Master Regulator Analysis for Cell Type/State/Phenotype Conversion (Bulk RNA-Seq Data)

This package is designed to predict master regulator genes that can drive the conversion of one cell type/phenotype/state to another, using bulk RNA sequencing data.

Steps:

1. Generate or Obtain a Gene Regulatory Network (GRN) for the Target Cell Type:
   * If a GRN for the target cell type is not available: You can generate one using the APPLE package included in this repository (see the README.md in the APPLE folder).
   * Example Network: The Example folder provides a pre-generated, fully functional GRN for human immune cells (human\_immune\_network.csv). This network can be used to predict regulators for converting any cell type into human immune cells.
   * Using Other Networks: You can use any publicly available GRN, as long as it conforms to the format described below (see "Input Files").
2. Perform Differential Expression Analysis:
   * Conduct a differential expression analysis between the source cell type/phenotype/state and the target cell type/phenotype/state.
   * You can use tools like edgeR or DESeq2 for this analysis.
   * The output should include gene names, log fold change (logFC), p-values, and adjusted p-values (FDR).
3. Rank Genes using the Master\_Regulator\_Scoring.R Script:
   * Run the Master\_Regulator\_Scoring.R script to calculate master regulator scores and generate a ranked list of genes. Detailed instructions are provided below.

Example Workflow:

The Example folder contains data for predicting master regulators to convert human glioblastoma cancer cells into dendritic cells. It includes:

* diff\_exp\_analysis.txt: Results from edgeR analysis comparing glioblastoma cells and dendritic cells.
* human\_immune\_network.csv: GRN for human immune cells, generated by APPLE.
* Master\_Regulator\_Scoring.R: The R script for calculating master regulator scores.
* run.sh: A shell script to run the analysis.
* nScore\_example: A folder containing the example analysis results.

To run the example:

1. Download the Example folder.
2. Make run.sh executable: chmod +x run.sh
3. Run the script: ./run.sh

This will generate results similar to those in the nScore\_example folder.

Master Regulator Scoring Script: Detailed Overview

Script Functionality:

The Master\_Regulator\_Scoring.R script identifies potential master regulator genes by integrating a gene regulatory network (GRN) with differential expression data. It performs the following:

1. Input:
   * A GRN file (tab-delimited: gene1, gene2, MI, where MI is Mutual Information or an edge weight).
   * A gene expression statistics file (tab-delimited: gene, logFC, pvalue, fdr).
2. Network Processing:
   * Reads the GRN and creates an igraph network object.
   * (Optional) Filters the network to a subnetwork of the top N genes based on a user-specified statistic (e.g., top 20% by logFC).
   * Calculates network centrality measures (degree, betweenness, coreness, PageRank, eigenvector centrality) if requested by the user.
3. Master Regulator Score Calculation:
   * Uses a neighborhood-based algorithm.
   * For each gene, it examines its network neighborhood up to a specified number of steps (hops).
   * Aggregates the gene expression statistics (e.g., logFC, p-value) of the neighbors, weighted by connection strength (MI or correlation derived from MI) and distance from the source gene.
   * Aggregation methods: sum, average, or median.
   * (Optional) Incorporates the source gene's statistics into the score (sum or product with neighborhood score).
   * (Optional) Performs iterative calculations, using scores from one round as input for the next.
4. Output:
   * A table of genes with master regulator scores.
   * A ranked list of genes by score.
   * (Optional) Validation against a list of known master regulators.

User Guide

1. Prerequisites

* R: Installed on your system.
* Packages:
  + igraph
  + optparse
  + data.table
  + Rmpi (optional, for parallel processing)

Install packages in R:

install.packages(c("igraph", "optparse", "data.table"))

# For parallel computing (optional)

install.packages("Rmpi")

2. Input Files

* Network File (-n or --network):
  + Defines the GRN of the target cell type.
  + Highly recommended: Use a network generated by the APPLE package for best accuracy.
  + Format: Tab-delimited text file.
  + Columns:
    1. gene1 (first gene)
    2. gene2 (second gene)
    3. MI (Mutual Information between gene1 and gene2) or other edge weight. Use 1 for all edges if no weight is available.
  + Example:

geneA geneB 0.5

geneB geneC 0.8

geneA geneC 0.2

* Gene Expression Statistics File (-x or --gene\_ex\_stat):
  + Provides differential expression statistics between the source and target cell types.
  + Generated from tools like edgeR or DESeq2.
  + Format: Tab-delimited text file or RDS file (R data file). If RDS, it should be a data frame or a list with an element named lr\_table containing the data frame.
  + Columns (order matters):
    1. gene (gene name)
    2. logFC (log2 fold change)
    3. pvalue (p-value)
    4. fdr (false discovery rate)
  + Example:

gene logFC pvalue fdr

geneA 2.5 0.01 0.05

geneB -1.2 0.001 0.02

geneC 0.8 0.05 0.1

3. Running the Script

Get Help:

Rscript Master\_Regulator\_Scoring.R --help

or

Rscript Master\_Regulator\_Scoring.R -h

Basic Syntax:

Rscript Master\_Regulator\_Scoring.R --cmd <command> [options]

Commands (--cmd):

* run: Calculate master regulator scores (main command).
* btw: Calculate node-weighted betweenness centrality (may be a separate analysis).
* btw\_jobs: Generate job scripts for calculating betweenness on a cluster (for large networks).
* parameter\_optimization: (Likely for optimizing script parameters – details not fully described in the script).
* help: Display the help message.
* All options should be entered without string (single/double) quotes. Do not enter “unw” or "logFC,pvalue,degree" in the command line

Example: run command:

Rscript Master\_Regulator\_Scoring.R --cmd run --network my\_network.txt --gene\_ex\_stat my\_gene\_stats.txt --outdir my\_results

4. Key Options (for run command):

* -n or --network: Path to the network file.
* -x or --gene\_ex\_stat: Path to the gene expression statistics file.
* -u or --consider\_positive\_values\_only: If TRUE, only positive values of input statistics (e.g., logFC) are considered.
* -f or --fdr\_to\_confidence: If TRUE, converts FDR to confidence scores (1 - FDR).
* -s or --step: Number of steps (hops) in the network neighborhood (default: 2).
* -g or --top\_gene\_statistics: Statistic for selecting top genes for subnetwork extraction (e.g., "logFC", "pvalue", "fdr") (default: "pvalue").
* -l or --gene\_statistics\_list: Comma-separated list of statistics for score calculation (e.g., "logFC,pvalue,degree", "fdr,betweenness") (default: "logFC,fdr,betweenness"). Please remember to enter the value without quote marks, otherwise the error will be thrown. Right syntax: --gene\_statistics\_list logFC,pvalue,degree, not --gene\_statistics\_list “logFC,pvalue,degree”.
* -r or --nround: Number of iterative rounds (default: 1).
* -e or --edge\_weight: Edge weight type: "rho" (correlation-like, from MI), "MI" (Mutual Information), "unw" (unweighted) (default: "unw").
* -p or --weight\_power: Power for edge weights (default: 1).
* -o or --step\_power: Power for combining scores from different steps (default: 0).
* -z or --steps\_combined: If TRUE, combine scores from different steps (default: FALSE).
* -y or --step\_normalization: If TRUE, normalize scores at each step before combining (default: FALSE).
* -w or --input\_normalization: If TRUE, normalize input statistics before calculations (default: FALSE).
* -t or --top\_genes: Proportion of top genes for subnetwork (e.g., 0.2 for 20%) (default: 0.2).
* -k or --use\_rank: If TRUE, use the rank of the statistic instead of its value (default: FALSE).
* -m or --master\_genes: Comma-separated list of known master regulators for validation (e.g., "geneA,geneB,geneC") (default: "POU5F1,SOX2,MYCN,NANOG,LIN28A").
* -d or --source\_node\_inclusion: How to include source gene statistics: "n" (no), "s" (sum with neighborhood score), "p" (multiply with neighborhood score), "m" (use only source node statistics) (default: "p").
* -a or --neighbor\_aggregation\_method: How to aggregate neighbor scores: "s" (sum), "a" (average), "m" (median) (default: "a").
* -v or --verbose: If TRUE, print verbose messages (default: TRUE).
* -i or --index: Index for the experiment (for output file naming) (default: 1).
* --outdir: Output directory (default: "nSCORE").

5. Output Files (for run command):

The script creates an output directory (specified by --outdir) with these files:

* scores.csv: Genes with their master regulator scores.
* ranks.csv: Genes ranked by their scores.
* master\_genes\_ranks.csv: Ranks of user-provided master genes (if any).
* experiment\_condition.csv: Options used in the experiment.
* experiment<index>.log: Log file with messages and results.
* <gep\_filename>\_master\_score\_result.RDS: RDS file with results (scores, ranks, master gene ranks, input differential expression data).

6. Betweenness Centrality Calculation (btw and btw\_jobs commands):

* btw command:

Rscript Master\_Regulator\_Scoring.R --cmd btw --network my\_network.txt --gene\_ex\_stat my\_gene\_stats.txt --gene\_index <start\_index> --batch\_size <size> --outdir DFs

* + --gene\_index: Starting index of genes to process.
  + --batch\_size: Number of genes to process in one batch.
  + --outdir: Output directory for intermediate files (default: "DFs").

Calculates node-weighted betweenness for a gene subset; saves results in "DFs".

* btw\_jobs command:

Rscript Master\_Regulator\_Scoring.R --cmd btw\_jobs --network my\_network.txt --gene\_ex\_stat my\_gene\_stats.txt --batch\_size <size> --outdir DFs

Generates job scripts for betweenness calculation on a cluster. Divides genes into batches, creates a job script per batch. Submit these to your cluster's job scheduler. Final betweenness calculation is done by nw\_betweenness\_nompi\_finalize (called within nw\_betweenness\_nompi\_manager, called by btw\_jobs) after cluster jobs complete.

7. Important Notes:

* Computational Cost: Can be high for large networks/multiple steps. Using a subnetwork of top genes reduces time.
* Parameter Tuning: Parameter choice (e.g., step, edge\_weight, weight\_power, source\_node\_inclusion, neighbor\_aggregation\_method) impacts results. Experiment to find the best settings for your data.
* Parallel Processing: Supports Rmpi for parallel processing. Requires Rmpi installation and configuration. Auto-detects cluster usage, adjusts slaves.
* Memory Management: Attempts to manage memory; very large networks may still cause issues.
* Convergence: With iterative calculations (nround > 1), checks for convergence by comparing gene ranks between rounds (based on average rank change).

8. Example Workflow:

1. Prepare Input Files: Create network and gene expression statistics files in the specified formats.
2. Run run Command: Execute with appropriate options.
3. Examine Output: Analyze scores.csv, ranks.csv, master\_genes\_ranks.csv for potential master regulators.
4. Parameter Optimization (Optional): Experiment with different parameters using a subset of data for faster exploration.
5. Betweenness Calculation (Optional): Use btw or btw\_jobs for node-weighted betweenness.

Troubleshooting:

* Errors: Check experiment<index>.log for error messages. Verify input file formats, package installation.
* Memory Issues: Use a smaller subnetwork (top\_genes) or fewer steps (step). Run on a machine with more RAM.
* Slow Execution: Use parallel processing (Rmpi), reduce steps, or use a subnetwork.

This guide should help you effectively use Master\_Regulator\_Scoring.R. Adapt parameters and workflow to your research question and data.