Gene network practical

Network computation even on as small a dataset as we have used last week is computationally demanding.

Aim

The aims of this tutorial are to perform the data retrieval and preparation (optional), build a meta-network (an ensemble of network inferences), analyse it, and export it as a graphml file and visualise it (optional)

Prerequisites

You need the following tools installed on your laptop / desktop for the visualisation (optional)

- Cytoscape or Gephi

Data preparation

I have performed the data preparation for you, see the files:

- `src/R/dataRetrieval.R`
- `src/R/dataPreparation.R`

You are welcome to knit them and take a look at the html reports it will generate. It will also set up the output directory.

To perform the analysis, I first copied the scripts Bastian used last week (from the Pathway-Enrichment-Exercises repository):

- `preproc_res.R`
- `preproc_res2.R`

and merged them into the first script listed above. I modifed the code to use a uniform ID for the genes (the gene symbols) across the datasets. The resulting raw data, I saved as an object.

I then copied the biologicalQA.R template from the `UPSCb-common/templates/R` submodule directory and modified it to load the data retrieved at the previous step, perform a number of quality assessment, devise the most appropriate normalisation, performed the data filtering and exported the data to be used by `seidr`.

Network inference

This part of the practical will make use of the CLI (command line interface). You can use any of the solution we provided, at your discretion. Either connect directly to the server using `ssh`, use `VS Code` or the `RStudio terminal`. For the latter, be aware that while useful, the terminal emulation in `RStudio` may be unstable at times.

Note: If you have not performed the knitting of the files in the section above, you need to create the analysis directory. I called it (creatively) seidr.

```
mkdir -p ~/GeneNetworksExample/analysis/gene_network/seidr
```

Then we need to link the data

```
ln -s ~/raw_data/downstream/seidr/*.tsv .
```

Now, let's start. First of all `cd` into your `seidr` directory

```
cd ~/GeneNetworksExample/analysis/gene_network/seidr
```

You can run any of the following inference methods, note that some will take some time.

To run them you need to enable `seidr` in your environment. It is provided as a `conda` environment that you need to activate.

```
conda activate seidr
```

Before we proceed, let's try if it worked by displaying the `seidr` help

```
seidr
```

The data we generated above for `seidr` consists of 2 files, named `headless.tsv` and `genes.tsv`. They contain 28562 genes. `headless.tsv` is a matrix of 15 samples (rows) x 28562 genes (colums), while the genes.tsv contains a single lane holding all the gene IDs (the column names extracted from the headless.tsv file).

Note: Some processes will take a long time to run. To ensure that jobs complete, you may want to be able to detach the session, using *e.g.* `screen` or `tmux`. For example, `screen -S seidr` would start a named session. Then if you get disconnected, you could use `seidr -r seidr` to restore it.

Let's get started finally! All methods have a `-h` flag if you need to check what the option mean. There is also, lots of documentation on `seidr` 's readthedoc.

- **Aracne** (mutual information, slow, ~5 hours)

```
mi -m ARACNE -M mi.tsv -O 5 -B 28562 --save-resume aracne.json \
-i headless.tsv -g genes.tsv -o aracne.tsv 2> aracne.err
```

- **CLR** (mutual information, fast, ~ minutes)

```
mi -m CLR -M mi.tsv -O 5 -B 28562 --save-resume clr.json \
-i headless.tsv -g genes.tsv -o clr.tsv > clr.out 2> clr.err
```

- **narromi** (mutual information, slow ~5 hours)

```
narromi -O 5 -B 28562 \
-i headless.tsv -g genes.tsv -o narromi.tsv 2> narromi.err
```

- **pcor** (correlation, fast, ~5 minutes)

```
pcor -i headless.tsv -g genes.tsv -o pcor.tsv 2> pcor.err
```

- **pearson** (correlation, fast, ~5 minutes)

```
correlation -m pearson \
-i headless.tsv -g genes.tsv -o pearson.tsv 2> pearson.err
```

- **spearman** (correlation, fast, ~5 minutes)

```
correlation -m spearman \
-i headless.tsv -g genes.tsv -o spearman.tsv 2> spearman.err
```

- **tomsimilarity** (correlation, fast, ~5 minutes)

```
tomsimilarity -i headless.tsv -g genes.tsv -O 5 \
-o tomsimilarity.tsv 2> tomsimilarity.err
```

- **plsnet** (partial least square, slow, ~7 hours)

```
plsnet -O 5 -B 28562 \
-i headless.tsv -g genes.tsv -o plsnet.tsv 2> plsnet.err
```

- **genie3** (random forest, slow, ~40 minutes)

```
genie3 -O 5 -B 28562 \
-i headless.tsv -g genes.tsv -o genie3.tsv 2> genie3.err
```

Import

Different methods have been used to compute network inference. All these methods have different scoring metrics, which pauses a problem for comparing and aggregating them. Hence, we first need to change the metrics into ranks (as these will be comparable). Further, we also scale the ranks so they are in the range [0,1].

Let's create a result dir

```
mkdir results
```

- Aracne

  You actually ran 2 inferences, `mi` and `aracne`

  ```
  seidr import -r -z -u -F lm -n MI -o results/mi.sf \
  -i mi.tsv -g genes.tsv -O 5 2> mi_import.err
  ```

  ```
  seidr import -r -z -u -F lm -n ARACNE -o results/aracne.sf \
  -i aracne.tsv -g genes.tsv -O 5 2> aracne_import.err
  ```

- CLR

  ```
  seidr import -r -z -u -F lm -n CLR -o results/clr.sf \
  -i clr.tsv -g genes.tsv -O 5 2> clr_import.err
  ```

- narromi

  ```
  seidr import -r -z -F m -n NARROMI -o results/narromi.sf \
  -i narromi.tsv -g genes.tsv -O 5 2> narromi_import.err
  ```

- pcor

```
seidr import -r -z -u -A -F lm -n PCOR -o results/pcor.sf \
-i pcor.tsv -g genes.tsv -O 5 2> pcor_import.err
```

- pearson

```
seidr import -r -z -u -A -F lm -n PEARSON -o results/pearson.sf \
-i pearson.tsv -g genes.tsv -O 5 2> pearson_import.err
```

- spearman

```
seidr import -r -z -u -A -F lm -n SPEARMAN -o results/spearman.sf \
-i spearman.tsv -g genes.tsv -O 5 2> spearman_import.err
```

- tomsimilarity

```
seidr import -r -z -u -A -F lm -n TOMSIMILARITY -o results/tomsimil
arity.sf \
-i tomsimilarity.tsv -g genes.tsv -O 5 2> tomsimilarity_import.err
```

- plsnet

```
seidr import -r -z -F m -n PLSNET -o results/plsnet.sf \
-i plsnet.tsv -g genes.tsv -O 5 2> plsnet_import.err
```

- genie3

```
seidr import -r -z -F m -n GENIE3 -o results/genie3.sf \
-i genie3.tsv -g genes.tsv -O 5 2> genie3_import.err
```

Cleanup

Now that we imported the files, we can clean up a bit

```
rm aracne.tsv clr.tsv genie3.tsv mi.tsv \
narromi.tsv pcor.tsv pearson.tsv plsnet.tsv \
spearman.tsv tomsimilarity.tsv
```

Aggregating

FROM THIS POINT ON, you are meant to come up with the commands to run

The SOLUTION is at the bottom, if you are stuck

Now that all the networks have been converted to the same format and use a similar score, we can aggregate them into a meta-network.

Take a look at `seidr` 's readthedoc or `seidr` 's command line help to figure out how to aggregate the network.

Despite the few samples, the aggregation will last approximately an hour

Backboning

The aggregated network is still full-rank, meaning that every node (gene) is connected to every other node. Hence, we need to filter the "noise" away. Find the command to keep the best ten (10) percent of every node edges.

Assessment

Calculation

The next step is to assess how good has the network inference been. This can be done using a gold standard and the `seidr roc` function. The gold standard is based on known edges present in KEGG pathways. The assessment is done by calculating a Receiver Operating Characteristic ROC curve, that displays the false positive rate (1 - specificity) *vs.* the true positive rate (sensitivity). An important metric is the Area Under the Curve (AUC), which gives an idea of the prediction accuracy.

Figure out how to run `seidr roc` on

1. the individual inference(s) you ran
2. on the aggregated network
3. on the ten percent backbone

The gold standard is available as `~/raw_data/downstream/seidr/hsa_gene_symbol_pos.tsv`

We want to run the ROC on the whole network (`-E 1`) and generate a 100 data points for every roc.

Visualisation

In your RStudio, visualise the three (or more) ROCs you have calculated.

Adapt the following code. If you are stuck, upload the roc.zip file linked above.

```
library(here)
install.packages("pracma")
library(pracma)

dat <- read.delim(
    here("analysis/gene_network/seidr/roc/aggregated-irp.roc"),
    header=FALSE,skip=1,
    col.names = c("TP","FP","PR"))

auc <- round(trapz(dat[,2],dat[,1]),digits=3)

plot(dat[,2],dat[,1],type="l",main=sprintf("%s (AUC = %s)","IRP",auc),
     xlab="False Positive Rate",ylab="True Positive Rate")

abline(0,1,lty=2)
```

If you feel like, try to use the `#'` comments to generate a report from your `R` script. Best is to write these comments as you do your analysis.

Navigating the network

First degree neighbours

We want to check how the genes we found to be differentially expressed are connected in the network. For this, we will find the first degree neighbours (the gene connected directly to a gene of interest, say e.g. `TP53`) and recreate a network using only the edges connecting them.

Take a look at the `nDegreeNeighbours.R` script. Once you have generated the first degree neighbour graph, you can export it from RStudio to your computer to visualise it. This requires Cytoscape or Gephi to be installed.

If you did not manage to create the graph, you can use the `~/raw_data/downstream/seidr/RB1SecondDegreeNeighbour.graphml` file.

Solutions

Aggregating

```
seidr aggregate results/*.sf -O 5 -o aggregated.sf
```

Backboning

```
seidr backbone aggregated.sf -F 1.28 -o backbone-10-percent.sf
```

ROC "-ing"

Calculation

```
# shows all the aggregated inference methods
seidr view -H aggregated.sf | grep '\[A\]'

# roc calculation (a few as example!)
## genie3
seidr roc -g ~/raw_data/downstream/seidr/hsa_gene_symbol_pos.tsv -i 3
\
-n aggregated.sf -E 1 -p 100 -o aggregated-genie3.roc

## aggregated
seidr roc -g ~/raw_data/downstream/seidr/hsa_gene_symbol_pos.tsv \
-n aggregated.sf -E 1 -p 100 -o aggregated-irp.roc

## backbone
seidr roc -g ~/raw_data/downstream/seidr/hsa_gene_symbol_pos.tsv -n ba
ckbone-10-percent.sf -E 1 -p 100 -o backbone-10-percent.roc
```

Visualisation

My code and report are in the repository.

Exporting an edge list

You did not run this, but I did it for you

```
seidr view backbone-10-percent.sf | \
awk '{print $1,$2}' > backbone-edgelist.txt
```