This document will summarize the code’s function for every file.   
  
1. figure1.R  
The provided R code appears to be a script for performing various data analysis tasks on RNA-Seq data from the TCGA database. The script includes multiple steps, such as data preprocessing, clustering, survival analysis, and visualization. Here's a summary of the function of each line in the script:

1. Comments explaining the purpose of the script.

2-10. Loading necessary libraries and defining functions for package installation.

12-18. Loading required packages and functions from local and online sources.

20-39. Setting parameters and loading the list of cancer types (Cancers) to be analyzed.

41-42. Loading the list of genes associated with invasiveness (INV\_genes).

45-50. Looping over each cancer type (Cancer) in the list.

53-57. Loading RNA-Seq data for the current cancer type and extracting the INV gene subset.

60-63. Calculating pairwise distances between INV gene expression profiles.

65-85. Performing consensus clustering using the ConsensusClusterPlus package to identify clusters in the data.

87-99. Processing and organizing the cluster assignments, calculating cluster means, and determining optimal clusters using the Calinsky-Harabasz index.

101. Removing temporary files.

103-105. Saving the final cluster assignment data and manual annotations.

108-142. Generating heatmaps of gene expression profiles for different invasiveness categories.

144-160. Performing survival analysis based on the invasiveness categories for each cancer type.

163-189. Creating a forest plot summarizing hazard ratios and p-values for each cancer type.

192-248. Creating boxplots of invasiveness scores for each cancer type.

Calinski-Harabasz (CH) Index (introduced by Calinski and Harabasz in 1974) can be used to evaluate the model when ground truth labels are not known where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset. The CH Index (also known as Variance ratio criterion) is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Here cohesion is estimated based on the distances from the data points in a cluster to its cluster centroid and separation is based on the distance of the cluster centroids from the global centroid.  
  
Performing Consensus Clustering using the ConsensusClusterPlus package:

Consensus clustering is a technique used to identify stable and robust clusters within a dataset by iteratively resampling the data and clustering it multiple times. The ConsensusClusterPlus package in R provides an implementation of this technique. Here's how the process works:

- Line 65: The pairwise distances between INV gene expression profiles are calculated using the `dist` function. This distance matrix (`ddist`) is a measure of how different each pair of samples is in terms of gene expression.

- Lines 67-84: The `ConsensusClusterPlus` function is used to perform consensus clustering. This function takes several parameters:

- `ddist`: The distance matrix calculated earlier.

- `maxK`: The maximum number of clusters to consider (in this case, up to 6 clusters).

- `pItem`: The proportion of resamples used to form a consensus. A higher value leads to more robust clusters.

- `reps`: The number of resamples.

- `clusterAlg`: The clustering algorithm to use (hierarchical clustering in this case).

- `innerLinkage` and `finalLinkage`: The linkage criteria for hierarchical clustering.

- `plot`: Output the consensus results as a PDF plot.

- `writeTable`: Write out the consensus class assignments.

- `verbose`: Print progress information.

- Lines 88-98: The script reads the consensus class assignments and processes them. It calculates the mean INVscore (a measure of invasiveness) for each cluster and organizes the data into a format suitable for further analysis.

2. Processing and Organizing the Cluster Assignments, Calculating Cluster Means, and Determining Optimal Clusters using the Calinsky-Harabasz Index:

- Lines 100-102: The hierarchical clustering (`hclust`) is performed on the INV gene expression data using the Ward.D2 method, and the distance matrix (`ddist`) calculated earlier.

- Line 103: The `calinsky` function is used to calculate the Calinsky-Harabasz index for different numbers of clusters (`gMax` up to 10). This index measures the separation between clusters, aiming to find the optimal number of clusters that provide the best separation.

- Lines 104-110: The script identifies the number of clusters with the highest Calinsky-Harabasz index within a specific range (usually from 2 to 6 clusters). This value represents the optimal number of clusters for the given dataset.

- Lines 112-116: The script saves the Calinsky-Harabasz curve as a PDF plot, showing the index values for different numbers of clusters.

In summary, the consensus clustering step uses the ConsensusClusterPlus package to find stable clusters within the data. The subsequent steps involve processing and organizing the cluster assignments, calculating cluster means, and determining the optimal number of clusters using the Calinsky-Harabasz index, which measures the quality of clustering based on the separation of clusters. These steps help in identifying meaningful clusters within the gene expression data related to invasiveness in different cancer types.  
  
Heatmaps of gene expression profiles for different invasiveness categories are important because they allow researchers to visually assess how gene expression patterns vary across different levels of invasiveness in various cancer types. This can provide insights into which genes are upregulated or downregulated as invasiveness increases, potentially revealing key molecular factors associated with cancer progression.

Survival analysis based on invasiveness categories for each cancer type is crucial for understanding the impact of invasiveness on patient outcomes. By analyzing survival rates and time-to-event data, researchers can determine whether higher invasiveness is associated with poorer prognosis and shorter survival times. This information can guide treatment strategies and prognosis prediction.

Forest plots summarizing hazard ratios and p-values for each cancer type help to quantitatively compare the effect of invasiveness on survival outcomes across different cancer types. Hazard ratios indicate the relative risk of an event (such as death) occurring in one group compared to another. Forest plots provide a clear visualization of these measures, aiding in the interpretation of the results and facilitating meta-analysis.

Boxplots of invasiveness scores for each cancer type allow for a graphical representation of the distribution of invasiveness levels within different cancer categories. This can help identify variations in invasiveness among cancer types and provide a concise summary of the data's central tendency, variability, and potential outliers.

Overall, these analytical and visualization approaches are essential for gaining comprehensive insights into the relationship between gene expression, invasiveness, survival outcomes, and their implications in different cancer types.

Boxplots and violin plots of invasiveness scores for each cancer type are important for visually summarizing and comparing the distribution of invasiveness levels within different cancer categories. These types of plots serve distinct purposes and provide complementary insights:

1. Boxplots: Boxplots are particularly useful for displaying the median, quartiles, and potential outliers in the distribution of invasiveness scores for each cancer type. They help researchers quickly identify the central tendency, spread, and skewness of the data. Outliers can be easily spotted as individual data points beyond the whiskers of the boxplot. Boxplots are concise and effective for making side-by-side comparisons of multiple cancer types, enabling the identification of variations in invasiveness levels.

2. Violin Plots: Violin plots combine aspects of boxplots and kernel density estimation to provide a more detailed representation of the data distribution. The width of the violin at a given point reflects the density of data points, with wider sections indicating higher data density. Violin plots offer a more nuanced view of the data's distribution and can reveal multi-modal distributions that might not be apparent in a simple boxplot. They are particularly useful when you want to understand both the distribution's summary statistics and the shape of the distribution itself.

Both boxplots and violin plots help researchers in the following ways:

- Comparison: Researchers can easily compare the spread and central tendency of invasiveness scores among different cancer types, identifying trends or differences.

- Outliers: Outliers, if present, can be detected and investigated for their potential significance.

- Data Distribution: Researchers can assess the symmetry, skewness, and multimodality of the invasiveness score distribution within each cancer type.

- Presentation: These plots are valuable for presentations and publications, as they convey information about the data distribution in a visually intuitive manner.

In summary, boxplots and violin plots provide researchers with a comprehensive understanding of the distribution of invasiveness scores for each cancer type, enabling better insights into the variation and characteristics of invasiveness levels across different types of cancer.  
  
2. figure2.R

This R code performs various analyses and visualizations related to gene expression and regulatory networks. Here's a summary of each line of the code:

1. Load necessary libraries for data manipulation and visualization.

2. Set the working directory to a specific location within the project.

3. Load external R scripts (`gene-reverse-network.R` and `get\_functions.R`).

4. Load a mechanistic network from a saved file.

5. Load RNASeq data for a specific cancer type (`BLCA`).

6. Filter the samples to include only those with a specific phenotype (`INV`).

7. Create a boxplot of quantile-normalized counts for a subset of primary tumor samples.

8. Order indices based on sample matching between data and cluster assignments.

9. Separate high and low phenotype indices for further analysis.

10. Perform hierarchical clustering on rows and columns of the filtered data.

11. Create a heatmap visualization of RNA-Seq expression data.

12. Load a regulatory network from a saved file.

13. Associate gene names with variables using a loaded dataset.

14. Load an activity matrix for transcription factors (TFs).

15. Normalize TF activity values based on the matrix.

16. Perform hierarchical clustering on rows and columns of the TF activity matrix.

17. Create a heatmap visualization of TF regulon activity matrix.

18. Calculate the difference in expression of target genes between high and low phenotypes.

19. Perform gene set enrichment analysis (GSEA) using sorted gene ranks and regulon sets.

20. Create a plot of top gene sets and enriched pathways.

21. Load information about top master regulators (MRs) from a saved file.

22. Create a new MR activity matrix and normalize its values.

23. Define colors for high and low phenotype groups.

24. Perform hierarchical clustering on the MR activity matrix.

25. Create a heatmap visualization of MR activity matrix, clustered by phenotype.

Gene-Reverse-Network.R

Certainly! In short, the provided R code implements functions related to gene regulatory network analysis, mutual information calculation, and cross-correlation computation. Here's a summary of what each function does:

aracne2 Function:

Calculates mutual information-based network edges between genes.

Generates permuted mutual information values and calculates p-values.

Applies FDR correction to p-values and sets network edges to 0 based on FDR-adjusted p-values.

dpi2 Function:

Extends mutual information matrix based on given source and target indices.

Applies the ARACNE algorithm to the extended matrix.

Returns a subset of the extended mutual information matrix between specified source and target indices.

cross.cor Function:

Calculates cross-correlation between two input matrices x and y.

Supports parallel processing for improved efficiency.

These functions are used for gene network analysis, mutual information calculation, and cross-correlation computation in the context of gene expression data. They facilitate the exploration and inference of gene regulatory relationships and interactions.

get\_functions.R

The provided R code encompasses a collection of functions for comprehensive gene expression analysis and regulatory network inference. It includes functionalities to load gene expression data, process it into modified matrices, and extract relevant gene information. The code facilitates the identification of high and low indices in phenotype data. It employs parallel computing for correlation matrix calculations and offers tools to infer gene regulons based on regulatory networks and correlation matrices. The functions aid in preparing gene networks suitable for the VIPER algorithm, and they perform Wilcoxon rank sum tests for gene expression comparisons. Additionally, the code enables the computation of gene activity scores while considering tumor purity. It supports the creation of regulons, performs gene set testing using the massiveGST algorithm, and calculates activity scores based on regulatory networks. These combined functionalities offer a versatile toolkit for gene expression analysis, network inference, and activity score computation within biological datasets.  
  
Let's dive into the detailed explanations of the terms VIPER algorithm, Wilcoxon rank sum test, and massiveGST algorithm.

VIPER Algorithm (Virtual Inference of Protein activity by Enriched Regulon):

The VIPER algorithm is a computational method used to infer the activity levels of transcription factors (TFs) and other regulatory molecules from gene expression data. It is particularly useful for studying gene regulatory networks and identifying key regulatory players. VIPER operates by scoring the activity of a given regulatory molecule based on the enrichment of its target genes within a dataset. It quantifies the extent to which the expression of target genes matches the predicted regulatory behavior. VIPER uses a ranked list of genes, often based on differential expression analysis, to determine the enrichment of target genes. It computes activity scores that reflect the regulatory influence of the TF or molecule, helping researchers gain insights into the underlying regulatory mechanisms in biological systems.

Wilcoxon Rank Sum Test:

The Wilcoxon rank sum test, also known as the Mann-Whitney U test, is a non-parametric statistical test used to compare two independent samples and assess whether they have different distributions. It is particularly useful when the assumptions of parametric tests, such as the t-test, are not met. The test ranks the combined data from both samples, then calculates the sum of ranks for each group. The test statistic measures the difference in ranks between the two groups, and its significance is determined by comparing it to the distribution of the test statistic under the null hypothesis of equal distributions. The Wilcoxon rank sum test is commonly employed in bioinformatics and genomics to identify genes or features that show significant differences in expression between experimental conditions, especially when dealing with small sample sizes or non-normal data.

massiveGST Algorithm (massive Gene Set Testing):

The massiveGST algorithm is a computational approach used for gene set testing, which assesses the enrichment or depletion of specific gene sets or pathways in high-dimensional genomic data, such as gene expression profiles. It extends the traditional gene set enrichment analysis (GSEA) framework to handle large-scale omics datasets efficiently. The algorithm works by ranking genes based on a relevant measure, often differential expression or other quantitative scores. It then evaluates the enrichment of gene sets by comparing the observed distribution of gene ranks within the set to a null distribution obtained through permutations or other methods. massiveGST provides a flexible and powerful way to analyze gene sets across various biological conditions and has applications in understanding functional differences between groups or conditions in large-scale genomics studies.

In summary, the VIPER algorithm is a tool for inferring regulatory molecule activity from gene expression data, the Wilcoxon rank sum test is a statistical method for comparing two sample distributions, and the massiveGST algorithm is an advanced approach for gene set testing in high-dimensional genomics datasets. These methods contribute to our understanding of gene regulation, differential expression, and functional analysis in biological research.

T-Test:

Assumptions:

Normality: The t-test assumes that the data in each group follows a normal distribution. Violations of this assumption can affect the accuracy of the test results.

Homogeneity of Variance: The variances of the two groups being compared should be roughly equal. Unequal variances can impact the validity of the t-test.

When to Use:

The t-test is appropriate when the data satisfies the assumptions of normality and homogeneity of variance. It is more powerful (sensitive) than the Wilcoxon test when these assumptions are met. It is commonly used when dealing with continuous data and large sample sizes.

Wilcoxon Rank Sum Test (Mann-Whitney U Test):

Assumptions:

Independent Samples: The test assumes that the observations in each group are independent of each other.

Similar Shape of Distributions: The test does not require the assumption of normality, but it assumes that the shape of the distribution for one group is not systematically shifted compared to the other group.

When to Use:

The Wilcoxon rank sum test is useful when the assumptions of normality and/or equal variances are not met. It is a non-parametric test and is robust to deviations from normality. Therefore, it is suitable for smaller sample sizes, skewed distributions, or situations where the data has outliers. It is commonly used when dealing with ordinal or interval data, and it provides a reliable alternative to the t-test in cases where parametric assumptions are not met.

In summary, choose the t-test when your data is normally distributed and has homogeneity of variance, especially for larger sample sizes. Choose the Wilcoxon rank sum test when your data deviates from normality, has unequal variances, or when dealing with smaller sample sizes or non-continuous data. It's important to assess the assumptions of each test and select the appropriate one based on the characteristics of your data and research question.  
  
3. figure3ab.R

The provided R code conducts a comprehensive analysis of gene expression data and master regulators in various cancer types. It starts by importing essential libraries for data manipulation, visualization, and parallel processing. An external script for heatmap plotting is sourced, and parallel computation with 20 workers is enabled. The code then sets the working directory and sources two scripts containing functions for gene network analysis and data loading. Subsequently, the analysis focuses on a list of different cancer types. Density plots are generated to visualize the distribution of activity values for transcriptional regulators in the first cancer type (LGG) and subsequently for all other types. A function is defined to identify common master regulators (MRs) across different analysis methods, utilizing gene set enrichment results. NES (Normalized Enrichment Score) plots are produced to display the significance and NES values of common MRs in each cancer type. The extracted NES information is saved to a CSV file for further exploration. A volcano plot illustrates the relationship between significance and NES values, color-coded by cancer type, and labeled with highly significant points. Supplementary figures are created, including a boxplot categorizing MR activities by NES and INV (INV High and INV Low). Another scatter plot visualizes the median activity of common MRs between INV High and INV Low phenotypes, with points colored by cancer type. Finally, the code arranges the NES and average activity plots into a revised version of Figure 3 for presentation or publication purposes. In essence, the code provides comprehensive insights into master regulator activity across different cancer types, combining data analysis, visualization, and statistical techniques to enhance our understanding of gene regulation in cancer.  
  
The provided R code performs a series of analyses and visualizations related to gene expression data and master regulators for different cancer types. Here's a summary of what the code does, line by line:

1. Library Imports: Various R packages, such as `data.table`, `ggplot2`, `doMC`, and others, are imported for data manipulation, visualization, and parallel processing.

2. Source URL: A URL is provided to source an R script (`heatmap.3.R`) from GitHub that likely contains custom heatmap plotting functions.

3. Parallel Processing: The code registers a parallel backend with 20 workers using the `registerDoMC` function to enable parallel computation.

4. Working Directory and Source Files: The working directory is set to a parent directory ('../scripts/') and two R scripts, 'gene-reverse-network.R' and 'get\_functions.R', are sourced. These scripts likely contain functions for gene network analysis and data loading, respectively.

5. Cancer Types: A vector `all\_inv` is defined, representing different cancer types (e.g., LGG, KIRP, PAAD, etc.).

6. Density Plots: Density plots of the activity matrices for the first cancer type (LGG) are generated using the `density` function and plotted using different colors and line types. These plots show the distribution of activity values of transcriptional regulators.

7. Common Master Regulators Function: A function `get\_common\_mrs` is defined to identify common master regulators (MRs) across different methods (RGBM + FGSEA, RGBM + GSVA, RGBM + Viper, ARACNE + Viper). It loads gene set enrichment analysis (GSEA) results and performs set intersections to find common MRs.

8. NES Plots: For each cancer type, NES (Normalized Enrichment Score) information for the common MRs is loaded and plotted. NES values are categorized based on cancer type, and the median NES values are computed for each cancer.

9. Write Output: The extracted NES information for common MRs is written to a CSV file named 'All\_TF\_Activity\_Information\_v2.csv' for further analysis.

10. Volcano Plot: A volcano plot is generated to visualize the significance (-log10(p-adjusted)) versus NES values for common MRs. Points are colored based on the cancer type, and some points with high significance are labeled.

11. Supplementary Figures: A boxplot is generated to show the distribution of median activities of common MRs categorized by NES and INV (INV High and INV Low) for supplementary analysis.

12. Average Activity Plot: A scatter plot is created to show the median activity of common MRs between INV High and INV Low phenotypes, with points colored by cancer type.

13. Revised Figure 3: The individual plots (NES and average activity) are arranged side by side as a revised version of Figure 3, likely for publication or presentation purposes.

The code primarily focuses on analyzing and visualizing the activity of master regulators in different cancer types, highlighting their significance and behavior across phenotypes. It combines data analysis, visualization, and statistical techniques to provide insights into the relationships between gene regulators and cancer phenotypes.  
  
4. Figure 3cd.R  
  
The provided R code conducts an in-depth analysis of gene expression data in the context of invasive cancers. It begins by importing a variety of R libraries for data manipulation, statistical analysis, and visualization. The code then processes and extracts relevant information from a dataset, identifying common molecular regulators (MRs) across different invasive cancer types. These common MRs' median activity is computed and visualized through a heatmap, which showcases their behavior across the various cancer types. The code also performs statistical tests on the MR activity data, producing a supplementary table of test results. Furthermore, for each invasive cancer, the code examines gene expression data and mechanistic network information, calculating and organizing MR activity values for distinct phenotypes. These findings are presented through boxplots, providing insights into the variations in MR activity within and between different invasive cancers. Overall, the code offers a comprehensive exploration of gene expression patterns and activity across multiple cancer types, contributing to a deeper understanding of the molecular underpinnings of invasive cancers.  
  
The provided R code appears to be an analysis and visualization script related to gene expression data and cancer research. It involves various data manipulation, statistical tests, and visualization steps. Here's a summary of what the code does:

1. Libraries: The code loads several R packages/libraries for data manipulation, statistical analysis, and visualization, including `data.table`, `ggplot2`, `doMC`, `heatmap3`, `gplots`, `igraph`, `lattice`, `parallel`, `reshape2`, `fgsea`, `factoextra`, `devtools`, `imager`, `gridExtra`, `purrr`, `dplyr`, `jpeg`, `png`, `grid`, `plyr`, `RColorBrewer`, `ggsignif`, `ggpubr`, `grImport2`, and `matrixStats`.

2. Function Definition: A function `revise\_wilcox\_test\_info` is defined to round and revise certain columns of the input data frame.

3. Data Loading and Manipulation:

- The code loads an external R script for a heatmap function from a GitHub repository.

- The code reads a data file `All\_TF\_Activity\_Information\_v2.csv`.

- The code extracts relevant columns from the data and stores them in the `output\_df` data frame.

- The code extracts unique cancer types and stores them in `all\_inv` vector.

- It identifies common MRs (molecular regulators) across different cancer types.

- It constructs a matrix `common\_mrs\_all\_activity\_matrix` to store the median activity of common MRs across all invasive cancers.

4. Heatmap Visualization:

- The code generates a heatmap using the `heatmap.3` function to visualize the median activity of common MRs across invasive cancers.

- The heatmap includes clustering of rows and columns, and color-coded activity values.

- The resulting heatmap is saved as a PDF file.

5. Statistical Analysis and Supplementary Information:

- The code performs a Wilcoxon rank sum test on the activity values of common MRs for different cancer types.

- The test results are processed and saved as a supplementary table `Supplementary\_Table\_S1\_Common\_MRs\_INV.csv`.

6. Additional Analysis and Visualization:

- For each invasive cancer type, the code loads an adjacency matrix for gene activity.

- It loads a mechanistic network and gene expression data.

- The code calculates high and low indices based on cluster assignments.

- The code iterates over common MRs and calculates their activity values for each phenotype (high and low).

- The results are stored in `inv\_activity\_df`.

7. Boxplot Visualization:

- The code creates boxplots to visualize the activity values of common MRs across different invasive cancers and phenotypes (high and low).

- The boxplots are facet-wrapped by cancer type.

- The resulting plots are saved as a PDF file.

Please note that this is a high-level summary of the code's functionality. If you have specific questions or need further details about any particular part of the code, feel free to ask.

5.figure 4.R  
  
The provided R code is a comprehensive computational analysis script focused on investigating gene regulatory networks and their activity in the context of specific cancer types. The script employs a wide range of R libraries for tasks such as data manipulation, visualization, statistical analysis, and enrichment analysis. It initially loads and processes data related to multiple cancers of interest, primarily focusing on a subset of cancers known as "INV enabled" cancers. These cancers are characterized by specific gene expression patterns associated with high and low invasive behavior. The analysis involves calculating median activities of transcription factors (TFs) in high and low invasive samples, identifying common master regulators (MRs) across INV enabled cancers, and evaluating their activities. The script further conducts Wilcoxon rank-sum tests to assess significant differences in TF activity between high and low invasive samples. Additionally, it performs downstream enrichment analyses to identify enriched Gene Ontology (GO) terms and pathways associated with the identified MRs in both INV High and INV Low phenotypes. The results are visualized through heatmaps, box plots, scatter plots, and enrichment plots, facilitating a comprehensive understanding of the gene regulatory landscape in these cancer types.  
  
The provided R code seems to be performing a comprehensive analysis of gene expression data for different cancer types, focusing on the INV High and INV Low phenotypes. The code involves various steps including data loading, processing, enrichment analysis, and generating visualizations. Here's a high-level summary of what the code does:

1. Data Preparation and Loading:

- Several R libraries are loaded, such as `data.table`, `ggplot2`, `doMC`, `heatmap3`, `gplots`, `igraph`, and others.

- A set of functions is sourced from external R scripts (`gene-reverse-network.R` and `get\_functions.R`).

2. Median Activity Calculation:

- For a list of cancer types (`INV\_enabled`), the median activity of transcription factors (TFs) is calculated in both INV High and INV Low samples.

- The data is organized in a data frame `tfs\_enabled\_activity\_info` with columns: TFs, Cancer, Median\_INV\_High, and Median\_INV\_Low.

3. Identification of Common TFs:

- TFs that are present in all the INV Enabled Cancers and have regulons of size >= 10 are selected.

- The list of common TFs is stored in `common\_tfs\_enabled\_list`.

4. Identification of Common MRs (Master Regulators):

- The script identifies common MRs across different cancers and their occurrences using a network-based approach.

5. Statistical Analysis - Median Activity Comparison:

- For each common MR, the code performs a Wilcoxon rank-sum test (Mann-Whitney test) to compare its activity between INV High and INV Low samples across different cancer types.

- The results of these tests are stored in the `inv\_enabled\_median\_comparison` data frame.

6. Visualization - Heatmap of Common MRs Activity:

- A heatmap is generated to visualize the median activity of common MRs across different cancers and phenotypes (INV High and INV Low).

7. Downstream Enrichment Analysis - GO Terms and Pathways:

- Enriched Gene Ontology (GO) terms are identified for INV High and INV Low specific MRs.

- Enriched pathways are identified for INV High and INV Low specific MRs.

8. Visualization - Enriched GO Terms and Pathways:

- Supplementary Figure S6A and Supplementary Figure S6B show the enriched GO terms for INV High and INV Low specific MRs.

- Supplementary Figure S7b shows the enriched pathways for INV Low specific MRs.

- Figure 4B visualizes enriched pathways for INV Low specific MRs.

9. Data Export:

- Various data frames containing results and lists of MRs and enriched terms are written to CSV files for further analysis and reporting.

The code performs an in-depth analysis of gene expression data and aims to identify and characterize master regulators and their activities in different cancer types. The code includes multiple steps of data manipulation, statistical analysis, and visualization to achieve its objectives.

6. Figure 5 a:  
  
The R code is part of a data analysis project focusing on gene expression and network analysis. The code involves importing necessary libraries and functions, setting the working directory, and sourcing relevant scripts. It reads data from specific files related to molecular regulators (MRs) in different cancer types. The code then defines a function called `get\_activity\_matrix\_info` to process and organize cancer activity matrices, which are collections of gene expression data across samples. Iterating through different cancer types, the code populates an `Activity\_Matrix` with MR activity information and saves it, along with associated vectors, to a file. Subsequently, the code creates a heatmap visualization using the `heatmap.3` function, customizing clustering and color assignments for both rows and columns. Lastly, the code conducts statistical tests on the `Activity\_Matrix`, generates a summary data frame, and saves it to a CSV file. Overall, the code aims to analyze and visualize gene expression patterns across various cancer types through heatmap representations and statistical assessments.  
  
This R code appears to be a part of a larger data analysis or research project related to gene expression and network analysis. It involves loading data, processing it, and creating visualizations such as heatmaps. Here's a summary of the code line by line:

1. The code imports various R libraries including `data.table`, `ggplot2`, `gplots`, `Matrix`, and others for data manipulation, plotting, and matrix operations.

2. The `source\_url` function is used to load a custom R function called `heatmap.3` from a specific URL.

3. The working directory is set to a specific location.

4. Two R scripts, `gene-reverse-network.R` and `get\_functions.R`, are sourced. These scripts likely contain additional functions or code needed for the analysis.

5. Data is read from files `All\_INV\_Low\_Specific\_MRs.csv` and `All\_INV\_High\_Specific\_MRs.csv` into `all\_cold\_mrs` and `all\_hot\_mrs` variables, respectively.

6. A list of cancer types called `inv\_neutral\_cancers` is created by excluding certain cancer types from the list of files in the `RGBM\_path`.

7. A function named `get\_activity\_matrix\_info` is defined. This function takes several arguments and performs operations on the provided data, presumably related to cancer activity matrices.

8. A loop iterates over each cancer type in the `inv\_neutral\_cancers` list. Within this loop, several operations are performed, including loading data, manipulating matrices, and appending information to various vectors.

9. A loop iterates over cancer types again, collecting additional information about the number of samples in each cancer type.

10. A matrix named `Activity\_Matrix` is created with initial values of 0. This matrix is intended to store activity information related to molecular regulators (MRs) across different cancer samples.

11. A loop starts for each cancer type, calling the `get\_activity\_matrix\_info` function to populate the `Activity\_Matrix` and other vectors.

12. The final `Activity\_Matrix`, along with other information, is saved to a file.

13. The code proceeds to create a heatmap visualization using the `heatmap.3` function. Clustering and color assignments are defined for rows and columns. The heatmap is customized with various parameters.

14. A data frame named `activity\_df` is created by performing statistical tests and calculations on the `Activity\_Matrix`.

15. The `activity\_df` is ordered and written to a CSV file.

This summary provides a general understanding of the code's purpose and functionality.  
  
7. Figure 5b:  
R script conducts a detailed analysis of gene expression data and immune response patterns across various cancer types. It encompasses multiple stages of data preprocessing, analysis, visualization, and result generation.

To begin, the script starts by loading essential R packages, such as `data.table`, `ggplot2`, `gplots`, and others, which are crucial for data manipulation, plotting, and statistical analysis.

The script proceeds to define a custom function, `convert\_character\_vector`, designed to convert specific columns within a data frame into character vectors. This function will be used later to process data efficiently.

Following the function definition, the script delves into the survival analysis phase. It reads data files pertaining to specific cancer types and constructs activity matrices based on gene expression data. This stage involves complex operations like loading and processing datasets, calculating high and low indices, and extracting relevant information for further analysis.

The subsequent step involves the analysis of the PRECOG dataset, which contains gene expression information for various cancer types. The script meticulously processes this dataset by mapping gene IDs to gene names, creating expression matrices, and identifying high and low indices based on specific criteria.

Moving on, the script shifts its focus to differential analysis. It conducts comparative analyses between "INV High" and "INV Low" groups, employing statistical tests and fold change calculations. The outcomes of these analyses are then used to generate a heatmap visualization, offering an insightful depiction of the differentially active genes and their expression patterns across cancer types.

The script also generates heatmaps to illustrate the gene activity matrices for different cancer types, emphasizing immune response categories such as "ICR High" and "ICR Low." This clustering visualization provides a clear picture of how gene expression varies across samples and cancer types.

Finally, the script wraps up by exporting the generated heatmaps and analysis results. PDF files are created to store the heatmap visualizations, while CSV files house comprehensive information, such as statistical measures, fold changes, and p-values derived from the differential analyses.

In summary, the script performs an intricate sequence of tasks, from loading and processing gene expression data to conducting statistical analyses and producing informative visualizations. It serves as a comprehensive analysis pipeline to uncover patterns of gene expression and immune response in the context of various cancer types.

Key steps :   
Library Loading: Several R packages are loaded using the library() function, including data.table, ggplot2, gplots, Matrix, and others.

Function Definition: The script defines a function named convert\_character\_vector, which converts columns of a data frame to character vectors.

Survival Analysis Data Loading and Processing: The script loads data for survival analysis related to different cancer types. It reads specific files, processes the data, and creates activity matrices for different gene sets.

PRECOG Dataset Analysis: The script loads and analyzes data from the PRECOG dataset, which contains gene expression information for different cancer types. It processes the data, maps gene IDs to gene names, and calculates gene activity matrices.

Differential Analysis and Heatmap Generation: The script performs differential analysis between "INV High" and "INV Low" groups and generates a heatmap visualization of the results. It also performs statistical tests, calculates fold changes, and saves the results to a CSV file.

Activity Matrix Visualization and Clustering: The script generates heatmaps of gene activity matrices for different cancer types, clustered by immune response categories ("ICR High" and "ICR Low"). It visualizes the gene activity patterns using heatmaps.

Data Export: The script exports the generated heatmaps and analysis results to PDF files and CSV files.

8. run\_ARCANE\_real.R  
  
The provided R code appears to be related to gene regulatory network (GRN) construction using the `corto` package and the ARACNE-AP algorithm. Let's break down the code step by step:

1. Importing Libraries:

- The code starts by loading several R packages, such as `corto`, `doMC`, `foreach`, and `igraph`, which are likely used for various data manipulation, parallel processing, and network analysis tasks.

2. Setting Working Directory and Sourcing Functions:

- The working directory is set to a specific path, likely where the script and data files are located.

- The code then sources two R scripts: `get\_functions.R` and `gene-reverse-network.R`, which presumably contain custom functions needed for the GRN construction.

3. Loading Mechanistic Network:

- The mechanistic network is loaded from an RData file located in the "../Data/Others/me\_net\_full.Rdata" path.

4. GRN Construction and Saving:

- A variable `filename` is assigned the value "MESO".

- The output path for the results is defined based on the `filename`.

- The variable `sample\_name` is created by concatenating `filename` with "\_Full\_".

- The `loading\_data` function (likely defined in the sourced scripts) is called with the arguments `filename` and `M`, resulting in the variable `out`.

- The log2-transformed data matrix `D` is created from the first element of `out`.

- The list of transcription factors (TFs) is extracted from the variable `M`.

5. Network Construction:

- The GRN construction process begins using the `corto` function.

- The `corto` function is called with various parameters:

- `D`: The data matrix.

- `tfs`: The list of transcription factors.

- `nbootstraps`: Number of bootstraps set to 100.

- `nthreads`: Number of threads set to 20 for parallel processing.

- `p`: A threshold parameter set to 1e-3.

- `verbose`: Set to `TRUE` to enable verbose output during the process.

- The resulting network is stored in the `net` variable.

6. Constructing Final Adjacency Matrix:

- A matrix `V\_final` is initialized with zeros, with rows and columns named according to the transcription factors and targets in the network.

- A loop iterates over each transcription factor in the network:

- For each TF, the associated target genes are extracted.

- The likelihood scores from the network results are stored in the appropriate cells of `V\_final` for the TF-target pairs.

- The total number of non-zero entries in `V\_final` is printed.

- The `V\_final` matrix is saved as a CSV file named "Final\_Adjacency\_Matrix\_v2.csv" in the specified `outputpath`.

Overall, this R code seems to perform gene regulatory network construction using the ARACNE-AP algorithm and saves the resulting network as an adjacency matrix. The code involves loading data, applying the algorithm, and processing the results for network analysis.