PACKAGES

To get started, make sure you have the following R packages installed: stringr optparse viper

You'll also need to install the following python libraries (using python 3): numpy pandas scanpy igraph

INTRO

This walkthrough will use the following two files from Peter Sims's CZI data as a starting point: PP001swap.filtered.matrix.txt.bz2 PP002swap.filtered.matrix.txt.bz2

Note that when these two files are specified as arguments to command line functions, you'll need to provide the full path to the files. The same goes for all files you specify as inputs.

To start, make a directory for the outputs of the analysis (which will be referred to as 'YOUR-OUT-DIRECTORY'). You'll also need an .rda file containing all the GTEx regulons and a conversion dictionary for converting gene names. Both are provided as 'gtex-interactome-list-entrez.rds' (in the CZI folder on the cluster) and 'gene-convert-dict.rds' respectively.

STEP 0: cziMerge.R

The CZI data arrives in two files; one for acting and one for resting. We'll merge these two files, giving the cells appropriate names, and create a unified data frame for all donstream analysis. This is a 'Step 0' because it is unique to the CZI data set.

Run the following command:

```
Rscript cziMerge.R --rest_file=PP001.swap.filtered.matrix.txt.bz2 --
act_file=PP002.swap.filtered.matrix.txt.bz2 --out_name=d1-lung --out_dir=YOUR-OUT-
DIRECTORY
```

STEP 1: preProcess.R

Next, we will take the raw data and perform some pre-processing steps. We'll remove cells with either too few or too many counts as well as any genes with no reads. We'll also save a CPM normalized copy of the data.

Run the following command:

```
Rscript preProcess.R --raw_file=YOUR-OUT-DIRECTORY/d1-lung_mergedRaw.rds --
out_name=d1-lung --out_dir=YOUR-OUT-DIRECTORY --min_count=1000 --max_count=100000
```

STEP 1.5: gene name conversion

Our GTEx interactomes are built using Entrez, but our data has Ensembl gene names. This step is necessary in order to convert gene names. Note: currently working on a new set of GTEx interactomes that will use Ensemble, making this step unnecessary.

Run the following command:

```
Rscript geneNameConvert.R --input_file=d1-lung_mergedCPM.rds --convert_dict=CONVERT-DICT --start_index=6 --dest_index=4 --out_name=d1-lung_mergedCPM_entrez.rds --out_dir=YOUR-OUT-DIRECTORY
```

STEP 2: single cell network

Generate an ARACNe network using a random subsample of 500 cells from all of the data available for the tissue. To Do: add guide on how to do this.

STEP 3: metaViper.R

We'll now compute protein activity of the data using metaVIPER. This is a three step process, incorporating both the single cell network and the GTEx interactormes. There are three steps to this process.

Step 1: run VIPER using the single cell network (To Do: add code to do this). Save the viper matrix as 'd1-lung_scNet-vip.rds'

Step 2: run metaVIPER using the GTEx bulk networks, then convert the file to ensembl gene names

```
mkdir d1-lung-GTEx-mVIP
```

```
bash metaViper.sh d1-lung_mergedCPM_entrez.rds d1-lung_mergedCPM_entrez.rds gtex-interactome-list-entrez.rds d1-lung-GTEx-mVIP
```

```
Rscript geneNameConvert.R --input_file=d1-lung_GTEx-mVip_entrez.rds --convert_dict=CONVERT-DICT --start_index=4 --dest_index=6 --out_name=d1-lung_GTEx-mVip.rds --out_dir=YOUR-OUT-DIRECTORY
```

Step 3: merge the two, using the GTEx bulk networks to fill in the inferred activity of regulators that the single cell network missed `Rscript viperMerge.R --priority_file=d1-lung_scNet-vip.rds --fill_file=d1-lung_GTEx-mVip.rds --out_name=d1-lung_mergedVip.rds --out_dir=YOUR-OUT-DIRECTORY`

To expedite the metaVIPER run, we will use Aaron's QSUB metaVIPER script.

STEP 3.5: scanpy-prep.R

Scanpy needs an anontation vector (indication source file of each cell) and cannot take RDS. Thus, a preprocessing step to create a scanpy friendly text file and an annotation vector is necessary. NOTE: I know this is ineffeicient and terrible, we're working on making it unnecessary.

Run the following command:

```
Rscript scanpy-prep.R --in_file=d1-lung_mergedVip.rds --out_name=d1-lung_mergedVip --out_dir=YOUR-OUT-DIRECTORY
```

STEP 4: scanpy-clust.py

Now, we'll cluster the protein activity using scanpy. This will use seurat's dispersion metrics to select the most variable proteins, then perform Louvain clustering on the cells and generate a text file with cluster labels for each cell.

Run the following command:

```
python3 scanpy-pActClust.py d1-lung_mergedVip_scanpy-inDat.txt d1-lung_mergedVip_scanpy-annotationVect.txt d1-lung_mergedVip YOUR-OUT-DIRECTORY
```

STEP 5: clustSubset.R

With our cluster labels in hand from scanpy, we will generate raw count files for each cluster that passes a specified size threshold.

Run the following command:

```
Rscript clustSubset.R --input_file=d1-lung_mergedFiltered.rds --cluster_labels=d1-lung_mergedVip_pAct-louvainClust.txt --sample_index=1 --cluster_index=3 --out_name=d1-lung_mergedVip-clust --out_dir=YOUR-OUT-DIRECTORY
```

STEP 6: makeMetaCells.R

Now, we generate meta cells for each cluster subset. For each subset from the previous step (where the cluster number replaces #), we create meta cells that can be run with ARACNe. In addition to the raw count matrix (from which meta cells will be computed and normalized), this script requires a matrix of protein activity, which is used to calculate inter-cell distance useing viper similarity. As a default, 5 neighbors are used, but that parameter can be changed with the 'num_cells' argument. Additionally, the meta cells will be subsampled to 200 cells by default, a parameter that can be changed with the 'subset_size' argument.

Run the following command (for each cluster):

```
Rscript makeMetaCells.R --input_file=d1-lung_mergedVip-clust_cluster-#.rds --activity_file=d1-lung_mergedVip.rds --out_name=d1-lung_mergedVip-cluster-# --out_dir=YOUR-OUT-DIRECTORY
```

STEP 7: run ARACNe

Finally, we run ARACNe on the meta cells from each cluster. Use the subClustAracne.sh script on each of the meta cell files from the previous file.

You should then re-run metaVIPER with the networks generated from the metacells for each cell, producing a new protein activity matrix. Then, recluster the data using the same script as earlier in the pipeline (see step 3, 3.5, and 4 for walkthroughs on these steps).

STEP 8: Master Regulators

Finally, we want to find the master regulators for each cluster, using stouffer integration.

Run the following command (for each cluster):

```
Rscript mrAnalysis.R --activity_file=cluster_proteinActivity.rds --out_name=cluster-name_MRs.txt --out_dir=YOUR-OUT-DIRECTORY
```