

# NETISCE Manual and Tutorials

Lauren Marazzi

2021-12-08



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>About</b>                                      | <b>5</b>  |
| <b>2</b> | <b>Installation and Usage</b>                     | <b>7</b>  |
| 2.1      | Download NETISCE . . . . .                        | 7         |
| 2.2      | Install Nextflow . . . . .                        | 7         |
| 2.3      | Docker Image . . . . .                            | 7         |
| 2.4      | Prerequisites . . . . .                           | 7         |
| 2.5      | Parameters and Configuration . . . . .            | 8         |
| 2.6      | Running NETISCE . . . . .                         | 9         |
| <b>3</b> | <b>NETSICE output</b>                             | <b>11</b> |
| <b>4</b> | <b>Toy Network Examples</b>                       | <b>13</b> |
| 4.1      | Overview . . . . .                                | 13        |
| 4.2      | Data . . . . .                                    | 14        |
| 4.3      | NETISCE run configuration . . . . .               | 14        |
| 4.4      | Run NETSICE . . . . .                             | 15        |
| 4.5      | NETSICE Results . . . . .                         | 15        |
| 4.6      | Toy Example with mutations . . . . .              | 20        |
| <b>5</b> | <b>Cell Fate Specification in Ascidian Embryo</b> | <b>23</b> |
| 5.1      | Input Data . . . . .                              | 23        |
| 5.2      | Run the simulation . . . . .                      | 24        |
| 5.3      | Results . . . . .                                 | 24        |
| 5.4      | Visualizing Results . . . . .                     | 25        |

|          |   |           |
|----------|---|-----------|
| <b>6</b> | <b>Pluripotent Stem Cell Example</b>                    | <b>27</b> |
| <b>7</b> | <b>Adaptive Resistance in Colorectal Cancer Example</b> | <b>29</b> |
| <b>8</b> | <b>NETISCE in Galaxy Project</b>                        | <b>31</b> |

# Chapter 1

## About

Welcome to the NETISCE manual and tutorials.

NETISCE is a network-based approach for cellular reprogramming. This manual contains instructions for installing the NETISCE pipeline tool and accessing the Galaxy Project web-based tool. We provide a simple toy example walk-through tutorial. Lastly, we include instructions for reproducing NETISCE cell reprogramming results in developmental, stem cell, and cancer biology.

NETISCE identifies combinations of perturbations to be applied on a gene regulatory or signaling network to trigger a shift from an undesired to a desired cell fate. The core of the pipeline is the application of structure-based control theory to identify control nodes that drive the system from an initial state that would lead to an attractor associated with an undesired phenotype and towards an attractor associated with the desired phenotype. For more information, please see the accompanying paper: **[link here](#)**



## Chapter 2

# Installation and Usage

### 2.1 Download NETISCE

NETISCE pipelines can be downloaded from our github repository: <https://github.com/veraliconaresearchgroup/netisce>

We recommend that you run NETISCE on a high-performance cluster (hpc), as you may generate files that are quite large, or run computations that may take a long time. However, we provide two Nextflow pipelines, one designed for hpcs (NETISCE\_hpc), and another for running NETISCE on a local machine (NETISCE\_local).

### 2.2 Install Nextflow

Nextflow is required to run the NETISCE pipeline. Please follow the instructions from <https://www.nextflow.io/> (see ‘Getting Started’ steps 1 & 2) to install Nextflow in the appropriate NETSICE folder (\_\_local or \_\_hpc).

### 2.3 Docker Image

*docker image provided here TBD*

### 2.4 Prerequisites

If you are not using the Docker image, the following packages will need to be installed:

- scipy
- pandas
- sklearn
- yellowbrick

## 2.5 Parameters and Configuration

Whether on your local machine or hpc, to run NETISCE you must specify the files and parameters within the `.nf` file

- **params.expressions**: csv file containing normalized expression data for network nodes in different samples
- **params.network**: network file (sif format)
- **params.samples**: text file specifying the phenotype for each sample in params.expressions file (tab delimited)
- **params.internal\_control**: text file containing a list of nodes to be used as internal marker nodes
- **params.alpha**: alpha parameter for signal flow analysis (default =0.9)
- **params.undesired**: string of the undesired phenotype (as labeled in the params.samples file)
- **params.desired**: string of the desired phenotype (as labeled in the params.samples file)
- **params.filter**: filtering parameter for criterion 2 (“strict” or “relaxed”)
- **params.kmeans\_min\_val**: minimum k-means value for clustering (default=2)
- **params.kmeans\_max\_val**: maximum k-means value for clustering (default=10)
- **params.num\_nodes**: number of nodes in network for which normalized expression data exists (within the params.expressions file)
- **params.num\_states**: number of randomly generated initial states (default=100000, or  $3^n$  where n is the number of network nodes and  $3^n$  is less than 100000)

Please see the **input\_data** folder for examples of files to match the formatting.

### 2.5.1 NETISCE\_mutations.nf

If you are interested in including mutational information, please use the `NETISCE_mutations.nf` pipeline. You must additionally specify **params.mutations**: a csv file containing mutational configuration for network nodes (0 for loss of function, 1 for gain of function). Please see example in `input_data` for formatting.



### 2.5.2 nextflow.config

If you are running nextflow on an hpc, please specify your executor, and clusterOptions within the nextflow.config file. Please see <https://www.nextflow.io/docs/latest/config.html> for more information regarding your executor.

## 2.6 Running NETISCE

Once you have specified the parameters, run NETSICE using the following command:

```
./nextflow run NETISCE.nf -resume ##or NETISCE_mutations.nf if including mutational data
```

We recommend using the -resume flag in the case that you change a file or parameter within your pipeline. This way, nextflow caches results that remain unchanged, preventing pipeline steps from being re-run.



## Chapter 3

# NETSICE output

After the NETISCE computations are complete, the output files will be located in the **results** folder. Please note that we have included in this folder the most relevant output files that you may want to use for further analysis. However, you can explore all outputs by checking within each step of the pipeline's work folder.

The contents of each file are briefly described below. For more details and to see example outputs, please see the Toy Network Examples.

### **exp\_\_internalmarkers.txt**

This file contains the resultant steady state values for the internal marker nodes for the provided experimental samples (those specified in `samples.txt`) from Signal Flow Analysis.

### **experimental\_\_internalmarkers.pdf**

This pdf is a figure of the steady state values for the internal marker nodes for the provided experimental samples. This can be used to verify the validity of the internal marker nodes.

### **elbow.png**

A graph of the elbow metric for determining the optimal k for k-means.

### **silhouette.pdf**

A graph of the silhouette metric for determining the optimal k for k-means.

**fvs.txt**

This file contains the node names for the FVS used as control nodes for the NETISCE run.

**crit1perts.txt**

This file contains a list of IDs for the control node perturbations that passed criterion 1.

**pert1\_internal\_markers.txt**

This file contains a table of the internal marker node steady state values from control node perturbations whose associated attractors passed the first filtering criterion.

**successful\_controlnode\_perturbations.txt**

This file contains a table of the control node perturbations that pass both the 1st and 2nd filtering criteria. it also contains the number of upregulation,downregulations, and total number of nodes perturbed for each perturbation set.

## Chapter 4

# Toy Network Examples

Here, we will walk through a brief tutorial of a NETISCE run. The files necessary to complete the tutorial are within the `input data` folder of both `NETISCE_local` and `NETISCE_hpc`. The results from these Toy examples can be found in the `toy_example_results` folder of the main github repository.

### 4.1 Overview

We will use a simple toy network of 6 nodes and 9 edges.

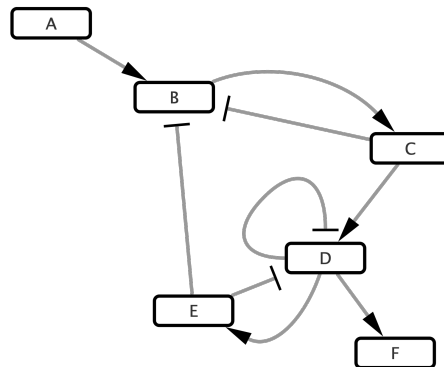


Figure 4.1: Simple Toy Network

## 4.2 Data

You can find the relevant data files in the `input_data` folder.

In this example, we have 2 samples, A and B, with three replicates each (A\_1,A\_2,A\_3, etc). The normalized expression data is housed in `expressions.csv`, and contains normalized expression values for 4 of the network nodes (in this case, node C was not found to be expressed in the samples).

| X | A_1 | A_2 | A_3 | B_1 | B_2 | B_3 |
|---|-----|-----|-----|-----|-----|-----|
| A | 2   | 1   | 2   | 6   | 7   | 6   |
| B | 6   | 7   | 6   | 2   | 1   | 1   |
| D | -1  | -1  | -2  | 6   | 7   | 6   |
| E | 2   | 1   | 2   | 6   | 7   | 6   |

The `samples.txt` file specifies that A is associated to a treatment sensitive phenotype, while B is associated to a resistance phenotype.

| name | phenotype |
|------|-----------|
| A_1  | sensitive |
| A_2  | sensitive |
| A_3  | sensitive |
| B_1  | resistant |
| B_2  | resistant |
| B_3  | resistant |

Note that you can use any term to describe the phenotypes. Just be sure to be consistent with the `param.desried` and `param.undesired` variables within the Nextflow `.nf` file.

Lastly, we need to include a list of internal marker nodes. This list is in `internal_marker.txt`. For our small network, the internal marker node is C.

|   |
|---|
| C |
|---|

## 4.3 NETISCE run configuration

With all your input data files loaded, next we configure the nextflow run. Within either `NETISCE_local` or `NETISCE_hpc` (**Note: while we do recommend you run NETISCE on a hpc, this example is small enough to run locally**).

Open up `NETISCE.nf`. Here, you need to specify the parameters for the Nextflow run on lines 3-19. Please refer to section 2.5 for parameter definitions.

For this example, your parameters should look like:

```
params.expressions = "$baseDir/input_data/expressions.csv"
params.network = "$baseDir/input_data/network.sif"
params.samples = "$baseDir/input_data/samples.txt"
params.internal_control="$baseDir/input_data/internal_marker.txt"
params.alpha = 0.9
params.undesired = 'resistant'
params.desired = 'sensitive'
params.filter = "strict"

params.kmeans_min_val = 2
params.kmeans_max_val = 10

params.num_nodes = 4 // that have expression data
params.num_states = 1000
```

**Some Notes:** make sure to include `$baseDir` before pointing to the folder containing your input data. Also, be sure that `params.num_nodes` is the number of nodes where there exists normalized expression data within `expression.csv`. Finally, in `NETISCE.nf`, mutations are not considered, so that like is commented out.

## 4.4 Run NETSICE

In your terminal/command prompt, navigate to the appropriate NETISCE folder `_hpc` or `local`. To start your run, enter `./nextflow run NETISCE.nf -resume`. While NETISCE is running, your terminal should look like this, where you can see the progress on each step of the pipeline:

Once the run has successfully completed, the process will end and the following will be displayed:

## 4.5 NETSICE Results

Let's take a look at the results of our NETISCE run, where the goal was to shift the system from the undesired state B, and towards the desired state A. These results can be found in the `toy_example_1` subfolder of the `toy_example_results` folder of the main github repository.

```

executor > local (5)
[36/320889] process > sfa_exp [ 0%] 0 of 1
[-] process > get_exp_internal_control_nodes -
[00/377f94] process > insilico_inits [ 0%] 0 of 1
executor > local (5)
[36/320889] process > sfa_exp [ 0%] 0 of 1
[-] process > get_exp_internal_control_nodes -
[00/377f94] process > insilico_inits [100%] 1 of 1 ✓
executor > local (6)
[36/320889] process > sfa_exp [ 0%] 0 of 1
[-] process > get_exp_internal_control_nodes -
[00/377f94] process > insilico_inits [100%] 1 of 1 ✓
executor > local (7)
[36/320889] process > sfa_exp [100%] 1 of 1 ✓
[81/db9f26] process > get_exp_internal_control_nodes [ 0%] 0 of 1
[00/377f94] process > insilico_inits [100%] 1 of 1 ✓
executor > local (9)
[36/320889] process > sfa_exp [100%] 1 of 1 ✓
[81/db9f26] process > get_exp_internal_control_nodes [100%] 1 of 1 ✓
[00/377f94] process > insilico_inits [100%] 1 of 1 ✓
[0e/1b8810] process > insilico [100%] 1 of 1 ✓
[da/35c1df] process > getFVS [100%] 1 of 1 ✓
[ad/6d8ec2] process > perturbation_inits [100%] 1 of 1 ✓
[41/65a794] process > sfa_perts [100%] 1 of 1 ✓
[0d/2f8569] process > check_icns [100%] 1 of 1 ✓
[12/dba132] process > kmeans_opt [ 0%] 0 of 1
[-] process > kmeans -
[-] process > classification -
[-] process > consensus -
[-] process > internal_control_node_analysis -
[-] process > filtering_by_icn -
[-] process > extract_perts -
[-] process > translate_perts -

```

Figure 4.2: Terminal when running NETISCE

```

Completed at: 02-Dec-2021 10:19:50
Duration    : 4m 29s
CPU hours   : 0.1
Succeeded   : 16

```

Figure 4.3: Terminal when running NETISCE

### exp\_internalmarkers.txt

Our internal marker node was node C. In this file we see the steady state values of the node in the A and B sample replicates (the output values from SFA).

| name | C         |
|------|-----------|
| A_1  | 0.4278056 |
| A_2  | 0.4802943 |
| A_3  | 0.4361991 |
| B_1  | 0.1590962 |
| B_2  | 0.0982107 |
| B_3  | 0.0935476 |

### experimental\_internalmarkers.pdf

The above numbers may be a little challenging to read! So, we have included a plot of the values in the `experimental_internalmarkers.pdf`:

On this histogram, we see bars for each of the samples and their replicates. The A (sensitive) samples are marked by a blue vertical line at their steady state value, while the B (resistant) samples are marked by a red vertical line at their steady state value. Here, we see that the values of node C are well separated between the two phenotypes (all of the A values are greater than all of the B



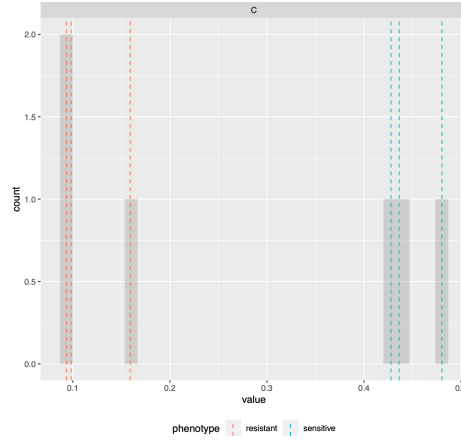


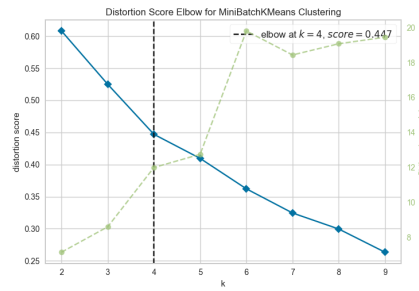
Figure 4.4: experimental marker node steady state values

values). We will assume that this also aligns with the biological knowledge of the system.

In this example, since there are only 4 network nodes that have normalized expression values, NETISCE generates the maximum number of random initial states,  $3^4$ , or 81.

After estimating attractors for the experimental and randomly generated initial states, the resultant attractors were clustered using k-means clustering. The elbow and silhouette metrics are used to determine the optimal number  $k$ .

elbow.png

Figure 4.5: elbow metric for optimal  $k$ 

The elbow metric found the optimal number of  $k$  clusters to be  $k=4$ .

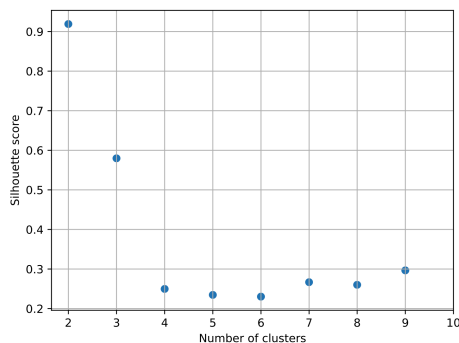
**silhouette.pdf**

Figure 4.6: silhouette metric for optimal k

The silhouette metric found the optimal number of k clusters to be k=2.

Since the optimal ks identified by the silhouette metric and the elbow metric do not match, NETISCE chooses the smaller k, as long as the phenotypes remain separate (NETISCE checks to make sure this is true).

**fvs.txt**

This file contains the node names that were identified by the FVS finding algorithm.

| name |
|------|
| D    |
| B    |

The FVS finding algorithm identified nodes B and D to be the minimal FVS control nodes in the toy network. Since the FVS control node set contained 2 nodes, 9 combinations of perturbations were performed on the control node sets.

**crit1perts.txt**

This file contains a list of IDs for the perturbations to FVS control nodes that passed criterion 1.

|        |
|--------|
| V1     |
| pert_0 |
| pert_1 |
| pert_2 |
| pert_4 |
| pert_5 |
| pert_6 |
| pert_7 |
| pert_8 |

8 out of the 9 perturbations passed the machine learning filtering criterion.

#### **pert1\_internal\_markers.txt**

This file contains a table of the internal marker node state values from control node perturbations whose associated attractors passed the first filtering criterion.

| name   | C          |
|--------|------------|
| pert_0 | -2.7000000 |
| pert_1 | 1.0209997  |
| pert_2 | 6.3000000  |
| pert_4 | -0.0427445 |
| pert_5 | 6.3000000  |
| pert_6 | -2.7000000 |
| pert_7 | -2.8528056 |
| pert_8 | 6.3000000  |

#### **successful\_controlnode\_perturbations.txt**

This file contains a table of the perturbations on FVS control nodes that passed both the 1st and 2nd filtering criteria. it also contains the number of upregulation, downregulations, and total number of nodes perturbed for each perturbation set.

|        | D        | B        | up | down | total |
|--------|----------|----------|----|------|-------|
| pert_1 | down     | nochange | 0  | 1    | 1     |
| pert_5 | nochange | up       | 1  | 0    | 1     |
| pert_2 | down     | up       | 1  | 1    | 2     |
| pert_8 | up       | up       | 2  | 0    | 2     |

Here, we see that four perturbations that passed both filtering criteria.

Let's take a quick look at the steady state values for these perturbations, and the attractors generated from the experimental data:

|    | name   | C         |
|----|--------|-----------|
| 1  | A_1    | 0.4278056 |
| 2  | A_2    | 0.4802943 |
| 3  | A_3    | 0.4361991 |
| 4  | B_1    | 0.1590962 |
| 5  | B_2    | 0.0982107 |
| 6  | B_3    | 0.0935476 |
| 21 | pert_1 | 1.0209997 |
| 31 | pert_2 | 6.3000000 |
| 51 | pert_5 | 6.3000000 |
| 8  | pert_8 | 6.3000000 |

Indeed, we see that the steady-state expression values of node C in the attractors generated by perturbations to the FVS control nodes are all greater than the steady-state expression values of node C in the attractors generated from the sensitive A sample. A successful reprogramming from resistant (B) to sensitive (A) cells has occurred!

## 4.6 Toy Example with mutations

Let's say that in our system, gene A exhibits a loss of function mutation in the sensitive phenotype (A samples). If we want to include this in our simulations, we will use the `NETISCE_mutations.nf` pipeline.

First, we must add to our `input_data` folder a `.csv` file containing the mutational profile. Let's call this file `mutations.csv`:

| X   | A  |
|-----|----|
| A_1 | 0  |
| A_2 | 0  |
| A_3 | 0  |
| B_1 | NA |
| B_2 | NA |
| B_3 | NA |

The loss of function mutation is encoded with 0 (gain-of-function mutations can be encoded with "1").

Next, we make sure that the parameters in `NETISCE_mutations.nf` on lines 3-19 are set correctly for the conditions

For this example, your parameters should look like:

```
params.expressions = "$baseDir/input_data/expressions.csv"
params.network = "$baseDir/input_data/network.sif"
params.samples = "$baseDir/input_data/samples.txt"
```

```

params.internal_control="$baseDir/input_data/internal_marker.txt"
params.mutations="$baseDir/input_data/mutations.csv"
params.alpha = 0.9
params.undesired = 'resistant'
params.desired = 'sensitive'
params.filter = "strict"

```

```

params.kmeans_min_val = 2
params.kmeans_max_val = 10

```

```

params.num_nodes = 4 // that have expression data
params.num_states = 1000

```

Note, the additional parameter `params.mutations` that points to the `mutations.csv`.

As above, to run Netisce, enter `./nextflow run NETISCE.nf -resume`.

## Results

By including mutational information, the results of NETISCE have changed. These results can be found in the `toy_example_2` subfolder of the `toy_example_results` folder of the main github repository. Now, our `successful_controlnode_perturbations.txt` file contains `pert_0` instead of `pert_8`

|        | B        | D        | up | down | total |
|--------|----------|----------|----|------|-------|
| pert_1 | down     | nochange | 0  | 1    | 1     |
| pert_5 | nochange | up       | 1  | 0    | 1     |
| pert_0 | down     | down     | 0  | 2    | 2     |
| pert_2 | down     | up       | 1  | 1    | 2     |

Let's take a look at the steady-state expression values of node C in the attractors generated from the successful perturbations and the experimental initial states when mutational information is included.

| name   | C          |
|--------|------------|
| A_1    | -0.6621166 |
| B_1    | 0.1590962  |
| pert_0 | -2.7000000 |
| pert_1 | -2.7000000 |
| pert_2 | -2.7000000 |
| pert_5 | -2.8528056 |

Though the values are different in this system with mutations, we still see that the steady-state expression values of node C in the attractors generated by

peturbations to the FVS control nodes are all are greater than the steady-state expression values of node C in the attractors generated from the sensitive A sample. A successful reprogramming from resistant (B) to sensitive (A) cells has occurred!

## Chapter 5

# Cell Fate Specification in Ascidian Embryo

This section contains instructions to reproduce the results of simulating FVS control node perturbations in a model of ascidian embryo cell specification. You can read the original report here: [link](#)

The input data, nextflow pipeline, and results of this simulation can be found in the `ascidian_embryo` folder in the github repository

These simulations were run on a high performance cluster that uses a SLURM executor. Although we recommend that you run NETISCE on an hpc, this simulation is small enough that it can be run on a local machine. If you choose to run it locally, then remove the `nextflow.config` file from the directory.

### 5.1 Input Data

The goal of this simulation was to reproduce the results of experimental perturbations to the FVS nodes of the cell fate specification GRN for ascidian embryos using Signal Flow Analysis. Therefore, we use a modified version of the NETISCE pipeline to simulate these specific perturbations. We are only interested in performing the 7 perturbations to the 6 FVS control nodes that were experimentally verified to induce cell tissue fates.

`expression.csv` contains the initial activities for the unperturbed state and the 7 FVS control node perturbations. Here, all simulations have `Gata.a` and `Zic-r.a`=1, as the activation of these two genes is required for normal embryonic development.

`perturbations.csv` contains the specified perturbations for each FVS node in the appropriate perturbation simulation. 0 denotes downregulation, whereas 1

| name                                   | Alp        | Bco        | Celf3.a    | Epi1       | Fli.Erg.a  | 1       |
|--|------------|------------|------------|------------|------------|---------|
| unperturbed                            | -0.0000341 | 0.0018729  | 0.0018729  | 0.0217995  | 0.0013966  | 0.0008  |
| Adentz (Endoderm perturbation)         | 0.1179951  | -0.4252100 | -0.4252100 | 0.2707276  | -0.1836794 | -0.1396 |
| adentZ (brain+pan-neural perturbation) | -0.3845691 | 0.4252100  | 0.4252100  | 0.2707276  | -0.2370737 | -0.0041 |
| adeNtz (pan-neural perturbation)       | -0.5243074 | -0.4252100 | -0.4252100 | 0.2707276  | -0.3002010 | -0.1063 |
| adEntZ (mesenchyme perturbation)       | -0.2630083 | 0.4252100  | 0.4252100  | -0.1563935 | 0.2841288  | -0.0871 |
| adentz (epidermis perturbation)        | -0.5243074 | -0.4252100 | -0.4252100 | 0.2707276  | -0.3002010 | -0.1237 |
| adenTz (muscle perturbation)           | -0.5243074 | -0.4252100 | -0.4252100 | 0.2707276  | -0.3002010 | 0.0837  |
| aDentz (notochord perturbation)        | -0.3792942 | -0.4252100 | -0.4252100 | 0.1988199  | -0.3966199 | -0.1218 |

|  | Alp   |
|--|---|
| Adentz (Endoderm perturbation)         | <span style=" border-radius: 4px; padding-right: 4px; padding-left: 4px; ba |
| adentZ (brain+pan-neural perturbation) | -0.385  |
| adeNtz (pan-neural perturbation)       | -0.524  |
| adEntZ (mesenchyme perturbation)       | -0.263  |
| adentz (epidermis perturbation)        | -0.524  |
| adenTz (muscle perturbation)           | -0.524  |
| aDentz (notochord perturbation)        | -0.379  |

encodes upregulation. If no value is set, then there is no fixed perturbation to the FVS node, as in the unperturbed case.

**internal-marker-nodes.txt** contains the 7 internal marker nodes used to verify if the specified cell reprogramming had been successfully simulated.

## 5.2 Run the simulation

To run the simulation, simply execute the **ascidian-embryo.nf** file using the following command: `./nextflow run ascidian-embryo.nf -resume`

## 5.3 Results

The nextflow pipeline generates 1 result file **exp\_internalmarkers.txt**, which contains the steady state values of the internal-marker nodes for the unperturbed attractor, and the attractors generated from the perturbations on FVS control nodes.

A perturbation is considered successful if the internal-marker node in the attractor generated from the perturbed FVS control nodes has a larger steady-state value than that in the unperturbed attractor. We determine this by subtracting the steady-state values of the unperturbed simulation from the steady-state values of the perturbations of FVS control nodes.

Here we see that for 6 out of the 7 perturbations to FVS control nodes, we were able to upregulate the desired tissue marker when compared to the unperturbed state.



## 5.4 Visualizing Results

We can use radar plots to visualize the results of the SFA simulations of perturbations to the FVS control nodes. This can help us identify which perturbations successfully induced the appropriate tissue fate. In this context.

The following code for generating radar plots was adapted from [datanovia.com](http://datanovia.com) is used to generate the radar charts. **Note:** you may need to adjust the formatting of `exp_internalmarkers.txt` so that the strings within quotations are placed into one column.

```
create_beautiful_radarchart <- function(data, color = "#00AFBB",
                                       vlabels = colnames(data), vlce = 1,
                                       caxislabels = NULL, title = row.names(data)[4], ...){
  radarchart(
    data, axistype = 1,
    # Customize the polygon
    pcol = color, pfc = scales::alpha(color, 0.5), plwd = 2, plty = 1,
    # Customize the grid
    cglcol = "grey", cglty = 1, cglwd = 0.8,
    # Customize the axis
    axislabcol = "grey",
    # Variable labels
    vlce = vlce, vlabels = vlabels,
    title = title,
    centerzero = F,
    caxislabels = caxislabels
  )
}

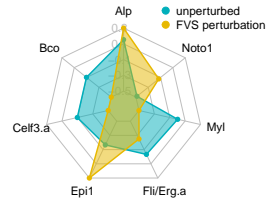
library(fmsb)
d1<-read.delim("ascidian_embryo/results/exp_internalmarkers.txt",sep="\t",row.names = 1,check.names=FALSE)

maxcol<-apply(d1, 2, max)
mincol<-apply(d1, 2, min)

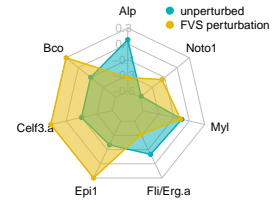
d2<-rbind(maxcol,mincol, d1)
rownames(d2)[1:2]<- c("Max", "Min")

par(mar = c(4, 0.1, 4, 0.1))
for (i in 4:nrow(d2)) {
  create_beautiful_radarchart(d2[c(1:3, i)], ,color = c("#00AFBB", "#E7B800"),caxislabels = seq(1,3))
}
```

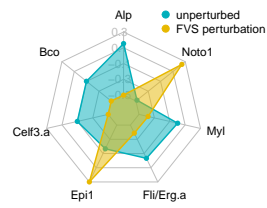
**Adentz (Endoderm perturbation)**



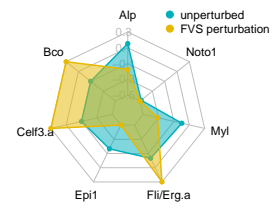
**adentZ (brain+pan-neural perturbation)**



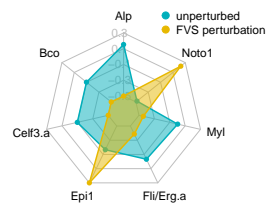
**adeNtz (pan-neural perturbation)**



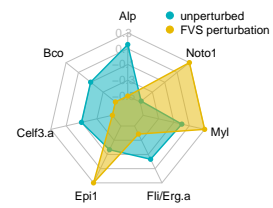
**adEntZ (mesenchyme perturbation)**



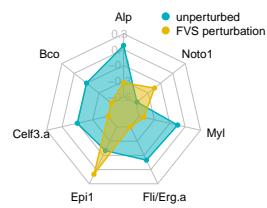
**adentz (epidermis perturbation)**



**adenTz (muscle perturbation)**



**aDentz (notochord perturbation)**



## Chapter 6

# Pluripotent Stem Cell Example



## Chapter 7

# Adaptive Resistance in Colorectal Cancer Example



## Chapter 8

# NETISCE in Galaxy Project