scGRN testing

Francesco Cecere

2024-09-25

```r
library(GENIE3)
library(doParallel)
library(igraph)
library(tidyverse)
library(DT)
library(reticulate)
```

```r
count_matrix <- readRDS("./../data/simatx.RDS")
adjm <- read.table("./../data/adjacency_matrix.csv", header = T, row.names = 1, sep = ",")
marker <- read.table("./../data/Tcell.marker.csv", header = T, sep = ",")

count_matrices <- list()
for (i in 1:5) {
  count_matrix_i <- as.data.frame(count_matrix[[i]])
  colnames(count_matrix_i) <- colnames(adjm)
  rownames(count_matrix_i) <- paste("cell", 1:nrow(count_matrix_i), sep = "")
  count_matrices[[i]] <- count_matrix_i
}
```

```r
count_matrices[[1]] %>%
    datatable(extensions = 'Buttons',
            options = list(
              dom = 'Bfrtip',
              buttons = c('csv', 'excel'),
              scrollX = TRUE,
              pageLength = 10),
            caption = "Simulated count matrix")
```

CSV Excel Search:

Simulated count matrix

	ACTN1	ADRB2	CASP8	CCR7	CD2	CD3D	CD3E	CD3G	CD4	CD5
cell1	0	0	0	7	7	0	0	1	7	8
cell2	1	2	1	7	2	10	37	1	0	29
cell3	0	0	0	0	29	0	8	2	1	0
cell4	0	2	0	0	23	15	2	5	0	8
cell5	0	0	0	0	5	0	16	11	12	14
cell6	1	0	0	2	7	1	0	0	1	2
cell7	3	21	0	20	8	12	24	1	1	38
cell8	7	1	8	0	3	17	3	1	8	8
cell9	0	2	6	1	39	2	0	0	18	0
cell10	0	38	0	0	1	4	9	0	3	2

Showing 1 to 10 of 1,000 entries Previous 1 2 3 4 5 … 100 Next

Gene Regulatory Network Inference Using GENIE3 and GRNBoost

Inferring Gene Regulatory Networks (GRNs) from gene expression data is a challenging task, typically tackled using machine learning algorithms. Both **GENIE3** and **GRNBoost** are widely used methods for GRN inference, based on ensemble learning models. GENIE3 employs **random forest regression**, while GRNBoost uses **gradient boosting**—each offering unique strengths for this problem.

GENIE3 Overview

GENIE3 and Random Forest Regression

GENIE3, which was the top-performing method in the **DREAM5 challenge** for GRN inference, utilizes **random forest** regression. Random forests are an ensemble method that constructs a large number of decision trees during training and outputs the mean prediction (for regression) of the individual trees.

For GRN inference, the random forest algorithm in GENIE3 is used as follows: 1. For each target gene g, a random forest is trained where the expression of the target gene g is predicted using the expression levels of all potential transcription factors (TFs) in the dataset. 2. Each tree in the random forest randomly samples a subset of the available features (transcription factors) and a bootstrap sample of the training data. 3. The importance of each transcription factor for predicting the target gene is measured by aggregating the feature importance scores across all trees.

The final output is a ranked list of transcription factors for each target gene, where the importance score reflects the strength of the regulatory relationship.

The random forest model can be described as:

$$f(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$$

Where T is the number of trees, and $h_t(x)$ is the prediction from the t-th tree. The importance of a transcription factor TF_i for predicting g is calculated based on the decrease in impurity (e.g., Gini impurity or variance reduction) across all splits where TF_i is used.

This approach is advantageous because: - **Non-linear relationships**: Random forests can model complex, non-linear interactions between transcription factors and target genes. - **Robustness to noise**: By averaging across many trees, random forests reduce the likelihood of overfitting to noise in the data.

Mathematical Formulation for Random Forest in GENIE3

Let $X = \{x_1, x_2, \ldots, x_p\}$ be the matrix of transcription factor expressions and y_g be the expression of the target gene g. For each g, the random forest minimizes the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_g^{(i)} - f(X^{(i)})\right)^2$$

Where n is the number of samples, and $f(X^{(i)})$ is the predicted expression for sample i.

1. Handling Zero-Inflation in Single-Cell Data

In single-cell RNA sequencing (scRNA-seq) data, zero-inflation is a common issue, where many genes have expression levels recorded as zero across a large number of cells. This sparsity can challenge many traditional statistical models. However, GENIE3 is based on tree-based methods, specifically **Random Forests**, which are inherently robust to sparse data.

While GENIE3 does not explicitly model zero-inflation, Random Forests naturally handle zero-inflated data due to their ability to partition data based on splits at specific thresholds, thus segregating zeros from non-zero values without overfitting to the zeros. This makes GENIE3 well-suited to noisy and sparse data like scRNA-seq.

2. Random Forest Model in GENIE3

GENIE3 constructs a **Random Forest** for each target gene, where the goal is to predict its expression level based on the expression of all other genes. Random Forest is an ensemble method that builds multiple decision trees during training. Each tree is constructed from a random subset of the data, and the final prediction is made by averaging the results (for regression) or taking a majority vote (for classification).

In the context of GENIE3:

- For each target gene, a separate Random Forest is trained.
- Each tree in the forest helps to identify key genes (predictors) that contribute to the expression of the target gene.
- The importance of a regulatory relationship is determined by how often a predictor gene is selected in the decision trees and the quality of the split it produces.
- At the end of this process, GENIE3 provides a ranked list of regulatory interactions, indicating which genes are most likely to regulate each other.

This method allows GENIE3 to infer gene regulatory networks from expression data, making it a powerful tool for discovering potential regulatory relationships, especially in high-dimensional, sparse datasets like scRNA-seq.

References for GENIE3

- Huynh-Thu, V. A., et al. (2010). “Inferring regulatory networks from expression data using tree-based methods.” *PLoS One*, 5(9): e12776.


```r
set.seed(123)
regulatory_network_genie3 <- GENIE3(t(count_matrices[[1]]))

# Extract link list (gene regulatory interactions) from GENIE3 results
link_list_genie3 <- getLinkList(regulatory_network_genie3)

link_list_genie3 %>%
    datatable(extensions = 'Buttons',
            options = list(
              dom = 'Bfrtip',
              buttons = c('csv', 'excel'),
              scrollX = TRUE,
              pageLength = 10),
            caption = "GENIE3 output")
```

CSV Excel Search:

GENIE3 output

	regulatoryGene	targetGene	weight
1	IL17A	IL17RA	0.322008055273475
2	IL17RA	IL17A	0.283669015131738
3	FAS	CASP8	0.251722854604797
4	CASP8	FAS	0.24668021682025
5	GIMAP8	GIMAP5	0.240384092675688
6	SLC9A3R1	ADRB2	0.235725109072563
7	GIMAP5	GIMAP8	0.222829474266123
8	VCL	ACTN1	0.210572262092281
9	CD5	CD4	0.186091924349679
10	ADRB2	SLC9A3R1	0.184547915337974

Showing 1 to 10 of 3,080 entries Previous 1 2 3 4 5 … 308 Next

```r
write.csv(link_list_genie3, "genie3_network.csv", row.names = FALSE)
```

```r
gene_names <- unique(c(link_list_genie3$regulator, link_list_genie3$target))

adj_matrix_genie3 <- matrix(0, nrow = length(gene_names), ncol = length(gene_names))
rownames(adj_matrix_genie3) <- colnames(adj_matrix_genie3) <- gene_names

# Fill the adjacency matrix based on the links from GENIE3 with a weight condition
for (i in 1:nrow(link_list_genie3)) {
  regulator <- link_list_genie3$regulator[i]
  target <- link_list_genie3$target[i]
  weight <- link_list_genie3$weight[i]

  # Only set 1 if the weight is >= 0.1
  if (weight >= 0.1) {
    adj_matrix_genie3[regulator, target] <- 1
  }
}

adj_matrix_genie3 %>%
    datatable(extensions = 'Buttons',
            options = list(
              dom = 'Bfrtip',
              buttons = c('csv', 'excel'),
              scrollX = TRUE,
              pageLength = 10),
            caption = "GENIE3 adjacency matrix 0.1")
```

CSV Excel Search:

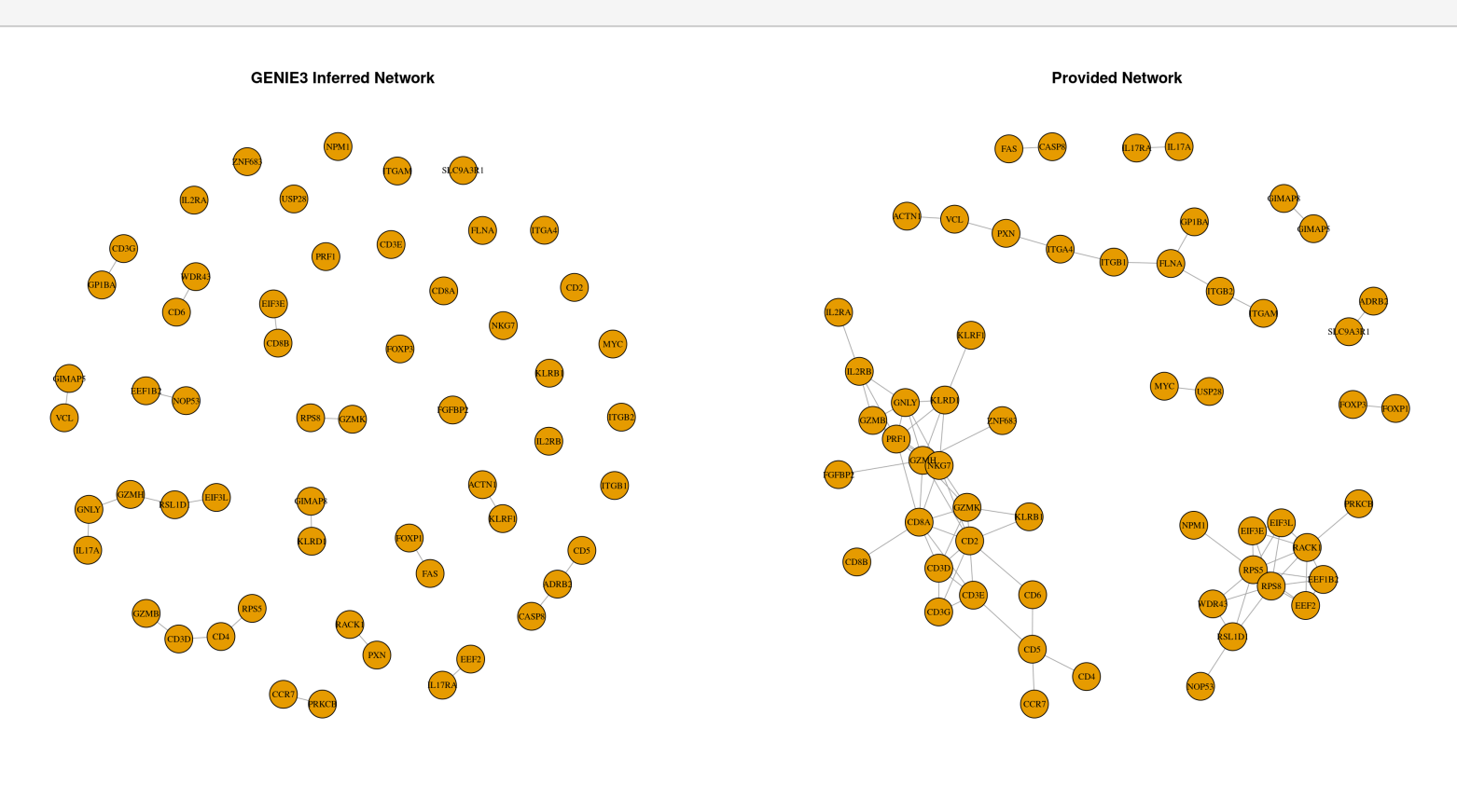
GENIE3 adjacency matrix 0.1

	IL17A	IL17RA	FAS	CASP8	GIMAP8	SLC9A3R1	GIMAP5	VCL	CD5
IL17A	0	0	0	0	0	0	0	0	0
IL17RA	0	0	0	0	0	0	0	0	0
FAS	0	0	0	0	0	0	0	0	0
CASP8	0	0	0	0	0	0	0	0	0
GIMAP8	0	0	0	0	0	0	0	0	0
SLC9A3R1	0	0	0	0	0	0	0	0	0
GIMAP5	0	0	0	0	0	0	0	1	0
VCL	0	0	0	0	0	0	0	0	0
CD5	0	0	0	0	0	0	0	0	0
ADRB2	0	0	0	1	0	0	0	0	1

Showing 1 to 10 of 56 entries Previous 1 2 3 4 5 6 Next

```r
write.csv(adj_matrix_genie3, "genie3_adjacency_matrix.csv")
```

```r
# Create igraph objects for both networks
graph_genie3 <- graph_from_adjacency_matrix(adj_matrix_genie3, mode = "undirected")
graph_provided <- graph_from_adjacency_matrix(as.matrix(adjm), mode = "undirected")

par(mfrow = c(1, 2))

# Plot GENIE3 Network
plot(graph_genie3, main = "GENIE3 Inferred Network", vertex.label.color = "black",
     vertex.size = 10, edge.arrow.size = 0.5, vertex.label.cex = 0.7)

# Plot Provided Network
plot(graph_provided, main = "Provided Network", vertex.label.color = "black",
     vertex.size = 10, edge.arrow.size = 0.5, vertex.label.cex = 0.7)
```




```r
par(mfrow = c(1, 1))
```

GRNBoost2 Overview

GRNBoost and Gradient Boosting

GRNBoost, an alternative to GENIE3, employs **gradient boosting machines (GBM)**, a method that iteratively builds a sequence of shallow decision trees, where each new tree corrects the errors made by the previous ensemble. It is based on boosting, where weak learners (typically decision stumps or shallow trees) are sequentially added, and the model is updated by minimizing a loss function.

The steps in GRNBoost are: 1. Similar to GENIE3, a regression model is built for each gene g to predict its expression using the expression levels of candidate transcription factors. 2. Instead of using random forests, GRNBoost uses gradient boosting, where each new tree is added to reduce the residual errors of the previous trees. 3. The importance of each transcription factor is computed based on how much each tree reduces the loss function when it includes that TF as a predictor.

Gradient boosting can be mathematically formulated as:

$$f(x) = \sum_{m=1}^{M} \lambda h_m(x)$$

Where $h_m(x)$ is a decision tree at step m, λ is the learning rate, and M is the total number of trees. The loss function is typically the squared error for regression tasks:

$$L(y, f(x)) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

The model is updated by adding a new tree that reduces the gradient of this loss function:

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \lambda h_m(x)$$

The main advantages of gradient boosting in GRNBoost include: - **Efficiency**: By iteratively refining the model with shallow trees, gradient boosting can achieve high accuracy without requiring deep trees like in random forests. - **Early stopping**: GRNBoost2 implements an early stopping mechanism based on the out-of-bag improvement estimates, which helps avoid overfitting and reduces computation time.

Mathematical Formulation for Gradient Boosting in GRNBoost

For a target gene g, the expression is modeled as:

$$f(X) = \sum_{m=1}^{M} \lambda h_m(X)$$

Where each $h_m(X)$ is a shallow tree built to minimize the loss $L(y_g, f(X))$, where $f(X)$ represents the predicted expression values for gene g.

References for GRNBoost

- Friedman, J.H. (2001). “Greedy function approximation: a gradient boosting machine.” *Annals of Statistics*, 29(5): 1189-1232.
- Moerman, T., et al. (2018). “GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks.” *Bioinformatics*, 35(12): 2159-2161.

1. Handling Zero-Inflation in Single-Cell Data

In single-cell RNA sequencing (scRNA-seq) data, the high occurrence of zero expression values, or zero-inflation, is a significant challenge for many computational models. Zero-inflation occurs when a large number of genes are either not expressed or their expression levels are too low to detect in many cells, leading to sparse datasets.

GRNBoost2 is built on **Gradient Boosting Machines (GBMs)**, which are tree-based models. Like Random Forests, GBMs can handle sparsity well because tree-based methods make decisions by splitting data based on feature values. This allows them to naturally partition and differentiate between zero and non-zero values without requiring explicit modeling of zero-inflation. The ability of GRNBoost2 to efficiently handle sparse data makes it a strong tool for gene regulatory network (GRN) inference in scRNA-seq datasets.

2. Gradient Boosting Model in GRNBoost2

GRNBoost2 is an implementation of the **Gradient Boosting Machine (GBM)** algorithm, which is an ensemble learning method that builds models sequentially. Each model attempts to correct the errors of the previous models, creating a stronger predictive model over time. In GRNBoost2, GBMs are used to infer gene regulatory networks by predicting the expression of a target gene based on the expression of other genes.

Here’s how it works in the context of GRNBoost2:

- For each target gene, a separate Gradient Boosting Machine model is built, using the expression of all other genes as input features.
- GBMs build multiple decision trees sequentially, where each tree attempts to minimize the residual errors made by the previous trees.
- The regulatory importance of each gene (predictor) is assessed based on how often and how effectively it is used as a splitting feature in the decision trees.
- At the end of this process, GRNBoost2 provides a ranked list of potential regulatory interactions between genes, with higher scores indicating stronger regulatory influence.

The sequential learning nature of Gradient Boosting allows GRNBoost2 to refine predictions iteratively, making it well-suited to complex data structures like those found in high-dimensional gene expression datasets.

```r
#sudo apt-get install python3-venv
use_python("/usr/bin/python3", required = TRUE)
py_config()
```

```
## python:         /usr/bin/python3
## libpython:      /usr/lib/python3.8/config-3.8-x86_64-linux-gnu/libpython3.8.so
## pythonhome:     //usr://usr
## version:        3.8.10 (default, Sep 11 2024, 16:02:53)  [GCC 9.4.0]
## numpy:          /home/francescoc/.local/lib/python3.8/site-packages/numpy
## numpy_version:  1.22.4
##
## NOTE: Python version was forced by RETICULATE_PYTHON
```

```r
arboreto <- import("arboreto.algo")
pandas <- import("pandas")
numpy <- import("numpy")

count_matrix_df <- as.data.frame(count_matrices[[1]])
genes <- colnames(count_matrix_df)
df_pandas <- pandas$DataFrame(data = count_matrix_df, columns = genes, index = rownames(count_matrix_d
f))

grn_links <- arboreto$grnboost2(df_pandas, gene_names = genes)

grn_links %>%
    datatable(extensions = 'Buttons',
            options = list(
              dom = 'Bfrtip',
              buttons = c('csv', 'excel'),
              scrollX = TRUE,
              pageLength = 10),
            caption = "GRNBoost2 links")
```

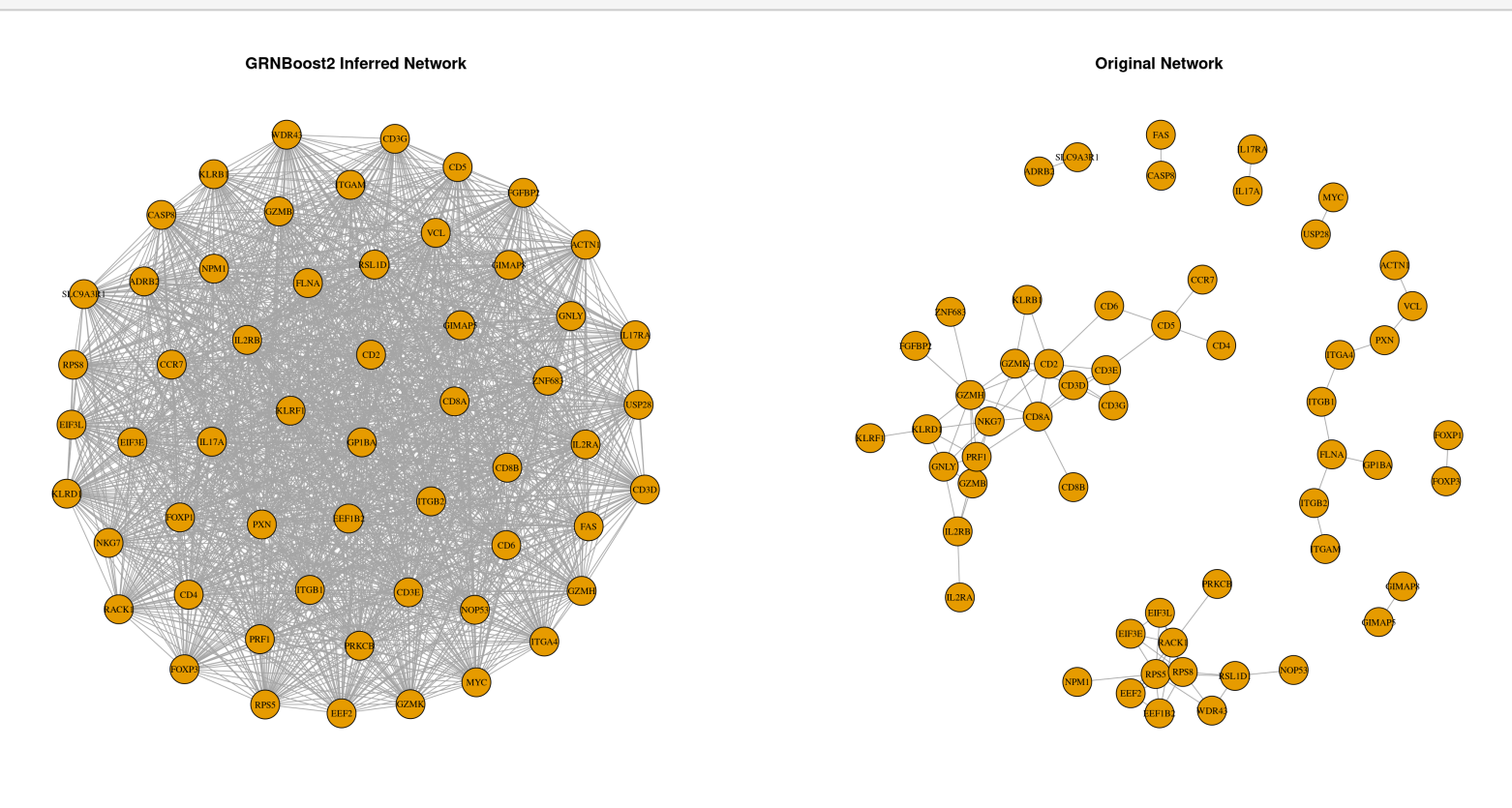
CSV Excel Search:

GRNBoost2 links

	TF	target	importance
1	IL17RA	IL17A	157.21415715766
2	IL17A	IL17RA	85.9466175617696
3	KLRF1	KLRD1	85.7620559348377
4	FAS	CASP8	77.5390350849445
5	GIMAP5	GIMAP8	66.7388536661882
6	ITGA4	ITGB1	65.7161926688149
7	CD5	CD4	59.0015586565449
8	ITGB2	FLNA	56.3414599796428
9	SLC9A3R1	ADRB2	55.3589777609396
10	CD4	CD5	55.3229298972722

Showing 1 to 10 of 3,024 entries Previous 1 2 3 4 5 … 303 Next

```r
unique_genes <- unique(c(grn_links$TF, grn_links$target))  # Get unique genes from GRNBoost2
adj_matrix_grnboost <- matrix(0, nrow = length(unique_genes), ncol = length(unique_genes))
rownames(adj_matrix_grnboost) <- unique_genes
colnames(adj_matrix_grnboost) <- unique_genes

for (i in 1:nrow(grn_links)) {
  tf <- grn_links$TF[i]
  target <- grn_links$target[i]
  adj_matrix_grnboost[tf, target] <- 1  # Set the edge in the adjacency matrix
}

adj_matrix_original <- as.matrix(adjm)

graph_grnboost <- graph_from_adjacency_matrix(adj_matrix_grnboost, mode = "undirected")
graph_original <- graph_from_adjacency_matrix(adj_matrix_original, mode = "undirected")

# Set up side-by-side plotting
par(mfrow = c(1, 2))

# Plot GRNBoost2 Network
plot(graph_grnboost, main = "GRNBoost2 Inferred Network", vertex.label.color = "black",
     vertex.size = 10, edge.arrow.size = 0.5, vertex.label.cex = 0.7)

# Plot Original Network
plot(graph_original, main = "Original Network", vertex.label.color = "black",
     vertex.size = 10, edge.arrow.size = 0.5, vertex.label.cex = 0.7)
```




```r
# Reset plotting layout
par(mfrow = c(1, 1))
```

Conclusion

Both GENIE3 and GRNBoost provide powerful and scalable methods for inferring GRNs from high-dimensional gene expression data. GENIE3 leverages the power of random forests to capture complex relationships, while GRNBoost uses gradient boosting for computational efficiency. These methods are widely applicable, especially in the context of large datasets from single-cell RNA sequencing experiments, enabling high-resolution understanding of gene regulatory dynamics.