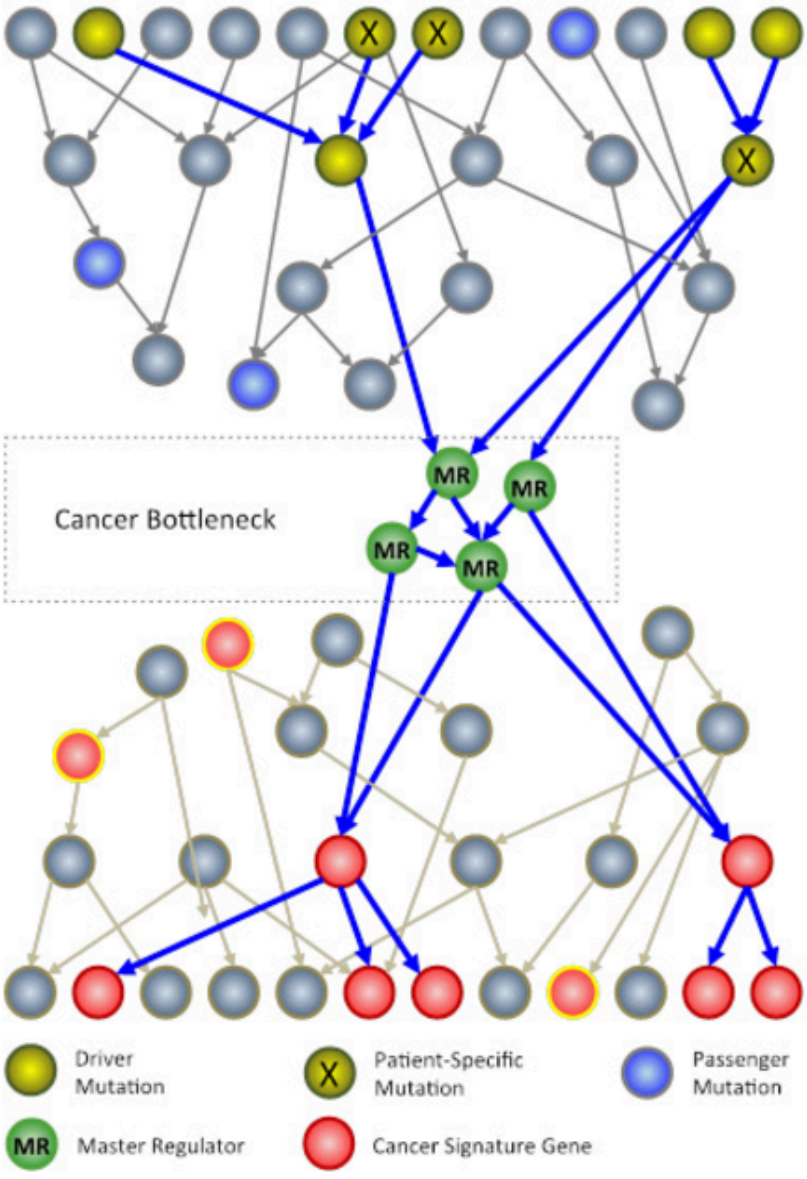
TASK 1: Define the problem statement

Background



Schematic representation of how cancer bottlenecks channel upstream genetic alterations to initiate cancer driving programs.

Source: http://califano.c2b2.columbia.edu/cancer-systems-biology

Alzheimer's disease (AD) is a progressive neurodegenerative disorder characterized by memory loss and cognitive decline. Despite extensive research, the molecular mechanisms driving its onset and progression remain poorly understood. While genome-wide studies have identified numerous genes associated with AD, understanding how these genes interact within regulatory networks to produce the disease phenotype is still a challenge.

Building on the work of Dr. Andrea Califano's lab (Columbia University), which focuses on the ***bottleneck hypothesis*** in cancer, this project seeks to apply similar concepts to Alzheimer's disease. The bottleneck hypothesis indicates that although different genetic events may lead to a uniform disease phenotype, their **effects converge on a small set of master regulator (MR) genes** that control disease-driving processes within highly connected regulatory networks. These MRs, while not necessarily mutated, act as key bottlenecks, regulating the pathways and cellular programs that maintain disease states. Identifying these bottlenecks can provide mechanistic insights into disease biology and uncover potential therapeutic vulnerabilities.

Using advanced systems biology approaches such as ARACNe (to construct gene regulatory networks) and VIPER (to infer protein activity), this project aims to identify the MRs/bottleneck genes driving AD-specific regulatory events. These MRs represent critical nodes of vulnerability, offering new targets for therapeutic intervention such as drug targets. This work builds on the hypothesis that targeting aberrant activity in these MRs can interrupt disease-driving programs and ultimately mitigate disease progression.

Domain Selection

Selected Domain: Regulatory Network Construction and Analysis
Specific Application: Identification of Bottleneck genes (master regulators) associated with Alzheimer's Disease (AD)

Bottleneck genes: Key regulatory proteins, known as bottlenecks or master regulators, play a crucial role in controlling gene activity. Bottlenecks are proteins that have a high betweenness

centrality, meaning they are network nodes that have many "shortest paths" going through them. Bottlenecks are key connector proteins that are more likely to be essential proteins.

Problem Statement and Biological Goal

The biological goal of this project is to identify and rank key bottleneck genes, specifically transcription factors, that act as master regulators (MRs) driving Alzheimer's disease (AD)-specific gene expression patterns. Using systems biology approaches such as ARACNe (Algorithm for the Reconstruction of Accurate Cellular Networks) and VIPER (Virtual Inference of Protein Activity by Enriched Regulon analysis), we aim to construct gene regulatory networks and infer protein activity levels from RNA-seq data derived from post-mortem brain tissue of Alzheimer's patients and age-matched healthy controls.

This analysis aims to validate the hypothesis that a disease phenotype may be controlled by a smaller, more manageable set of master regulators, despite having different initiating mutations.

Project Goals

1. Analyze RNA-seq data for Alzheimer's from the GEO database.
2. Identify transcription factors (genes) that act as bottlenecks/master regulators in the dataset.
3. Validate the biological significance of these shared bottleneck genes by ranking them.
4. Characterize the regulatory networks controlled by these common master regulators

Expected Outcomes

1. A comprehensive list of shared bottleneck transcription factors in a chosen Alzheimer dataset.
2. Validation of their biological significance through network analysis.
3. Understanding of common regulatory mechanisms in Alzheimer's disease.
4. Potential identification of novel therapeutic targets for the disease.

Data Sources

Primary data source: Gene Expression Omnibus (GEO) RNA-sequencing data.

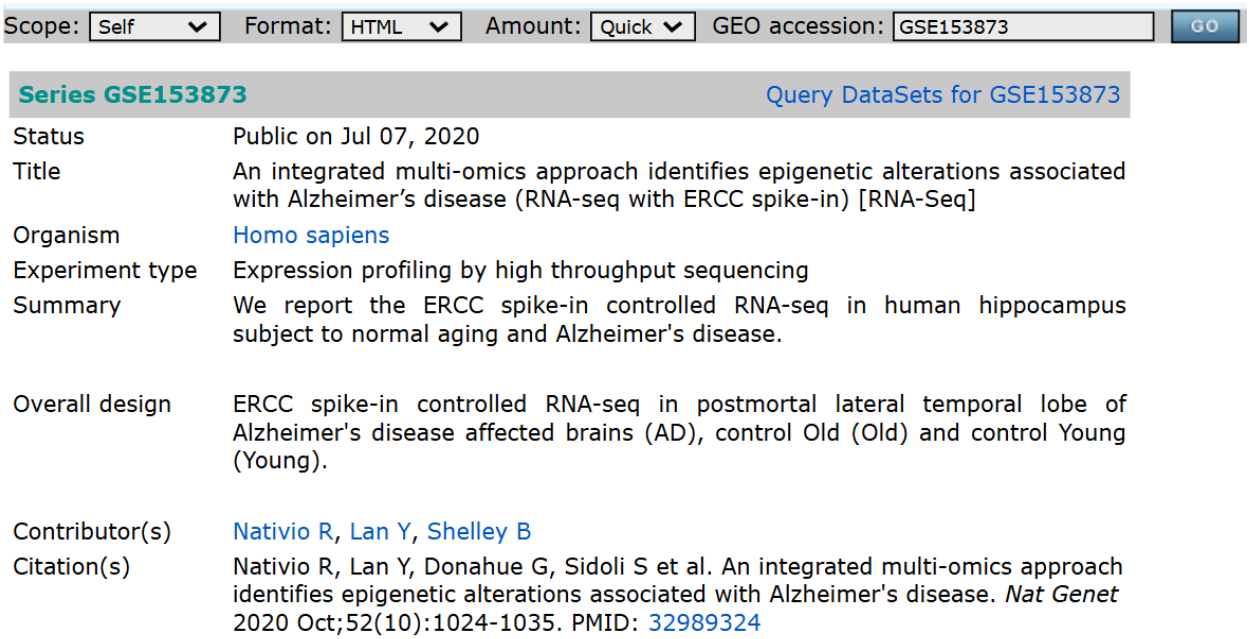
Chosen dataset: GSE153873

Title: An integrated multi-omics approach identifies epigenetic alterations associated with Alzheimer's disease (RNA-seq with ERCC spike-in) [RNA-Seq]

This study investigates **epigenetic and gene expression changes** associated with Alzheimer's Disease (AD) using an RNA sequencing (RNA-seq) approach. The researchers performed **RNA-seq with ERCC spike-ins** on samples from the hippocampus, a brain region crucial for memory and often impacted in Alzheimer's.

RNA was collected from the lateral temporal lobe of postmortem brains, and classified into three groups:

- AD group (patients with Alzheimer's disease).
- Control Old group (elderly individuals without AD).
- Control Young group (younger individuals without AD).



Scope: Self Format: HTML Amount: Quick GEO accession: GSE153873 GO

Series GSE153873 Query DataSets for GSE153873

Status	Public on Jul 07, 2020
Title	An integrated multi-omics approach identifies epigenetic alterations associated with Alzheimer's disease (RNA-seq with ERCC spike-in) [RNA-Seq]
Organism	Homo sapiens
Experiment type	Expression profiling by high throughput sequencing
Summary	We report the ERCC spike-in controlled RNA-seq in human hippocampus subject to normal aging and Alzheimer's disease.
Overall design	ERCC spike-in controlled RNA-seq in postmortal lateral temporal lobe of Alzheimer's disease affected brains (AD), control Old (Old) and control Young (Young).
Contributor(s)	Nativio R, Lan Y, Shelley B
Citation(s)	Nativio R, Lan Y, Donahue G, Sidoli S et al. An integrated multi-omics approach identifies epigenetic alterations associated with Alzheimer's disease. *Nat Genet* 2020 Oct;52(10):1024-1035. PMID: 32989324

Tools/software/algorithms

- ARACNE
- VIPER
- Random forest

TASK 2: Prepare a project plan

Problem Statement and Biological Goal

The biological goal of this project is to identify and rank key bottleneck genes, specifically transcription factors, that act as master regulators (MRs) driving Alzheimer's disease (AD)-specific gene expression patterns. Using systems biology approaches such as ARACNe (Algorithm for the Reconstruction of Accurate Cellular Networks) and VIPER (Virtual Inference of Protein Activity by Enriched Regulon analysis), we aim to construct gene regulatory networks and infer protein activity levels from RNA-seq data derived from post-mortem brain tissue of Alzheimer's patients and age-matched healthy controls.

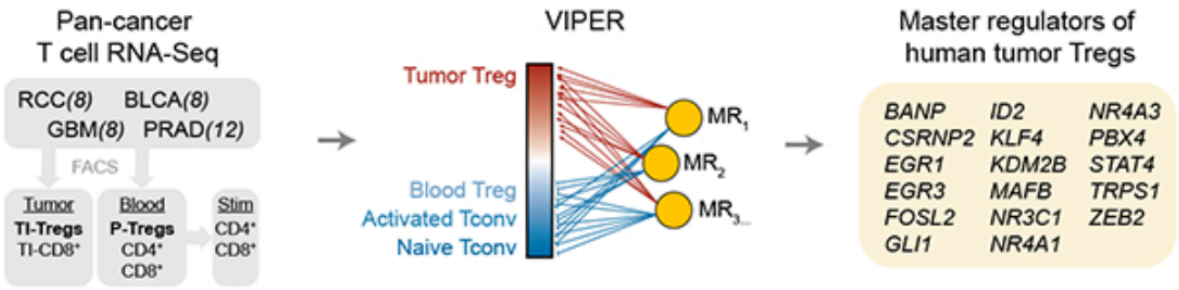
This analysis aims to validate the hypothesis that a disease phenotype may be controlled by a smaller, more manageable set of master regulators, despite having different initiating mutations.

Overall approach

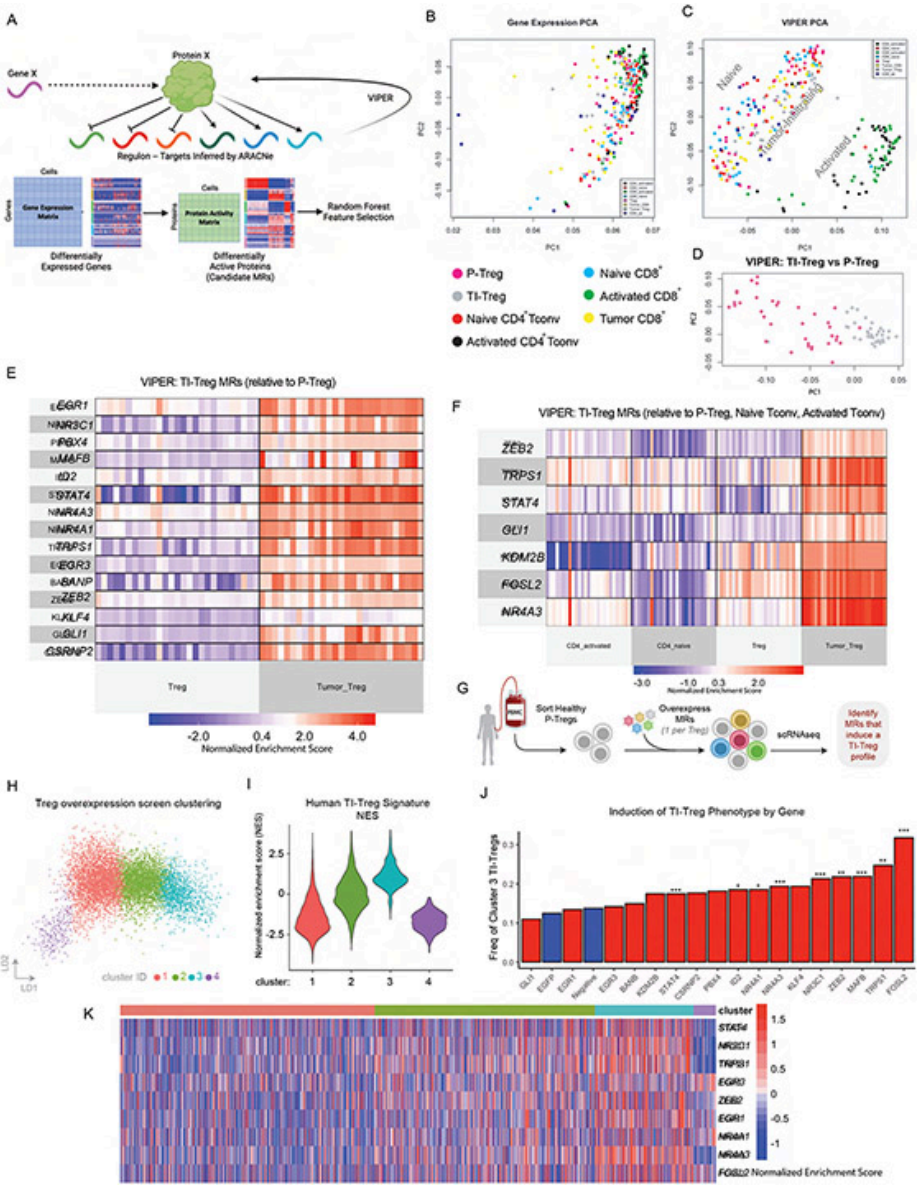
Literature review

To understand the general workflow of our project, we referred to the work done by Dr. Andrea Califano and his team where they have used a similar approach.

We went through his paper: Systematic Elucidation and Pharmacological Targeting of Tumor-Infiltrating Regulatory T Cell Master Regulators



In this, they have also used certain tools and software to identify and rank key master regulators (bottlenecks) in the case of a cancer database. The researchers in this paper are investigating the transcriptional signatures of tumor-infiltrating regulatory T cells (TI-Tregs) across various cancer types, including glioblastoma and renal carcinoma. They utilize advanced computational methods, such as the VIPER algorithm and ARACNe, to identify candidate master regulator (MR) proteins that are differentially expressed in TI-Tregs compared to peripheral blood Tregs and other T cell subtypes. Their findings suggest that these MR proteins play a crucial role in the infiltration and retention of TI-Tregs in the tumor microenvironment.



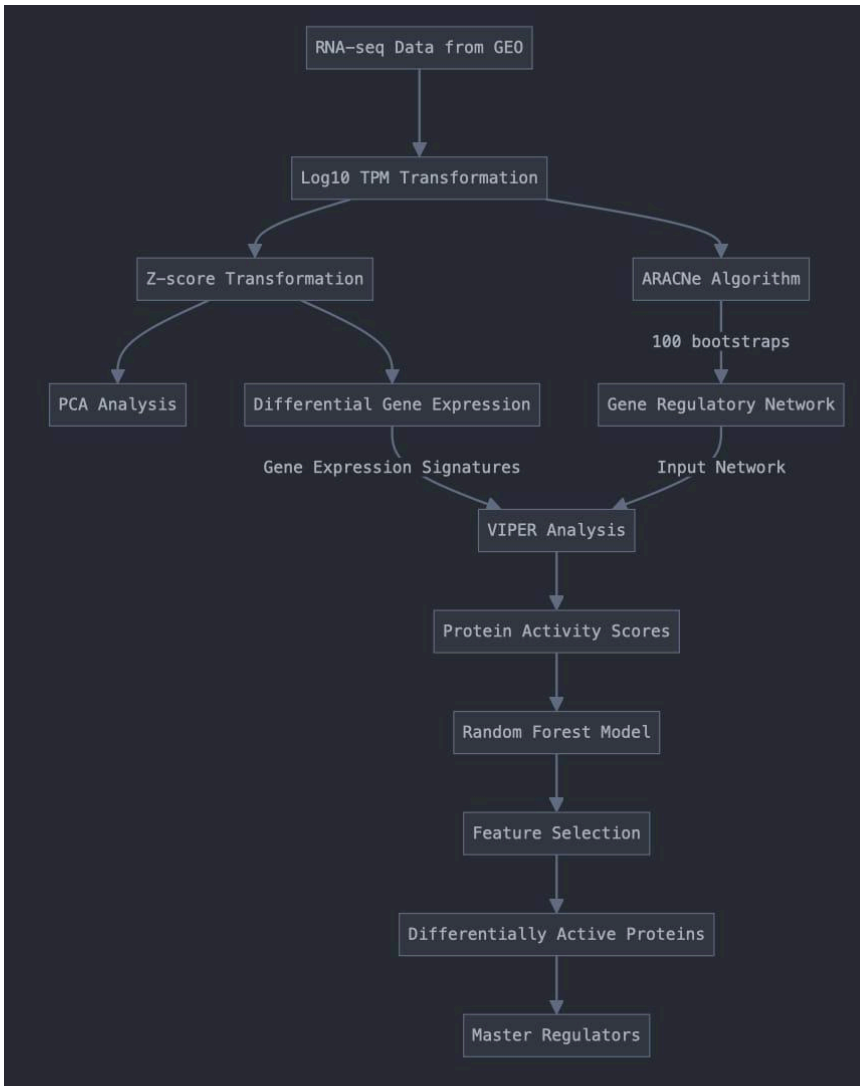
Following his approach, we planned to do the same with an Alzheimer's dataset.

Project Workflow/Methodology

Using systems biology approaches such as ARACNe (Algorithm for the Reconstruction of Accurate Cellular Networks) and VIPER (Virtual Inference of Protein Activity by Enriched Regulon analysis), we aim:

- To construct gene regulatory networks and
- infer protein activity levels

- Consolidate and rank Master regulators



NOTE: The above workflow was the ideal way we should have done it, but due to time constraints, we were not able to complete everything. Following is the methodology that we were able to complete during the 2 days of the project.

Data Extraction and Normalization

RNA-seq data was obtained from the Gene Expression Omnibus (GEO) repository. The datasets included gene expression values measured in star counts, facilitating the comparison of gene expression across different samples. GEO was chosen due to its extensive database of publicly available RNA-seq data for diseases such as Alzheimer's.

To prepare the data for analysis, a log10 TPM transformation was applied. This transformation was used to reduce the skewness of the data distribution, thereby making the expression values more suitable for downstream statistical analyses. The log transformation helped in normalizing the wide range of gene expression levels, ensuring that extreme values did not disproportionately influence the results.

Following the log10 TPM transformation, the data were standardized using a Z-score transformation. This process involved converting each gene's expression value into a Z-score, which represented the number of standard deviations the value was from the mean. By applying this transformation, the dataset was centered around a mean of zero and a standard deviation of one. This step was essential for facilitating comparisons across different datasets and for identifying outliers in the gene expression profiles.

Construction of the Gene Regulatory Network Using the ARACNe Algorithm

To visualize the relationships between genes, a gene regulatory network (GRN) was constructed using the ARACNe (Algorithm for the Reconstruction of Accurate Cellular Networks) algorithm. The ARACNe algorithm inferred regulatory interactions based on mutual information between gene pairs. To ensure robustness and reliability, the algorithm was run with **100 bootstrap iterations**. The output from ARACNe was used to generate an input network that served as a foundation for further analyses.

The input for ARACNe was the log10 TPM transformed file of the gene expression matrix and a list of transcription factors taken from the TRRUST database.

How does ARACNe work?

ARACNe (**Algorithm for the Reconstruction of Accurate Cellular Networks**) is a computational method designed to infer gene regulatory networks (GRNs) from gene expression data. By identifying direct interactions between transcription factors (TFs) and their target genes, ARACNe reconstructs networks that reflect the underlying biological regulatory mechanisms. The algorithm relies on the concept of **mutual information (MI)**, a non-linear measure that captures dependencies between gene expression levels (ARACNe-AP: gene network reverse engineering through adaptive partitioning inference of

mutual information).

The ARACNe pipeline:

ARACNe requires two main inputs: **Gene Expression Profile (GEP) data** and a **predefined list of gene regulators** (such as transcription factors, TFs). The process of running ARACNe consists of three key steps:

1. Mutual Information (MI) Threshold Estimation: In the preprocessing stage, a significance threshold for mutual information (MI) values is determined based on the provided GEP data. This threshold depends on the number of samples in the dataset and helps identify statistically significant MI values for further analysis.

Mutual information (MI) quantifies the degree of dependency between two variables—in this case, the expression levels of two genes. Unlike correlation, which only captures linear relationships, MI can detect both linear and non-linear dependencies. In ARACNe, MI is calculated for all possible pairs of genes in the dataset to identify potential regulatory relationships. High mutual information between two genes suggests that one may regulate the other directly.

2. Bootstrapping and MI Network Reconstruction: In this step, MI networks are generated through a bootstrapping process, where the GEP data is randomly sampled multiple times. For N bootstrap iterations, N MI networks are reconstructed. This process includes three sub-steps:

> **(a) MI Calculation:** The mutual information is computed for every TF-target gene pair after rank-transforming the GEP data. This transformation ensures robustness against outliers.
>
> **(b) Filtering Insignificant Connections:** Connections that do not meet the previously determined MI significance threshold are removed, ensuring only statistically significant interactions are retained.
>
> **(c) Indirect Interaction Removal (DPI):** To eliminate indirect regulatory relationships, the Data Processing Inequality (DPI) filter is applied. This step reduces false positives by ensuring that only direct interactions are maintained.

3. Building the Consensus Network

A consensus network is constructed by evaluating the statistical significance of the edges

(interactions) detected across all bootstrap iterations. The significance of each edge is assessed using a Poisson distribution to estimate the likelihood of observing that interaction repeatedly. Only edges with a P-value < 0.05 (corrected using the Bonferroni method) are retained in the final network, ensuring that only reliable and consistent interactions are included.

Output: Network file

1. The regulator.
2. The target.
3. The MI (Mutual Information) of the pair.
4. The p-value of the pair (assessed during the consolidation step by integrating the bootstraps).

Network Visualization and Betweenness Centrality Analysis Using Cytoscape

The gene regulatory network generated by ARACNe was imported into Cytoscape for visualization and interpretation. Cytoscape provided an interactive platform to explore the network structure, visualize relationships between genes, and identify key nodes. Betweenness centrality was calculated within Cytoscape to determine the importance of individual nodes in the network. Nodes with high betweenness centrality represented possible hubs or bottlenecks in the network, which were likely to influence the flow of information between other genes. This analysis would allow for the identification of potential regulatory genes that played central roles in the network topology.

VIPER Analysis

The VIPER (Virtual Inference of Protein-activity by Enriched Regulon) algorithm was applied to infer protein activity levels from the gene expression signatures. This analysis integrated the regulatory network generated by ARACNe with the gene expression data (which was z-transformed). VIPER calculated activity scores for various proteins, offering a functional interpretation of the gene expression changes. These inferred protein activity scores allowed for the identification of proteins that were likely to be active or inactive in the biological context under study.

How does VIPER work?

The **VIPER (Virtual Inference of Protein-activity by Enriched Regulon)** algorithm is a computational method designed to infer protein activity levels based on gene expression data and gene regulatory networks. Unlike traditional differential expression analysis, which focuses solely on changes in mRNA levels, VIPER aims to identify the functional activity of proteins (e.g., transcription factors or signaling proteins) by considering their regulatory impact on target genes (also known as **regulons**). VIPER is particularly useful in interpreting complex gene expression profiles in contexts such as cancer biology, where protein activity can be more informative than mRNA expression alone.

1. Gene Expression Input

The initial step involves obtaining the necessary data inputs for VIPER. Specifically, VIPER requires:

- **Gene Expression Data**: This is typically derived from RNA-seq experiments or microarray studies. The data captures the expression levels of thousands of genes across different conditions, such as disease versus normal samples or treated versus untreated samples. The expression data is often transformed into log-transformed Transcripts Per Million (**Log10 TPM**) and Z-scores to ensure normalization and comparability across samples.
- **Regulatory Network**: The regulatory network is generated using the ARACNe algorithm. ARACNe infers relationships between transcription factors and their target genes based on mutual information, which identifies statistically significant correlations between gene pairs. The output is a **Gene Regulatory Network (GRN)** that maps transcription factors to their regulons — the set of genes they regulate. These interactions can be either activating or repressing, and the GRN may include confidence scores reflecting the strength of each inferred relationship.

Together, the gene expression data and the GRN serve as the foundational inputs that VIPER uses to infer protein activity.

2. Regulon Activity Inference

At this stage, VIPER focuses on evaluating the activity of each transcription factor by examining the expression patterns of its associated regulon — the set of target genes identified in the regulatory network. This process involves several key steps:

Regulon Analysis: For each transcription factor, VIPER examines how the expression levels of its target genes in the regulon deviate from a background distribution or null model. The background model is generated through permutations of the data to create a distribution of expected random behavior. This allows VIPER to detect patterns that are significantly different from what would occur by chance.

Enrichment Calculation: Using a statistical approach known as aREA (analytic Rank-based Enrichment Analysis), VIPER measures the extent to which the target genes of a transcription factor are enriched among the differentially expressed genes. If a transcription factor is an activator, its regulon is expected to show coordinated upregulation when the TF is active. Conversely, if the TF is a repressor, its targets should exhibit coordinated downregulation when the TF is active. The degree of enrichment (or depletion) serves as an indicator of the transcription factor's activity.

3. Protein Activity Measurement

The final step in the VIPER process is the calculation of Protein Activity Scores for each transcription factor. These scores quantify the inferred activity of each TF in the biological context being studied.

VIPER assigns a score to each transcription factor that reflects the extent and significance of its inferred activity. A positive score indicates that the transcription factor is likely activated, driving the upregulation of its target genes. Conversely, a negative score suggests that the transcription factor is repressed or inhibited, leading to the downregulation of its targets. To ensure the reliability of the activity scores, VIPER employs permutation testing or bootstrapping to compare the observed enrichment against a null distribution. This allows VIPER to determine the p-values and false discovery rates (FDR) for each inferred activity score, helping to identify the most significant regulators. The resulting protein activity scores can be visualized and analyzed to identify key regulators driving the biological changes observed in the gene expression data (through a heatmap).

We were able to complete till VIPER analysis, further steps will be completed in near future.

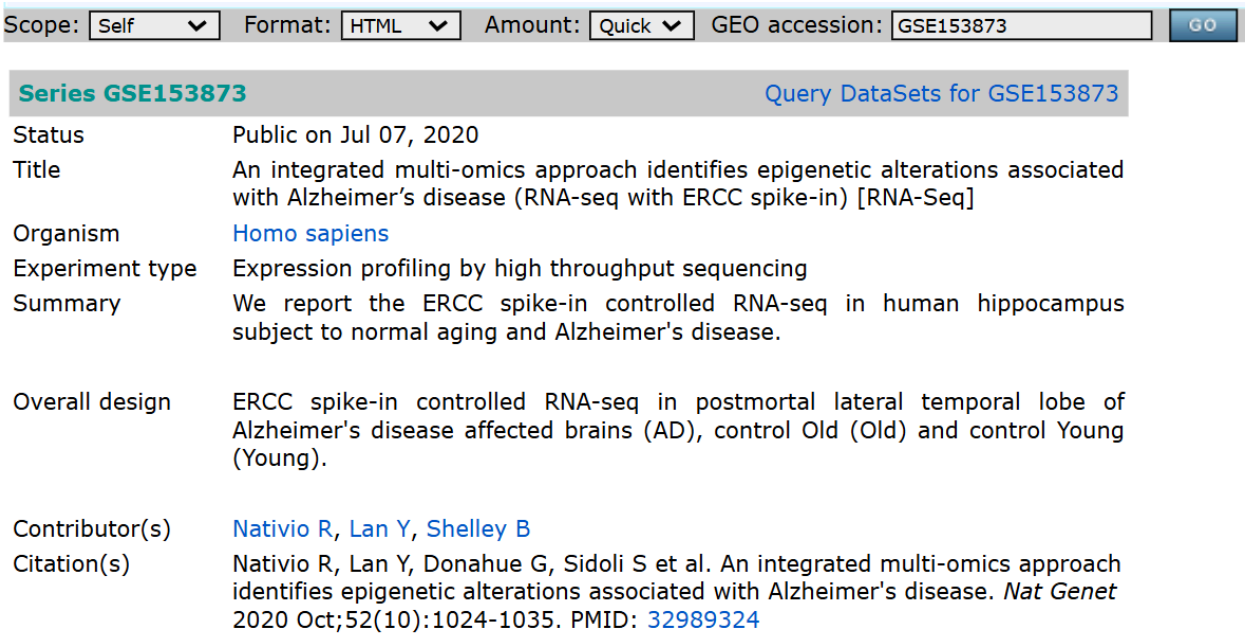
Input/Output sources:

ARACNe

Input

1. Log10 TPM transformed gene expression data

Primary data source: Gene Expression Omnibus (GEO) RNA-sequencing data.


Scope: Self Format: HTML Amount: Quick GEO accession: GSE153873 GO

Series GSE153873 Query DataSets for GSE153873

Status	Public on Jul 07, 2020
Title	An integrated multi-omics approach identifies epigenetic alterations associated with Alzheimer's disease (RNA-seq with ERCC spike-in) [RNA-Seq]
Organism	Homo sapiens
Experiment type	Expression profiling by high throughput sequencing
Summary	We report the ERCC spike-in controlled RNA-seq in human hippocampus subject to normal aging and Alzheimer's disease.
Overall design	ERCC spike-in controlled RNA-seq in postmortal lateral temporal lobe of Alzheimer's disease affected brains (AD), control Old (Old) and control Young (Young).
Contributor(s)	Nativio R, Lan Y, Shelley B
Citation(s)	Nativio R, Lan Y, Donahue G, Sidoli S et al. An integrated multi-omics approach identifies epigenetic alterations associated with Alzheimer's disease. *Nat Genet* 2020 Oct;52(10):1024-1035. PMID: 32989324

Chosen dataset: GSE153873

Title: An integrated multi-omics approach identifies epigenetic alterations associated with Alzheimer's disease (RNA-seq with ERCC spike-in) [RNA-Seq]

This study investigates **epigenetic and gene expression changes** associated with Alzheimer's Disease (AD) using an RNA sequencing (RNA-seq) approach. The researchers performed **RNA-seq with ERCC spike-ins** on samples from the hippocampus, a brain region crucial for memory and often impacted in Alzheimer's.

RNA was collected from the lateral temporal lobe of postmortem brains, and classified into three groups:

- AD group (patients with Alzheimer's disease).
- Control Old group (elderly individuals without AD).

- Control Young group (younger individuals without AD).

27136 genes | 30 samples

This particular database was chosen because this was the only dataset available in the format of star counts (all other data was in RAW) for alzheimers on GEO. STAR counts represent aligned read counts that have been pre-processed and aligned to a reference genome using the STAR aligner. This means the data is already cleaned and processed, reducing the need for time-consuming and complex pre-processing steps. Unlike raw RNA-seq data, which requires quality control and normalization steps, STAR counts are typically already normalized for library size and sequencing depth, which makes them directly usable for comparative gene expression analysis. This is crucial, especially when the dataset is being used for downstream analyses like identifying gene expression patterns and inferring regulatory networks.

The above gene expression data was then transformed to log10 TPM:

Initially, gene expression data comes as raw read counts from RNA-seq experiments. These counts reflect the number of sequencing reads that map to each gene, but they need to be normalized to account for factors like differences in gene length and sequencing depth (library size).

Calculating TPM: The TPM for each gene is calculated using the following steps:

- **Normalize for gene length**: The raw count for each gene is divided by the length of the gene (in kilobases) to correct for gene length biases. This results in a value that represents the number of reads per kilobase of transcript per million total reads.
- **Normalize for sequencing depth**: The sum of the normalized counts for all genes is then divided by one million to account for differences in sequencing depth across samples. This gives the final **TPM** value, representing the relative abundance of each gene in a sample, normalized for both gene length and sequencing depth.

After TPM normalization, the values are often transformed using a **log10 transformation**. This is done because gene expression data typically spans several orders of magnitude, and applying the log transformation compresses the scale of these values, making them more manageable.

$$\text{Log10(TPM)} = \log_{10}(\text{TPM} + 1)$$

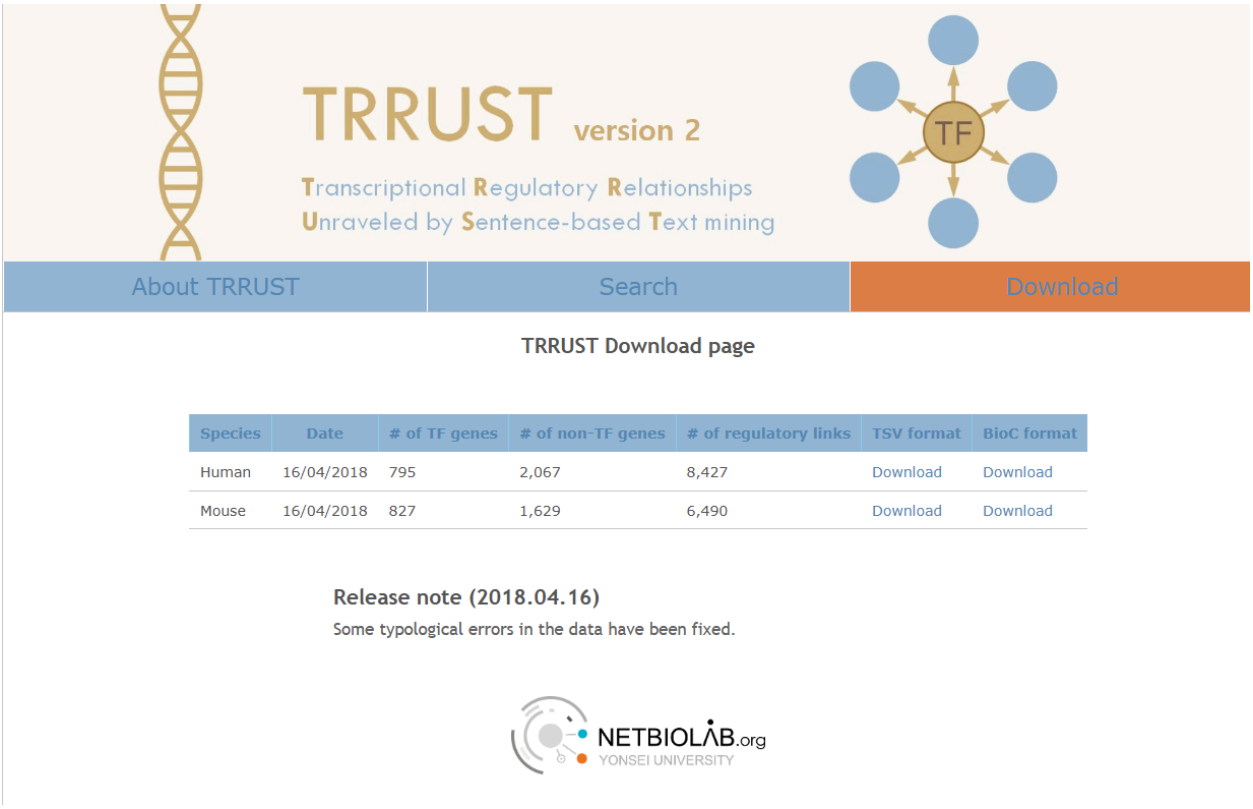
(The "+1" is added to avoid taking the log of zero, which would be undefined.)

In order to perform the log10 TPM transformation, gene length of transcription factors was taken from ENSEMBL Biomart.

2. List of transcription factors (regulators)

In addition to the gene expression matrix, ARACNe also needs a predefined list of transcription factors.

The list of transcription factors was taken from the TRRUST database (for humans).



Source: https://www.grnpedia.org/trrust/downloadnetwork.php

Output

The output of the ARACNe (Algorithm for the Reconstruction of Accurate Cellular Networks) algorithm consists of an inferred gene regulatory network (GRN), which describes the relationships and interactions between genes based on their co-expression patterns. ARACNe uses gene expression data to identify pairwise relationships between genes and their potential regulators, particularly focusing on transcription factors and target genes. The output can be visualized, analyzed, and interpreted in various ways to study gene regulation in

a given biological context. Below is an elaboration on the specific elements included in the output network file:

1. The regulator: A regulator is typically a transcription factor (TF) or any gene that is presumed to influence the expression of other genes. In the context of ARACNe, the regulator is the gene whose expression is considered to control or modulate the expression of a target gene. In the output, each edge (connection) between two genes will list the regulator gene. For example, if Gene A is a transcription factor that regulates Gene B, Gene A will be listed as the regulator for that specific interaction.

Key Points:

- Transcription Factors are often the primary regulators.
- Regulators can also be other genes that play an indirect role in the regulatory network, not necessarily transcription factors.

2. The target: The target refers to the gene that is regulated by the regulator. In the case of ARACNe, the target gene is predicted to be affected by the expression of the regulator, often through a transcriptional mechanism. The target gene can be any gene in the network, including other transcription factors, signaling molecules, or structural genes, depending on the context of the biological process being studied.

3. The MI (Mutual Information) of the pair: Mutual Information (MI) is a measure of the statistical dependence or relationship between two genes (the regulator and the target). MI quantifies how much knowing the expression of one gene reduces uncertainty about the expression of another gene. MI is calculated by evaluating the co-expression patterns between the two genes across multiple samples or conditions. A high MI value indicates that the expression of the two genes is highly correlated, which suggests that the genes are co-regulated or related.

- High MI values typically suggest that the regulator and target genes have strong co-expression patterns, implying a significant regulatory relationship.
- Low MI values indicate weaker or no co-expression, suggesting a minimal or absent regulatory interaction.

4. The p-value of the pair (assessed during the consolidation step by integrating the bootstraps): The p-value assesses the statistical significance of the inferred regulatory relationship between the regulator and target gene. It is calculated during the consolidation step of ARACNe by

integrating the results from bootstrapping (resampling) procedures. Bootstrapping is used to generate multiple random resamples of the data, and the p-value represents the likelihood that the observed interaction is not due to chance. A low p-value suggests that the interaction between the regulator and target gene is statistically significant, while a high p-value suggests that the interaction may be due to random noise in the data.

Cytoscape

Input

Network File from ARACNe: The input for Cytoscape is a network file generated byARACNe, which contains the details of the gene regulatory network.

Here's what the network file contains:

1. Nodes (Genes/Transcription Factors): Each node in the network represents a gene or a transcription factor (TF). The nodes are typically connected to one another based on the inferred regulatory relationships from ARACNe.

2. Edges (Regulatory Relationships):

- The edges represent the regulatory interactions between genes, with one gene being the regulator and the other being the target.
- The edges are accompanied by key data such as:
    - Mutual Information (MI): A measure of the strength of the co-expression relationship between the genes.
    - p-value: The statistical significance of the interaction, derived from the bootstrapping process in ARACNe.

Output

Once the network file from ARACNe is loaded into Cytoscape, the output consists of several layers of information, including a visual network graph and various network analysis results.

Visual Representation of the Network:

- Cytoscape generates a graphical representation of the gene regulatory network.
    - Nodes are displayed as circles or other shapes, and their size can be customized based on attributes such as gene expression levels, degree, or centrality.
    - Edges are drawn as lines between nodes to represent regulatory interactions, with the thickness or color of the edges often reflecting the strength of the relationship, such as the MI score or p-value.
- The network can be customized visually by adjusting the layout, node sizes, and colors for better clarity and interpretation.

Network Summary and Analysis Tools: Cytoscape provides a suite of network analysis tools to evaluate the structural and functional properties of the network. After loading the network into Cytoscape, users can analyze the network through tools such as Analyze Network or NetworkAnalyzer.

Key analysis features in Cytoscape include:

- **Degree Distribution**: The degree of a node indicates the number of interactions (edges) a gene has in the network. Cytoscape calculates the degree distribution, highlighting highly connected genes (hubs) and isolated nodes.
- **Centrality Measures**: These quantify a node's influence in the network:
    - **Betweenness Centrality**: Measures how often a node acts as a bridge between other nodes, indicating its central role.
    - **Closeness Centrality**: Reflects how quickly a node can influence others by being close to all other nodes.
    - **Degree Centrality**: Based on the number of direct connections a node has.
- **Clustering Coefficient**: Measures how connected a node's neighbors are. High clustering indicates the node is part of a dense, potentially functionally significant sub-network.
- **Path Lengths**: The average shortest path length indicates how closely genes are connected, with shorter paths suggesting tighter functional relationships.

VIPER

Input

The VIPER (Virtual Inference of Protein-activity by Enriched Regulon analysis) analysis utilized the following inputs:

1. Gene Expression Data:

- File: `modified_file.txt`
- Description: Log10-transformed TPM (Transcripts Per Million) values derived from the GSE153873 dataset, consisting of 27,136 genes across 30 samples. The data was standardized using Z-score transformation to normalize gene expression values.

2. Regulatory Network (Regulon):

- File: `network.txt`
- Description: Output from ARACNe-AP containing inferred gene regulatory interactions. The file includes as mentioned above:
    1. Regulator: The transcription factor or regulatory protein.
    2. Target: Genes regulated by the transcription factor.
    3. Mutual Information (MI): A measure of the strength of the regulatory interaction.
    4.P-value: Significance of the interaction, adjusted during the bootstrapping step

3. Additional Parameters for VIPER:

- Regulon format: A nested list created from the ARACNe output, associating regulators with their targets and interaction strengths.
- Method: Rank-based enrichment to infer protein activity scores.
- Minimum target size per regulator: 4.

Output

The VIPER (Virtual Inference of Protein-activity by Enriched Regulon analysis) algorithm transforms gene expression data into a protein activity landscape, providing insights into the functional states of regulatory proteins. VIPER builds on the regulatory networks inferred by ARACNe and calculates the activity of transcription factors (TFs) and other regulatory

proteins. Below is a detailed description of the specific elements included in the VIPER output:

1. Regulatory Protein Activity Scores:

- VIPER outputs a matrix of activity scores for regulatory proteins, including transcription factors. These scores reflect the inferred functional activity of each protein in the biological context under study.
- A positive activity score indicates activation of the regulator, while a negative score suggests repression.
- Scores are computed using the expression levels of target genes and the weighted relationships (mutual information and p-values) from the input regulatory network.

Key Points:

- Activity scores provide a more biologically relevant understanding of regulatory influence compared to raw gene expression data.
- Regulators with highly positive or negative scores are of particular interest, as they may serve as key drivers of the observed phenotypes.

2. Hierarchical Clustering of Samples and Regulators:

- The activity scores are used to cluster samples (columns) and regulators (rows) based on similarity patterns.
- The clustering reveals regulatory modules, with regulators that display similar activity patterns grouped together.
- Samples are grouped into distinct clusters corresponding to their biological conditions (e.g., Young, Old, and AD).

Key Points:

- Clustering helps identify condition-specific regulatory patterns, such as regulators uniquely active in AD.
- Regulators grouped in clusters may represent co-regulated pathways or biological processes.

3. Top Regulators by Variance:

- VIPER ranks regulators by the variability of their activity scores across samples.

- Regulators with the highest variance are highlighted as they exhibit the most distinct activity changes between conditions, making them potential master regulators.

4. Potential Biological Insights:

- VIPER highlights regulatory proteins likely involved in AD-specific processes, such as synaptic dysfunction or inflammatory signaling.
- The results suggest that regulators with high activity in the AD samples may drive disease-specific transcriptional programs.

Key Points:

- These insights are hypothesis-generating, providing a starting point for experimental validation.
- Regulators identified in the output could serve as biomarkers or therapeutic targets.

The VIPER output provides a comprehensive view of regulatory protein activity, enabling the identification of key drivers of biological processes in Alzheimer's Disease. This data lays the groundwork for further refinement and validation in subsequent computational or experimental studies.

Timeline

Task	Time	Milestone to be achieved
Conduct a literature review. Go through the work done by Califano and understand his approach to finding bottleneck genes.	Day 1	- A clear understanding of the project and key outputs that need to be achieved. - A clear understanding of the tools and methods to be used. - A clear understanding of the datasets (input/output used for the tasks and from where to get it).
Select the data source for Alzheimer's in the required format	Day 2	Downloaded file with required gene expression matrix (RNA seq data)

Transform and normalize the data for downstream uses		Transformed and normalized data file
Perform ARACNe		Gene regulatory network (GRN) file
Perform Cytoscape analysis		Visualization and network analysis summary with interpretation and analysis of the data.
Perform VIPER analysis	Day 3	Protein activity level (heatmap)
Prepare presentation and finalize documentation		Presentation and documentation files

TASK 3: Execute the project plan

Gene expression data from GEO:

GSE153873_summary_count.star.txt (downloaded file from GEO)

refGene	20-1T-AD		13-11T-Old		15-13T-Old		16-14T-Old		3-17T-Young		5-18T-Young		7-19T-Young		21-1A-AD		23-2A-AD		26-3A-AD		27-5A-AD		
	22-2T-AD		12-6A-Old		14-7A-Old		10-8A-Old		17-9A-Old		18-10A-Old		19-11A-Old		2-12A-Young		4-13A-Young		6-14A-Young		8-15A-Young		
	24-3T-AD		9-16A-Young		25-5T-AD		29-6T-AD		31-7T-AD		28-8T-AD		30-9T-AD		11-10T-Old								
SGIP1	1405	1405	1169	2408	859	1164	1402	1441	1003	1265	987	1325	2238	1075	2756	2413	1748	1322	2602	3115	2953	1541	795
	1614	833	1284	919	1004	1593	1174																
NECAP2	295	460	334	347	617	585	372	343	374	315	355	491	356	344	217	470	403	338	409	196	220	424	426
	387	552	328	353	474	344	350																
AZIN2	356	306	385	507	787	751	577	238	209	241	211	440	421	296	427	394	360	264	611	323	396	394	179
	424	567	292	301	319	348	575																
AGBL4	191	200	173	323	36	89	184	229	129	154	102	124	268	200	327	282	202	213	328	335	431	38	46
	231	45	123	155	108	247	156																
CLIC4	876	1443	639	792	4806	5968	1392	1117	1247	929	1151	3674	1526	1269	719	4716	1361	819	2493	1332	729	8812	936
	689	2534	856	646	1406	1255	665																
SLC45A1	291	329	298	636	139	204	278	193	152	178	142	224	253	195	474	349	269	231	551	441	506	221	156
	409	138	222	251	178	272	350																
TGFBR3	639	650	506	370	425	447	282	792	727	461	728	499	464	576	452	515	453	597	522	347	383	172	348
	571	264	292	588	498	547	460																
DBT	623	726	513	633	751	758	498	714	756	655	530	517	892	472	698	830	700	596	1050	982	759	1028	397
	455	549	514	480	463	707	412																
PRUNE1	297	237	200	390	231	254	252	342	228	300	235	248	306	196	316	347	253	188	380	411	325	305	289
	217	227	226	297	258	299	266																
C1orf21	2456	2006	2092	3146	848	1147	1551	2325	2439	2471	2344	1468	2543	1846	3213	2422	2353	1813	2965	3800	3339	525	1932
	1940	1225	2088	2030	1905	2214	1829																
RFWD2	537	614	472	612	581	586	534	591	482	533	394	491	672	366	699	712	621	433	785	758	664	671	415
	554	527	479	483	563	516	515																
LIN9	54	33	29	37	22	34	30	76	49	48	42	45	48	33	47	53	34	36	47	61	57	79	39
	57	32	34	45	37	38	45																
PRKCZ	1786	1062	1229	2280	1123	1239	1613	1100	859	1404	1038	1249	1423	895	1689	1586	1228	877	2519	2484	1785	1077	1367
	1536	1041	1555	1551	1094	1367	1758																
PRDM16	517	958	849	482	359	353	398	638	849	706	1079	410	432	817	457	554	606	1018	676	593	409	117	541
	666	281	676	963	624	712	578																
ICMT	759	625	529	609	647	535	466	673	715	764	631	439	658	524	682	793	538	488	971	927	680	888	582
	582	548	483	745	695	592	729																
CAMTA1	2549	2063	2246	3669	1031	1511	2065	2471	1789	2323	1647	1508	3073	2036	3940	3404	2305	2124	4562	4625	4160	1097	1034
	2826	967	1724	2482	1400	2926	3006																
C1orf159		160	281	276	291	231	273	185	175	137	185	161	267	114	240	171	201	209	195	207	204	159	217
	122	377	208	252	205	213	212	325															
PRAMEF18		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0															
PRAMEF10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This is what the downloaded file looked like for the gene expression data (RNA-seq) from Alzheimer's disease (AD) from GEO. (GSE153873_summary_count.star.txt)

Chosen dataset: GSE153873

Title: An integrated multi-omics approach identifies epigenetic alterations associated with Alzheimer's disease (RNA-seq with ERCC spike-in) [RNA-Seq]

This study investigates **epigenetic and gene expression changes** associated with Alzheimer's Disease (AD) using an RNA sequencing (RNA-seq) approach. The researchers performed **RNA-seq with ERCC spike-ins** on samples from the hippocampus, a brain region crucial for memory and often impacted in Alzheimer's.

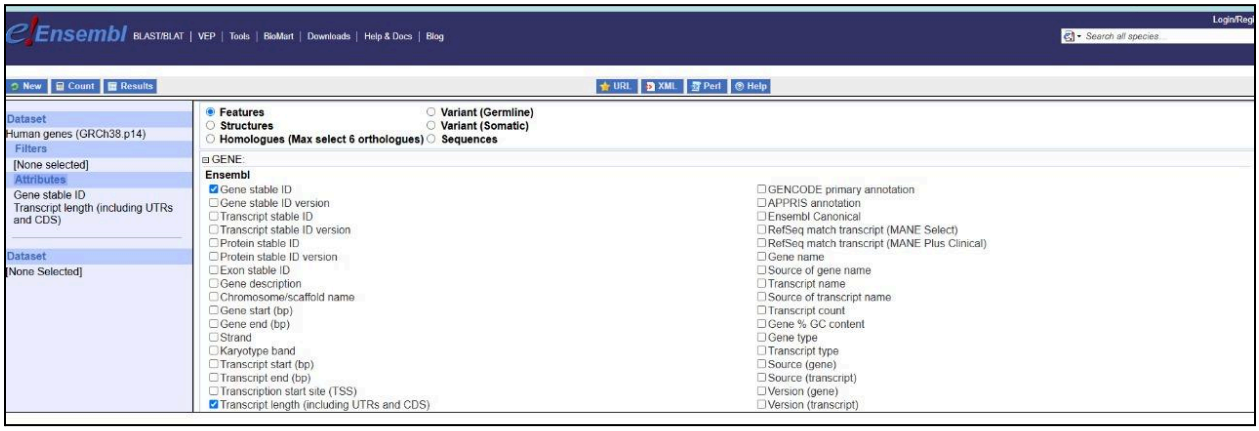
RNA was collected from the lateral temporal lobe of postmortem brains, and classified into three groups:

- AD group (patients with Alzheimer's disease).
- Control Old group (elderly individuals without AD).
- Control Young group (younger individuals without AD).

27136 genes | 30 samples

Log10 TPM transformation using R:

To transform the data to log10 TPM we needed the length of the genes in the dataset. This was taken from the ENSEMBL Biomart database → using the Biomart data mining tool.



Selected parameters:

To get the gene length of transcription factors from Ensembl BioMart:

The gene length of transcription factors was obtained from Ensembl BioMart following a systematic process. First, the Ensembl BioMart page was accessed by visiting the official website. The **Ensembl Genes** database was selected from the "Select Database" dropdown, and the appropriate species dataset, such as *Homo sapiens genes* for humans, was chosen from the "Select Dataset" dropdown.

Next, the **Attributes** section was navigated to, and under the **"Gene"** tab, the **"Trancript length (including UTRs and CDS)"** attribute was selected. This was typically located under the **"Gene Information"** category. Additionally, the **"Gene stable ID"** was selected to allow for proper gene identification. As I was not able to find a specific dataset for just the transcription factor, I downloaded every gene in the Gene stable ID data.

After setting the attributes and filters, the query was executed by clicking the **"Results"** button. The results table was generated, providing the gene length alongside the other selected attributes. Finally, the results were exported in the desired format, TSV, by using the **"Export"** option

Output: Gene stable IDs (ENSEMBL IDs) and the transcript lengths.

Gene stable ID	Transcript length (including UTRs and CDS)
ENSG00000210049	71
ENSG00000211459	954
ENSG00000210077	69
ENSG00000210082	1559
ENSG00000209082	75
ENSG00000198888	956
ENSG00000210100	69
ENSG00000210107	72
ENSG00000210112	68
ENSG00000198763	1042
ENSG00000210117	68
ENSG00000210127	69
ENSG00000210135	73
ENSG00000210140	66
ENSG00000210144	66
ENSG00000198804	1542
ENSG00000210151	69
ENSG00000210154	68
ENSG00000198712	684
ENSG00000210156	70
ENSG00000228253	207
ENSG00000198899	681
ENSG00000198938	784
ENSG00000210164	68
ENSG00000198840	346
ENSG00000210174	65
ENSG00000212907	297
ENSG00000198886	1378
ENSG00000210176	69
ENSG00000210184	59

This is a glimpse of what the downloaded file looked like for the gene length from ENSEMBL Biomart.

```
# Load required libraries
library(org.Hs.eg.db)
library(dplyr)

# Step 1: Load the data
```

```r
count_matrix <-
read.table("/Users/diyahafiz/Documents/Systems_Biology/Final_Project/GSE153873_summary
_count.star.txt",
                           header = TRUE,
                           row.names = 1,
                           sep = "\t")

gene_info <-
read.table("/Users/diyahafiz/Documents/Systems_Biology/Final_Project/mart_export.txt",
                        header = TRUE,
                        sep = "\t")

# Step 2: Clean up gene_info
names(gene_info) <- c("ENSEMBL", "Length")
gene_info <- gene_info[, c("ENSEMBL", "Length")]  # Keep only needed columns

# Step 3: Map gene symbols to Ensembl IDs
gene_symbols <- rownames(count_matrix)
symbol_to_ensembl <- mapIds(
 org.Hs.eg.db,
 keys = gene_symbols,
 column = "ENSEMBL",
 keytype = "SYMBOL",
 multiVals = "first"
)

# Step 4: Create clean mapping dataframe
mapping_df <- data.frame(
 SYMBOL = names(symbol_to_ensembl),
 ENSEMBL = symbol_to_ensembl,
 stringsAsFactors = FALSE
)

# Step 5: Merge data
# Add symbols to count matrix
count_matrix$SYMBOL <- rownames(count_matrix)

# Merge counts with mapping
merged_counts <- merge(mapping_df, count_matrix, by = "SYMBOL")

# Merge with gene lengths
final_data <- merge(merged_counts, gene_info, by = "ENSEMBL")
```

```r
write.table(final_data,"Merged_IDS_starcount.csv")



# Step 6: Calculate TPM
# Get count columns
count_cols <- setdiff(colnames(final_data), c("SYMBOL", "ENSEMBL", "Length"))

# Calculate RPK (reads per kilobase)
rpk_matrix <- as.matrix(final_data[, count_cols]) / (final_data$Length / 1000)

# Calculate scaling factors
scaling_factors <- colSums(rpk_matrix) / 1e6

# Calculate TPM
tpm_matrix <- sweep(rpk_matrix, 2, scaling_factors, FUN = "/")
rownames(tpm_matrix) <- final_data$SYMBOL

# Step 7: Verify results
print("Data dimensions:")
print(paste("Original count matrix:", nrow(count_matrix), "genes"))
print(paste("After mapping and cleaning:", nrow(tpm_matrix), "genes"))
print(paste("Number of unique genes:", length(unique(rownames(tpm_matrix)))))
print(paste("Number of duplicated genes:", sum(duplicated(rownames(tpm_matrix)))))

# Step 8: Save results
# Save TPM matrix
write.table(tpm_matrix,
            "tpm_matrix_unique.txt",
            sep = "\t",
            quote = FALSE,
            row.names = TRUE)

# Create and save log-transformed version
log10_matrix <- log10(tpm_matrix + 1)
write.table(log10_matrix,
            "log10_tpm_matrix_unique.txt",
            sep = "\t",
            quote = FALSE,
            row.names = TRUE)

# Keep only the first occurrence of each SYMBOL
unique_data <- final_data[!duplicated(final_data$SYMBOL), ]
```

```
# Print dimensions to verify
print("Original dimensions:")
print(dim(final_data))
print("Dimensions after keeping unique symbols:")
print(dim(unique_data))
# Get count columns (exclude SYMBOL, ENSEMBL, Length)
count_cols <- setdiff(colnames(unique_data), c("SYMBOL", "ENSEMBL", "Length")
# Create log-transformed version
log10_matrix <- log10(tpm_matrix + 1)
write.table(log10_matrix,
            "final_log10_tpm_matrix.txt",
            sep = "\t",
            quote = FALSE,
            row.names = TRUE)
```

To add gene column:

```
# Step 1: Load the data from the text file
# Replace "your_file.txt" with the path to your file
data <- read.table("C:/Users/Gunja Gupta/Downloads/final_log10_tpm_matrix.txt", header = TRUE, row.names = 1)

# Step 2: Convert row names into a column
data$RowNames <- rownames(data)

# Step 3: (Optional) Reorder columns to place "RowNames" first
data <- data[, c("RowNames", setdiff(names(data), "RowNames"))]

# Step 4: Save the modified data frame to a new text file
write.table(data, "modified_file.txt", sep = "\t", row.names = FALSE, quote = FALSE)

# View the updated data frame
print(data)
```

```
# Step 1: Read the text file
data <- read.table("modified_file.txt", header = TRUE)

# Step 2: Change the name of the first column
colnames(data)[1] <- "Gene"  # Replace "NewColumnName" with your
desired column name

# Step 3: Save the modified data frame back to a text file
(optional)
write.table(data, "modified_file.txt", sep = "\t", quote =
FALSE)

# View the updated data frame# View the updated data frameTRUE
print(data)
```

This code performs several crucial preprocessing steps:

1. Loads necessary tools for gene identifier conversion and data handling
2. Imports raw count data and gene annotation information
3. Standardizes gene information format
4. Creates a mapping between different gene identifier systems
5. Combines all information into a single comprehensive dataset

This preparation is essential for:

- Ensuring consistent gene identification across datasets
- Enabling proper length normalization for TPM calculation
- Creating a clean, well-annotated dataset for downstream analysis

Log10 TPM transformed file which was used as an input for ARACNe-AP:

Gene	X20.1T.AD	X13.11T.Old	X15.13T.Old	X16.14T.Old
TSPAN6	0.91782015	1.08256751	0.99426398	1.032410018
TNMD	0.05598940	0.05685462	0.06229114	0.046118021
DPM1	0.71431662	0.73099019	0.60475658	0.705279279
SCYL3	0.69532708	0.77134500	0.77058882	0.754118722
FGR	1.05358133	1.35869317	1.11708758	0.894424441
CFH	0.93246482	1.32553465	0.98727976	0.728927468
FUCA2	1.33124305	1.39914509	1.28722202	1.210649798
GCLC	1.27089174	1.30052966	1.21416275	1.265497789
NFYA	0.88185778	0.95704397	0.89127437	0.812005772
STPG1	0.74381703	0.73321692	0.80985846	0.809027383
NIPAL3	2.35415908	2.25324341	2.25842332	2.384692981
LAS1L	1.22248126	1.36086989	1.38430829	1.150982303
ENPP4	0.96027839	0.96507382	0.79753976	0.854824123
SEMA3F	1.53238746	1.78431673	1.44784179	1.563263383

First few rows and columns of the log10 TPM transformed file taken from R, which was then used for ARACNe-AP. (final_log10_tpm_matrix.txt)

ARACNe-AP:

First, we used the githup codes from Califano lab to build the ARACNe-AP software on Powershell.

Building ARACNe-AP

`ARACNe-AP` requires JDK > 1.8 and ANT. Use the following command in the repository root directory to build the `jar` and documentation:

```
ant main
```

The jar will be placed in `dist/aracne.jar`. The documentation can be found in `docs/index.html`.

The command `ant main` was used to build the ARACNe-AP software. It was required that JDK version 1.8 or higher and ANT were installed. The build process was initiated by executing the command, which resulted in the compilation of the code and the generation of

output artifacts. The compiled JAR file was placed in the `dist/aracne.jar` directory, and the documentation was generated in the `docs/index.html` file.

Here's a breakdown of the process:

1. **JDK > 1.8**: The software requires Java Development Kit (JDK) version 1.8 or higher to compile and run.
2. **ANT**: Apache Ant is a build automation tool used to compile and package software. It automates the process of building software projects, including compiling source code, generating JAR files (Java Archive), and creating documentation.
3. **`ant main`**: This command triggers the build process for the ARACNe-AP project. It compiles the code, runs necessary tasks, and creates the output artifacts. The output includes:
    - A **JAR file** (`dist/aracne.jar`), which is the compiled program you can run.
    - **Documentation** (`docs/index.html`), which provides details on how to use ARACNe-AP and other relevant information.

So, by running this command in PowerShell, we built the ARACNe-AP tool to generate a JAR file for running the tool, along with documentation for understanding and using it.

Running ARACNe-AP for our dataset:

```
PS C:\Users\Gunja
Gupta\Downloads\ARACNe-AP-master\ARACNe-AP-master> java -Xmx5G
-jar aracne.jar -e "C:\Users\Gunja
Gupta\OneDrive\Documents\modified_file.txt" -o project --tfs
"C:\Users\Gunja Gupta\Downloads\trrust_rawdata.human.txt"
--pvalue 1E-8 --seed 1 --calculateThreshold
>>
MI threshold file was already there, but I am recalculating it.
Finding threshold for 30 samples
Parameters for fitted threshold function: [0.5725240073980846,
6.22541231158068E-6]
MI threshold: 0.6570874234418606
```

```
PS C:\Users\Gunja
Gupta\Downloads\ARACNe-AP-master\ARACNe-AP-master> java -Xmx5G
-jar aracne.jar -e "C:\Users\Gunja
Gupta\OneDrive\Documents\modified_file.txt" -o project --tfs
"C:\Users\Gunja Gupta\Downloads\trrust_rawdata.human.txt"
--pvalue 1E-8 --seed 1
>>
Bootstrapping input matrix 1 with 23125 genes and 30 samples
MI threshold file is present
Calculate network from: C:\Users\Gunja
Gupta\OneDrive\Documents\modified_file.txt
TFs processed: 2808
Time elapsed for calculating MI: 428 sec

DPI time elapsed: 1 sec
Edges removed by DPI:   201342
Final Network size:     98777
Total time elapsed: 430 sec
PS C:\Users\Gunja
Gupta\Downloads\ARACNe-AP-master\ARACNe-AP-master> for ($i=1; $i
-le 100; $i++) {
>>     java -Xmx5G -jar aracne.jar -e "C:\Users\Gunja
Gupta\OneDrive\Documents\modified_file.txt" -o project --tfs
"C:\Users\Gunja Gupta\Downloads\trrust_rawdata.human.txt"
--pvalue 1E-8 --seed $i
>> }
>>
```

Step 1: calculate threshold with a fixed seed

The code was used to run the ARACNe-AP (ARACNe-Algorithm for the Reconstruction of Accurate Cellular Networks) tool for the analysis of gene expression data and the generation of gene regulatory networks. In the first step, the ARACNe algorithm was initialized with a specified memory allocation of 5GB (`-Xmx5G`), and the input files for gene expression data (`modified_file.txt`) and a list of transcription factors (`trrust_rawdata.human.txt`) were provided. The `-e` flag was used to specify the

expression data file, while the `--tfs` flag was used to indicate the transcription factors file. The `--pvalue` flag was set to a significance threshold of 1E-8 for the mutual information (MI) calculations, and the `--seed` flag was used to initialize randomization with a seed value of 1. The `--calculateThreshold` flag was included to recalculate the MI threshold based on the data, ensuring accurate network construction. After execution, the MI threshold was recalculated, and the parameters for the threshold function, as well as the resulting threshold value, were provided for further use in network construction.

Step 2: calculate MI an apply DPI

In the second step, the ARACNe tool was run again with the same input parameters, but without the `--calculateThreshold` flag. The gene expression data matrix, consisting of 23,125 genes and 30 samples, was processed to calculate the mutual information (MI) between all pairs of genes. The network construction process involved bootstrapping, where the data was resampled to assess stability, and the Data Processing Inequality (DPI) was applied to refine the gene-gene interactions by removing statistically insignificant edges. The results from this step provided the number of transcription factors processed, the time taken for the MI calculation, the time spent on DPI, and the number of edges removed during DPI. The final network size was reported, representing the refined gene regulatory network after the MI and DPI processes.

Step 4: run 100 reproducible bootstraps

In the third block of code, a loop was used to run the ARACNe tool 100 times, each time with a different seed value for randomization. The loop iterated from seed 1 to 100, and each iteration processed the data in a similar manner to the previous steps, recalculating the mutual information and generating a new network.

Step 5: consolidate bootstraps in the output folder

```
PS C:\Users\Gunja
Gupta\Downloads\ARACNe-AP-master\ARACNe-AP-master> java -Xmx5G
-jar aracne.jar -o project --consolidate
Integrating 100 bootstraps...
```

The last step of the code is used for combining the results from 100 separate bootstrap runs to form a final network representation.

Parameters:

- p-value 1E-8
- Seed 1
- Parameters for fitted threshold function: [0.5725240073980846, 6.22541231158068E-6]
- MI threshold calculated : 0.6570874234418606
- Bootsrapping: 100 times (to get a more robust output, could have been increased for better results but we didn't have that much time to run for longer).

ARACNe-AP Output:

Regulator	Target	MI	pvalue
MAOA	DDAH1	0.7968290729333158	0.006762486313700911
BARD1	LDHAL6B	0.8765042332114081	0.006762486313700911
SFRP5	SRC	0.9620350075614169	0.0
SRD5A3	ATN1	0.8026277016007901	0.0
STAT4	ELAVL4	1.0367059611494118	0.0
TP73	PPCDC	0.8919209104693847	0.0
MCAT	PTPRN	0.8631705848099901	0.0
SMAD7	SLC5A2	0.7585079507124848	0.006762486313700911
RRM2B	EIF2AK1	0.9268946377403312	0.0
CFLAR	HMGCR	0.8121321811202855	3.626351880114953E-6
SUGP1	LONP1	0.750412605020343	0.006762486313700911
TRAF1	SESN2	0.8287551604622748	1.6501900880927511E-4
E4F1	SELENOI	0.7255743712592357	1.6501900880927511E-4
E4F1	SELENOO	0.7999177346024948	0.0
PPARGC1A	EGOT	0.8595961060616645	0.0
RTN3	CISD1	0.9556492713820098	0.0
ETS2	GPR22	0.8368810160067751	3.626351880114953E-6
VMP1	FLYWCH1	0.7608020045916077	1.6501900880927511E-4
MXD4	ZFPM1	0.8024455796861841	1.6501900880927511E-4
MAT2B	IQCD	0.7433615555826419	0.006762486313700911
GYPA	SMIM17	0.8080332765895533	0.006762486313700911
FTO	RRM1	0.7241543037137304	3.626351880114953E-6
MTF1	ZMAT2	0.8789824066029543	3.626351880114953E-6
IRF5	PHOSPHO1	0.7934375834077982	3.626351880114953E-6
CDK6	PHLPP1	0.9302311192679114	0.006762486313700911
CDO1	RUNDC3B	0.9058613251160551	7.332696816408912E-8
PCSK1	NRN1	0.8786836443512964	0.0
ECE1	MAP3K3	0.8862956456428245	0.0
TCF7	LOC100288208	0.8531585768038563	3.626351880114953E-6
LMO7	GNG3	0.9949966111359	0.0
TSHR	FAM174A	0.7205369835748601	1.6501900880927511E-4
TFAP2C	CLDN10	0.8465328324894611	3.626351880114953E-6
ECM1	BMPER	0.835001003934865	0.0
TAF7	ZDHHC17	0.7977753428790743	1.6501900880927511E-4
DRAP1	JADE3	0.737154329362456	0.0
TJP1	METTL26	0.9403460871151258	0.0
CFTR	ANO4	0.9618347328806516	0.0
LGR5	ANLN	0.9421270496732843	0.0

The ARACNe-AP output Txt file of showing the regulators, target, MI value and P value for each.

(https://drive.google.com/file/d/1WTA0sj9ZeGXBF6ATVQb71_jXrj6jbCy6/view?usp=drive_link)

Cytoscape:

Regulator	Target	MI	pvalue
MAOA	DDAH1	0.7968290729333158	0.006762486313700911
BARD1	LDHAL6B	0.8765042332114081	0.006762486313700911
SFRP5	SRC	0.9620350075614169	0.0
SRD5A3	ATN1	0.8026277016007901	0.0
STAT4	ELAVL4	1.0367059611494118	0.0
TP73	PPCDC	0.8919209104693847	0.0
MCAT	PTPRN	0.8631705848099901	0.0
SMAD7	SLC5A2	0.7585079507124848	0.006762486313700911
RRM2B	EIF2AK1	0.9268946377403312	0.0

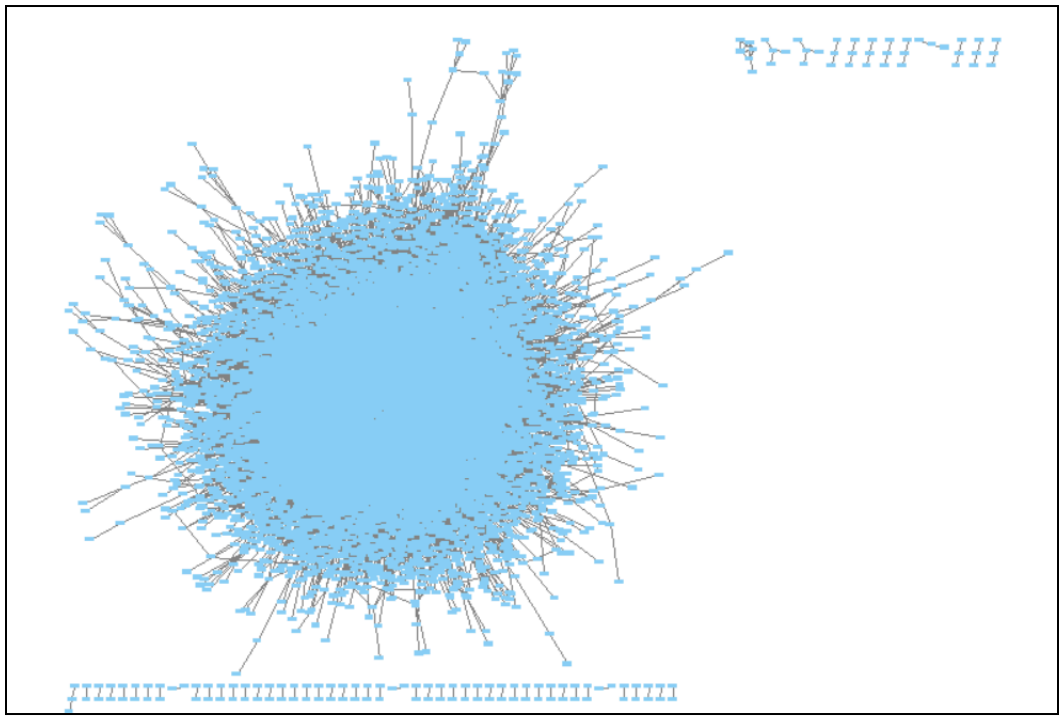
The input file in cytoscape

The output network file from ARACNe was then used to visualise through Cytoscape.

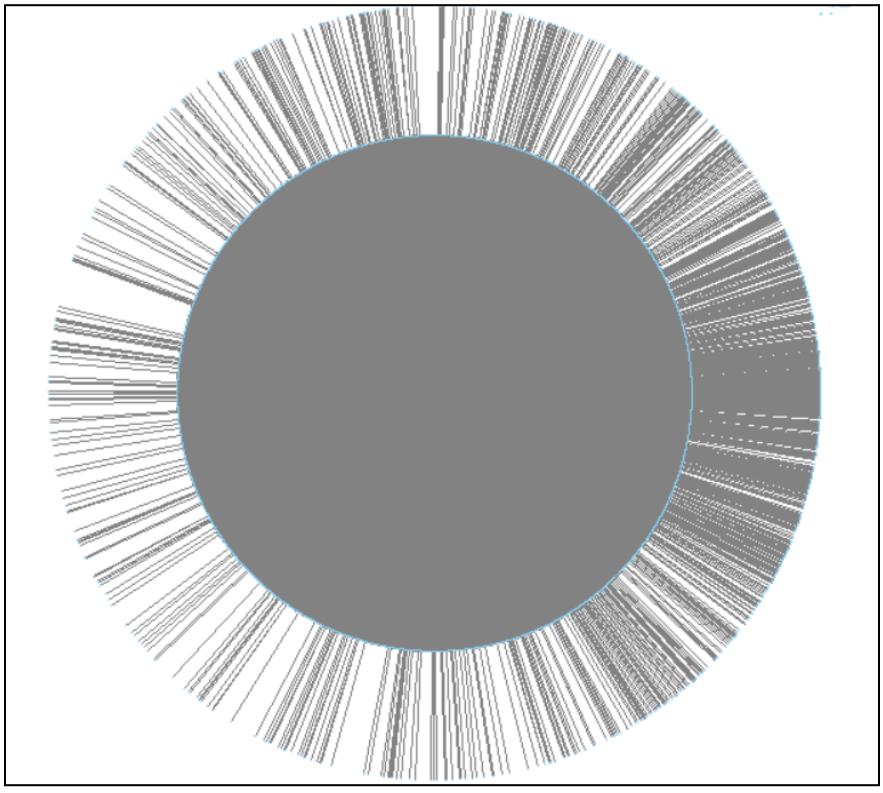
- Regulator - Source
- Target - Target
- MI - Edge attribute
- P-value - Edge attribute

The MI and p-values were set as edge attributes hoping that this information could be visualised by the thickness difference in the edge of different nodes - indicating how strong the network is (weighted network).

Output from Cytoscape:



Default network



Circular layout

Analyzing network:

Summary Statistics
Number of nodes: 15003
Number of edges: 197554
Avg. number of neighbors: 24.895
Network diameter: 12
Network radius: 7
Characteristic path length: 3.887
Clustering coefficient: 0.165
Network density: 0.002
Network heterogeneity: 1.904
Network centralization: 0.032
Connected components: 60
Analysis time (sec): 160.444

Summary of data

AverageShortestPath	BetweennessCentrali	ClosenessCentrality	ClusteringCoefficien	Degree	Eccentricity	IsSingleNode	name	NeighborhoodConne
3.472285753	6.42E-04	0.2879947306	0.06346153846	72	8	FALSE	MAOA	30.04615385
3.51479887	2.01E-04	0.2845113012	0.1333333333	21	7	FALSE	DDAH1	111.5714286
3.503228844	2.44E-04	0.2854509495	0.06253585772	92	7	FALSE	BARD1	48.5952381
3.485739271	1.48E-04	0.2868831896	0.1086956522	24	7	FALSE	LDHAL6B	157.5
3.403874613	0.001051214662	0.2937828544	0.05806858826	109	8	FALSE	SFRP5	47.03061224
3.242297861	0.001489434955	0.3084232365	0.06583333333	252	8	FALSE	SRC	54.60444444
3.225144625	0.001590943669	0.3100636146	0.03785672685	114	8	FALSE	SRD5A3	36
3.397282389	1.32E-04	0.2943529225	0.2085365854	41	8	FALSE	ATN1	206.195122
3.160365936	0.002683918132	0.3164190541	0.04339016555	311	7	FALSE	STAT4	49.3754386
3.374949549	1.06E-04	0.2963007255	0.2551282051	40	7	FALSE	ELAVL4	222.75
3.294362976	0.001881816543	0.3035488218	0.02626358054	157	7	FALSE	TP73	31.45205479
3.569958294	4.54E-05	0.2801153172	0.08974358974	13	7	FALSE	PPCDC	174.6923077
3.356249159	0.002925613687	0.2979516575	0.0347985348	209	7	FALSE	MCAT	36.33673469
3.452643616	1.13E-04	0.2896331366	0.2672268908	35	7	FALSE	PTPRN	223.5714286
3.745863043	6.65E-05	0.2669611752	0.1538461538	18	8	FALSE	SMAD7	53.07692308
3.733082201	2.72E-05	0.2678751622	0.1388888889	9	8	FALSE	SLC5A2	116.1111111
3.235503834	0.005526911384	0.3090708747	0.03843060148	364	8	FALSE	RRM2B	41.66567164
3.434077761	0.001168936696	0.2911989971	0.05350988572	160	8	FALSE	EIF2AK1	40.01342282
3.215727163	0.001911521271	0.3109716557	0.0525280698	259	7	FALSE	CFLAR	54.07172996
3.199179335	0.001619186034	0.3125801636	0.05289607865	258	7	FALSE	HMGCR	58.07264957
3.432530607	0.002103322525	0.2913302501	0.04188083	117	8	FALSE	SUGP1	32.18446602
3.842190233	6.44E-06	0.2602682167	0.1666666667	4	8	FALSE	LONP1	174.5
3.248351944	0.002036963598	0.3078484158	0.05410733844	184	7	FALSE	TRAF1	51.85542169
3.437104803	9.80E-05	0.2909425395	0.1956521739	24	8	FALSE	SESN2	192.8333333
3.297793623	0.001577999814	0.3032330444	0.05345559186	223	8	FALSE	E4F1	54.99014778
3.438719225	1.27E-04	0.2908059468	0.2117647059	35	8	FALSE	SELENOI	219.5142857
3.605341047	9.26E-06	0.2773662705	0.3235294118	17	8	FALSE	SELENOO	233
3.377303915	0.002160639014	0.2960941701	0.03972618112	151	8	FALSE	PPARGC1A	32.03731343
3.766446926	3.58E-05	0.2655022146	0.1944444444	9	9	FALSE	EGOT	129.5555556

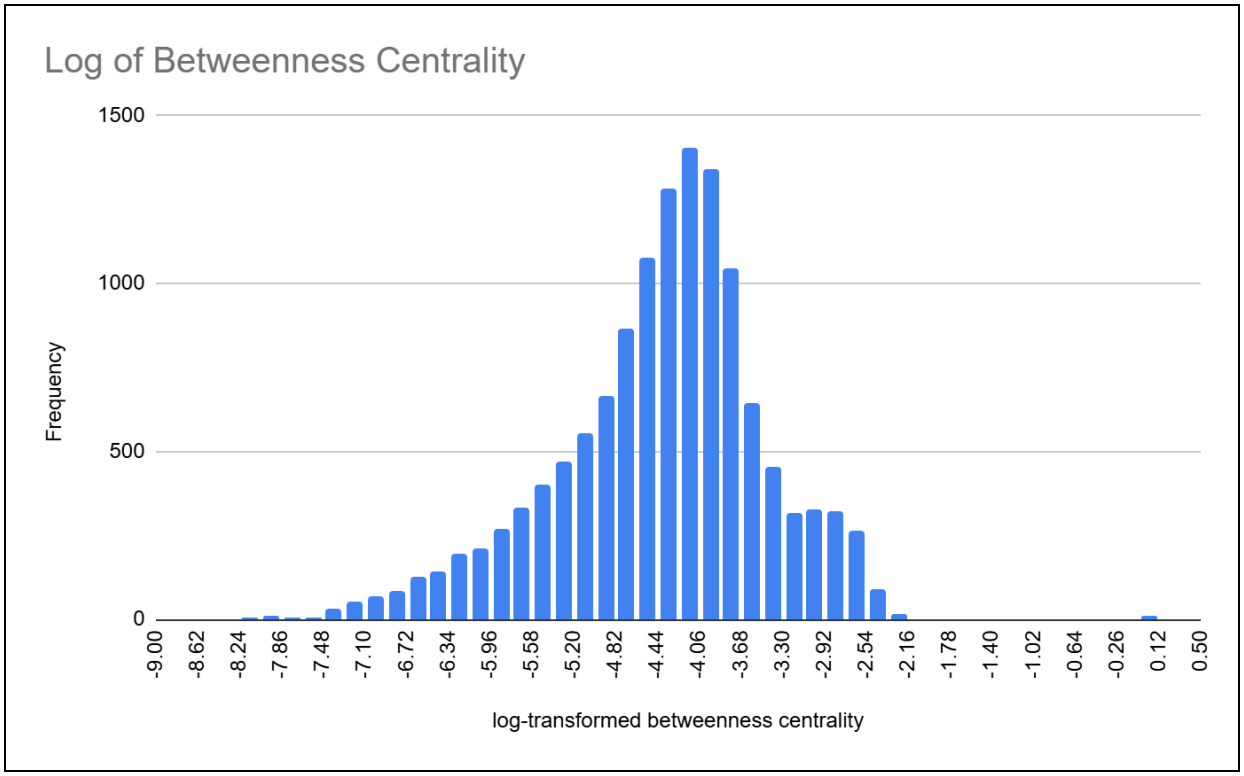
First few rows and columns of the newtork analysis on Cytoscape. (https://drive.google.com/file/d/1PHXj6smz5TAjJluV3EcDoVzeeDlLXKHA/view?usp=drive_link)

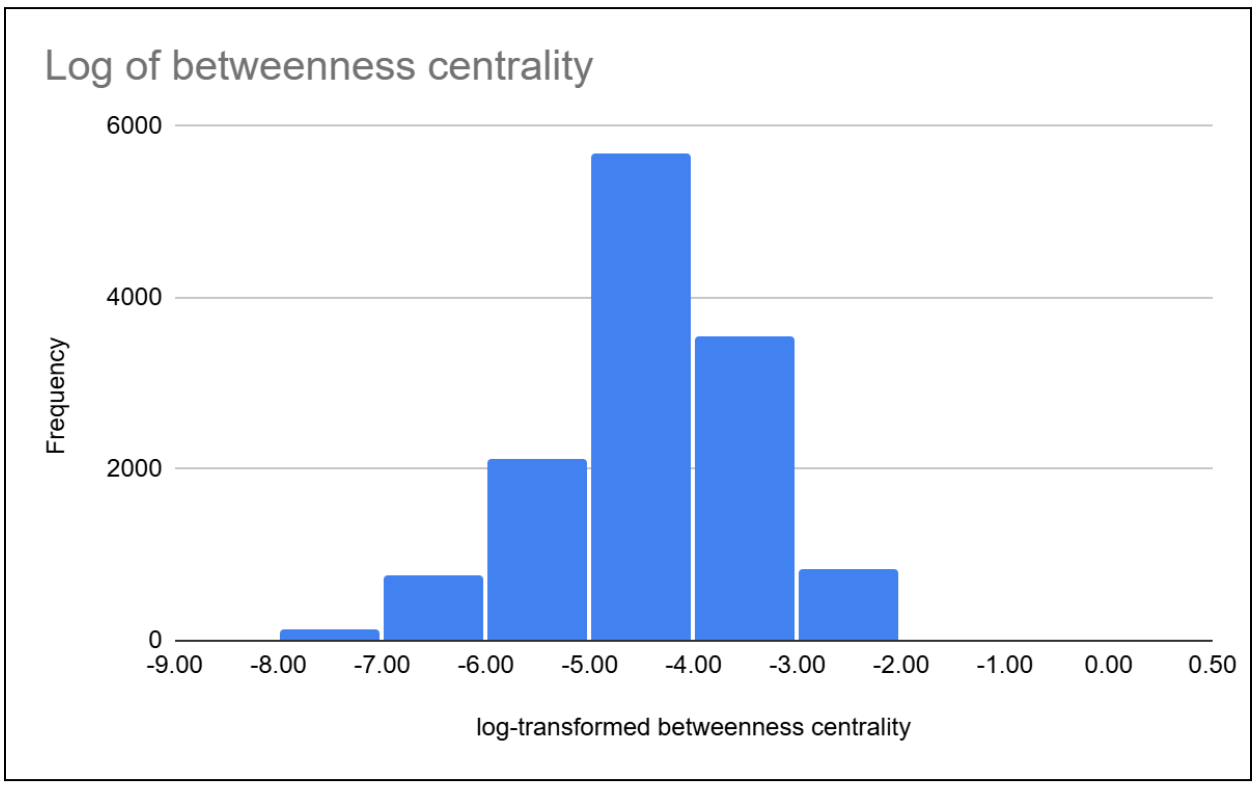
BetweennessCentrality	ClosenessCentrality	Degree	name
1	1	2	ARG1
1	1	2	UPF1
1	1	2	LCK
1	1	3	SUPT3H
1	1	3	MFN2
1	1	2	FCGR2B
1	1	2	PTGER2
1	1	3	DAP3
1	1	2	DES
1	1	2	KDR
1	1	2	DHRS2
1	1	2	RB1CC1
0.4	0.8333333333	7	HBD
0.06666666667	0.8333333333	8	HBB
0.06666666667	0.8333333333	8	HBA2
0.06666666667	0.8333333333	8	HBA1
0.009897100637	0.3409007522	475	GSK3B
0.006525183327	0.3295792134	279	NOTUM
0.006046013828	0.3236875912	199	GLI3

Analysed and interpreted the betweenness centrality of the network.

Bottlenecks are proteins that have a high betweenness centrality, meaning they are network nodes that have many "shortest paths" going through them. Bottlenecks are key connector proteins that are more likely to be essential proteins.

First I transformed the betweenness centrality values into log10 and then added in bucket of 1.





This histogram visualizes the distribution of the logarithm of betweenness centrality values across nodes in the network file from ARACNe-AP.

The histogram uses a logarithmic scale for betweenness centrality values, where values closer to zero (the right side of the graph) indicate nodes with high betweenness centrality. The values that are negative and go further to the left represent low betweenness centrality.

The peak of the distribution is around -4.5 to -3.5 on the x-axis, indicating that most nodes in the network have low betweenness centrality (compared to a betweenness of 1 in non log transformed graph). As the values move toward zero, the frequency of nodes significantly decreases, which indicates that only a few nodes in the network are highly central in terms of betweenness.

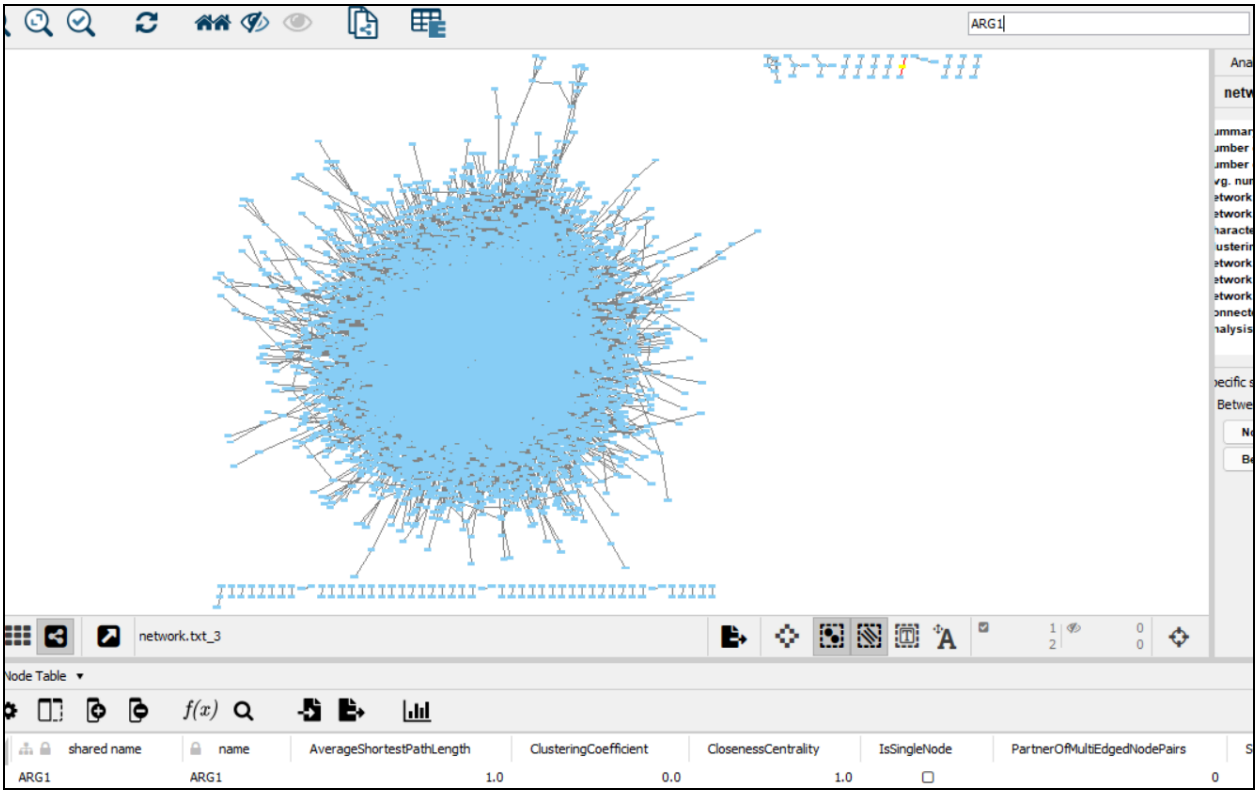
High betweenness centrality (right side/values closer to zero): These are critical nodes in the network that lie on many of the shortest paths between pairs of other nodes. They act as bridges and are often referred to as key connectors. Low betweenness centrality (left side/negative values): Most of the nodes have lower betweenness, indicating that they do not serve as bridges or critical connectors between other nodes in the network.

The network we analysed has a small number of highly influential nodes (high betweenness centrality), and most nodes are less central in terms of connecting different parts of the network. This kind of distribution is typical in many real-world networks, where a few nodes are highly important in terms of facilitating communication, while most nodes have a local or peripheral role. This distribution is consistent with many scale-free networks, where a few nodes (hubs) dominate the connectivity, and the majority of nodes are more isolated.

Key observations:

There were a set of genes which had a betweenness of 1, and could have been the ones that are highly connected in the network; however, these were the outlier.

For example, ARG1 (with highest betweenness centrality of 1) when searched in the network was found in the isolated nodes (highlighted by yellow-red in the network below).



Additionally, the next highest betweenness centrality which was in the central main network had many genes that fell in this value. So it is difficult to analyze each and find which could be bottlenecks.

VIPER

Introduction to Viper in Proteomic Analysis

In this project, we utilized the Viper package in R to perform a comprehensive analysis of a Gene Expression Omnibus (GEO) dataset, focusing on patients with Alzheimer's Disease (AD) compared to control samples. Viper, which stands for Virtual Inference of Protein-activity by Enriched Regulon analysis, is designed to infer protein activity from gene expression data by leveraging the regulatory relationships between genes and proteins.

Viper is available through Bioconductor, a repository for bioinformatics software in R, ensuring that researchers can easily access and integrate it into their analyses. The package

implements both the VIPER and msVIPER algorithms, allowing for single-sample and multiple-sample analyses of gene expression data. This flexibility makes Viper particularly useful for transforming gene expression matrices into matrices of protein activity, thereby providing insights into the functional state of proteins in various biological contexts.

In our analysis, we employed Viper to complement the output from the ARACNe algorithm, which was used to infer gene regulatory networks. The integration of Viper's capabilities allows us to gain a more nuanced understanding of proteomic changes associated with Alzheimer's Disease. Specifically, Viper applies functions that estimate the activity of regulatory proteins based on their target genes, taking into account factors such as interaction confidence and the pleiotropic nature of target gene regulation. This probabilistic approach enhances the robustness of our findings.

By utilizing Viper alongside ARACNe, we can effectively bridge the gap between transcriptomic data and protein activity, enabling us to identify key regulatory proteins that may drive disease processes in AD. This dual approach not only enriches our analysis but also provides a clearer picture of the underlying biological mechanisms at play.

Code for Viper Analysis

gene_matrix_modified <- read.table("/Users/diyahafiz/Downloads/modified_file.txt",

```
                        header = TRUE,
                        sep = "\t")
z_scores <- t(scale(t(gene_matrix_modified)))
str(gene_matrix_modified)
# Save gene names separately
gene_names <- gene_matrix_modified$Gene

# Save both gene names and column names
gene_names <- gene_matrix_modified$Gene
col_names <- colnames(gene_matrix_modified)

# Create numeric matrix while preserving names
expression_matrix <- as.matrix(gene_matrix_modified[,-1])  # Remove Gene column
rownames(expression_matrix) <- gene_names
colnames(expression_matrix) <- col_names[-1]  # All column names except 'Gene'
```

```r
# Z-score transformation across rows (genes)
z_scores <- t(scale(t(expression_matrix)))

# Verify names were preserved
head(rownames(z_scores))
head(colnames(z_scores))

# If you need it as a data frame for VIPER:
z_scores_df <- as.data.frame(z_scores)
z_scores_df <- cbind(Gene = gene_names, z_scores_df)
# Check summary statistics
summary(rowMeans(z_scores, na.rm=TRUE))  # Should be close to 0
summary(apply(z_scores, 1, sd, na.rm=TRUE))  # Should be close to 1

# Check structure
str(z_scores_df)  # For data frame version
str(z_scores)     # For matrix version

# 1. Load required libraries
library(viper)
library(dplyr)
library(ggplot2)

# 2. Read ARACNe output with exact path
regulon_data <- read.table("/Users/diyahafiz/Downloads/network.txt",
                           header = TRUE,
                           sep = "\t")

# 1. First convert the regulon data to the correct format
regulon_list <- list()
for(reg in unique(regulon_data$Regulator)) {
 targets <- regulon_data[regulon_data$Regulator == reg, ]
 regulon_list[[reg]] <- list(
   tfmode = setNames(targets$MI, targets$Target),
   likelihood = setNames(1-targets$pvalue, targets$Target)
 )
}

# 2. Run VIPER analysis
viper_result <- viper(eset = z_scores_df,
                      regulon = regulon_list,
                      method = "rank",
```

```
                    minsize = 4,
                    eset.filter = FALSE,
                    verbose = TRUE)
str(viper_result)
# Convert viper_result to a matrix for visualization
viper_matrix <- as.matrix(viper_result)  # viper_result is already a matrix

# Verify that row and column names are preserved
head(rownames(viper_matrix))  # Should show gene names
head(colnames(viper_matrix))  # Should show sample/condition names
# Calculate variance for each gene across samples/conditions
var_genes <- apply(viper_matrix, 1, var)

# Get the top 50 most variable genes
top_50_var <- names(sort(var_genes, decreasing = TRUE)[1:50])

# Create a heatmap for the top 50 most variable genes
library(pheatmap)
pheatmap(viper_matrix[top_50_var, ],
        scale = "row",
        show_rownames = TRUE,
        main = "Top 50 Variable Regulators")

ordered_samples = c("X3.17T.Young", "X5.18T.Young", "X7.19T.Young", "X2.12A.Young",
"X4.13A.Young", "X6.14A.Young", "X8.15A.Young", "X9.16A.Young",
                    "X13.11T.Old", "X15.13T.Old", "X16.14T.Old", "X10.8A.Old",
"X12.6A.Old", "X14.7A.Old", "X17.9A.Old", "X18.10A.Old", "X19.11A.Old", "X11.10T.Old",
                    "X20.1T.AD", "X21.1A.AD", "X22.2T.AD", "X23.2A.AD",
"X24.3T.ADGeneGene", "X25.5T.AD", "X26.3A.AD", "X27.5A.AD", "X28.8T.AD", "X29.6T.AD",
"X30.9T.AD", "X31.7T.AD")

pdf(file = "/Users/diyahafiz/Documents/Systems_Biology/Heatmap_Viper_Matrix2.pdf",
width = 15, height=50)
heatmap(viper_matrix[,ordered_samples], Colv = NA)
dev.off()

gene_exp_matrix = gene_matrix_modified
rownames(gene_exp_matrix)= gene_matrix_modified$Gene
gene_exp_matrix$Gene = NULL

pdf(file = "/Users/diyahafiz/Documents/Systems_Biology/Heatmap_Gene_exp.pdf", width =
15, height=50)
```

```
heatmap(as.matrix(gene_exp_matrix[,ordered_samples]), Colv = NA)
dev.off()
```

Step 1: Data Loading and Preprocessing

The code begins by loading the normalized gene expression data and performing z-score transformation. The modified_file.txt contains log10 transformed TPM values. The data is structured to preserve gene names and sample information while converting to appropriate matrix format for VIPER analysis. The z-score transformation standardizes gene expression across samples, making the data comparable and suitable for regulatory network analysis.

Step 2: Format and Structure Setup

- Gene names are extracted and preserved separately
- Data is converted to a numeric matrix while maintaining sample annotations
- Z-score transformation is applied across genes (rows)
- Results are verified through summary statistics to ensure proper normalization (mean ≈ 0, SD ≈ 1)
- Data structure is checked and validated for both matrix and data frame formats

Step 3: VIPER Preparation

The ARACNe output network (network.txt) is processed into the specific format required by VIPER:

- Regulon objects are created for each transcription factor
- Mutual Information (MI) scores are used as interaction weights
- Likelihood values are calculated from p-values (1-pvalue)
- Data is organized into a nested list structure required by VIPER

Step 4: VIPER Analysis Execution

VIPER analysis is run with the following key parameters:

- method = "rank": Uses rank-based enrichment for robustness
- minsize = 4: Requires minimum 4 targets per regulator
- eset.filter = FALSE: Retains all expression data
- verbose = TRUE: Provides detailed progress information

Step 5: Visualization and Analysis

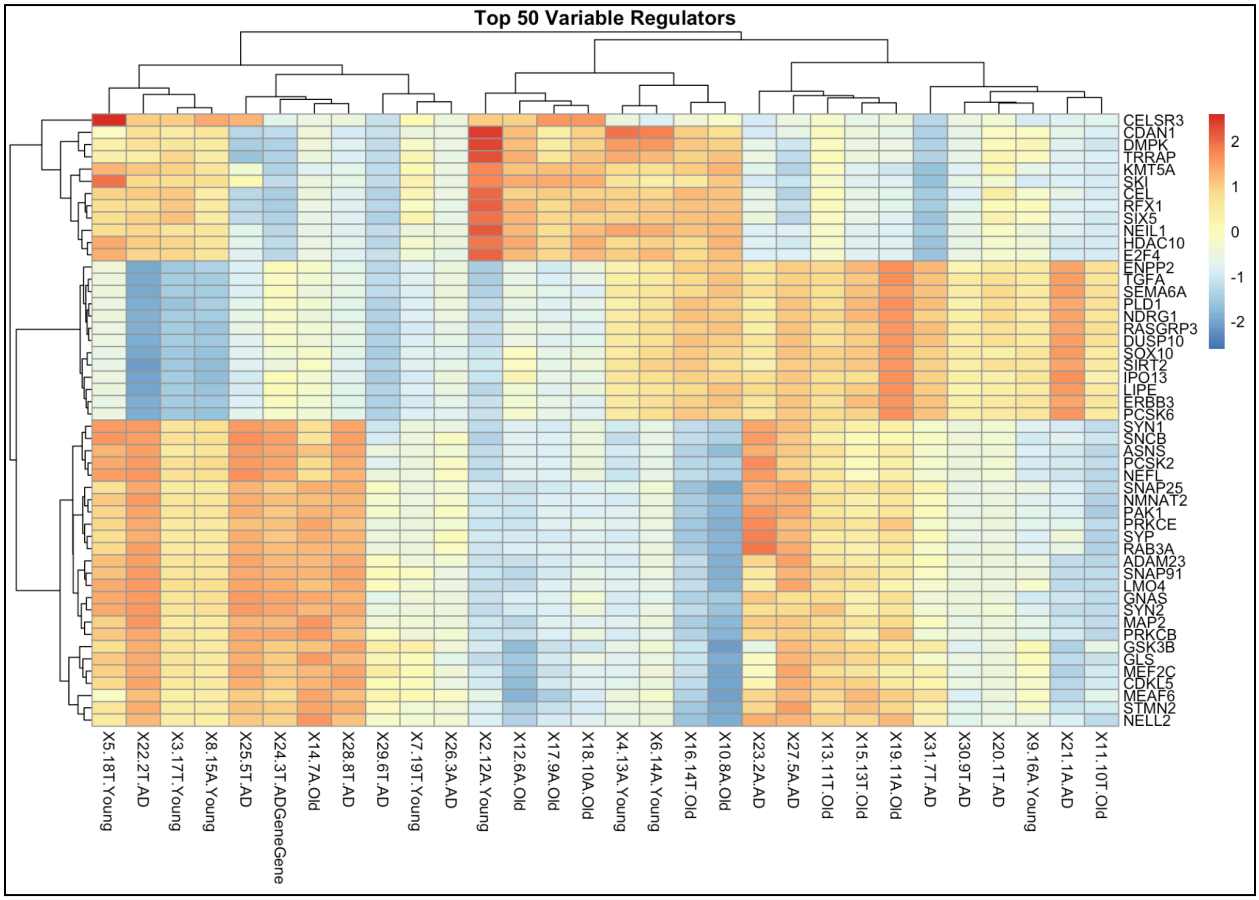
Multiple visualization approaches are implemented:

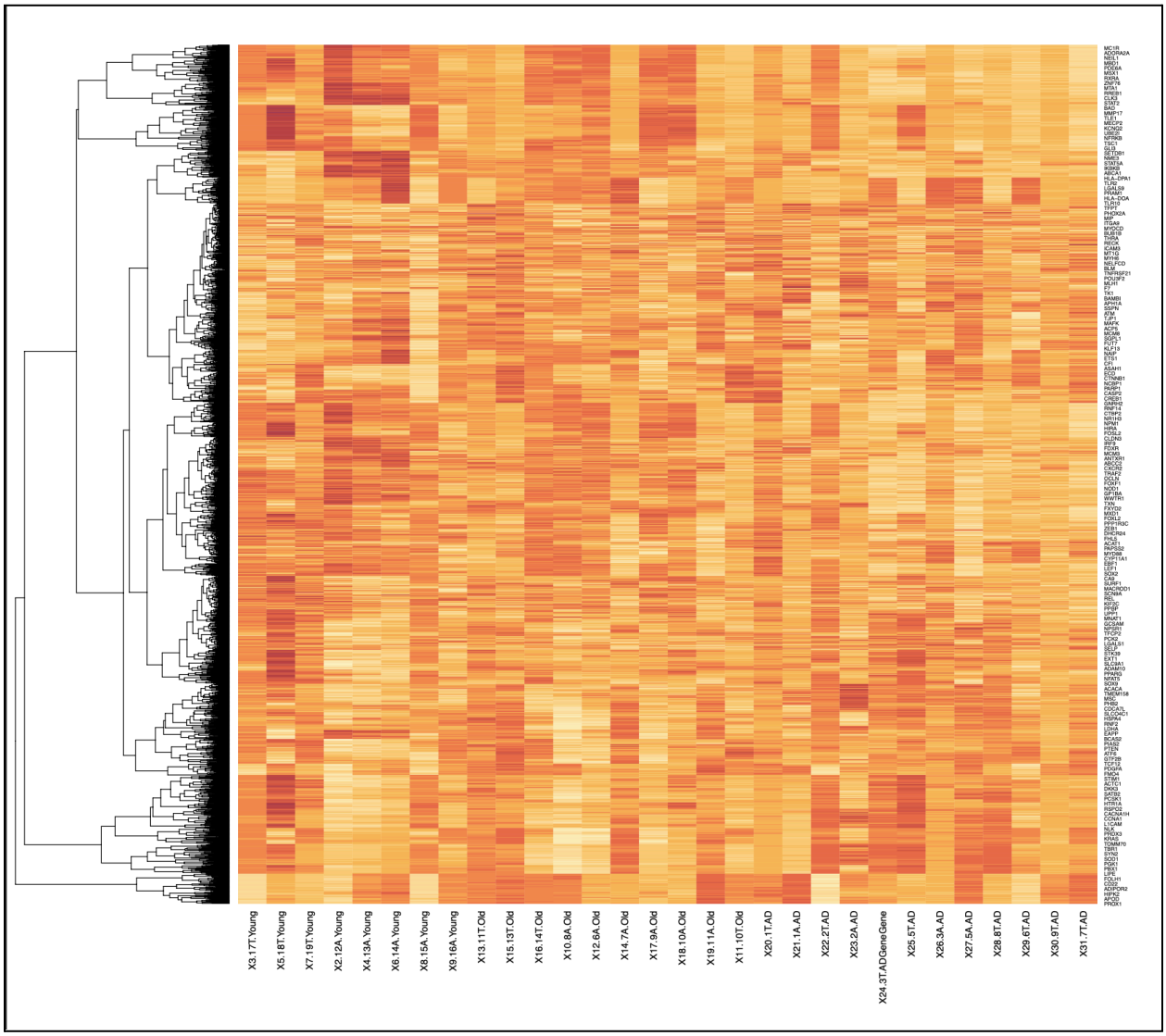
- Heatmap of top 50 most variable regulators
- Sample ordering by condition (Young, Old, AD)
- PDF output generation with specific dimensions
- Separate visualizations for VIPER results and gene expression

Parameters:

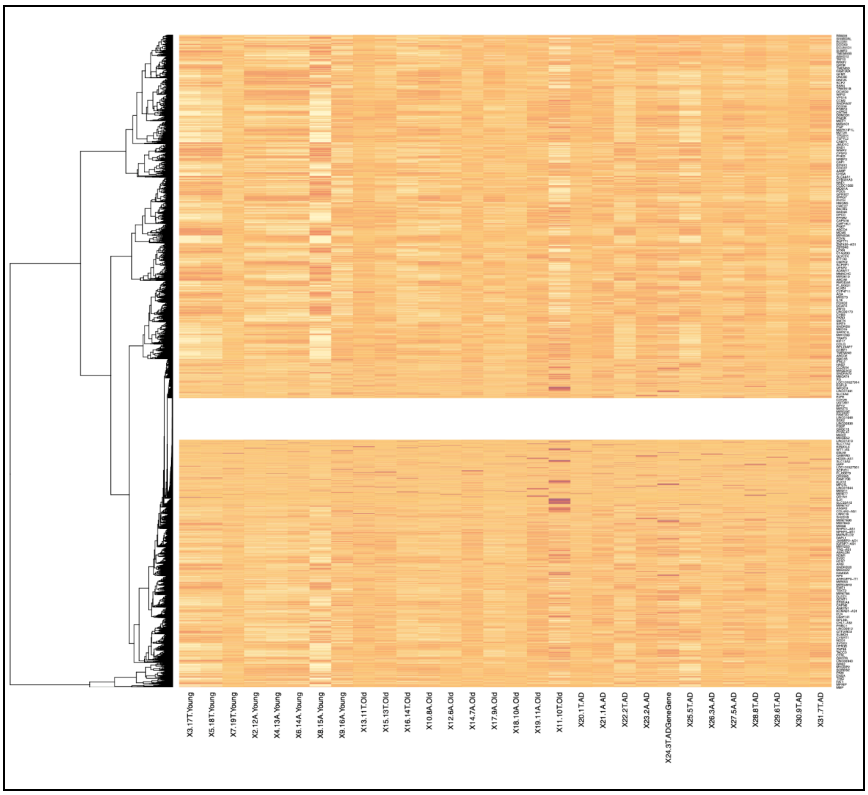
- Input: Z-score transformed expression data and ARACNe regulon network
- Visualization dimensions: width = 15, height = 50
- Sample ordering: Custom order from Young to AD samples
- Analysis focus: Top 50 most variable regulators
- Output format: PDF files for both VIPER and gene expression heatmaps

Viper Output





Heatmap_Viper_Matrix2.pdf



Heatmap_Gene_exp.pdf

Key observations:

1. Regulatory Patterns:
    1) Several distinct clusters of regulatory activity are visible but not definable/differentiable.
    2) Some regulators show strong activation (red) in specific conditions while being repressed (blue) in others.

2. Hierarchical Structure:
    1) The dendrogram at the top shows several major branches of sample clusters
    2) The regulators are also hierarchically organized on the left side, grouping those with similar activity patterns
3. Notable Regulators:
    1) CELSR3, CDAN1, and DMPK appear at the top of the heatmap of the Variability matrix. This could potentially be a key regulatory in AD if pattern is seen in a larger sample size.

2) Several transcription factors and signaling molecules like E2F4, TCFA, and HDAC10 show variable activity across conditions

This VIPER analysis reveals key regulatory proteins that may be driving the biological differences between young, aged, and diseased states, potentially highlighting therapeutic targets or disease mechanisms. The clustering patterns aren't defined enough to suggest coordinated regulatory programs that distinguish between these biological states.

Discussion

This final project explored the identification of bottleneck genes (master regulators) in Alzheimer's Disease (AD) using systems biology approaches. The integration of **ARACNe** and **VIPER** allowed for the construction of gene regulatory networks and the inference of protein activity based on RNA-Seq data from the GSE153873 dataset. While the project achieved significant progress, key steps remain to refine and validate the findings.

Key Insights

1. **Network Construction**:
    - The ARACNe analysis identified several transcription factors and regulatory interactions, laying a foundation for further exploration of master regulators in AD.
    - Topological analysis using Cytoscape revealed nodes with high betweenness centrality, indicative of their potential role as bottlenecks in the regulatory network. However, outliers like ARG1 highlight the need for a more nuanced interpretation.
2. **Protein Activity Inference**:
    - The VIPER output demonstrated regulatory activity patterns across Young, Old, and AD groups. While proteins like **CELSR3, CDAN1, and DMPK** emerged as potentially influential, their role as definitive master regulators cannot yet be confirmed without further validation.
3. **Unfinished Steps**:

- The project did not narrow down or rank master regulators. The planned application of machine learning (Random Forest) will address this limitation, prioritizing candidate regulators based on their relevance to AD pathology.

Challenges

- The clustering of regulators was not sufficiently distinct to define coordinated regulatory programs.
- The ARACNe-VIPER pipeline is sensitive to input variability, necessitating further refinement of preprocessing steps and data normalization.

Conclusion

This project marks a promising start in leveraging systems biology to identify regulatory elements in AD. The ARACNe and VIPER analyses established a framework for understanding gene-protein regulatory relationships, highlighting the complexity of AD's molecular mechanisms. However, the findings are preliminary, and further computational and experimental work is required.

Future Scope

Building on this foundation, the next steps include:

1. **Random Forest Ranking**:
    - Implement machine learning algorithms to rank master regulators based on their importance, integrating additional metrics like network centrality and expression variance.
2. **Validation of Findings**:
    - Experimentally validate candidate regulators using in vitro models of AD, focusing on transcription factors with high activity scores or centrality measures.
3. **Expansion of Analytical Frameworks**:
    - Integrate additional omics datasets, such as proteomics or epigenomics, to corroborate regulatory interactions.
    - Explore dynamic regulatory network models to capture temporal aspects of AD progression.

4. **Collaboration with AI/ML Courses**:
    - Use advanced AI/ML tools for feature selection and predictive modeling, potentially incorporating external datasets to enhance generalizability.
5. **Translation to Therapeutics**:
    - Investigate the therapeutic potential of master regulators through pathway analysis and drug repurposing studies.

This project sets the stage for future research, combining computational tools with experimental approaches to tackle the complexities of AD. The continued focus on refining analytical pipelines and applying machine learning methods promises to yield actionable insights into AD's regulatory landscape.

References

1. VIPER Package for Virtual Inference of Protein Activity by Enriched Regulon Analysis:
   Alvarez, M. J., Shen, Y., Giorgi, F. M., Lachmann, A., Ding, B. B., Ye, B. H., & Califano, A. (2016). Functional characterization of somatic mutations in cancer using network-based inference of protein activity. *Nature Genetics, 48*(8), 838–847. https://doi.org/10.1038/ng.3593

2. Network-Based Inference of Protein Activity:
   Alvarez, M. J., & Califano, A. (2017). Network-based inference of protein activity helps functionalize the genetic landscape of cancer. *PLoS Computational Biology, 13*(7), e1005596. https://doi.org/10.1371/journal.pcbi.1005596

3. VIPER Source: R/aracne.r:
   Califano Lab. (n.d.). *VIPER source code*. Retrieved from GitHub repository

4. VIPER: Visualization Pipeline for RNA-seq:
   Diaz, F., Stransky, N., & Engelman, J. A. (2020). VIPER: Visualization Pipeline for RNA-seq, a Snakemake workflow for efficient and complete RNA-seq analysis. *BMC Bioinformatics, 21*, 199. https://doi.org/10.1186/s12859-020-3514-6

5. ARACNe by Andrea Califano:
   Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Favera, R. D., & Califano, A. (2006). ARACNe: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics, 7*, S7. https://doi.org/10.1186/1471-2105-7-S1-S7

6. GitHub - ARACNe-AP Repository:
   Califano Lab. (n.d.). *ARACNe-AP: Network reverse engineering through AP inference of mutual information* [GitHub repository]. Retrieved from https://github.com/califano-lab/ARACNe-AP

7. Reverse Engineering Cellular Networks:
   Margolin, A. A., Wang, K., Lim, W. K., Kustagi, M., Nemenman, I., &

Califano, A. (2006). Reverse engineering cellular networks. *Nature Protocols*, *1*(2), 662–671. https://doi.org/10.1038/nprot.2006.106

8. Systematic Elucidation and Pharmacological Targeting of Tumor-Infiltrating Regulatory T Cell Master Regulators:
   You, R., Dehennaut, V., Lecomte, S., Dubus, P., Lesourne, R., & Rouquette, A. (n.d.). *Systematic elucidation and pharmacological targeting of tumor-infiltrating regulatory T cell master regulators*. Retrieved from https://pmc.ncbi.nlm.nih.gov/articles/PMC10193511/#ABS1

9. TRRUST Download Page:
   Han, H., Cho, J. W., Lee, S., Yun, A., Kim, H., Bae, D., … & Lee, I. (2018). TRRUST v2: an expanded reference database of human and mouse transcriptional regulatory interactions. *Nucleic Acids Research*, *46*(D1), D380–D386. https://doi.org/10.1093/nar/gkx1013

10. ARACNe-AP: Gene Network Reverse Engineering through Adaptive Partitioning Inference of Mutual Information:
    Lachmann, A., Giorgi, F. M., Lopez, G., & Califano, A. (2016). ARACNe-AP: gene network reverse engineering through adaptive partitioning inference of mutual information. *Bioinformatics*, *32*(14), 2233–2235. https://doi.org/10.1093/bioinformatics/btw216

11. GEO - NCBI:
    Edgar, R., Domrachev, M., & Lash, A. E. (2002). Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, *30*(1), 207–210. https://doi.org/10.1093/nar/30.1.207

12. BioMart - Ensembl:
    Kinsella, R. J., Kahari, A., Haider, S., Zamora, J., Proctor, G., Spudich, G., … & Flicek, P. (2011). Ensembl BioMarts: a hub for data retrieval across taxonomic space. *Database*, *2011*, bar030. https://doi.org/10.1093/database/bar030

13. VIPER Package:
    Alvarez, M. J., Shen, Y., Giorgi, F. M., Lachmann, A., Ding, B. B., Ye, B. H., & Califano, A. (2016). Functional characterization of somatic mutations in cancer using network-based inference of protein activity. *Nature Genetics*, *48*(8), 838–847. https://doi.org/10.1038/ng.3593

Appendix:

https://drive.google.com/drive/folders/1qGbHOPhxCWQ6l5KMar4OUat_5XeRUwGG?usp=drive_link (drive folder with all the input, output, data, code, and documentation for the project).

Name	Owner	Last modified	File size
ARACNe-AP-master	me	Dec 7, 2024 me	—
ARACNe_output_network.txt	me	Dec 7, 2024 me	8.8 MB
CytoscapeResult_network.txt default node.csv	me	Dec 7, 2024 me	2.9 MB
Final project documentation_T3	me	9:10 AM me	4.3 MB
final_log10_tpm_matrix.txt	me	Dec 6, 2024 me	10 MB
GSE153873_summary_count.star.txt	me	Dec 4, 2024 me	2.7 MB
Heatmap_Gene_exp.pdf	Diya Hafiz	Dec 6, 2024 Diya Hafiz	2.8 MB
Heatmap_Viper_Matrix2.pdf	Diya Hafiz	Dec 6, 2024 Diya Hafiz	325 KB
mart_export.txt	me	Dec 5, 2024 me	15.1 MB
modified_file.txt	Diya Hafiz	Dec 6, 2024 Diya Hafiz	10 MB
Powershell	me	Dec 7, 2024 me	140 KB
Tf_DatabaseExtract_v_1.01.csv	me	Dec 7, 2024 me	2 MB
trrust_rawdata.human.bioc.xml	me	Dec 7, 2024 me	3.7 MB