**Workflow Summary:**

**Construct target databases** (./databases/)

**Score TFs using NetAct** (./data.tfs/)

**Combine TF sets from three methods NetAct, MARINa, and RI** (tfSets/Phase.05) - selTopTFs.byMethod.R

**Infer circuits** (./networks/) – pip.pre\_sim.sh

**11** TF binding probability levels (feature ratio cut offs): 0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.12, 0.14, 0.16, 0.18, and 0.20. (outer loop – L1)

**15** Top TF count for each method: 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, and 60. (inner loop – L2)

20 interaction strengths (correlation values): 0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, and 0.95. (inner loop – L3)

We have 165(=11x15) TF sets obtained by combining top TFs from each method. For each TF set, we infer a candidate gene regulatory circuit by following the steps described below.

**Construct initial network**

1. Construct initial network form from TF-target DB – by expanding the source target relations in the TF-target DB.
2. Retain subnetwork restricted to only core TFs – accomplished by retaining only those top TFs that are also found in the target DB
3. Retain subnetwork restricted to TFs whose NetAct activities can be calculated – this amount to retaining the TFs that are also available in the DE gene sets (TFs) obtained from experimental data using NetAct.
4. Add interactions (+ve or -ve type) to the network to obtain the initial network
   1. First, calculate pair-wise correlations between the gene activity vectors (activities calculated using NetAct)
   2. Then, assign interaction type to the network interactions based on their values – positive correlations are assigned as activation type (1) and negative correlations are assigned as inhibition type (2).

**Infer candidate networks from initial network**

1. Create an ensemble of networks
2. Remove the duplicated networks
3. Further refine the ensemble as follows – for each network retain the largest connected component having nodes more than 80 percent. (select.largest.connected.subgraph)

**Simulate inferred circuits** (./networks) – pip.sim.sh

**Find optimum circuits** (./networks) – pip.post\_sim.sh

**Scripts and required inputs**

|  |  |  |  |
| --- | --- | --- | --- |
| **L1 scripts: construct TF-target DBs and combined TFs** | | | |
| Step # | Script name | inputs | outputs |
| 1 | create.targetDB.list.R  (./databases) | 1. Lit. based target DB 2. TF-target binding affinity from ATAC-seq | 1. targetDB.list.rds |
| 2 | selTopTFs.byMethod.R  (./tfSets) | 1. TFs.combined.rds   ranked TFs from each method   1. targetDB.list.rds   list of target DBs | coreTFs.rds – core TFs at each level of top TFs |

|  |  |  |  |
| --- | --- | --- | --- |
| **L2 scripts: Infer candidate circuits** | | | |
| Step # | Script name | inputs | outputs |
| 1 | infer.circuits.R | 1. coreTFs.rds 2. targetDB.list.rds | circuits.rds |
| 2 | cal.circuit\_summary.R | circuits.rds | 1. summary.circuits.csv 2. summary.circuits.asorted.csv |
| 3 | select.uniq.circuits.R | 1. circuits.rds 2. summary.circuits.csv | 1. circuits.uniq.rds |
|  |  |  |  |
| 4 | sim.circuits.para.savedbyfile.R | 1. circuits.uniq.rds | circuit\_simulated\_xx.xx.xx.rds |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **L3 scripts: Find the optimum circuit** | | | |
| Step # | Script name | inputs | outputs |
| 1 | cal.circuit\_TFactivities.R  As this program does not need simulation results, it can before or while sim.circuits.para.savedbyfile.R is running | 1. summary.circuits.csv 2. circuits.uniq.rds 3. coreTFs.rds 4. targetDB.list.rds | 1. circuit\_TFactivities.rds |
| 2 | cal.metrics.sim\_circuits.R | 1. summary.circuits.csv 2. circuits.uniq.rds 3. circuit\_simulated\_ xx.xx.xx.rds | 1. summary.circuits.sim.csv 2. hS\_xx.xx.xx.rds |
| 3 | cal.metrics.mean.sd.R | 1. summary.circuits.csv 2. circuits.uniq.rds 3. clusterCut.REF.csv 4. circuit\_TFactivities.rds 5. circuit\_simulated\_ xx.xx.xx.rds | 1. summary.circuits.sim.mean\_sd.csv |

|  |  |  |  |
| --- | --- | --- | --- |
| **L4 scripts: Perturbation analysis (./pathway.anno)** | | | |
| Step # | Script name | inputs | outputs |
| 1 | obtain.DEGs.R | 1. de.resultst.rda | 1. DEGs.txt |
| 2 | obtain.DEGs\_in\_circuit.R | 1. de.resultst.rda 2. targetDB.list.rds | 1. DEGs\_in\_circuit.txt |
| 3 | Web interface: enrichr | 1. DEGs\_in\_circuit.txt | 1. KEGG\_2019\_Human\_table.txt |
| 4 | map\_TFs\_2\_pathways.R – Fisher exact test | 1. DEGs\_in\_circuit.txt 2. KEGG\_2019\_Human\_table.txt | 1. TFsWith\_singleAnnotedPathway.csv |
| 5 | Visualization: Cytoscaspe | 1. circuit-0.09-32-0.85.tpo 2. TFsWith\_singleAnnotedPathway.csv | 1. circuit-annotated.pdf |

|  |  |  |  |
| --- | --- | --- | --- |
| **L6 scripts: Convergence analysis**  100K models for top 10 circuits  10 samples at each sample size level (1K ~ 10K) are drawn from 100K models  This gives us 10 sample sets: 1K size sample ~ 10K size sample | | | |
| Step # | Script name | inputs | outputs |
| 1 | cal.metrics.sim\_circuits.R – calculate Accuracy and AvgDist | 1. clustercut.REF.csv | (1) circuit\_metrics.sim.list.rds |
| 2 | rankCircuits.R | (1) circuit\_metrics.sim.list.rds | (1) circuit\_metrics.sim.sorted.rds |
| 3 | cal.mean.sd.R | (1) circuit\_metrics.sim.sorted.rds | (1) metric.comb.sorted.csv |
| Steps 1 ~ 3: apply on each level of sample size  Steps 4 ~ 5: apply on the combined metric obtained from Steps 1 ~ 3 | | | |
| 4 | cal.mean\_SD.R – Calculate mean of the means of the combined index across samples | 1. metric.comb.sorted.csv – from each sample size level | 1. mean.SDs.pdf |
| 5 | boxplot.top\_circuits.bySampleSize.R | 1. mean.SDs.pdf | 1. figures |