CIA project

Sandra Lilja, Martin Smelink, Xinxiu Lee, Danuta Gawel, Oleg Sysoev

07/04/2022

scRNA-seq analysis for construction of MCDMs, MO-MCDMs, and Connective Pathway Analysis

Define data structure and input files

```
if (dir.exists("data/IPA/curation")==FALSE){
  dir.create("data/IPA/curation", recursive = T)
}
if (dir.exists("data/sorted_DGEs")==FALSE){
  dir.create("data/sorted_DGEs")
}
if (dir.exists("data/NicheNet_analysis_curated")==FALSE){
  dir.create("data/NicheNet_analysis_curated")
}
if (dir.exists("data/clustering_and_celltype_identification")==FALSE){
  dir.create("data/clustering_and_celltype_identification")
}
if (dir.exists("data/scVI_normalized")==FALSE){
  dir.create("data/scVI_normalized")
}
```

The expected data input are five csv.gz matrices, one per organ, with cells over columns and genes over rows.

```
### Read data from GSE and save it to data/
input_file = "data/UMI_expression_matrix.txt.gz"
```

Quality assessment and sorting

To ensure good quality data for downstream analyses, it is recommended to remove poor quality cells and genes from the analyses. We performed these analyses on the input_file, for all the organs combined. We define and keep the good quality cells as those having a minimum of 400 transcripts, 200 genes, and less than 20% mitochondrial genes. The good quality genes kept were defined as those being identified in at least 1% of the cells.

Due to the risk of duplicates in the library resulting in two or more cells sharing a cell barcode, it is also recommend to remove outliers. Based on empirical evaluation of the distribution an overestimation of transcripts count over the cells, we removed all cells with >6000 transcripts.

The output file is saved in data/sorted_DEGs.

```
source('R/sc_data_quality_sorting.R')
sc_data_quality_sorting(input_file)
```

Clustering and cell type identification by Leiden's algorithm

```
#module add anaconda3
#source activate dca
python3 Python/ConvertReferencesSandra.ipynb
#source deactivate
```

Cell type identification using marker genes

As the reference used for cell type identification in above method does not contain all cell types, some clusters remained unidentified after this analysis. To verify the cell types which were identified, as well as to further identify the unknown clusters, we looked for known marker genes among the cell types/clusters.

DEGs were firsts calculated between each cell type/cluster (as identified by the Leiden's algorithm) described above and all other cell types in the data, identifying the cell type specific marker genes. See a detailed description of the DEG analysis below.

```
module add anaconda3
source activate cia # if needed, define user-specific python environment

# define in-/output directories and files
infile=data/sorted_DGEs/sorted_expression_matrix.csv
outdir=data/scVI_normalized/
clust_file=data/clustering_and_celltype_identification/cluster_ids.csv
outdir_DEG=data/DEG_analysis/cellTypeX_vs_allOtherCellTypes
mkdir -p $outdir_DEG
mkdir -p $outdir_DEG/change_mode
# Calculate DEGs for each cell type and organ between the cell type and all other cell types
python scVI_v0.7.1.py --infile $infile --outdir $outdir --n_epochs 400 --DEG_analysis --clust_file $clus
wait
```

The significant DEGs were then identified based on fdr corrected p-velues, as described below.

```
source('R/DEG_sort_significant.R')
outdir_DEG <- 'data/DEG_analysis/cellTypeX_vs_allOtherCellTypes'
DEG_files <- list.files(outdir_DEG, full.names = TRUE, pattern = 'DEGs_')
# subset the DEGs
DEG_sort_significant(DEG_files, outdir_DEG)
```

We then searched for known marker genes among the significantly DEGs in each cell type/cluster. The marker genes which we used for cell type verification/identification for each cluster can be seen in Table 1.

Table 1. Marker genes used for cell type identification.

Data normalization and Differential expression analysis

For each cell type in each organ, we calculate the DEGs between sick and healthy mice using scVI-tools (version 0.7.1)[1]. The data was trained and normalized, including batch effect removal based on the individual samples sequenced (i.e. between organs and mice IDs). DEGs were then calculated between the two groups including the parameter mode='change'.

Note that the training will produce somewhat different output from each run, so in case the analyses need to be reproduced, the trained data is saved to 'outdir', '/full_posterior_400/model_params'. If this directory exists, the trained data will be loaded from there. As the training can be time consuming, this also saves computational time. Note also that the training results will depend on the previous set-ups, and if any of the parameters 'n_epochs', 'use_batches', or 'batch_id' are changed, the training need to be re-run.

[1]Gayoso, Adam, et al. A Python library for probabilistic analysis of single-cell omics data. *Nature Biotechnology* 40.2: 163-166, doi:10.1038/s41587-021-01206-w (2022).

** Not provided yet, too big?** To reproduce our results, use the trained data provided in 'data/scVI_normalized/full_posterior_400/model_params'.

```
module add anaconda3
source activate cia # if needed, define user-specific python environment

# define in-/output directories and files
infile=data/sorted_DGEs/sorted_expression_matrix.csv
outdir=data/scVI_normalized/
clust_file=data/clustering_and_celltype_identification/cluster_ids.csv
outdir_DEG=data/DEG_analysis/sick_vs_healthy
mkdir -p $outdir_DEG
mkdir -p $outdir_DEG/change_mode

# Calculate DEGs for each cell type and organ, between sick and healthy mice
python3 Python/scVI_v0.7.1.py --infile $infile --outdir $outdir --n_epochs 400 --DEG_analysis --clust_f
wait
```

The output from this code consists of one table of DEGs for each cell type and organ as defined by 'sort_column', saved in 'outdir_DEGs'. These files contains values which describes the data; scale values from the training, log fold-change values, non-zero proportions, and raw normalized mean values, for group 1 and group 2 respectively. The group-ids can be interpreted based on the output file names, group1_vs_group2 (eg. sick_vs_healthy). We define the significant DEGs based on the column 'is_de_fdr_0.05'=True.

Additionally, the normalized and latent data is saved to 'outdir' for downstream analyses.

```
module add anaconda3
source activate cia # if needed, define user-specific python environment
python3 Python/scVI_v0.7.1.py --help
```

```
## usage: scVI_v0.7.1.py [-h] --infile INFILE --outdir OUTDIR [--use_batches]
##                       [--batch_id BATCH_ID [BATCH_ID ...]]
##                       [--n_epochs N_EPOCHS] [--cluster_analysis]
##                       [--resolution RESOLUTION] [--DEG_analysis]
##                       [--clust_file CLUST_FILE] [--outdir_DEG OUTDIR_DEG]
##                       [--sort_column SORT_COLUMN [SORT_COLUMN ...]]
##                       [--group_column GROUP_COLUMN [GROUP_COLUMN ...]]
##
## Analyse single cell data using scVI
##
## optional arguments:
##   -h, --help            show this help message and exit
##   --infile INFILE       Relative path to the input data. The input file need
##                         to be a comma separated csv file, with cells as
##                         columns and genes as rows
##   --outdir OUTDIR       The path to the cluster_analysis output directory
##   --use_batches         add if normalization over batches are needed.
##   --batch_id BATCH_ID [BATCH_ID ...]
##                         Define which parts of the colnames (infile) should be
##                         used to define batches (count from 0, sep = "_"). eg
##                         cellname; "BatchID_cell1". Default = 0. If multiple
##                         sources of batch effect, input a space-separated list
##                         of integers
##   --n_epochs N_EPOCHS   This argument will define the amount of training and
##                         should depend on the number of cells in the input
##                         data. Recommended amount is 100 or more. Default =
```

```
##                         400.
##   --cluster_analysis    add if you want to perform the cluster analysis.
##   --resolution RESOLUTION
##                         This argument will define the resolution of
##                         clustering. Default = 0.5. Only used if
##                         cluster_analysis == True
##   --DEG_analysis        add this command if you want to perform the
##                         differential expression analysis.
##   --clust_file CLUST_FILE
##                         Relative path to the cluster data file. Required for
##                         --DEG_analysis
##   --outdir_DEG OUTDIR_DEG
##                         The path to the DEG_analysis output directory.
##                         Required for --DEG_analysis
##   --sort_column SORT_COLUMN [SORT_COLUMN ...]
##                         Column in clust_file to sort the data for DEG
##                         analysis. DEGs will be calculated for each unique
##                         variable in this column separetly. (count first column
##                         as 0). If multiple sources to sort on, input a space-
##                         separated list of integers
##   --group_column GROUP_COLUMN [GROUP_COLUMN ...]
##                         Column in clust file containing the groups between
##                         which to identify DEGs. If the column contains more
##                         that two groups, DEGs will be calculated for each
##                         unique value vs all others. (count first column as 0)
```

We further subset these tables only to include significant DEGs, and columns relevant for downstream analyses. The output from this analysis consists of one table of significant DEGs for each input table of DEGs, which are saved to '[outdir_DEG]/subset_significant' (Table 2).

```
source('R/DEG_sort_significant.R')
outdir_DEG <- 'data/DEG_analysis/sick_vs_healthy/change_mode'
DEG_files <- list.files(outdir_DEG, full.names = TRUE, pattern = 'DEGs_')
# subset the DEGs
DEG_sort_significant(DEG_files, outdir_DEG)
```

```
# Example output from DEG_sort_significant.R
Joint_Mac <- read.csv('data/DEG_analysis/sick_vs_healthy/change_mode/subset_significant/significant_DEGs
knitr::kable(Joint_Mac[1:5,], caption = "Table 2. Example output from differential expression analysis"
```

Table 1: Table 2. Example output from differential expression analysis

	proba_de	proba_not_de	bayes_factor	lfc_mean	is_de_fdr_0.05
Ppp2r5d	0.9958	0.0042	5.468459	6.584303	True
Ifi27	0.9918	0.0082	4.795386	3.217298	True
Snx20	0.9908	0.0092	4.679308	4.120939	True
Slc25a20	0.9898	0.0102	4.575114	-4.061067	True
Bag6	0.9874	0.0126	4.361378	3.659021	True

Identification of cellular interactions and MCDM/MO-MCDM construction

To identify cell-cell interactions, we used NicheNet[2], a tool to model intercellular communications. To define the genes of interest for identification of the intercellular interactions, we used the lists of DEGs for each cell type in each organ. The interactions were thus predicted, between each pair of cell types and each organ separately.

As the NicheNet database consists of intercellular interactions in humans, the gene names were translated to their human orthologs. The translation file used for our studies is provided in /data/orthologous_translation_file.txt. To run this code on any other dataset, it is recommended to download the orthologs from Ensembl as described here.

The input files to run NicheNet_analysis.R, are;

- the normalized expression data. Output from scVI_v0.7.1.py
- the cell type identification file. Output from cell type analysis script
- a translation file of human orthologs
- a matrix listing all DEGs for each cell type and organ combination, with celltype_organ over columns, as output from the differential expression analysis, DEG_sort_significant.R.

The code output one tab separated txt file per interacting cell type pair, within and between organs, containing "test_ligand" (i.e upstream regulator of interaction), "auroc", "aupr", "pearson" (Pearson Correlation Coefficient, PCC), "target" (genes), and "target_weight" over the columns and interactions over rows.

Additionally, it creates one file containing all predicted interactions between each pair of cell types, all_ligand_activity.txt, which apart from above mentioned columns contains "Sender" and "Target" cell type and organ (named: celltype_organ)

Note: that the interactions between organs in these files are not yet curated only to include URs secreated in blood

To curate the inter-organ interactions, only to include interactions through URs secreted in blood, and to sort out the strongest interactions (PCC > 0) considered for further analyses we ran NicheNet_network_curation.R. As input to this script is the all_ligand_activity.txt output from NicheNet_analysis.R, and a curation file (eg., data/IPA/curation/curation_file.txt) containing information about the cellular location of each UR.

The curation file can be retrieved from IPA by following IPA - Generate curation file

It is important that the curation file contains two columns named "Symbol" (containing the gene names of the URs) and "Location" (containing information about cellular location). The URs which will be kept for inter-organ interactions by the code are those with "Location" == "Extracellular Space".

The output from this script consists of one file containing all inter- and intra-organ interactions, all_pos_curated_ligand_activity.txt (Fig 1)

Ranking of URs based on their downstream effect

/codes/rank_by_targets_and_heatmap.R

[2]Browaeys, R., Saelens, W. & Saeys, Y. NicheNet: modeling intercellular communication by linking ligands to target genes. *Nature Methods* 17, 159-162, doi:10.1038/s41592-019-0667-5 (2020).