‘DFDgeneration’

June 14, 2024

Version 1.0

Date 2024-06-14

Title Locating and Analyzing Domains of Focal Deregulation in RNA-seq count data

Description Locating and Analyzing Domains of Focal Deregulation in RNA-seq count data. DFDgeneration contains modules for locating DFDs, creating DFD descriptives, Clustering and XGBoost classification based on various DFD properties, Analyzing DEG enrichments in DFDs and Jaccard Similarity Index, Creating Bipartite functional-positional networks, Bipartite functional-positional network metrics analysis and XGBoost classification, Bipartite functional positional network visualization. The DFDgeneration toolkit features the creation of output files, metrics and plots and utilizes existing methods like the F test (Chow test) framework, Gene Ontology Enrichments (gprofiler2), igraph, XGBoost. Specials efforts have been made to visualize bipartite functional-positional networks to extract meaningful biological output.

LazyData yes

Depends/Imports R (>= 2.10.0), TTR, strucchange, xgboost, caret, pROC, ROCit, tidyverse, reshape, dendextend, stats, ggpubr, gprofiler2, igraph, rgl, viridis, gplots, gridExtra, ggplot2, tsne

Table of Contents

1. Module 1: Creation of z-score count matrix from input data

1.1. create_zeta_log_table

Description
This function creates receives a count read file and outputs a log10 z-score table count file. Keep in mind that this step can be omitted if a different type of normalization and/or standardization is desired. Initially the log10 of the counts is calculated and then the counts are converted to z-score. Encode gene IDs are added along with the genomic coordinates and gene names.

Usage
create_zeta_log_table(initial_count_file, gene_names_path, count_table_path)

Arguments

initial_count_file	A path to a file containing the input RNA-seq read counts
gene_names	A file containing the following columns: “Gene stable ID”, “Gene stable ID version”, “Gene start (bp)”, “Gene end (bp)”, “Gene name”, “Version (gene)”, “Chromosome/scaffold name”
count_table_path	A path to save the output file

Value
Output file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start coordinates, #end coordinates, #chromosome

Examples

```
##Use counts.csv as input read counts file
##Use encode_genes.txt as gene names file
##Save output file to Desktop
create_zeta_log_table("/home/$USER/Desktop/counts.csv",
"/home/$USER/Desktop/encode_genes.txt", "/home/$USER/Desktop/")
```

2. Module 2: DFD Creation

2.1. create_DFDs

Description
This module creates breakpoints with the strucchange package and extracts DFDs (Domains of Focal Deregulation) from those breakpoints based on a user-defined average expression change threshold.

Usage
create_DFDs(count_matrix_file_path, breakpoint_path, DFD_path, chromosome_levels, significance_threshold = 0.1)

Arguments

count_matrix_file_path	A path to an input file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start, #end, #chromosome
breakpoint_path	A path where all the strucchange output breakpoint files will be stored
DFD_path	A path where all the output DFD files will be stored
chromosome_levels	The chromosome identifiers of the organism in question
significance_threshold	A mean gene expression value threshold above the absolute value of which (over-and-under expression) a breakpoint will have a significant enough value to constitute a DFD. Default value is **0.1**.

Value
Genomic coordinate files containing the breakpoints (chow test) and the DFDs created for each experiment

See also
convert_to_bed

Examples

```
##Use file count_table as input in the format described in the arguments above
##Compute and save breakpoints in the breakpoints folder in Desktop
##Compute and save DFDs in the DFDs folder in Desktop
##Use 0.15 (15% expression change from breakpoint to breakpoint) as a mean expression change
##threshold
create_DFDs("/home/$USER/Desktop/count_table.txt","/home/$USER/Desktop/breakpoints/","/
home/$USER/Desktop/DFDs/",0.15)
```

2.2. convert_to_bed

Description
This is a complementary function that will convert the DFD files produced earlier to bed format.

Usage
convert_to_bed(DFD_path, bed_path, number_of_files)

Arguments

DFD_path	A path where all the DFD files are stored
bed_path	A path where all the bed files will be stored
number_of_files	Number that will be read sequentially to access and convert all DFD files created with create_DFDs

NOTE
The DFD files should named #DFD_path/$number_tot.txt#. If a different convention is followed the code will have to be modified to be properly executed.

Value
bed files containing the DFDs

See Also
create_DFDs

Examples
##We obtain the DFD files from the “DFDs” folder in the user’s Desktop
##We store the bed files in the “DFDs_bed” folder in the user’s Desktop
##We convert 9 DFD files to bed
convert_to_bed("/home/$USER/Desktop/DFDs/", "/home/$USER/Desktop/DFDs_bed", 9)

3. Module 3: DFD Descriptives

3.1. DFD_descriptives

Description
This module provides basic descriptive statistics for DFDs. Number of DFDs as well as number of genes and DEGs in DFDs, percentage of total genome covered by DFDs, and an odds ratio (deg_number/gene_number) / (genome_cover_percentage) are calculated. The module performs Kruskal-Wallis ANOVA in the odds ratios of different user-provided categories and the p-value is included in the return list. The module also performs Kruskal-Wallis ANOVA in the chromosome cover percentages of different user-provided categories and the vector containing all p-values is included in the return list. Finally, the module produces various plots to aid in the visualization of the descriptive statistics provided.

Usage
DFD_descriptives(num_of_experiments, chrom_length_path, chromosome_levels, DFD_path, plot_path, count_matrix_file, categories_index, deg_threshold)

Arguments

num_of_experiments	Number of experiments in the input count matrix
chrom_length_path	path to a file containing the lenghts of the chromosomes of the organism in question
chromosome_levels	The chromosome identifiers of the organism in question
DFD_path	A path where all the DFD files are stored
plot_path	A path were all the output plots are stored
count_matrix_file	A path to an input file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start, #end, #chromosome
categories_index	A vector file of a single column where each row contains the category value of the category corresponding to the experiment
deg_threshold	An expression value threshold, such that IF abs(gene1_expression) >= deg_threshold THEN gene1 = DEG

NOTE
The DFD files should named #DFD_path/$number_tot.txt#. If a different convention is followed the code will have to be modified to be properly executed.

Value
The output is a list containing:

1) the absolute number of DFDs per experiment
2) the absolute number of genes in DFDs per experiment
3) the absolute number of DEGs in DFDs per experiment
4) the genome coverage percentage of DFDs per experiment
5) the odds_ratio per experiment (deg_number/gene_number) / (genome_cover_percentage)

6) the odds_ratio p-value after Kruskal-Wallis ANOVA between the different categories
7) matrix of cover percentage per chromosome per experiment
8) the chromosome cover percentage p-value vector after Kruskal-Wallis ANOVA between the different categories

And the following plots saved in the user-defined folder:

1) mean DFD number per chromosome boxplot
2) DFD number per experiment boxplot
3) DFD number per experiment histogram
4) DFD length boxplot
5) genome cover percentage histogram
6) genome cover percentage boxplot
7) genome cover percentage per category boxplot
8) odds ratio boxplot
9) total chromosome cover percentage boxplot
10) separate boxplots of chromosome cover percentages per category

See also

deg_enrichments_DFDs

Examples

```
##In this example we have 45 experiments
##We obtain the lengths of the chromosomes from the file “chromosomes_lengths.txt” in the user’s
##Desktop
##We have human genome and use all chromosomes excluding Y
##We obtain the DFD files from the “DFDs” folder in the user’s Desktop
##We store the plots in the “plots” folder in the user’s Desktop
##Use file “count_table.txt” as input in the format described in the arguments above
##Use file “categories.txt” as input to compare between experiments in the format described in the
##arguments above
##Give a threshold of 1.5, so that if the absolute value of the normalized counts is > 1.5 we define
##this gene as a DEG
DFD_descriptives(45, “/home/$USER/Desktop/chromosomes_lengths.txt, c(seq(1,22),”X”),
"/home/$USER/Desktop/DFDs/", "/home/$USER/Desktop/plots/",
"/home/$USER/Desktop/count_table.txt", "/home/$USER/Desktop/categories.txt", 1.5)
```

3.2. deg_enrichments_DFDs

Description
This function will work in conjunction with the bedoverlapper.sh script (Andreadis C, Nikolaou C, Fragiadakis GS, Tsiliki G, Alexandraki D. Rad9 interacts with Aft1 to facilitate genome surveillance in fragile genomic sites under non-DNA damage-inducing conditions in S. cerevisiae. Nucleic Acids Res 2014;42(20):12650–67). Assuming that overlap of all the DEGs with the DFDs is provided as input, this function will compare the overlap of DEGs with DFDs and use Kruskal-Wallis ANOVA to detect statistically significant differences between the different experiment categories that the user supplies.

Usage
deg_enrichments_DFDs(DEG_DFD_Enrichment_path, categories_index)

Arguments

DEG_DFD_Enrichment_path	Path to a file containing the enrichments output of bedoverlapper.sh. The path should have the following format: No column names, #File1 = DEGsX.bed, #File2 = DFDsX.bed, #Ratio 1 = R1, #Ratio = R, #Enrichment = E, #p-value = pval. The bedoverlapper.sh script should produce a similar output upon completion. Some manipulation is needed to remove intermediate lines and only keep the relevant information (overlap_convert function)
categories_index	A vector file of a single column where each row contains the category value of the category corresponding to the experiment

Value
The output is a list containing:
1) Kruskal-Wallis p-value of all DEG-DFD enrichments
2) Kruskal-Wallis p-value of significant DEG_DFD enrichments

See Also
bedoverlapenr.sh, overlap_convert

Examples
```
##Use the file  “DEG_DFD_Enrichments.txt” as input containing the overlap enrichments as was
##described in the “Arguments” section
##Use file “categories.txt” as input to compare between experiments in the format described in the
##arguments above
enrichment_pval <-
deg_enrichments_DFDs("/home/$USER/Desktop/DEG_DFD_Enrichments.txt",
"/home/$USER/Desktop/categories.txt")
```

4. Module 4: Clustering and XGBoost classification based on various DFD properties

4.1. dfd_Cluster

Description
This module attempts to cluster experiments based on DFDs. This module is relevant only if the user inputs different categories for the dataset. Ideally the clusters should coincide with the original experiment categories. The module outputs cluster tables and plots. The model also offers XGBoost classification and validation of the model.

Usage
dfd_Cluster(genome_cover_percentage,dfd_num,gene_num, tot_chrom_cover_percentage, categories_index, plot_path, tsne = 0, dfds_overlap = 0)

Arguments

genome_cover_precentage	the genome coverage percentage of DFDs per experiment, as output by DFD_descriptives
dfd_num	the absolute number of DFDs per experiment, as output by DFD_descriptives
gene_num	the absolute number of genes in DFDs per experiment, as output by DFD_descriptives
tot_chrom_cover_percentage	matrix of cover percentage per chromosome per experiment, as output by DFD_descriptives
categories_index	A vector file of a single column where each row contains the category value of the category corresponding to the experiment
plot_path	A path were all the output plots are stored
xgboost	Binary variable, 0 will not run an XGBoost model, 1 will use XGBoost. Default is 0. NOTE: We use 10-fold cross-validation, so for datasets that cannot meaningfully be split into 10 this feature cannot be used
tsne	Binary variable, 0 will not use t-sne reducted tables in the XGBoost model, 1 will use t-sne reduction. Default is 0
dfds_overlap	Path to a file containing the all vs all experiments DFDs ovelap matrix, as output by bedoverlapper.sh. If tsne variable == 1 a file must be provided or an error will occur

Values
The output is a list containing:

1) RMSE of the XGBoost model
2) R^2 of the XGBoost model
3) MAE of the XGBoost model
4) AUC of the XGBoost model
5) Cluster tables for clustering based on DFD genome coverage
6) Cluster tables for clustering based on DFD chromosome coverage

And the following plots saved in the user-defined folder:

1) Clusters based on DFD genome coverage barplot
2) Clusters based on DFD chromosome coverage barplot
3) Clusters based on DFD chromosome coverage heatmap

See Also

Module 2: DFD Descriptives, bedoverlapper.sh

Examples

```
##Example 1, NO tsne
##Use cover percentage, dfd_num, gene_num and tot_chrom_cover_precentage as output from the
##DFD_descriptives function
##We store the plots in the “plots” folder in the user’s Desktop
##Use file “count_table.txt” as input in the format described in the arguments above
##Use file “categories.txt” as input to compare between experiments in the format described in the
##arguments above
##No tsne for XGBoost classification, variable is ommited so default value of 0 will be used and no
##overlap matrix file is provided
BoostResults <- dfd_Cluster(descriptives$cover_percentage, descriptives$dfd_num,
descriptives$gene_num, descriptives$tot_chrom_cover_percentage,
"/home/$USER/Desktop/categories.txt", "/home/$USER/Desktop/plots")

##Example 2, YES tsne
##Use cover percentage, dfd_num, gene_num and tot_chrom_cover_precentage as output from the
##DFD_descriptives function
##We store the plots in the “plots” folder in the user’s Desktop
##Use file “count_table.txt” as input in the format described in the arguments above
##Use file “categories.txt” as input to compare between experiments in the format described in the
##arguments above
##Use t-sne for XGBoost classification (tsne = 1)
##Overlap matrix file is provided, DFDs_overlap.txt
BoostResults <- dfd_Cluster(descriptives$cover_percentage, descriptives$dfd_num,
descriptives$gene_num, descriptives$tot_chrom_cover_percentage,
"/home/$USER/Desktop/categories.txt", "/home/$USER/Desktop/", 1,
"/home/$USER/Desktop/DFDs_overlap.txt")
```

5. Module 5: GO Enrichments in DFDs and Jaccard Similarity Index

5.1. go_terms_DFD

Description
This function uses gprofiler2 to discover the GO enrichments of the genes in the DFDs. For each experiment and each DFD the script locates all the genes in each DFD, inputs them into gprofiler, and then outputs a total concatenated file for each experiment containing all the GO enrichments for this experiment

Usage
go_terms_DFD(count_matrix, DFD_path, GO_path, Organism)

Arguments

count_matrix	A path to an input file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start, #end, #chromosome
DFD_path	A path where all the DFD files are stored
GO_terms_path	A path where all the GO Term Enrichment files will be stored

Values
The output is a txt file for each experiment/sample containing all the GO terms enrichments of the genes contained in the DFDs of this sample

Examples
##Use file “count_table.txt” as input in the format described in the arguments above
##Use DFD files from DFDs folder in User Desktop
##Store the GO term enrichment files on the User’s Desktop
##Use “hsapiens” organism
go_terms_DFD("/home/$USER/Desktop/count_table.txt", "/home/$USER/Desktop/DFDs/", "/home/$USER/Desktop/", "hsapiens")

5.2. get_jaccard_matrix

Description
This function receives the count matrix and the path where the GO Enrichment files are saved as input and produces an all Vs all Jaccard Index Similarity matrix as output

Usage
get_jaccard_matrix(count_matrix, GO_terms_path)

Arguments

count_matrix	A path to an input file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start, #end, #chromosome
GO_terms_path	A path where all the GO Term Enrichment files are stored

Values
The output is a matrix containing all versus all Jaccard similarity indexes for all the experiments. The main diagonal (each experiments Jaccard index with itself) is set to 0.

See Also
get_GO_terms, jaccard_bootstrapping

Examples
##Use file “count_table.txt” as input in the format described in the arguments above
##Use the GO term enrichment files stored on the GO_terms folder on the User’s Desktop
jaccard_similarity_matrix <- get_jaccard_matrix("/home/$USER/Desktop/count_table.txt", "/home/$USER/Desktop/GO_terms/")

5.3. jaccard_bootstrapping

Description
This function will receive a index of categories of the different samples and a Jaccard Similarity Matrix and will perform bootstrapping with replacement to ascertain whether there are statistically significant differences among the categories of the experiments based on their Jaccard Similarity Indexes

Usage
jaccard_bootstrapping(jaccard_similarity_matrix, categories_index, num_permutations = 10000)

Arguments

jaccard_similarity_matrix	A matrix containing all vs all Jaccard similarity
categories_index	A vector file of a single column where each row contains the category value of the category corresponding to the experiment
num_permutations	The number of bootstrapping permutations. Default is 10000 permutations

Values
The output is a vector of expected median similarity values, observed median similarity values and p-values, each corresponding to the relevant categories provided by the User in the "categories_index" file. Bootstrapping tests for the Jaccard Similarity Index within the samples of a category being larger (so the samples are more similar among them) than randomly expected. In case of a significant result towards the other end of the distribution (samples being less similar than randomly expected), this will be denoted with a negative p-value

See Also
get_jaccard_matrix

Examples

```
##Use jaccard_similarity_mattrix double matrix
##Use file "categories.txt" as input to compare between experiments in the format described in the
##arguments above
##Perform 1000 permutations
bootstrap_result = jaccard_bootstrapping(jaccard_similarity_matrix,
"/home/$USER/Desktop/categories.txt", 1000)
```

6. Module 6: Bipartite Functional-Positional Networks Creation

6.1. create_bipartite

Description
This module will receive a file with normalized counts and the DFD files of the experiments and will create, for each experiment, a bipartite functional-positional network. Vertexes will represent either DFDs of GO term enrichments (gprofiler2) and edges will appear if the genes contained in a DFD are enriched in such a term. The bipartite networks created will be stored as igraph objects.

Usage
create_bipartite(count_matrix, DFD_path, Organism = “hsapiens”, bipartite_path)

Arguments

count_matrix	A path to an input file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start, #end, #chromosome
DFD_path	A path where all the DFD files are stored
Organism	The organism in question for gprofiler2 input. The values of this variable are the ones determined by gprofiler2 (see gprofiler2 vignette for more info). Default is hsapiens
bipartite_path	A path where all the output bipartite networks will be stored as igraph objects in txt files

Values
One bipartite network for each experiment, as an igraph object, stored in a txt file. As a bipartite network there will be two types of vertexes: DFD vertexes and GO term enrichment vertexes. If the genes contained in a DFD are enriched in a specific term, there will be an edge connecting this DFD to this specific term

See also
create_DFDs

Examples
##Use file “count_table.txt” as input in the format described in the arguments above
##We obtain the DFD files from the “DFDs” folder in the user’s Desktop
##Organism here is hsapiens
##Store the bipartite network files in Desktop
create_bipartite("/home/$USER/Desktop/count_table.txt", "/home/$USER/Desktop/DFDs/", "hsapiens", "/home/$USER/Desktop/")

7. Module 7: Bipartite functional-positional network metrics analysis and XGBoost classification

7.1. bipartite_cluster

Description
This function will receive as input the bipartite networks files and will produce a series of metrics (modularity, mean degree, connected components). Kruskal-Wallis ANOVA will be performed to assess difference between provided categories. The distribution of those metrics will be plotted and attempts at clustering will be made. Also this functions offers the selection of XGBoost classification based on bipartite features.

Usage
bipartite_cluster(count_matrix, bipartite_path, categories_index, plot_path, xgboost = 0)

Arguments

count_matrix	A path to an input file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start, #end, #chromosome
bipartite_path	A path where the bipartite networks (igraph objects) are stored as output by create_bipartite
categories_index	A vector file of a single column where each row contains the category value of the category corresponding to the experiment
plot_path	Path where the output plots will be stored
xgboost	XGBoost selector. 0 will not perform XGBoost. 1 will perform XGBoost based on recommended feature (connected components mean size). 2 will use all available features (modularity, mean degree, connected component mean size) to classify into categories. Default = 0. NOTE: We use 10-fold cross-validation, so for datasets that cannot meaningfully be split into 10 this feature cannot be used

Values

metrics	A dataframe containing the columns “modularity”, “mean degree”, “number_of_components” (the total number of connected components for each experiment), “component_mean_size” (the mean size in vertexes of the connected components for each experiment), “total_component_size” (the total size in vertexes of all the connected components for each experiment), “category” (by categories_index user input), “comp_mean_size_cluster” (cluster classification with hclust and Wards D based on connected components mean size), “all features cluster” (cluster classification

	with hclust and Wards D based on modularity, mean degree and connected components mean size)
model	The XGBoost model as output by the “xgboost” library
boost metrics	Evaluation of XGBoost performance with 10-fold cross validation output is RMSE, R^2, MAE and AUC. Will only be output if xgboost == 1, 2
modularity_pval	Kruskal-Wallis ANOVA p-value for modularity among different categories
degree_pval	Kruskal-Wallis ANOVA p-value for mean degree among different categories
mean_components_size_pval	Kruskal-Wallis ANOVA p-value for connected components mean size among different categories
total_components_size_pval	Kruskal-Wallis ANOVA p-value for connected components total size among different categories

The function will also produce plots for:

1) Mean Degree distributions by category boxplot
2) Mean Modularity distributions by category boxplot
3) Mean Connected Component Size distributions by category boxplot
4) Mean Connected Component Size Clusters by category barplot
5) All Features (modularity, mean degree, connected components mean size) Clusters by category barplot

See also

create_bipartite

Examples

```
##Example 1, NO XGBoost
##Use file “count_table.txt” as input in the format described in the arguments above
##Use as input bipartite networks stored in the bipartite folder in Desktop
##Use file “categories.txt” as input to compare between experiments in the format described in the
##arguments above
##Store output plots in Desktop
bmetrics <- bipartite_cluster("/home/$USER/Desktop/count_table.txt",
"/home/$USER/Desktop/bipartite/", "/home/loukos/Desktop/categories.txt",
"/home/$USER/Desktop")

##Example 2, XGBoost = 1, Use connected components mean size for classification
##Use file “count_table.txt” as input in the format described in the arguments above
##Use as input bipartite networks stored in the bipartite folder in Desktop
##Use file “categories.txt” as input to compare between experiments in the format described in the
##arguments above
##Store output plots in Desktop
##Force 1 in xgboost argument
bmetrics <- bipartite_cluster("/home/$USER/Desktop/count_table.txt",
"/home/$USER/Desktop/bipartite/", "/home/loukos/Desktop/categories.txt",
"/home/$USER/Desktop", 1)


##Example 3, XGBoost = 2, Use all features for classification
##Use file “count_table.txt” as input in the format described in the arguments above
##Use as input bipartite networks stored in the bipartite folder in Desktop
```

```
##Use file “categories.txt” as input to compare between experiments in the format described in the
##arguments above
##Store output plots in Desktop
##Force 2 in xgboost argument
bmetrics <- bipartite_cluster("/home/$USER/Desktop/count_table.txt",
"/home/$USER/Desktop/bipartite/", "/home/loukos/Desktop/categories.txt",
"/home/$USER/Desktop", 1)
```

8. Module 8: Bipartite Functional-Positional Network Visualization

8.1. plot_bipartite

Description
This function will trim the bipartite network degree to a user-defined threshold to make the network more legible and create a plot. The plot will highlight community structures to make it easier for the user to detect bundles of DFD vertexes related to specific GO terms.

Usage
plot_bipartite(count_matrix, bipartite_path, degree_trim = 5, plot_path, plot_width = 8000, plot_height = 8000)

Arguments

count_matrix	A path to an input file containing a number of RNA-seq experiments in the following format: #geneid, #experiment 1, #experiment 2, ... , #experiment n, #genename, #start, #end, #chromosome
bipartite_path	A path where the bipartite networks (igraph objects) are stored as output by create_bipartite
degree_trim	Threshold for the minimum degree allowed on the plot to make the graph plot more legible. Default is 5
plot_path	Path where the output plots will be stored
plot_width	Width in pixels of the plot width. Default is 8000
plot_height	Width in pixels of the plot height. Default is 8000

Values
The output is one plot for each bipartite network in the input folder

See also
create_bipartite

Examples
##Use file “count_table.txt” as input in the format described in the arguments above
##Use as input bipartite networks stored in the bipartite folder in Desktop
##Use trimming of vertexes with degree < 8
##Store output plots in Desktop
##Output plots 6000x6000 pixels
plot_bipartite("/home/$USER/Desktop/count_table.txt", "/home/$USER/Desktop/bipartite/", 8, "/home/loukos/Desktop/",6000, 6000)

9. Miscellaneous Tools

9.1. bedoverlapenr

Description
This bash tool will use bedtools (reference) to calculate the overlap between 2 genomic areas of interest (observed overlap). Then it will randomly pick other areas of the same size from the same chromosomes to calculate the randomly expected overlap (bootstrapping). There will be comparisons of observed versus expected overlaps to calculate bootstrap p-values assessing the significance of the observed enrichment of overlap.

Usage
bash bedoverlapenr.sh $FILE1.bed $FILE2.bed chromosome_lengths number_of_permutations

Arguments

$FILE1.bed, $FILE2.bed	Files of bed format for the genomic areas of interest that will be compared
chromosome_lengths	2 column file, containing column1: #chromosome_identifier, column2: chromosome length in bp. Usually first file is the “peaks” file, second file is the “context” file (promoters, genes, etc)
number_of_permutations	User defines the desired random permutations that the tool will execute

Values

Ratio1	Coverage of context file
Ratio	Coverage of intersection
Enrichment	Enrichment of Peak/Context Overlap vs Expected Ratio
p-value	Statistical significance of Enrichment

See Also
overlap_convert

Examples
##Use the DFDs as “peaks” file
##Use TADs as context
##hg38.genome file should contain chromosome lengths in bp
##Run 1000 permutations
##Output result in results.txt
bash bedoverlapenr.sh DFDs.bed hg38_TAD_regions.bed hg38.genome 1000 >> results.txt

9.2. overlap_convert

Description
A simple R script that will remove redundant lines from the bedoverlapenr.sh tool to allow easy loading of the tool output in R as a dataframe. It will also create a matrix of all Vs all overlap values

Usage
overlap_convert(dfds_overlap_input, dfds_overlap_output, create_overlap_matrix = 0)

Arguments

dfds_overlap_input	Name and full path of the file that was produced by bedoverlapenr.sh
dfds_overlap_output	Name and full path of the file that will be created containing the formatted results
create_overlap_matrix	If value == 1, for cases of All vs All enrichment of overlap user can select this option to get a square matrix of All vs All enrichment of overlap values. NOTE: This will create a square matrix so only works for All vs All enrichments. Default value == 0

Values
The produced file will contain the following format:
File1= File1.bed File2= File1.bed Ratio1= xxx Ratio= yyy Enrichment= zzz p-value= www
The produced matrix will be a square matrix containing All vs All enrichments of overlap

See Also
bedoverlapenr.sh

Examples
##Execute bedoverlapenr.sh and output in results.txt
##Use this file as conversion input and output in results_convert.txt
##Store square All Vs All overlap in variable overlap_matrix
bash bedoverlapenr.sh DFDs.bed hg38_TAD_regions.bed hg38.genome 1000 >> results.txt
overlap_matrix ← overlap_convert(/home/$USER/Desktop/results.txt,
/home/$USER/Desktop/results_convert.txt, 1)