Base workflow of flowSpy in use case 1 and 2

Yuting Dai

2020-02-24

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 Mass 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 <- "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)
```

```
## 2020-02-24 15:18:34 [INFO] Number of cells in processing: 236187
```

```
## 2020-02-24 15:18:34 [INFO] rownames of meta.data and raw.data will be named using column
```

```
## 2020-02-24 15:18:34 [INFO] Index of markers in processing
```

```
## 2020-02-24 15:18:34 [INFO] Creating FSPY object.
```

```
## 2020-02-24 15:18:34 [INFO] No normalization and transformation
```

```
## 2020-02-24 15:18:35 [INFO] Build FSPY object succeed
```

```r
# Cluster cells by SOM algorithm
# Set random seed to make results reproducible
set.seed(8)
fspy <- runCluster(fspy, cluster.method = "som", xdim = 10, ydim = 10, verbose = T)
```

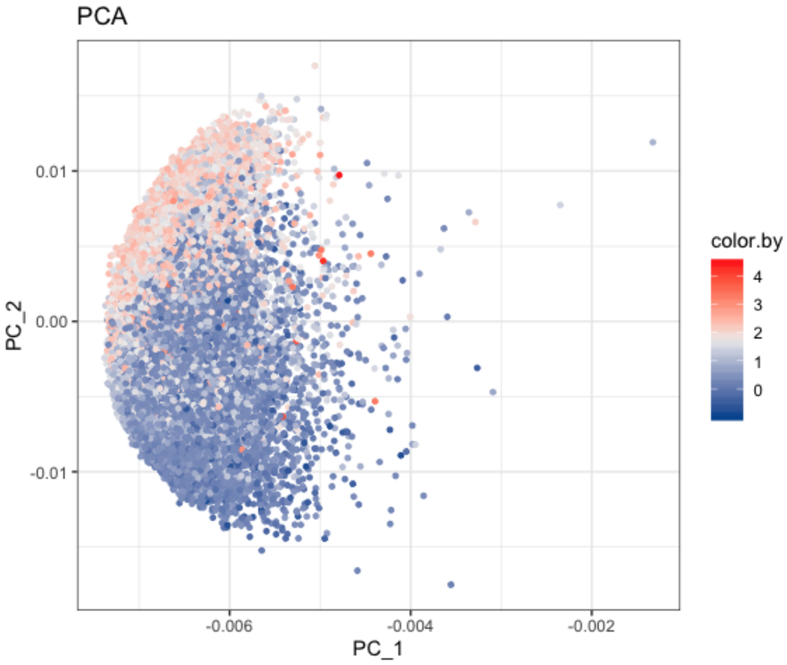
```
## 2020-02-24 15:18:35 [INFO] Calculating FlowSOM.
```

```
## 2020-02-24 15:18:38 [INFO] Calculating FlowSOM completed.
```

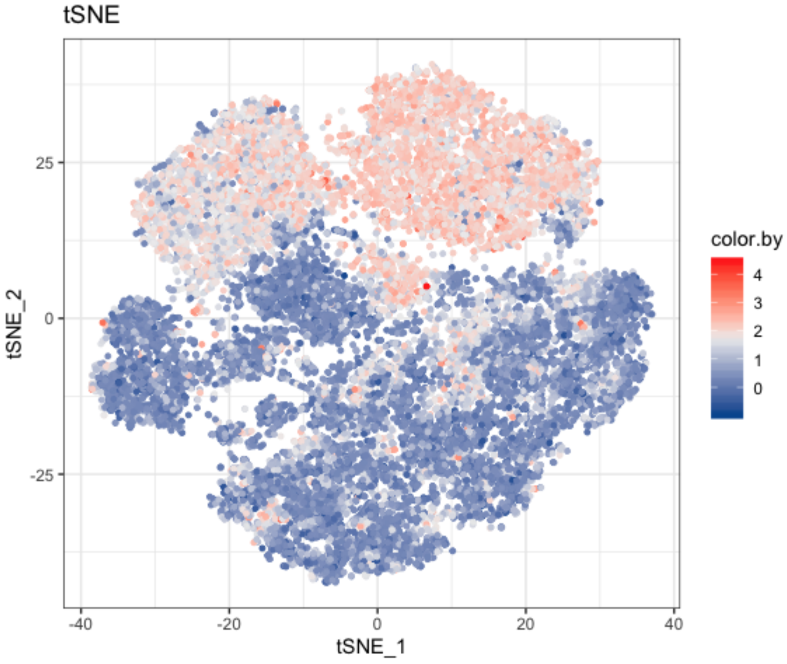
```r
# Cluster based downsampling
# The total cell number is 236,187, and we can just keep 10% cells to reduce
# computation load and improve computation time.
# Downsampling by setting downsampleing.size 0.1
fspy <- processingCluster(fspy, perplexity = 5, downsampling.size = 0.1)

# 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: 23662  markers
```

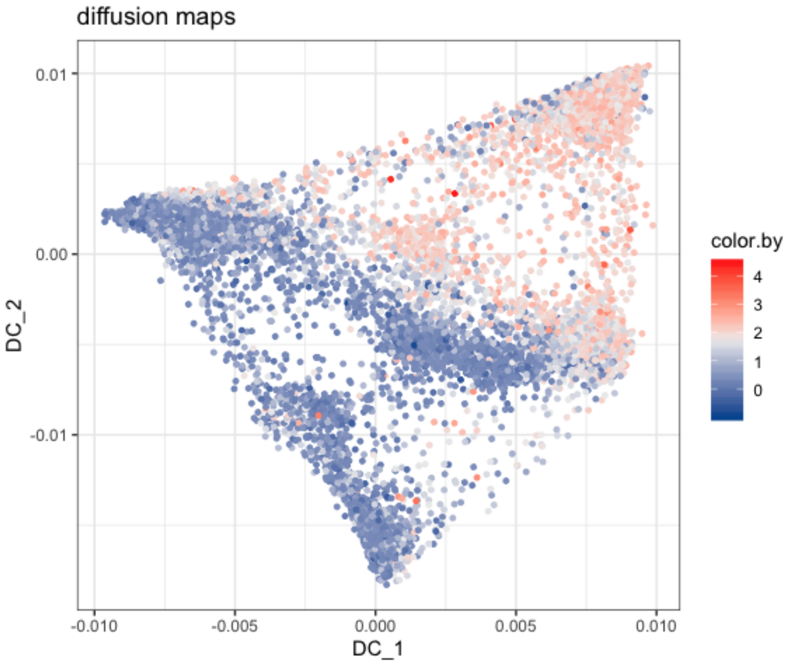
```r
# run Principal Component Analysis (PCA)
fspy <- runFastPCA(fspy)

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

# run Diffusion map
fspy <- runDiffusionMap(fspy)

# run Uniform Manifold Approximation and Projection (UMAP)
fspy <- runUMAP(fspy)

############################################
# 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"))
```



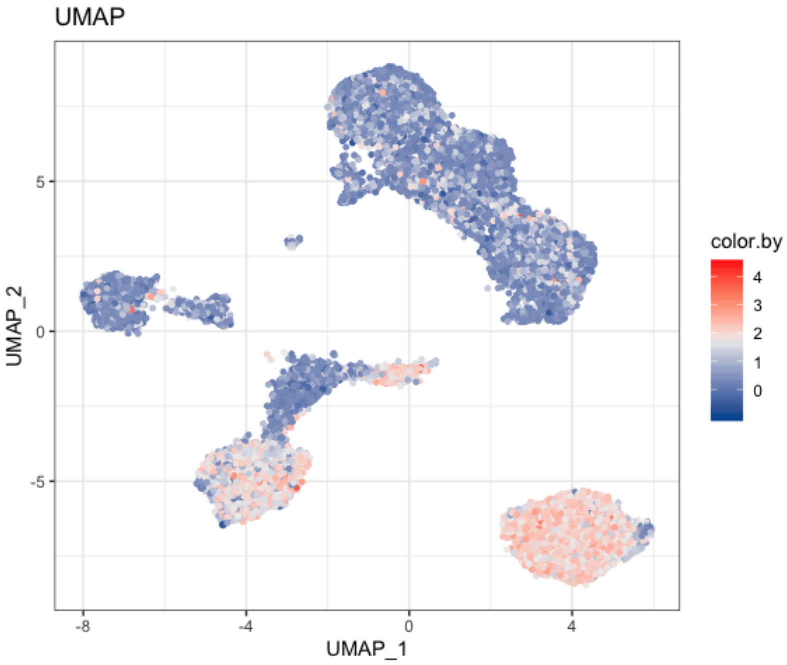
```r
# Plot 2D tSNE. And cells are colored by CD3 expression
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "CD3",
       alpha = 1, main = "tSNE", category = "numeric") +
  scale_colour_gradientn(colors = c("#00599F","#EEEEEE","#FF3222"))
```



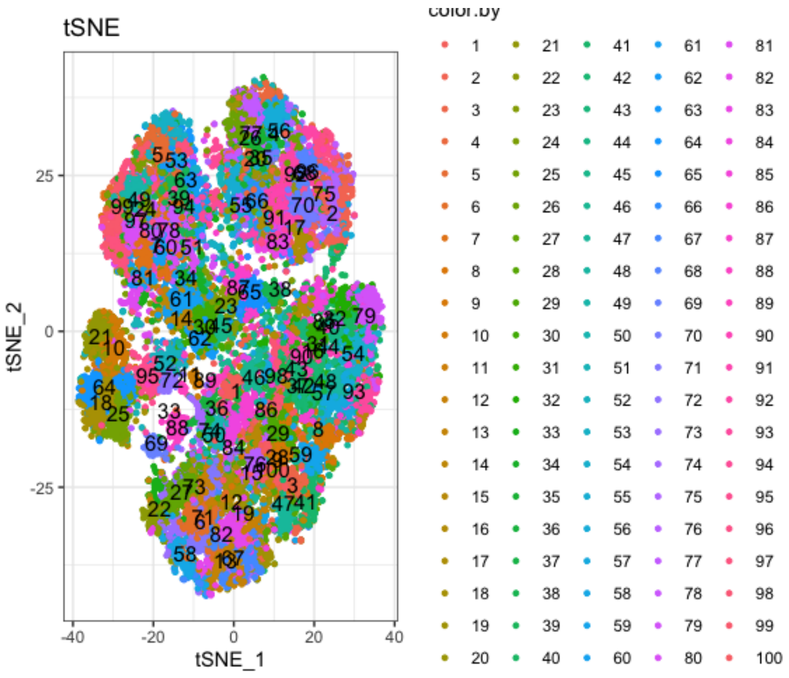
```r
# Plot 2D diffusion maps. And cells are colored by CD3 expression
plot2D(fspy, item.use = c("DC_1", "DC_2"), color.by = "CD3",
       alpha = 1, main = "diffusion maps", category = "numeric") +
  scale_colour_gradientn(colors = c("#00599F","#EEEEEE","#FF3222"))
```



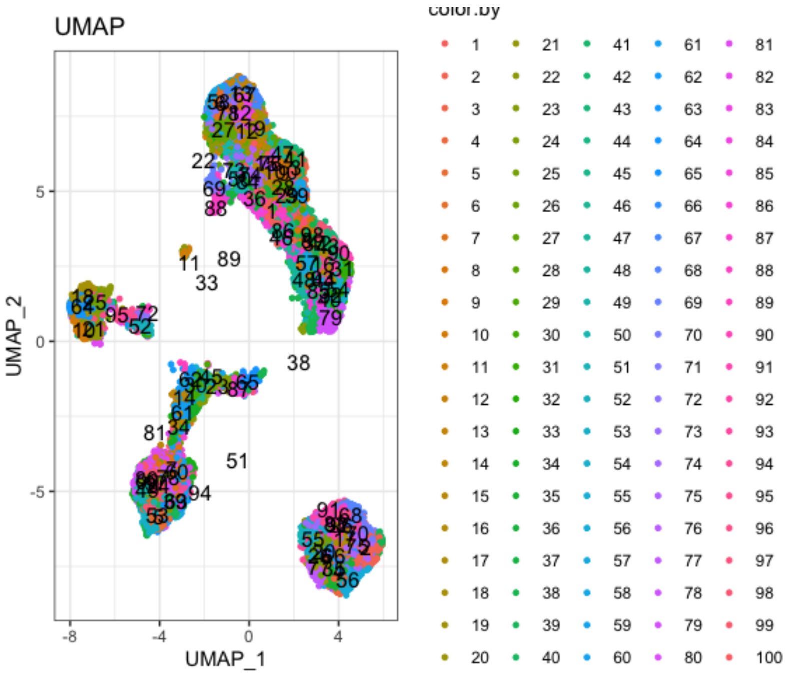
```
# Plot 2D UMAP. And cells are colored by CD3 expression
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "CD3",
       alpha = 1, main = "UMAP", category = "numeric") +
  scale_colour_gradientn(colors = c("#00599F","#EEEEEE","#FF3222"))
```



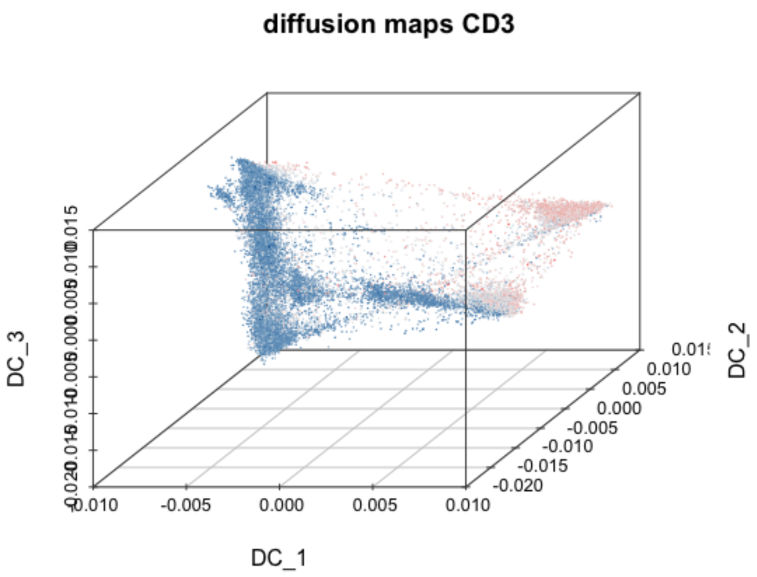
```
# Plot 2D tSNE. And cells are colored by cluster id
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "cluster.id",
       alpha = 1, main = "tSNE", category = "categorical", show.cluser.id = T)
```



```
# 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)
```

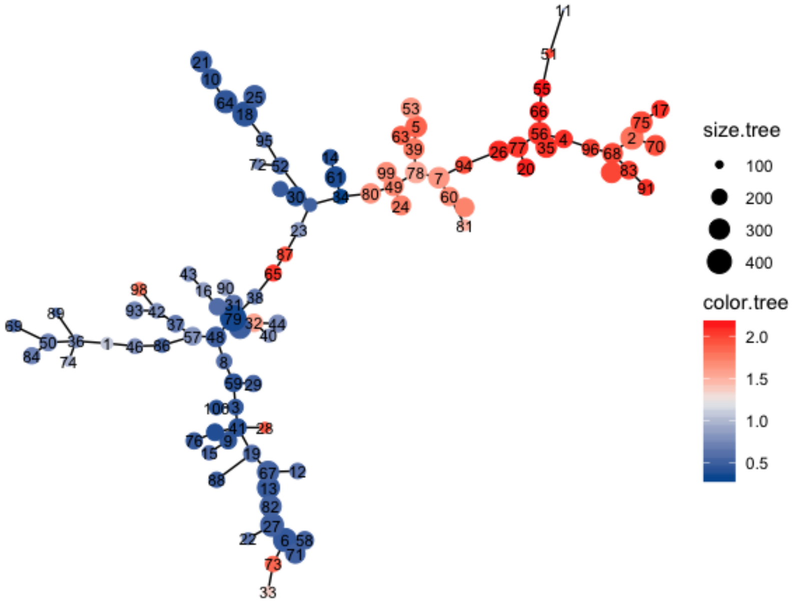


```
# Plot 3D UMAP. And cells are colored by CD45RA markers expression
plot3D(fspy, item.use = c("DC_1", "DC_2", "DC_3"), color.by = "CD3",
       main = "diffusion maps CD3", category = "numeric", size = 0.2,
       color.theme = c("#00599F","#EEEEEE","#FF3222"))
```



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

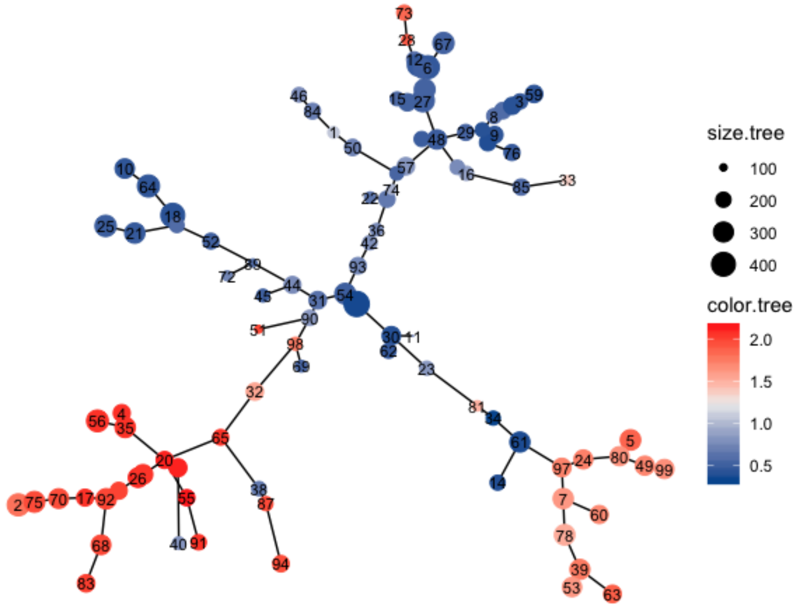
Tree plot, color.by: CD3, size.by: cell.number



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

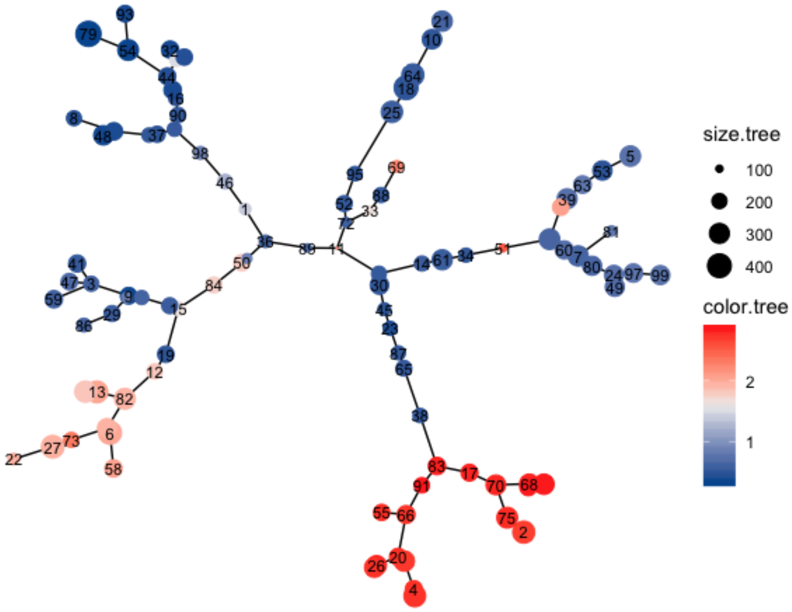
```
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

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

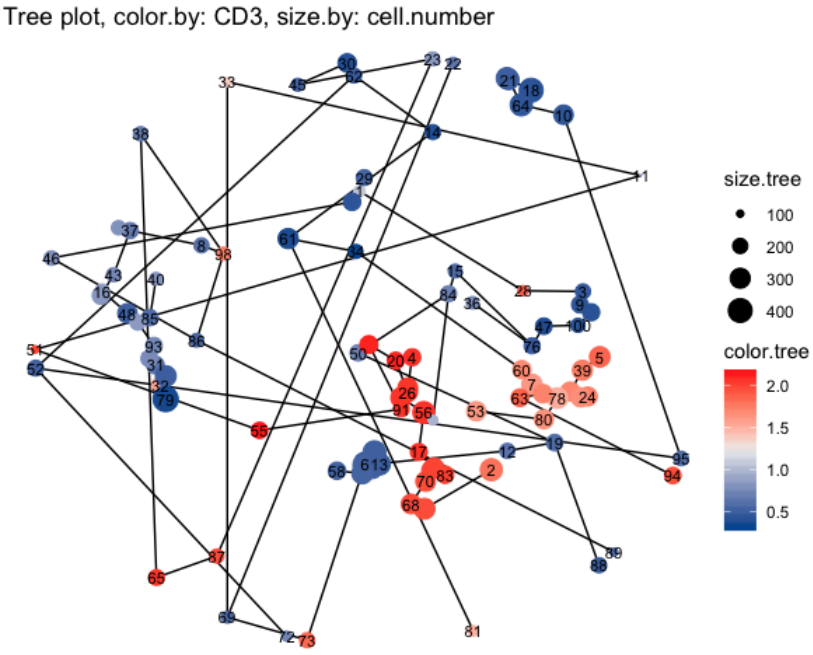


```
# 3. tSNE
fspy <- buildTree(fspy, dim.type = "tsne", dim.use = 1:2)
# Tree plot
plotTree(fspy, color.by = "CD4", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

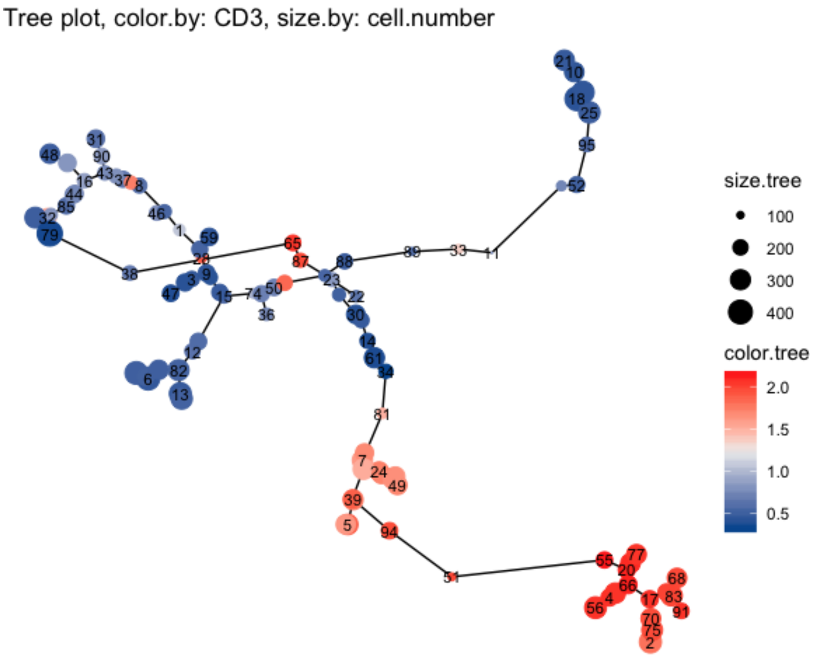
Tree plot, color.by: CD4, size.by: cell.number



```
# 4. Diffusion maps
fspy <- buildTree(fspy, dim.type = "dc", dim.use = 1:3)
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

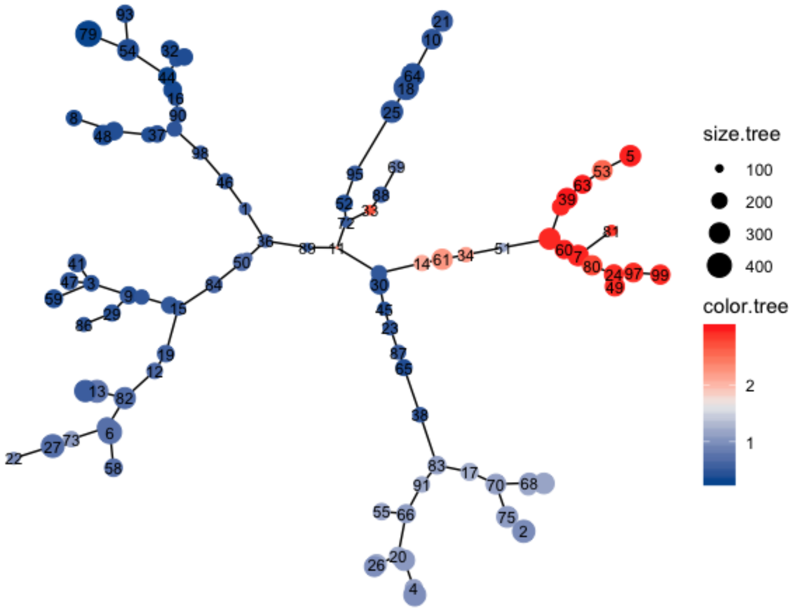


```
# 5. UMAP
fspy <- buildTree(fspy, dim.type = "umap", dim.use = 1:2)
# Tree plot
plotTree(fspy, color.by = "CD3", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



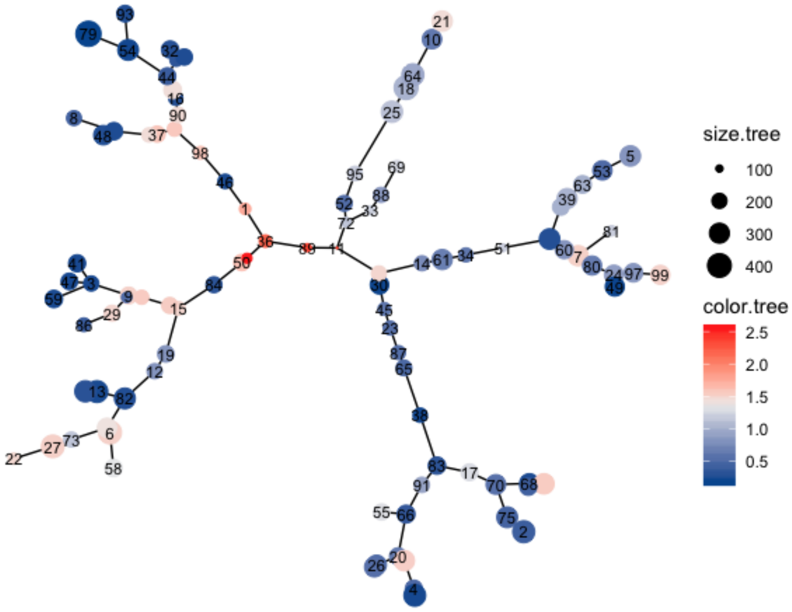
```
# The topology of a trajectory is mainly based on the interrelation
# of cell clusters, coordinates and dimensions, and in use case 1 and
# 2, we use "tsne" to construct the trajectory
fspy <- buildTree(fspy, dim.type = "tsne", dim.use = 1:2)
# Tree plot
plotTree(fspy, color.by = "CD8", show.node.name = T, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```

Tree plot, color.by: CD8, size.by: cell.number



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

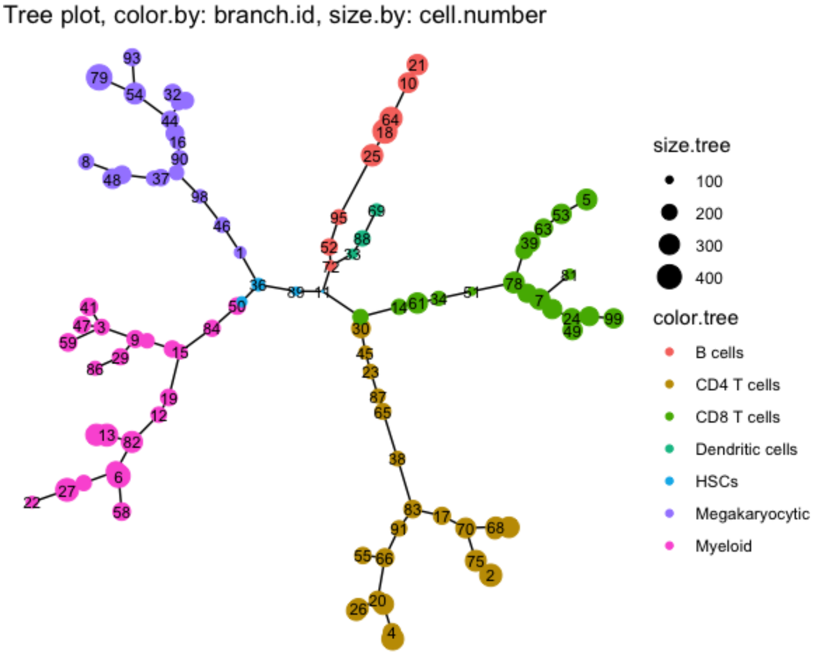
Tree plot, color.by: CD34, size.by: cell.number



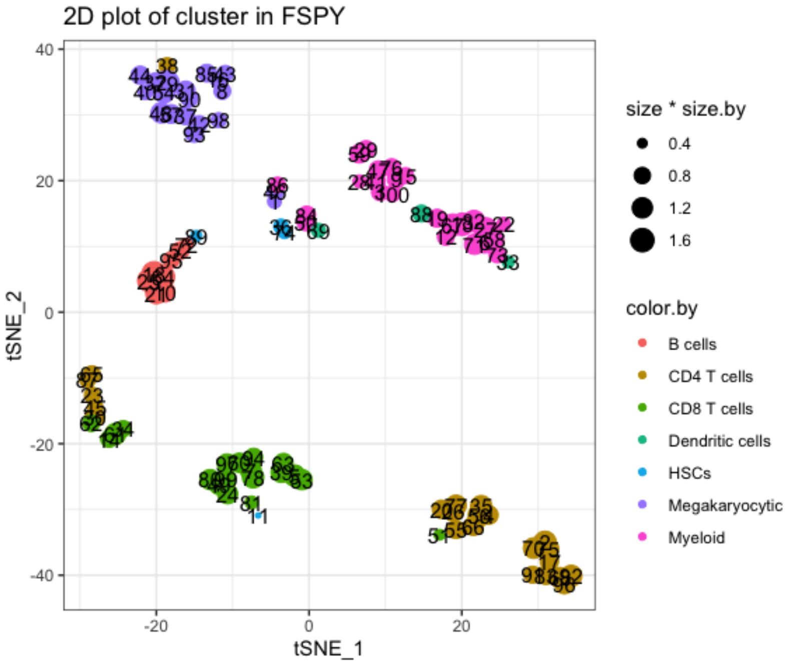
```
############# Modify branch id
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(5,2,10)] = "CD4 T cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(7,13)] = "CD8 T cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(1,6,12)] = "Megakaryocytic"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(3)] = "Dendritic cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(11)] = "B cells"
fspy@meta.data$branch.id[fspy@meta.data$branch.id %in% c(4,8,9,14)] = "Myeloid"

fspy@meta.data$branch.id[fspy@meta.data$cluster.id %in% c(74,36,89,11)] = "HSCs"
fspy@meta.data$branch.id[fspy@meta.data$cluster.id %in% c(62,14)] = "CD8 T cells"
fspy@meta.data$branch.id[fspy@meta.data$cluster.id %in% c(72)] = "B cells"

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

```
##              logFC  AveExpr         t       P.Value     adj.P.Val          B
## CD8      1.9703425 1.077508 196.15117  0.000000e+00  0.000000e+00 11415.3546
## CD45RA   0.8177313 1.618059  59.23528  0.000000e+00  0.000000e+00  1625.7650
## CD45     0.7563940 2.342068  55.48696  0.000000e+00  0.000000e+00  1437.0040
## CD33    -0.7952377 1.794166 -43.38217  0.000000e+00  0.000000e+00   895.2394
## CD4     -0.6033601 1.250727 -37.34068 1.403120e-296 3.648112e-296   667.1611
## CD3      0.4563564 1.097588  35.70954 4.560437e-272 9.880948e-272   610.7608
##              branch.contrast   Gene
## CD8     CD8 T cells_vs_other    CD8
## CD45RA CD8 T cells_vs_other CD45RA
## CD45    CD8 T cells_vs_other   CD45
## CD33    CD8 T cells_vs_other   CD33
## CD4     CD8 T cells_vs_other    CD4
## CD3     CD8 T cells_vs_other    CD3
```

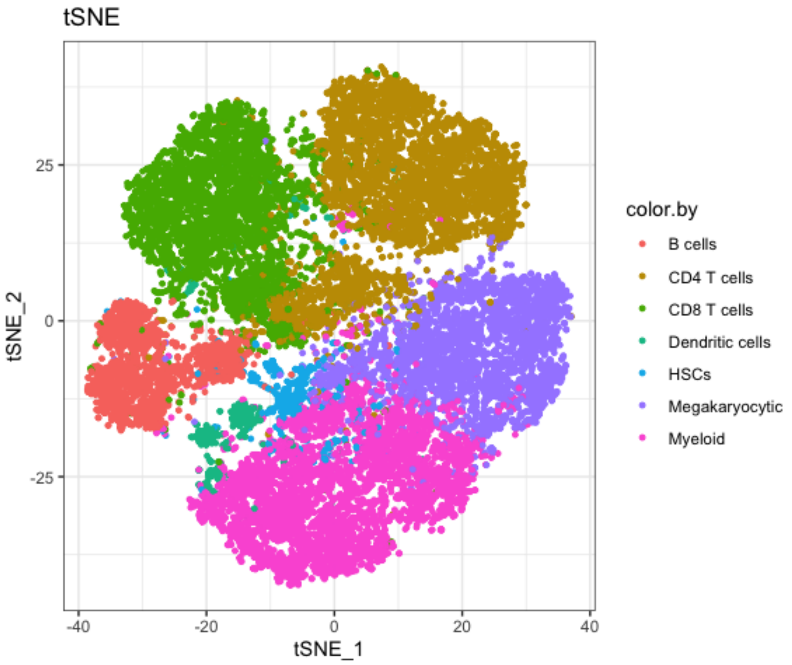
```
# plot tree
plotTree(fspy, color.by = "branch.id", show.node.name = T, cex.size = 1)
```



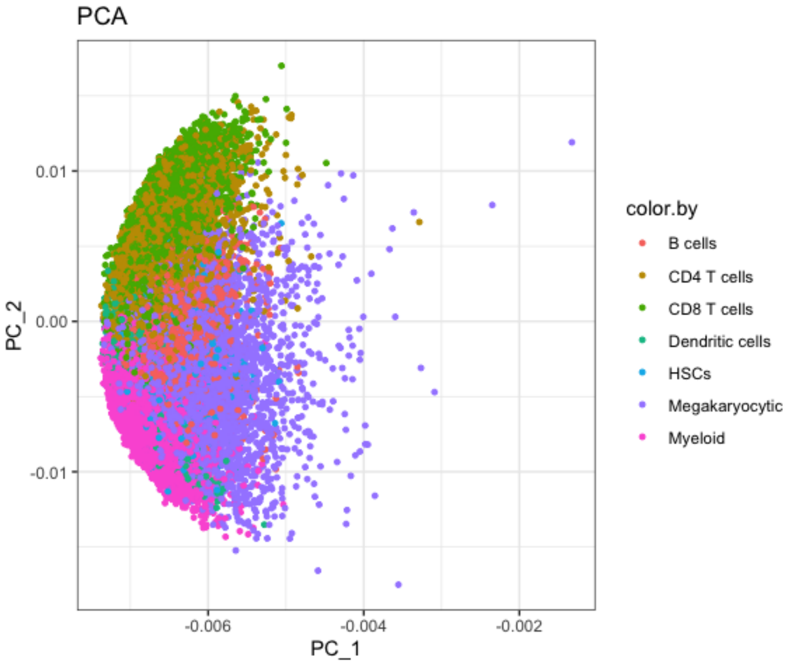
```
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "categorical",
            size = 100, color.by = "branch.id", show.cluser.id = T)
```



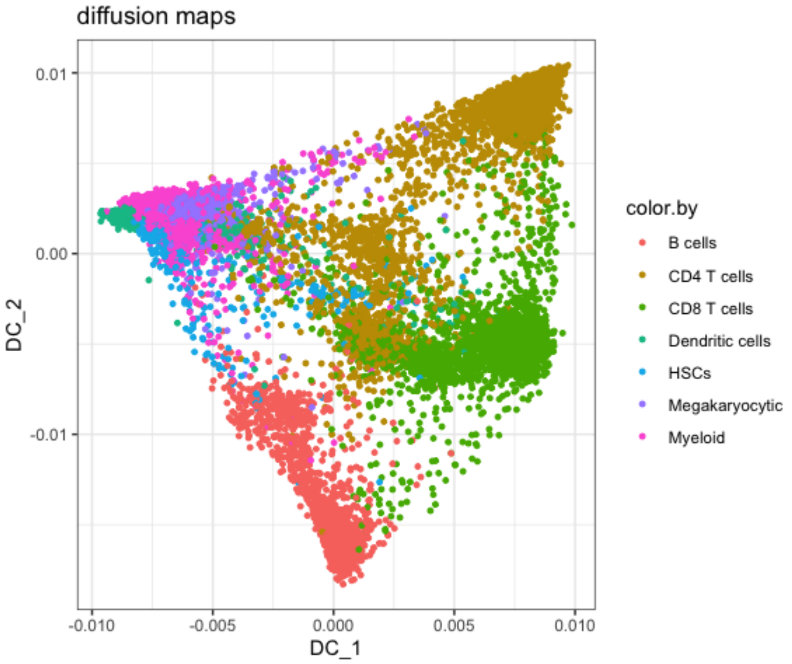
```r
# Plot 2D and colored by branch id
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "branch.id",
       alpha = 1, main = "tSNE", category = "categorical", show.cluser.id = F)
```



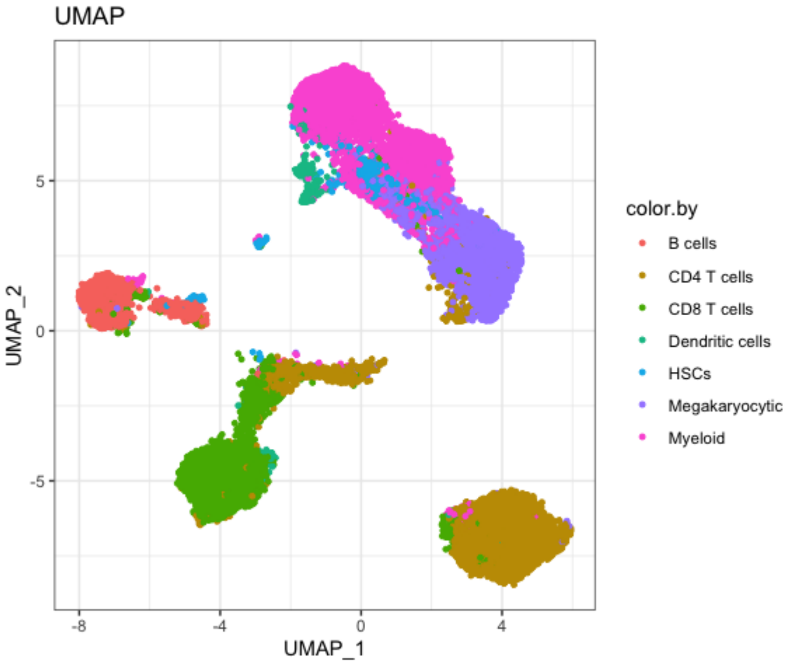
```r
plot2D(fspy, item.use = c("PC_1", "PC_2"), color.by = "branch.id",
       alpha = 1, main = "PCA", category = "categorical", show.cluser.id = F)
```



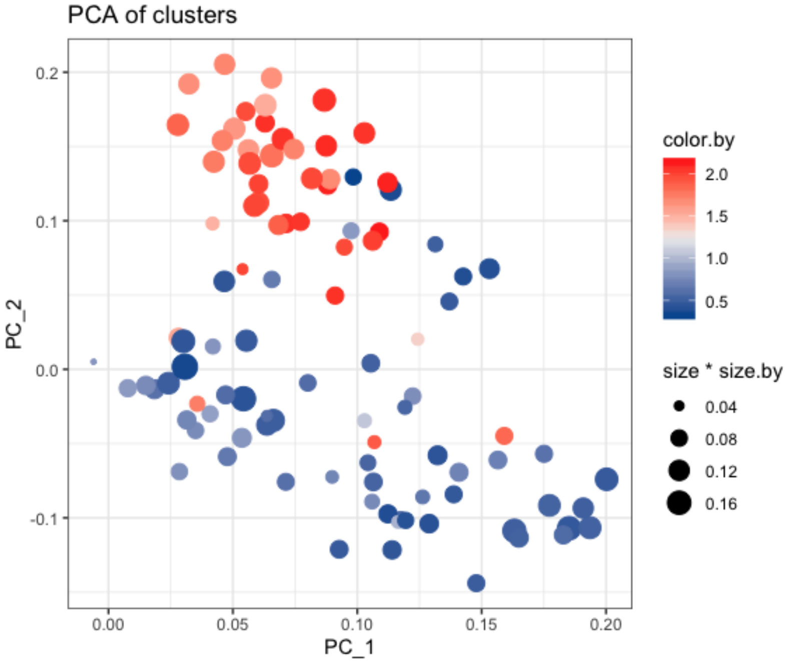
```
plot2D(fspy, item.use = c("DC_1", "DC_2"), color.by = "branch.id",
       alpha = 1, main = "diffusion maps", category = "categorical", show.cluser.id = F)
```



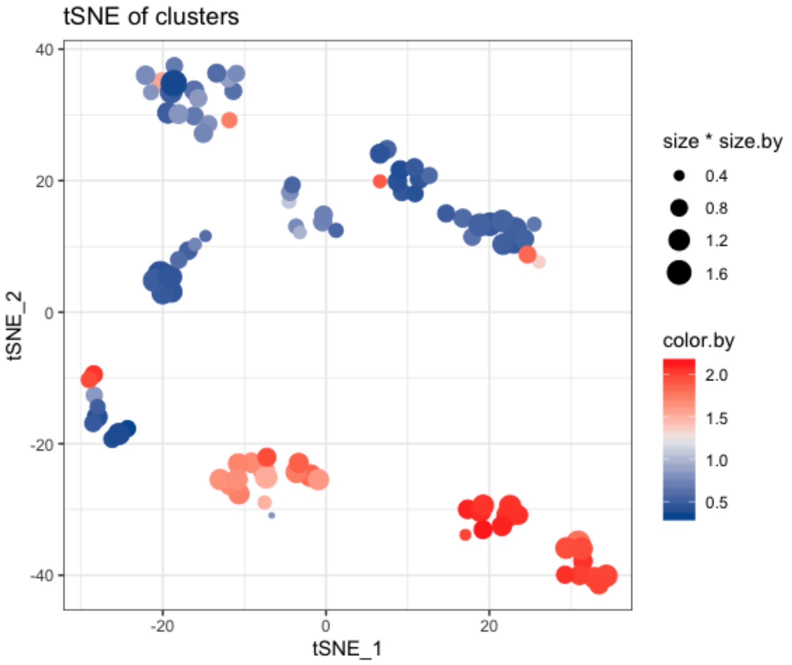
```
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "branch.id",
       alpha = 1, main = "UMAP", category = "categorical", show.cluser.id = F)
```



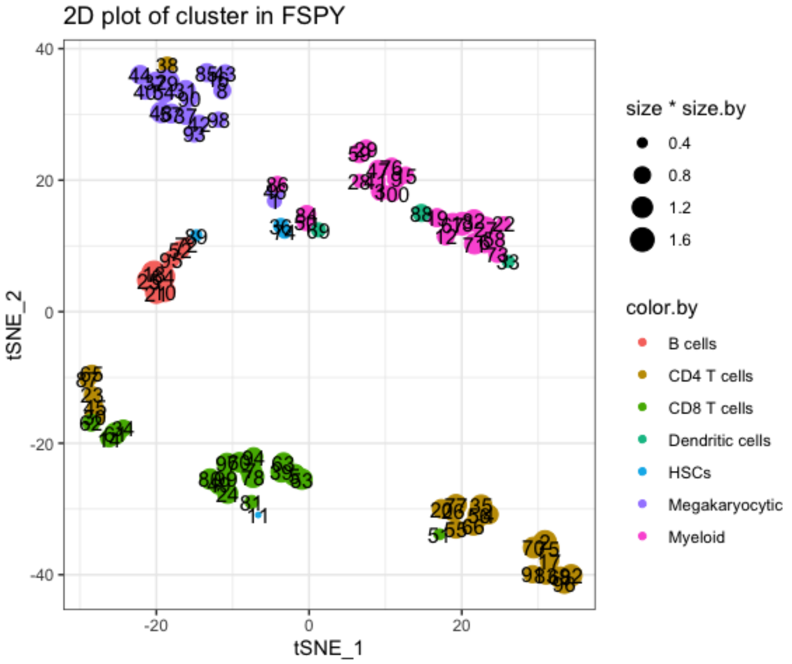
```r
# plot for clusters
plotCluster(fspy, item.use = c("PC_1", "PC_2"), category = "numeric",
            size = 10, color.by = "CD3", main = "PCA of clusters") +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



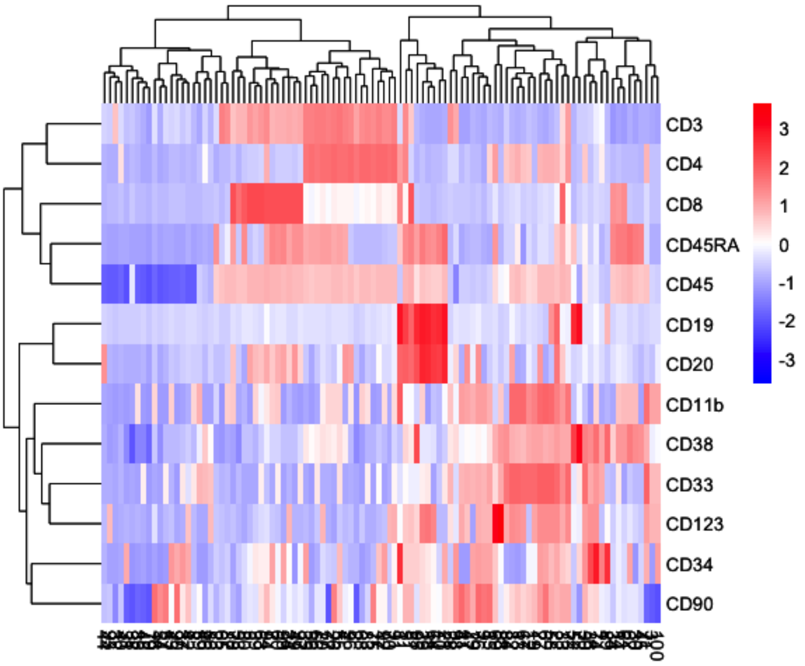
```r
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "numeric",
            size = 100, color.by = "CD3", main = "tSNE of clusters") +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



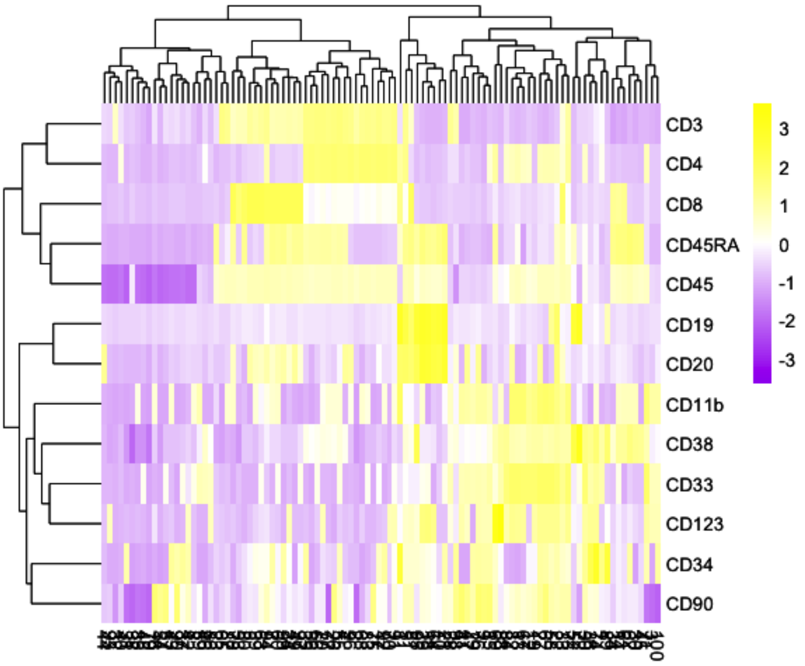
```r
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "categorical",
            size = 100, color.by = "branch.id", show.cluser.id = T)
```



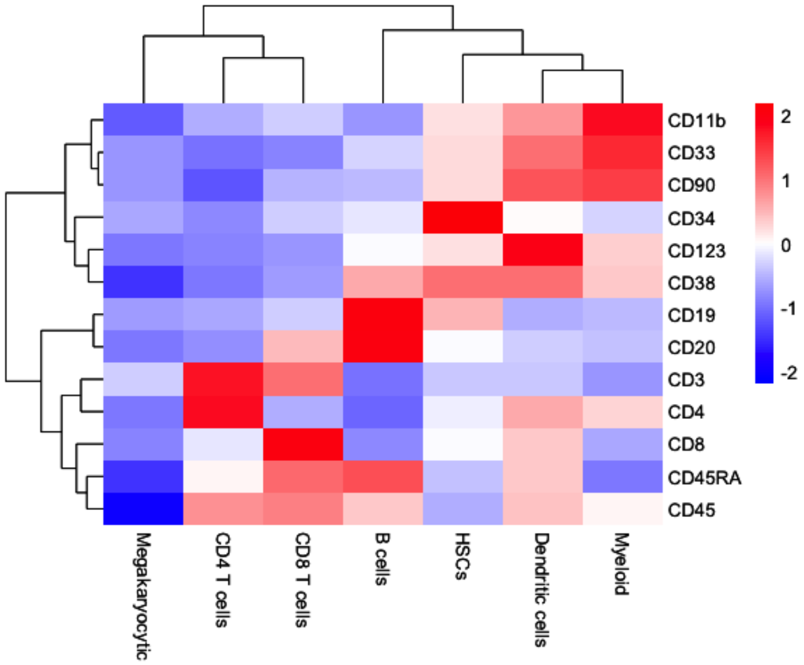
```r
# plot heatmap of clusters and branches
plotClusterHeatmap(fspy)
```



```
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(36,89,11))
fspy <- runPseudotime(fspy, verbose = T, dim.type = "raw")
```

```
## 2020-02-24 15:29:22 [INFO] Calculating Pseudotime.
```

```
## 2020-02-24 15:29:22 [INFO] Pseudotime exists in meta.data, it will be replaced.
```

```
## 2020-02-24 15:29:22 [INFO] The log data will be used to calculate trajectory
```

```
## 2020-02-24 15:31:08 [INFO] Calculating Pseudotime completed.
```

```
###### Intermediate state cells for CD8 T cells
fspy <- defLeafCells(fspy, leaf.cells = c(99,97))
fspy <- runWalk(fspy, backward.walk = F, verbose = T)
```

```
## 2020-02-24 15:31:08 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2020-02-24 15:31:12 [INFO] Generating an adjacency matrix.
```

```
## 2020-02-24 15:32:43 [INFO] Walk forward.
```

```
## 2020-02-24 15:32:47 [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(56,4,2))
```

```
## 2020-02-24 15:32:47 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

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

```
## 2020-02-24 15:32:47 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2020-02-24 15:32:51 [INFO] Generating an adjacency matrix.
```

```
## 2020-02-24 15:34:14 [INFO] Walk forward.
```

```
## 2020-02-24 15:34:22 [INFO] Calculating walk completed.
```

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

###### Intermediate state cells for DCs
fspy <- defLeafCells(fspy, leaf.cells = c(88))
```

```
## 2020-02-24 15:34:22 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

```
fspy <- runWalk(fspy, backward.walk = F)
fspy@meta.data$traj.value.log.DC <- fspy@meta.data$traj.value.log

###### Intermediate state cells for B cells
fspy <- defLeafCells(fspy, leaf.cells = c(10,21))
```

```
## 2020-02-24 15:35:59 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

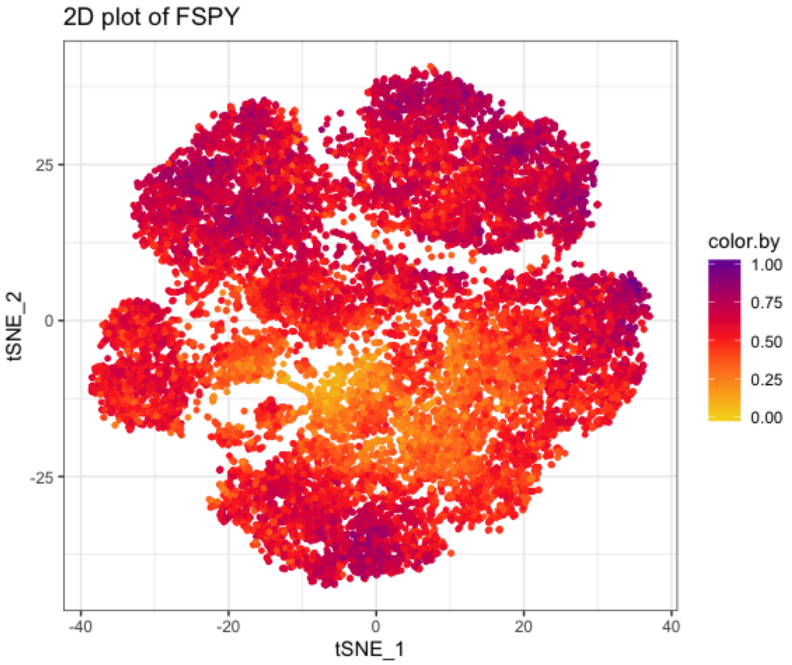
```
fspy <- runWalk(fspy, backward.walk = F)
fspy@meta.data$traj.value.log.B <- fspy@meta.data$traj.value.log

###### Intermediate state cells for monocytes and granulocytes
fspy <- defLeafCells(fspy, leaf.cells = c(22,27))
```

```
## 2020-02-24 15:37:40 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

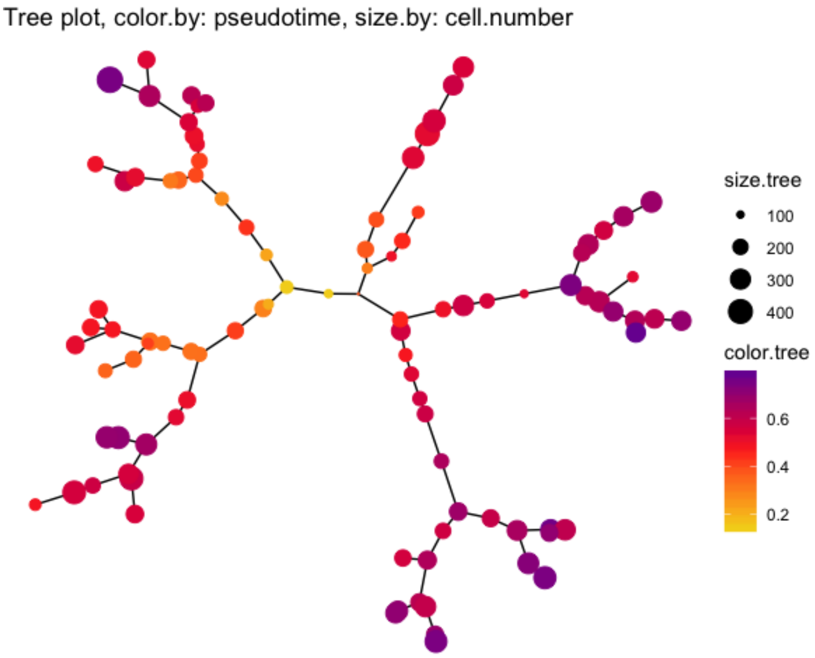
```
fspy <- runWalk(fspy, backward.walk = F)
fspy@meta.data$traj.value.log.MY <- fspy@meta.data$traj.value.log

###### Intermediate state cells for megakaryocyte and erythrocyte
fspy <- defLeafCells(fspy, leaf.cells = c(79,93))
```

```
## 2020-02-24 15:39:09 [INFO] leaf.cells in FSPY object exist, they will be replaced.
```

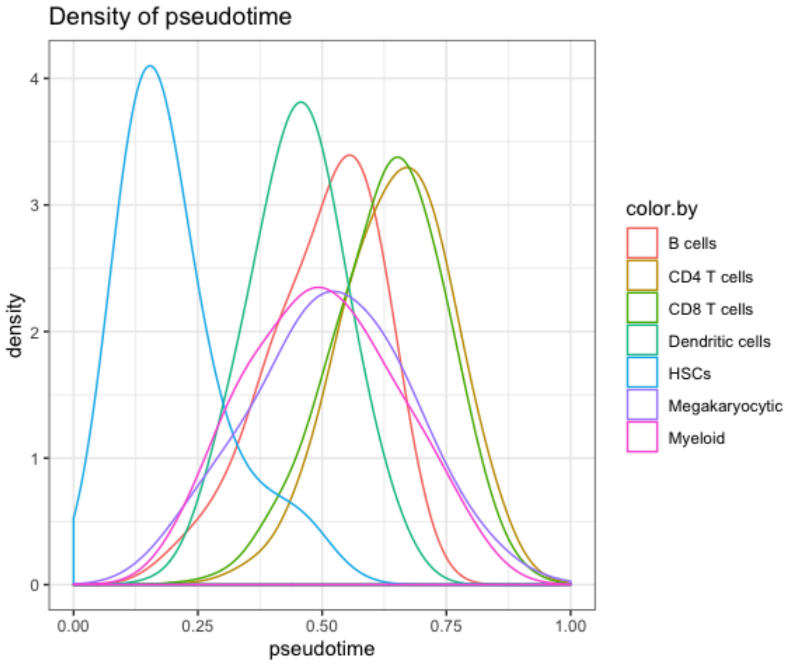
```
fspy <- runWalk(fspy, backward.walk = F)
fspy@meta.data$traj.value.log.ME <- fspy@meta.data$traj.value.log

# 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"))
```



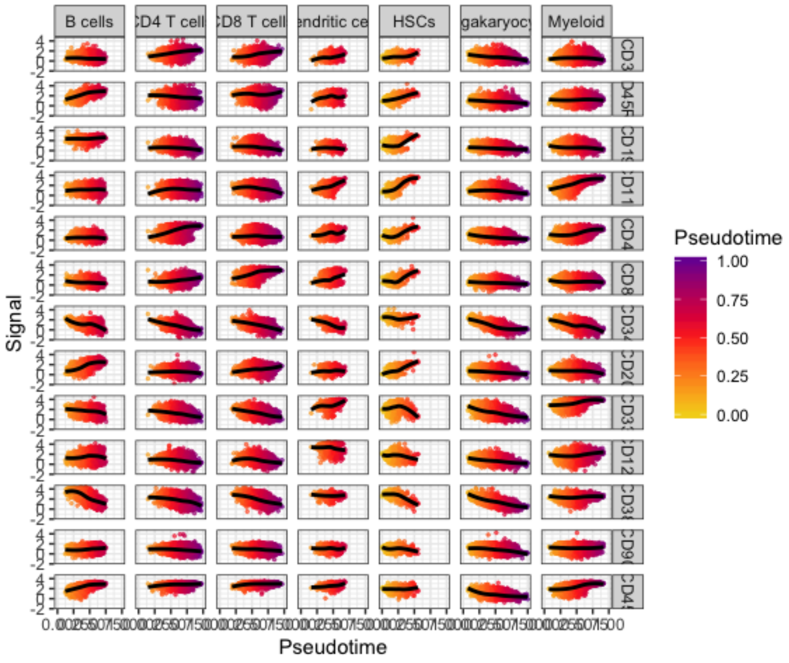
```
# Tree plot
plotTree(fspy, color.by = "pseudotime", cex.size = 1) +
  scale_colour_gradientn(colors = c("#F4D31D","#FF3222","#7A06A0"))
```



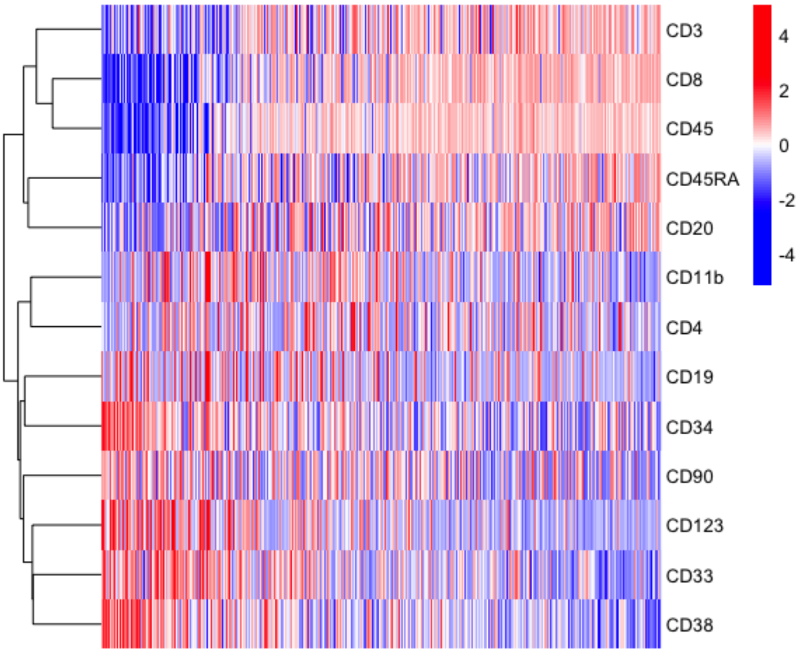
```
# pseudotime density
plotPseudotimeDensity(fspy, adjust = 2)
```



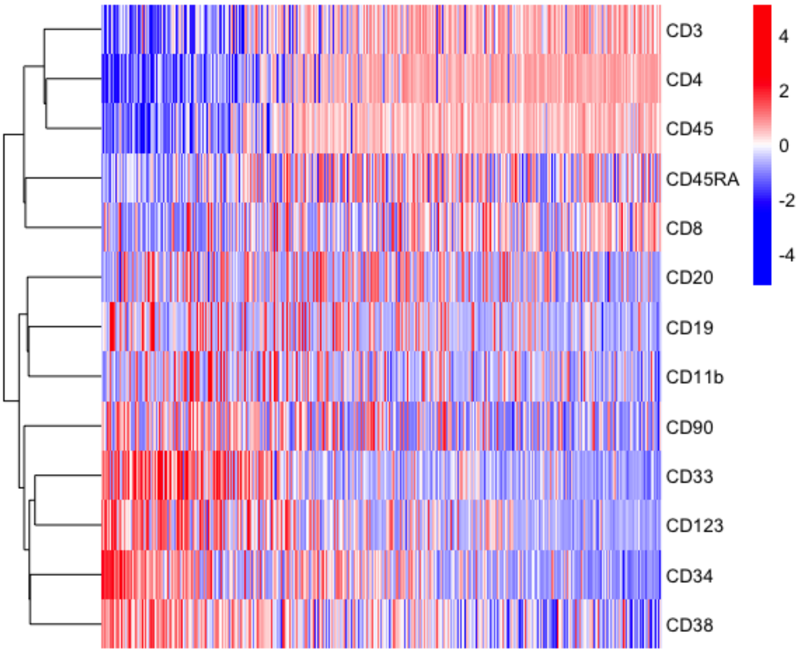
```r
plotPseudotimeTraj(fspy, var.cols = T) +
 scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```



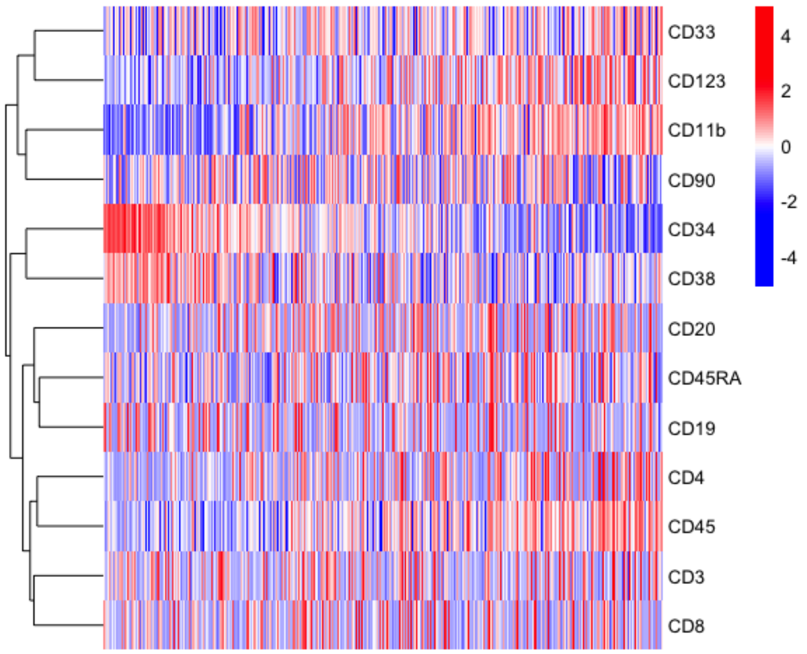
```r
### fetch plot information
plot.meta <- fetchPlotMeta(fspy, markers = markers)

# 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)
```



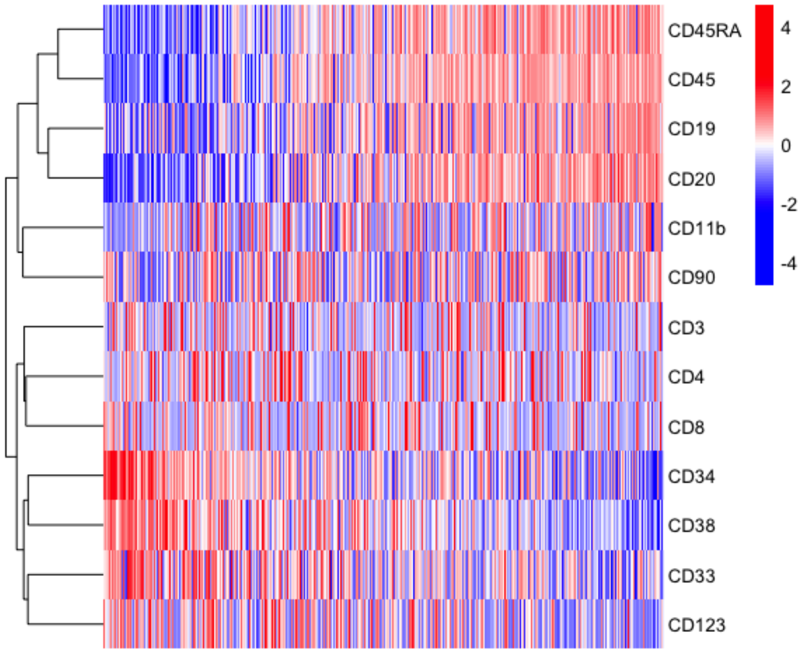
```
# heatmap for CD4 T cells
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.CD4T > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
       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



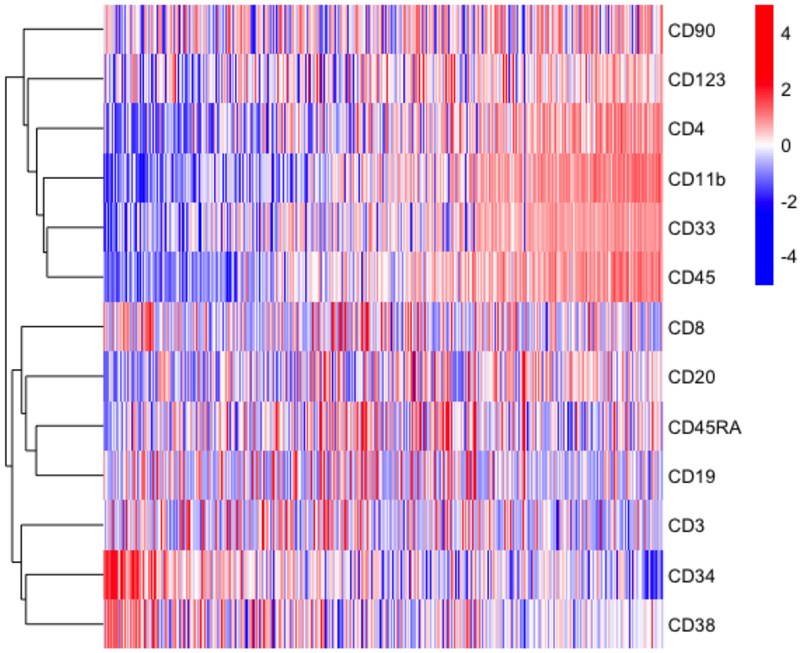
```
# heatmap for Dendritic cells
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.DC > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
     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



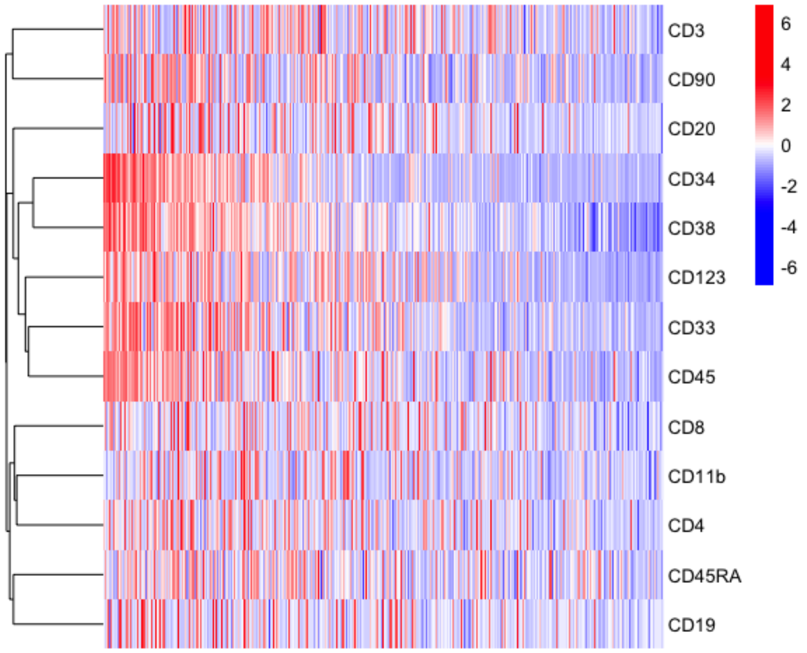
```
# heatmap for B cells
plot.meta.sub <- plot.meta[which(plot.meta$traj.value.log.B > 0), ]
plot.meta.sub <- plot.meta.sub[order(plot.meta.sub$pseudotime), ]
pheatmap(t(plot.meta.sub[, markers]), scale  = "row",
         cluster_rows = T, cluster_cols = F, cluster_method = "ward.D",
         color = 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 Catalina 10.15.3
##
## 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_1.0.2   igraph_1.2.4.2  pheatmap_1.0.12
## [5] flowCore_1.52.1 ggplot2_3.2.1  
## 
## loaded via a namespace (and not attached):
##   [1] reticulate_1.14            R.utils_2.9.2              
##   [3] ks_1.11.7                  RUnit_0.4.32               
##   [5] tidyselect_1.0.0           RSQLite_2.2.0              
##   [7] AnnotationDbi_1.48.0       grid_3.6.1                 
##   [9] ranger_0.12.1              BiocParallel_1.20.1        
##  [11] Rtsne_0.15                 scatterpie_0.1.4           
##  [13] munsell_0.5.0              destiny_3.0.1              
##  [15] codetools_0.2-16           umap_0.2.4.1               
##  [17] withr_2.1.2                colorspace_1.4-1           
##  [19] flowViz_1.50.0             Biobase_2.46.0             
##  [21] knitr_1.28                 stats4_3.6.1               
##  [23] SingleCellExperiment_1.8.0 flowClust_3.24.0           
##  [25] robustbase_0.93-5          vcd_1.4-5                  
##  [27] openCyto_1.24.0            VIM_5.1.0                  
##  [29] TTR_0.23-6                 labeling_0.3               
##  [31] GenomeInfoDbData_1.2.2     mnormt_1.5-6               
##  [33] polyclip_1.10-0            bit64_0.9-7                
##  [35] farver_2.0.3               flowWorkspace_3.34.1       
##  [37] vctrs_0.2.2                xfun_0.12                  
##  [39] ggthemes_4.2.0             R6_2.4.1                   
##  [41] GenomeInfoDb_1.22.0        clue_0.3-57                
##  [43] RcppEigen_0.3.3.7.0        bitops_1.0-6               
##  [45] DelayedArray_0.12.2        assertthat_0.2.1           
##  [47] scales_1.1.0               nnet_7.3-12                
##  [49] gtable_0.3.0               sva_3.34.0                 
##  [51] rlang_0.4.4                genefilter_1.68.0          
##  [53] scatterplot3d_0.3-41       flowUtils_1.50.0           
##  [55] splines_3.6.1              lazyeval_0.2.2             
##  [57] hexbin_1.28.1              reshape2_1.4.3             
##  [59] BiocManager_1.30.10        yaml_2.2.1                 
##  [61] abind_1.4-5                IDPmisc_1.1.20             
##  [63] RBGL_1.62.1                tools_3.6.1                
##  [65] RColorBrewer_1.1-2         proxy_0.4-23               
##  [67] BiocGenerics_0.32.0        Rcpp_1.0.3                 
##  [69] plyr_1.8.5                 base64enc_0.1-3            
##  [71] zlibbioc_1.32.0            purrr_0.3.3                
##  [73] RCurl_1.98-1.1             FlowSOM_1.18.0             
##  [75] openssl_1.4.1              S4Vectors_0.24.3           
##  [77] zoo_1.8-7                  SummarizedExperiment_1.16.1
##  [79] haven_2.2.0                cluster_2.1.0              
##  [81] fda_2.4.8.1                magrittr_1.5               
##  [83] ncdfFlow_2.32.0            data.table_1.12.8          
##  [85] RSpectra_0.16-0            openxlsx_4.1.4             
##  [87] gmodels_2.18.1             lmtest_0.9-37              
##  [89] RANN_2.6.1                 pcaMethods_1.78.0          
##  [91] mvtnorm_1.0-12             matrixStats_0.55.0         
##  [93] xtable_1.8-4               hms_0.5.3                  
##  [95] evaluate_0.14              smoother_1.1               
##  [97] XML_3.99-0.3               rio_0.5.16                 
##  [99] jpeg_0.1-8.1               mclust_5.4.5               
## [101] readxl_1.3.1               IRanges_2.20.2             
## [103] gridExtra_2.3               ggcyto_1.14.0
## [105] compiler_3.6.1              ellipse_0.4.1
## [107] tibble_2.1.3                flowStats_3.44.0
## [109] KernSmooth_2.23-16          crayon_1.3.4
## [111] R.oo_1.23.0                 htmltools_0.4.0
## [113] mgcv_1.8-31                 corpcor_1.6.9
## [115] pcaPP_1.9-73                tidyr_1.0.2
## [117] rrcov_1.5-2                 RcppParallel_4.4.4
## [119] DBI_1.1.0                   tweenr_1.0.1
## [121] MASS_7.3-51.5               boot_1.3-24
## [123] Matrix_1.2-18               car_3.0-6
## [125] R.methodsS3_1.8.0           gdata_2.18.0
## [127] parallel_3.6.1              GenomicRanges_1.38.0
## [129] forcats_0.4.0               pkgconfig_2.0.3
## [131] rvcheck_0.1.7               prettydoc_0.3.1
## [133] foreign_0.8-75              laeken_0.5.1
## [135] sp_1.3-2                    annotate_1.64.0
## [137] XVector_0.26.0              digest_0.6.24
## [139] tsne_0.1-3                  ConsensusClusterPlus_1.50.0
## [141] graph_1.64.0                rmarkdown_2.1
## [143] cellranger_1.1.0            curl_4.3
## [145] gtools_3.8.1                ggplot.multistats_1.0.0
## [147] nlme_3.1-144                lifecycle_0.1.0
## [149] jsonlite_1.6.1              carData_3.0-3
## [151] BiocNeighbors_1.4.1         askpass_1.1
## [153] limma_3.42.2                pillar_1.4.3
## [155] lattice_0.20-40             survival_3.1-8
## [157] DEoptimR_1.0-8              glue_1.3.1
## [159] xts_0.12-0                  zip_2.0.4
## [161] png_0.1-7                   bit_1.1-15.2
## [163] Rgraphviz_2.30.0            ggforce_0.3.1
## [165] class_7.3-15                stringi_1.4.6
## [167] blob_1.2.1                  RcppHNSW_0.2.0
## [169] CytoML_1.12.0               memoise_1.1.0
## [171] latticeExtra_0.6-29         dplyr_0.8.4
## [173] knn.covertree_1.0           irlba_2.3.3
## [175] e1071_1.7-3
```

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.