Quick start of flowSpy

Yuting Dai

2019-09-11

Abstract

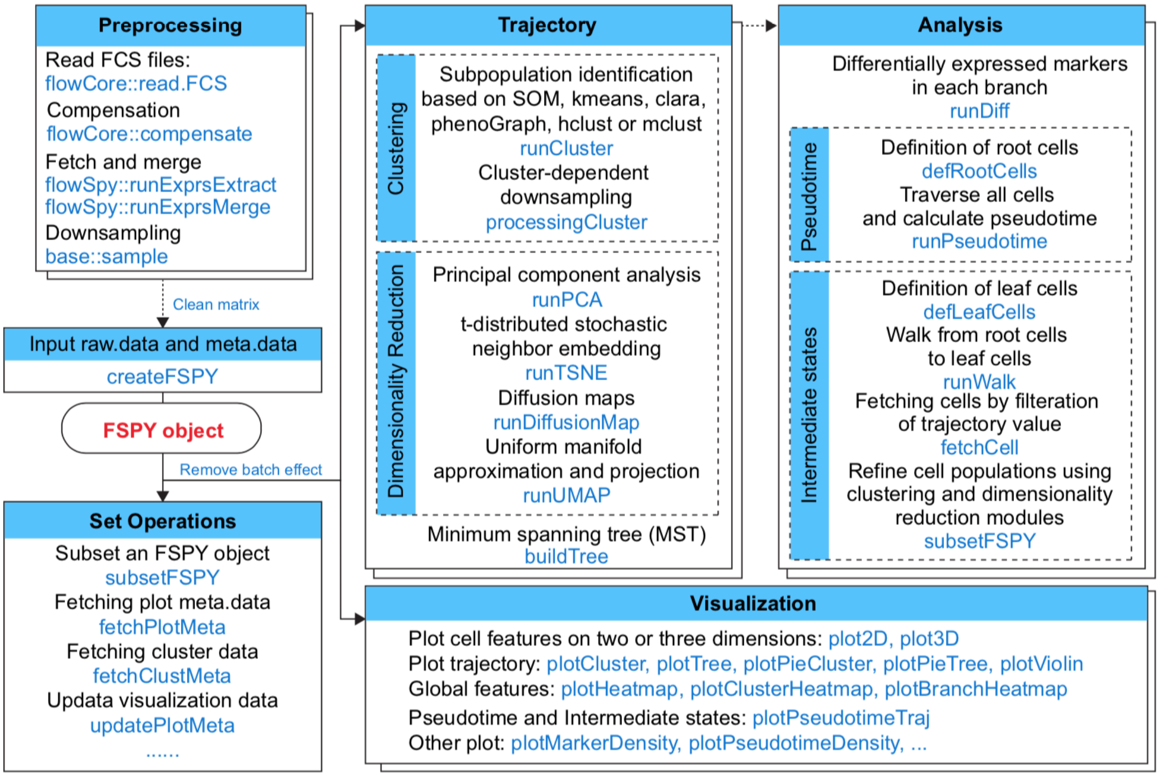
Although multidimensional single-cell-based flow and mass cytometry have been increasingly applied to microenvironmental composition and stem-cell research, integrated analysis workflows to facilitate the interpretation of experimental cytometry data remain underdeveloped. We present flowSpy, a comprehensive R package designed for the analysis and interpretation of flow and mass cytometry data. We applied flowSpy to mass cytometry and time-course flow cytometry data to demonstrate the usage and practical utility of its computational modules. flowSpy is a reliable tool for multidimensional cytometry data workflows and produces compelling results for trajectory construction and pseudotime estimation.

Introduction

Multidimensional single-cell-based assays with flow or mass cytometric signals are widely used in cellular subpopulations identification, tissue microenvironment composition, clinical immunophenotyping and differential lineage reconstruction [1]. Moden fluorescence-based flow cytometers typically can detect up to 20 features on the single-cell level in one routine case, whereas mass cytometers promise to increase this capacity towards 50 [2]. Despite the traditional manual gating strategies in flow cytometry data, effective analysis methods on multi-dimensional cytometric data still require promotion on trajectory inference, pseudotime estimation, visualization and workflow standardization. Especialy for human hematopoietic differentiation and cancer research, reconstructing lineage relationships between cells within a tissue is a long-standing aim. With the fast development of Single-cell RNA Sequencing (scRNA-seq), some effective computational modules and methods refined from scRNA-seq can be enrolled in the design of flow and mass cytometry analysis workflow, such as Seurat [3], Monocle [4] and scmap [5]. However, different from scRNA-seq data, flow and mass cytometry data do not contain a large proportion of zeros for expression genes and only make concentration on a subset of cellular makers on protein expression levels [3]. Generally, most automated pipelines designed for scRNA-seq do not perform well for flow and mass cytometry data, while several basic algorithms can be enrolled in the flow and mass cytometry workflow. For example, linear or nonlinear dimensionality reduction techniques such as principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE, renamed viSNE) [6] and diffusion maps [7]. Recently, uniform manifold approximation and projection (UMAP) has claimed good performance on dimensionality reduction as well as visualization for single-cell data [8]. It brings new insights into flow and mass cytometry data analysis.

Both flow and mass cytometry data are stored in the standard format which called Flow Cytometry Standard (FCS). FCS file can be read and written by tools on different operating platforms [9]. Several classical tools developed for characterizing performance standardization, calibration and gating control of FCS data such as flowCore [10], flowVis [11], FlowSOM [12], SPADE [13] and Cytofkit [14]. Most of them run in the R statistical computing environment and perform well on gating, clustering and visualization. Facing with increasing degrees of multi-dimensional FCS data, several emerging research areas such as developmental trajectory inference and lineage tracing are still under challenge [15]. In accordance with well-established standards and practices [10-14, 16, 17], we present flowSpy, a trajectory reconstruction, pseudotime estimation and visualization toolkit for flow and mass cytometry data. The flowSpy package is built in R and can complete analyzing workflow for flow and mass cytometry data such as subpopulation classfication, dimensionality reduction, trajectory inference, pseudotime estimation, intermediate states identification and visualization. flowSpy runs across platforms such as UNIX, Windows and MacOS and provides an up-to-date, feature-rich and readily scalable workflow.

Overview of flowSpy workflow


Preprocessing
Read FCS files:
flowCore::read.FCS
Compensation
flowCore::compensate
Fetch and merge
flowSpy::runExprsExtract
flowSpy::runExprsMerge
Downsampling
base::sample
Clean matrix
Input raw.data and meta.data
createFSPY
FSPY object
Remove batch effect
Set Operations
Subset an FSPY object
subsetFSPY
Fetching plot meta.data
fetchPlotMeta
Fetching cluster data
fetchClustMeta
Updata visualization data
updatePlotMeta
......
Trajectory
Clustering
Subpopulation identification based on SOM, kmeans, clara, phenoGraph, hclust or mclust
runCluster
Cluster-dependent downsampling
processingCluster
Dimensionality Reduction
Principal component analysis
runPCA
t-distributed stochastic neighbor embedding
runTSNE
Diffusion maps
runDiffusionMap
Uniform manifold approximation and projection
runUMAP
Minimum spanning tree (MST)
buildTree
Analysis
Differentially expressed markers in each branch
runDiff
Pseudotime
Definition of root cells
defRootCells
Traverse all cells and calculate pseudotime
runPseudotime
Intermediate states
Definition of leaf cells
defLeafCells
Walk from root cells to leaf cells
runWalk
Fetching cells by filteration of trajectory value
fetchCell
Refine cell populations using clustering and dimensionality reduction modules
subsetFSPY
Visualization
Plot cell features on two or three dimensions: plot2D, plot3D
Plot trajectory: plotCluster, plotTree, plotPieCluster, plotPieTree, plotViolin
Global features: plotHeatmap, plotClusterHeatmap, plotBranchHeatmap
Pseudotime and Intermediate states: plotPseudotimeTraj
Other plot: plotMarkerDensity, plotPseudotimeDensity, ...

Workflow of flowSpy

The flowSpy package is developed to complete the majority of standard analysis and visualization workflow for FCS data. In flowSpy workflow, an S4 object in R is built to implement the statistical and computational approach, and all computational modules are integrated into one single channel which only requires a specified input data format. Computational modules of flowSpy can be divided into four main parts (Fig. 1): preprocessing, trajectory, analysis and visualization. - **Preprocessing**. Data import, compensation, quality control, filtration, normalization and merge cells from different samples can be implemented in the preprocessing module. After preprocessing, a matrix contains clean cytometric signaling data, a data.frame containing meta-information of the experiment and a vector contains all markers enrolled in the computational process are required to build an FPSY object. - **Trajectory**. Cells built in the FSPY object are classfied into different clusters based on the expression level of input markers. You can choose different clustering methods by inputting different parameters. After clustering, cells are downsampled in a cluster-dependent fashion to reduce the total cell size and avoid small cluster deletion. Dimensionality reduction for both cells and clusters are also implemented in the clustering procedure. After dimensionality reduction, we use Minimus Spanning Tree (MST) to construct cell trajectory. - **Analysis**. This module is design for time couse FCS data. Before running pseudotime, root cells must be defined first based on users' priori knowledge. Root cells in flowSpy workflow are the initial cells of the trajectory tree. So it can be set using one vertex node of the tree or a cluster of cells with specific antibodies combination. Intermediate state evaluation is also envolved in the pseudotime module. Leaf cells are defined by the end node of trajectory tree or the end stage of the experiment. Intermediate state cells are cells with higher betweenness in the graph built on cell-cell connection, which play an important role between the connection of root cells and leaf cells. - **Visualization**. The visualization module can provide clear and concise visualization of FCS data in an effective and easy-to-comprehend manner. flowSpy package offers various plotting functions to generate customizable and publication-quality plots. Two-dimensional or three-dimensional plot can fit most requirement from dimensionality reduction results. And tree-based plot can visualize cell trajectory as force-directed layout tree. Other special plots such as heatmap and violin plot are also provided in flowSpy.

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

# Read fcs files
fcs.path <- system.file("extdata", package = "flowSpy")
fcs.files <- list.files(fcs.path, pattern = '.FCS$', full = TRUE)

fcs.data <- runExprsMerge(fcs.files, comp = FALSE, transformMethod = "none")

# Refine colnames of fcs data
recol <- c(`FITC-A<CD43>` = "CD43", `APC-A<CD34>` = "CD34",
           `BV421-A<CD90>` = "CD90", `BV510-A<CD45RA>` = "CD45RA",
           `BV605-A<CD31>` = "CD31", `BV650-A<CD49f>` = "CD49f",
           `BV 735-A<CD73>` = "CD73", `BV786-A<CD45>` = "CD45",
           `PE-A<FLK1>` = "FLK1", `PE-Cy7-A<CD38>` = "CD38")
colnames(fcs.data)[match(names(recol), colnames(fcs.data))] = recol
fcs.data <- fcs.data[, recol]

day.list <- c("D0", "D2", "D4", "D6", "D8", "D10")
meta.data <- data.frame(cell = rownames(fcs.data),
                        stage = str_replace(rownames(fcs.data), regex(".FCS.+"), "") )
meta.data$stage <- factor(as.character(meta.data$stage), levels = day.list)

markers <- c("CD43","CD34","CD90","CD45RA","CD31","CD49f","CD73","CD45","FLK1","CD38")

# Build the FSPY object
fspy <- createFSPY(raw.data = fcs.data, markers = markers,
                   meta.data = meta.data,
                   normalization.method = "log",
                   verbose = TRUE)
```

```
## 2019-09-11 22:08:47 [INFO] Number of cells in processing: 600
```

```
## 2019-09-11 22:08:47 [INFO] rownames of meta.data and raw.data will be named using column
```

```
## 2019-09-11 22:08:47 [INFO] Index of markers in processing
```

```
## 2019-09-11 22:08:47 [INFO] Creating FSPY object.
```

```
## 2019-09-11 22:08:47 [INFO] Determining normalization factors
```

```
## 2019-09-11 22:08:47 [INFO] Normalization and log-transformation.
```

```
## 2019-09-11 22:08:47 [INFO] Build FSPY object succeed
```

```
# See information
fspy
```

```
## FSPY Information:
##  Input cell number: 600  cells
##  Enroll marker number: 10  markers
##  Cells after downsampling: 600  markers
```

```
# Cluster cells by SOM algorithm
# Set random seed to make results reproducible
set.seed(1)
fspy <- runCluster(fspy, cluster.method = "som")
```

```
## Mapping data to SOM
```

```
# Do not perform downsampling
set.seed(1)
fspy <- processingCluster(fspy)

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

# run t-Distributed Stochastic Neighbor Embedding (tSNE)
fspy <- runTSNE(fspy)

# run Diffusion map
fspy <- runDiffusionMap(fspy)

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

# build minimum spanning tree based on tsne
fspy <- buildTree(fspy, dim.type = "tsne", dim.use = 1:2)

# DEGs of different branch
diff.list <- runDiff(fspy)

# define root cells
fspy <- defRootCells(fspy, root.cells = c(28,26))

# run pseudotime
fspy <- runPseudotime(fspy, verbose = TRUE, dim.type = "raw")
```

```
## 2019-09-11 22:08:54 [INFO] Calculating Pseudotime.
```

```
## 2019-09-11 22:08:54 [INFO] Pseudotime exists in meta.data, it will be replaced.
```

```
## 2019-09-11 22:08:54 [INFO] The log data will be used to calculate trajectory
```

```
## 2019-09-11 22:08:54 [INFO] Calculating Pseudotime completed.
```

```
# define leaf cells
fspy <- defLeafCells(fspy, leaf.cells = c(27, 13), verbose = TRUE)
```

```
## 2019-09-11 22:08:54 [INFO] 37 cells will be added to leaf.cells .
```

```
# run walk between root cells and leaf cells
fspy <- runWalk(fspy, verbose = TRUE)
```

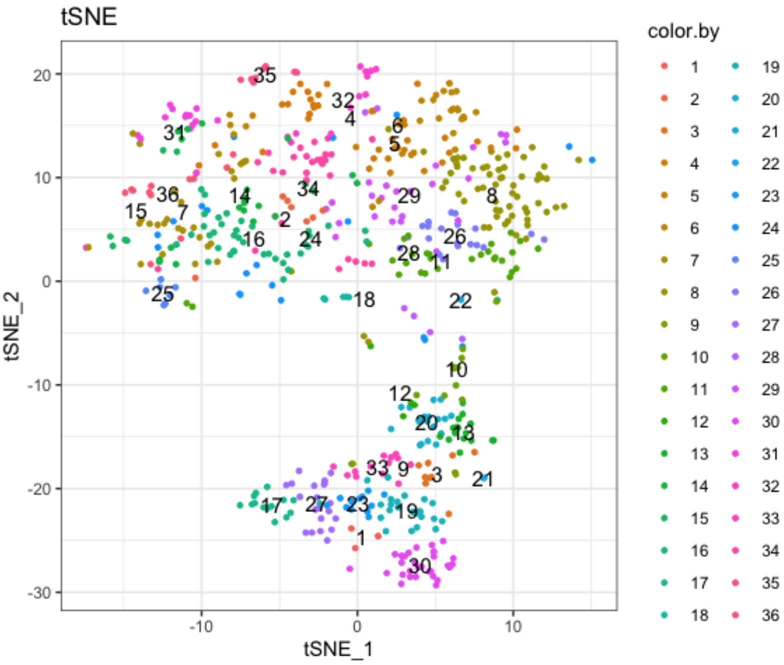
```
## 2019-09-11 22:08:54 [INFO] Calculating walk between root.cells and leaf.cells .
```

```
## 2019-09-11 22:08:54 [INFO] Generating an adjacency matrix.
```

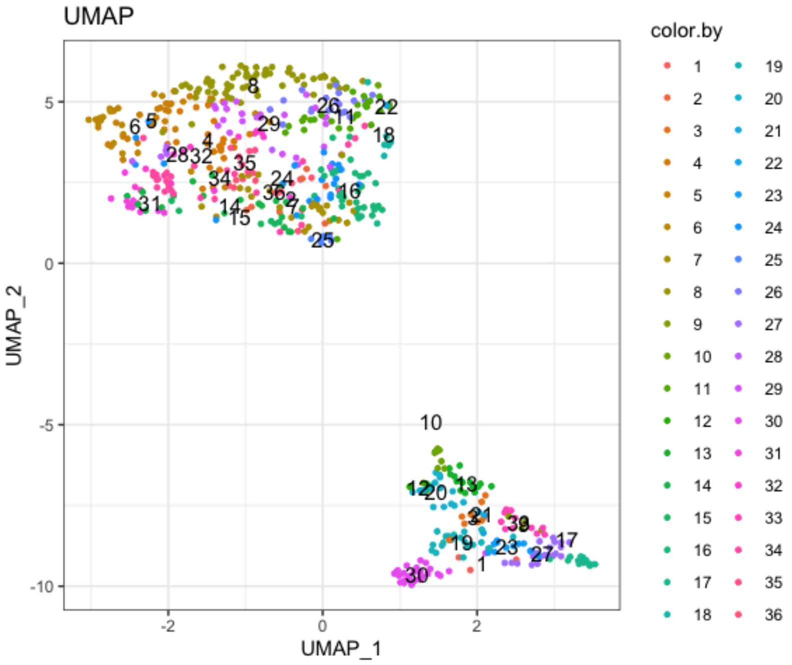
```
## 2019-09-11 22:08:54 [INFO] Walk forward.
```

```
## 2019-09-11 22:08:54 [INFO] Calculating walk completed.
```

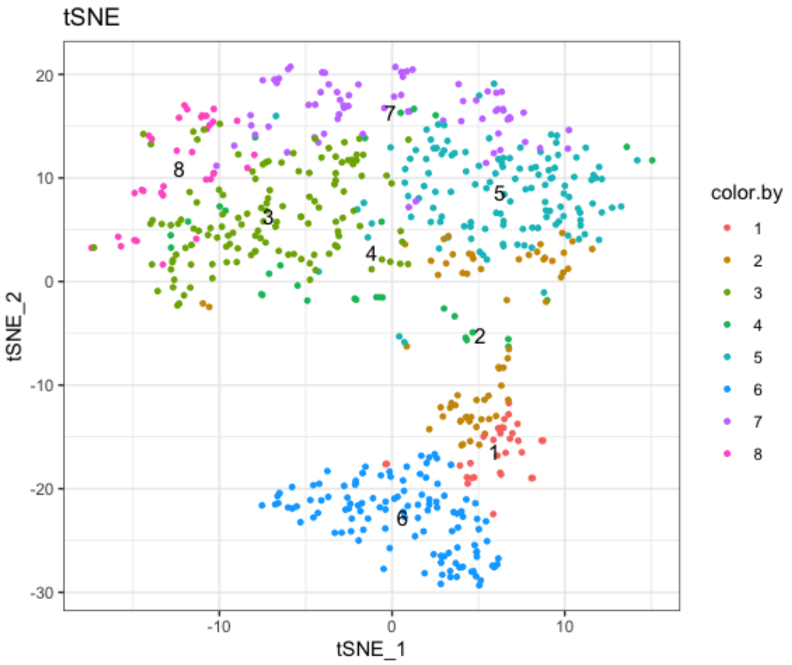
```
# Save object
if (FALSE) {
  save(fspy, file = "Path to you output directory")
}

######################## Visualization

# 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 = TRUE)
```



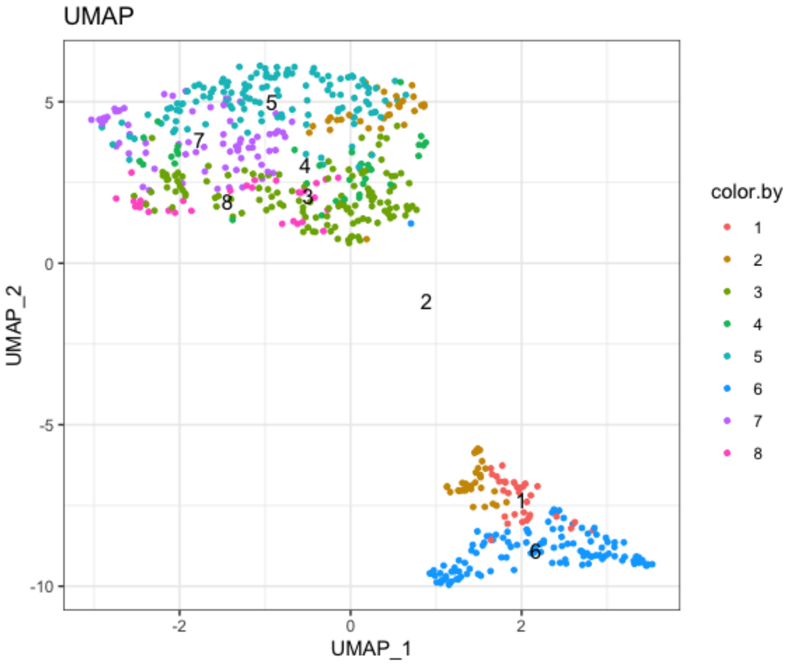
```
# Plot 2D UMAP. And cells are colored by cluster id
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "cluster.id",
       alpha = 1, main = "UMAP", category = "categorical", show.cluser.id = TRUE)
```



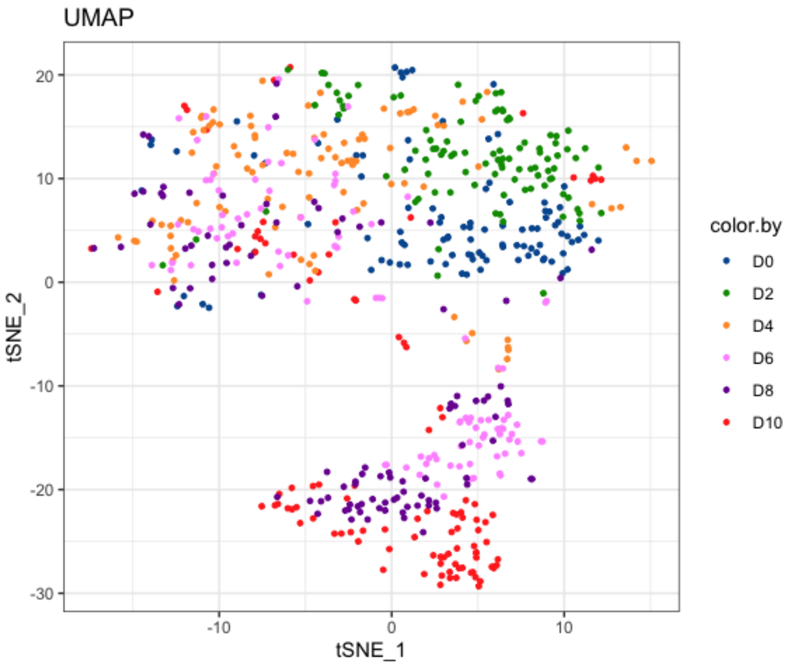
```
# Plot 2D tSNE. And cells are colored by cluster id
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "branch.id",
       alpha = 1, main = "tSNE", category = "categorical", show.cluser.id = TRUE)
```



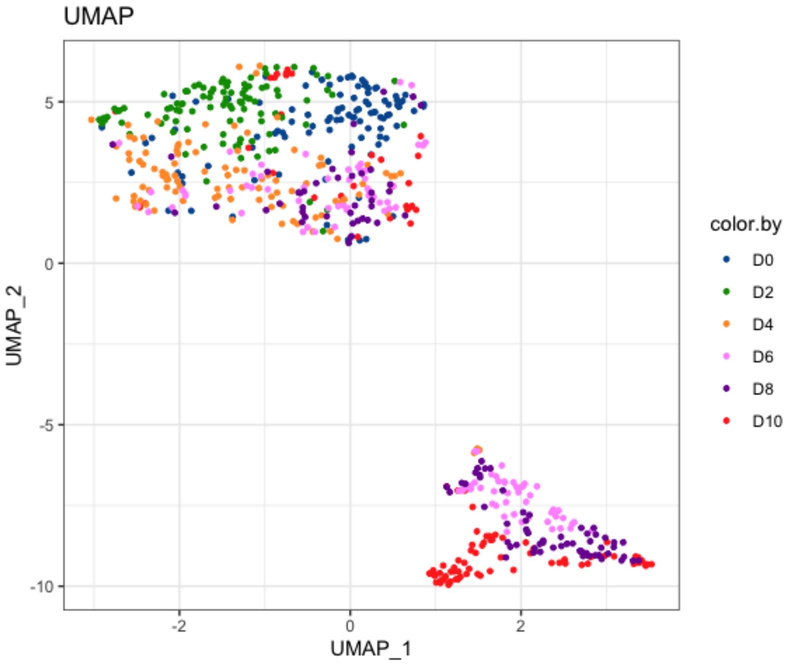
```
# Plot 2D UMAP. And cells are colored by cluster id
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "branch.id",
       alpha = 1, main = "UMAP", category = "categorical", show.cluser.id = TRUE)
```



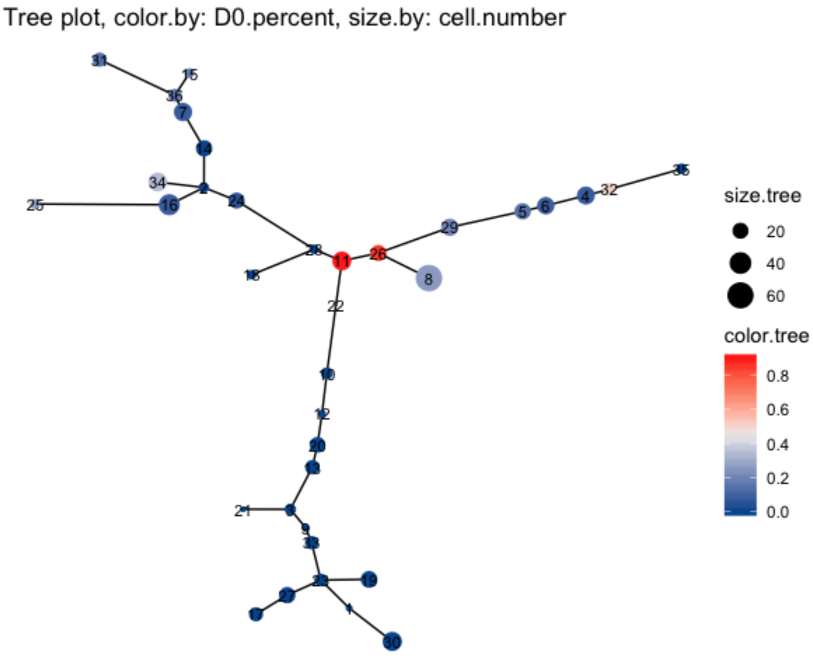
```
# Plot 2D tSNE. And cells are colored by stage
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "stage",
       alpha = 1, main = "UMAP", category = "categorical") +
   scale_color_manual(values = c("#00599F","#009900","#FF9933",
                                 "#FF99FF","#7A06A0","#FF3222"))
```



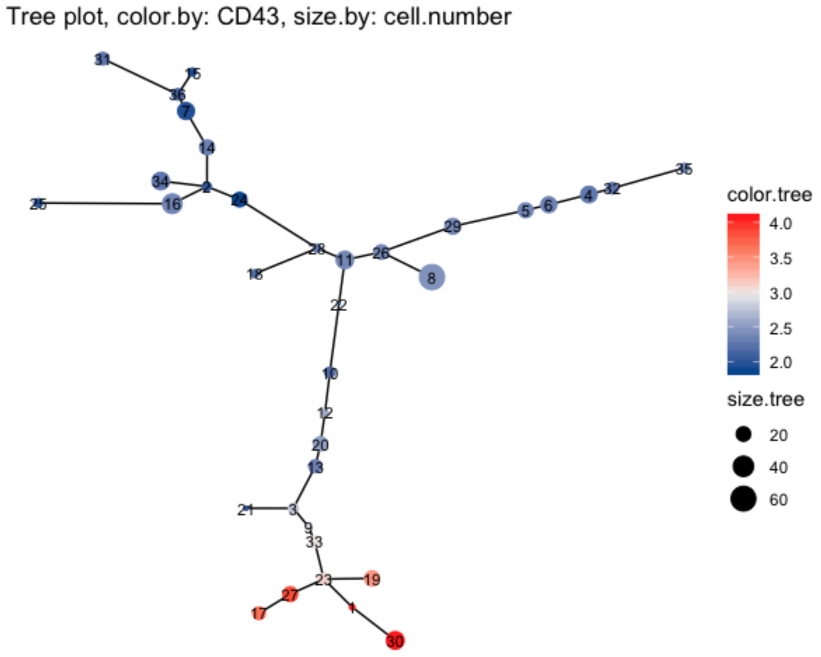
```
# Plot 2D UMAP. And cells are colored by stage
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), color.by = "stage",
       alpha = 1, main = "UMAP", category = "categorical") +
   scale_color_manual(values = c("#00599F","#009900","#FF9933",
                                 "#FF99FF","#7A06A0","#FF3222"))
```



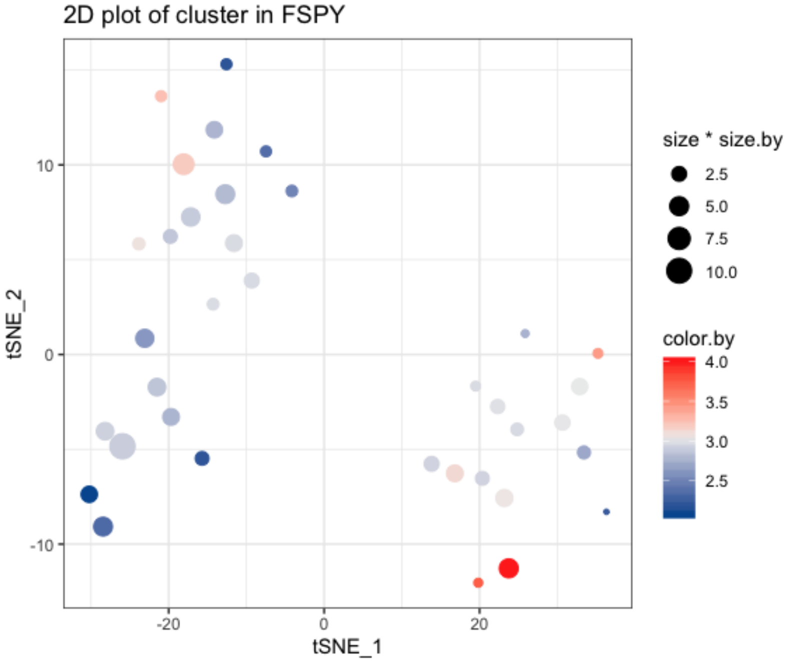
```
# Tree plot
plotTree(fspy, color.by = "D0.percent", show.node.name = TRUE, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



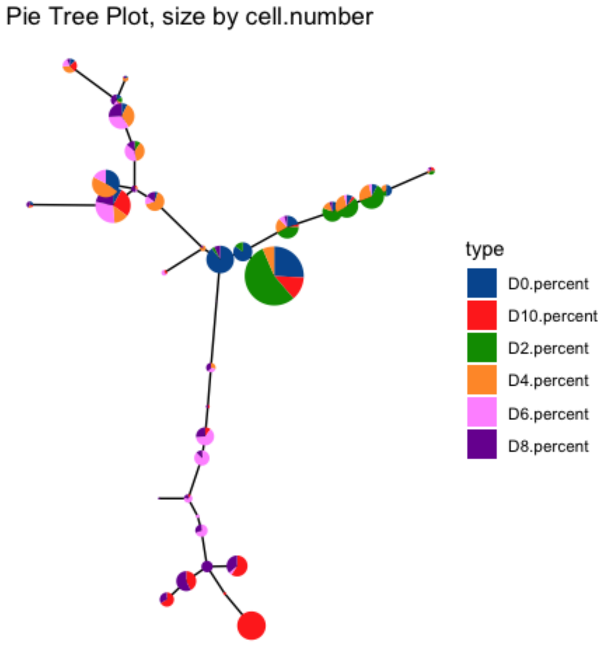
```
plotTree(fspy, color.by = "CD43", show.node.name = TRUE, cex.size = 1) +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



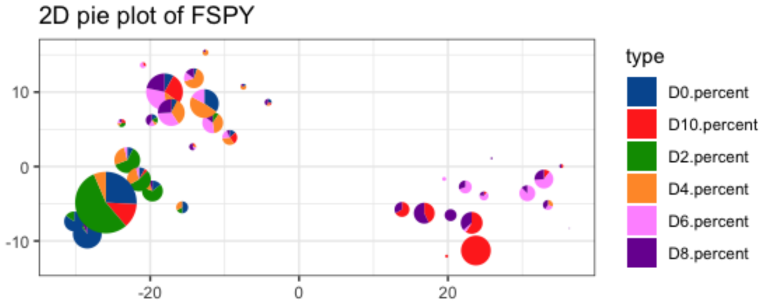
```
# plot clusters
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "numeric",
            size = 100, color.by = "CD45RA") +
  scale_colour_gradientn(colors = c("#00599F", "#EEEEEE", "#FF3222"))
```



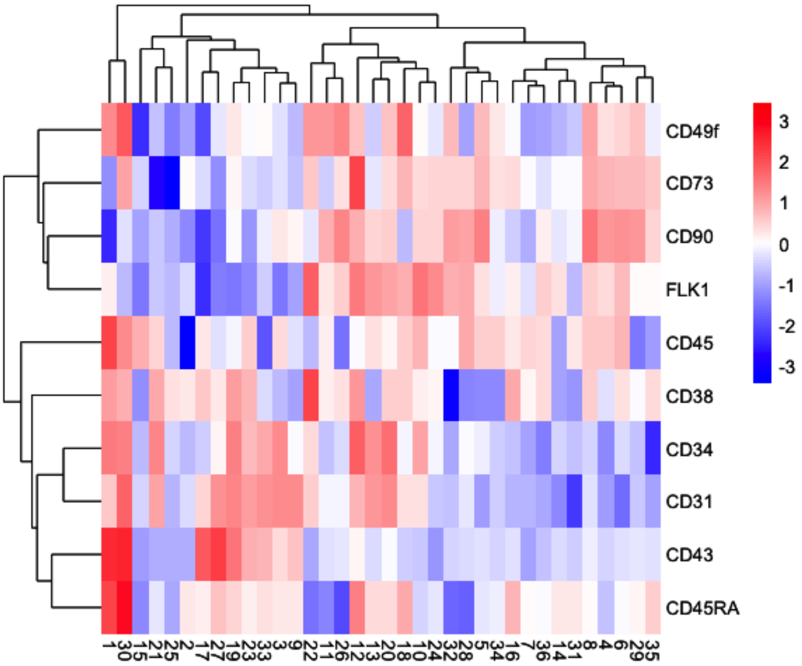
```
# plot pie tree
plotPieTree(fspy, cex.size = 3, size.by.cell.number = TRUE) +
  scale_fill_manual(values = c("#00599F","#FF3222","#009900",
                               "#FF9933","#FF99FF","#7A06A0"))
```



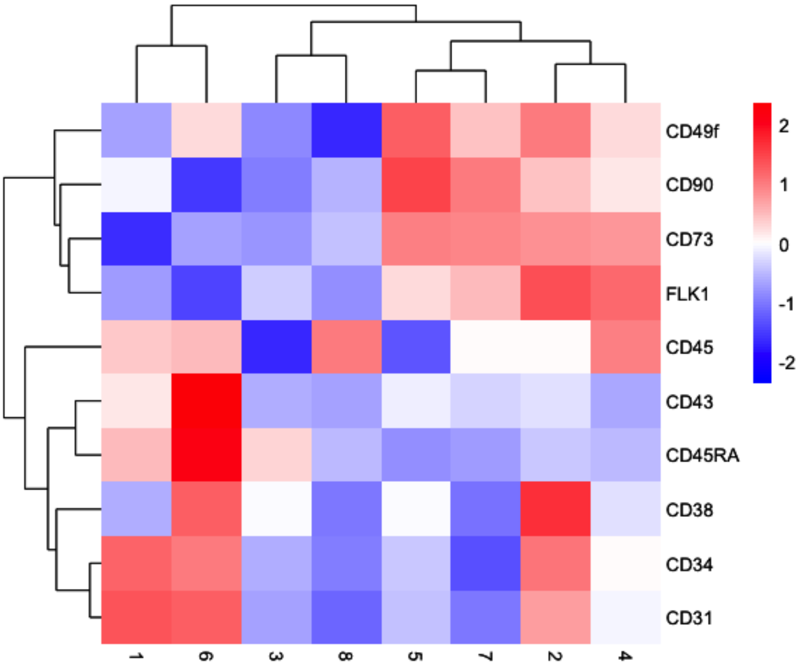
```
# plot pie cluster
plotPieCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), cex.size = 40) +
  scale_fill_manual(values = c("#00599F","#FF3222","#009900",
                               "#FF9933","#FF99FF","#7A06A0"))
```



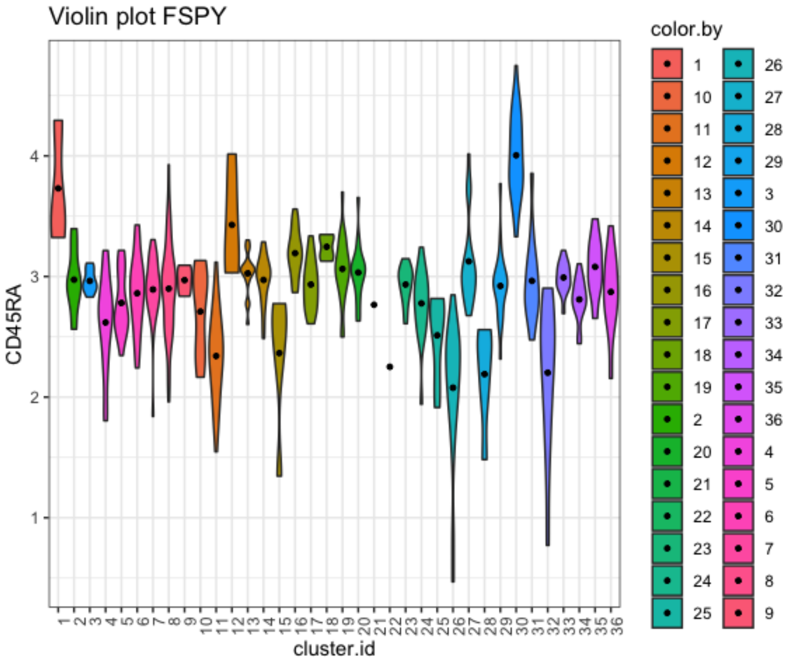
```
# plot heatmap of cluster
plotClusterHeatmap(fspy)
```



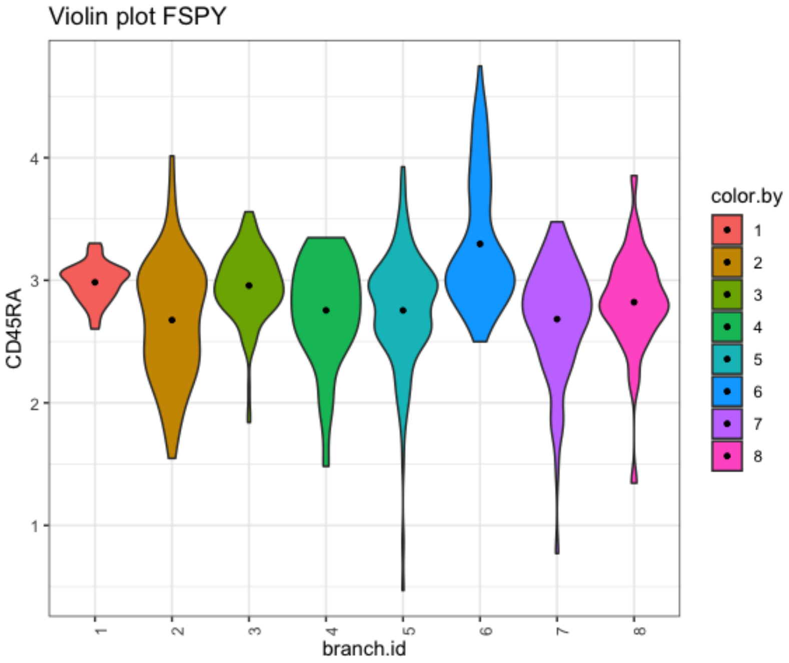
```
plotBranchHeatmap(fspy)
```



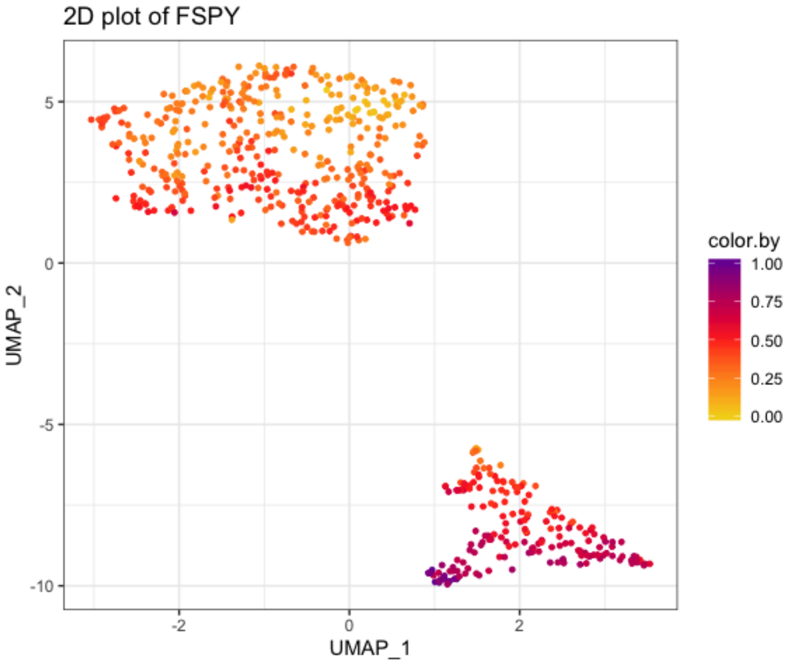
```
# Violin plot
plotViolin(fspy, color.by = "cluster.id", marker = "CD45RA", text.angle = 90)
```



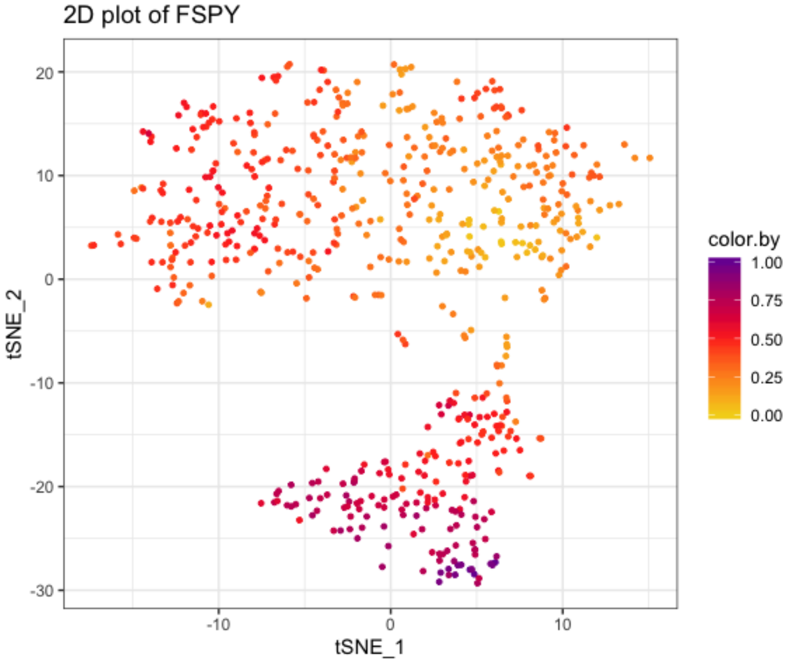
```
plotViolin(fspy, color.by = "branch.id", marker = "CD45RA", text.angle = 90)
```



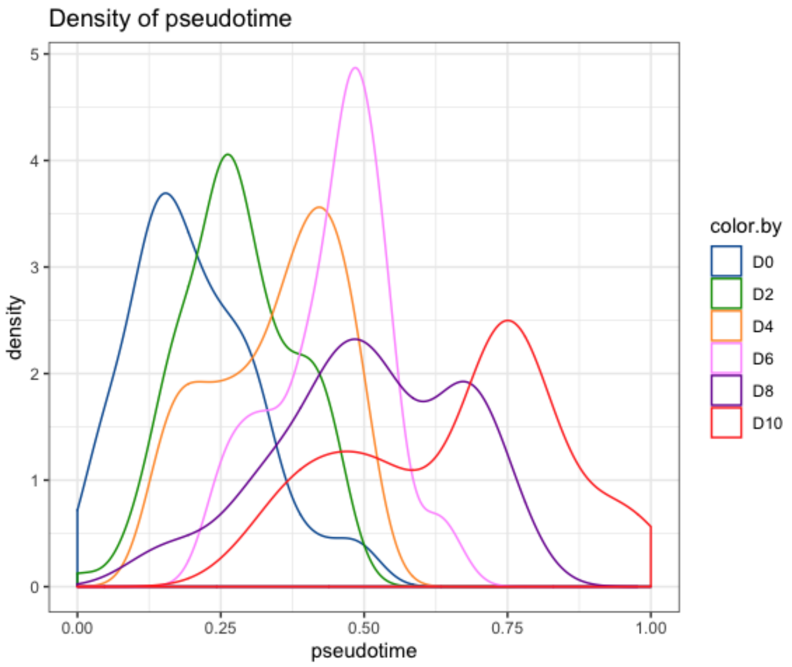
```
# UMAP plot colored by pseudotime
plot2D(fspy, item.use = c("UMAP_1", "UMAP_2"), category = "numeric",
            size = 1, color.by = "pseudotime") +
  scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```



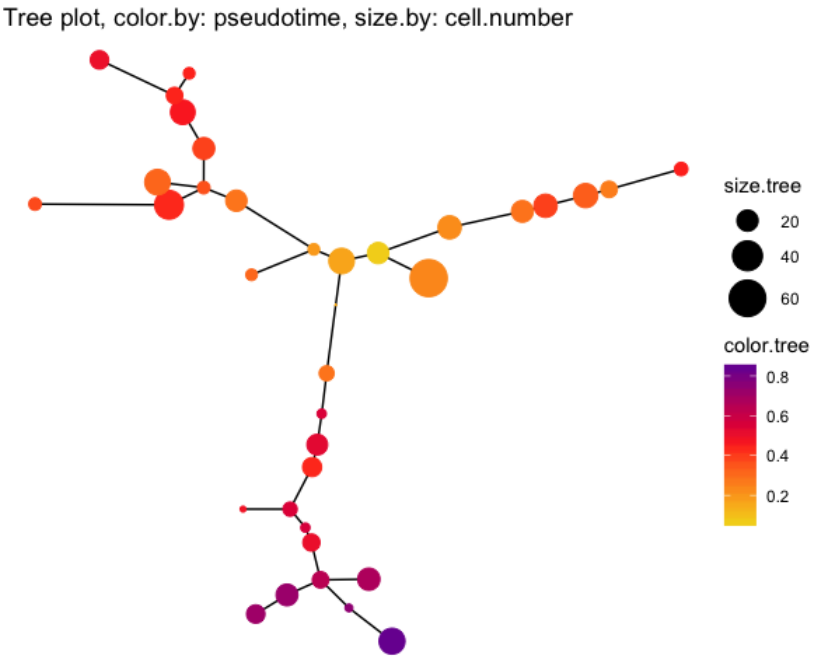
```r
# tSNE plot colored by pseudotime
plot2D(fspy, item.use = c("tSNE_1", "tSNE_2"), category = "numeric",
            size = 1, color.by = "pseudotime") +
 scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```



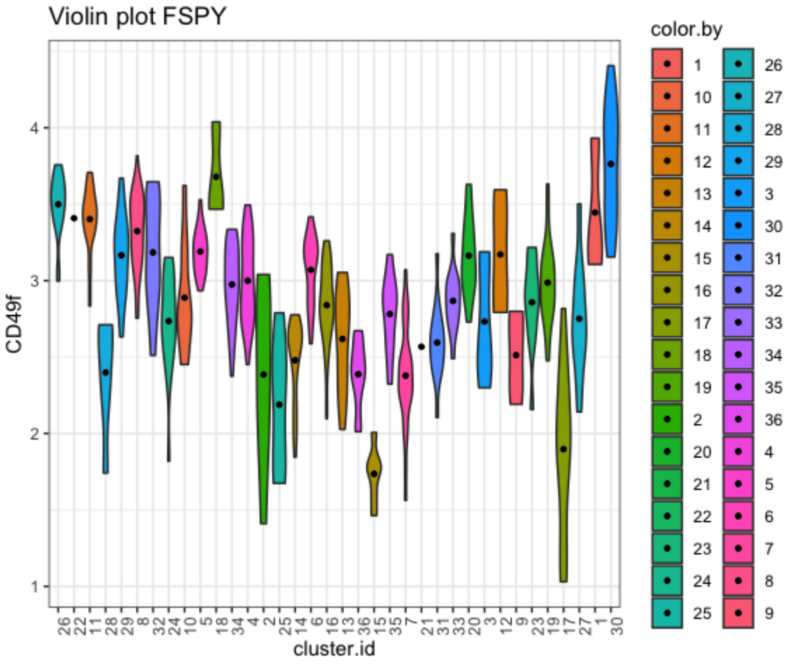
```r
# denisty plot by different stage
plotPseudotimeDensity(fspy, adjust = 1) +
  scale_color_manual(values = c("#00599F","#009900","#FF9933",
                                "#FF99FF","#7A06A0","#FF3222"))
```



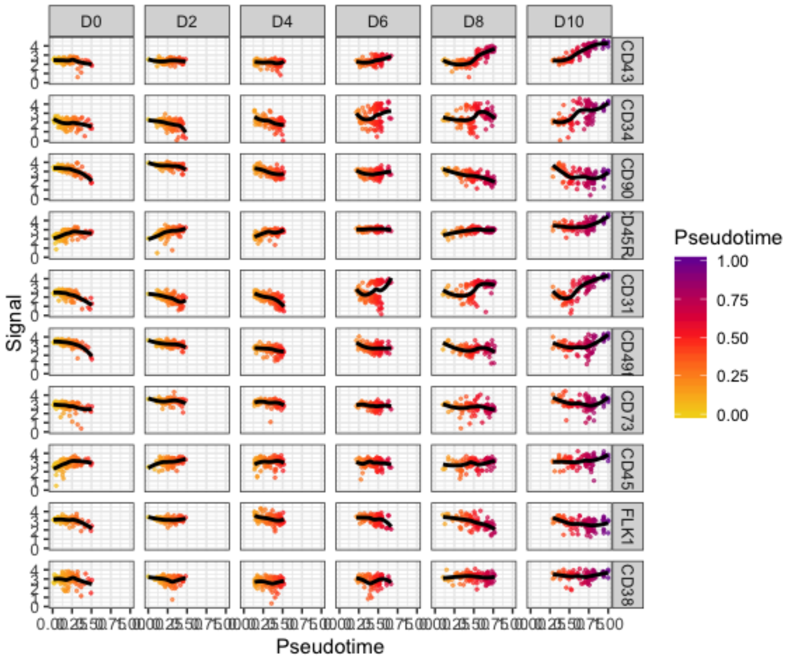
```
# Tree plot
plotTree(fspy, color.by = "pseudotime", cex.size = 1.5) +
 scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```



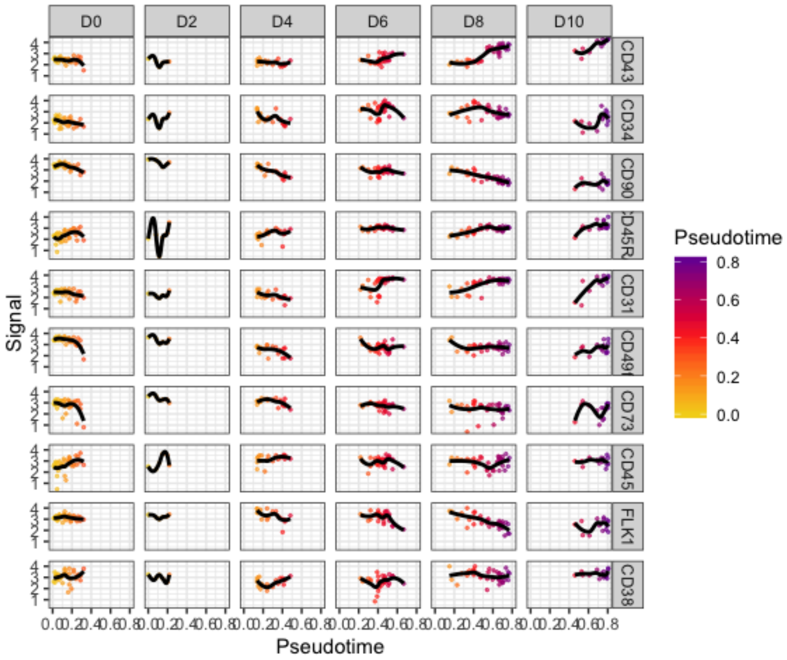
```
plotViolin(fspy, color.by = "cluster.id", order.by = "pseudotime",
           marker = "CD49f", text.angle = 90)
```



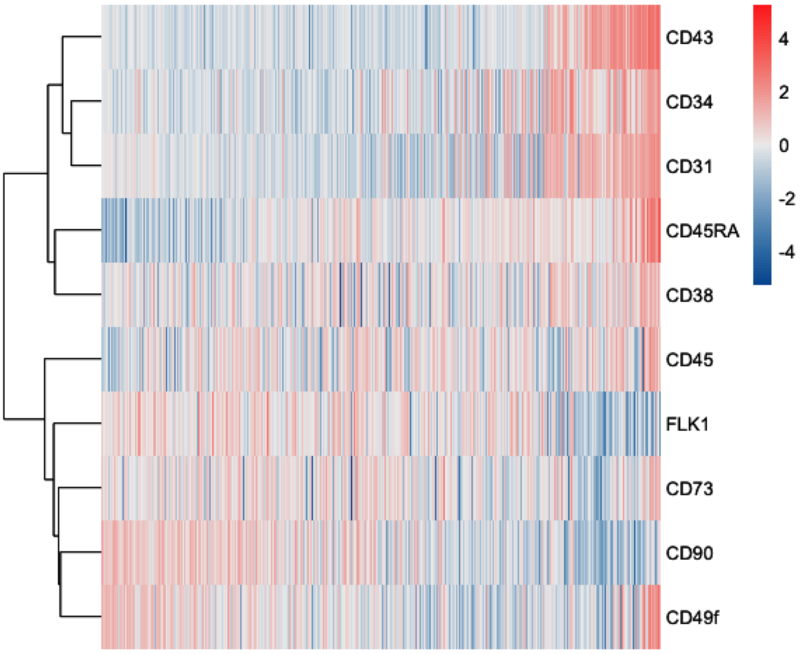
```
# trajectory value
plotPseudotimeTraj(fspy, var.cols = TRUE) +
 scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```



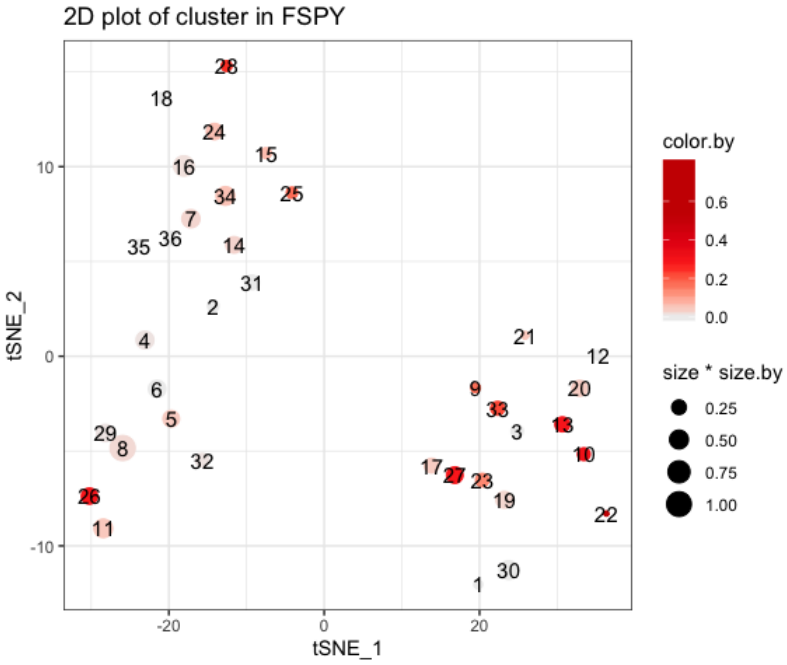
```
plotPseudotimeTraj(fspy, cutoff = 0.05, var.cols = TRUE) +
 scale_colour_gradientn(colors = c("#F4D31D", "#FF3222","#7A06A0"))
```



```r
plotHeatmap(fspy, downsize = 1000, cluster_rows = TRUE, clustering_method = "ward.D",
            color = colorRampPalette(c("#00599F","#EEEEEE","#FF3222"))(100))
```



```r
# plot cluster
plotCluster(fspy, item.use = c("tSNE_1", "tSNE_2"), color.by = "traj.value.log",
            size = 10, show.cluser.id = TRUE, category = "numeric") +
 scale_colour_gradientn(colors = c("#EEEEEE", "#FF3222", "#CC0000", "#CC0000"))
```



```r
# Show 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   flowCore_1.50.0 flowSpy_1.2.7   igraph_1.2.4.1
## [5] ggplot2_3.2.1
##
## 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_0.55.0          gmodels_2.18.1
##  [23] xts_0.11-2                  askpass_1.1
##  [25] colorspace_1.4-1            blob_1.2.0
##  [27] rrcov_1.4-7                 haven_2.1.1
##  [29] xfun_0.9                    dplyr_0.8.3
##  [31] crayon_1.3.4                RCurl_1.95-4.12
##  [33] jsonlite_1.6                graph_1.62.0
##  [35] scatterpie_0.1.2            genefilter_1.66.0
##  [37] zeallot_0.1.0               survival_2.44-1.1
##  [39] zoo_1.8-6                   glue_1.3.1
##  [41] polyclip_1.10-0             gtable_0.3.0
##  [43] zlibbioc_1.30.0             XVector_0.24.0
##  [45] DelayedArray_0.10.0         car_3.0-3
##  [47] BiocGenerics_0.30.0         DEoptimR_1.0-8
##  [49] abind_1.4-5                 VIM_4.8.0
##  [51] scales_1.0.0                pheatmap_1.0.12
##  [53] mvtnorm_1.0-11              DBI_1.0.0
##  [55] ggthemes_4.2.0              Rcpp_1.0.2
##  [57] xtable_1.8-4                laeken_0.5.0
##  [59] reticulate_1.13             foreign_0.8-72
##  [61] bit_1.1-14                  proxy_0.4-23
##  [63] mclust_5.4.5                FlowSOM_1.16.0
##  [65] stats4_3.6.1                tsne_0.1-3
##  [67] umap_0.2.3.1                vcd_1.4-4
##  [69] RColorBrewer_1.1-2          pkgconfig_2.0.2
##  [71] XML_3.98-1.20               farver_1.1.0
##  [73] nnet_7.3-12                 reshape2_1.4.3
##  [75] labeling_0.3                tidyselect_0.2.5
##  [77] rlang_0.4.0                 AnnotationDbi_1.46.1
##  [79] munsell_0.5.0               cellranger_1.1.0
##  [81] tools_3.6.1                 RSQLite_2.1.2
##  [83] ranger_0.11.2               evaluate_0.14
##  [85] yaml_2.2.0                  knitr_1.24
##  [87] bit64_0.9-7                 zip_2.0.4
##  [89] robustbase_0.93-5           purrr_0.3.2
##  [91] RANN_2.6.1                  nlme_3.1-141
##  [93] compiler_3.6.1              curl_4.0
##  [95] e1071_1.7-2                 smoother_1.1
##  [97] tibble_2.1.3                tweenr_1.0.1
##  [99] pcaPP_1.9-73                stringi_1.4.3
## [101] RSpectra_0.15-0             forcats_0.4.0
## [103] lattice_0.20-38             Matrix_1.2-17
## [105] vctrs_0.2.0                 pillar_1.4.2
## [107] RUnit_0.4.32                lmtest_0.9-37
## [109] BiocNeighbors_1.2.0         data.table_1.12.2
## [111] bitops_1.0-6                corpcor_1.6.9
## [113] GenomicRanges_1.36.1        R6_2.4.0
## [115] rio_0.5.16                  IRanges_2.18.2
## [117] flowUtils_1.48.0            boot_1.3-23
## [119] MASS_7.3-51.4               gtools_3.8.1
## [121] assertthat_0.2.1            destiny_2.14.0
## [123] SummarizedExperiment_1.14.1 openssl_1.4.1
## [125] withr_2.1.2                 S4Vectors_0.22.1
## [127] GenomeInfoDbData_1.2.1      mgcv_1.8-28
## [129] parallel_3.6.1              hms_0.5.1
## [131] grid_3.6.1                  prettydoc_0.3.0
## [133] tidyr_0.8.3                 class_7.3-15
## [135] rmarkdown_1.15              rvcheck_0.1.3
## [137] carData_3.0-2               Rtsne_0.15
## [139] TTR_0.23-4                  ggforce_0.3.1
## [141] scatterplot3d_0.3-41        Biobase_2.44.0
```

References

1. Hahne F, Arlt D, Sauermann M, Majety M, Poustka A, Wiemann S, Huber W: Statistical methods and software for the analysis of highthroughput reverse genetic assays using flow cytometry readouts. Genome Biol 2006, 7:R77.
2. Olsen LR, Leipold MD, Pedersen CB, Maecker HT: The anatomy of single cell mass cytometry data. Cytometry A 2019, 95:156-172.
3. Butler A, Hoffman P, Smibert P, Papalexi E, Satija R: Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol 2018, 36:411-420.

4. Trapnell C, Cacchiarelli D, Grimsby J, Pokharel P, Li S, Morse M, Lennon NJ, Livak KJ, Mikkelsen TS, Rinn JL: The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nat Biotechnol 2014, 32:381-386.
5. Kiselev VY, Yiu A, Hemberg M: scmap: projection of single-cell RNA-seq data across data sets. Nat Methods 2018, 15:359-362.
6. Amir el AD, Davis KL, Tadmor MD, Simonds EF, Levine JH, Bendall SC, Shenfeld DK, Krishnaswamy S, Nolan GP, Pe'er D: viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. Nat Biotechnol 2013, 31:545-552.
7. Haghverdi L, Buettner F, Theis FJ: Diffusion maps for high-dimensional single-cell analysis of differentiation data. Bioinformatics 2015, 31:2989-2998.
8. Becht E, McInnes L, Healy J, Dutertre CA, Kwok IWH, Ng LG, Ginhoux F, Newell EW: Dimensionality reduction for visualizing single-cell data using UMAP. Nat Biotechnol 2018.
9. Wang L, Hoffman RA: Standardization, Calibration, and Control in Flow Cytometry. Curr Protoc Cytom 2017, 79:1 3 1-1 3 27.
10. Hahne F, LeMeur N, Brinkman RR, Ellis B, Haaland P, Sarkar D, Spidlen J, Strain E, Gentleman R: flowCore: a Bioconductor package for high throughput flow cytometry. BMC Bioinformatics 2009, 10:106.
11. Sarkar D, Le Meur N, Gentleman R: Using flowViz to visualize flow cytometry data. Bioinformatics 2008, 24:878-879.
12. Van Gassen S, Callebaut B, Van Helden MJ, Lambrecht BN, Demeester P, Dhaene T, Saeys Y: FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data. Cytometry A 2015, 87:636-645.
13. Qiu P, Simonds EF, Bendall SC, Gibbs KD, Jr., Bruggner RV, Linderman MD, Sachs K, Nolan GP, Plevritis SK: Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE. Nat Biotechnol 2011, 29:886-891.
14. Chen H, Lau MC, Wong MT, Newell EW, Poidinger M, Chen J: Cytofkit: A Bioconductor Package for an Integrated Mass Cytometry Data Analysis Pipeline. PLoS Comput Biol 2016, 12:e1005112.
15. Chattopadhyay PK, Winters AF, Lomas WE, 3rd, Laino AS, Woods DM: High-Parameter Single-Cell Analysis. Annu Rev Anal Chem (Palo Alto Calif) 2019, 12:411-430.
16. Bendall SC, Davis KL, Amir el AD, Tadmor MD, Simonds EF, Chen TJ, Shenfeld DK, Nolan GP, Pe'er D: Single-cell trajectory detection uncovers progression and regulatory coordination in human B cell development. Cell 2014, 157:714-725.
17. Nowicka M, Krieg C, Crowell HL, Weber LM, Hartmann FJ, Guglietta S, Becher B, Levesque MP, Robinson MD: CyTOF workflow: differential discovery in high-throughput high-dimensional cytometry datasets. F1000Res 2017, 6:748.