyphs in the plot.
first	The first principle component to plot against
second	The second PC to plot against

Value

a ggplot2 plot of principle components 1 and 2.

See Also

ggplot2

Examples

```
## Not run:
 plots <- pca_plot_largebatch(svd_stuff)

## End(Not run)
```

pca_plot_smallbatch *ggplot2 plots of PCA data with <= 5 batches.*

Description

This uses hard-coded scale_shape_manual values 21-25 to have solid shapes in the plot.

Usage

```
pca_plot_smallbatch(df, size = 5, first = "PC1", second = "PC2")
```

Arguments

df	A dataframe of PC1/PC2 and other arbitrary data.
size	The size of glyphs in the plot.
first	The first component
second	The second component

Value

a ggplot2 plot of principle components 1 and 2.

See Also

ggplot2

Examples

```
## Not run:
 plots <- pca_plot_smallbatch(svd_stuff)

## End(Not run)
```

plot_essentiality *Plot the essentiality of a library as per DeJesus et al.*

Description

Plot the essentiality of a library as per DeJesus et al.

Usage

```
plot_essentiality(file)
```

Arguments

file	a file created using the perl script 'essentiality_tas.pl'

Value

A couple of plots

---

plot_pcs *A quick and dirty PCA plotter of arbitrary components against one another.*

Description

A quick and dirty PCA plotter of arbitrary components against one another.

Usage

```
plot_pcs(data, first = "PC1", second = "PC2", variances = NULL,
  design = NULL, plot_title = NULL, plot_labels = NULL)
```

Arguments

data	a dataframe of principle components PC1 .. PCN with any other arbitrary information.
first	principle component PCx to put on the x axis.
second	principle component PCy to put on the y axis.
variances	a list of the percent variance explained by each component.
design	the experimental design with condition batch factors.
plot_title	a title for the plot.
plot_labels	a parameter for the labels on the plot.

Value

a ggplot2 PCA plot

See Also

ggplot2 geom_dl

Examples

```
## Not run:
 pca_plot = plot_pcs(pca_data, first="PC2", second="PC4", design=expt$design)

## End(Not run)
```

plot_topgo_densities *Plot the density of categories vs. the possibilities*

Description

This can make a large number of plots

Usage

```
plot_topgo_densities(godata, table)
```

Arguments

godata	the result from topgo
table	a table of genes

print_ups_downs *Reprint the output from extract_significant_genes()*

Description

I found myself needing to reprint these excel sheets because I added some new information. This shortcuts that process for me.

Usage

```
print_ups_downs(upsdowns, sig_table = "excel/significant_genes.xlsx")
```

Arguments

upsdowns	the output from extract_significant_genes()
sig_table	table to write to

Value

the return from write_xls

See Also

combine_de_tables

pval_plot *Make a pvalue plot from a df of IDs, scores, and p-values.*

Description

This function seeks to make generating pretty pvalue plots as shown by clusterprofiler easier.

Usage

```
pval_plot(df, ontology = "MF")
```

Arguments

df	some data from topgo/goseq/clusterprofiler.
ontology	an ontology to plot (MF,BP,CC).

Value

a plot!

See Also

goseq **ggplot2**

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

Description

Load a library, install it first if necessary.

Usage

```
require.auto(lib, github_path = NULL, verbose = FALSE, update = FALSE)
```

Arguments

lib	string name of a library
github_path	an optional github username/path.
verbose	print some information while loading.
update	update packages?

Details

This was taken from: http://sbamin.com/2012/11/05/tips-for-working-in-r-automatically-install-missing-package/

Value

NULL currently

See Also

biocLite install.packages

Examples

```
## Not run:
require.auto("ggplot2")

## End(Not run)
```

---

saveme *Make a backup rdata file for future reference*

Description

I often use R over a sshfs connection, sometimes with significant latency, and I want to be able to save/load my R sessions relatively quickly. Thus this function uses pxz to compress the R session maximally and relatively fast. This assumes you have pxz installed and >= 4 CPUs.

Usage

```
saveme(directory = "savefiles", backups = 4)
```

Arguments

directory	the directory to save the Rdata file.
backups	how many revisions?

Value

the command used to save the global environment

See Also

save pipe

Examples

```
## Not run:
saveme()

## End(Not run)
```

semantic_copynumber_filter *Remove multicopy genes from up/down gene expression lists*

Description

Remove multicopy genes from up/down gene expression lists

Usage

```
semantic_copynumber_filter(de_list, max_copies = 2, semantic = c("mucin",
  "sialidase", "RHS", "MASP", "DGF"), semantic_column = "1.tooltip")
```

Arguments

de_list	a list of sets of genes deemed significantly up/down with a column expressing approximate count numbers
max_copies	Keep only those genes with <= n putative copies
semantic	a set of strings to exclude
semantic_column	a column to use to find the above mentioned strings

Value

a smaller list of up/down genes

sillydist *A stupid distance function of a point against two axes.*

Description

A stupid distance function of a point against two axes.

Usage

```
sillydist(firstterm, secondterm, firstaxis = 0, secondaxis = 0)
```

Arguments

firstterm	the x-values of the points.
secondterm	the y-values of the points.
firstaxis	the x-value of the vertical axis.
secondaxis	the y-value of the second axis.

Value

dataframe of the distances This just takes the abs(distances) of each point to the axes, normalizes them against the largest point on the axes, multiplies the result, and normalizes against the max of all points.

See Also

ggplot2

Examples

```
## Not run:
mydist <- sillydist(df[,1], df[,2], first_median, second_median)
first_vs_second <- ggplot2::ggplot(df, ggplot2::aes_string(x="first", y="second"), environment=hpgl_env) +
  ggplot2::xlab(paste("Expression of", df_x_axis)) +
  ggplot2::ylab(paste("Expression of", df_y_axis)) +
  ggplot2::geom_vline(color="grey", xintercept=(first_median - first_mad), size=line_size) +
  ggplot2::geom_vline(color="grey", xintercept=(first_median + first_mad), size=line_size) +
  ggplot2::geom_vline(color="darkgrey", xintercept=first_median, size=line_size) +
  ggplot2::geom_hline(color="grey", yintercept=(second_median - second_mad), size=line_size) +
  ggplot2::geom_hline(color="grey", yintercept=(second_median + second_mad), size=line_size) +
  ggplot2::geom_hline(color="darkgrey", yintercept=second_median, size=line_size) +
  ggplot2::geom_point(colour=grDevices::hsv(mydist$dist, 1, mydist$dist), alpha=0.6, size=size) +
  ggplot2::theme(legend.position="none")
first_vs_second  ## dots get colored according to how far they are from the medians
## replace first_median, second_median with 0,0 for the axes

## End(Not run)
```

---

simple_clusterprofiler

Perform a simplified clusterProfiler analysis

Description

Perform a simplified clusterProfiler analysis

Usage

```
simple_clusterprofiler(de_genes, goids = NULL, golevel = 4, pcutoff = 0.1,
  qcutoff = 1, fold_changes = NULL, include_cnetplots = FALSE,
  showcategory = 12, universe = NULL, organism = "lm", gff = NULL,
  wrapped_width = 20, method = "Wallenius", padjust = "BH", ...)
```

Arguments

de_genes	a data frame of differentially expressed genes, containing IDs and whatever other columns
goids	a file containing mappings of genes to goids in the format expected by topgo
golevel	a relative level in the tree for printing p-value plots, higher is more specific
pcutoff	a p-value cutoff
qcutoff	a q-value cutoff
fold_changes	a df of fold changes for the DE genes
include_cnetplots	the cnetplots are often stupid and can be left behind
showcategory	how many categories to show in p-value plots
universe	universe to use
organism	name of the species to use
gff	gff file to generate the universe
wrapped_width	width of ontology names in the pvalue plots
method	pvalue calculation method
padjust	a method for adjusting the p-values
...	more options!

Value

a big list including the following: mf_interesting: A table of the interesting molecular function groups bp_interesting: A table of the interesting biological process groups cc_interesting: A table of the interesting cellular component groups mf_pvals: A histogram of the molecular function p-values bp_pvals: Ditto, biological process cc_pvals: And cellular component... mf_enriched: A table of the enriched molecular function groups by adjusted p-value. bp_enriched: yep, you guessed it cc_enriched: cellular component, too mf_all/bp_all/cc_all: A table of all go categories observed (mf/bp/cc respectively) mfp_plot/bpp_plot/ccp_plot: ggplot2 p-value bar plots describing the over represented groups mf_cnetplot/bp_cnetplot/cc_cnetplot: clusterProfiler cnetplots mf_group_barplot/bp_group_barplot/cc_g The group barplots from clusterProfiler

Examples

```
## Not run:
 up_cluster = simple_clusterprofiler(mga2_ll_thy_top, goids=goids, gff="reference/genome/gas.gff")
## > Some chattery while it runs
## tail(head(up_cluster$bp_interesting, n=10), n=1)
## > ID ont GeneRatio BgRatio     pvalue   p.adjust    qvalue
## > 10 GO:0009311  BP     5/195 10/1262 0.01089364 0.01089364 0.1272835
## >   geneID Count
## >   10 M5005_Spy1632/M5005_Spy1637/M5005_Spy1635/M5005_Spy1636/M5005_Spy1638     5
## >   Description
## >   10 oligosaccharide metabolic process

## End(Not run)
```

simple_comparison *Perform a simple experimental/control comparison This is a function written primarily to provide examples for how to use limma. It does the following: 1. Makes a model matrix using condition/batch 2. Optionally uses sva's combat (from cbcbSEQ) 3. Runs voom/lmfit 4. Sets the first element of the design to "changed" and the second to "control". 5. Performs a makeContrasts() of changed - control. 6. Fits them 7. Makes histograms of the two elements of the contrast, cor.tests() them, makes a histogram of the p-values, ma-plot, volcano-plot, writes out the results in an excel sheet, pulls the up/down significant and p-value significant (maybe this should be replaced with write_limma()? 8. And returns a list containining these data and plots.*

Description

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.

Usage

```
simple_comparison(subset, workbook = "simple_comparison.xls",
  sheet = "simple_comparison", basename = NA, batch = TRUE,
  combat = FALSE, combat_noscale = TRUE, pvalue_cutoff = 0.05,
  logfc_cutoff = 0.6, tooltip_data = NULL, verbose = FALSE, ...)
```

Arguments

subset	an experimental subset with two conditions to compare.
workbook	an excel workbook to which to write.
sheet	an excel worksheet to which to write.
basename	a url to which to send click evens in clicky volcano/ma plots.
batch	whether or not to include batch in limma's model.
combat	whether or not to use combatMod().
combat_noscale	whether or not to include combat_noscale (makes combat a little less heavy-handed).
pvalue_cutoff	p-value definition of 'significant.'
logfc_cutoff	fold-change cutoff of significance. 0.6 on the low end and therefore 1.6 on the high.
tooltip_data	text descriptions of genes if one wants google graphs.
verbose	be verbose?
...	more parameters!

Value

A list containing the following pieces: amean_histogram = a histogram of the mean values between the two conditions coef_amean_cor = a correlation test between the mean values and coefficients (this should be a p-value of 1) coefficient_scatter = a scatter plot of condition 2 on the y axis and condition 1 on x coefficient_x = a histogram of the x axis coefficient_y = a histogram of the y axis coefficient_both = a histogram of both coefficient_lm = a description of the line described by y=slope(y/x)+b where coefficient_lmsummary = the r-squared and such information for the linear model coefficient_weights = the weights against the linear model, higher weights mean closer to the line comparisons = the result from eBayes() contrasts = the result from contrasts.fit() contrast_histogram = a histogram of the coefficients downsignificant = a subset from toptable() of the 'down-regulated' genes (< 1 Z from the mean) fit = the result from lmFit(voom_result) ma_plot = an ma plot using the voom$E data and p-values psignificant = a subset from toptable() of all genes with p-values <= pvalue_cutoff pvalue_histogram = a histogram of all the p-values table = everything from toptable() upsignificant = a subset from toptable() of 'up-regulated' genes (> 1 Z from the mean) volcano_plot = a volcano plot of x/y voom_data = the result from calling voom() voom_plot = a plot from voom(), redunant with voom_data

See Also

hpgl_gvis_ma_plot toptable voom voomMod hpgl_voom lmFit makeContrasts contrasts.fit

Examples

```
## Not run:
 model = model.matrix(~ 0 + subset$conditions)
 simple_comparison(subset, model)

## End(Not run)
```

---

simple_gadem *run the rGADEM suite*

Description

This is another function I started but never had cause to finish for the test sequences it works though

Usage

```
simple_gadem()
```

simple_goseq *simple_goseq() Perform a simplified goseq analysis*

Description

simple_goseq() Perform a simplified goseq analysis

Usage

```
simple_goseq(de_genes, all_genes = NULL, lengths = NULL, goids = NULL,
  doplot = TRUE, adjust = 0.1, pvalue = 0.1, qvalue = 0.1,
  goseq_method = "Wallenius", padjust_method = "BH", species = NULL,
  length_db = "ensGene", gff = NULL, ...)
```

Arguments

de_genes	a data frame of differentially expressed genes, containing IDs and whatever other columns
all_genes	the universe of possible genes
lengths	the length of each gene with an ID in de_genes
goids	a list of ontology accessions to gene accessions
doplot	include pwf plots
adjust	minimum adjusted pvalue
pvalue	minimum pvalue
qvalue	minimum qvalue
goseq_method	testing used by goseq
padjust_method	which method to adjust the pvalues
species	optionally choose a species from supportedOrganisms()
length_db	Source of gene lengths
gff	gff file source of gene lengths
...	extra parameters which I do not recall

Value

a big list including: the pwd:pwf function, alldata:the godata dataframe, pvalue_histogram:p-value histograms, godata_interesting:the ontology information of the enhanced groups, term_table:the goterms with some information about them, mf_subset:a plot of the MF enhanced groups, mfp_plot:the pvalues of the MF group, bp_subset:a plot of the BP enhanced groups, bpp_plot, cc_subset, and ccp_plot

See Also

goseq goseq nullp

simple_gostats *A simplification function for gostats, in the same vein as those written for clusterProfiler, goseq, and topGO.*

Description

GOstats has a couple interesting peculiarities: Chief among them: the gene IDs must be integers. As a result, I am going to have this function take a gff file in order to get the go ids and gene ids on the same page.

Usage

```
simple_gostats(de_genes, gff, goids, universe_merge = "ID",
  second_merge_try = "locus_tag", organism = "fun", pcutoff = 0.1,
  direction = "over", conditional = FALSE, categorysize = NULL,
  gff_type = "CDS", ...)
```

Arguments

de_genes	input list of differentially expressed genes
gff	The annotation information for this genome
goids	The set of GOids, as before in the format ID/GO
universe_merge	column from which to create the universe of genes
second_merge_try	if the first universe merge fails, try this
organism	genbank organism to use
pcutoff	pvalue cutoff for deciding significant
direction	under or over represented categories
conditional	perform a conditional search?
categorysize	category size below which to not include groups
gff_type	gff column to use for creating the universe
...	more parameters!

Value

dunno yet

See Also

GSEABase Category

`simple_topgo` *Perform a simplified topgo analysis*

Description

This will attempt to make it easier to run topgo on a set of genes.

Usage

```
simple_topgo(de_genes, goid_map = "reference/go/id2go.map", goids_df = NULL,
  pvals = NULL, limitby = "fisher", limit = 0.1, signodes = 100,
  sigforall = TRUE, numchar = 300, selector = "topDiffGenes",
  pval_column = "adj.P.Val", overwrite = FALSE, densities = FALSE,
  pval_plots = TRUE, ...)
```

Arguments

`de_genes`	a data frame of differentially expressed genes, containing IDs and whatever other columns
`goid_map`	a file containing mappings of genes to goids in the format expected by topgo
`goids_df`	a data frame of the goids which may be used to make the goid_map
`pvals`	a set of pvalues in the DE data which may be used to improve the topgo results
`limitby`	test to index the results by
`limit`	ontology pvalue to use as the lower limit
`signodes`	I don't remember right now
`sigforall`	provide the significance for all nodes?
`numchar`	character limit for the table of results
`selector`	a function name for choosing genes to include
`pval_column`	column from which to acquire scores
`overwrite`	yeah I do not remember this one either
`densities`	the densities, yeah, the densities
`pval_plots`	include pvalue plots of the results a la clusterprofiler
`...`	other options which I do not remember right now

Value

a big list including the various outputs from topgo

spirograph *Make spirographs!*

Description

Taken (with modifications) from: http://menugget.blogspot.com/2012/12/spirograph-with-r.html#more A positive value for 'B' will result in a epitrochoid, while a negative value will result in a hypotrochoid.

Usage

```
spirograph(radius_a = 1, radius_b = -4, dist_bc = -2, revolutions = 158,
  increments = 3160, center_a = list(x = 0, y = 0))
```

Arguments

radius_a	The radius of the primary circle.
radius_b	The radius of the circle travelling around a.
dist_bc	A point relative to the center of 'b' which rotates with the turning of 'b'.
revolutions	How many revolutions to perform in the plot
increments	The number of radial increments to be calculated per revolution
center_a	The position of the center of 'a'.

Value

something which I don't yet know.

subset_ontology_search *Perform ontology searches on data subsets.*

Description

Perform ontology searches on data subsets.

Usage

```
subset_ontology_search(changed_counts, doplot = FALSE, ...)
```

Arguments

changed_counts	the list of changed counts as ups and downs
doplot	include plots in the results
...	extra arguments which I don't realize

Value

a list of ontology search results, up and down for each contrast

---

`sum_exons` *Given a data frame of exon counts and annotation information, sum the exons.*

Description

This function will merge a count table to an annotation table by the child column. It will then sum all rows of exons by parent gene and sum the widths of the exons. Finally it will return a list containing a df of gene lengths and summed counts.

Usage

```
sum_exons(data, gff = NULL, annotdf = NULL, parent = "Parent",
  child = "row.names")
```

Arguments

`data`	a count table by exon
`gff`	a gff filename
`annotdf`	a dataframe of annotations (probably from gff2df)
`parent`	a column from the annotations with the gene names
`child`	a column from the annotations with the exon names

Value

a list of 2 data frames, counts and lengths by summed exons

See Also

rtracklayer

Examples

```
## Not run:
summed <- sum_exons(counts, gff='reference/xenopus_laevis.gff.xz')

## End(Not run)
```

`tnseq_saturation` *Make a plot and some simple numbers about tnseq saturation*

Description

This function takes as input a tab separated file from essentiality_tas.pl This is a perl script written to read a bam alignment of tnseq reads against a genome and count how many hits were observed on every TA in the given genome. It furthermore has some logic to tell the difference between reads which were observed on the forward vs. reverse strand as well as reads which appear to be on both strands (eg. they start and end with 'TA').

Usage

```
tnseq_saturation(file)
```

Arguments

`file`	a file created using the perl script 'essentiality_tas.pl'

Value

A plot and some numbers

`topDiffGenes` *A very simple selector of strong scoring genes (by p-value)*

Description

This function was provided in the topGO documentation, but not defined. It was copied/pasted here. I have ideas for including up/down expression but have so far deemed them not needed because I am feeding topGO already explicit lists of genes which are up/down/whatever. But it still is likely to be useful to be able to further subset the data.

Usage

```
topDiffGenes(allScore)
```

Arguments

`allScore`	The scores of the genes

topgo_pval_plot *Make a pvalue plot from topgo data*

Description

Make a pvalue plot from topgo data

Usage

```
topgo_pval_plot(topgo, wrapped_width = 20, cutoff = 0.1, n = 12,
  type = "fisher")
```

Arguments

topgo	some data from topgo!
wrapped_width	maximum width of the text names
cutoff	p-value cutoff for the plots
n	maximum number of ontologies to include
type	type of score to use

Value

a list of MF/BP/CC pvalue plots

See Also

topgo goseq

topgo_tables *Make pretty tables out of topGO data*

Description

The topgo function GenTable is neat, but it needs some simplification to not be obnoxious

Usage

```
topgo_tables(result, limit = 0.1, limitby = "fisher", numchar = 300,
  orderby = "classic", ranksof = "classic")
```

Arguments

result	a topgo result
limit	a pvalue limit defining 'significant'
limitby	what type of test to perform
numchar	how many characters to allow in the description
orderby	which of the available columns to order the table by?
ranksof	which of the available columns are used to rank the data?

---

topgo_trees *Print trees from topGO*

Description

Print trees from topGO

Usage

```
topgo_trees(tg, score_limit = 0.01, sigforall = TRUE,
  do_mf_fisher_tree = TRUE, do_bp_fisher_tree = TRUE,
  do_cc_fisher_tree = TRUE, do_mf_ks_tree = FALSE, do_bp_ks_tree = FALSE,
  do_cc_ks_tree = FALSE, do_mf_el_tree = FALSE, do_bp_el_tree = FALSE,
  do_cc_el_tree = FALSE, do_mf_weight_tree = FALSE,
  do_bp_weight_tree = FALSE, do_cc_weight_tree = FALSE)
```

Arguments

tg	data from simple_topgo()
score_limit	score limit to decide whether to add to the tree
sigforall	add scores to the tree?
do_mf_fisher_tree	Add the fisher score molecular function tree?
do_bp_fisher_tree	Add the fisher biological process tree?
do_cc_fisher_tree	Add the fisher cellular component tree?
do_mf_ks_tree	Add the ks molecular function tree?
do_bp_ks_tree	Add the ks biological process tree?
do_cc_ks_tree	Add the ks cellular component tree?
do_mf_el_tree	Add the el molecular function tree?
do_bp_el_tree	Add the el biological process tree?
do_cc_el_tree	Add the el cellular component tree?

do_mf_weight_tree
: Add the weight mf tree?

do_bp_weight_tree
: Add the bp weighted tree?

do_cc_weight_tree
: Add the guess

Value

a big list including the various outputs from topgo

`transform_counts` *Perform a simple transformation of a count table (log2)*

Description

the add argument is only important if the data was previously cpm'd because that does a +1, thus this will avoid a double+1 on the data.

Usage

```
transform_counts(count_table, transform = "raw", converted = "raw",
  base = NULL, add = 0.5)
```

Arguments

count_table	A matrix of count data
transform	A type of transformation to perform: log2/log10/log
converted	Whether or not the data has been converted.
base	for other log scales
add	to avoid attempting a log(0)

Value

dataframe of logx(counts)

Examples

```
## Not run:
filtered_table = transform_counts(count_table, transform='log2', converted='cpm')

## End(Not run)
```

u_plot — *Plot the rank order svd$u elements to get a view of how much the first genes contribute to the total variance by PC.*

Description

Plot the rank order svd$u elements to get a view of how much the first genes contribute to the total variance by PC.

Usage

```
u_plot(plotted_us)
```

Arguments

plotted_us	a list of svd$u elements

Value

a recordPlot() plot showing the first 3 PCs by rank-order svd$u.

write_go_xls — *Write gene ontology tables for excel*

Description

Combine the results from goseq, cluster profiler, topgo, and gostats and drop them into excel Hopefully with a relatively consistent look.

Usage

```
write_go_xls(goseq, cluster, topgo, gostats, file = "excel/merged_go",
  dated = TRUE, n = 30, overwritefile = TRUE)
```

Arguments

goseq	The goseq result from simple_goseq()
cluster	The result from simple_clusterprofiler()
topgo	Guess
gostats	Yep, ditto
file	the file to save the results.
dated	date the excel file
n	the number of ontology categories to include in each table.
overwritefile	overwrite an existing excel file

Value

the list of ontology information

`write_limma` *Writes out the results of a limma search using toptable() However, this will do a couple of things to make one's life easier: 1. Make a list of the output, one element for each comparison of the contrast matrix 2. Write out the toptable() output for them in separate .csv files and/or sheets in excel 3. Since I have been using qvalues a lot for other stuff, add a column for them.*

Description

Writes out the results of a limma search using toptable() However, this will do a couple of things to make one's life easier: 1. Make a list of the output, one element for each comparison of the contrast matrix 2. Write out the toptable() output for them in separate .csv files and/or sheets in excel 3. Since I have been using qvalues a lot for other stuff, add a column for them.

Usage

```
write_limma(data, adjust = "fdr", n = 0, coef = NULL,
  workbook = "excel/limma.xls", excel = FALSE, csv = FALSE,
  annot_df = NULL)
```

Arguments

`data`	the output from eBayes()
`adjust`	the pvalue adjustment chosen.
`n`	the number of entries to report, 0 says do them all.
`coef`	which coefficients/contrasts to report, NULL says do them all.
`workbook`	an excel filename into which to write the data
`excel`	write an excel workbook?
`csv`	write out csv files of the tables?
`annot_df`	an optional data frame including annotation information to include with the tables.

Value

a list of data frames comprising the toptable output for each coefficient, I also added a qvalue entry to these toptable() outputs.

See Also

toptable write_xls

Examples

```
## Not run:
 finished_comparison = eBayes(limma_output)
 data_list = write_limma(finished_comparison, workbook="excel/limma_output.xls")

## End(Not run)
```

`write_subset_ontologies` *Write gene ontology tables for data subsets*

Description

Given a set of ontology results, this attempts to write them to an excel workbook in a consistent and relatively easy-to-read fashion.

Usage

```
write_subset_ontologies(kept_ontology, outfile = "excel/subset_go",
  dated = TRUE, n = 50, overwritefile = TRUE, add_plots = TRUE,
  table_style = "TableStyleMedium9", ...)
```

Arguments

`kept_ontology`	A result from subset_ontology_search()
`outfile`	Workbook to which to write.
`dated`	Append the year-month-day-hour to the workbook.
`n`	How many ontology categories to write for each search
`overwritefile`	Overwrite an existing workbook?
`add_plots`	Add the various p-value plots to the end of each sheet?
`table_style`	The chosen table style for excel
`...`	some extra parameters

Value

a set of excel sheet/coordinates

Examples

```
## Not run:
 all_contrasts <- all_pairwise(expt, model_batch=TRUE)
 keepers <- list(bob = ('numerator','denominator'))
 kept <- combine_de_tables(all_contrasts, keepers=keepers)
 changed <- extract_significant_genes(kept)
 kept_ontologies <- subset_ontology_search(changed, lengths=gene_lengths, goids=goids, gff=gff, gff_type='gene')
 go_writer <- write_subset_ontologies(kept_ontologies)

## End(Not run)
```

write_xls *Write a dataframe to an excel spreadsheet sheet. I like to give folks data in any format they prefer, even though I sort of hate excel. Most people I work with use it, so therefore I do too. This function has been through many iterations, first using XLConnect, then xlsx, and now openxlsx. Hopefully this will not change again.*

Description

Write a dataframe to an excel spreadsheet sheet. I like to give folks data in any format they prefer, even though I sort of hate excel. Most people I work with use it, so therefore I do too. This function has been through many iterations, first using XLConnect, then xlsx, and now openxlsx. Hopefully this will not change again.

Usage

```
write_xls(data, sheet = "first", file = "excel/workbook.xlsx",
  overwrite_file = TRUE, newsheet = FALSE, overwrite_sheet = TRUE,
  dated = TRUE, first_two_widths = c("30", "60"), start_row = 1,
  start_col = 1, ...)
```

Arguments

data	A data frame to print
sheet	Name of the sheet to write
file	The filename for the workbook.
overwrite_file	required for XLConnect, still used but perhaps not needed.
newsheet	same, but makes sure we don't overwrite an existing sheet
overwrite_sheet	yeah, I need to prune these options
dated	Append a date to the excel filename?
first_two_widths	I add long titles to the tops of the sheets setting this makes sure that those columns are not too wide
start_row	The first row of the sheet to write
start_col	The first column to write
...	the set of arguments given to for openxlsx

Value

a list containing the sheet and workbook written as well as the bottom-right coordinates of the last row/column written of the table.

See Also

openxlsx writeDataTable

Examples

```
## Not run:
 xls_coords <- write_xls(dataframe, sheet="hpgl_data")
 xls_coords <- write_xls(another_df, sheet="hpgl_data", start_row=xls_coords$end_col)

## End(Not run)
```

Index