Mafb-restricted local monocyte proliferation precedes lung interstitial macrophage differentiation

7-RNAvelocity and scVelo analysis

2022-03-09 15:56:01 +0100

Abstract

Resident tissue macrophages (RTM) are differentiated immune cells populating distinct niches and exhibiting important tissue-supportive functions. RTM maintenance is thought to rely on either monocyte engraftment and differentiation, or RTM self-renewal. Here, we developed an inducible mouse model of lung interstitial macrophage (IM) niche depletion and repopulation to investigate IM development in vivo. Using time-course single-cell RNA-sequencing analyses, bone marrow chimeras and gene targeting, we found that engrafted Ly6C+ classical monocytes could self-renew locally in a CSF1R-dependent manner before their differentiation into RTM. We further showed that the switch from monocyte proliferation towards IM subset specification was controlled by MafB, while c-Maf specifically regulated the identity of the CD206+ IM subset. Our data shed new light on the transcriptional regulation of IM development and provide evidence that, in the mononuclear phagocyte system, self-renewal is not merely restricted to myeloid progenitor cells and mature macrophages, but is also a tightly regulated capability of mature monocytes developing into RTM in vivo.

Contents

1 Description

To confirm the differentiation from classical monocytes to IM passing by the transit cells, the cell fate or differentiation direction of each single cells were predicted by RNA velocity. Briefly, the counts for spliced-, 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] 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 Figure 4B.

2 Load data

```r
# packages
suppressMessages(library(Seurat))
suppressMessages(library(tidyverse))
suppressMessages(library(RColorBrewer))

obj3d.combined <- readRDS("../6-Merge_two_experiments/immune_imdtr3.
   seuratObject.Rds")
```

3 Prepare data from Seurat object for RNAvelocyto

```r
list.name.so <- unique(obj3d.combined$treatment)
list.name.sample <- list.name.so

for (i in 1:length(list.name.so)) {
  so <- obj3d.combined[, obj3d.combined$treatment == list.name.so[i]]
  assign(paste(list.name.sample[i], "seuratObject", sep = "."), so)
}

list3d.name.so <- paste(list.name.sample, "seuratObject", sep = ".")

obj.list <- list()
for (name.so in list3d.name.so) {
  obj.list <- c(obj.list, get(name.so))
}

names(obj.list) <- list.name.so
```

4 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 spliced, 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" \
                       /GRCm38/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.
- `/GRCm38/genes/genes.gtf` is the gene reference used for Cellranger counts.

5 Add cell spliced/unspliced counts to Seurat objects

```
suppressMessages(library("velocyto.R"))
```

5.1 Load IM-DTR1 loom data (Exp1)

```
list.path.loom <- list.dirs("../IM-DTR1/counts/loom")
list.path.loom <- list.path.loom[-1] # remove the first parent directory: Counts/loom/
```

```
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...
```

5.2 Load IM-DTR2 loom data (Exp2)

```
list.path.loom <- list.dirs("../IM-DTR2/counts/loom")
list.path.loom <- list.path.loom[-1] # remove the first parent directory: Counts/loom/
```

```
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...
```

6 Make consistancy of cell names in Loom and Seurat object

6.1 For 4-day and 0-day data

for the 4d and 0d data, we added "Cplus_" and "Plusplus_":

```
colnames(`4d.seuratObject`)[1:2]
```

```
## [1] "CPlus_AAACCCAAGAGGTCAC-1" "CPlus_AAACCCATCATCCTAT-1"
```

```
colnames(`0d.seuratObject`)[1:2]
```

```
## [1] "Plusplus_AAACCCAAGACATAAC-1" "Plusplus_AAACCCAAGGCATTTC-1"
```

Remove the "-1" from cell names:

```
obj.list$`4d` <- RenameCells(obj.list$`4d`,
                                    new.names = sub(colnames(obj.list$`4d
                                       `), pattern = "-1", replacement = "
                                       "))

obj.list$`0d` <- RenameCells(obj.list$`0d`,
                                    new.names = sub(colnames(obj.list$`0d
                                       `), pattern = "-1", replacement = "
                                       "))
```

add the prefix to loom file:

```
prefix <-  c("CPlus", "Plusplus")

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
     ]))
}
```

Filter loom with Seurat gene/cell list

```
`0d.loom` <- Plusplus_NGS20_Q148.loom
`4d.loom` <- CPlus_NGS20_Q147.loom

list.name.sample2 <- c("0d", "4d")
```

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

# list.name.ldat <- paste(list.name.sample, "ldat", "filtered", sep = ".")
ldat.list <- list()

```

```
for (name.sample in list.name.sample2) {
  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]]))
                            )

}
```

6.2 For IM-DTR2 data

For the 12h, 48h and 24h data, we don't have prefix in Seurat object (no integration done)

```
colnames(obj.list$`2d`)[1:2]
```

```
## [1] "AAACCCAAGAGATTCA-1" "AAACCCAAGGACTTCT-1"
```

```
colnames(obj.list$`0.5d`)[1:2]
```

```
## [1] "AAACCCAAGCGTCGAA-1" "AAACCCAGTGTCCGTG-1"
```

```
colnames(obj.list$`1d`)[1:2]
```

```
## [1] "AAACCCAAGGGCTAAC-1" "AAACCCACACACAGCC-1"
```

Remove the "-1" from cell names:

```
obj.list$`0.5d` <- RenameCells(obj.list$`0.5d`,
                               new.names = sub(colnames(obj.list$`0.5
                                  d`), pattern = "-1", replacement =
                                  ""))

obj.list$`1d` <- RenameCells(obj.list$`1d`,
                               new.names = sub(colnames(obj.list$`1d
                                  `), pattern = "-1", replacement = "
                                  "))

obj.list$`2d` <- RenameCells(obj.list$`2d`,
                               new.names = sub(colnames(obj.list$`2d
                                  `), pattern = "-1", replacement = "
                                  "))
```

Loom file name check:

```
colnames(CD45Plus_NGS21_S229.loom$spliced)[1:2]
```

```
## [1] "CPlus_AAAGGATCACGTCATA" "CPlus_AAACGCTAGACTCTTG"
```

```
CD45Plus_NGS21_S229.loom <- aggregateLoom(CD45Plus_NGS21_S229.loom, Ori.ID
    = "")
```

Filter loom with Seurat gene/cell list

```
list.name.sample3 <- c("0.5d", "1d", "2d")
```

```
ldat.list <- list()

for (name.sample in list.name.sample3) {
  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 = CD45Plus_NGS21_S229.loom, # here we
             use only this object.
                                  geneList = rownames(obj.list[[obj.name]]),
                                  cellList = colnames(obj.list[[obj.name]]))
                                     )

}
```

7 Prepare data by group all the Seurat and Loom objects

7.1 Make list

```
list.name.sample <- c("0d", "0.5d", "1d", "2d", "4d")

obj.all <- list()

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

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

  obj.all[[name.sample]] <- tmp

  }
```

Save to file

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

7.2 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"]]
}

# the following way doesn't conserve the dimension reduction.
# seurat.merge <- merge(x = seurat.all[[1]],
#                       y = unlist(seurat.all[2:length(seurat.all)]),
#                       merge.data = TRUE)

# try to extract cellnames from the integrated seurat object:
# seurat.combined <- obj.combined # which is a seurat object.
#
# cellnames <- character()
# for (sample.name in list.name.sample) {
#   obj <- seurat.all[[sample.name]]
#
#   cellnames <- append(cellnames, colnames(obj))
# }

# seurat.merge <- seurat.combined[ , cellnames]
seurat.merge <- obj3d.combined
```

As the names were not changed in the original obj.combined, we change that manually:

```
# -1
  seurat.merge <- RenameCells(seurat.merge, new.names = sub(colnames(
     seurat.merge), pattern = "-1", replacement = ""))
```

Now merge the ldat:

```
for (sample.name in list.name.sample) {

  obj <- ldat.all[[sample.name]]
  if (sample.name == list.name.sample[1]) {
  spliced <- obj$spliced
  unspliced <- obj$unspliced
  ambiguous <- obj$ambiguous
  } else {
    spliced <- cbind(spliced, obj$spliced)
    unspliced <- cbind(unspliced, obj$unspliced)
    ambiguous <- cbind(ambiguous, obj$ambiguous)
  }

}

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

```
dim(ldat.all[[1]]$spliced)
```

```
## [1] 22597  2039
```

```
dim(ldat.merge$spliced)
```

```
## [1] 22597 47072
```

```
dim(ldat.merge$unspliced)
```

```
## [1] 22597 47072
```

```
dim(ldat.merge$ambiguous)
```

```
## [1] 22597 47072
```

7.3 Regroup data by treatment

```
groupBy <- "treatment"
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)
}
```

save the merged obj:

```
saveRDS(seurat.merge, file = "seurat3d.merge.seuratObject.Rds")
saveRDS(ldat.merge, file = "ldat.merge.ldat.Rds")
```

8 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.

```
# 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")
```

8.1 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)
```

8.2 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)
```

8.3 Plot single-cell RNA velocity with calculated embeddings

```
scv.pl.velocity_embedding_stream(ldata_basal,
    basis='umap_cell_embeddings', color='cell.type2', figsize=(10,10),
    palette=["#FDBF6F","#33A02C","#A6CEE3","#B2DF8A","#1F78B4","#E31A1C","
       #FB9A99"],
    linewidth=3, arrow_color="black",
    title="pc␣1,2", alpha = 0.9, legend_fontsize = 0,
    add_outline=True,
    components='1,2'
                                )
```

```
scv.pl.velocity_embedding_stream(ldata_basal,
    basis='umap_cell_embeddings', color='cell.type2', figsize=(10,10),
    palette=["#FDBF6F","#33A02C","#A6CEE3","#B2DF8A","#1F78B4","#E31A1C","
       #FB9A99"],
    linewidth=3, arrow_color="black",
    title="pc␣1,3", alpha = 0.9, legend_fontsize = 0,
    add_outline=True,
    components='1,3'
                                )
```

9 Session information

```
sessionInfo()
```

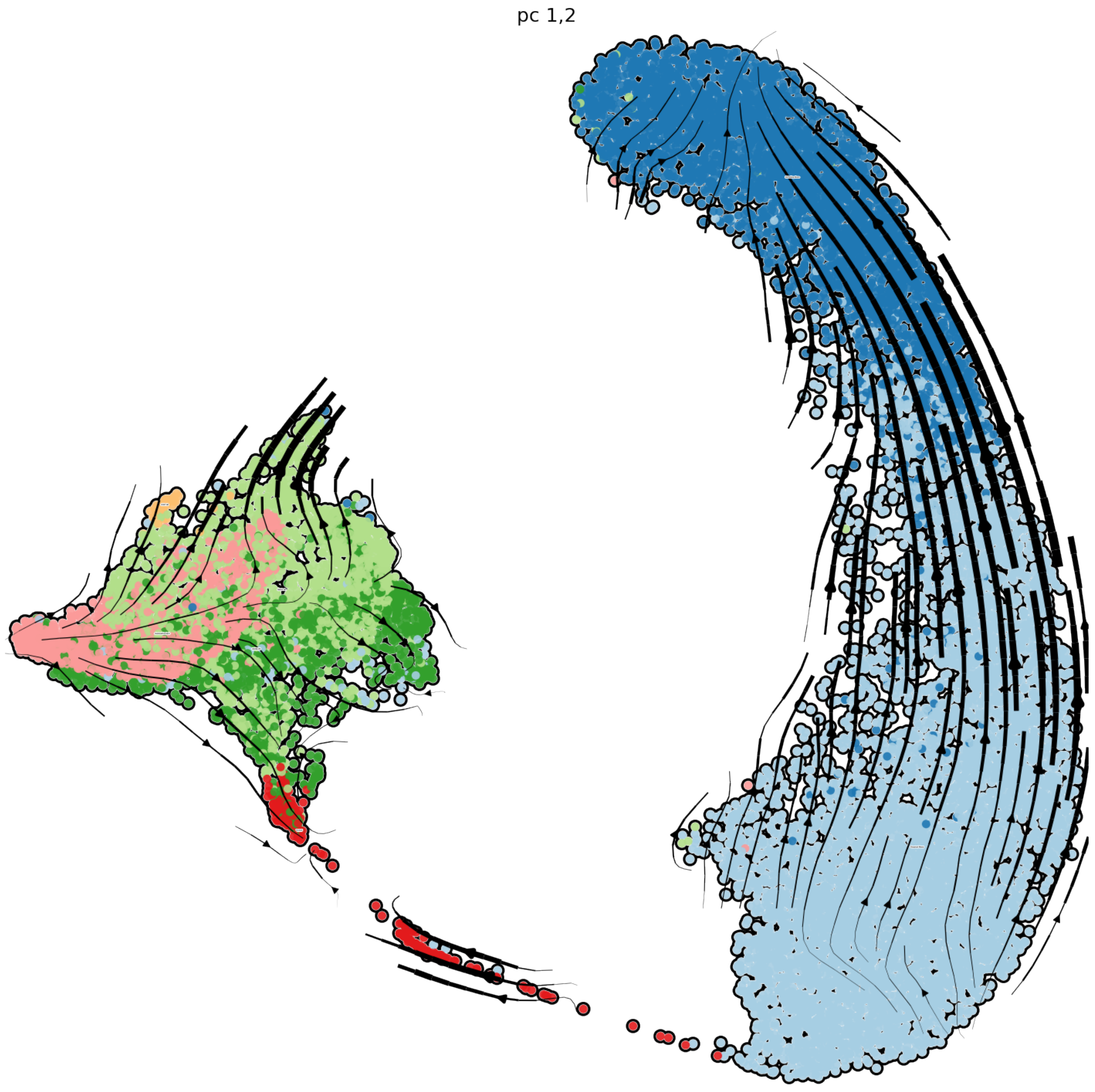

Figure 1: scVelo results for merged sample

pc 1,3

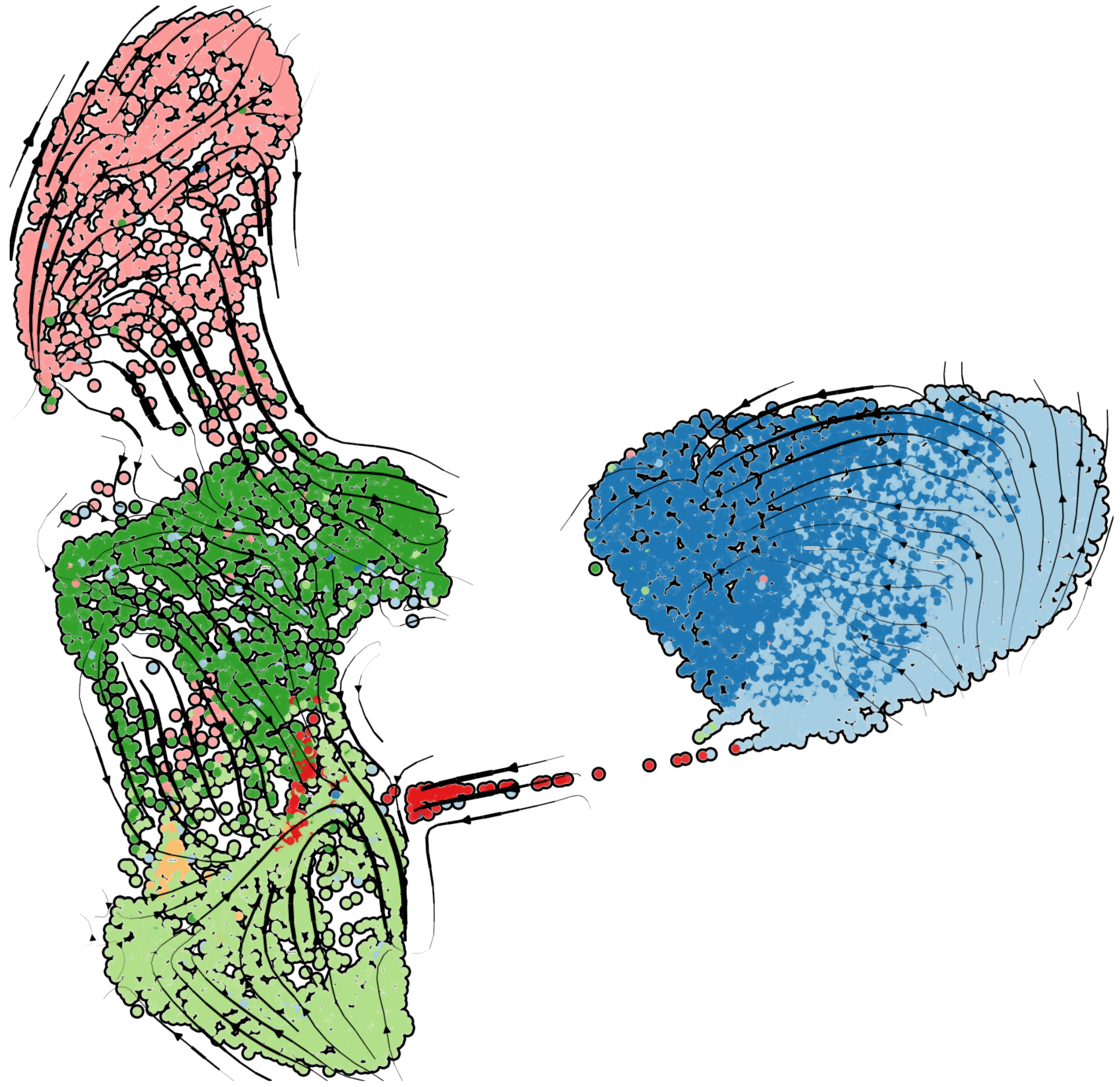

Figure 2: scVelo results for merged sample

```
## 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_1.4-0       RColorBrewer_1.1-2 forcats_0
   .5.1
##  [5] stringr_1.4.0      dplyr_1.0.8        purrr_0.3.4        readr_2
   .1.2
##  [9] tidyr_1.2.0        tibble_3.1.6       ggplot2_3.3.5
   tidyverse_1.3.1
## [13] SeuratObject_4.0.4 Seurat_4.1.0
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.3.1          backports_1.4.1       plyr_1.8.6
##   [4] igraph_1.2.11         lazyeval_0.2.2        splines_4.0.3
##   [7] listenv_0.8.0         scattermore_0.8       digest_0.6.29
##  [10] htmltools_0.5.2       fansi_1.0.2           magrittr_2.0.2
##  [13] tensor_1.5            cluster_2.1.0         ROCR_1.0-11
##  [16] tzdb_0.2.0            globals_0.14.0        modelr_0.1.8
##  [19] matrixStats_0.61.0    spatstat.sparse_2.1-0 colorspace_2.0-3
##  [22] rvest_1.0.2           ggrepel_0.9.1         haven_2.4.3
##  [25] xfun_0.29             crayon_1.5.0          jsonlite_1.7.3
##  [28] spatstat.data_2.1-2   survival_3.2-7        zoo_1.8-9
##  [31] glue_1.6.1            polyclip_1.10-0       gtable_0.3.0
##  [34] leiden_0.3.9          future.apply_1.8.1    BiocGenerics_0.36.1
##  [37] abind_1.4-5           scales_1.1.1          DBI_1.1.2
##  [40] spatstat.random_2.1-0 miniUI_0.1.1.1        Rcpp_1.0.8
##  [43] viridisLite_0.4.0     xtable_1.8-4          reticulate_1.24
##  [46] spatstat.core_2.4-0   bit_4.0.4             htmlwidgets_1.5.4
##  [49] httr_1.4.2            ellipsis_0.3.2        ica_1.0-2
##  [52] pkgconfig_2.0.3       uwot_0.1.11           dbplyr_2.1.1
##  [55] deldir_1.0-6          utf8_1.2.2            tidyselect_1.1.1
##  [58] rlang_1.0.1           reshape2_1.4.4        later_1.3.0
##  [61] munsell_0.5.0         cellranger_1.1.0      tools_4.0.3
##  [64] cli_3.2.0             generics_0.1.2        broom_0.7.12
##  [67] ggridges_0.5.3        evaluate_0.15         fastmap_1.1.0
##  [70] yaml_2.3.5            goftest_1.2-3         knitr_1.37
##  [73] bit64_4.0.5           fs_1.5.2              fitdistrplus_1.1-6
##  [76] RANN_2.6.1            pbapply_1.5-0         future_1.24.0
##  [79] nlme_3.1-155          mime_0.12             xml2_1.3.3
##  [82] hdf5r_1.3.5           compiler_4.0.3        rstudioapi_0.13
##  [85] plotly_4.10.0         png_0.1-7             spatstat.utils_2.3-0
##  [88] reprex_2.0.1          stringi_1.7.6         lattice_0.20-41
##  [91] vctrs_0.3.8           pillar_1.7.0          lifecycle_1.0.1
##  [94] spatstat.geom_2.3-2   lmtest_0.9-39         RcppAnnoy_0.0.19
##  [97] data.table_1.14.2     cowplot_1.1.1         irlba_2.3.5
## [100] httpuv_1.6.5          patchwork_1.1.1       R6_2.5.1
## [103] pcaMethods_1.82.0     promises_1.2.0.1      KernSmooth_2.23-20
## [106] gridExtra_2.3         parallelly_1.30.0     codetools_0.2-18
## [109] MASS_7.3-53           assertthat_0.2.1      withr_2.4.3
## [112] sctransform_0.3.3     mgcv_1.8-33           parallel_4.0.3
## [115] hms_1.1.1             grid_4.0.3            rpart_4.1-15
## [118] rmarkdown_2.11        Rtsne_0.15            Biobase_2.50.0
## [121] shiny_1.7.1           lubridate_1.8.0
```

References

1. La Manno, G. *et al.* RNA velocity of single cells. *Nature* (2018) doi:10.1038/s41586-018-0414-6.

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