NetID Tutorial

Ken

In this tutorial, I would use hematopoietic single cell RNA-seq dataset to show how to conduct network inference through NetID.

Load the single cell datasets and source the function.

```
> source("NetID_restreamlined_code.R")
> sce <- readRDS("blood_sce.rds")
```

The sce objects contains the spliced/unspliced read count matrices and the metadata of cells. For showing the performance, I also loaded the transcriptional factors gene sets and the tissue-unspecific GRN curated from ChIP-seq datasets as the ground truth.

```
> TF <- read.csv("mouse-tfs.csv",header=TRUE,stringsAsFactors=FALSE)
> gt_net <- read.csv("Non-Specific-ChIP-seq-network.csv",header=TRUE, stringsAsFactors=FALSE)
```

1. Learning GRN skeleton from sketched and aggregated single cell RNA-seq datasets

At the first step, NetID would samples the single cell RNA-seq dataset through sketching methods, e.g. geosketch or Seurat sketch. These sketching methods would sample the cells as the “meta-cells” which would cover all latent manifold of single cell transcriptome. Then, the nearest neighborhoods of those meta-cells would specifically assign to the one meta cell according to the edges *P*-values calculated from VarID, and finally perform aggregation. The resulted GEP of meta-cells would be used as the input of network inference methods like GENIE3 (default), to learn the basic network structure.

```
> dyn.out <- RunNetID(sce,regulators = TF$TF, targets = TF$TF,netID_params =
list(normalize=FALSE), dynamicInfer = FALSE)
```

The parameters of *RunNetID:*

1. sce: the single cell experiments object
2. min_counts: minimum read counts of each genes within all cells, default: 10
3. varID_res: input the varID object if exist, default: NULL
4. knn: k-nearest-neighbors, default: 30
5. regulators: candidate regulator gene list
6. targets: candidate target gene list
7. netID_params: a list object contains following parameters
    1. var: if using variable gene to calculate principal components, used by geosketch method, default: FALSE
    2. sampled_cells: the barcode of sampled cells, default: NULL
    3. sketch.method: perform sketching sampling on single cell datasets, "geosketch" or "SeuratSketching", default: “geosketch”
    4. ndim: dimensions of PCs, used by geosketch method, default: 30
    5. n_cell: the number of sampled cells, default: 500

6. Threshold_Num: the minimum number of nearest-neighbors of each meta cells after assignments
7. normalize: if perform normalization to single cell datasets, default: FALSE
8. prior_net: the prior knowledge of network structure, e.g. scATAC-seq

8. velo: if consider the cell fate prediction results learn from RNA velocity (cellrank), default: FALSE
9. dynamicInfer: if consider the cell fate prediction results, required for cell fate specific GRN prediction, default: TRUE
10. maxState: NetID would internally clusters cells into different cell states according fate probability of each cells, and assign the lineage identity of each cell states. NetID using Gaussian Mixture Model to perform clustering from 2 to maxState, and select the best clustering results according to BIC. Default:5.
11. cut_off: calculate enrichment score of each branch, and setting the cutoff to determine the which branch that cell state belongs to. Default: 2
12. work_dir: work dictionary, default: NULL

This step only output the global GRN learned from all sampled meta-cells. To learn lineage-specific GRN, we need to run cellrank and Palantir at first. NetID provided a function for end-to-end cell fate analysis through cellrank and Palantir.

```
> FateDynamic(sce,global_params = list(cluster_label = "celltype",min_counts = 10,nhvgs = 3000))
```

The FateDynamic contains following parameters:

1. work_dir: work dictionary, default: NULL
2. sce: single cell experiments object
3. global_params: a list object contains the global parameters
    1. min_counts: minimum read counts of each genes within all cells, default: 20
    2. nhvgs: number of highly variable genes, default: 2000
    3. npcs: number of principal components, default: 30
    4. knn: number of nearest-neighbors, default: 30
    5. velo: perform RNA velocity analysis, default: TRUE
    6. fate_predict: perform cell fate prediction, default: TRUE
    7. cluster_label: which variable would be used as the cluster label, default: NULL
4. palantir_params: a list object contains the palantir parameters
    1. ndcs: number of diffusion components, default: 10
    2. start_cell: the barcode of root cell, default: NULL
    3. nwps: number of waypoint sampling, default: 500
    4. plot: plot the results, default: TRUE
    5. method: the method to infer the root cells of differentiation when start_cell = NULL, “SCENT” or “CytoTRACE” default: “SCENT”
    6. species: specific the species number (used by SCENT).
5. cellrank_params: a list object contains the cellrank parameters
    1. mode: the RNA velocity mode, “deterministic”, “stochastic” or “dynamical”, default: “stochastic”
    2. plot: plot the results, default: TRUE
    3. weight_connectivities: the weights for integrate velocity graph and nn graph, default: 0.2.

4. ncores: The number of cores. plot: TRUE
5. tolerance: the tolerance for updating. Default: 10^-6.

This function would generate two files in current work dictionary including /output/FateRes.h5ad and ./output_velo/FateRes.h5ad, which represent the results of Palantir and CellRank respectively. Then we could re-run the RunNetID function, and add the cell fate prediction results in output.

```
> dyn.out <- RunNetID(sce,regulators = TF$TF, targets = TF$TF,netID_params = list(normalize=FALSE), dynamicInfer = TRUE)
```

2. Learning cell fate specific GRN through granger causal regression model.

The dyn.out object contains the skeleton of global network (learned from all cells without lineage information) and cell fate probability information. NetID would run L2-penalized granger regression model for each target genes to re-calculate the regulatory coefficients of the skeleton, with using cell fate probability as the “time-series”, and then aggregate the learned coefficients with the global coefficients through rank method.

```
> GRN <- FateCausal(dyn.out,velo_infor = FALSE,L=10,weight=0.1)
```

The FateCausal contains following parameters:

1. dyn.out: The list object returned from RunNetID with dynamicInfer = TRUE
2. velo_infor: integrate velocity information when calculate cell fate specific GRN
3. L: the delayed time stamps for granger causal regression. Default: 10.
4. Lambda: the L2 panelize coefficients. Default: 100
5. weight = 0.2: the weights for linear combination of granger causal coefficient and global network coefficients, a larger weight means the cell type-specific GRN would borrow more regulatory information from global network.
6. redirected: re-calculate the regulatory direction for each pair of genes using granger causal test. Default: FALSE
7. cutoff: the *P*-value cutoff for granger causal test. Default: 0.25
8. fate_method: Using the fate probability derived from “palantir” or “cellrank”, Default: palantir

The output of FateCausal is a list object contains the lineage specific GRN.
And finally, we benchmarked the learned cell type specific GRN, with the network learned from GENIE3 in the first stage. We used the ChIP-seq datasets as the ground truth for benchmarking.

```
> baseNet <- BaselineNetwork2(dyn.out$skeleton$skeleton)
# the results of GEINI3 GRN
> net <- netCompare(baseNet$g_pair, dyn.out$skeleton$GENIE3_net)
> EPR(net,gt_net)$roc
Setting levels: control = 0, case = 1
Setting direction: controls < cases

Call:
roc.default(response = g3_pair$link, predictor = g3_pair$EdgeWeight,     direction = direction)
```

```
Data: g3_pair$EdgeWeight in 43362 controls (g3_pair$link 0) < 776 cases (g3_pair$link 1).
Area under the curve: 0.572
# the results of cell type specific GRN
> net <- netCompare(baseNet$g_pair,t(GRN[[1]]+GRN[[2]]))
> EPR(net,gt_net)$roc
Setting levels: control = 0, case = 1
Setting direction: controls < cases

Call:
roc.default(response = g3_pair$link, predictor = g3_pair$EdgeWeight,     direction = direction)

Data: g3_pair$EdgeWeight in 43362 controls (g3_pair$link 0) < 776 cases (g3_pair$link 1).
Area under the curve: 0.6084
```