dspNgs

June 17, 2020

Type Package

Title dspNgs Hydra Pipeline

Version 0.3.2

Author Hydra Team

Maintainer The package maintainer `<yourself@somewhere.net>`

Description More about what it does (maybe more than one line)
Use four spaces when indenting paragraphs within the Description.

License What license is it under?

Encoding UTF-8

biocViews Infrastructure, org.Hs.eg.db, ReactomePA, fgsea, GSVA, reactome.db

Depends R (>= 3.5.0),
circlize,
config,
dplyr,
edgebundleR,
EnvStats,
factoextra,
FactoMineR,
fgsea,
flux,
futile.logger,
GGally,
ggraph,
ggplot2,
ggrepel,
ggridges,
ggthemes,
GSVA,
igraph,
lattice,
lme4,
lmerTest,
migest,
org.Hs.eg.db,
parallel,
pheatmap,
plotly,

plotROC,
processx,
psych,
RColorBrewer,
ReactomePA,
reactome.db,
readxl,
reshape2,
rstudioapi,
Rtsne,
scales,
stringr,
tidyr,
umap,
pbapply

LazyData true

RoxygenNote 7.1.0

R topics documented:

check_input *Internal helper function for checking input paratmeters*

Description

Internal helper function for checking input paratmeters

Usage

```
check_input(
  de_results,
  norm_counts,
  samp_notes,
  grouping_var,
  base_level,
  control_var
)
```

convert_target_notes *convert_target_notes Converts target notes from Erin to DSP-DA format*

Description

convert_target_notes

Converts target notes from Erin to DSP-DA format

Usage

```
convert_target_notes(targetNotes)
```

Arguments

`targetNotes`	target notes data frame

Value

targetNotes data frame formatted like DSP-DA

Examples

```
targetNotes <- convert_target_notes(targetNotes)
```

`de_internal` *Internal lower-level function that actually computes the DE. returns the data.frame of results. This function does the following: 1. Transposes and transforms the entire norm_counts dataset. 2. Facets the dataframe into a list of data.frames of base_line vs non-baseline level 3. lapply over each element of 2. 4. parellel::parLapply over each gene in 3. 5. Collate data.frames (or lists) and return.*

Description

Internal lower-level function that actually computes the DE. returns the data.frame of results. This function does the following: 1. Transposes and transforms the entire norm_counts dataset. 2. Facets the dataframe into a list of data.frames of base_line vs non-baseline level 3. lapply over each element of 2. 4. parellel::parLapply over each gene in 3. 5. Collate data.frames (or lists) and return.

Usage

```
de_internal(
  norm_counts,
  samp_notes,
  grouping_var,
  base_level,
  control_var,
  n_processors,
  the_formula,
  show_pb
)
```

`de_logic` *Internal lower-level function that checks de_ressults attributes and runs de_internal if needed. Returns the de_results (whether it was computed, referenced by file, or referenced by object).*

Description

Internal lower-level function that checks de_ressults attributes and runs de_internal if needed. Returns the de_results (whether it was computed, referenced by file, or referenced by object).

Usage

```
de_logic(
  de_results,
  norm_counts,
  samp_notes,
  grouping_var,
  base_level,
  control_var,
  n_processors,
  the_formula,
  show_pb
)
```

`dsp_de_analysis`	*DSP NGS Analysis Pipeline for DSP NGS Analysis that creates ROC curves, calculates AUC, runs DE analysis and creates downstream visualizations Requires: rstudioapi to be installed prior to running on Rstudio*

Description

DSP NGS Analysis

Pipeline for DSP NGS Analysis that creates ROC curves, calculates AUC, runs DE analysis and creates downstream visualizations Requires: rstudioapi to be installed prior to running on Rstudio

Usage

```
dsp_de_analysis(
  de_results = NULL,
  norm_counts,
  genes,
  grouping_var,
  base_level,
  results_dir,
  de_dir,
  control_var,
  residuals_dir,
  qq_dir,
  pval_cutoff,
  fc_cutoff,
  samp_notes
)
```

Arguments

`config`	Configuration file, see config.yml example file
`DSPcounts`	DSP negative normalized counts spreadsheet with annotations

Value

Resulting visualizations and DE analysis

PNG file of ROC curve for top 5 genes with highest coeff

df of FC and Pvals above threshold specified in config file,

TODO

NA

Log Params

NA

Parsing Dataframe

NA

ROC curve

NA

DE Analysis

NA

Examples

```
#' @title DSP NGS Analysis
#'
#' Pipeline for DSP NGS Analysis that creates ROC curves, calculates AUC, runs DE analysis and creates downstrea
#' Requires: rstudioapi to be installed prior to running on Rstudio
#'
#' @param config Configuration file, see config.yml example file
#' @param DSPcounts DSP negative normalized counts spreadsheet with annotations
#'
#' @return Resulting visualizations and DE analysis
#'
#' @example R/DSP_NGS.R
#'
#' @export dsp_de_analysis
#'
#'
#' @section TODO
# #### plot GOI's vs all other genes, one color per GOI vs background
# #### label top 20 DE genes
# #### box plots of top 5 groups of genes by median of de significance (-log10P) + GOI (or custom set) on y vs gene
# Plot log normalized counts for genes as a function of gene_group

# main function
dsp_de_analysis <- function(de_results = NULL,
                            norm_counts,
                            genes,
                            grouping_var,
                            base_level,
                            results_dir,
                            de_dir,
                            control_var,
                            residuals_dir,
                            qq_dir,
                            pval_cutoff,
                            fc_cutoff,
                            samp_notes) {

  #' @section Log Params
  # Write input parameters to log file
 flog.info("\n \n ############  Experimental Parameters  ################## \n", name="DSP_NGS_log")
  flog.info(paste0("Gene(s) of interest: ", gene_group), name="DSP_NGS_log")
  flog.info(paste0("Gene grouping variable: ", grouping_var), name="DSP_NGS_log")
  flog.info(paste0("Base Level: ", base_level), name="DSP_NGS_log")
  flog.info(paste0("Control: ", control_var), name="DSP_NGS_log")
  flog.info(paste0("Pvalue Cutoff: ", pval_cutoff), name="DSP_NGS_log")
  flog.info(paste0("Fold Change Cutoff: ", fc_cutoff), name="DSP_NGS_log")
  flog.info(paste0("Gene Set Colors: ", color_genes), name="DSP_NGS_log")
  flog.info(paste0("Sample Colors: ", color_samples), name="DSP_NGS_log")

  # log2 normalize counts and transpose dataframe
  rownames(norm_counts) <- norm_counts$Gene
  tdf <- within(norm_counts, rm("Gene"))
  trans_df <- data.frame(t(tdf))
  gene_log2 <- data.frame(log2(trans_df))
  gene_log2$Sample_ID <- rownames(trans_df)

  #' @section Parsing Dataframe
  # Subset samp_notes for gene_grouping variable
  sub_list <-c("Sample_ID", eval(grouping_var), eval(control_var))
  samps <- samp_notes[,sub_list]

  # add gene_grouping labels
  goi_df <- merge(gene_log2, samps, by="Sample_ID")

  # convert labels to binary classifier
  class_label <- ifelse(goi_df[grouping_var] == eval(base_level), 1, 0)
  if (sum(class_label) == 0){
   print("Check that config base level variable matches a group in ROI/AOI annotations sheet")
   flog.error("Check that config base level variable matches a group in ROI/AOI annotations sheet", name="DSP_N
  }
  colnames(class_label) <- "level"
  goi_df <- cbind(goi_df, class_label) # add labels column to goi_df
  #goi_df$level <- factor(goi_df$level, levels=c(0,1), ordered="True")

  #' @section ROC curve
  #' @return PNG file of ROC curve for top 5 genes with highest coeff
  paste("Plot ROC Curve...")
  # plot ROC curve for top5 genes
  ROC_plot(goi_df, grouping_var, "poisson", gene_group, de_dir, genes)


  #' @section DE Analysis
  # Run LME4 linear mixed model on log negative normalized counts for genes of interest
  #' @return df of FC and Pvals above threshold specified in config file,
  # as well as residual and qqplots for goi

  # reset goi_df and let them loop through levels
  goi_df <- merge(gene_log2, samps, by="Sample_ID")

  run_de_file <- paste(getwd(), "de_analysis.R", sep="/")

  if(is.null(de_results)) {
    de_res <- run_DE(df = goi_df,
                     grouping_var = grouping_var,
                     control_var = control_var,
                     base_level = base_level,
                     residuals_dir = residuals_dir,
                     qq_dir = qq_dir,
                     pval_cutoff = pval_cutoff)
    results_file = paste(de_dir, "de_results.csv", sep="/")
    write.csv(de_res, results_file)
  } else if(is.character(de_results)) {
    de_res <- read.delim(de_results, sep = ',', header = TRUE, as.is = TRUE)
 } else if(is.data.frame(de_results) & all(c('FC','Pval','gene') %in% colnames(de_results))) {
    de_res <- de_results
  } else {
   stop('There is an issue with provided data frame, please cehck de_results variable\n')
  }
  #### uncomment for testing ###
  # saveRDS(de_res, file="de_res.R")
  # de_results <- readRDS(file = "de_res.R")

  # Generate volcano plot
  # return PNG and Plotly versions of Volcano Plot

  tests <- unique(de_res$test)
  for (test in tests) {
    de_vol <- de_res[de_res$test == test, ]
    paste("Generating volcano plot...")
    volcano_file <- paste0(de_dir, "/", test, "_", gene_group, "_volcano.", fileType)
    plot_volcano(de_results = de_vol,
                 base_level = base_level,
                 target_group = gene_group,
                 targets = genes,
                 fav_targets = fav_genes,
                 outfile = volcano_file,
                 point_color = color_genes,
                 n_targets = n_top,
                 plt_title = test,
                 color_list = colors$gene_groups)
  }
  return(de_res)
 }
```

---

dsp_de_analysis_parallel

DSP differential expression analysis (parallel)

Description

Wrapper function to run differential expression in parallel. This function is used prior to `dsp_de_analysis`.

Usage

```
dsp_de_analysis_parallel(
  de_results = NULL,
  norm_counts,
  samp_notes,
  grouping_var,
  base_level,
  control_var,
  de_dir,
  n_processors = 4,
  the_formula = "classic",
  show_pb = TRUE
)
```

Arguments

`de_results`	object that can be one of the following: 1. NULL. If not provided, the differential expression analysis will run 2. object of type `character` specifying the location of the previous saved DE results 3. object of type `data.frame` providing the loaded DE results
`norm_counts`	is `data.frame` containing the normalized count data to be analyzed. Each row corresponds to a gene of interest. The first column is assumed to be labeled 'Gene'. Each additional column corresponds to a Sample_ID. Sample_ID values have a matching row in the annotations objects (see `samp_notes` argument below).
`samp_notes`	is a `data.frame` that provides annotations. If DE is to be run (i.e., `!is.null(de_results)`), `samp_notes` must have the grouping_var column, with containing base_level factor, and it must have a control_var column.
`grouping_var`	is the column in the annotations data.frame (i.e., `samp_notes`) to use for grouping. This is the level 2 factor in the Mixed Effects model for which estimates are made. This can be a global object generated from `read_config()` if running the Hydra pipeline.
`base_level`	is a level in the `grouping_var` column used as base. This can be a global object generated from `read_config()` if running the Hydra pipeline.
`control_var`	is the column name in the annotations data.frame (i.e., `samp_notes`) used at the level 1 random effect. For DSP, this is usually an individual ID for which several AOIs are nest within. If it is invariable (e.g., a vector of 1s), the default behavor is to run a linear model using R's base::lm function. Otherwise, a Mixed-effects model will be used via lme4::lmer.
`de_dir`	the parent directory to store the de_results
`n_processors.`	The number of processors to use. Default is 4.
`the_formula.`	Either the classic formula (default) or a custom formula. Do to customization possibilities, there is no guarantee that this will run, converge, or make sense. Example of custom formula = 'a_gene ~ Disease * Tissue + (1 + Disease | DSP_scan)'. Note that 'a_gene' should be left as is.
`show_pb.`	Logical to show the progress bar (TRUE) or not (FALSE). Default is TRUE.

Details

The function strips down the main `dsp_de_analysis` function and performs the differential expression analysis over a specified number of processors. It uses parallel::parLapply to process the normalized expression for each gene across different groups of interest (i.e., as specified by the `grouping_var` object). By default, this function runs the same as `dsp_de_analysis` in terms of the formulas used. If the `control_var` column is invariant. A simple linaer model will be used. Otherwise, a Mixed effects model will be used of the form 'a_gene ~ grouping_var + (1 + grouping_var | control_var)'. In both of these scenarios, a data.frame will be saved to disk and returned. If, however, `the_formula` differs from its default value of "classic", a custom formula will be used. In such a case, a list of list where the final list provides the 1) gene, the 2) test (e.g., 'normal vs disease'), and the model. There is no error checking for convergence etc. if the_formual != "classic". In all Mixed effect models, REML=TRUE.

Value

Objects and files returned depend on the input:

1. If `the_formula=="classic"`, an object of class `data.frame` is returned. The dataframe is also sent to disk.
2. Otherwise, a list of list is return (see details)

Author(s)

Tyler Hether

See Also

`dsp_de_analysis`

Examples

```
# Simulate 2 genes in log2 space. Look at the true fold
# change (for gene_a) and compare to the estimated fold change.
n_samples <- 50
n_aois <- 10
constant <- 4.5
# Simulate normalized, log2
make_aois <- function(mu, sd, n_aois, group){
 out <- data.frame(Group=rep(group, n_aois),
     gene_a=rnorm(n=n_aois, mean=mu, sd=sd))
     return(out)
}
# Different effect size of group A from mean
diff_a <- rnorm(n=n_samples, mean=2, sd=1)
# Make samples.
A <- do.call(rbind, lapply(1:n_samples, function(i){
 samp <- make_aois(constant+diff_a[i], 1, n_aois, "A")
 samp$ind <- i
 return(samp)
}))
diff_b <- rnorm(n=n_samples, mean=-2, sd=1)
B <- do.call(rbind, lapply(1:n_samples, function(i){
 samp <- make_aois(constant+diff_b[i], 1, n_aois, "B")
 samp$ind <- i+max(A$ind)
 return(samp)
}))
df <- rbind(A, B)
require(plyr)
require(dplyr)
df <- ddply(df, .(Group, ind), function(x){
 x$aoi <- 1:nrow(x)
 return(x)
})
df <- df %>% mutate(Sample_ID=paste(Group, ind, aoi, sep="_"), gene_b=gene_a+rnorm(1))
# Make sure there's no negative values.
df$gene_a[which(df$gene_a<=0)] <- 1e-03
df$gene_b[which(df$gene_b<=0)] <- 1e-03
library(tidyr)
norms <- rbind(spread(df %>% dplyr::select(-aoi, -gene_b, -Group, -ind), Sample_ID, gene_a),
 spread(df %>% dplyr::select(-aoi, -gene_a, -Group, -ind), Sample_ID, gene_b))
```

```
# "Convert" to linear scale since the function will log2 transform internally
norms <- 2^norms
# Combine
norms <- cbind(data.frame(Gene=c("gene_a", "gene_b")), norms)
row.names(norms) <- norms$Gene
# Convert the df to respective dataframes
annotations <- df %>% select(Sample_ID, Group, ind)
library(ggplot2)
ggplot(df, aes(x=factor(ind), y=gene_a)) +
geom_jitter(height=0) + ylab("log2 expression for simulated gene_a")
# Means of the groups in log2 space
means <- ddply(df, .(Group), summarize, mean(gene_a))[,2]
# Fold change where vec is the means in log2 space
# returns FC in linear space
calc_fc <- function(vec){
 vec2 <- 2^vec
 return(vec2[2] / vec2[1])
}
# This is the simulated fold change in log2 space
print(log2(calc_fc(means)))
#Compare that fold change to the estimate for gene_a
library(dspNgs)
dsp_de_analysis_parallel(de_results=NULL, norm_counts=norms,
samp_notes=annotations, grouping_var="Group", base_level="A",
 control_var="ind", de_dir="./", n_processors=4, the_formula="classic")
# End run.
```

find_topGS *find_topGS Generic call to heatmap functionality*

Description

find_topGS

Generic call to heatmap functionality

Usage

```
find_topGS(
  targetNotes = NULL,
  de_results = NULL,
  name = NULL,
  outdir = NULL,
  fileType = NULL,
  width = 1500,
  height = 850
)
```

Arguments

targetNotes	target annotations - list of lists of genes output from parse_GeneSets
de_results	table of DE results
name	annotations to include on heatmap

`outdir`	where to print file
`fileType`	function to call for file printing
`width`	file width
`height`	file height

`parallel_read` *Reads in DSP dataset*

Description

Reads in dsp excel workbook for DE analysis and visualiation

Usage

```
parallel_read(file)
```

Arguments

`data_path`	path to dsp dataset excel workbook

Value

list of dataframes, one for each tab

Examples

```
read_dataset(data_path)
```

`plot_ROI2D` *ROI Plot*

Description

Takes data matrix, including expression, signatures, pathways and segment annotations, and creates graphs based on the ROI ID

Usage

```
plot_ROI2D(
  data_df = NULL,
  annot_df = NULL,
  title = NULL,
  plot_type = "joy",
  skip_ROIs = TRUE,
  targets = "median",
  target_thresh = 0.5,
  target_quantile = TRUE,
  plot_clusters = TRUE,
  cluster_pal = "Set1",
  cluster_method = "complete",
  cluster_dist = "pearson",
  cluster_n = 6,
  cluster_min = 20,
  ROI_ID = "ROI_ID",
  AOI_ID = "Sample_ID",
  segment_ann = "Segment",
  segment_names = c("Tumor", "Stroma"),
  segment_colors = c("green3", "magenta3"),
  transf = NULL,
  cluster_transf = NULL,
  scale = TRUE,
  shape = "circle",
  hole_diameter = 5,
  width = 1,
  draw_bgd = TRUE,
  split_groups = TRUE,
  bgd_color = "white",
  bgd_fill = alpha("black", 0.5),
  joy_scale = 5,
  joy_alpha = 0.6,
  joy_lwd = 0.5,
  heat_theme = "Dark",
  heat_alpha = c(0, 0.9),
  heat_color = alpha("white", 0.5),
  ...
)
```

Arguments

`data_df`	a data frame of targets / pathways / signatures scores / cell types vs AOIs to be plotted
`annot_df`	a data frame of AOI annotations necessary to plot the data
`title`	title to be added after ROI ID to each plot
`plot_type`	type of plot, values include 'joy', 'bar', 'heatmap'
`skip_ROIs`	convenience to just plot legend & grab gene sets
`targets`	subset list or logic, either vector of booleans, character vector, or function for subseting including 'CV', 'IQR', 'sd', and other standard functions
`target_thresh`	if a function is provided, use this threshold value to pick samples with value > thresh
`target_quantile`	whether the target_thresh is a value or a quantile to filter to
`plot_clusters`	whether clusters should be plotted in the static graph as well
`cluster_pal`	cluster palette to use when plotting
`cluster_method`	clustering type, 'average', 'complete', or 'ward' recommended
`cluster_dist`	clustering distance, 'pearson', 'spearman', or values used by dist
`cluster_n`	number of clusters to idenitfy
`cluster_min`	the minimum size of a cluster for it to be considered distinct, otherwise breaks will be plotted with a neighboring cluster

`ROI_ID`	name of ROI column or columns, list if multiple
`AOI_ID`	name of AOI column
`segment_ann`	segment annotation column
`segment_names`	labels for segment, must be labels within the segment_ann column
`segment_colors`	colors for segments used in joy plots and heatmap, not bar
`transf`	'log2','logit','fraction','scale'
`cluster_transf`	'log2','logit','fraction','scale'
`shape`	type of plot to make based on original AOI shape - 'circle' or 'rect', rectangular is still in development
`hole_diameter`	relative center diameter for window into ROI
`width`	relative plotting area width
`draw_bgd`	whether to draw the background behind plot if 'joy' or 'bar' used
`split_groups`	whether to add a buffer between clusters and draw backgrounds seperately
`bgd_color`	color for the border around backgrounds
`bgd_fill`	color for the background fill
`joy_scale`	scaling factor for joy plot
`joy_alpha`	alpha for the fill of the joy plot
`joy_lwd`	lwd for the joyplot
`heat_theme`	Dark' = blacks with alpha, 'Light' = whites with alpha
`heat_alpha`	range of alpha to used with the heatmap. 0 is pure color and 1 is a total overlay of the color
`heat_color`	color for the border around the heatmap

Value

a list of ROI plots and legends to be subsequently saved to pdf or plotted with ggplotly

---

plot_volcano *Volcano Plot*

Description

Takes results of differential expression analysis and generates a volcano plot

Usage

```
plot_volcano(
  de_results = NULL,
  base_level = NULL,
  target_group = "All Probes",
  fav_targets = NULL,
  targets = NULL,
  outfile = "VolcanoePlot-AllGenes",
  point_color = "RdPu",
  top_method = "Significance",
  n_targets = 15,
  plt_title = "Volcano Plot for All Genes",
  target_ID = "gene",
  FC_ID = "FC",
  Pval_ID = "Pval",
  color_list = NULL,
  save_plot = TRUE
)
```

Arguments

`de_results`	output of dsp_de_analysis function with list of up and downregulated targets, e.g. genes or gene sets
`base_level`	baseline level for DE analysis, used for plotting x-axis label
`target_group`	name of target groups, or , used for adding color contrast to graph
`targets`	a list of targets, e.g. genes, within the target_group, used for subsetting graph
`outfile`	name of file for output
`point_color`	a specific color to be used for plotting points, should be updated to a gradient method
`top_method`	column within the de_results that should be used to select the top genes, selecting highest values from the column. Use Significance rather than Pval for this purpose
`n_targets`	number of genes to show in the cloud
`target_ID`	column within the de_results data frame that contains the target names to be used for labeling graphs
`FC_ID`	column within the de_results data frame that contains the FC x-axis values to be used for plotting
`Pval_ID`	column within the de_results data frame that contains the P-value y-axis values to be used for plotting
`color_list`	the targetset color values if multiple targetsets are being used, can be NULL if only one targetset passed
`save_plot`	boolean of whether to save the plot as a file or just output to the console, defaults to TRUE and uses config information for file format

Value

image file of volcano plot

plotly interactive volcano plot

Examples

```
plot_volcano(de_results, base_level)
```

`qc_plots` *QC_plots Creates QC plots of normalization factors and distributions of counts*

Description

QC_plots

Creates QC plots of normalization factors and distributions of counts

Usage

```
qc_plots(
  data_matrix,
  norm_method = NULL,
  grp_var = NULL,
  ctrl_var = NULL,
  HKs = NULL,
  outdir,
  colors
)
```

Arguments

`data_matrix`	the data matrix list
`norm_method`	the selected norm method for output data matrix
`grp_var`	the selected column name for the grouping analysis
`HKs`	the list of housekeeping genes
`outdir`	the output directory for plots
`colors`	the list of specified colors
`ctr_var`	the selected column name to control by for DE

Value

norm_counts chosen normalization method

Examples

```
qc_plots(data_matrix = dfs, norm_method = "Q3", outdir = od)
```

`read_config` *DSP_utils Collection of DSP_hydra functions*

Description

DSP_utils

Collection of DSP_hydra functions

Usage

```
read_config()
```

`ROC_plot` *Plot ROC Curve*

Description

Transposes counts df and creates ROC curve plot with top 5 genes with largest coefficients

Usage

```
ROC_plot(df, grouping_var, family, gene_group, outdir, genes)
```

Arguments

`df`	negative normalized DSP count data transposed to wide format, one column per gene
`grouping_var`	column from annotations to group by
`family`	glm distribution, aka poisson
`gene_group`	gene group of interest
`outdir`	path to output directory
`genes`	list of genes of interest

Value

image file of top 5 genes ROC curve

dataframe of AUC for each gene

Examples

```
ROC_plot(top5_coeff, gene_group, outdir)
```

`run_cell_decon` *run_cell_decon Runs our internal immune cell deconvolution algorithm, "InSituSort"*

Description

run_cell_decon

Runs our internal immune cell deconvolution algorithm, "InSituSort"

Usage

```
run_cell_decon(
  df = dfs,
  norm_method,
  outdir = cellType_dir,
  tumor_high_ids = NULL,
  tumor_low_ids = NULL,
  x = NULL,
  y = NULL,
  xlab = "",
  ylab = ""
)
```

Arguments

`norm_method`	Raw data matric
`tumor_high_ids`	Names of AOIs with nearly 100 If NULL, the decon model will ignore tumo-intrinsic expression, which usually works just fine.
`tumor_low_ids`	Names of AOIs with low tumor content. Must be elements of the colnames of norm_counts. If NULL, the decon model will ignore tumo-intrinsic expression, which usually works just fine.
`x`	Horizontal coordinates on which to plot decon results (e.g. from t-SNE) (optional)
`y`	Vertical coordinates on which to plot decon results (e.g. from t-SNE) (optional)
`xlab`	Name of the x-dimension on which results are plotted
`ylab`	Name of the y-dimension on which results are plotted
`counts`	Raw data matrix
`norm_counts`	Normalized data matrix
`samp_notes`	Raw data matric
`probe_notes`	Raw data matric

Value

clustering information

Examples

```
deconresults <- run_cell_decon(df = dfs,
               norm_method = "Neg", outdir = cellType_dir,
               tumor_high_ids = NULL, tumor_low_ids = NULL,
               x = tsne.result[,1], y = tsne.result[,2],
               xlab = "tSNE dim 1", ylab = "tSNE dim 2")
```

`run_DE` *DE Analysis*

Description

Runs a linear mixed effects model on the dataset to analyze differential expression using the LME4 package

Usage

```
run_DE(
  df,
  control_var,
  grouping_var,
  base_level,
  residuals_dir,
  qq_dir,
  pval_cutoff,
  make_plots = TRUE,
  random_slope = TRUE
)
```

Arguments

`df`	negative normalized DSP count data transposed to wide format, one column per gene
`control_var`	column from annotations to use as experimental control variable
`grouping_var`	column from annotations to test during DE analysis
`base_level`	variable to use as base level for testing, all other vars looped through during analysis
`residuals_dir`	path to output directory image subfolder for residual images
`qq_dir`	path to output directory image subfolder for qq plots
`pval_cutoff`	pvalue cutoff for significance from config file
`make_plots`	logical as to whether qq & residual plots should be written
`random_slope`	logical to turn on or off random slopes depending on study

Value

PNG file of top 5 genes ROC curve

dataframe of AUC for each gene

Examples

```
run_DE(df, control_var, base_level, genes,residuals_dir, qq_dir, pval_cutoff)
```

---

run_GSEA *run_GSEA Pathway Analysis helper function to run GSEA*

Description

run_GSEA

Pathway Analysis helper function to run GSEA

Usage

```
run_GSEA(
  geneList = geneList,
  pathways = pathways,
  test = test,
  outdir = outdir
)
```

Arguments

`geneList`	a list of ranked genes, the way that they are ranked is determined in run_pathways.R
`pathways`	are the genesets that we are working with, default is Reactome
`test`	is the test or contrast that is being compared for pathway analysis, from DE results
`outdir`	the output directory for plots

Value

fgseaRes The results from GSEA analysis

Examples

```
run_GSEA(geneList = geneList, pathways = pathways, test = test, outdir = outdir)
```

run_heatmap *run_heatmap Generic call to heatmap functionality*

Description

run_heatmap

Generic call to heatmap functionality

Usage

```
run_heatmap(
  data = NULL,
  data_TargetID = "Gene",
  annot = NULL,
  samp_vars = NULL,
  fav_genes = NULL,
  ann_colors = NULL,
  logData = TRUE,
  targets = NULL,
  name = NULL,
  scale = "row",
  outdir = ".",
  fileType = "tiff",
  sortBy = NULL,
  sortBaseline = NULL,
  ...
)
```

Arguments

`data`	the data to plot, assumed targets * samples
`annot`	sample annotations
`samp_vars`	annotations to include on heatmap
`fav_genes`	favorite genes to highlight
`ann_colors`	list of lists of colors
`logData`	T/F - log2 transform data
`targets`	used to subset dataset to only genes of interest or NULL for all
`name`	title of graph and file
`scale`	passed to pheatmap
`outdir`	where to print file
`fileType`	function to call for file printing
`sortBy`	annotation variable to use for sorting the heatmap
`width`	file width
`height`	file height

run_ora	*run_ora Pathway Analysis helper function to run Over Representation Analysis*

Description

run_ora

Pathway Analysis helper function to run Over Representation Analysis

Usage

```
run_ora(
  geneList = geneList,
  pathways = pathways,
  test = test,
  path_fc = path_fc,
  path_pval = path_pval,
  outdir = outdir
)
```

Arguments

`geneList`	a list of ranked genes, the way that they are ranked is determined in run_pathways.R
`pathways`	are the genesets that we are working with, default is Reactome
`test`	is the test or contrast that is being compared for pathway analysis, from DE results
`path_fc`	the fold change cutoff to use for gene list
`path_pval`	the pvalue to threshold the enrichment results by
`outdir`	the output directory for plots

Value

ora The results from over representation analysis

Examples

```
run_ora(geneList = geneList, pathways = pathways, test = test, path_fc = path_fc, path_pval = path_pval, outdir
```

run_pathways *run_pathways Pathway Analysis giving either coverage or GSEA*

Description

run_pathways

Pathway Analysis giving either coverage or GSEA

Usage

```
run_pathways(
  norm_data,
  de_results,
  geneListrank,
  path_fc,
  path_pval,
  enrichment,
  outdir,
  custom_pathways_path = NULL,
  exclude_reactome = FALSE,
  ...
)
```

Arguments

`norm_data`	the chosen normalized data
`de_results`	the differential expression data
`path_fc`	the fold change cutoff to use for gene list
`path_pval`	the pvalue to threshold the enrichment results by
`enrichment`	the type of enrichment to be returned
`outdir`	the output directory for plots
`custom_pathways_path`	the directory where custom gmt files are located
`exclude_reactome`	to exclude reactome pathways from custom pathway analysis
`genListrank`	the way to rank the gene list either by foldchange, pval or FDR

Value

enrich_res The results from your chosen enrichment

Examples

```
 run_pathways(norm_data = norm_counts, de_data = de_results, geneListrank=geneListrank, path_fc=path_fc, path_
```

run_PCA *run_PCA Runs principle component analysis (PCA) on dsp data*

Description

run_PCA

Runs principle component analysis (PCA) on dsp data

Usage

```
run_PCA(
  df,
  outdir,
  color = "cluster",
  symbol = "dsp_slide",
  size = TRUE,
  colors = colors
)
```

Arguments

df	read in dsp excel workbook dataset
outdir	output directory for figures
color	category to color the graph by
symbol	category to change symbols in the graph
size	should the size of the points change by gene_count
outlier_cutoff	multiplier for outlier cutoff

Value

clustering information

Examples

```
pca <- run_PCA(df, "file/path", outlier_cutoff)
```

run_Q3 *run_Q3 Run Q3 normalization*

Description

run_Q3

Run Q3 normalization

Usage

```
run_Q3(dataset = dfs)
```

Arguments

`dataset`	the full dataset

Value

dfs the new data object post q3 normalization, updated norm counts and properties with new q3 norm factor

Examples

```
run_Q3(dataset = dfs)
```

`run_seqQC` *DSP-NGS Sequencing QC*

Description

Generates sequecing QC metrics for provided dataset

Usage

```
run_seqQC(
  dataframelist,
  loq = 2.5,
  qc_dir = "./",
  facet_column = "",
  fileType = "pdf"
)
```

Arguments

`dataframelist`	List of all 5 dataframes
`loq`	Number of standard deviations desired for loq. Column must exist as GeoLOQ|SD|_|Pool| for each pool in the dataset, ex. GeoLOQ2.5_01. Default = 2.5
`qc_dir`	Directory to output plot + QC table. Default = './'
`facet_column`	Column to facet the line plot on if desired. Default = No facet.
`fileType`	Type of file for ggsave to generate for the QC Line plot.

Value

Write PDF/PNG/... file with the QC line plot

Write QC metrics to file in QC Directory

Examples

```
run_seqQC(dataframelist=dfs, loq=2.5, qc_dir='/QC_Dir/', facet_column='dsp_slide', fileType='pdf')
```

run_ssGSEA *run_ssGSEA Pathway Analysis helper function to run ssGSEA*

Description

run_ssGSEA

Pathway Analysis helper function to run ssGSEA

Usage

```
run_ssGSEA(
  norm_data = norm_data,
  pathways = pathways,
  test = test,
  outdir = outdir
)
```

Arguments

`norm_data`	the normalized log2 transformed data matrix
`pathways`	are the genesets that we are working with, default is Reactome
`test`	is the test or contrast that is being compared for pathway analysis, from DE results
`outdir`	the output directory for plots

Value

gsvaRes The results from ssGSEA analysis

Examples

```
run_ssGSEA(norm_data = norm_data, pathways = pathways, test = test, outdir = outdir)
```

run_tsne *run_tsne Runs t-Distributed Stochastic Neighbor Embedding (tsne) on dsp data*

Description

run_tsne

Runs t-Distributed Stochastic Neighbor Embedding (tsne) on dsp data

Usage

```
run_tsne(
  df,
  pca = NULL,
  outdir,
  color = "cluster",
  symbol = "dsp_slide",
  size = TRUE,
  perplexity = 30,
  colors = colors
)
```

Arguments

`df`	read in dsp excel workbook dataset
`outdir`	output directory for figures
`color`	category to color the graph by
`symbol`	category to change symbols in the graph
`size`	should the size of the points change by gene_count
`outlier_cutoff`	multiplier for outlier cutoff

Value

none

Examples

```
run_tsne(df, "file/path", outlier_cutoff, color = "slide_id", symbol = "TIS", size = TRUE)
```

---

run_umap *run_umap Runs t-Distributed Stochastic Neighbor Embedding (tsne) on dsp data*

Description

run_umap

Runs t-Distributed Stochastic Neighbor Embedding (tsne) on dsp data

Usage

```
run_umap(
  df,
  pca = NULL,
  outdir,
  color = "cluster",
  symbol = "dsp_slide",
  size = TRUE,
  colors = colors
)
```

Arguments

`df`	read in dsp excel workbook dataset
`outdir`	output directory for figures
`color`	category to color the graph by
`symbol`	category to change symbols in the graph
`size`	should the size of the points change by gene_count
`outlier_cutoff`	multiplier for outlier cutoff

Value

none

Examples

```
run_umap(df, "file/path", outlier_cutoff, color = "slide_id", symbol = "TIS", size = TRUE)
```

`send_to_log` *Internal helper function for writing to log.*

Description

Internal helper function for writing to log.

Usage

```
send_to_log(grouping_var, base_level, control_var)
```

Index