**CAFE – Coefficient-based Analysis of Fitness by read Enrichments**

**User functions**

*loadData(inputFileName)*

Loads data from a file called <<inputFileName>>. If no filename or path to a file is given, CAFE tries to load a file called “all\_rep\_inseq.txt” in the current working directory. The function creates the following objects that can be used by the user (or other functions): *countData, undefCounts, unmappedCounts, countMatrix, geneNames, sampleNames, treatmentClasses, treatments*

*fitCof(testCondition, control, dataTable, undefCountTable, correction)*

Generates fitness coefficients for two different conditions <<testCondition>> and <<control>>. See the possible conditions by typing "treatments" (showing the content of the *treatments* object). If <<dataTable>> and <<undefCountTable>> are not supplied, *countData* and *undefCounts* are, respectively, used by the function by default. The <<correction>> option controls what type of bias correction that should be used for skewed data distributions in samples (e.g. if one gene – or a few – dominates the total read count of a sample). The <<correction>> option can be one of “undefined” (default, will use the reads not mapping to a specific gene to control for bias), “median” (normalizing each sample by the median read count), or “none” (not accounting for skewed data at all). The function returns a list of fitness coefficients for each gene and each replicate (i.e. a comparison between the test condition and the control for each gene and replicate).

*fitCofComp(control, treatmentList, dataTable, undefCountTable, correction)*

Generates fitness coefficients for all conditions in <<treatmentList>> compared to a control condition (<<control>>). All other options are similar to the *fitCof* function. Returns a table of fitness coefficients similar to *fitCof*, but with multiple samples in the same table.

*rankByP(fitCofTable, test , correction, pcut, toplist)*

Generates a ranked list of genes using a table of fitness coefficients <<fitCofTable>> generated using the *fitCof* command. By default, the table is sorted by p-value, but if <<toplist>> is set to FALSE, the order from <<fitCofTable>> will be preserved in the generated list. By default, significance is assessed using Student’s t-test, but this can be changed using the <<test>> options to either the log-transformed Student’s t-test (“t.test”, default), the Wilcoxon signed rank test (“wilcox”) or a non-transformed Student’s t-test (“nonlog.t.test”, not recommended unless you have a very good reason to use it). By default, p-values are corrected using Benjamini-Hochberg false discovery rate. This can be changed using the <<correction>> option, to any of the adjustment methods used for the *p.adjust* command. By default, all genes are included in the output, but by changing <<pcut>> to a value smaller than 1, the list will be restricted to only the genes with an adjusted p-value smaller than <<pcut>>.

*upDown(testCondition, control, p, dataTable, undefCountTable, correction, drawPlot)*

Shows the number of genes with a significantly positive and negative fitness coefficient between the <<testCondition>> and the <<control>>. Most of the options are the same as for the *fitCof* function. By default, the significance threshold <<p>> is set to 0.05 (uncorrected p-value is used for this function). The *upDown* function can also draw a plot showing the fitness coefficients of each gene in the dataset, ranked by p-value, by setting <<drawPlot>> to TRUE. The output of the function is three numbers: the number of genes with a positive fitness coefficient, the number of genes with a negative fitness coefficient, and the expected number of genes with a significant positive and negative fitness coefficient based on the p-value cutoff and the number of genes.

*generateGSC(file, columns, header)*

Loads a gene set collection from tab-separated <<file>>. The function will use the columns specified by <<columns>>, by default column 1 and 3. The first column will be assumed to be gene names and the other column will be assumed to contain gene sets, e.g. pathways. By default, no header row is assumed in the input file, but if <<header>> is set to TRUE, the first row in the input file will be regarded to contain no information. The function returns a gene set collection (GSC) to be used with the *genesetAnalysis* function.

*genesetAnalysis(GSC, dataset, p, test, center)*

Performs a gene set analysis based on a <<GSC>> (see *generateGSC* function) and a <<dataset>> produced by the *rankByP* function. By default, only genes with a significantly different fitness coefficient (*p* < 0.05) are included in the gene set analysis. This can be changed by changing the significance cutoff <<p>> to any value between 0 and 1, setting the new significance cutoff. On each gene set, a statistical test of differential sum of fitness coefficients is performed. Be default, this is a t-test, but this can be changed using the <<test>> option to either “t.test” or “wilcox” (for the Wilcoxon signed rank test). If <<center>> is set to TRUE, the mean fitness coefficient of each sample is subtracted from that sample before gene set analysis is performed. By default, this centration of data is not performed. The function outputs a table of gene sets (e.g. pathways) and the number of genes in each set with a positive fitness coefficient, a negative fitness coefficient, the number of non-significant genes, the estimated average fitness coefficient in the set, the uncorrected p-value for the enrichment, and the FDR corrected p-value for enrichment.

**Internal functions**

*sizeNormalize(countDataTable, undefCountTable, returnUndef)*

Normalizes each count by the total number of mapped reads. If <<countDataTable>> and <<undefCountTable>> is not supplied, *countData* and *undefCounts* are, respectively, used by the function by default. If <<returnUndef>> is TRUE, the function will return normalized values for the *undefCountTable* rather than the *countDataTable* (default).

*generatePseudoCount(countDataTable, undefCountTable, controlDataTable, controlUndefTable, includeAdapt)*

Generates a suitable pseudo-count number for the input samples. (Will be further documented later).

**Objects**

*countData*

Contains the raw counts for each sample and gene (including adapter contaminations etc.)

*undefCounts*

Contains the raw data counts for the number of reads mapping to the genome but not to a defined gene for each sample (including adapter contaminations etc.)

*unmappedCounts*

Contains the raw data counts for the number of reads not mapping to the genome for each sample (including adapter contaminations etc.)

*countMatrix*

Contains the same information as *countData*, but in matrix format.

*geneNames*

List of all gene names.

*sampleNames*

List of all sample names, including replicates and adapter contaminations etc.

*treatmentClasses*

Same information as in *sampleNames*, but organized as factors.

*treatments*

Contains all treatments used for the study (excluding e.g. adapter contaminations).

**Example workflow**

>source("cafe.R")

>loadData()

>treatments

>fcCol = fitCof("Pa\_bf\_control","Pa\_ON")

>rCol = rankByP(fcCol)

>head(rCol, 10)