Project Overview

Due to the multifactorial nature of drug responses, depicting the systemic control mechanisms governing these responses is challenging yet essential for pharmacogenomic research. This project introduces a novel statistical mechanics framework called idopNetworks (information-driven omnidirectional personalized networks). This framework enables the reconstruction of dynamic, omnidirectional, and personalized networks that encompass all pharmacogenomic factors and their interconnections, dependencies, and mechanistic interactions.idopNetworks offers a detailed representation of how genes and proteins influence drug responses through cell-to-cell cascades, identifying critical pathways for information flow and propagation. By mapping these pathways, idopNetworks provides insights into how drug efficacy and toxicity are modulated at a genomic level. This framework has the potential to enhance our understanding of the genomic mechanisms behind drug responses and guide the design of drugs that maximize efficacy at lower doses. We believe that idopNetworks can serve as a valuable tool in pharmacogenomics, providing a scientific foundation for the development of personalized drugs that achieve optimal therapeutic effects with minimal doses.

File Structure

The project's file structure and descriptions are as follows:

code/: Contains R scripts for data processing and network analysis.

- Fitting.R: This script is used for model fitting, primarily adjusting and fitting the FPKM data to generate inputs suitable for network analysis, providing a solid foundation for constructing idopNetworks.
- Network.R: This script constructs the idopNetworks. By integrating various pharmacogenomic factors, it generates a dynamic and personalized network, supporting the analysis of drug response mechanisms.
- cluster.R: This script performs clustering analysis on the data. Genes with similar expression patterns may have similar functions, and in the network, they can be grouped into the same functional module. This study uses a mixture model-based functional clustering method to cluster all genes in single-omics data, helping identify potential functional gene modules and supporting functional annotation and pathway identification

in network analysis.

data/: Contains raw data files for network analysis in CSV format with FPKM values.

- NonWGpost.csv: Post-treatment data file for the NonWG group, containing FPKM values after treatment.
- NonWGpre.csv: Pre-treatment data file for the NonWG group, containing FPKM values before treatment.
- WGpost.csv: Post-treatment data file for the WG group, containing FPKM values after treatment.
- WGpre.csv: Pre-treatment data file for the WG group, containing FPKM values before treatment.

Installation Instructions

The project is developed based on R language. Please ensure that the following required R packages are installed before running the analysis code.

Install R and RStudio

Firstly, you need to install R and the recommended IDE RStudio. Please download and install the appropriate version of R and RStudio according to your operating system.

Install Necessary R Packages

This project depends on several R packages for data processing, network construction, and visualization. Run the following code in the R console to install the required packages:

install.packages(c("mvtnorm", "pbapply", "parallel", "orthopolynom", "glmnet", "ggplot2", "reshape2"))

- mvtnorm: Used for computations involving multivariate normal distributions.
- pbapply: Provides apply functions with progress bars, enhancing feedback during computations.
- parallel: Provides parallel computing capabilities to accelerate complex analyses (usually installed by default in R).
- orthopolynom: Used for generating orthogonal polynomials, supporting advanced statistical analysis.
- glmnet: Implements regularization methods like Lasso and Ridge regression for high-

dimensional data analysis.

- ggplot2: Used for visualizing analysis results.
- reshape2: Used for data reshaping and manipulation.

Set Working Directory

To ensure that the code can correctly read data files and save output files, set the working directory to the project's root directory. You can set the working directory in R using the following command:

```
setwd("path/to/your/project")
```

Running the Code

After installing all dependencies and setting the working directory, you can execute the analysis by loading and running the individual script files. Refer to the "Usage Instructions" section below for specific running methods.

Usage Instructions

This project includes three main R scripts, each responsible for specific analytical tasks, including data fitting, clustering analysis, and network construction.

1. Preparation

Ensure that all necessary input data files and script files are placed in the main project folder, with the following file structure:

- Fitting.R: Script for data fitting.
- Network.R: Script for network construction.
- cluster.R: Script for clustering analysis.
- NonWGpost.csv: Contains post-treatment data for the NonWG group in CSV format with FPKM values.
- NonWGpre.csv: Contains pre-treatment data for the NonWG group in CSV format with FPKM values.
- WGpost.csv: Contains post-treatment data for the WG group in CSV format with FPKM values.
- WGpre.csv: Contains pre-treatment data for the WG group in CSV format with FPKM values.

Before starting the analysis, set the working directory to the project folder in your R console to ensure all scripts can correctly read the data files. For example:

setwd("path/to/your/project")

After setting the working directory, you can proceed to run each script in sequence to complete the data analysis workflow.

2. Data Fitting: Fitting.R

First, run the Fitting.R script to complete data preprocessing and fitting. This script will transform and fit the original data to generate a parameter matrix suitable for subsequent analyses.

Run the following command in the R console:

source("Fitting.R")

Main Functions:

Environment Cleanup and Package Loading: Clears the R global environment and loads the required R packages, including glmnet, deSolve, orthopolynom, etc.

Data Loading and Processing: The script reads the NonWGpost.csv file and performs initial processing, including column sorting and log transformation.

Define Power Equation Fitting Functions:

- power_equation(): Defines the power equation used to generate predicted values.
- power_par(): Performs power equation fitting for each gene based on the provided time series data and expression values, optimizing parameters by minimizing the sum of squared residuals.
- power_fit(): Applies the fitted parameters to the time series data to calculate the model predictions.

Parameter Fitting and Output:

Uses power_par() to calculate the parameter matrix par.mu1, where each gene's expression pattern has an estimated parameter.

Generates a fitted results data frame line_data1, which includes predicted values for each gene across different time points for use in clustering analysis and visualization.

Output: After fitting, the script produces a data frame with fitted parameters and predicted values, providing foundational data for subsequent clustering and network analysis.

3. Clustering Analysis: cluster.R

After completing data fitting, use the cluster.R script to perform clustering analysis. This script clusters genes based on time series data, grouping genes with similar expression patterns into the same module.

Run the following command in the R console:

source("cluster.R")

Main Functions:

Environment Cleanup and Package Loading: Clears the R global environment and loads required packages, including mvtnorm, orthopolynom, ggplot2, etc., to ensure the environment is configured correctly.

Data Loading: Loads the WGpre.Rdata file containing time series data for clustering analysis.

Define Clustering and Fitting Functions:

- power_equation(): Defines the power equation used to compute predicted values.
- get_init_par(): Performs initial clustering using the K-means algorithm and generates initial parameters for each cluster. This function returns initial clustering parameters, including initial curve parameters and covariance matrix parameters for each cluster center.
- get_cluster(): Implements a mixture model-based clustering method that iteratively optimizes gene clustering. Each iteration includes an Expectation (E) step and a Maximization (M) step, calculating posterior probabilities for clusters and optimizing parameters until convergence.

The final output includes cluster assignments, log-likelihood, BIC values, and clustering plots.

Generate and Output Clustering Results:

Generates a visualization of clustering assignments and clustering curves, allowing users to observe temporal trends in gene expression across different modules.

The result object includes clustering parameters, BIC values, log-likelihood, and visualization plots for further analysis.

Output: After clustering, the script produces data frames and plots with cluster assignments and inter-module relationships, providing a clear visualization of dynamic patterns within different gene modules.

4. Network Construction: Network.R

Finally, run the Network.R script to build a network based on the clustering results. This script identifies and quantifies the dependencies between gene modules and constructs a dynamic, omnidirectional network structure.

Run the following command in the R console:

source("Network.R")

Main Functions:

Environment Cleanup and Package Loading: Clears the R global environment and loads required packages, including mvtnorm, orthopolynom, glmnet, etc., to ensure the environment is configured correctly.

Define Network Construction Functions:

- get_legendre_par(): Calculates inter-module relationships using generalized orthogonal polynomials. This function selects significant interactions using Lasso regression and estimates interaction coefficients using orthogonal polynomials.
- ode_optimize() and get_effect(): Define optimization functions for differential equations to estimate the effects of time series data, calculating the main and secondary effects of gene expression on each module.
- Parallel Computing and Network Construction:
- Uses multi-core parallel processing to compute interactions for each module, generating generalized orthogonal polynomial parameters for each module's interactions through the pblapply() function.
- get_output() converts the constructed parameters and network relationships into visualizable data for subsequent plotting and analysis.

Generate Network Results:

Outputs a relationship network matrix aF, containing dependency relationships between each gene and other modules, interaction types (positive or negative effects), and effect strengths.

The aF data frame serves as the final network data, with four columns: "from" indicating the influencing gene, "to" indicating the influenced gene, "dep_effect" representing the dependency effect size, and "effect_type" representing positive or negative effect.

Output: After running Network.R, the script outputs a network relationship data frame aF for further network analysis and visualization. The output includes:

- from: Start node representing the source gene.
- to: Target node representing the influenced gene.
- dep_effect: Interaction strength.
- effect_type: Type of effect, either positive ("+") or negative ("-").

5. Summary of Results

After running all scripts, you will obtain the following results:

Fitted Parameter Matrix: Describes gene expression patterns.

Clustering Results: Includes cluster assignments and lists of genes in each module.

Network Data Frame aF: Contains dependency relationships and interaction types between each pair of gene modules.

These results enable further analysis of gene interactions within pharmacogenomics, offering insights into the genomic mechanisms of drug response.

Notes

Ensure Correct Data File Format: Please make sure that all data files are in the correct format. Incorrect file formats may cause the code to fail.

Check R Environment Configuration: Before running the R code, ensure that your R environment is properly configured. Compatibility issues may arise if using an unsupported R version or if required packages are not installed.

Contact Information

If you have any questions or need access to data files, please contact the author: chejincan@bjfu.edu.cn.