Base workflow of flowSpy in use case 1 and 2

Yuting Dai

2019-09-09

- Introduction
- Workflow
- Session information
- References

Introduction

To validate the cellular subpopulations identified by flowSpy, we used a 13-marker panel mass cytometry dataset of healthy human bone marrow. This dataset was generated from Bendall et al [1] and completed quality control by Herring et al [2], which could be downloaded from FlowRepository database [3] (https://flowrepository.org/id/FR-FCM-ZY9R). The aim of this use case was to identify the cellular subpopulations and construct a tree-shaped trajectory, which could reveal the human hematopoietic differentiation hierarchy.

This tutorial contains key steps of `flowSpy` base workflow, including how to build an FSPY object, how to run clustering and dimensionality reduction, how to build a tree based on minimum spanning tree (MST) algorithm, how to run pseudotime and how to identify intermediate state cells.

Workflow

```
# Loading packages
suppressMessages({
library(ggplot2)
library(flowCore)
library(pheatmap)
library(flowSpy)
library(stringr)
})

#########################
# Read Flow Cytometry Data
# It can be downloaded via `git clone https://github.com/ytdai/flowSpy-dataset.git`
# fcs.path musted be modified based on the download directory from GitHub
fcs.path <- "../../flowSpy-dataset/FCS/usecase1_2/"
fcs.file <- paste0(fcs.path, "FR-FCM-ZY9R-Bone_Marrow_cytof.fcs")

########################################
#   Get the expression matrix from FCS file
########################################

# Solution 1
# Read FCS data via flowCore::read.FCS
# Expression data matrix from this method need to
# be performed compensation adjustment and transformation
# manually using flowCore
cytof.data <- flowCore::read.FCS(filename = fcs.file)
```

```
# show elements in mass cytometry data
cytof.data
```

```
## flowFrame object 'FR-FCM-ZY9R-Bone_Marrow_cytof.fcs'
## with 236187 cells and 13 observables:
##        name desc range  minRange maxRange
## $P1     CD3 <NA> 16384 -36.47161    16383
## $P2  CD45RA <NA> 16384 -49.93872    16383
## $P3    CD19 <NA> 16384 -85.81519    16383
## $P4   CD11b <NA> 16384 -50.06744    16383
## $P5     CD4 <NA> 16384 -22.94810    16383
## $P6     CD8 <NA> 16384 -81.21340    16383
## $P7    CD34 <NA> 16384 -52.97938    16383
## $P8    CD20 <NA> 16384 -78.41647    16383
## $P9    CD33 <NA> 16384 -27.77563    16383
## $P10  CD123 <NA> 16384 -51.42798    16383
## $P11   CD38 <NA> 16384 -77.72259    16383
## $P12   CD90 <NA> 16384 -31.92096    16383
## $P13   CD45 <NA> 16384 -34.86934    16383
## 95 keywords are stored in the 'description' slot
```

```
# fetching expression data
exp.data <- cytof.data@exprs

# Solution 2
# Read FCS data via flowSpy::runExprsExtract
# ** This solution is recommended
# ** Use case 1 and 2 follow this solution
exp.data <- runExprsExtract(fcs.file, showDesc = FALSE, transformMethod = "autoLgcl")

# Fetching CD markers
markers <- colnames(exp.data)
markers.idx <- match(markers, colnames(exp.data))

# Build an FSPY object
# If you don't want to see the running log information, set verbose FALSE
# If there is only one case in your analysis workflow, you can just set stage "D0"
meta.data <- data.frame(cell = rownames(exp.data),
                        stage = "D0")
fspy <- createFSPY(raw.data = exp.data, markers = markers,
                   meta.data = meta.data,
                   normalization.method = "none",
                   verbose = T)
```

```
## 2019-09-09 22:08:06 [INFO] Number of cells in processing: 236187
```

```
## 2019-09-09 22:08:06 [INFO] rownames of meta.data and raw.data will be named using column
```

```
## 2019-09-09 22:08:06 [INFO] Index of markers in processing
```

```
## 2019-09-09 22:08:06 [INFO] Creating FSPY object.
```

```
## 2019-09-09 22:08:06 [INFO] No normalization and transformation
```

```
## 2019-09-09 22:08:06 [INFO] Build FSPY object succeed
```

```r
# Cluster cells by SOM algorithm
# Set random seed to make results reproducible
set.seed(6)
fspy <- runCluster(fspy, cluster.method = "som", xdim = 10, ydim = 10, verbose = T)
```

```
## 2019-09-09 22:08:06 [INFO] Calculating FlowSOM.
```

```
## 2019-09-09 22:08:11 [INFO] Calculating FlowSOM completed.
```

```r
# Cluster based downsampling
# The total cell number is 236,187, and we can just keep 10% cells to reduce
# computation load and improve computation time.
# Downsampling by setting downsampleing.size 0.1
set.seed(1)
fspy <- processingCluster(fspy, perplexity = 5,downsampling.size = 0.1)

# Now only 23,664 cells are enrolled in the dimensionality reduction
fspy
```

```
## FSPY Information:
##  Input cell number: 236187  cells
##  Enroll marker number: 13  markers
##  Cells after downsampling: 23664  markers
```

```r
# run Principal Component Analysis (PCA)
fspy <- runFastPCA(fspy, verbose = T)
```

```
## 2019-09-09 22:08:19 [INFO] Calculating PCA.
```

```
## 2019-09-09 22:08:19 [INFO] Calculating PCA completed.
```

```r
# run t-Distributed Stochastic Neighbor Embedding (tSNE)
set.seed(1)
fspy <- runTSNE(fspy, dims = 2, verbose = T)
```

```
## 2019-09-09 22:08:19 [INFO] Calculating tSNE.
```

```
## 2019-09-09 22:10:52 [INFO] Calculating tSNE completed.
```

```r
# run Diffusion map
fspy <- runDiffusionMap(fspy, verbose = T)
```

```
## 2019-09-09 22:10:52 [INFO] Calculating Diffusion Map.
```

```
## 2019-09-09 22:10:52 [INFO] Destiny determined an optimal global sigma: 1.449
```

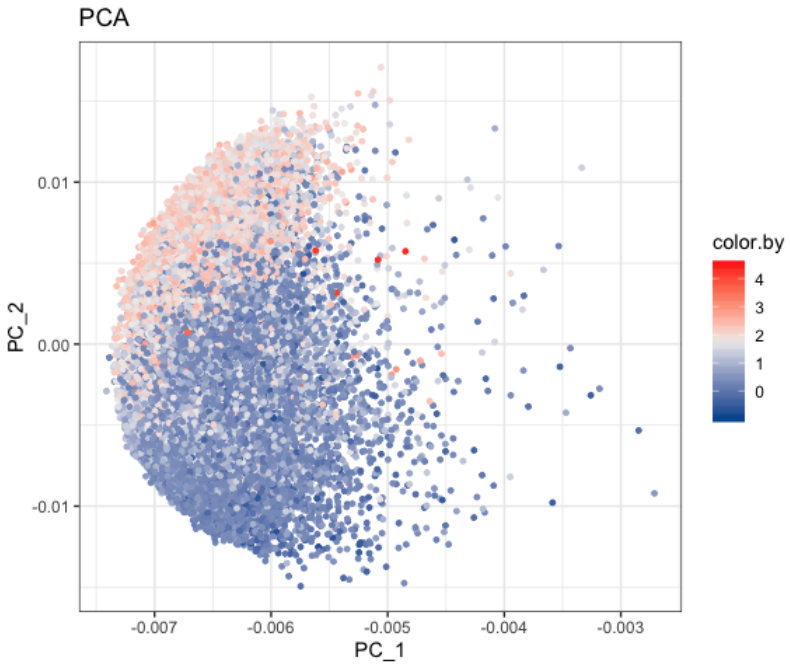
```
## 2019-09-09 22:13:27 [INFO] Calculating Diffusion Map completed
```

```r
# run Uniform Manifold Approximation and Projection (UMAP)
fspy <- runUMAP(fspy, verbose = T)
```

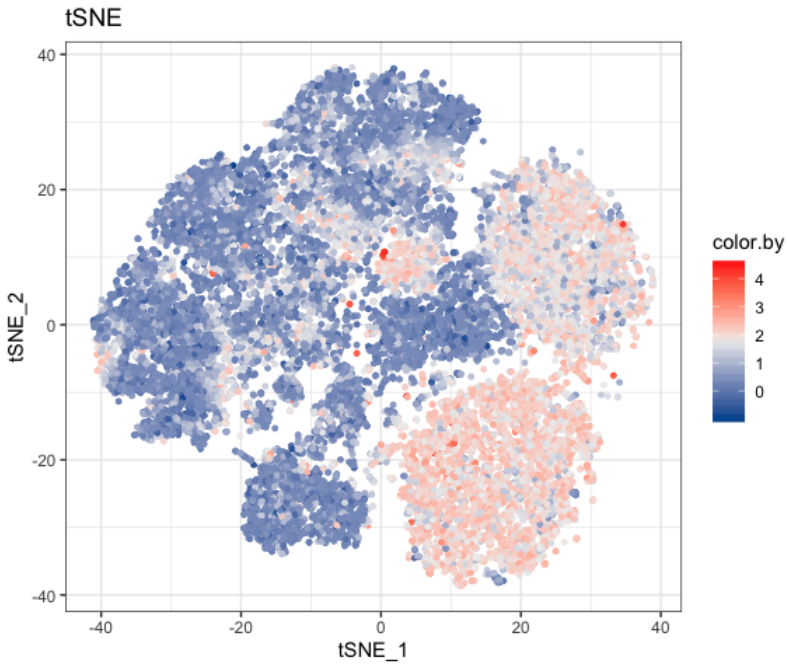
```
## 2019-09-09 22:13:27 [INFO] Calculating Umap.
```

```
## 2019-09-09 22:17:05 [INFO] Calculating Umap.
```

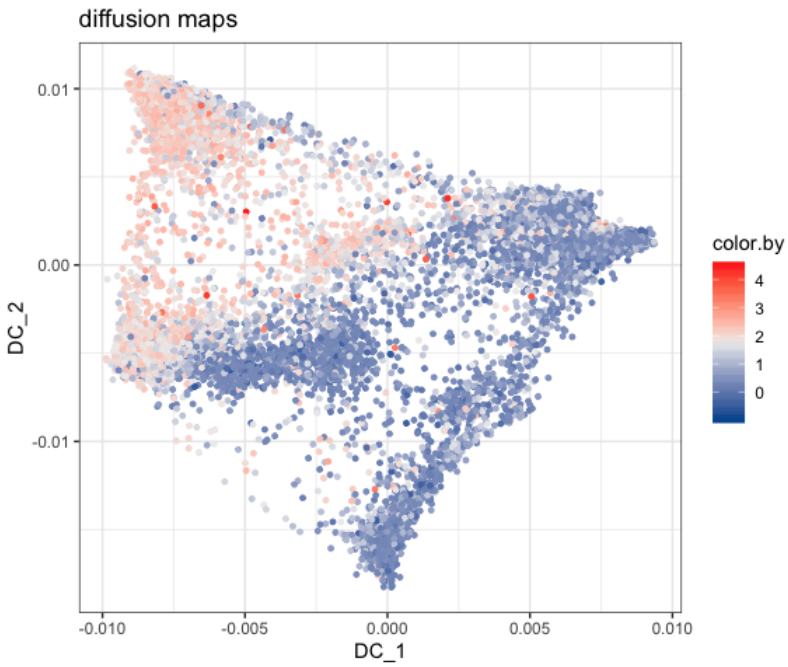
```
##########################################
# This is visualization module
##########################################

# Plot 2D PCA. And cells are colored by CD3 expression
plot2D(fspy, item.use = c("PC_1", "PC_2"), color.by = "CD3",
       alpha = 1, main = "PCA", category = "numeric") +
  scale_colour_gradientn(colors = c("#00599F","#EEEEEE","#FF3222"))
```



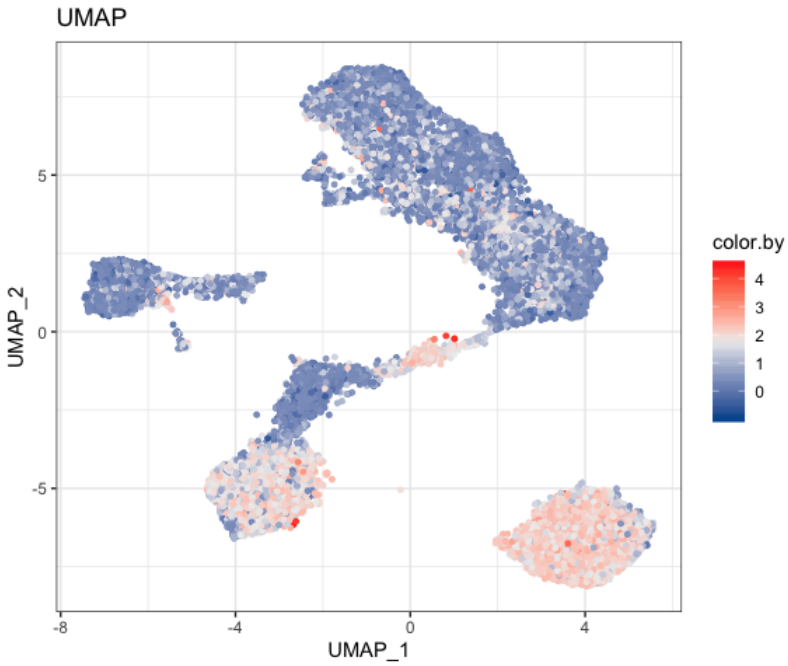

```
# Plot 2D tSNE. And cells are colored by CD3 expression
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "CD3",
       alpha = 1, main = "tSNE", category = "numeric") +
  scale_colour_gradientn(colors = c("#00599F","#EEEEEE","#FF3222"))
```



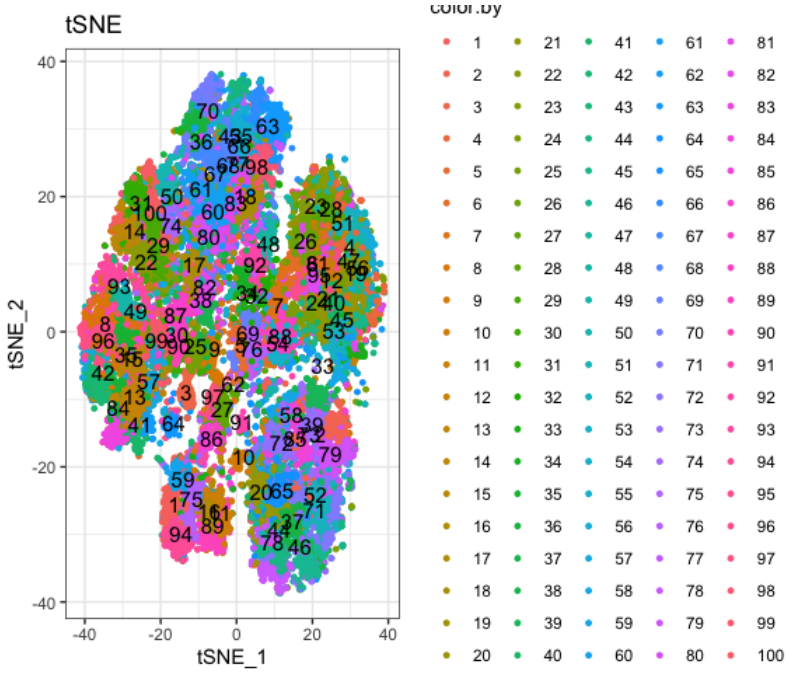

```
# Plot 2D diffusion maps. And cells are colored by CD3 expression
plot2D(fspy, item.use = c("DC_1", "DC_2"), color.by = "CD3",
       alpha = 1, main = "diffusion maps", category = "numeric") +
  scale_colour_gradientn(colors = c("#00599F","#EEEEEE","#FF3222"))
```



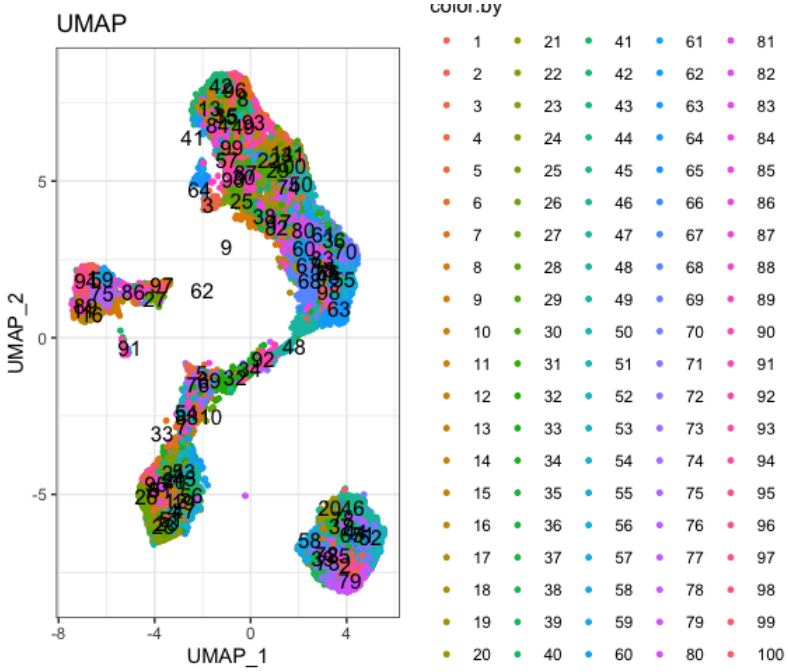

```
# Plot 2D UMAP. And cells are colored by CD3 expression
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "CD3",
       alpha = 1, main = "UMAP", category = "numeric") +
  scale_colour_gradientn(colors = c("#00599F","#EEEEEE","#FF3222"))
```



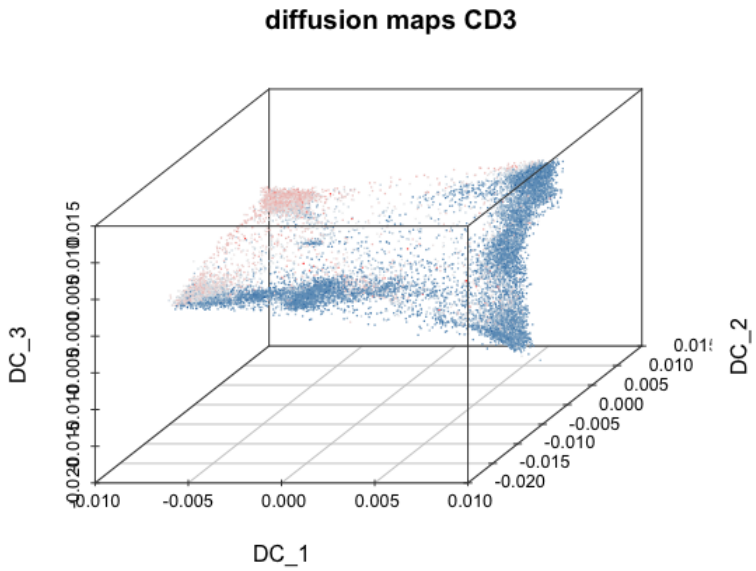

```
# Plot 2D tSNE. And cells are colored by cluster id
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "cluster.id",
       alpha = 1, main = "tSNE", category = "categorical", show.cluser.id = T)
```




```
# Plot 2D UMAP. And cells are colored by cluster id
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "cluster.id",
       alpha = 1, main = "UMAP", category = "categorical", show.cluser.id = T)
```




```
# Plot 3D UMAP. And cells are colored by CD45RA markers expression
plot3D(fspy, item.use = c("DC_1", "DC_2", "DC_3"), color.by = "CD3",
       main = "diffusion maps CD3", category = "numeric", size = 0.2,
       color.theme = c("#00599F","#EEEEEE","#FF3222"))
```




```
##############################################
#   Trajectory
##############################################

# flowSpy provides five method to build the tree-shaped trajectory:
# 1. Raw expression matrix
# 2. PCA
# 3. tSNE
# 4. Diffusion maps
# 5. UMAP

# 1. Raw expression matrix
fspy <- buildTree(fspy, dim.type = "raw", verbose = T)
```

```
## 2019-09-09 22:17:18 [INFO] Calculating buildTree.
```

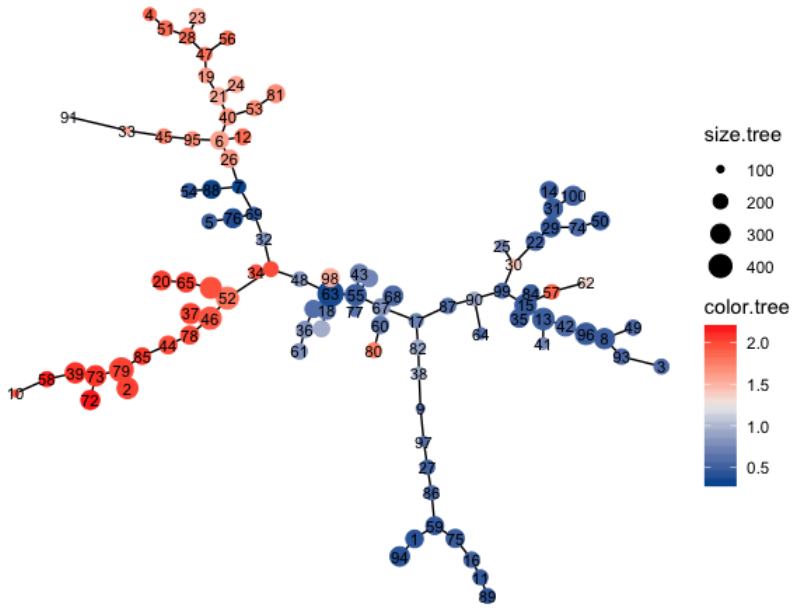
```
## 2019-09-09 22:17:18 [INFO] The log data will be used to calculate trajectory
```

```
## 2019-09-09 22:17:19 [INFO] Initialization for root.cells and leaf cells
```

```
## 2019-09-09 22:17:19 [INFO] Calculating buildTree completed.
```

```
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD3, size.by: cell.number




```
# 2. PCA
fspy <- buildTree(fspy, dim.type = "pca", dim.use = 1:4, verbose = T)
```

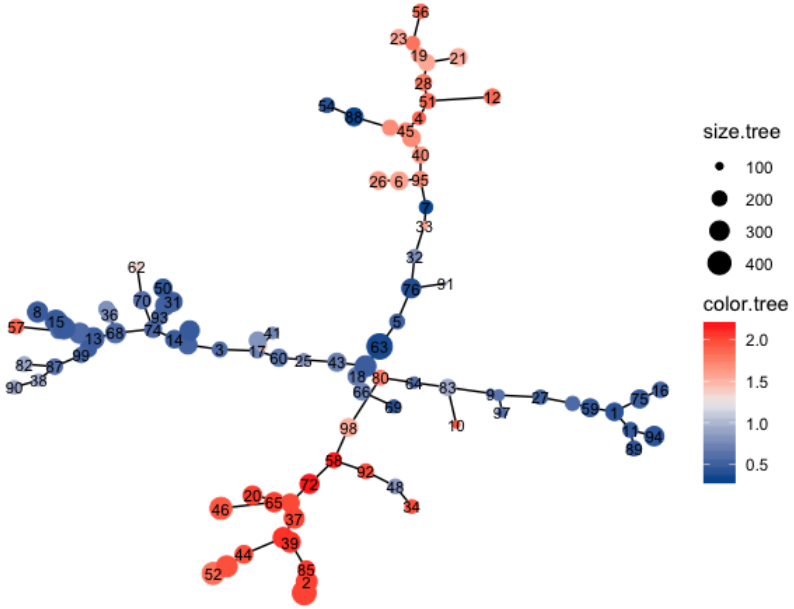
```
## 2019-09-09 22:17:20 [INFO] Calculating buildTree.
```

```
## 2019-09-09 22:17:20 [INFO] Initialization for root.cells and leaf cells
```

```
## 2019-09-09 22:17:20 [INFO] Calculating buildTree completed.
```

```
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD3, size.by: cell.number




```
# 3. tSNE
fspy <- buildTree(fspy, dim.type = "tsne", dim.use = 1:2, verbose = T)
```

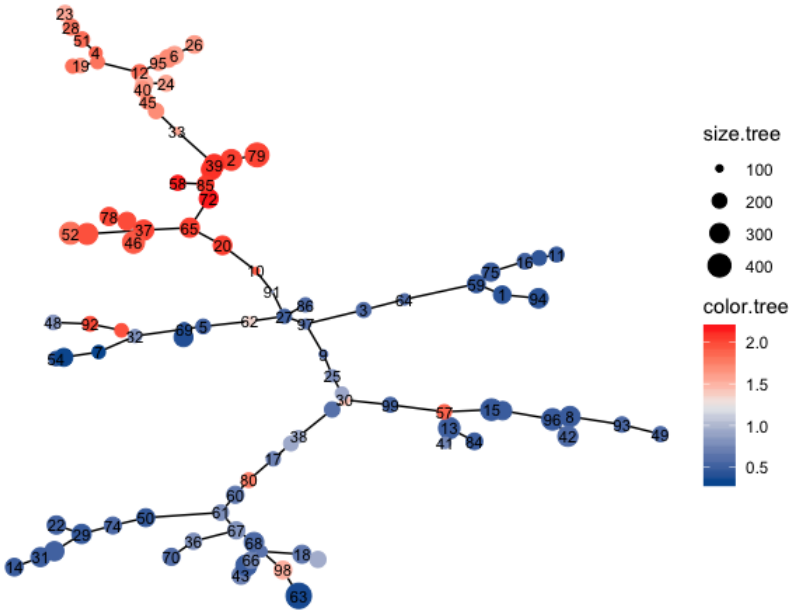
```
## 2019-09-09 22:17:21 [INFO] Calculating buildTree.
```

```
## 2019-09-09 22:17:21 [INFO] Initialization for root.cells and leaf cells
```

```
## 2019-09-09 22:17:21 [INFO] Calculating buildTree completed.
```

```
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD3, size.by: cell.number




```
# 4. Diffusion maps
fspy <- buildTree(fspy, dim.type = "dc", dim.use = 1:3, verbose = T)
```

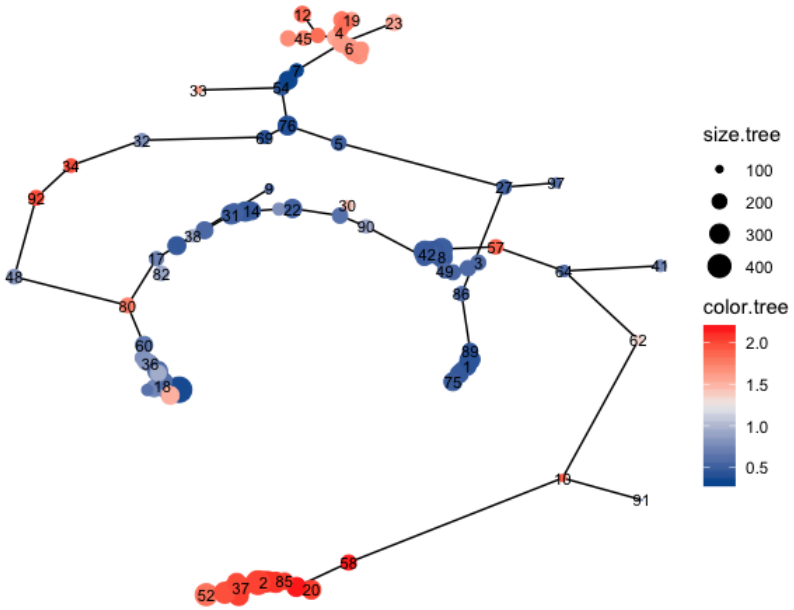
```
## 2019-09-09 22:17:22 [INFO] Calculating buildTree.
```

```
## 2019-09-09 22:17:22 [INFO] Initialization for root.cells and leaf cells
```

```
## 2019-09-09 22:17:22 [INFO] Calculating buildTree completed.
```

```
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD3, size.by: cell.number




```
# 5. UMAP
fspy <- buildTree(fspy, dim.type = "umap", dim.use = 1:2, verbose = T)
```

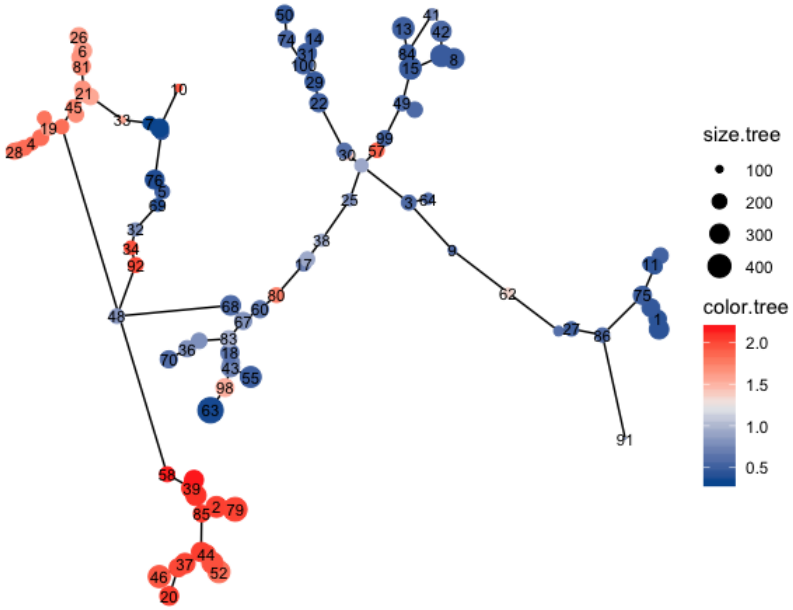
```
## 2019-09-09 22:17:23 [INFO] Calculating buildTree.
```

```
## 2019-09-09 22:17:23 [INFO] Initialization for root.cells and leaf cells
```

```
## 2019-09-09 22:17:23 [INFO] Calculating buildTree completed.
```

```
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD3, size.by: cell.number




```
# The topology of a trajectory is mainly based on the interrelation
# of cell clusters, coordinates and dimensions, and in use case 1 and
# 2, we use "tsne" to construct the trajectory
fspy <- buildTree(fspy, dim.type = "tsne", dim.use = 1:2, verbose = T)
```

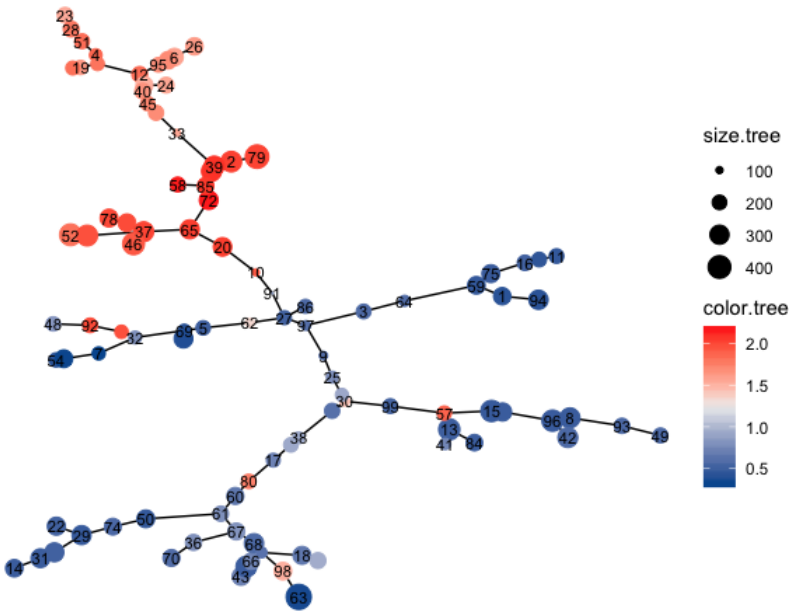
```
## 2019-09-09 22:17:24 [INFO] Calculating buildTree.
```

```
## 2019-09-09 22:17:24 [INFO] Initialization for root.cells and leaf cells
```

```
## 2019-09-09 22:17:24 [INFO] Calculating buildTree completed.
```

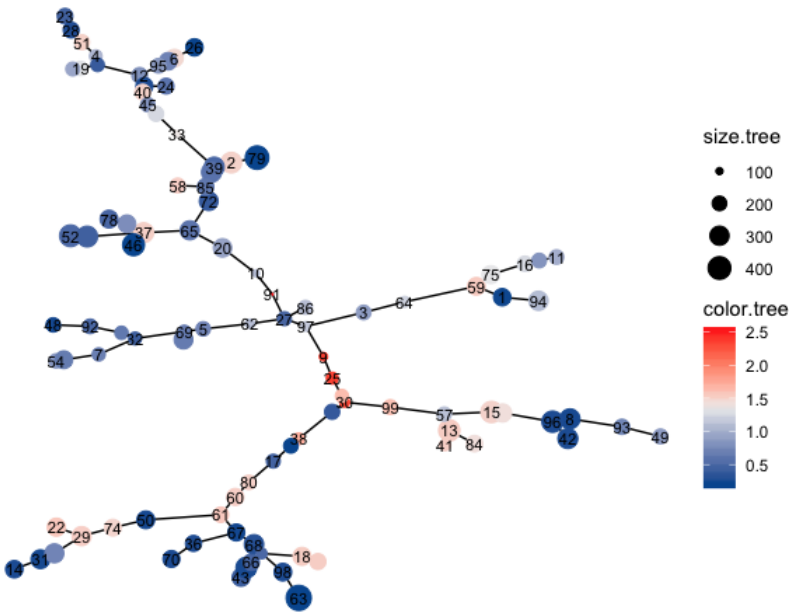
```
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD3, size.by: cell.number



```
plotTree(fspy, color.by = "CD34", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD34, size.by: cell.number

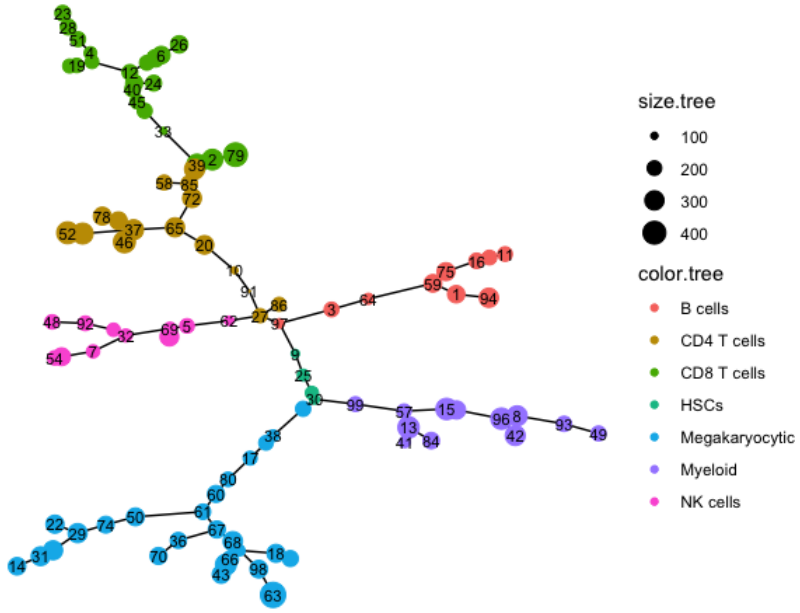



```
############## Modify branch id
fspy@meta.data$branch.id[fspy@meta.data$cluster.id %in% c(9,25,90,30)] = 12
fspy@meta.data$branch.id[fspy@meta.data$cluster.id %in% c(5,62)] = 6
fspy@meta.data$branch.id[fspy@meta.data$cluster.id %in% c(38,87)] = 3

fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(12)] = "HSCs"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(1,2)] = "CD8 T cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(4,5)] = "CD4 T cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(3,9,11)] = "Megakaryocytic"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(6)] = "NK cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(7)] = "B cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(8,10)] = "Myeloid"

# Run differential expressed markers of different branch
diff.info <- runDiff(fspy)
head(diff.info)
```

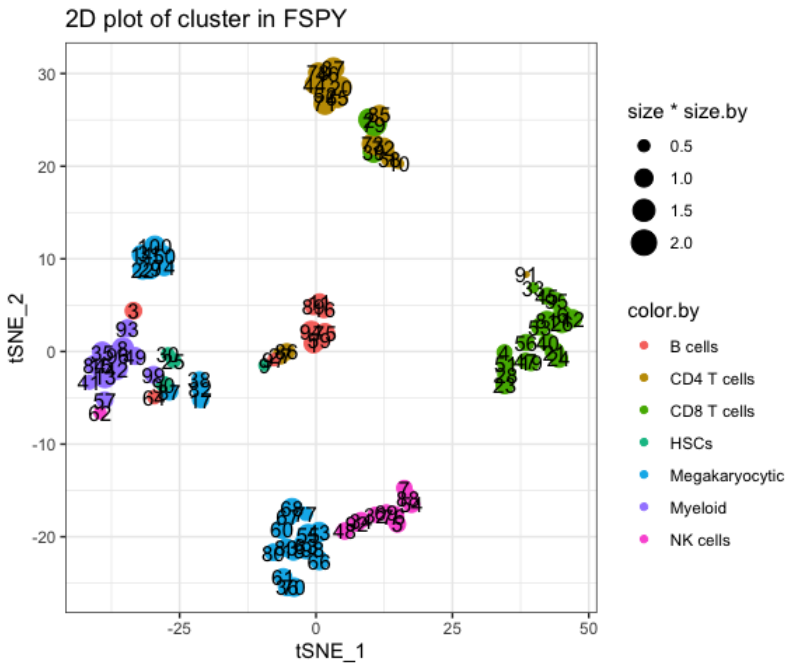
```
##              logFC  AveExpr         t P.Value adj.P.Val         B
## CD8     1.7827096 1.079246 158.52189       0         0 8554.0978
## CD3     0.8365197 1.096834  71.34905       0         0 2295.5803
## CD45    0.7322583 2.343466  54.20401       0         0 1374.7889
## CD45RA  0.7396298 1.617103  53.30852       0         0 1331.8850
## CD33   -0.9383986 1.796045 -52.36995       0         0 1287.5212
## CD123  -0.5759454 1.105198 -41.20916       0         0  810.0004
##              branch.contrast   Gene
## CD8    CD8 T cells_vs_other    CD8
## CD3    CD8 T cells_vs_other    CD3
## CD45   CD8 T cells_vs_other   CD45
## CD45RA CD8 T cells_vs_other CD45RA
## CD33   CD8 T cells_vs_other   CD33
## CD123  CD8 T cells_vs_other  CD123
```

```
# plot tree
plotTree(fspy, color.by = "branch.id", show.node.name = T, cex.size = 1)
```

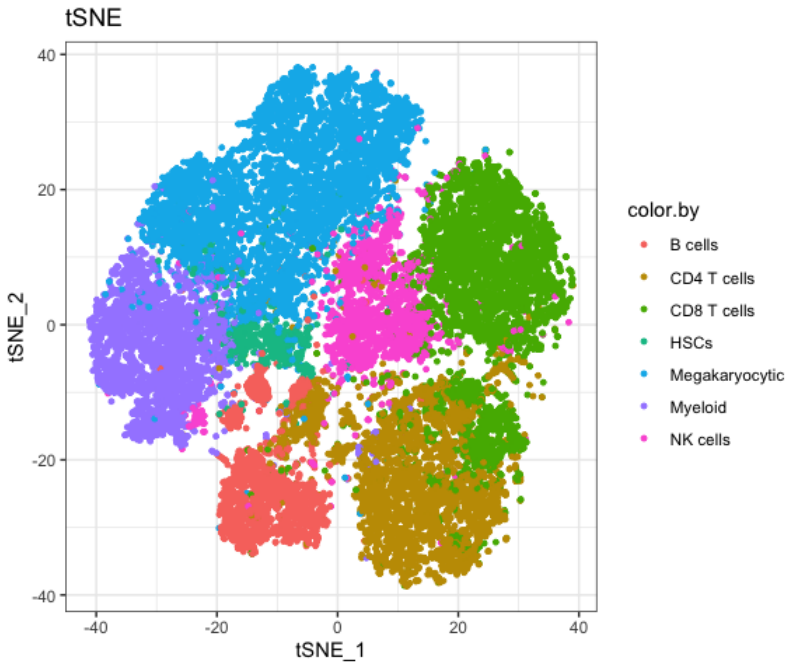
Tree plot, color.by: branch.id, size.by: cell.number




```
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "categorical",
            size = 100, color.by = "branch.id", show.cluser.id = T)
```

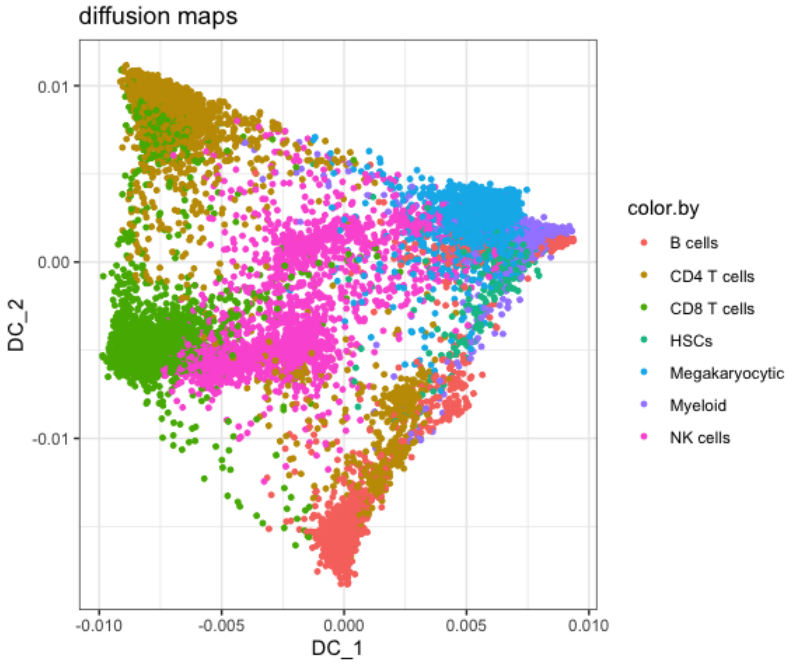



```
# Plot 2D and colored by branch id
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "branch.id",
       alpha = 1, main = "tSNE", category = "categorical", show.cluser.id = F)
```

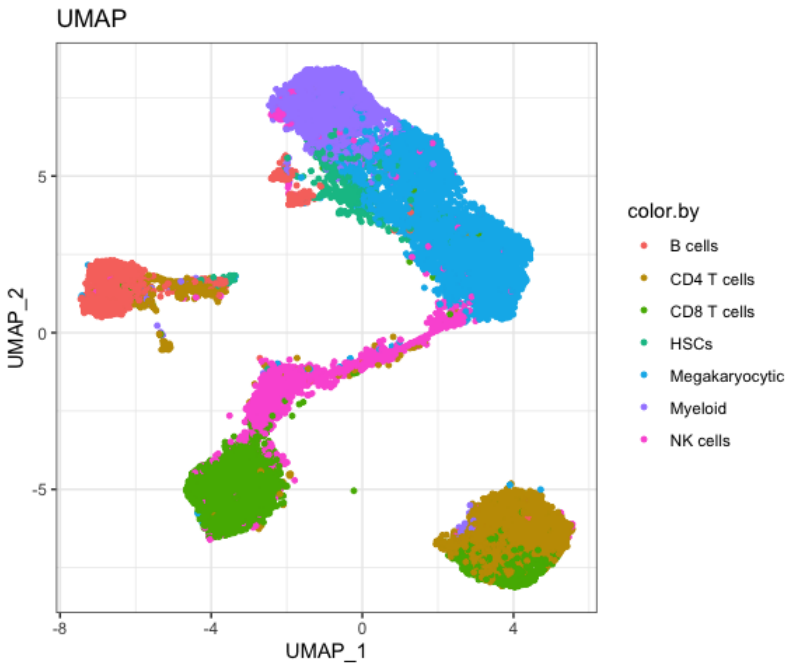



```
plot2D(fspy, item.use = c("PC_1", "PC_2"), color.by = "branch.id",
       alpha = 1, main = "PCA", category = "categorical", show.cluser.id = F)
```

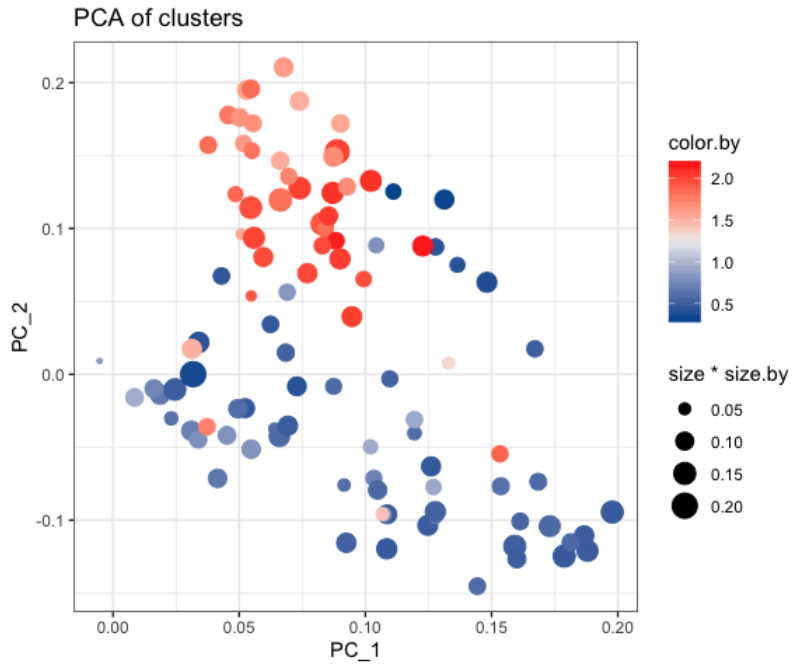

```
plot2D(fspy, item.use = c("DC_1", "DC_2"), color.by = "branch.id",
       alpha = 1, main = "diffusion maps", category = "categorical", show.cluser.id = F)
```



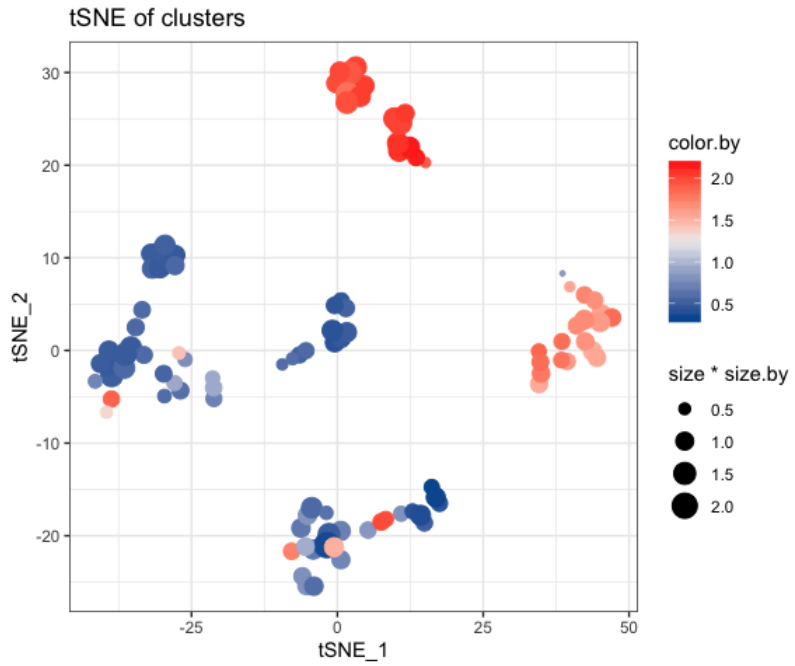

```
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "branch.id",
       alpha = 1, main = "UMAP", category = "categorical", show.cluser.id = F)
```



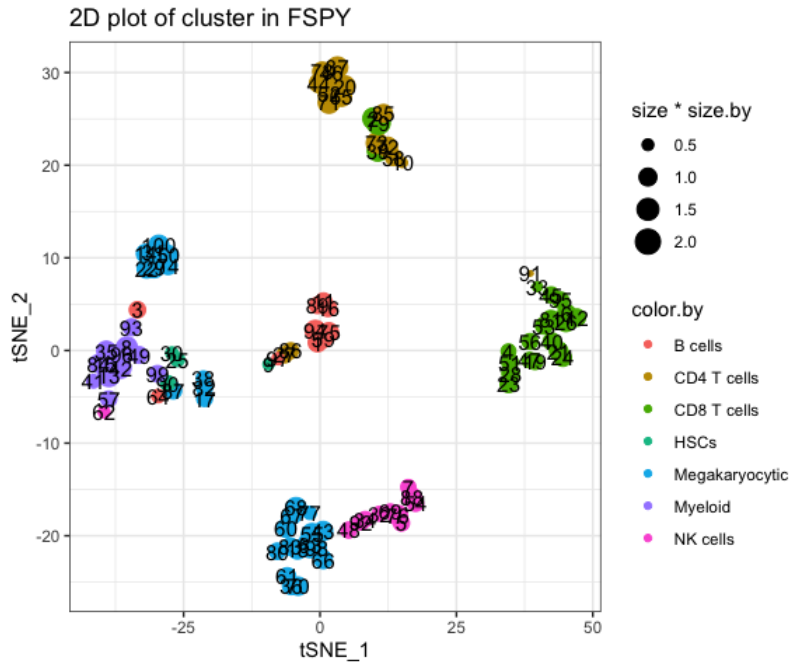

```
# plot for clusters
plotCluster(fspy, item.use = c("PC_1", "PC_2"), category = "numeric",
            size = 10, color.by = "CD3", main = "PCA of clusters") +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



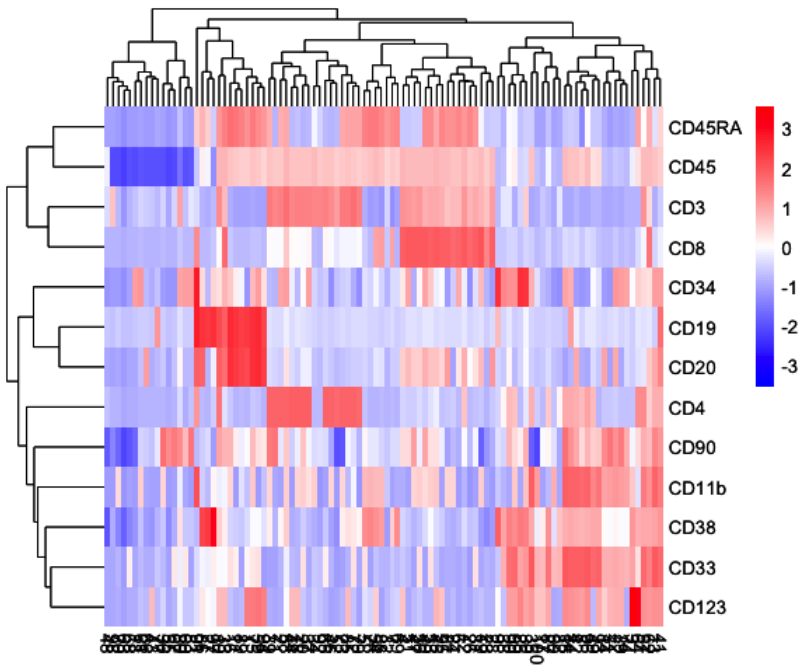

```
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "numeric",
            size = 100, color.by = "CD3", main = "tSNE of clusters") +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



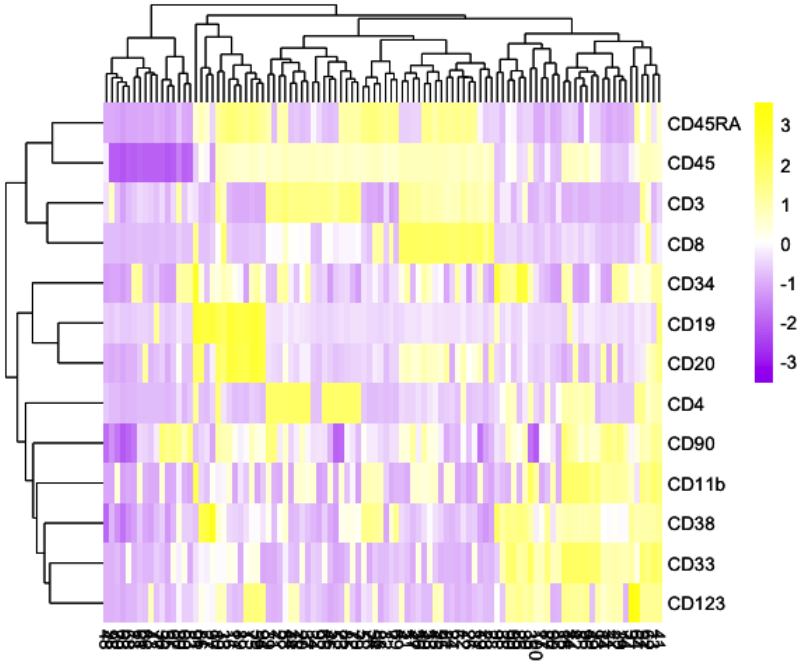

```
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "categorical",
            size = 100, color.by = "branch.id", show.cluser.id = T)
```



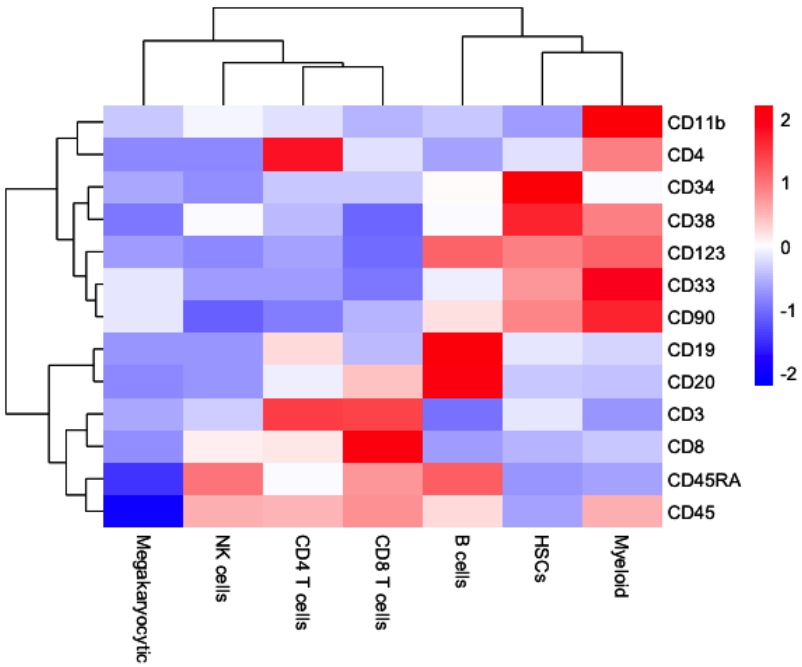

```
# plot heatmap of clusters and branches
plotClusterHeatmap(fspy)
```




```
plotClusterHeatmap(fspy, color = colorRampPalette(c("purple","white","yellow"))(100))
```




```
plotBranchHeatmap(fspy, clustering_method = "ward.D")
```




```
##############################################
#   Pseudotime
##############################################

# Set HSPCs as root cells
fspy <- defRootCells(fspy, root.cells = c(9,25,90,30), verbose = T)
```

```
## 2019-09-09 22:17:37 [INFO] 570 cells will be added to root.cells .
```

```
fspy <- runPseudotime(fspy, verbose = T, dim.type = "raw")
```

```
## 2019-09-09 22:17:37 [INFO] Calculating Pseudotime.
```

```
## 2019-09-09 22:17:37 [INFO] Pseudotime exists in meta.data, it will be replaced.
```

```
## 2019-09-09 22:17:37 [INFO] The log data will be used to calculate trajectory
```

```
## 2019-09-09 22:19:18 [INFO] Calculating Pseudotime completed.
```

```
###### Intermediate state cells for CD8 T cells
fspy <- defLeafCells(fspy, leaf.cells = c(23,28,51), verbose = T)
```

```
## 2019-09-09 22:19:18 [INFO] 650 cells will be added to leaf.cells .
```

```
fspy <- runWalk(fspy, backward.walk = F, verbose = T)
```

```
## 2019-09-09 22:19:18 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2019-09-09 22:19:22 [INFO] Generating an adjacency matrix.
```

```
## 2019-09-09 22:20:42 [INFO] Walk forward.
```

```
## 2019-09-09 22:20:47 [INFO] Calculating walk completed.
```

```
fspy@meta.data$traj.value.log.CD8T <- fspy@meta.data$traj.value.log

###### Intermediate state cells for CD4 T cells
fspy <- defLeafCells(fspy, leaf.cells = c(52,78,46), verbose = T)
```

```
## 2019-09-09 22:20:47 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

```
## 2019-09-09 22:20:47 [INFO] 1012 cells will be added to leaf.cells .
```

```
fspy <- runWalk(fspy, backward.walk = F, verbose = T)
```

```
## 2019-09-09 22:20:47 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2019-09-09 22:20:51 [INFO] Generating an adjacency matrix.
```

```
## 2019-09-09 22:22:10 [INFO] Walk forward.
```

```
## 2019-09-09 22:22:19 [INFO] Calculating walk completed.
```

```
fspy@meta.data$traj.value.log.CD4T <- fspy@meta.data$traj.value.log

###### Intermediate state cells for NK cells
fspy <- defLeafCells(fspy, leaf.cells = c(54,48), verbose = T)
```

```
## 2019-09-09 22:22:19 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

```
## 2019-09-09 22:22:19 [INFO] 392 cells will be added to leaf.cells .
```

```
fspy <- runWalk(fspy, backward.walk = F, verbose = T)
```

```
## 2019-09-09 22:22:19 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2019-09-09 22:22:22 [INFO] Generating an adjacency matrix.
```

```
## 2019-09-09 22:23:54 [INFO] Walk forward.
```

```
## 2019-09-09 22:23:56 [INFO] Calculating walk completed.
```

```
fspy@meta.data$traj.value.log.NK <- fspy@meta.data$traj.value.log

###### Intermediate state cells for B cells
fspy <- defLeafCells(fspy, leaf.cells = c(11,89,94), verbose = T)
```

```
## 2019-09-09 22:23:56 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

```
## 2019-09-09 22:23:56 [INFO] 704 cells will be added to leaf.cells .
```

```
fspy <- runWalk(fspy, backward.walk = F, verbose = T)
```

```
## 2019-09-09 22:23:56 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2019-09-09 22:24:00 [INFO] Generating an adjacency matrix.
```

```
## 2019-09-09 22:25:32 [INFO] Walk forward.
```

```
## 2019-09-09 22:25:37 [INFO] Calculating walk completed.
```

```
fspy@meta.data$traj.value.log.B <- fspy@meta.data$traj.value.log

###### Intermediate state cells for monocytes and granulocytes
fspy <- defLeafCells(fspy, leaf.cells = c(49,93,42), verbose = T)
```

```
## 2019-09-09 22:25:37 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

```
## 2019-09-09 22:25:38 [INFO] 739 cells will be added to leaf.cells .
```

```
fspy <- runWalk(fspy, backward.walk = F, verbose = T)
```

```
## 2019-09-09 22:25:38 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2019-09-09 22:25:40 [INFO] Generating an adjacency matrix.
```

```
## 2019-09-09 22:26:59 [INFO] Walk forward.
```

```
## 2019-09-09 22:27:04 [INFO] Calculating walk completed.
```

```
fspy@meta.data$traj.value.log.MY <- fspy@meta.data$traj.value.log

###### Intermediate state cells for megakaryocyte and erythrocyte
fspy <- defLeafCells(fspy, leaf.cells = c(14,31,100,63,98), verbose = T)
```

```
## 2019-09-09 22:27:04 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

```
## 2019-09-09 22:27:04 [INFO] 1583 cells will be added to leaf.cells .
```

```
fspy <- runWalk(fspy, backward.walk = F, verbose = T)
```

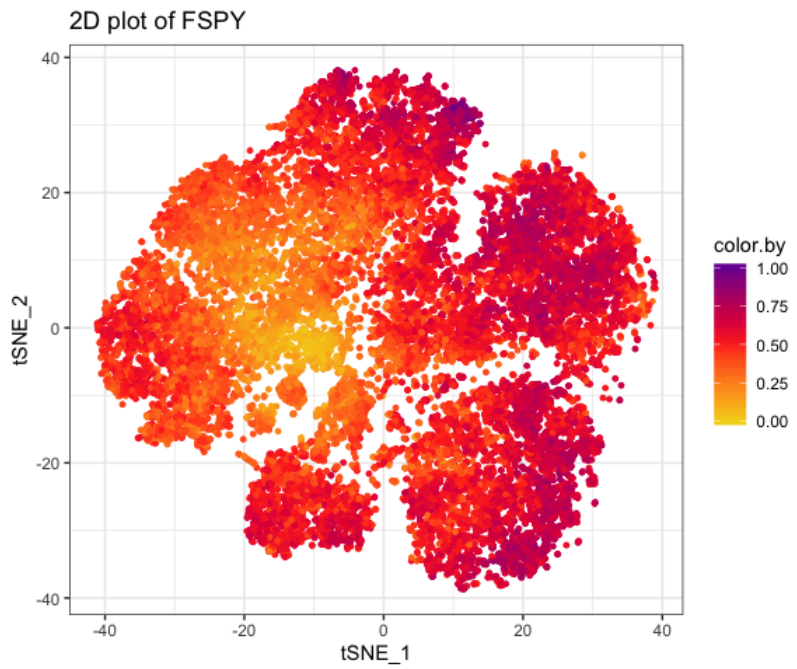
```
## 2019-09-09 22:27:04 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2019-09-09 22:27:08 [INFO] Generating an adjacency matrix.
```

```
## 2019-09-09 22:28:36 [INFO] Walk forward.
```

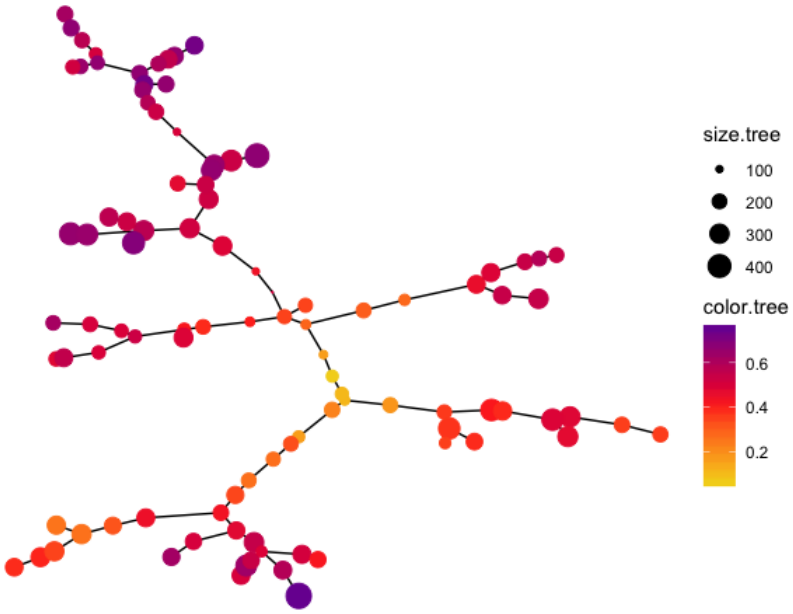
```
## 2019-09-09 22:28:47 [INFO] Calculating walk completed.
```

```
fspy@meta.data$traj.value.log.ME <- fspy@meta.data$traj.value.log

# Plot 2D tSNE.
fspy@meta.data$stage <- fspy@meta.data$branch.id

plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "numeric",
            size = 1, color.by = "pseudotime") +
 scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```

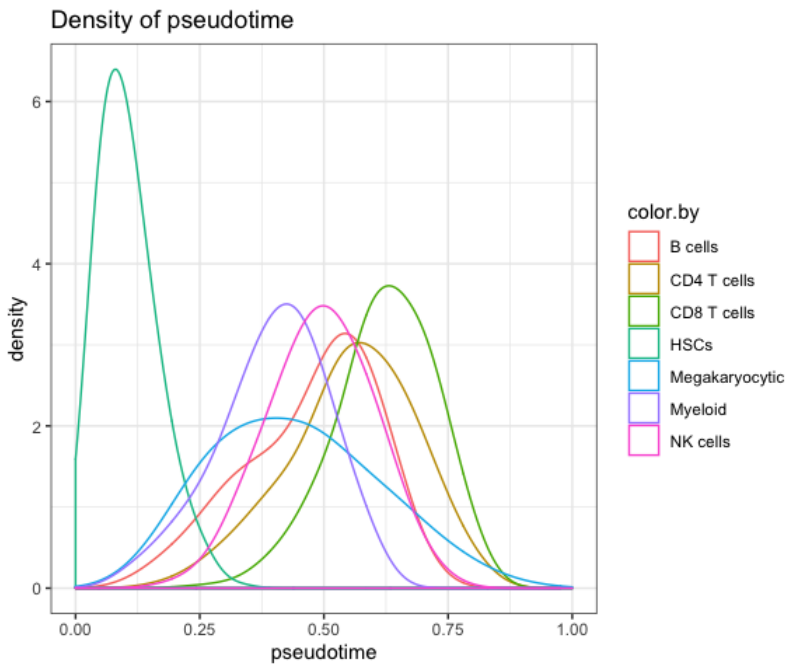


```
# Tree plot
plotTree(fspy, color.by = "pseudotime", cex.size = 1) +
 scale_colour_gradientn(colors = c("#F4D31D","#FF3222","#7A06A0"))
```

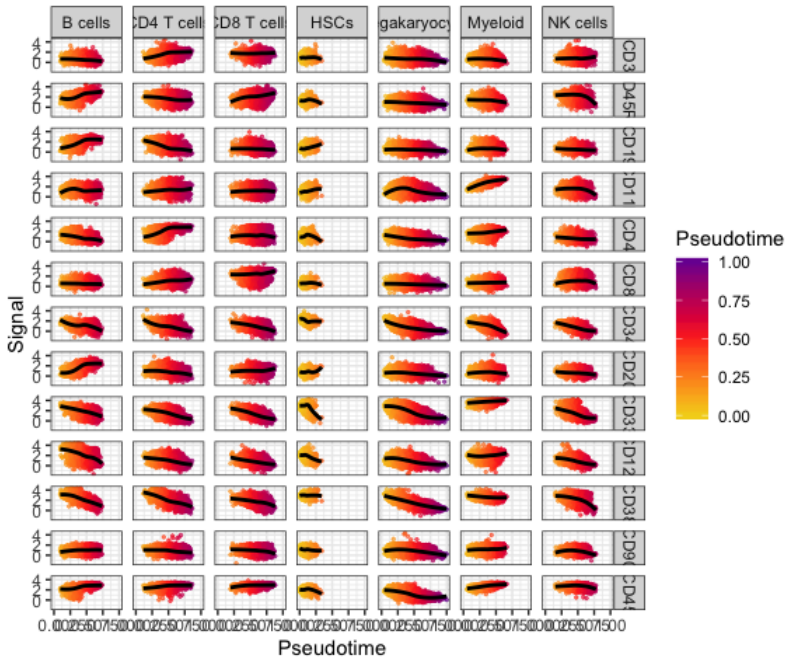
Tree plot, color.by: pseudotime, size.by: cell.number



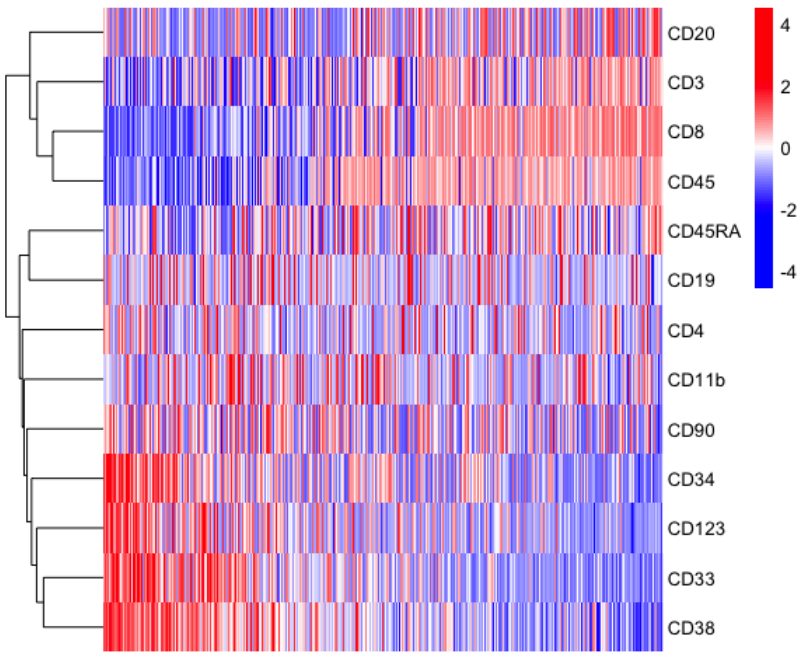

```
# pseudotime density
plotPseudotimeDensity(fspy, adjust = 2)
```



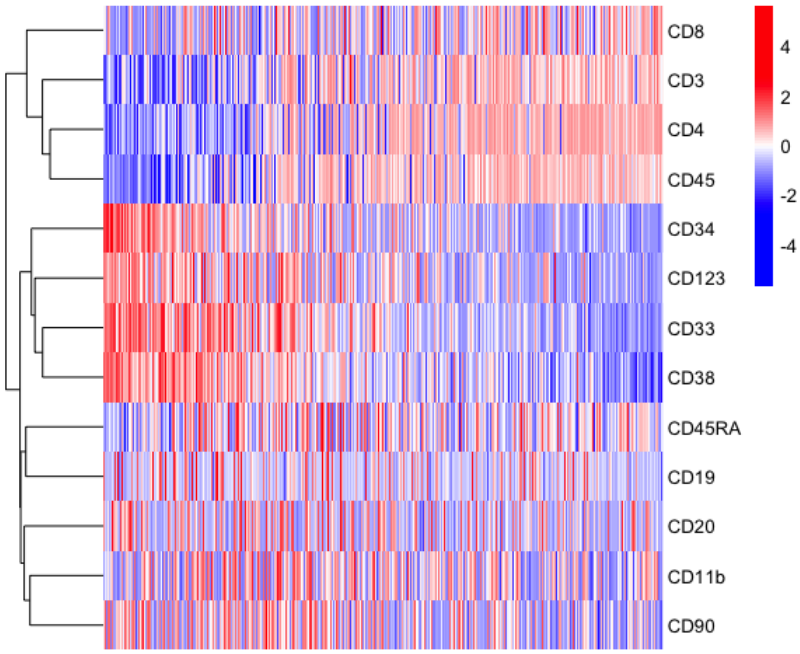

```
plotPseudotimeTraj(fspy, var.cols = T) +
 scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```



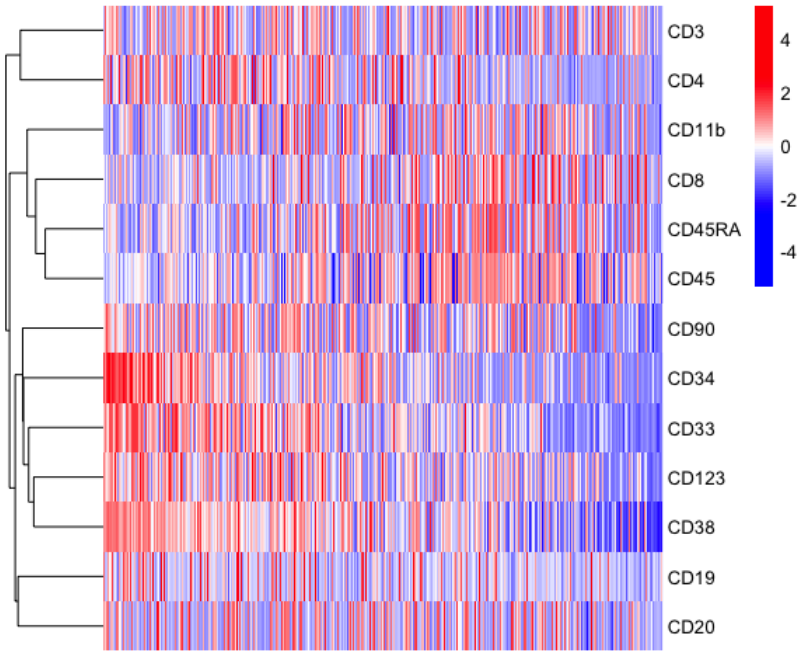

```
### fetch plot information
plot.meta <- fetchPlotMeta(fspy, markers = markers)

# heatmap for CD8 T cells
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.CD8T > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
         color = colorRampPalette(c("blue","blue","white","red","red"))(100),
         fontsize_col = 0.01)
```



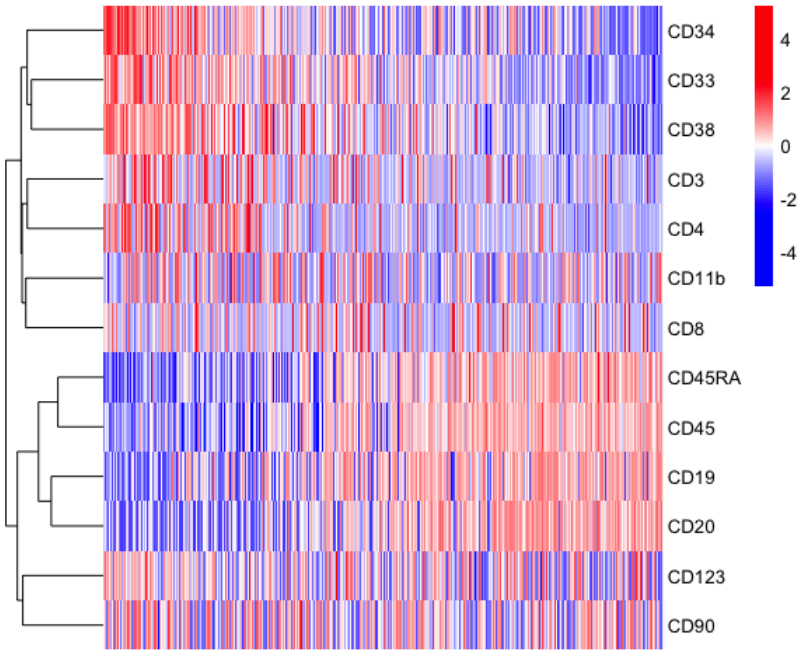

```
# heatmap for CD4 T cells
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.CD4T > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
         color = colorRampPalette(c("blue","blue","white","red","red"))(100),
         fontsize_col = 0.01)
```



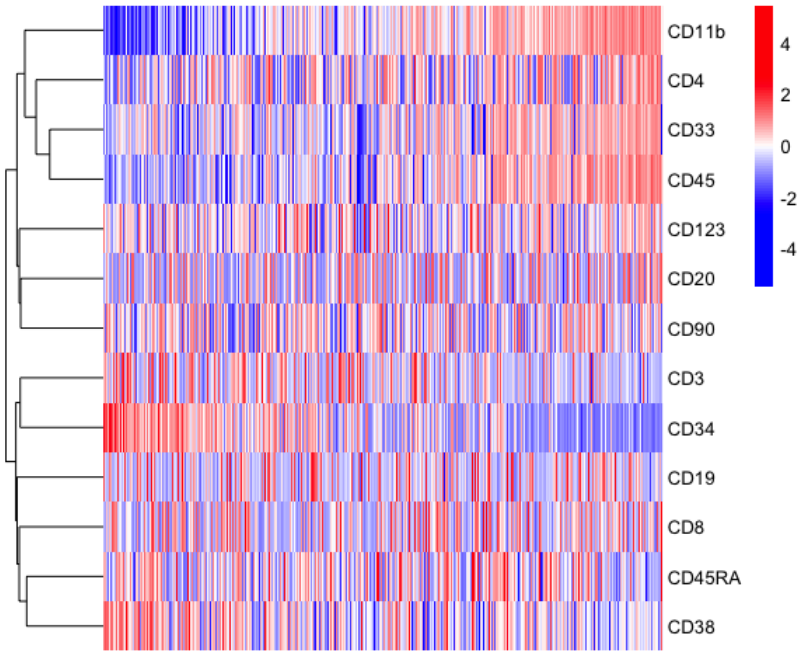

```r
# heatmap for NK cells
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.NK > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
         color = colorRampPalette(c("blue","blue","white","red","red"))(100),
         fontsize_col = 0.01)
```



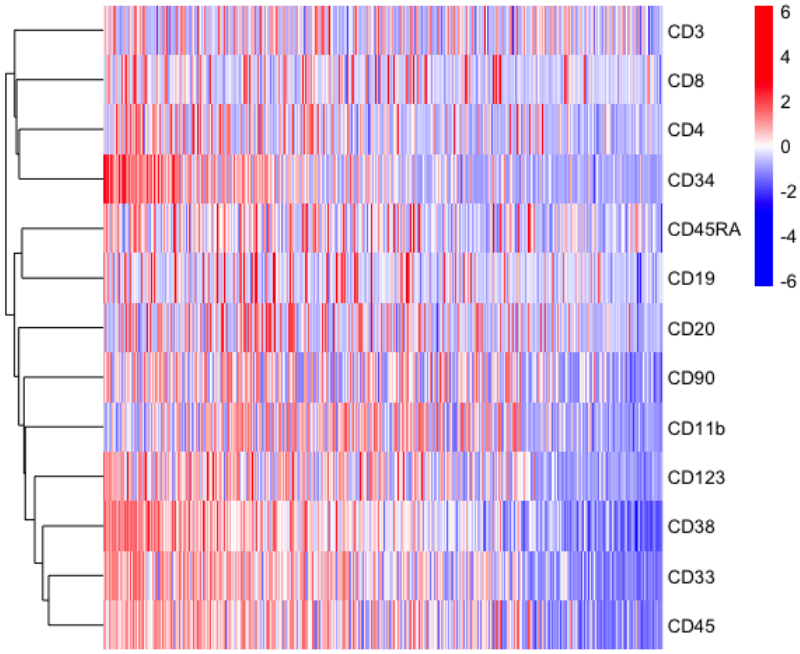

```r
# heatmap for B cells
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.B > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
         color = colorRampPalette(c("blue","blue","white","red","red"))(100),
         fontsize_col = 0.01)
```




```
# heatmap for monocytes and granulocytes
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.MY > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
         color = colorRampPalette(c("blue","blue","white","red","red"))(100),
         fontsize_col = 0.01)
```




```
# heatmap for megakaryocyte and erythrocyte
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.ME > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
         color = colorRampPalette(c("blue","blue","white","red","red"))(100),
         fontsize_col = 0.01)
```



Session information

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] stringr_1.4.0   flowSpy_1.2.7   igraph_1.2.4.1  pheatmap_1.0.12
## [5] flowCore_1.50.0 ggplot2_3.2.0
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.3.1                backports_1.1.4
##   [3] RcppEigen_0.3.3.5.0         plyr_1.8.4
##   [5] ConsensusClusterPlus_1.48.0 lazyeval_0.2.2
##   [7] sp_1.3-1                    splines_3.6.1
##   [9] BiocParallel_1.18.1         GenomeInfoDb_1.20.0
##  [11] sva_3.32.1                  digest_0.6.20
##  [13] htmltools_0.3.6             gdata_2.18.0
##  [15] magrittr_1.5                memoise_1.1.0
##  [17] cluster_2.1.0               openxlsx_4.1.0.1
##  [19] limma_3.40.6                annotate_1.62.0
##  [21] matrixStats_0.54.0          gmodels_2.18.1
##  [23] xts_0.11-2                  colorspace_1.4-1
##  [25] blob_1.2.0                  rrcov_1.4-7
##  [27] haven_2.1.1                 xfun_0.8
##  [29] dplyr_0.8.3                 crayon_1.3.4
##  [31] RCurl_1.95-4.12             jsonlite_1.6
##  [33] graph_1.62.0                scatterpie_0.1.2
```

```
##  [35] genefilter_1.66.0           zeallot_0.1.0
##  [37] survival_2.44-1.1           zoo_1.8-6
##  [39] glue_1.3.1                  polyclip_1.10-0
##  [41] gtable_0.3.0                zlibbioc_1.30.0
##  [43] XVector_0.24.0              DelayedArray_0.10.0
##  [45] car_3.0-3                   BiocGenerics_0.30.0
##  [47] DEoptimR_1.0-8              abind_1.4-5
##  [49] VIM_4.8.0                   scales_1.0.0
##  [51] mvtnorm_1.0-11              DBI_1.0.0
##  [53] ggthemes_4.2.0              Rcpp_1.0.2
##  [55] xtable_1.8-4                laeken_0.5.0
##  [57] reticulate_1.13             foreign_0.8-72
##  [59] bit_1.1-14                  proxy_0.4-23
##  [61] mclust_5.4.5                FlowSOM_1.16.0
##  [63] stats4_3.6.1                tsne_0.1-3
##  [65] umap_0.2.2.0                vcd_1.4-4
##  [67] RColorBrewer_1.1-2          pkgconfig_2.0.2
##  [69] XML_3.98-1.20               farver_1.1.0
##  [71] nnet_7.3-12                 reshape2_1.4.3
##  [73] labeling_0.3                tidyselect_0.2.5
##  [75] rlang_0.4.0                 AnnotationDbi_1.46.0
##  [77] munsell_0.5.0               cellranger_1.1.0
##  [79] tools_3.6.1                 RSQLite_2.1.2
##  [81] ranger_0.11.2               evaluate_0.14
##  [83] yaml_2.2.0                  knitr_1.24
##  [85] bit64_0.9-7                 zip_2.0.3
##  [87] robustbase_0.93-5           purrr_0.3.2
##  [89] RANN_2.6.1                  nlme_3.1-141
##  [91] compiler_3.6.1              curl_4.0
##  [93] e1071_1.7-2                 smoother_1.1
##  [95] tibble_2.1.3                tweenr_1.0.1
##  [97] pcaPP_1.9-73                stringi_1.4.3
##  [99] RSpectra_0.15-0             forcats_0.4.0
## [101] lattice_0.20-38             Matrix_1.2-17
## [103] vctrs_0.2.0                 pillar_1.4.2
## [105] RUnit_0.4.32                lmtest_0.9-37
## [107] BiocNeighbors_1.2.0         data.table_1.12.2
## [109] bitops_1.0-6                corpcor_1.6.9
## [111] GenomicRanges_1.36.0        R6_2.4.0
## [113] rio_0.5.16                  IRanges_2.18.1
## [115] flowUtils_1.48.0            boot_1.3-23
## [117] MASS_7.3-51.4               gtools_3.8.1
## [119] assertthat_0.2.1            destiny_2.14.0
## [121] SummarizedExperiment_1.14.1 withr_2.1.2
## [123] S4Vectors_0.22.0            GenomeInfoDbData_1.2.1
## [125] mgcv_1.8-28                 parallel_3.6.1
## [127] hms_0.5.0                   grid_3.6.1
## [129] prettydoc_0.3.0             tidyr_0.8.3
## [131] class_7.3-15                rmarkdown_1.14
## [133] rvcheck_0.1.3               carData_3.0-2
## [135] Rtsne_0.15                  TTR_0.23-4
## [137] ggforce_0.2.2               scatterplot3d_0.3-41
## [139] Biobase_2.44.0
```

References

1. Bendall SC, Simonds EF, Qiu P, Amir el AD, Krutzik PO, Finck R, Bruggner RV, Melamed R, Trejo A, Ornatsky OI, et al: Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. Science 2011, 332:687-696.

2. Herring CA, Banerjee A, McKinley ET, Simmons AJ, Ping J, Roland JT, Franklin JL, Liu Q, Gerdes MJ, Coffey RJ, Lau KS: Unsupervised Trajectory Analysis of Single-Cell RNA-Seq and Imaging Data Reveals Alternative Tuft Cell Origins in the Gut. Cell Syst 2018, 6:37-51 e39.

3. Spidlen J, Breuer K, Rosenberg C, Kotecha N, Brinkman RR: FlowRepository: a resource of annotated flow cytometry datasets associated with peer-reviewed publications. Cytometry A 2012, 81:727-731.