Scribe: Gene regulation visualization, causal network inference for single-cell RNA-Seq experiments

Xiaojie Qiu

University of Washington,
Seattle, Washington, USA
xqiu@uw.edu

Arman Rahimzamani

University of Washington,
Seattle, Washington, USA
armanrz@uw.edu

September 18, 2017

Abstract

Cellular fate commitment is governed by hierarchical gene regulatory networks. Previous network inference approaches are inherently incapable of resolving complex causal relationships since they are designed for small-scale and static bulk measurements. Here we propose *Scribe*, a toolkit that employs *Direct Information* to reconstruct causal regulatory networks using single-cell RNA-seq data. *Scribe* detects pairs of genes that interact and determines *causality* through strength of *information transfer* from one to the other. Our technique exploits the fact that an upstream regulatory gene's expression changes before its downstream targets undergo coherent changes. To calibrate the expected time lag between upstream and downstream genes, Scribe analyzes single-cell expression kinetics as the cells progress through pseudotime along a cellular trajectory.

Scribe outperforms alternative approaches, including *Granger causality* and *cross-convergence mapping* (CCM), across systematic synthetic simulation datasets. Applying *Scribe* to several single-cell RNA-seq datasets spanning diverse biological processes including dendritic cells' response to LPS stimulation and cellular reprogramming, we find that *Scribe* reconstructs networks which are supported by either literature or ChIP-Seq/ATAC-seq datasets. Finally, we use *Scribe* to yield novel insights into the gene regulatory hierarchy of hematopoiesis. We anticipate that *Scribe* will enable single-cell biologists to reconstruct regulatory networks governing cell lineage differentiation for each of the many cell types in the human body.

Contents

1 Introduction

The *Scirbe* package provides a toolkit for visualizing and reconstructing complex gene regulation for single cell genomics data. This vignette provides an overview of causal regulatory network reconstruction with single cell genomics data by *Scirbe* (In this initial vignettee, we mainly rely on simulation data to demonstrate *Scribe*'s power). *Scirbe* is a collaboration between *Trapnell lab* at UW Genome Sciences department and *Kannan lab* at UW Electrical Engineering department. At the core of *Scribe*, we used *restricted direct information*, which has been shown to out-performan than "symmetric" mutual information, "linear" Granger causality and the newly proposed "deterministic" CCM method. The users of *Scribe* are expected to have already correctly reconstructed the developmental trajectory for their single-cell genomics dataset with *Monocle 2* because it relies on the pseudotime resolved data to infer the network causality.

The restricted direct information is an extension of mutual information which takes into account the direction of information flow, under the assumption that the underlying processes can be described by a first order Markov model. Mutual information is estimated using the Kraskov method (KSG method), which builds on a nearest-neighbor framework.

Scribe can help you perform three main types of analysis:

1. **Estimating and visualzing pairwise gene regulation.** Understanding how one gene activates or inhibits another gene in development, disease and through life is important. Scribe provides various function to estimate the newly developed restricted direct information (RDI) and conditional RDI (cRDI) to reveal the strenght of gene regulation. Those gene regulations can be also intuitively visualized through improved advanced visualization approach(based on the DREVI). Additionally, Scribe also supports other popular regulation inference methods, including correlation, mutual information, Granger causality and CCM.

2. **Inferring spatial/temporal gene regulation.** Gene regulation is dynamic and spatial-specific. Scribe helps to identify the period of active gene regulation which may correspond to critical time point for gene intervention for genetic or immunotherapy, etc. In theory, *Scribe* can be also applied to spatially resolved data to reveal spatial-specific gene regulation.

3. **Reconstructing large-scale sparse causal regulatory network.** Knowing the gene regulation at systems level is important for us to better understand the mechanisms of cell differentiation and carcinogenesis, etc. The regulatory hierarchy, network motifs / hubs are also of great clinic interests. Scribe provides a newly invented method for recovering a sparse direct network capturing the network regulation from the data but also satisfying well-studied large-scale network topologies.

2 Reconstructing causal network with Scribe for scRNA-seq experiment

Scribe relies on *Monocle 2* for reconstructing the single-cell trajectory before inferring causal network. We extended *Monocle 2* in a few aspects to account for the specific requirement for *Scribe*. For example, in order to avoid the over-crowded nature of the cells at the terminal regions of the trajectory, we applied a simple strategy to first run Monocle 2 using DDRTree to get the trajectory with complex branches and then reorder the trajectory for each branch (from the source state to the terminal state) using *principal curve* (or *simplePPT*) with the low dimension space obtained from the diffusion map dimension reduction technique. *Scribe* provides various methods for infering the regulatory relationship, mutual information, Granger causality and CCM. We used the implementation from *parmigene*, *vars* and *rEDM* respectively. To intuitively visualize the casual regulation between pairs of genes, we generalized the *DREVI* approach ([]). The network visualization relies on the *igraph*, *netbiov*, *arcdiagram* as well as the *HiveR* packages.

Before we can apply *Scribe* for causal network inference, we need to first run Monocle 2 for trajectory reconstruction. In the following expression we will use the pre-prepared dataset from the neuron simulation introduced in the Monocle 2 paper. For analyzing on a real dataset, we need to consider the over-crowded nature of the cells at the terminal regions of the Monocle 2 trajectory. For resolving this issue, we will apply a simple strategy by first running *Monocle 2* using *DDRTree* to get the trajectory with complex branches and then reorder the trajectory for each branch (from the source state to the terminal state) using *principal curve* (*simplePPT*) with the low dimension space obtained from the *diffusion map* dimension reduction technique. See a demo for the lung dataset below. Note that *Scribe* doesn't consider the relative value of the pseudotime and only use the order of cell in pseudotime for causal network inference. The non-uniform distribution of cells across pseudotime is partially adjusted when we calculate the RDI (See details in the method section).

2.1 Prepare the data for Scribe analysis

```
rm(list = ls())
load("/Users/xqiu/Dropbox (Cole Trapnell's Lab)/Monocle 2/first_revision/Monocle2_revision/RData/fig3.RD
load('/Users/xqiu/Dropbox (Personal)/Projects/Causal_network/causal_network/RData/neuron_network') # net
work data

# neuron_network not exist
neuron_network$Type <- c('Neuron', 'Oligo', 'Astro', 'Neuron', 'AO',
                         'Neuron', 'Neuron', 'Neuron', 'Neuron', "Neuron",
                         'AO', 'AO', 'Astro', 'Oligo', 'Olig', 'Astro',
                         'Astro', 'Astro', 'Olig', 'Astro', 'Oligo')

#
fData(neuron_sim_cds)$gene_short_name <- fData(neuron_sim_cds)$gene_short_names
fData(na_sim_cds)$gene_short_name <- fData(na_sim_cds)$gene_short_names
```

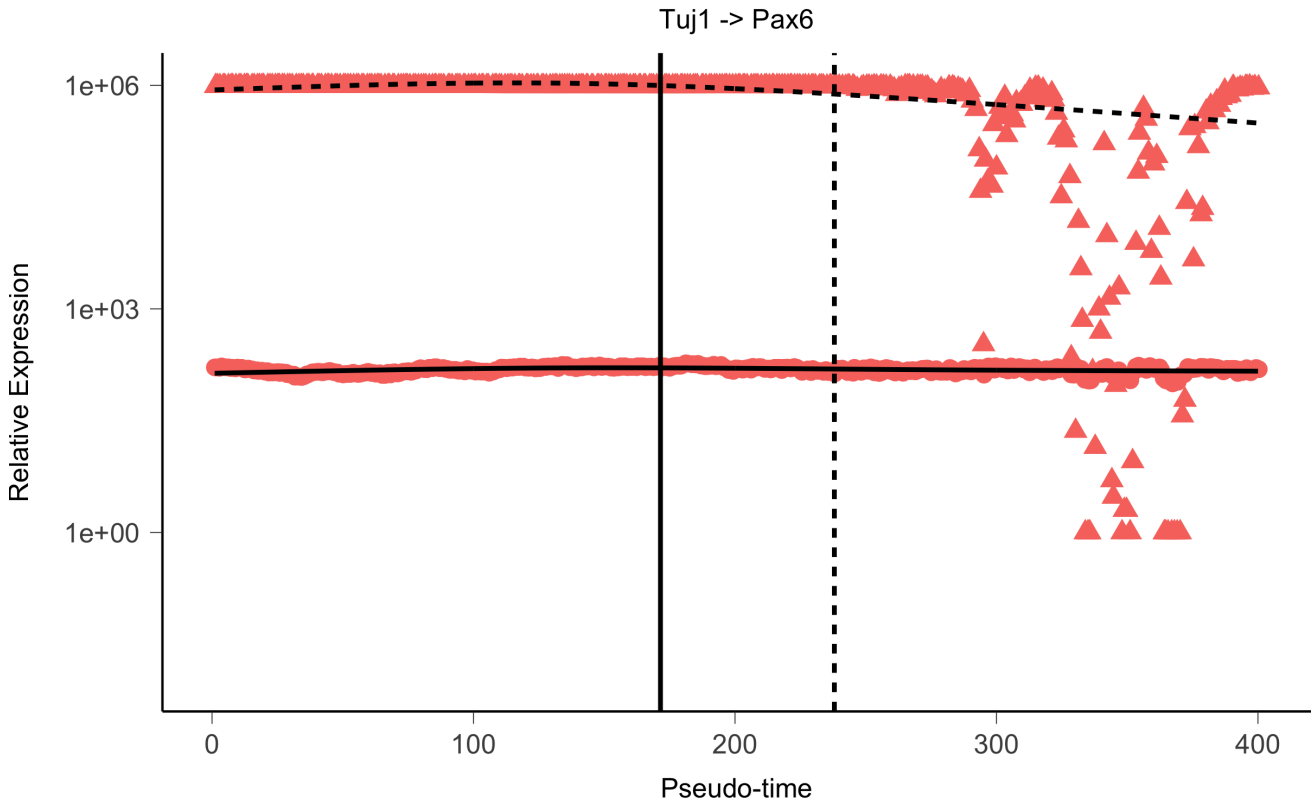
Add the demo for the lung dataset here:

```
# add code here for demo of the lung dataset
```

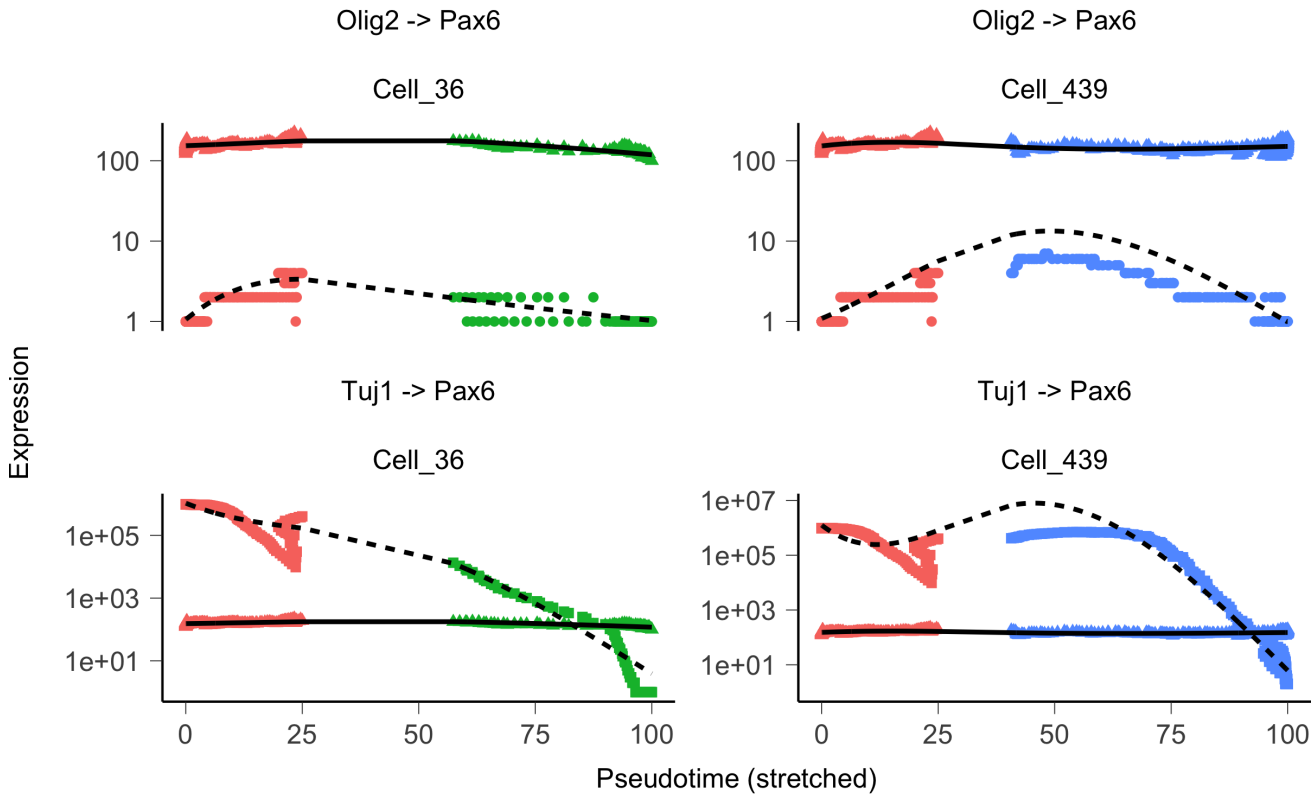
2.2 Visualize pairwise gene interaction

The easiest way to visualize the pairwise causal gene interaction is to use **plot_gene_pairs_in_pseudotime** to plot the expression dynamics of the gene pair along the pseudotime. For a branched trajectory, we can also use **plot_gene_pairs_branched_pse** to plot the expression dynamics of the gene pair along the pseudotime in two separate lineages. Note that **neuron_sim_cds** is the cds describing the commitment of neuron lineage while the **na_sim_cds** for both of the neuron and the astrocyte lineages.

```
# show the pair-wise gene plot:
neuron_sim_cds@lowerDetectionLimit <- 0.01
exprs(neuron_sim_cds) <- 10^(exprs(neuron_sim_cds)) #increase the data avoid the fitting issue
exprs(na_sim_cds) <- 10^(exprs(na_sim_cds)) #increase the data avoid the fitting issue
plot_gene_pairs_in_pseudotime(neuron_sim_cds[, ], gene_pairs_mat =  as.matrix(neuron_network[1, 1:2]), n
  monocle:::monocle_theme_opts() + xacHelper::nm_theme()

## gene_name is Pax6
## inflection_point is 171.5
## gene_name is Tuj1
## inflection_point is 238
```



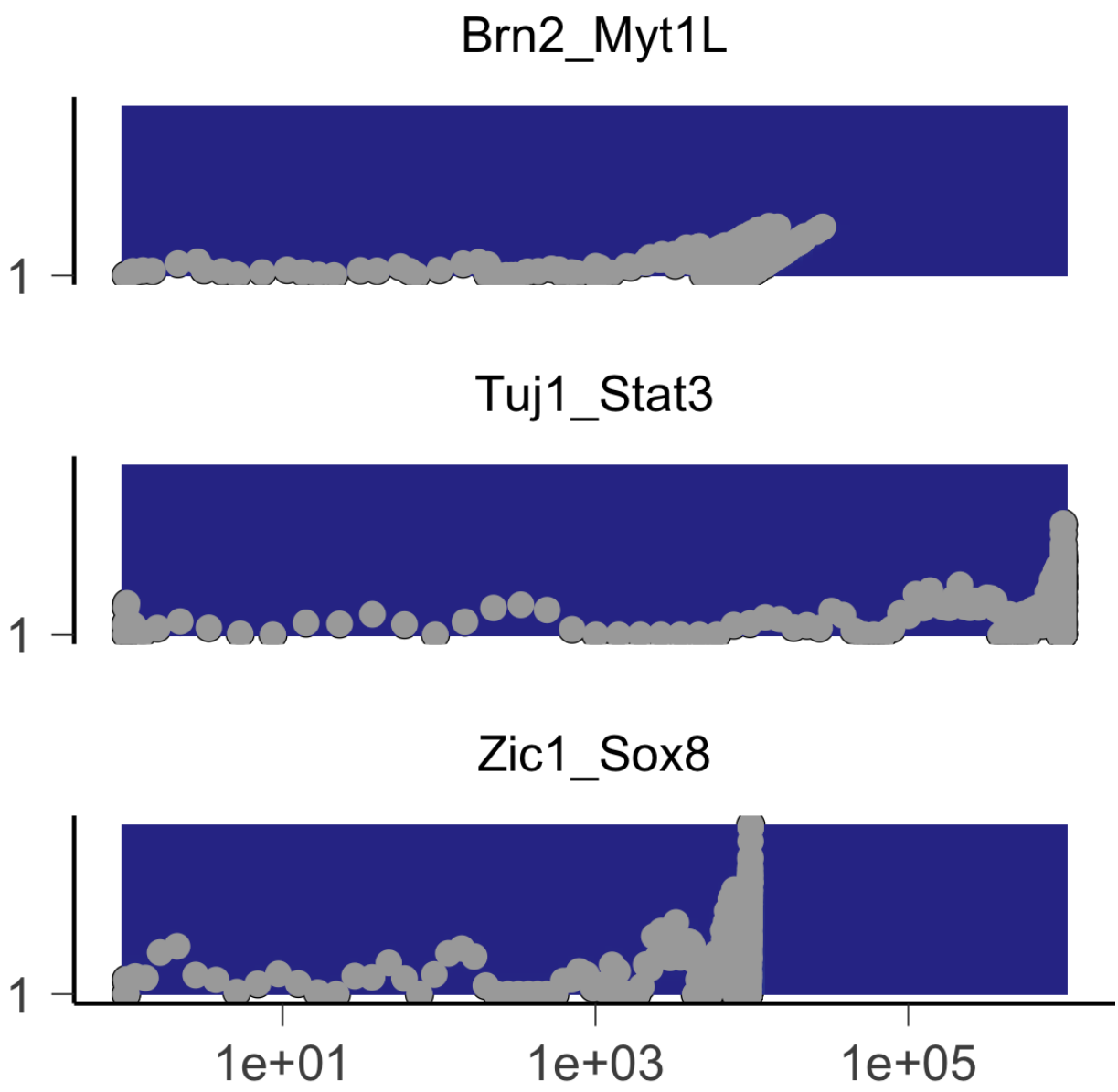

```
plot_gene_pairs_branched_pseudotime(na_sim_cds, gene_pairs_mat = as.matrix(neuron_network)[1:2, 1:2], n_
  monocle:::monocle_theme_opts() + xacHelper::nm_theme() # neuron_sim_cds is not a branch tra-
jectory
```



Scribe also supports plotting the scatter plot for the gene pair. **plot_gene_pairs** does exactly that but also reveal more abundantly distributed regions with a contour plot. In order to calculate the RDI values, Scribe computes the conditional mutual information of the state for $X_{t-\sigma}$, Y_t and Y_{t-1} (see methods). We can use **plot_rdi_gene_pairs** to visualize this state space. An interesting finding from the CCM is that time-lagged data for a single-variable is able

to recover a shadow manifold corresponding to the original manifold. Here I will show the 3D space for the variable $Pax6, Mash1, Hes5$ and the shadow manifold for Y_t, Y_{t-1}, Y_{t-2} (Y corresponds to $Mash1$). You can immediately identify that the time-lagged shadow manifold captures the topography of the original manifold.

```
# we skip some functions below because they uses plotly package whose figures can be rendered in a pdf f
plot_gene_pairs(neuron_sim_cds, matrix(c('Zic1', 'Sox8', 'Brn2', 'Myt1L', 'Tuj1', 'Stat3'), ncol = 2, by
row = T)) +
  monocle:::monocle_theme_opts() + xacHelper::nm_theme()
```



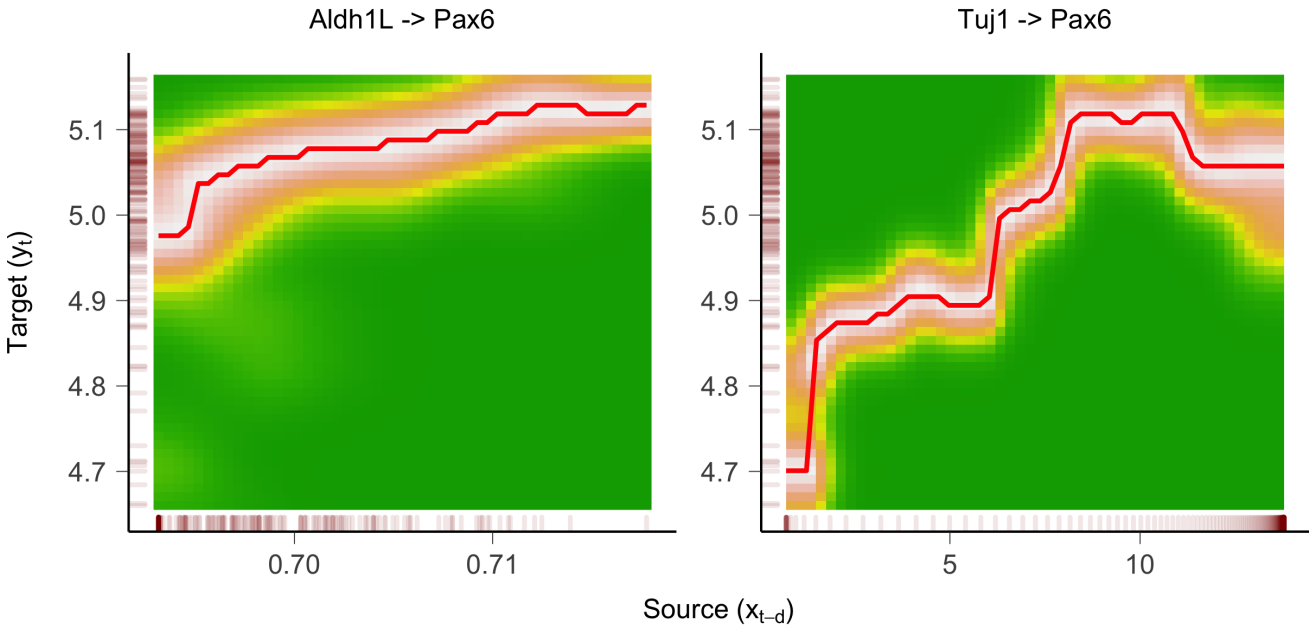

```
x <- exprs(neuron_sim_cds)['Pax6', ]
y <- exprs(neuron_sim_cds)['Mash1', ]
z <- exprs(neuron_sim_cds)['Hes5', ]

# plot_rdi_gene_pairs(x, y)

# plot_ly(type = 'scatter3d', x = log10(x), y = log10(y), z = log10(z), mode = 'markers') # show the ori
inal space
#
```

```
# # plot CCM state space and the the scatter plot in three dimension
# plot_ccm(x) # Pax lagged state space looks pretty random
#
# plot_ccm(log10(y)) #Pax lagged state space looks pretty random
#
# plot_ccm(z) #Pax lagged state space looks pretty random
```

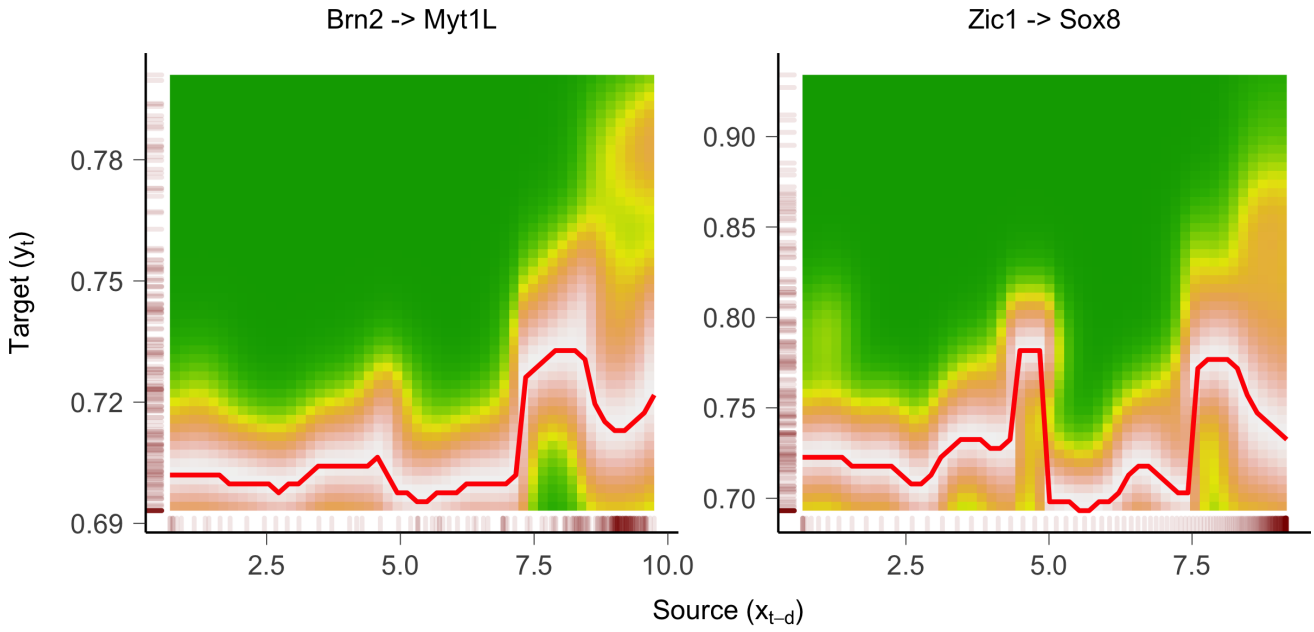
In order to intuitively visualize the relationship between two genes, we extend the DREVI approach, originally proposed by Smita Krishnaswamy, et. al. In the original approach, DREVI doesn't consider time delay, the **plot_rdi_pairs** function in *Scribe* can be used to visualize both the original *DREVI* method as well as time-delayed *DREVI* results. From the positive and negative gene pairs, we can see the response curve for the interacting gene pairs are pretty smooth while the non-interacting genes looks pretty noisy and less smooth. Moreover, from the *DREVI* plot for the non-interacting gene pair, a give target value can have many more corresponding source values and that the response curve is pretty much flat.

Moreover, in order to better represent the RDI values, we also implemented a generalized approach to consider the effect of Y at earlier time point. We achieve this first by fitting a curve between Y_t, Y_{t-1} and then obtain the residual for this fitting. We then plot these residual and $X_{t-\sigma}$ time-delayed DREVI to visualize the gene regulation.

```
# show the drevi plot result for all existing edges:
plot_rdi_pairs(neuron_sim_cds[, 1:200], gene_pairs_mat = as.matrix(neuron_network[c(1, 3), 1:2]), d = 1,
  monocle:::monocle_theme_opts() + xacHelper::nm_theme()
```



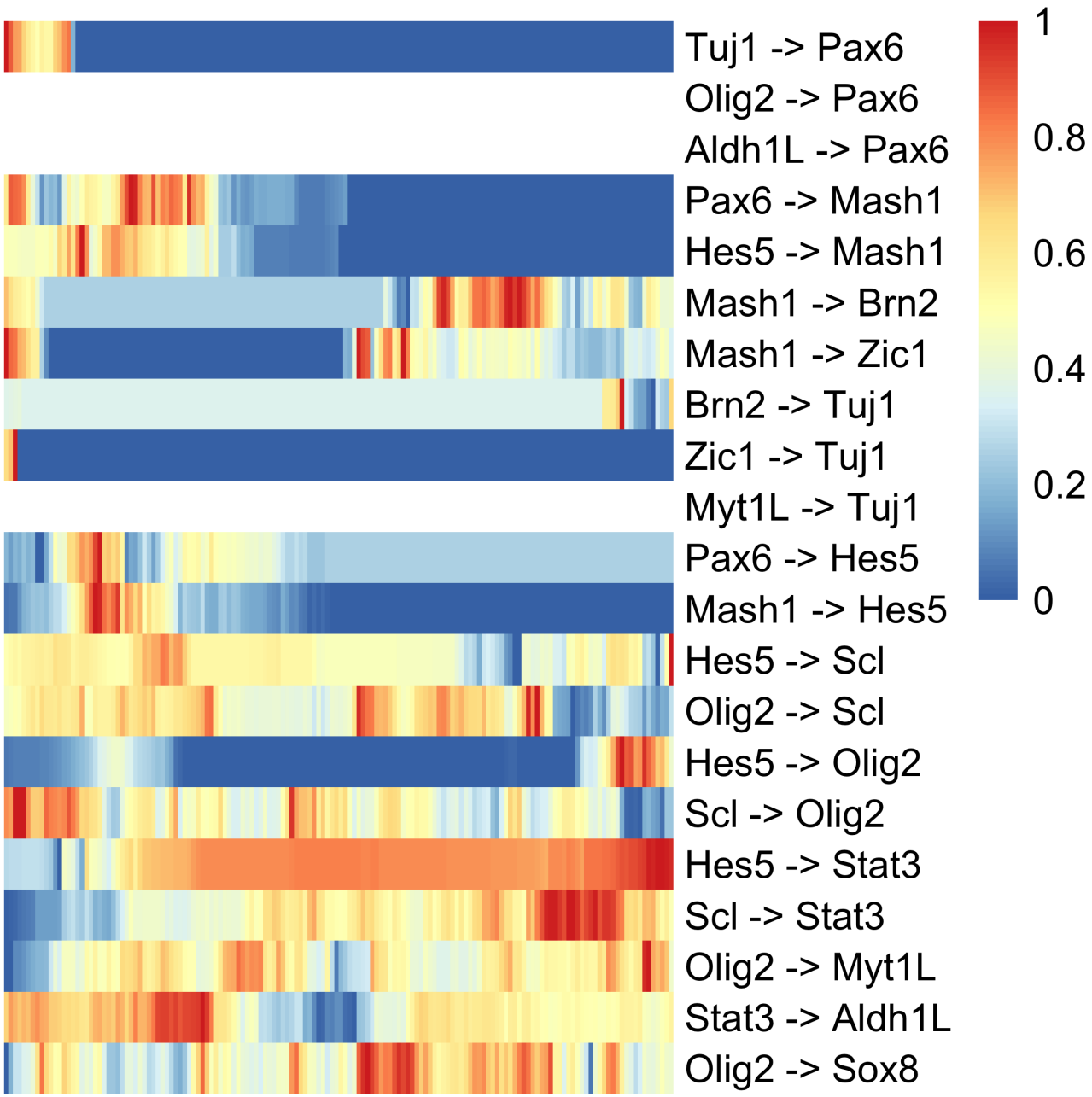

```
plot_rdi_pairs(neuron_sim_cds[, 1:200], gene_pairs_mat = matrix(c('Zic1', 'Sox8', 'Brn2', 'Myt1L'), by-
row = T, nrow = 2), n_row = 1, n_col = 2, scales = 'free') +
  monocle:::monocle_theme_opts() + xacHelper::nm_theme()
```



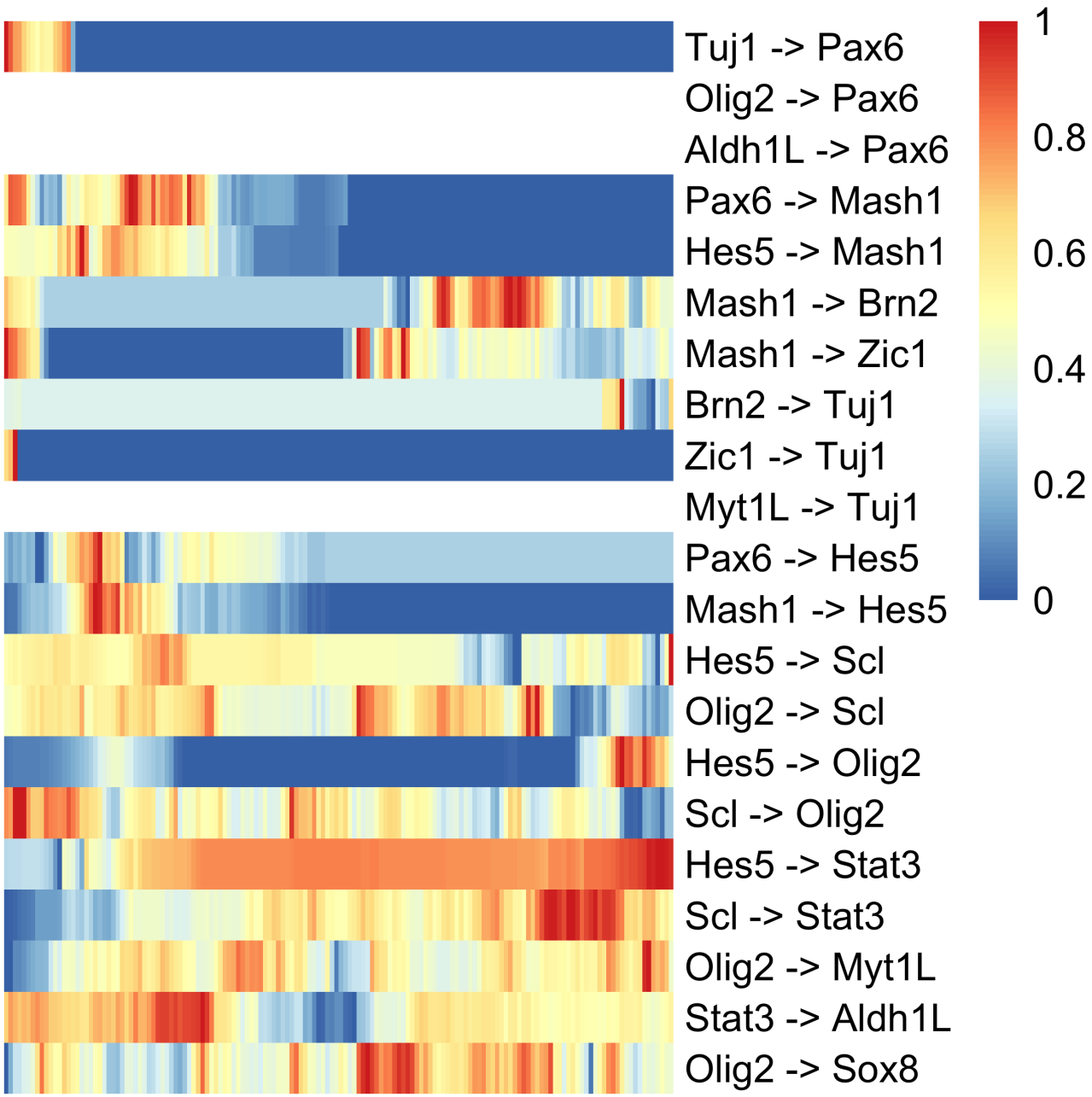
2.3 Assess temporal causal gene regulation

Gene regulation is dynamic and the timing of regulation varies from gene-pair to gene-pair. We developed a strategy (implemented through the **rdi_crdi_pseudotime** function) to estimate the temporal causal regulation in *Scribe* to helps to identify the period of strong gene regulation.

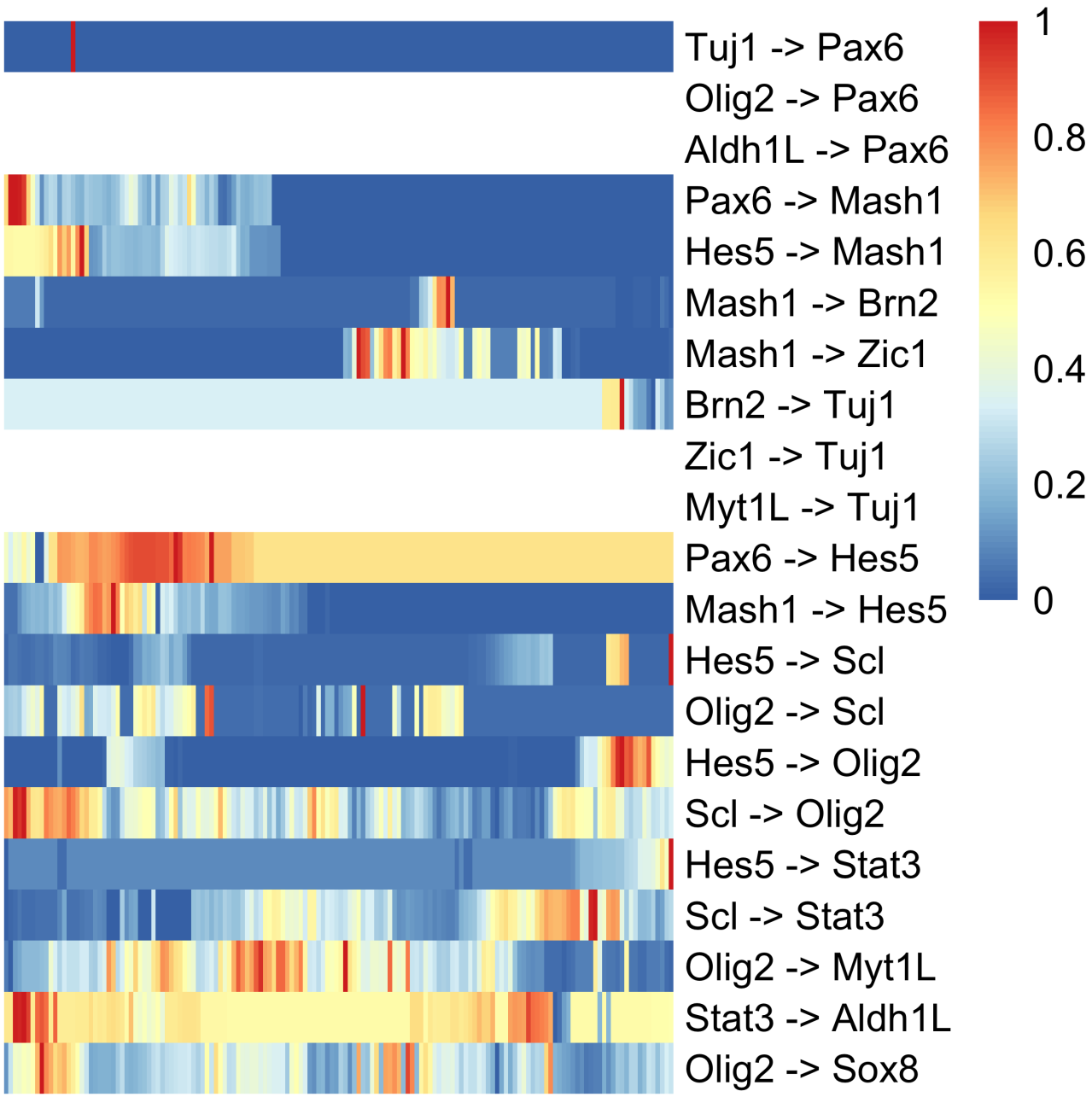
```
# test temporal RDI, etc.
gene_name_vec <- c('Pax6', 'Mash1', 'Brn2', 'Zic1', 'Tuj1', 'Hes5', 'Scl', 'Olig2', 'Stat3', 'Myt1L', 'A
ture')

rdi_crdi_pseudotime_res_list <- rdi_crdi_pseudotime(t(exprs(na_sim_cds)[1:12, 1:200]), window_size = 50)
ture gives Na values

rdi_res <- rdi_crdi_pseudotime_res_list$rdi_res
crdi_res <- rdi_crdi_pseudotime_res_list$crdi_res

dim(rdi_res) <- c(dim(rdi_res)[1], dim(rdi_res)[2] * dim(rdi_res)[2])

all_cmbns <- expand.grid(gene_name_vec[1:12], gene_name_vec[1:12])
valid_all_cmbns_df <- data.frame(pair = paste((all_cmbns$Var1), (all_cmbns$Var2), sep = ' -> '), pval =

row.names(valid_all_cmbns_df) <- valid_all_cmbns_df$pair
rdi_res <- as.data.frame(rdi_res)
colnames(rdi_res) <- valid_all_cmbns_df$pair

valid_all_cmbns_df <- data.frame(pair = paste((as.character(neuron_network[, 1])), (as.character(neuron_
> '), pval = 0)
valid_rdi_res <- rdi_res[, as.character(valid_all_cmbns_df$pair)]
norm_valid_rdi_res <- apply(valid_rdi_res, 2, function(x) (x - min(x)) / (max(x) - min(x)))
pheatmap::pheatmap(t(norm_valid_rdi_res[, ]), cluster_rows = F, cluster_cols = F, annotation_names_row =
der_color = NA)
```



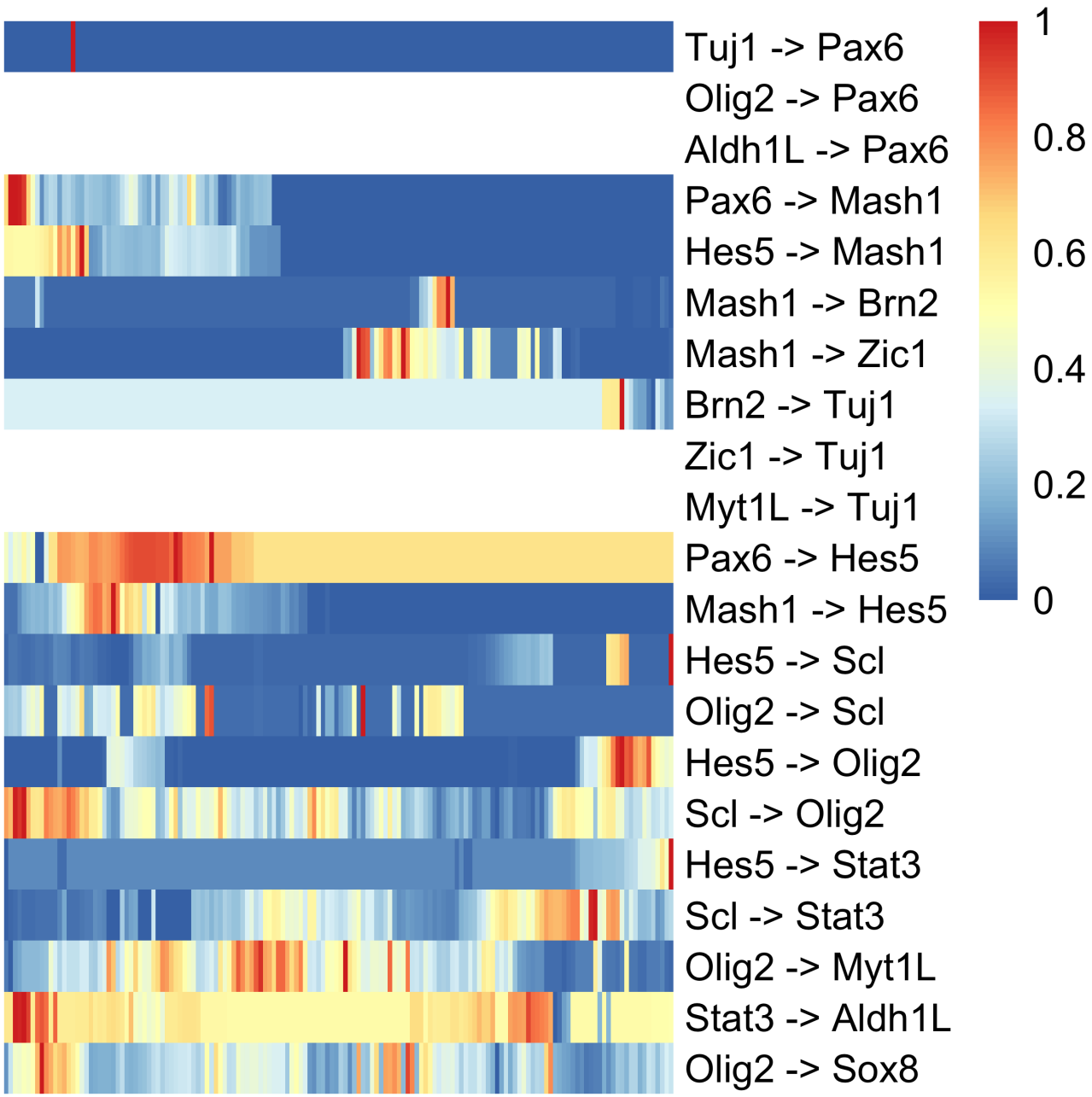

```
valid_all_cmbns_df_back <- data.frame(pair = paste((as.character(neuron_network[, 1])), (as.character(ne
> '), pval = 0)
valid_rdi_res_back <- rdi_res[, as.character(valid_all_cmbns_df_back$pair)]
norm_valid_rdi_res_back <- apply(valid_rdi_res_back, 2, function(x) (x - min(x)) / (max(x) - min(x)))
pheatmap::pheatmap(t(norm_valid_rdi_res_back[, ]), cluster_rows = F, cluster_cols = F, annota-
tion_names_row = T, border_color = NA)
```



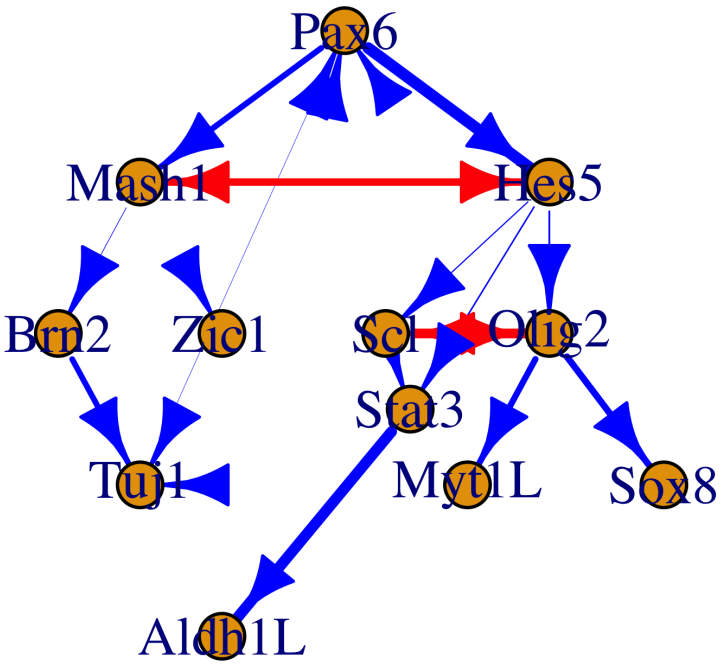

```
# plot crdi_res
dim(crdi_res) <- c(dim(crdi_res)[1], dim(crdi_res)[2] * dim(crdi_res)[2])

valid_all_cmbns_df <- data.frame(pair = paste((all_cmbns$Var1), (all_cmbns$Var2), sep = ' -> '), pval =
colnames(crdi_res) <- valid_all_cmbns_df$pair

valid_all_cmbns_df <- data.frame(pair = paste((as.character(neuron_network[, 1])), (as.character(neuron_
> '), pval = 0)
valid_crdi_res <- crdi_res[, as.character(valid_all_cmbns_df$pair)]
norm_valid_crdi_res <- apply(valid_crdi_res, 2, function(x) (x - min(x)) / (max(x) - min(x)))
# norm_valid_crdi_res_ordered <- norm_valid_crdi_res[, order(unlist(apply(norm_valid_crdi_res, 2, which.
pheatmap::pheatmap(t(norm_valid_crdi_res[, ]), cluster_rows = F, cluster_cols = F, annotation_names_row
der_color = NA)
```



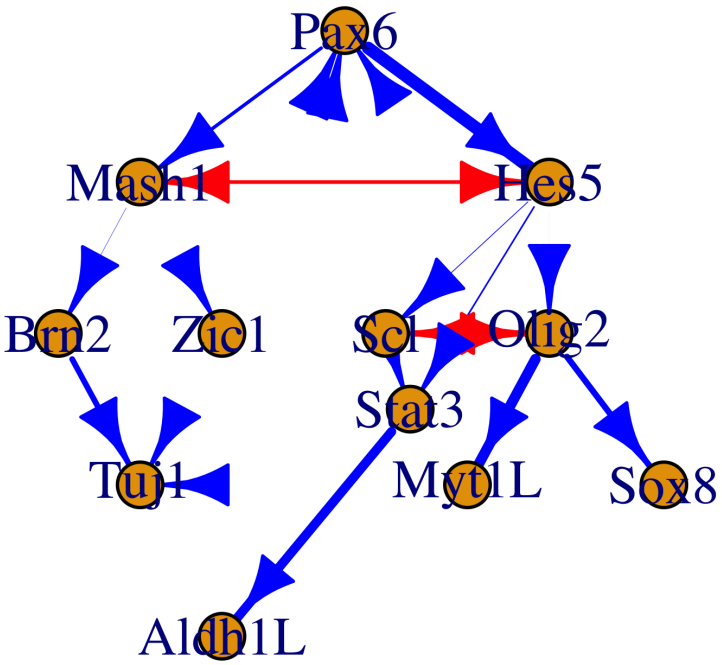

```
pheatmap::pheatmap(t(norm_valid_crdi_res[, ]), cluster_rows = F, cluster_cols = F, annotation_names_row
der_color = NA)
```



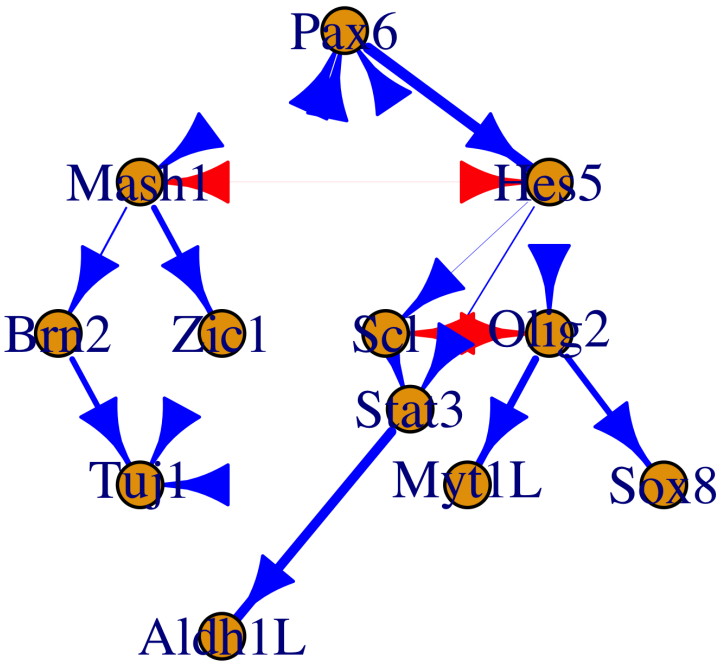

```
neuron_net <- graph_from_edgelist(as.matrix(neuron_network[, 1:2]), directed = T)

layout_coord <- layout_as_tree(neuron_net)
row.names(layout_coord) <- V(neuron_net)$name
layout_coord["Tuj1", ] <- c(-1.25,  0)

layout_coord["Scl", ] <- layout_coord["Olig2", ]
layout_coord["Olig2", ] <- c(1.25, 1.00)
layout_coord["Stat3", ] <- c(0.4, 0.5)
layout_coord["Aldh1L", ] <- c(-0.75, -1)
layout_coord["Myt1L", ] <- layout_coord["Sox8", ]
layout_coord["Sox8", ] <- c(1.95, 0)

color = rep('blue', 21)
color[c(5, 12, 14, 16)] <- 'red'

res <- apply(norm_valid_crdi_res, 2, function(x) c(mean(x[1:37]), mean(x[38:64]), mean(x[65:111]), mean(
plot(neuron_net, layout = layout_coord, edge.width=res[1, ] * 5, edge.color = color)
```



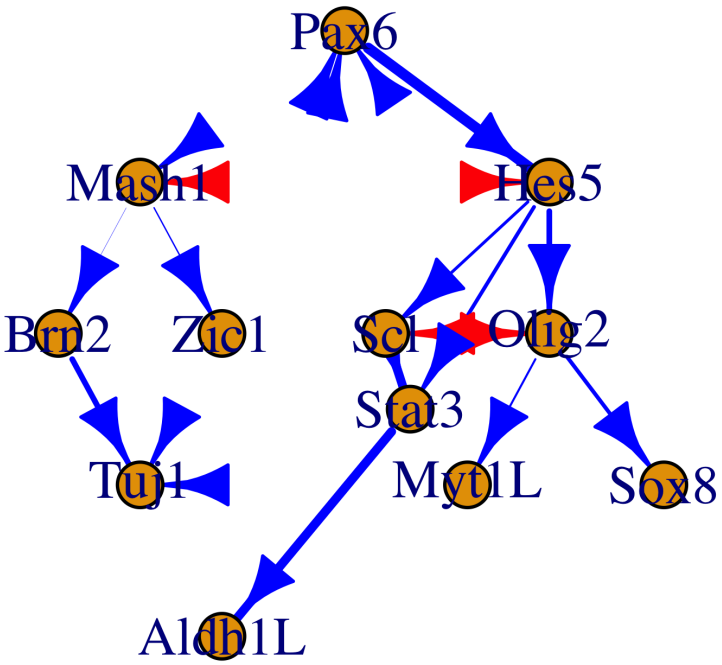

```
plot(neuron_net, layout = layout_coord, edge.width=res[2, ] * 5, edge.color = color)
```




```
plot(neuron_net, layout = layout_coord, edge.width=res[3, ] * 5, edge.color = color)
```



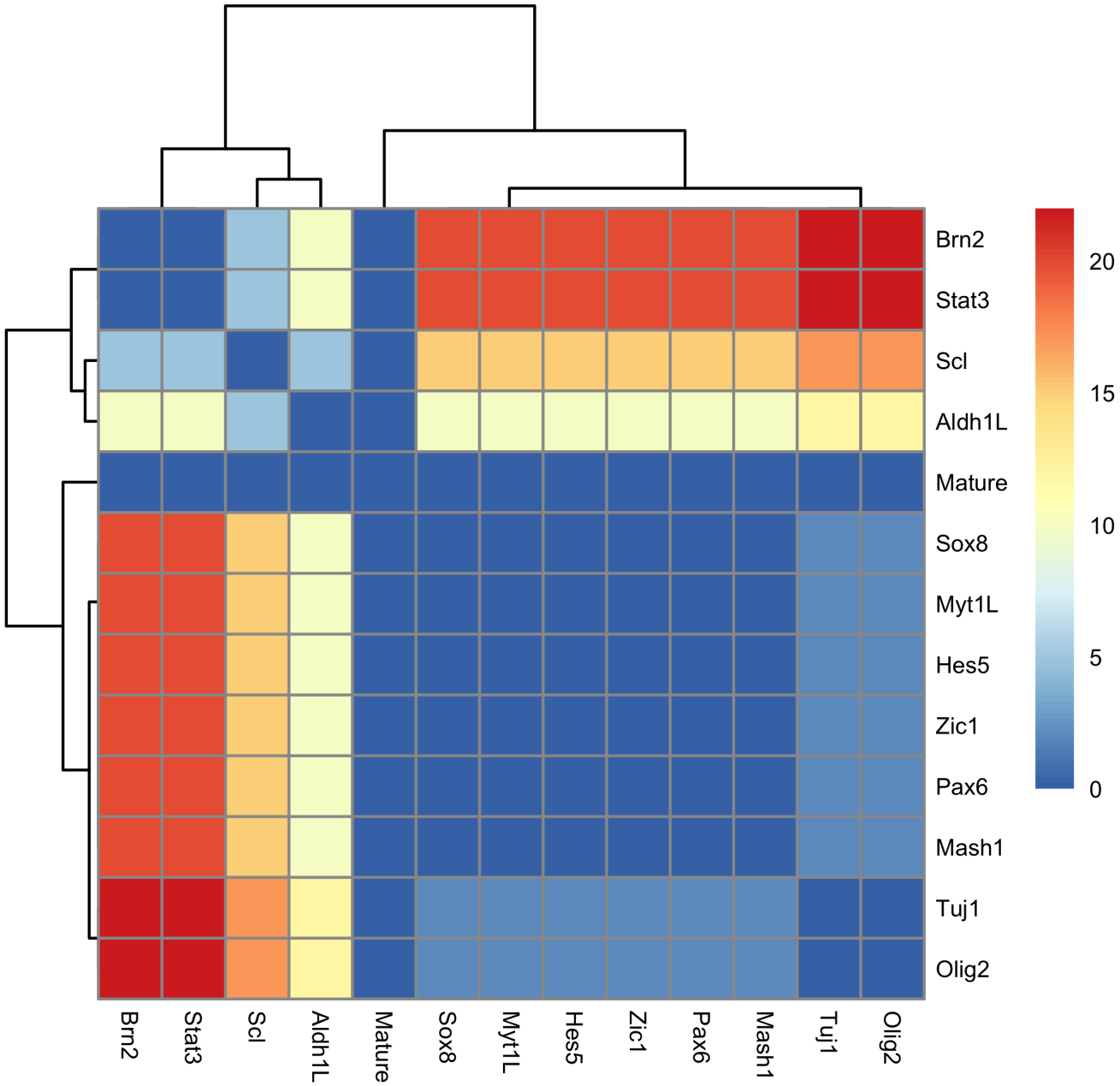

```
plot(neuron_net, layout = layout_coord, edge.width=res[4, ] * 5, edge.color = color)
```



2.4 Estimate time delay for gene pairs

An important parameter required for causal network inference in Scribe is the proper time-delay between two genes. *Scribe* provides two empirical approaches to estimate this parameter: for the linear trajectory, *Scribe* identifies the inflection point for each gene and the difference between this two genes is used as the time delay; for a branch trajectory, *Scribe* identifies the branch time point for each gene and the difference between this two genes is used as the time delay. **estimate_turning_point** is the function to estimate the time delay between genes while the **plot_time_delay_heatmap** can be used to visualize the time delay between any pairs of genes from the cds.

```
na_sim_cds <- estimate_turning_point(na_sim_cds)
plot_time_delay_heatmap(na_sim_cds, use_gene_short_name = F)

## There is NA values in turining points calculated, the time delay is set to 0 by default
```



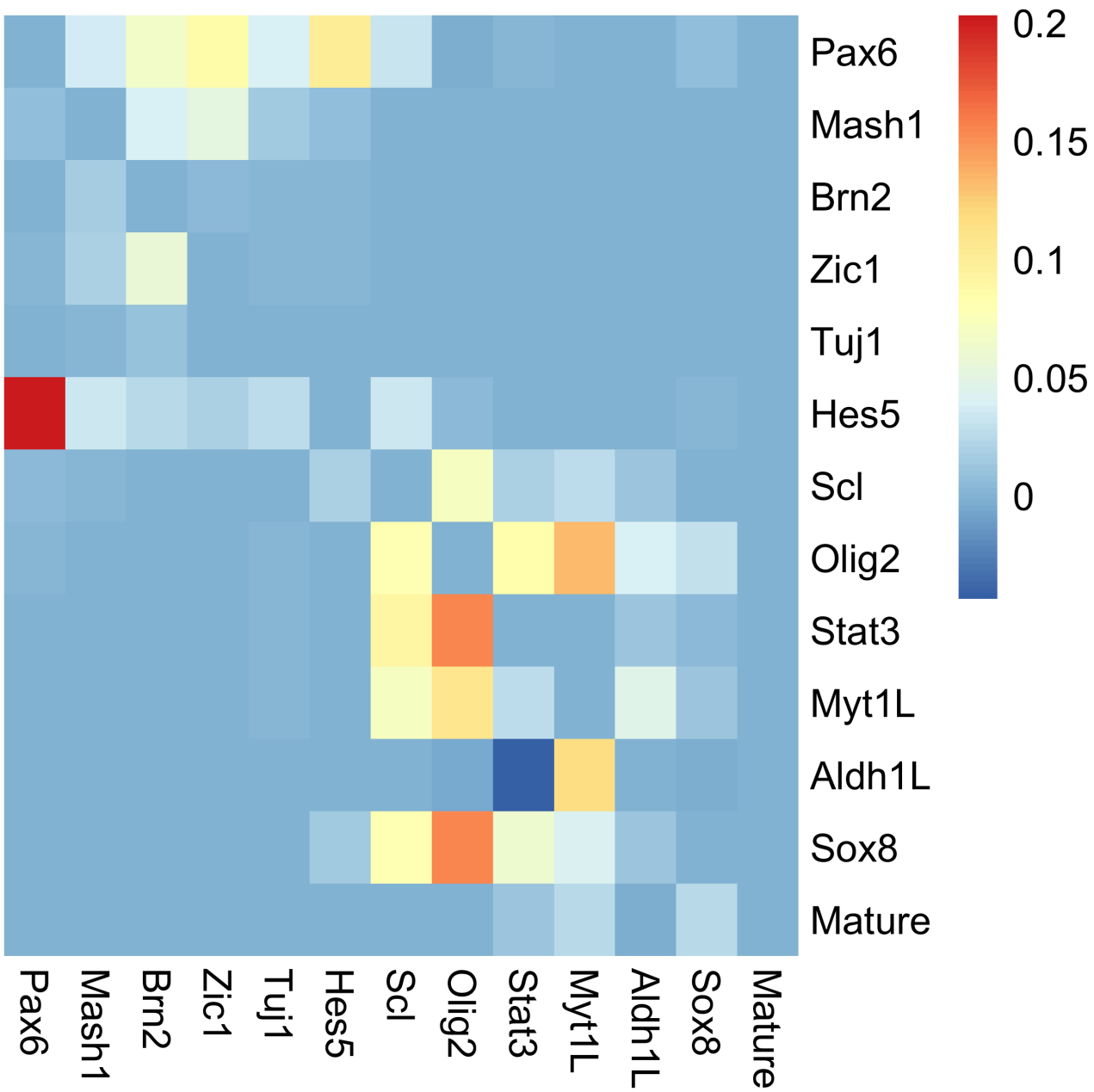
2.5 Infer and visualize gene regulatory network

We can infer the gene regulatory network through **calculate_rdi_multiple_run_cpp** function. This function will return a list with the following elements: a. **RDI** (dimension is number of genes X length of delays times number of genes), vector of **delays**, **max_rdi_value** and **max_rdi_delays** (the matrices for the all possible pairs of gene regulation). After we run **calculate_rdi_multiple_run_cpp**, we can further calculate the conditional RDI values between all possible pair of genes by conditioning the top highest incoming node(s) based on the RDI results. *Scribe* can concatenate multiple directly related trajectory (for example different lineage commitment from the same developmental trajectory) for RDI calculation (that is why the term "multiple_run" appears in the function). *Scribe* will also provide a procedure to sparsify the RDI/cRDI matrix to ensure the distribution of indegree of the network follows the exponential distribution while the distribution of outdegree follows the power-law distribution (this part is still under development).

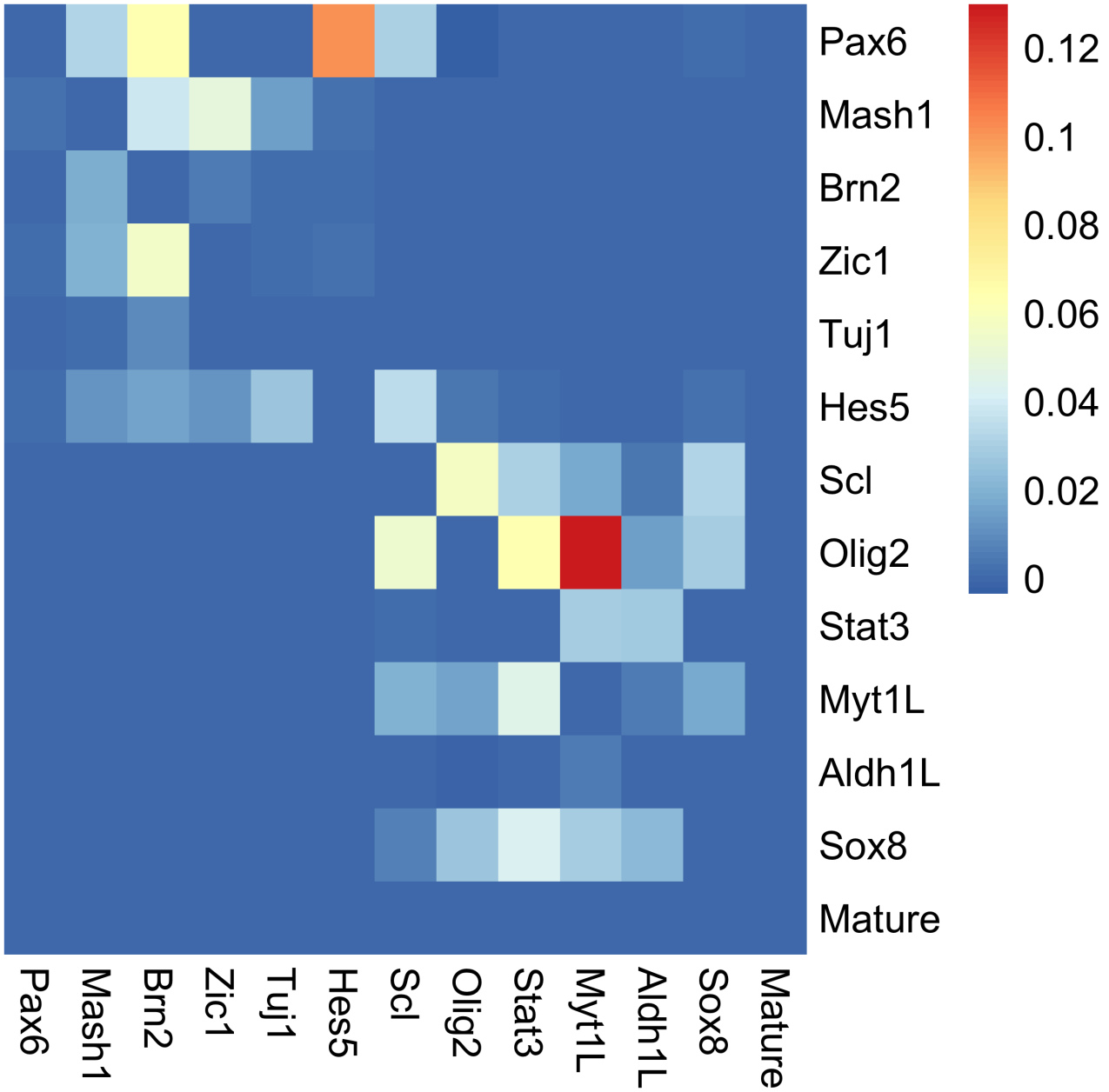
In *Scribe*, we provided a variety of approaches to visualize the RDI network we retrieved in the end. This can be done either by using the *heatmap*, *igraph* plotting function, the *hiearical* plot (based on **level.plot** in *netbiov* package), the arc diagram plot as well as the *Hive* plot.

```
data <- t(exprs(neuron_sim_cds)[, 1:200]) # prepare the data (row is sample, column is gene)
noise = matrix(rnorm(mean = 0, sd = 1e-10, nrow(data) * ncol(data)), nrow = nrow(data))
```

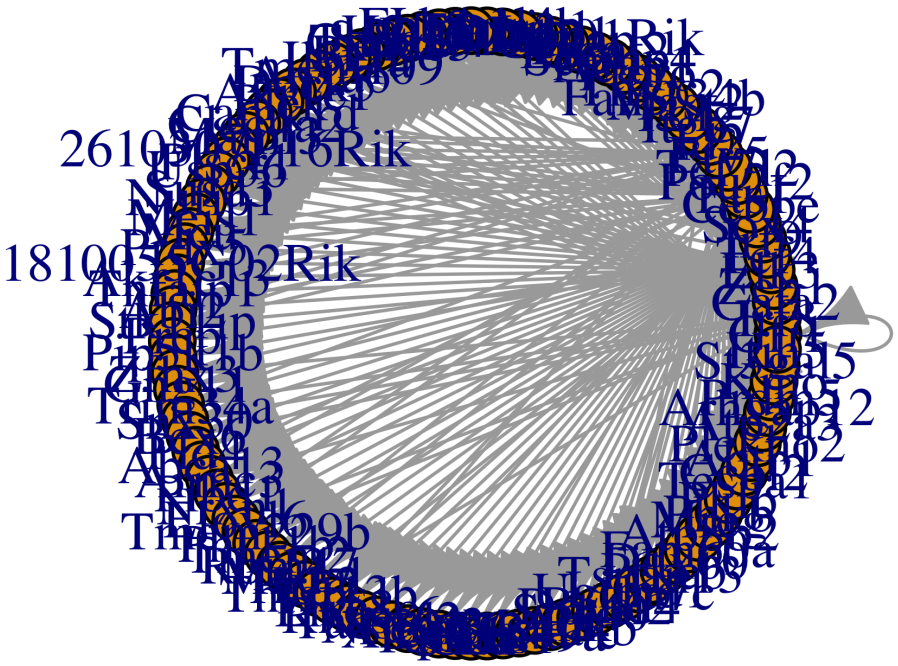
```
run_vec <- rep(1, nrow(data)) # run information for each cell

# create the network graph we want to estimate RDI (here we calculate all possible pair of gene reg-
ulation)
tmp <- expand.grid(1:ncol(data), 1:ncol(data), stringsAsFactors = F)
super_graph <- tmp[tmp[, 1] != tmp[, 2], ] - 1 #
super_graph <- super_graph[, c(2, 1)]

rdi_list <- calculate_rdi_multiple_run_cpp(data + noise, delay = c(1), run_vec - 1, as.matrix(super_grap
ing_points = 0) #* 100 + noise

dimnames(rdi_list$max_rdi_value) <- list(gene_name_vec, gene_name_vec)
con_rdi_res_test <- calculate_multiple_run_conditioned_rdi_wrap(data + noise, as.matrix(super_graph), as
 1, 1)
dimnames(con_rdi_res_test) <- list(gene_name_vec, gene_name_vec)

# visualize the network by heatmap
pheatmap::pheatmap(rdi_list$max_rdi_value, cluster_rows = F, cluster_cols = F, annotation_names_col = T,
der_color = NA)
```



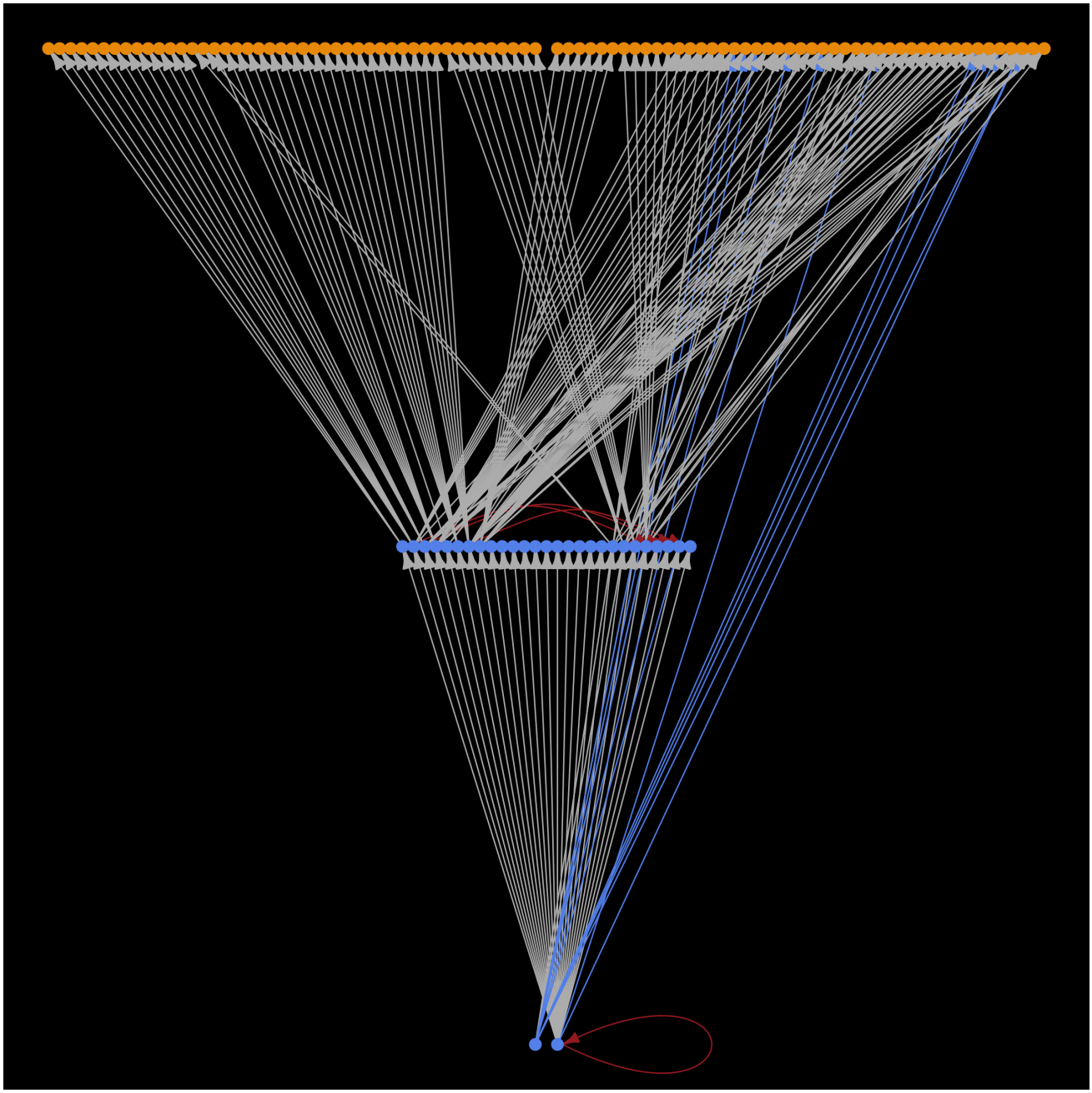

```
pheatmap::pheatmap(con_rdi_res_test, cluster_rows = F, cluster_cols = F, annotation_names_col = T, bor-
der_color = NA)
```



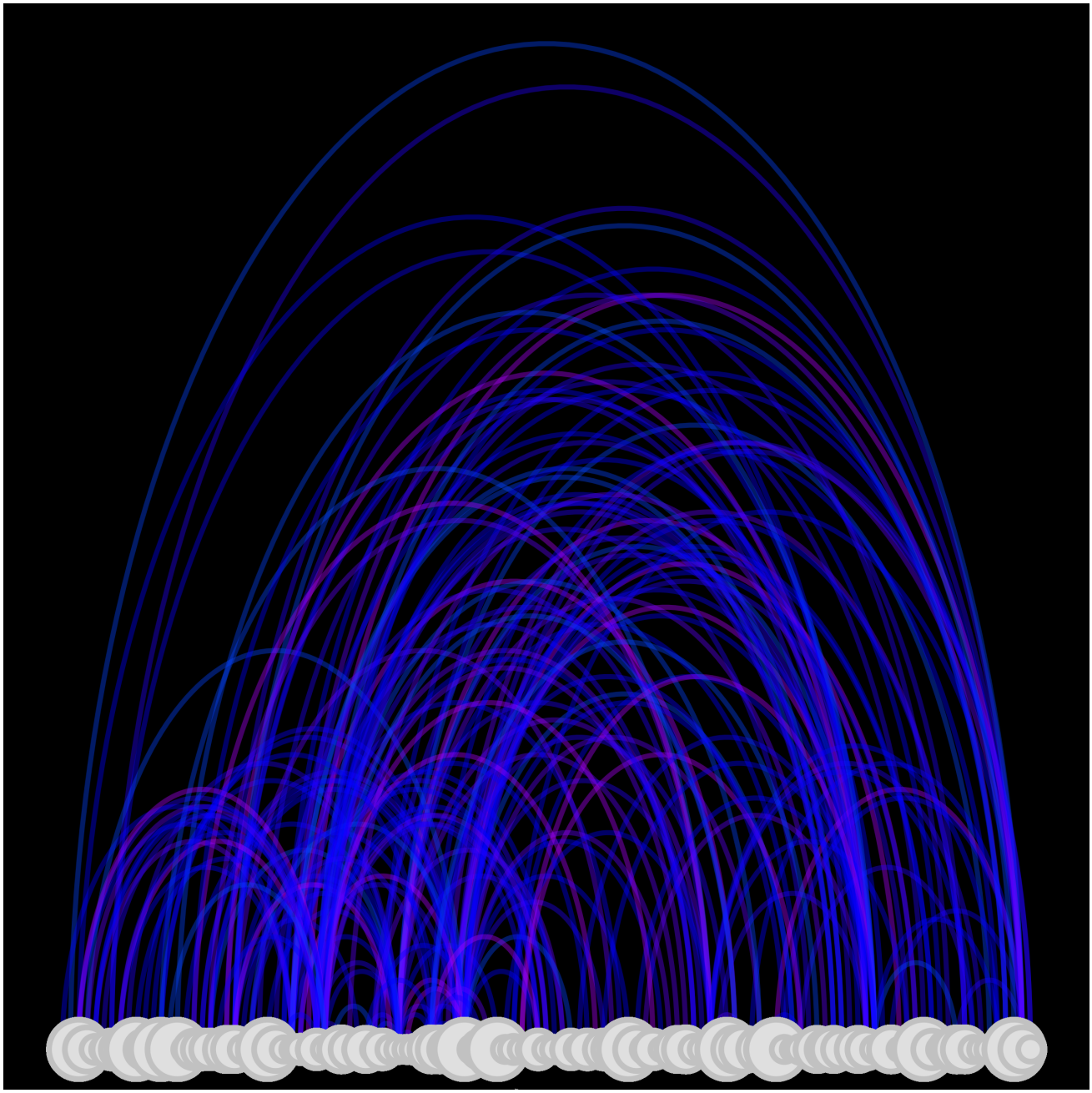

```
# in the following, I will show how to visualize a large scale network
# load the network:
load("/Users/xqiu/Dropbox (Personal)/Projects/Monocle2_revision/RData/res") #load the res (we need to ma
nually set the initial_nodes 1:2 in .process_graph1 function)

plot_network(res$g, type = 'igraph', layout = layout_in_circle)
```



```
# plot_network(res$g, type = 'hiearchy')
plot.netbiov(res)
```



```
plot_network(res$g, type = 'arcdiagram')
```



3 The Scribe object

The *monocle* package takes a matrix of gene expression values as calculated by Cufflinks [**?**] or another gene expression estimation program. Monocle can work with relative expression values (e.g. FPKM or TPM units) or absolute transcript counts (e.g. from UMI experiments). Monocle also works "out-of-the-box" with the transcript count matrices produced by CellRanger, the software pipeline for analyzing experiments from the 10X Genomics Chromium instrument. Monocle also works well with data from other RNA-Seq workflows such as sci-RNA-Seq and instruments like the Biorad ddSEQ. Although Monocle can be used with raw read counts, these are not directly proportional to expression values unless you normalize them by length, so some Monocle functions could produce nonsense results. If you don't have UMI counts, We recommend you load up FPKM or TPM values instead of raw read counts.

3.1 The Scribe class: need to discuss

Scribe holds single cell expression data in objects of the *CellDataSet* class. The class is derived from the Bioconductor *ExpressionSet* class, which provides a common interface familiar to those who have analyzed microarray experiments with Bioconductor. The class requires three input files:

1. `exprs`, a numeric matrix of expression values, where rows are genes, and columns are cells

2. `phenoData`, an *AnnotatedDataFrame* object, where rows are cells, and columns are cell attributes (such as cell type, culture condition, day captured, etc.)

3. `featureData`, an *AnnotatedDataFrame* object, where rows are features (e.g. genes), and columns are gene attributes, such as biotype, gc content, etc.

The expression value matrix *must* have the same number of columns as the `phenoData` has rows, and it must have the same number of rows as the `featureData` data frame has rows. Row names of the `phenoData` object should match the column names of the expression matrix. Row names of the `featureData` object should match row names of the expression matrix. Also, one of the columns of the `featureData` must be named "gene_short_name".

You can create a new *CellDataSet* object as follows:

This will create a CellDataSet object with expression values measured in FPKM, a measure of relative expression reported by Cufflinks. By default, Monocle assumes that your expression data is in units of transcript counts and uses a negative binomial model to test for differential expression in downstream steps. However, if you're using relative expression values such as TPM or FPKM data, see below for how to tell Monocle how to model it in downstream steps.

NOTE: if you do have UMI data, you should *not* normalize it yourself prior to creating your `CellDataSet`. You should also *not* try to convert the UMI counts to relative abundances (by converting it to FPKM/TPM data). You should *not* use `relative2abs()` as discussed below in section **??**. Monocle will do all needed normalization steps internally. Normalizing it yourself risks breaking some of Monocle's key steps.

4 Relationship between Scribe and Monocle 2 as well as other relevant packages

Scribe relies on *Monocle 2* for reconstructing the single-cell trajectory before inferring causal network. We extended *Monocle 2* in a few aspects to account for the specific requirement for *Scribe*. For example, in order to avoid the over-crowded nature of the cells at the terminal regions of the trajectory, we applied a simple strategy to first run Monocle 2 using DDRTree to get the trajectory with complex branches and then reorder the trajectory for each branch (from the source state to the terminal state) using principal curve (simplePPT) with the low dimension space obtained from the diffusion map dimension reduction technique. *Scribe* provides various methods for infering the regulatory relationship, mutual information, Granger causality and CCM. We used the implementation from parmigene, vars and rEDM respectively. To intuitively visualize the casual regulation between pairs of genes, we generalized the DREVI approach ([]). The network visualization relies on the igraph, netbiov, arcdiagram as well as the HiveR packages.

We are preparing a manuscript describing *Scribe* for reconstructing causal regulatory network.

5 Theory behind Scribe

5.1 Restricted direct information

A very well-known measure of interdependence of two random variables X and Y was first proposed by [cite] called mutual information, is defined as:

$$I(X;Y) = \sum_{x,y} p_{XY}(x,y) \log \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)} \tag{1}$$

The mutual information between X and $Y$is equal to zero if and only if X and Y are independent. The mutual information of two variables given a third random variable Z is defined as:

$$I(X;Y|Z) = \sum_{x,y,z} p_{XYZ}(x,y,z) \log \frac{p_{XY|Z}(x,y|z)}{p_{X|Z}(x|z)p_{Y|Z}(y|z)} \tag{2}$$

Similarly, the mutual information between X and Y given Z is equal to zero if and only if X and Y are independent given Z. In general, X, Y and Z can be multi-dimensional random variables with arbitrary dimensions, such as in the case that the variables are time-series $\underline{X}^t$, $\underline{Y}^t$ and $\underline{Z}^t$. So the mutual information between the time-series can be defined similarly as $I(\underline{X}^t, \underline{Y}^t, \underline{Z}^t)$. Now the question we might ask ourselves is is it possible to define a measure for the amount of information the past state(s) of X provide about the current state of the variable Z denoted by $Z(t-1)$? [cite: DI paper] provides such a measure called Directed Information, sometimes referred to as "Transfer Entropy" [cite: T. Schreiber, Measuring information transfer," Physical review letters, vol. 85, no. 2, p. 461, 2000]. The directed information from the time-series $\underline{X}^T$ to $\underline{Y}^T$ is defined as:

$$DI(X \to Y) = \sum_{t=1}^{T} I(\underline{X}^{t-1}; Y(t) | \underline{Y}^{t-1}) \tag{3}$$

The conditioned version of it also can be defined similarly:

$$DI(X \to Y | Z) = \sum_{t=1}^{T} I(\underline{X}^{t-11}; Y(t) | \underline{Y}^{t-1}, \underline{Z}^{t-1}) \tag{4}$$

(Mention the theorem for DI) [Cite DI paper]

In [Cite DI paper] it is shown that if the system is not purely deterministic, then the directed information graph (DIG) and the causality graph (CG) are identical. However, they have shown that if the relationship in the dynamic system is purely deterministic, then the algorithm is unable to return the correct causality graph. It is contrary to CCM method which only deals with the deterministic systems and is unable to perceive the graph in a stochastic (noisy) system. Thus in the case of the first-order Markov systems we introduce an alternate version of DI which employs only the immediate past of the system instead of the whole past samples:

$$DI(X \to Y) = I\left(X(t-1); Y(t) | Y(t-1)\right) \tag{5}$$

and the conditioned version of it can be expressed as:

$$DI(X \to Y | Z) = I\left(X(t-1); Y(t) | Y(t-1), Z(t-1)\right) \tag{6}$$

In [Cite: Our Allerton paper] it's shown that this method works in many stochastic or deterministic cases and under some mild assumptions it's capable of inferring the correct graph.

5.2 Network sparsifier

6 Citation

?? If you use *Scribe* to analyze your experiments, please cite:

```
citation("Scribe")

##
## To cite package 'Scribe' in publications use:
##
##   ANN Library: David Mount, Sunil Arya. flann Packge:
##   http://www.cs.ubc.ca/research/flann/. InformationEstimator:
##   Xiaojie Qiu, Arman Rahimzamani, Sreeram Kannan and Cole Trapnell
##   (2017). Scribe: Restricted direction information for network
##   inference. R package version 0.1.
##   https://CRAN.R-project.org/package=Scribe
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {Scribe: Restricted direction information for network inference},
##     author = {ANN Library: David Mount and Sunil Arya. flann Packge: http://www.cs.ubc.ca/research/fl
##     year = {2017},
##     note = {R package version 0.1},
##     url = {https://CRAN.R-project.org/package=Scribe},
##   }
##
## ATTENTION: This citation information has been auto-generated from
## the package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

7 Acknowledgements

Scribe was written by Xiaojie Qiu which is based on the python package originally developed by Arman Rahimzamani, with substantial design input Cole Trapnell and Sreeram Kannan. The core of Scribe on estimating restricted direction information (RDI) is written in Rcpp for speed accerlationp. We are grateful to all members from Trapnell and Kannan lab, especially XXX for technical assistance, and Andysheh Mohajeri, Jonanthan Packer, XXX etc. for helpful discussions. This work was supported by US National Institutes of Health (NIH) grants DP2 HD088158 (C.T.) and U54 DK107979 (C.T.), XXXXXX; C.T. is partly supported by a Dale. F. Frey Award for Breakthrough Scientists and an Alfred P. Sloan Foundation Research Fellowship.

This vignette was created from Wolfgang Huber's Bioconductor vignette style document, and patterned after the vignette for *DESeq*, by Simon Anders and Wolfgang Huber.

8 Session Info

```
sessionInfo()

## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X Yosemite 10.10.5
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
##  [1] splines   stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] plyr_1.8.4          Scribe_0.1          arcdiagram_0.1.11
##  [4] netbiov_1.8.0       igraph_1.1.2        shiny_1.0.3
##  [7] plotly_4.6.0        inflection_1.3      vars_1.5-2
## [10] lmtest_0.9-35       urca_1.3-0          strucchange_1.5-1
## [13] sandwich_2.3-4      zoo_1.8-0           MASS_7.3-47
## [16] rEDM_0.5.4          monocle_2.5.3       DDRTree_0.1.5
## [19] irlba_2.2.1         VGAM_1.0-3          Matrix_1.2-10
## [22] ggplot2_2.2.1       reshape2_1.4.2      Biobase_2.34.0
## [25] BiocGenerics_0.20.0 knitr_1.15.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.12           lattice_0.20-35        tidyr_0.6.3
##  [4] assertthat_0.2.0       digest_0.6.12          mime_0.5
##  [7] slam_0.1-40            R6_2.2.1               qlcMatrix_0.9.5
## [10] evaluate_0.10          httr_1.2.1             highr_0.6
## [13] rlang_0.1.1            lazyeval_0.2.0         combinat_0.0-8
## [16] labeling_0.3           Rtsne_0.13             stringr_1.2.0
## [19] htmlwidgets_0.8        pheatmap_1.0.8         munsell_0.4.3
## [22] httpuv_1.3.3           pkgconfig_2.0.1        htmltools_0.3.6
## [25] tibble_1.3.1           xacHelper_0.0.1.0000   codetools_0.2-15
## [28] matrixStats_0.52.2     viridisLite_0.2.0      dplyr_0.5.0
## [31] grid_3.3.2             densityClust_0.2.1     xtable_1.8-2
## [34] nlme_3.1-131           jsonlite_1.4           gtable_0.2.0
## [37] DBI_0.6-1              magrittr_1.5           scales_0.4.1
## [40] stringi_1.1.5          limma_3.30.13          fastICA_1.2-0
## [43] RColorBrewer_1.1-2     tools_3.3.2            purrr_0.2.2.2
## [46] HSMMSingleCell_0.108.0 colorspace_1.3-2       cluster_2.0.6
```

References