5-Single-cell RNA velocity estimation

BAI Qiang[*]

2021-09-29 16:22:32 +0200

Contents

[*]University Liege, mail qiang.bai@uliege.be

1 Description

For each sample, the counts for unspliced- and ambiguous transcripts were calculated from CellRanger output using velocyto command-line tool (http://velocyto.org)[1] and saved in loom files. The single-cell RNA velocities were estimated using scVelo toolkit (https://scvelo.readthedocs.io)[2]. Briefly, the loom files were used as input for scVelo analysis. Genes with minimum 20 of both unspliced and spliced counts and on the top list of 2000 genes were filtered, normalized and log transformed (scv.pp.filter_and_normalize with default parameters). Thirty principal components (PCs) and 30 neighbors obtained from euclidean distances in PCA space were used for computing first-/second-order moments for each cell. We used generalized dynamical modeling to recover the full splicing kinetics of spliced genes and the single-cell RNA velocities were plotted with the same cluster labels and embedding as in Fig4A.

2 Load data and packages

```r
library(Seurat)
library(ggplot2)
library(dplyr)
library(loomR)
library(tidyverse)


seurat.combined <- readRDS(file = "../3-Merge␣and␣cell␣typing/so.merged_
   clusters.seuratObject.Rds")
```

3 Prepare data RNAvelocyto

Prepare individual Seurat objects for each sample.

```r
list.name.so <- unique(seurat.combined$origin)
list.name.sample <- list.name.so

for (i in 1:length(list.name.so)) {
  so <- seurat.combined[, seurat.combined$origin == list.name.so[i]]
  assign(paste(list.name.sample[i], "seuratObject", sep = "."), so)
}

list.name.so <- paste(list.name.sample, "seuratObject", sep = ".")

obj.list <- list()
for (name.so in list.name.so) {
  obj.list <- c(obj.list, get(name.so))
}
list.name.so <- sub("␣", "_", list.name.so)
names(obj.list) <- list.name.so
```

3.1 Generate loom files:

The intermediate loom files were too big to be uploaded to the platform but they can be produced by the following steps.

We counted unspliced- and ambiguous transcripts using velocyto command-line tool (http://velocyto.org)[1].

For each sample, the following code was used to generate the loom file:

```
velocyto run    -b "${sampleID}/outs/filtered_feature_bc_matrix/barcodes.
   tsv.gz" \
                    -o "outputDir/${sampleID}.loom" \
                    "${sampleID}/outs/possorted_genome_bam.bam" \
                    /refdata-cellranger-GRCh38-3.0.0/genes/genes.gtf
```

- `${sampleID}` is the sample ID.
- `${sampleID}/outs` is the output directory of CellRanger.
- `${sampleID}/outs/possorted_genome_bam.bam` is the BAM file generated from CellRanger.
- `/refdata-cellranger-GRCh38-3.0.0/genes/genes.gtf` is the gene reference used for Cellranger counts.

3.2 Read loom files and prepare cellnames

```
suppressMessages(library("velocyto.R"))
list.path.loom <- list.dirs("outputDir/loom")
```

Read loom files and create loom objects under sample names

```
list.path.loom <- list.path.loom[-1] # remove the first entry which is
   parent directory

list.name.loom <- basename(list.path.loom)
list.name.loom <- str_replace(list.name.loom, pattern = "-", replacement =
    "_")
list.path.loom <- list.files(list.path.loom, pattern = "\\.loom$", full.
   names = TRUE)

for (i in 1:length(list.name.loom)) {
  assign(make.names(list.name.loom[i]), read.loom.matrices(list.path.loom[
     i]))
}
```

```
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
## reading loom file via hdf5r...
```

```
list.name.loom <- make.names(list.name.loom)
```

Make cell names consistent in both loom objects and Seurat objects.

```
# A prefix was added to each cell in Seurat objects during merge step. We
   should also add prefix to each cell.
```

```
prefix <- str_remove(list.name.loom, pattern = ".loom")

# Add prefix to cellnames.
source("../../R/aggregateLoom.R")
for (i in 1:length(list.name.loom)) {
  loom <- get(list.name.loom[i])
  assign(list.name.loom[i], value = aggregateLoom(loom, Ori.ID = prefix[i
     ]))
}
```

3.3 Filter cellnames and feature names in loom with Seurat gene/cell list

As the Seurat object contains only filtered cells and genes, the genes and cells in loom files should be also filtered.

```
source("../../R/filterLoom.R")

ldat.list <- list()

for (name.sample in list.name.sample) {
  obj.name <- paste(name.sample, "seuratObject", sep = ".")
  loom.name <- paste(name.sample, "loom", sep = ".")
  ldat.name <- paste(name.sample, "ldat", "filtered", sep = ".")

  assign(ldat.name,
         value = filterLoom(loomObj = get(loom.name),
                            geneList = rownames(obj.list[[obj.name]]),
                            cellList = colnames(obj.list[[obj.name]]))
                               )
}
```

```
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
## [1] "Following genes are not in the loom gene list:"
##  [1] "RGS5.1"      "TBCE.1"      "LINC01238.1" "CYB561D2.1"  "ATXN7.1"
##  [6] "CCDC39.1"    "MATR3.1"     "POLR2J3.1"   "ABCF2.1"     "TMSB15B
   .1"
## [11] "PINX1.1"     "HSPA14.1"    "EMG1.1"      "DIABLO.1"    "COG8.1"
```

All the samples have the same unfound genes (15 genes with redundant symbols).

Remove the redundant genes:

```
for (name.sample in list.name.sample) {
  obj.name <- paste(name.sample, "seuratObject", sep = ".")
  loom.name <- paste(name.sample, "loom", sep = ".")
  ldat.name <- paste(name.sample, "ldat", "filtered", sep = ".")
  genes.toRemove <- get(paste(name.sample, "ldat.filtered", sep = "."))

  genes <- rownames(obj.list[[obj.name]])
  genes.new <- genes[-which(genes %in% genes.toRemove)]

  assign(ldat.name,
         value = filterLoom(loomObj = get(loom.name),
                            geneList = genes.new,
                            cellList = colnames(obj.list[[obj.name]])
                               ))

  }
```

```
## [1] "Following cells are not in the loom cell list:"
## [1] "LBA_Hum3_GACGCTGAGGATTTAG" "LBA_Hum3_GAGTTACCACGTGTGC"
## [1] "Following cells are not in the loom cell list:"
## [1] "LBA_Hum4_AGACAGGTCTGGACCG" "LBA_Hum4_AGTACTGTCAACCTTT"
## [1] "Following cells are not in the loom cell list:"
## [1] "LBA_Hum5_AACCATGGTAGGTGCA" "LBA_Hum5_ACCGTTCAGGCGTCCT"
## [3] "LBA_Hum5_AGTAACCGTCGAAGCA"
## [1] "Following cells are not in the loom cell list:"
## [1] "LBA_Hum8_GCACATACACGCTATA" "LBA_Hum8_GGAGCAAGTTCAGCGC"
## [1] "Following cells are not in the loom cell list:"
##  [1] "LBA_Hum9_AACCTTTCAACCAATC" "LBA_Hum9_AATCGACCACGAGGTA"
##  [3] "LBA_Hum9_ACCAAACGTGTCTTAG" "LBA_Hum9_AGCCAATTCGCTGACG"
##  [5] "LBA_Hum9_AGCTTCCTCTCATAGG" "LBA_Hum9_ATGGGAGTCAAGAATG"
##  [7] "LBA_Hum9_CGGGCATGTTCTCTAT" "LBA_Hum9_GACCCTTTCTACCTTA"
##  [9] "LBA_Hum9_TAGGTACCACATGACT" "LBA_Hum9_TTAATCCGTCCTACGG"
## [11] "LBA_Hum9_TTCTCTCAGCAGCCCT"
## [1] "Following cells are not in the loom cell list:"
## [1] "LBA_Hum10_AGTAACCAGGAGCTGT" "LBA_Hum10_GAAGAATTCCCAGTGG"
## [3] "LBA_Hum10_GTTTACTTCGCTGTTC"
## [1] "Following cells are not in the loom cell list:"
## [1] "LBA_Hum11_CGAATTGCAATCGTCA" "LBA_Hum11_CTCAACCGTCCGACGT"
## [1] "Following cells are not in the loom cell list:"
## [1] "LBA_Hum12_AATGGCTGTGAATTGA" "LBA_Hum12_TGGGCTGGTCTTGCGG"
## [3] "LBA_Hum12_TGTGGCGAGGACGCTA" "LBA_Hum12_TTGTTCACATGGCGCT"
```

Except LBA_Hum7, other loomsome cells are not in other loom files. So remove them.

Remove cells that not in Seurat object.

```
# important to skip the all-done sample:
list.name.sample.remained <- list.name.sample[!list.name.sample == "LBA_
   Hum7"]


for (name.sample in list.name.sample.remained) {
  obj.name <- paste(name.sample, "seuratObject", sep = ".")
  loom.name <- paste(name.sample, "loom", sep = ".")
  ldat.name <- paste(name.sample, "ldat", "filtered", sep = ".")
  cells.toRemove <- get(paste(name.sample, "ldat.filtered", sep = "."))

  cellnames <- colnames(obj.list[[obj.name]])
  cellnames.new <- cellnames[-which(cellnames %in% cells.toRemove)]

  assign(ldat.name,
         value = filterLoom(loomObj = get(loom.name),
                            geneList = genes.new,
                            cellList = cellnames.new ))

  # now we have to also filter seurat object by new cells:
  assign(obj.name,
         value = obj.list[[obj.name]] [ , cellnames.new] )
  }
```

Make list of Seurat objects and ldat objects, each under ther sample same.

```
obj <- list()

for (name.sample in list.name.sample) {
  obj.name <- paste(name.sample, "seuratObject", sep = ".")
  ldat.name <- paste(name.sample, "ldat", "filtered", sep = ".")

  tmp <- list(ldat = get(ldat.name), seurat = get(obj.name) )

  obj[[name.sample]] <- tmp
  }
```

Save for other analyses.

```
saveRDS(obj, file = "./obj.list.loom_surat.Rds")
```

3.4 Group loom/Seurat objects by treatment

Merge all Seurat objects to one, with only filtered cells. Merge all ldat objects to one, with only filtered cells.

```
obj.all <- obj
# now create merged seurat object and loom data.

# 1. merged seurat object.
list.name.sample <- names(obj.all)

seurat.all <- list()
ldat.all <- list()

for (sample.name in list.name.sample) {
  obj <- obj.all[[sample.name]]

  seurat.all[[sample.name]] <- obj[["seurat"]]
  ldat.all[[sample.name]] <- obj[["ldat"]]
}

cellnames <- character()
for (sample.name in list.name.sample) {
  obj <- seurat.all[[sample.name]]

  cellnames <- append(cellnames, colnames(obj))
}

seurat.merge <- seurat.combined[ , cellnames]
```

```
# 2. merged loom data;
# source("~/Desktop/velocyto/Script/aggregateLoom.R")

i=1
for (sample.name in list.name.sample) {

  obj <- ldat.all[[sample.name]]
  if (i==1) {
  spliced <- obj$spliced
  unspliced <- obj$unspliced
  ambiguous <- obj$ambiguous
  } else {
    spliced <- cbind(spliced, obj$spliced)
    unspliced <- cbind(unspliced, obj$unspliced) # note: previous code
       here was wrong.
    ambiguous <- cbind(ambiguous, obj$ambiguous) # note: previous code
       here was wrong.
  }

  i=1+1
}

ldat.merge <- list(spliced=spliced,
                   unspliced=unspliced,
                   ambiguous=ambiguous)
```

Now separate them by group.

```
groupBy <- "group"
sample.groupBy <- unique(seurat.merge@meta.data[[groupBy]])

obj <- list()
for (sample.name in sample.groupBy) {
  seurat <- seurat.merge[ , seurat.merge@meta.data[[groupBy]] == sample.
     name]
  cellnames <- colnames(seurat)
  ldat <- list(spliced = ldat.merge$spliced[, cellnames],
               unspliced = ldat.merge$unspliced[, cellnames],
               ambiguous = ldat.merge$ambiguous[, cellnames])
  obj[[sample.name]] <- list(ldat=ldat,
                             seurat=seurat)
}
```

OPTIONAL: save data for other presentations

```
saveRDS(obj, file = "./obj_group_by_group.list.loom_surat.Rds")
```

4 scVelo analysis

4.1 Correct NA in Seurat Metadata

```
 obj <- lapply(obj, function(x)
{
  obj <- x[["seurat"]]
  ldat <- x[["ldat"]]
    for(j in 1:ncol(obj@meta.data)){
          if(is.factor(obj@meta.data[,j]) == T){
          obj@meta.data[,j][is.na(obj@meta.data[,j])] <- "N.A"
      }
          if(is.character(obj@meta.data[,j]) == T){
          obj@meta.data[,j][is.na(obj@meta.data[,j])] <- "N.A"
      }
      }
  x[["seurat"]] <- obj
  x[["ldat"]] <- ldat

  return(x)
}
  )
```

4.2 Make loom file from Seurat/loom object

To facilitate the work, we optimized Seurat `Convert` function to merge a Seurat/Loom list to one Loom file, containing the var matrix with spliced, unspliced layers and obs with all embedding, tsne, umap, pca, clustering, etc. A Loom file issue from the function above `Convert.seurat_loom` will be saved in the current working folder.

For "Non fumer"

```
library(loomR)
source("../../R/Convert_Seurat_loom.R")
obj.sl <- obj$`No fumer`
pfile <- Convert.seurat_loom(from = obj.sl, to = "loom", filename = "No_Fumer.loom" )
pfile$close_all()
```

For "Fumer":

```
obj.sl <- obj$Fumer
pfile <- Convert.seurat_loom(from = obj.sl, to = "loom", filename = "Fumer.loom" )
pfile$close_all()
```

For "COPD":

```
obj.sl <- obj$COPD
pfile <- Convert.seurat_loom(from = obj.sl, to = "loom", filename = "COPD.loom" )
pfile$close_all()
```

4.3 scVelo analysis with dynamical model

For the details in the estimation of single-cell RNA velocity using dynamical model, refer to the original report[2]:

Bergen, V., Lange, M., Peidli, S., Wolf, F. A. & Theis, F. J. Generalizing RNA velocity to transient cell states through dynamical modeling. Nat. Biotechnol. (2020) doi:10.1038/s41587-020-0591-3.

The following codes were used to calculate scRNA velocity and presenting with the existing embedding and labels.

4.3.1 For "Non-fumer"

```
# python below
import scvelo as scv
scv.settings.verbosity = 3  # show errors(0), warnings(1), info(2), hints(3)
scv.settings.presenter_view = True  # set max width size for presenter view
scv.set_figure_params('scvelo')  # for beautified visualization

# load data
ldata_basal = scv.read("./No_Fumer.loom")

# Preprocess the Data
```

```
scv.pp.filter_and_normalize(ldata_basal, min_shared_counts=20, n_top_genes
   =2000)
scv.pp.moments(ldata_basal, n_pcs=30, n_neighbors=30)

# Estimate RNA velocity with dynamical model
scv.tl.recover_dynamics(ldata_basal)
scv.tl.velocity(ldata_basal, mode='dynamical')
scv.tl.velocity_graph(ldata_basal)
scv.pl.velocity_embedding_stream(ldata_basal, basis='umap_cell_embeddings'
   , color='seurat_clusters',
                                  figsize=(10,10), components='1,2',
                                  palette=["#2E359A", "#FC990E", "#720D0D",
                                      "#6E9BD8"],
                                  linewidth=1.4,
                                  title="scVelo␣analysis", save="No_Fumer.
                                     png"
                                 )
```

4.3.2 For “Fumer”

```
# load data
ldata_basal = scv.read("./Fumer.loom")

# Preprocess the Data
scv.pp.filter_and_normalize(ldata_basal, min_shared_counts=20, n_top_genes
   =2000)
scv.pp.moments(ldata_basal, n_pcs=30, n_neighbors=30)

# Estimate RNA velocity with dynamical model
scv.tl.recover_dynamics(ldata_basal)
scv.tl.velocity(ldata_basal, mode='dynamical')
scv.tl.velocity_graph(ldata_basal)
scv.pl.velocity_embedding_stream(ldata_basal, basis='umap_cell_embeddings'
   , color='seurat_clusters',
                                  figsize=(10,10), components='1,2',
                                  palette=["#2E359A", "#FC990E", "#720D0D",
                                      "#6E9BD8"],
                                  linewidth=1.4,
                                  title="scVelo␣analysis", save="Fumer.png"
                                 )
```

4.3.3 For “COPD”

```
# load data
ldata_basal = scv.read("./COPD.loom")

# Preprocess the Data
scv.pp.filter_and_normalize(ldata_basal, min_shared_counts=20, n_top_genes
   =2000)
scv.pp.moments(ldata_basal, n_pcs=30, n_neighbors=30)

# Estimate RNA velocity with dynamical model
scv.tl.recover_dynamics(ldata_basal)
```

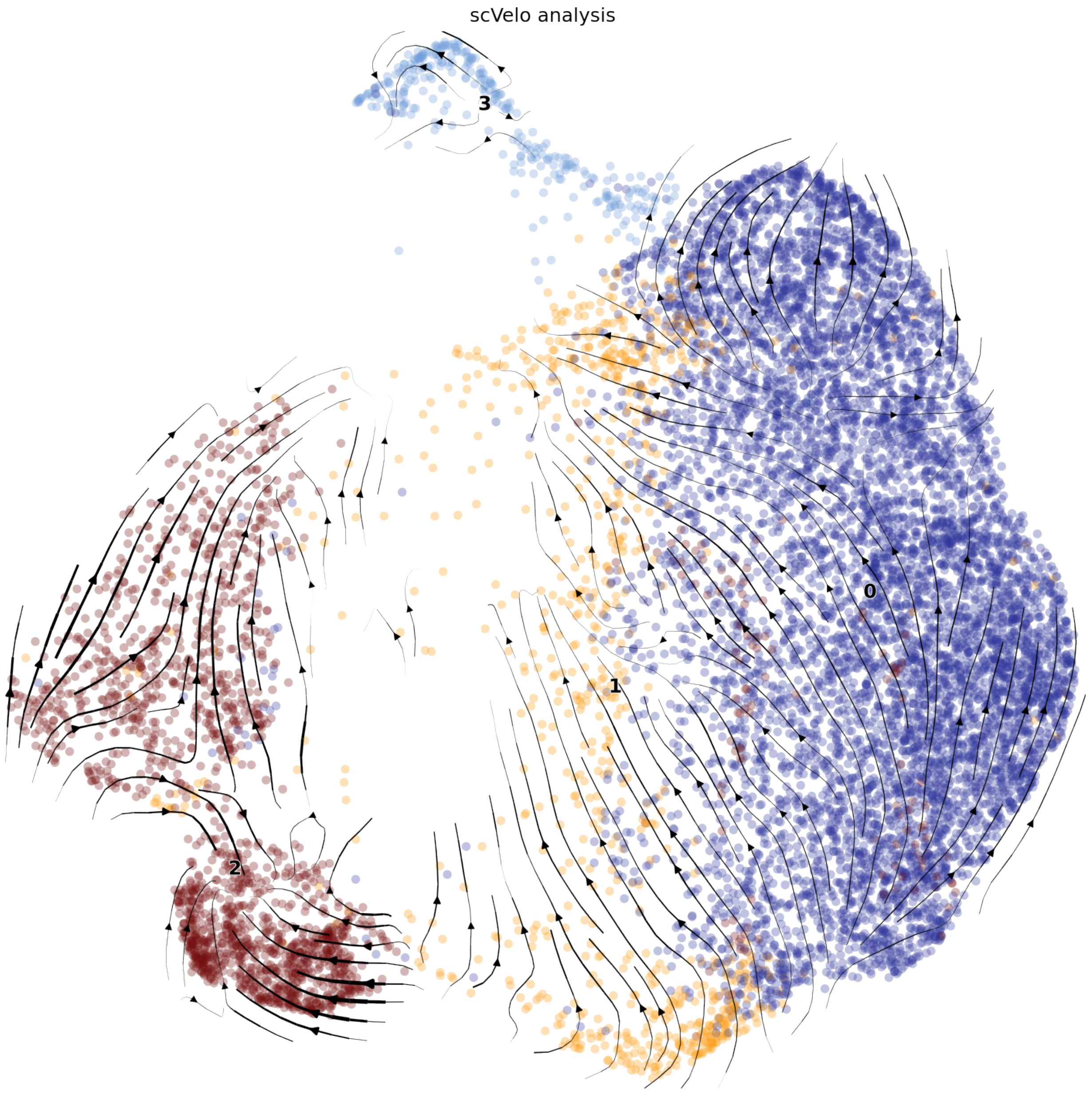

Figure 1: scVelo results for “Non fumer” grouped sample

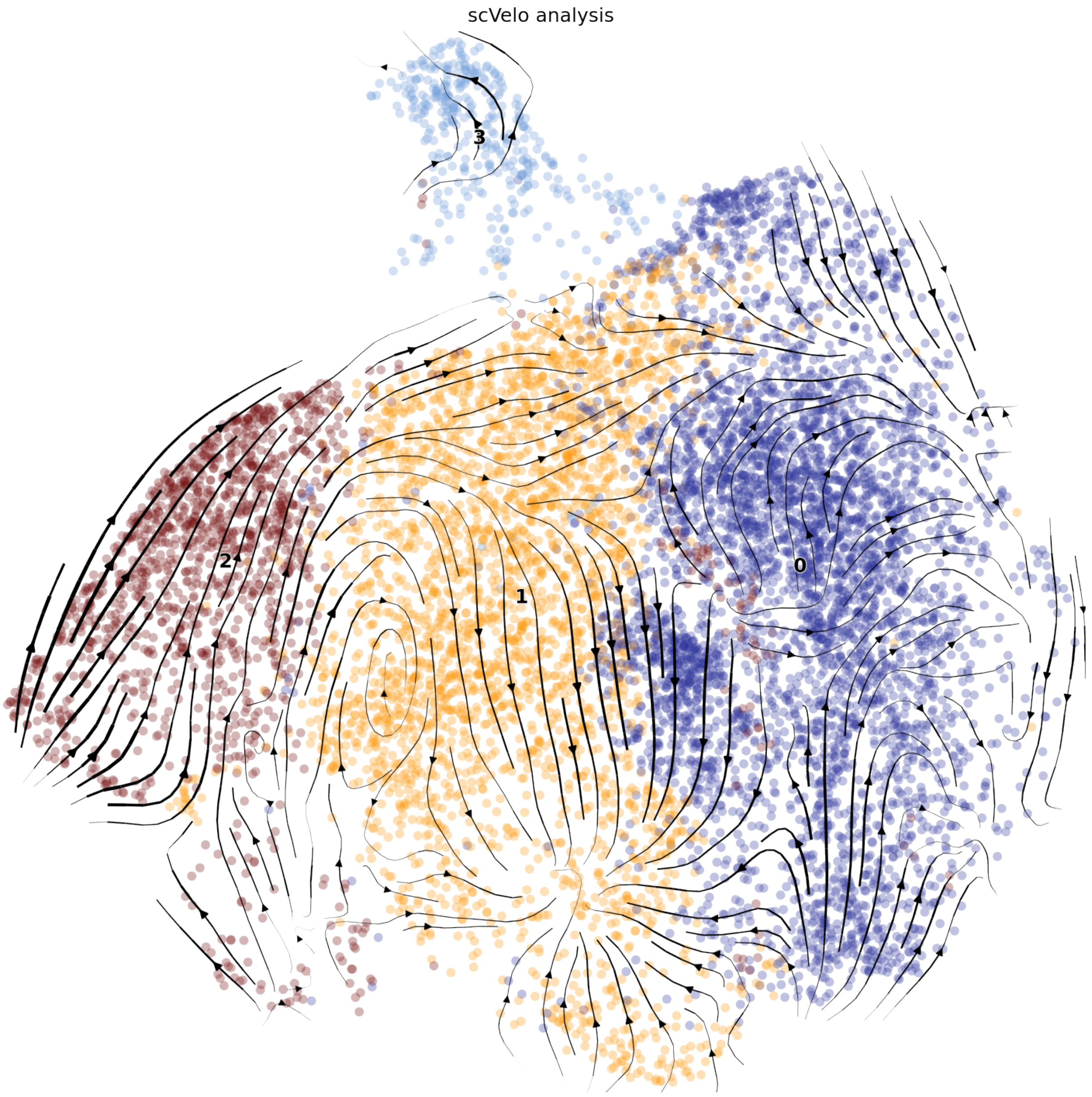

Figure 2: scVelo results for “Fumer” grouped sample

```
scv.tl.velocity(ldata_basal, mode='dynamical')
scv.tl.velocity_graph(ldata_basal)
scv.pl.velocity_embedding_stream(ldata_basal, basis='umap_cell_embeddings'
    , color='seurat_clusters',
                                  figsize=(10,10), components='1,2',
                                  palette=["#2E359A", "#FC990E", "#720D0D",
                                      "#6E9BD8"],
                                  linewidth=1.4,
                                  title="scVelo␣analysis", save="COPD.png"
                                 )
```

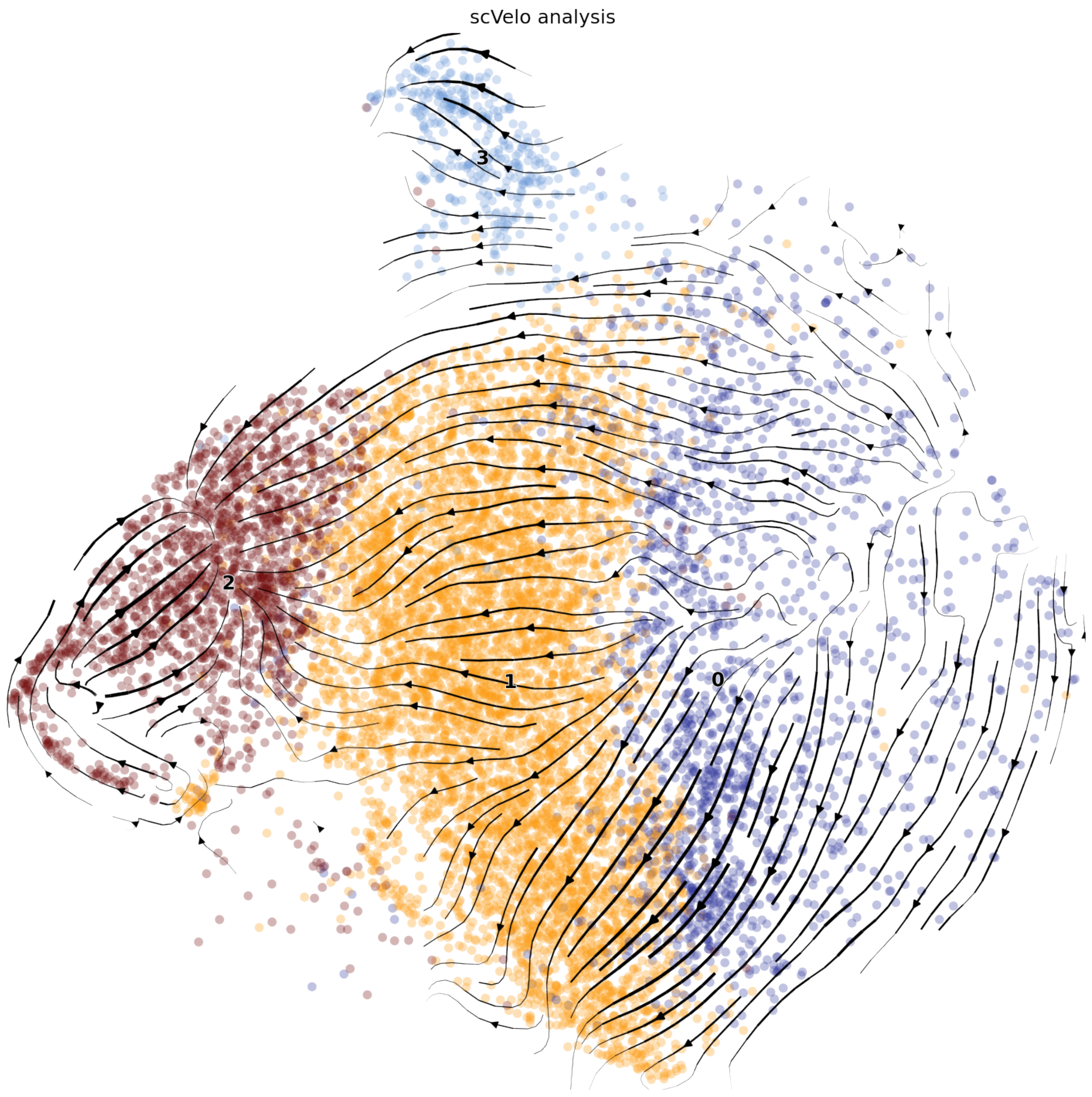

Figure 3: scVelo results for “COPD” grouped sample

5 Session information

R sesssion:

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] velocyto.R_0.6     Matrix_1.3-4       forcats_0.5.1      stringr_1
   .4.0
##  [5] purrr_0.3.4        readr_2.0.0        tidyr_1.1.3        tibble_3
   .1.3
##  [9] tidyverse_1.3.1    loomR_0.2.1.9000   hdf5r_1.3.3        R6_2.5.0
## [13] dplyr_1.0.7        ggplot2_3.3.5      SeuratObject_4.0.2 Seurat_4
   .0.3
##
## loaded via a namespace (and not attached):
##   [1] Rtsne_0.15            colorspace_2.0-2      deldir_0.2-10
##   [4] ellipsis_0.3.2        ggridges_0.5.3        fs_1.5.0
##   [7] rstudioapi_0.13       spatstat.data_2.1-0   leiden_0.3.9
##  [10] listenv_0.8.0         ggrepel_0.9.1         bit64_4.0.5
##  [13] lubridate_1.7.10      fansi_0.5.0           xml2_1.3.2
##  [16] codetools_0.2-18      splines_4.0.3         knitr_1.33
##  [19] polyclip_1.10-0       jsonlite_1.7.2        broom_0.7.9
##  [22] ica_1.0-2             dbplyr_2.1.1          cluster_2.1.0
##  [25] png_0.1-7             uwot_0.1.10.9000      shiny_1.6.0
##  [28] sctransform_0.3.2     spatstat.sparse_2.0-0 compiler_4.0.3
##  [31] httr_1.4.2            backports_1.2.1       assertthat_0.2.1
##  [34] fastmap_1.1.0         lazyeval_0.2.2        cli_3.0.1
##  [37] later_1.2.0           htmltools_0.5.1.1     tools_4.0.3
##  [40] igraph_1.2.6          gtable_0.3.0          glue_1.4.2
##  [43] RANN_2.6.1            reshape2_1.4.4        Rcpp_1.0.7
##  [46] Biobase_2.50.0        scattermore_0.7       cellranger_1.1.0
##  [49] vctrs_0.3.8           nlme_3.1-152          lmtest_0.9-38
##  [52] xfun_0.24             globals_0.14.0        rvest_1.0.1
##  [55] mime_0.11             miniUI_0.1.1.1        lifecycle_1.0.0
##  [58] irlba_2.3.3           goftest_1.2-2         future_1.21.0
##  [61] MASS_7.3-53          zoo_1.8-9             scales_1.1.1
##  [64] spatstat.core_2.3-0  pcaMethods_1.82.0     hms_1.1.0
##  [67] promises_1.2.0.1     spatstat.utils_2.2-0  parallel_4.0.3
##  [70] RColorBrewer_1.1-2   yaml_2.2.1            reticulate_1.20
##  [73] pbapply_1.4-3        gridExtra_2.3         rpart_4.1-15
##  [76] stringi_1.7.3        BiocGenerics_0.36.1   rlang_0.4.11
##  [79] pkgconfig_2.0.3      matrixStats_0.60.0    evaluate_0.14
##  [82] lattice_0.20-41      ROCR_1.0-11           tensor_1.5
##  [85] patchwork_1.1.1      htmlwidgets_1.5.3     cowplot_1.1.1
##  [88] bit_4.0.4            tidyselect_1.1.1      parallelly_1.27.0
##  [91] RcppAnnoy_0.0.19     plyr_1.8.6            magrittr_2.0.1
##  [94] generics_0.1.0       DBI_1.1.1             haven_2.4.3
##  [97] pillar_1.6.2         withr_2.4.2           mgcv_1.8-33
## [100] fitdistrplus_1.1-5   survival_3.2-7        abind_1.4-5
## [103] future.apply_1.7.0   modelr_0.1.8          crayon_1.4.1
## [106] KernSmooth_2.23-20   utf8_1.2.2            spatstat.geom_2.2-2
## [109] plotly_4.9.4.1       tzdb_0.1.2            rmarkdown_2.9
## [112] readxl_1.3.1         grid_4.0.3            data.table_1.14.0
## [115] reprex_2.0.0         digest_0.6.27         xtable_1.8-4
## [118] httpuv_1.6.1         munsell_0.5.0         viridisLite_0.4.0
```

velocyto version:

```
velocyto --version
```

```
velocyto, version 0.17.17
```

scVelo (python package)

```
import scvelo
print(scvelo.__version__)
```

```
0.2.3
```

References

1. La Manno G, Soldatov R, Zeisel A, Braun E, Hochgerner H, Petukhov V, Lidschreiber K, Kastriti ME, Lönnerberg P, Furlan A, Fan J, Borm LE, Liu Z, Bruggen D van, Guo J, He X, Barker R, Sundström E, Castelo-Branco G, Cramer P, Adameyko I, Linnarsson S, Kharchenko PV. RNA velocity of single cells. *Nature* 2018;

2. Bergen V, Lange M, Peidli S, Wolf FA, Theis FJ. Generalizing RNA velocity to transient cell states through dynamical modeling. *Nature Biotechnology* 2020;