SA11b: Hydra URD Interstitial Lineage Trajectory Analysis

Jeff Farrell

October 1, 2018

```
# Load required libraries
library(URD)
```

```
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.4.4
## Loading required package: Matrix
## Warning: package 'Matrix' was built under R version 3.4.2
```

```
# Set main
opts_chunk$set(root.dir = "~/Dropbox/HydraURDSubmission/")
main.path <- "~/Dropbox/HydraURDSubmission/"

# Build output directory
dir.create(paste0(main.path, "URD/IC/"), recursive = T)
```

```
## Warning in dir.create(paste0(main.path, "URD/IC/"), recursive = T): '/
## Users/jaf2030/Dropbox/HydraURDSubmission/URD/IC' already exists
```

Load subsetted URD data for this lineage

```
# Load the IC lineage data (with doublets removed) output from SA11a.
hydra.ic <- readRDS(file = paste0(main.path, "objects/Hydra_URD_Input_IC.rds"))
```

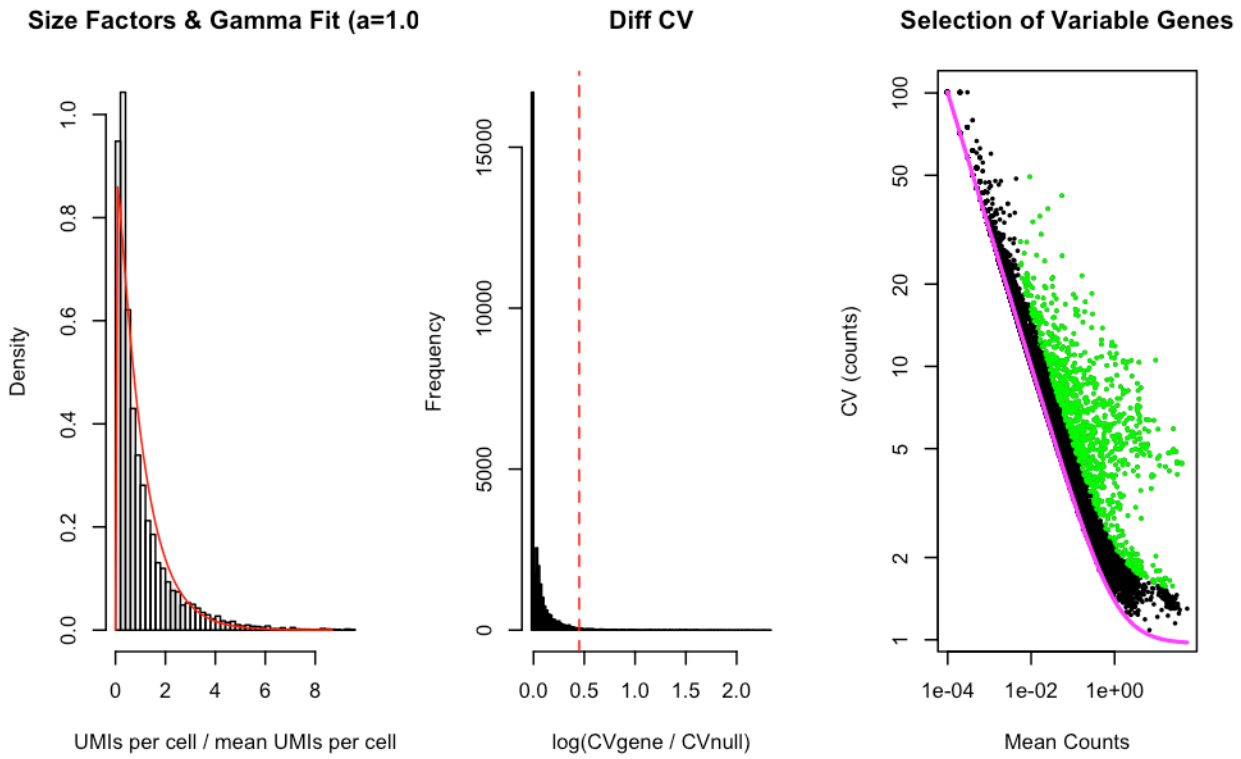
Variable genes

Calculate variable genes

We need to find the genes that are more likely to encode biological variability (rather than solely technical variability) for building the diffusion map.

```
# A more restrictive list of variable genes
new.var <- findVariableGenes(object = hydra.ic, set.object.var.genes = F, diffCV.cutoff = 0.45,
    do.plot = T)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 2305 x values <= 0 omitted
## from logarithmic plot
```



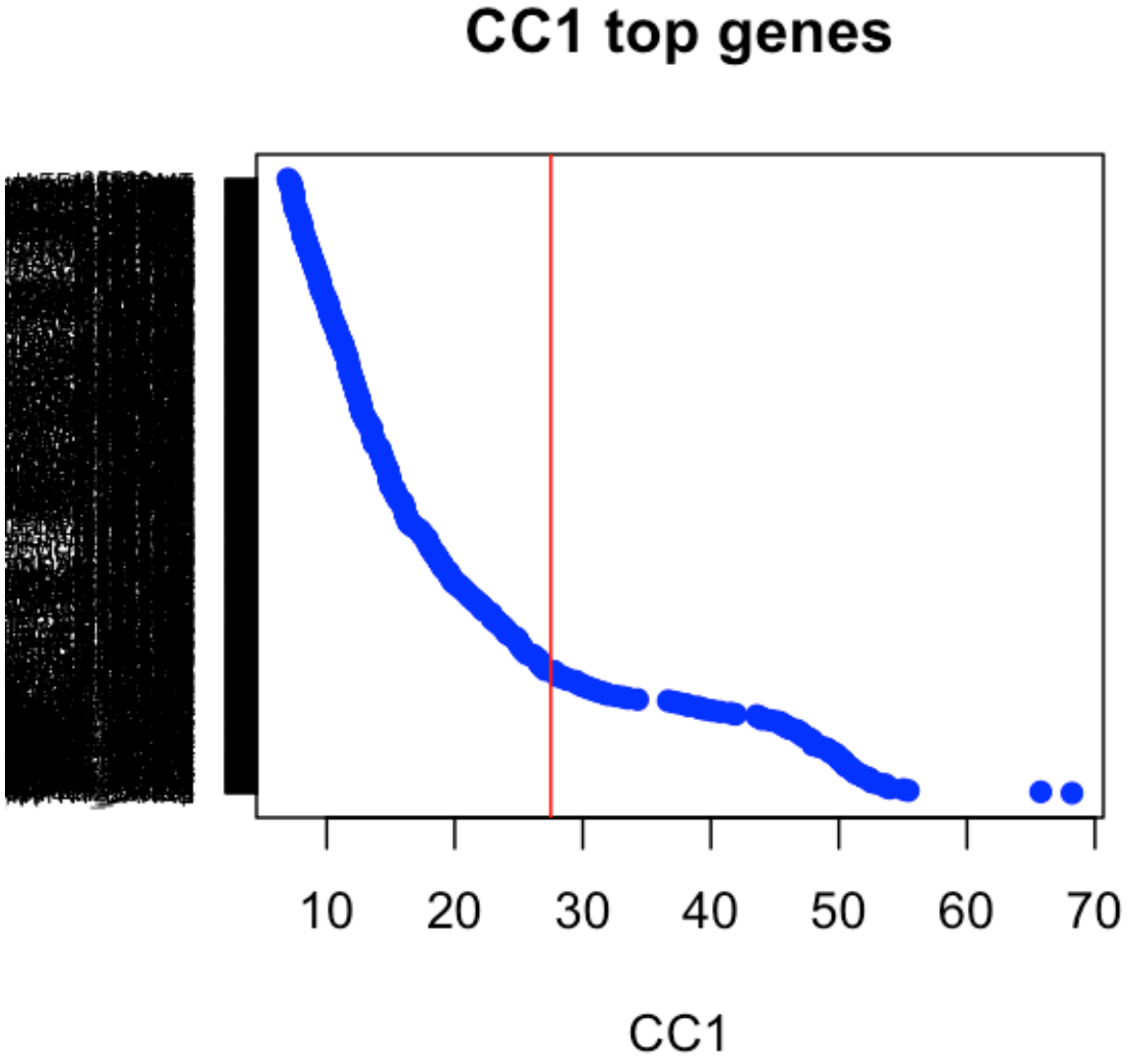
Removal of FACS batch genes

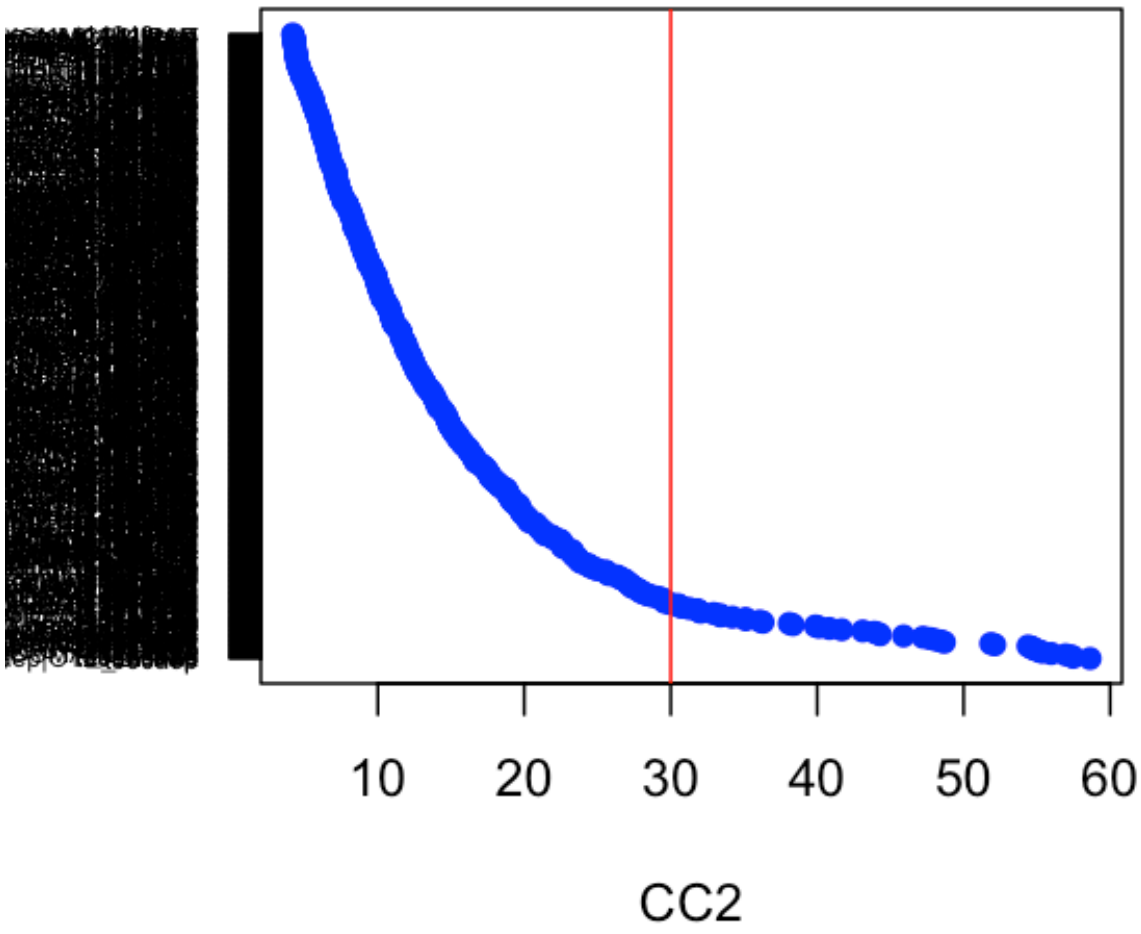
In order to get good representation of the neuronal cells in the interstitial lineage, some runs were enriched for neurons by FACS-sorting potential neuronal cells using a transgenic line that marks them fluorescently. However, FACS sorting introduced a fairly strong batch effect into the data. Since this batch effect is focused on a particular group of genes, one way to deal with this for trajectory reconstruction is to remove those genes that are strongly associated with the batch effect from the variable gene list; then they will not be considered when determining the distance between cells and will not affect the trajectory reconstruction.

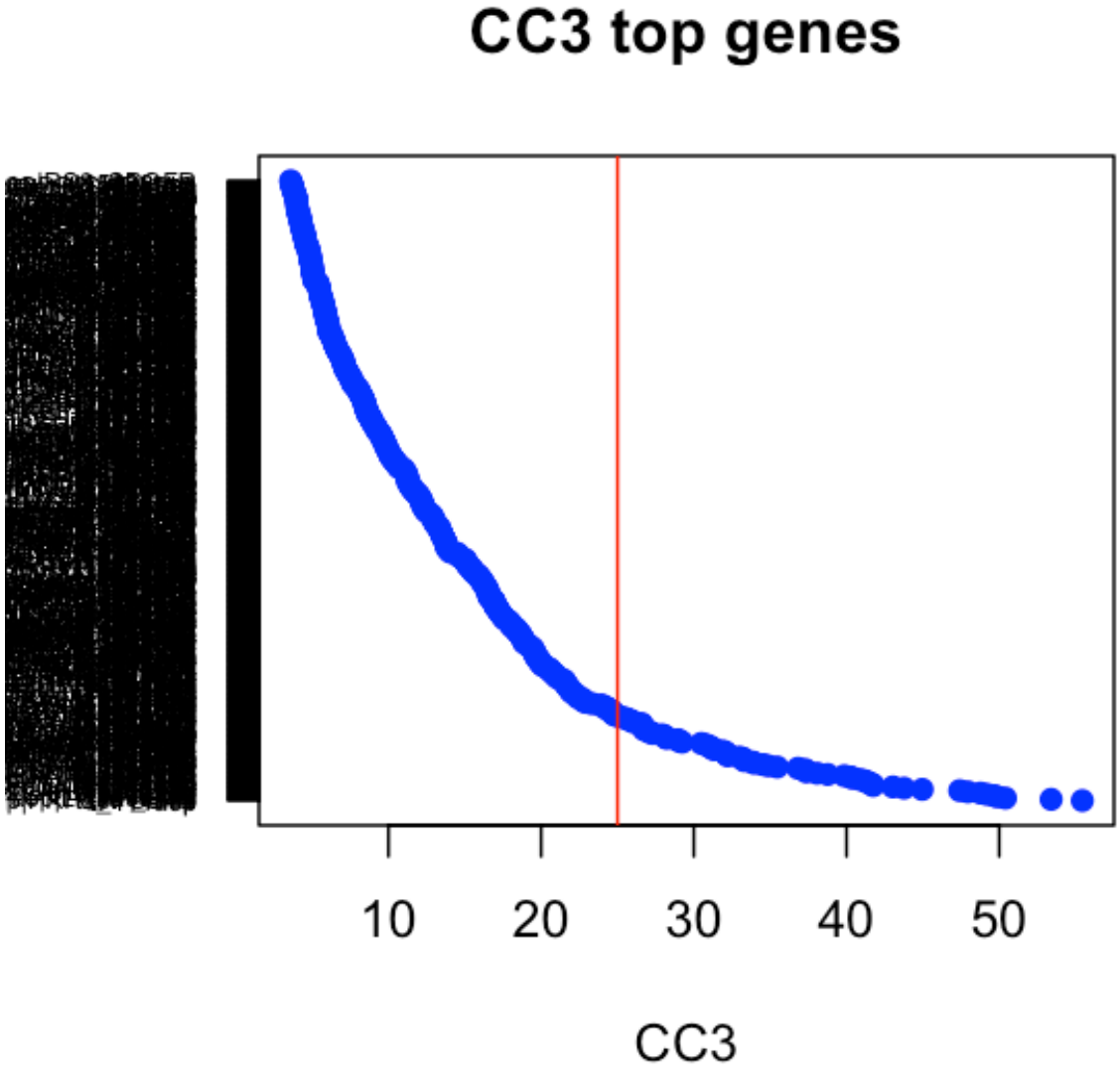
Since Stefan successfully integrated these batches using Seurat's alignment approach based on canonical correlation analysis (CCA), we determined those genes that were strongly loaded into the top CCAs and removed those as FACS-induced batch effect genes.

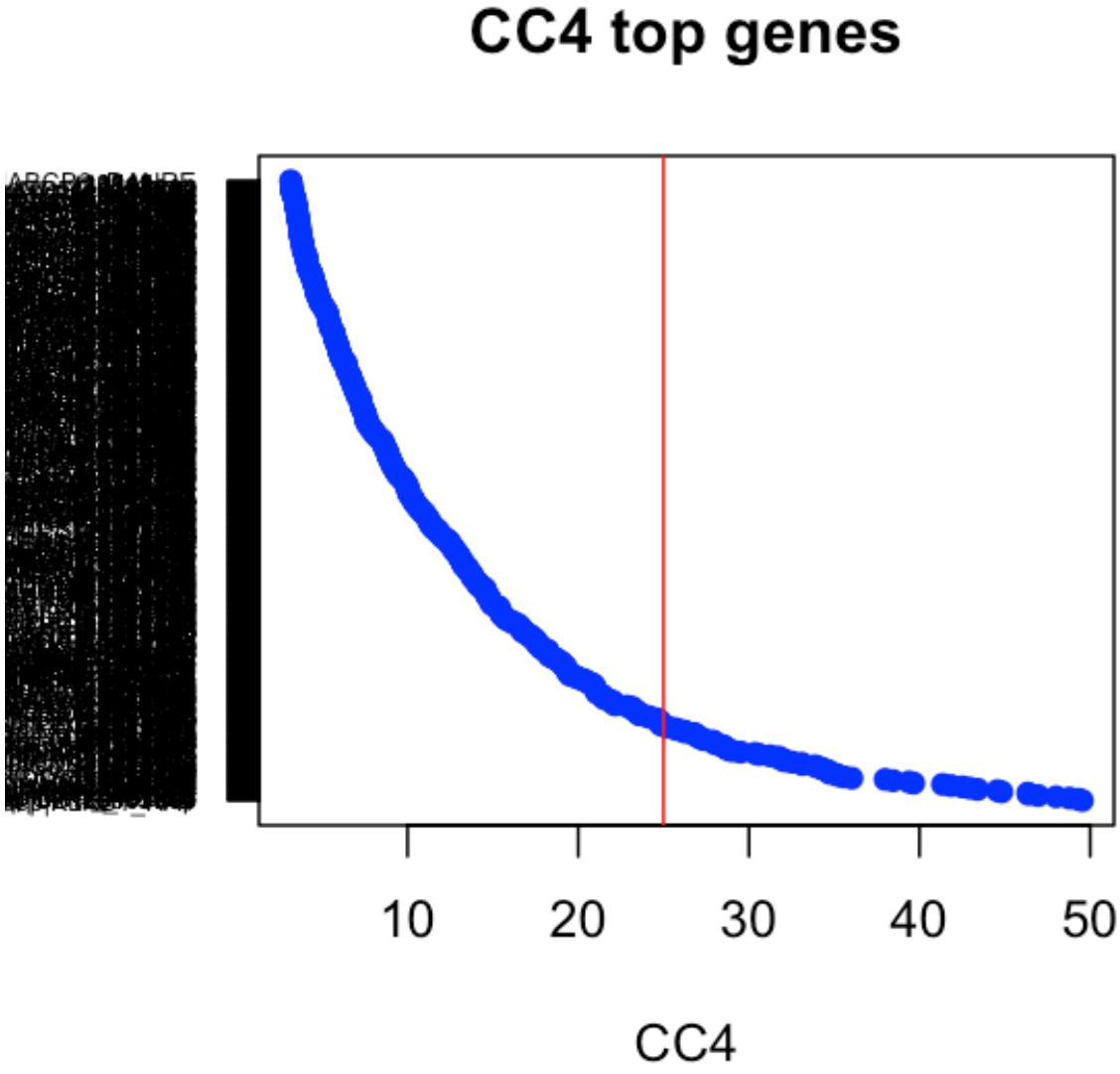
```
# Load the CCA-aligned Seurat object
hydra.seurat.nc.cca <- readRDS(paste0(main.path, "objects/Hydra_Seurat_Neurons.rds"))

# VizDimReductionAbs - Visualizes the loading of genes in their reduction, but as
# absolute value, so you can consider genes with the strongest contribution,
# regardless of their direction.  This function is BARELY modified from Seurat
# VizDimReduction
VizDimReductionAbs <- function(object, reduction.type = "cca", dims.use = 1:5, num.genes = 30,
    use.full = FALSE, font.size = 0.5, nCol = NULL, do.balanced = FALSE) {
    if (use.full) {
        dim.scores <- Seurat::GetDimReduction(object = object, reduction.type = reduction.type,
            slot = "gene.loadings.full")
    } else {
        dim.scores <- Seurat::GetDimReduction(object = object, reduction.type = reduction.type,
            slot = "gene.loadings")
    }
    if (is.null(x = nCol)) {
        if (length(x = dims.use) > 6) {
            nCol <- 3
        } else if (length(x = dims.use) > 9) {
            nCol <- 4
```

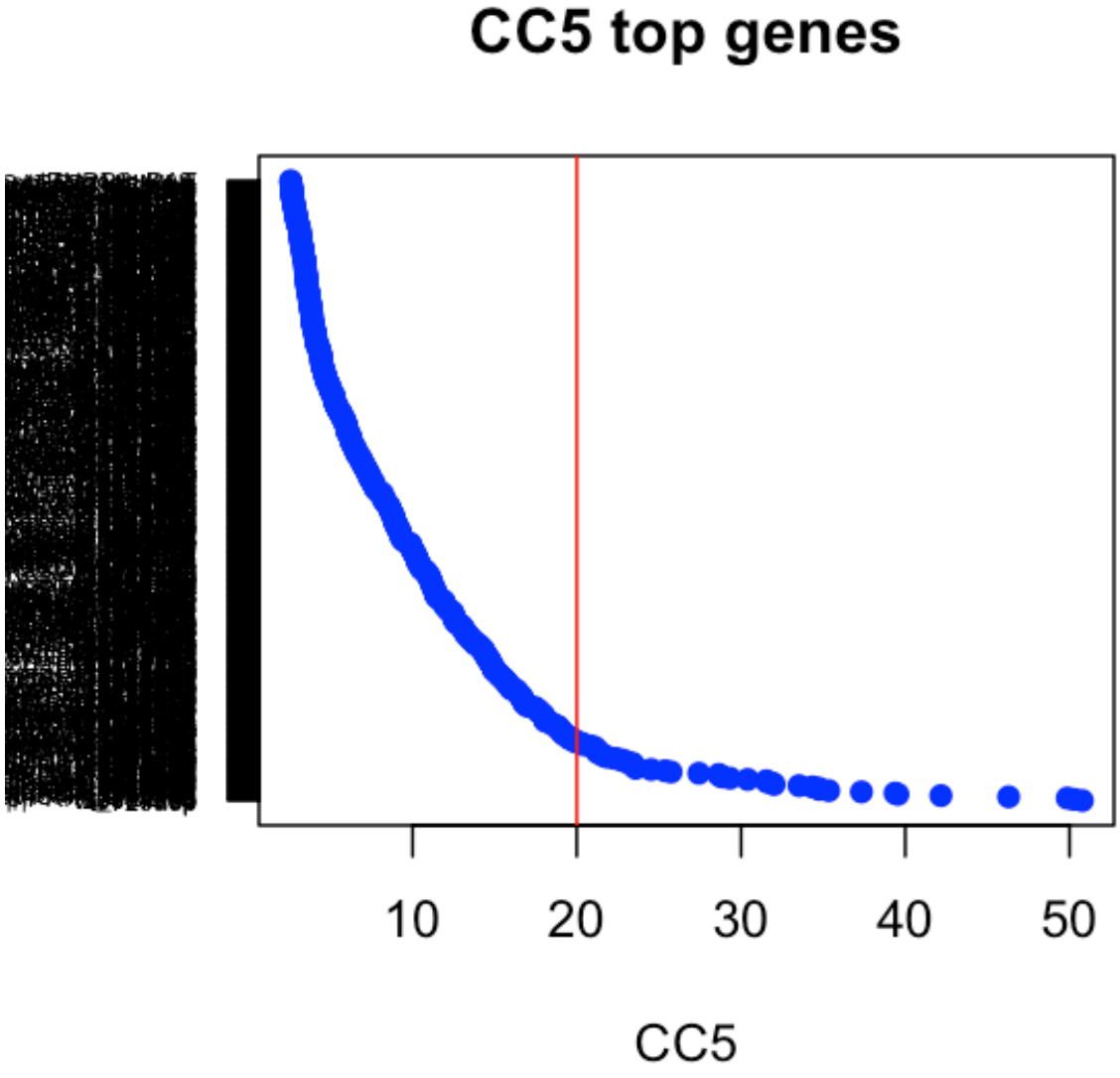
```
        } else {
            nCol <- 2
        }
    }
    num.row <- floor(x = length(x = dims.use)/nCol - 1e-05) + 1
    par(mfrow = c(num.row, nCol))
    for (i in dims.use) {
        subset.use <- dim.scores[Seurat::DimTopGenes(object = object, dim.use = i,
            reduction.type = reduction.type, num.genes = num.genes, use.full = use.full,
            do.balanced = do.balanced), ]
        subset.use <- abs(subset.use)
        subset.use <- subset.use[order(subset.use[, i], decreasing = T), ]
        plot(x = subset.use[, i], y = 1:nrow(x = subset.use), pch = 16, col = "blue",
            xlab = paste0("CC", i), yaxt = "n", ylab = "")
        axis(side = 2, at = 1:nrow(x = subset.use), labels = rownames(x = subset.use),
            las = 1, cex.axis = font.size)
    }
}

# Take the most strongly loaded genes in the top 6 CCs Here, plot and choose the
# elbow in the plot to identify the strongly loaded ones.
cca.thresh <- c(27.5, 30, 25, 25, 20, 25)
for (i in 1:length(cca.thresh)) {
    VizDimReductionAbs(hydra.seurat.nc.cca, reduction.type = "cca", dims.use = i,
        num.genes = 500, nCol = 1)
    title(paste0("CC", i, " top genes"))
    abline(v = cca.thresh[i], col = "red")
}
```

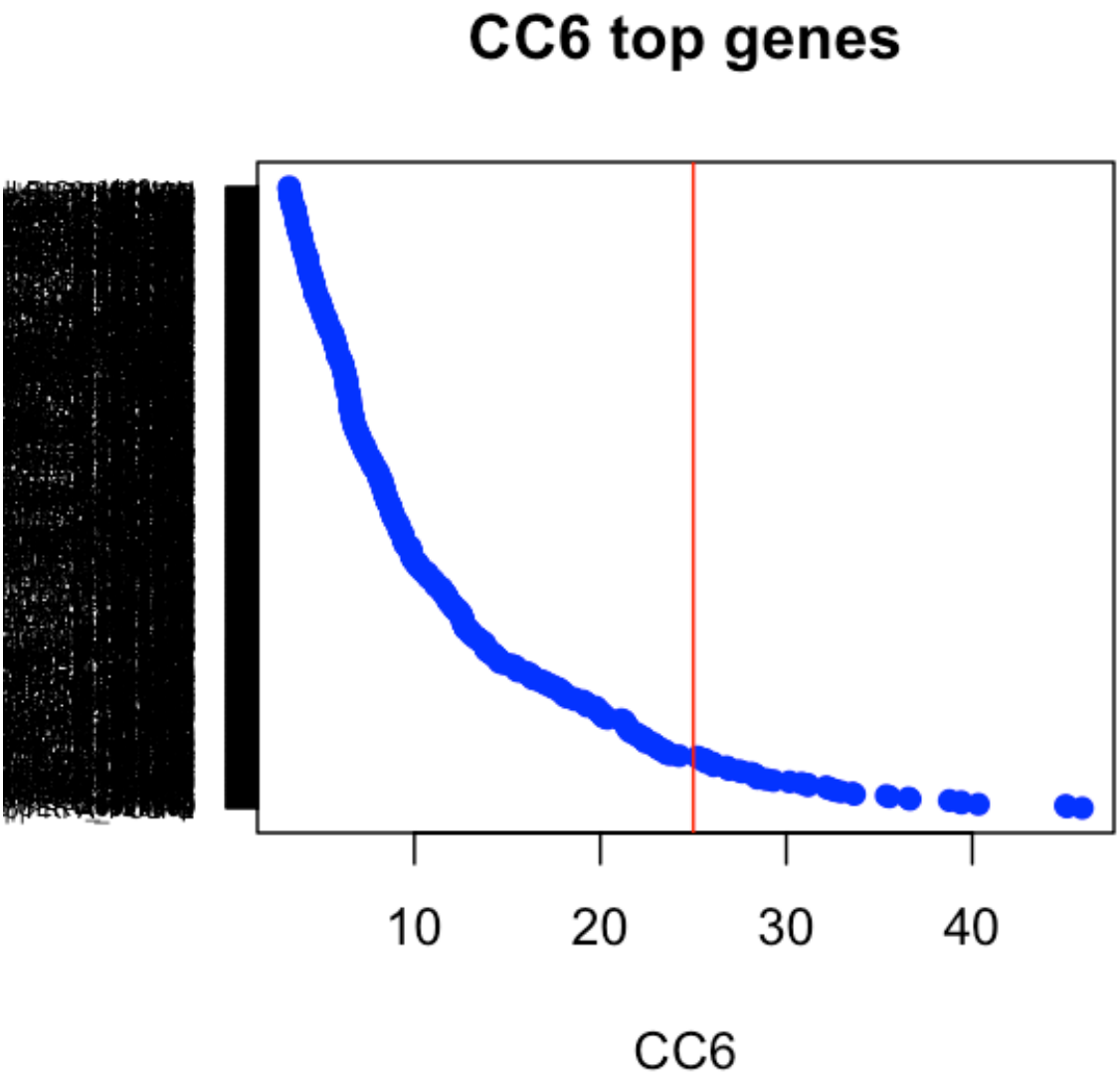

CC1 top genes
10
20
30
40
50
60
70
CC1


CC2 top genes
10
20
30
40
50
60
CC2


CC3 top genes
10
20
30
40
50
CC3


CC4 top genes
10
20
30
40
50
CC4


CC5 top genes
10
20
30
40
50
CC5




```
# Get those genes that pass the threshold
cca.genes <- lapply(1:length(cca.thresh), function(i) {
    ordered.loadings <- hydra.seurat.nc.cca@dr$cca@gene.loadings[order(abs(hydra.seurat.nc.cca@dr$cca@gene.loadings[,
        i]), decreasing = T), i]
    ordered.loadings <- ordered.loadings[abs(ordered.loadings) >= cca.thresh[i]]
    return(names(ordered.loadings))
})
cca.genes <- sort(unique(unlist(cca.genes)))

# Remove CCA genes from the variable genes and set then into the object
new.var.no.cca <- setdiff(new.var, unlist(cca.genes))
hydra.ic@var.genes <- new.var.no.cca
```

Graph Clustering

Generated several more fine-grained clusterings in order to have better options for choosing root and tip cells. In order to prevent errors from a non-deterministic clustering, we load our previous cluster identities

from a table. However, the commands previously run were:

```
# Graph clustering with various parameters
hydra.ic <- graphClustering(hydra.ic, num.nn = c(20, 30, 40, 50), do.jaccard = T,
    method = "Infomap")
hydra.ic <- graphClustering(hydra.ic, num.nn = c(5, 10, 15, 20, 30), do.jaccard = T,
    method = "Louvain")
hydra.ic <- graphClustering(hydra.ic, num.nn = c(6, 7, 8), do.jaccard = T, method = "Louvain")

clusters <- read.table("cluster_results/ic-clusters.txt", header = T, stringsAsFactors = F,
    colClasses = "character")
colnames(clusters) <- gsub("\\.", "-", colnames(clusters))
hydra.ic@group.ids <- cbind(hydra.ic@group.ids, clusters[rownames(hydra.ic@group.ids),
    ])
```

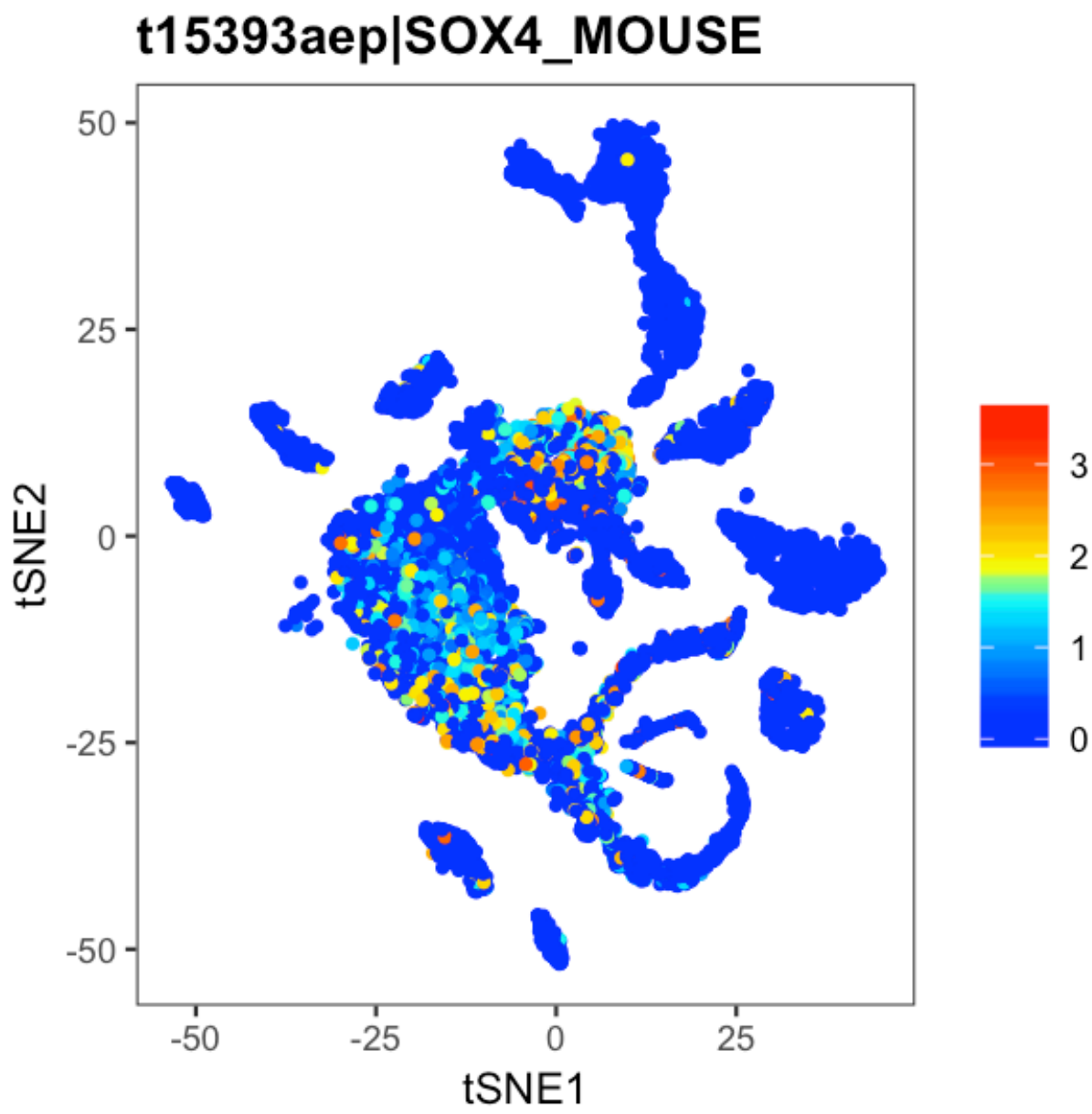
Choose Root and Tips

URD requires that the user define the starting points (the root) and the terminal points of the trajectories (the tips).

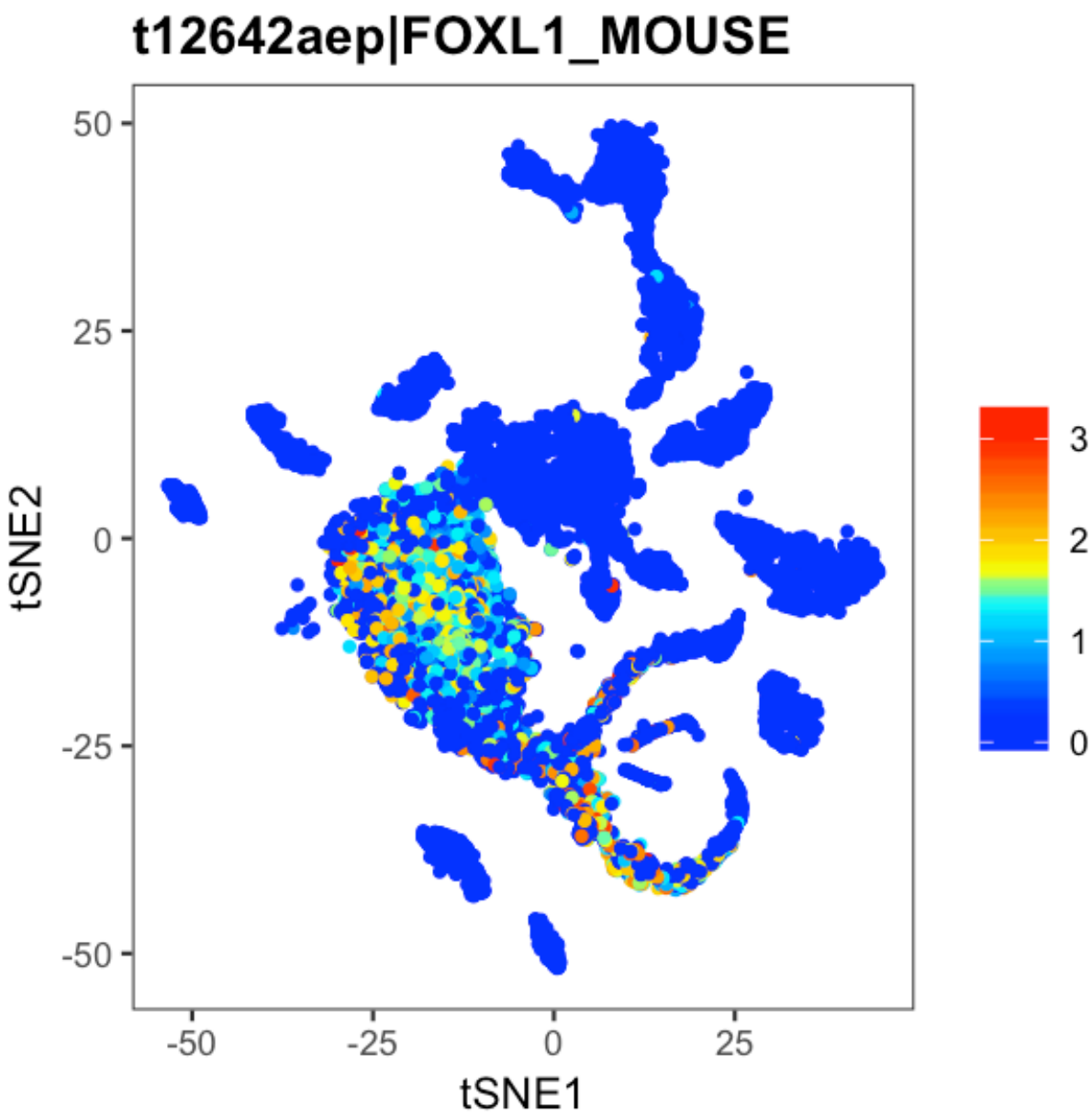
Root

While the molecular details of the interstital stem cells were previously unknown, we hypothesized that they have low HvSoxC (Swiss-Prot SOX4, t15393) and high FOXL1. This was based on the observation that HvSoxC might be a general differentiation marker, since it turns on in cells in the process of nematocyte, neuron, and gland, so those cells that do not express it are the likely stem cells. Thus, the root was chosen as several clusters of cells that were predominantly HvSoxC- FOXL1+.

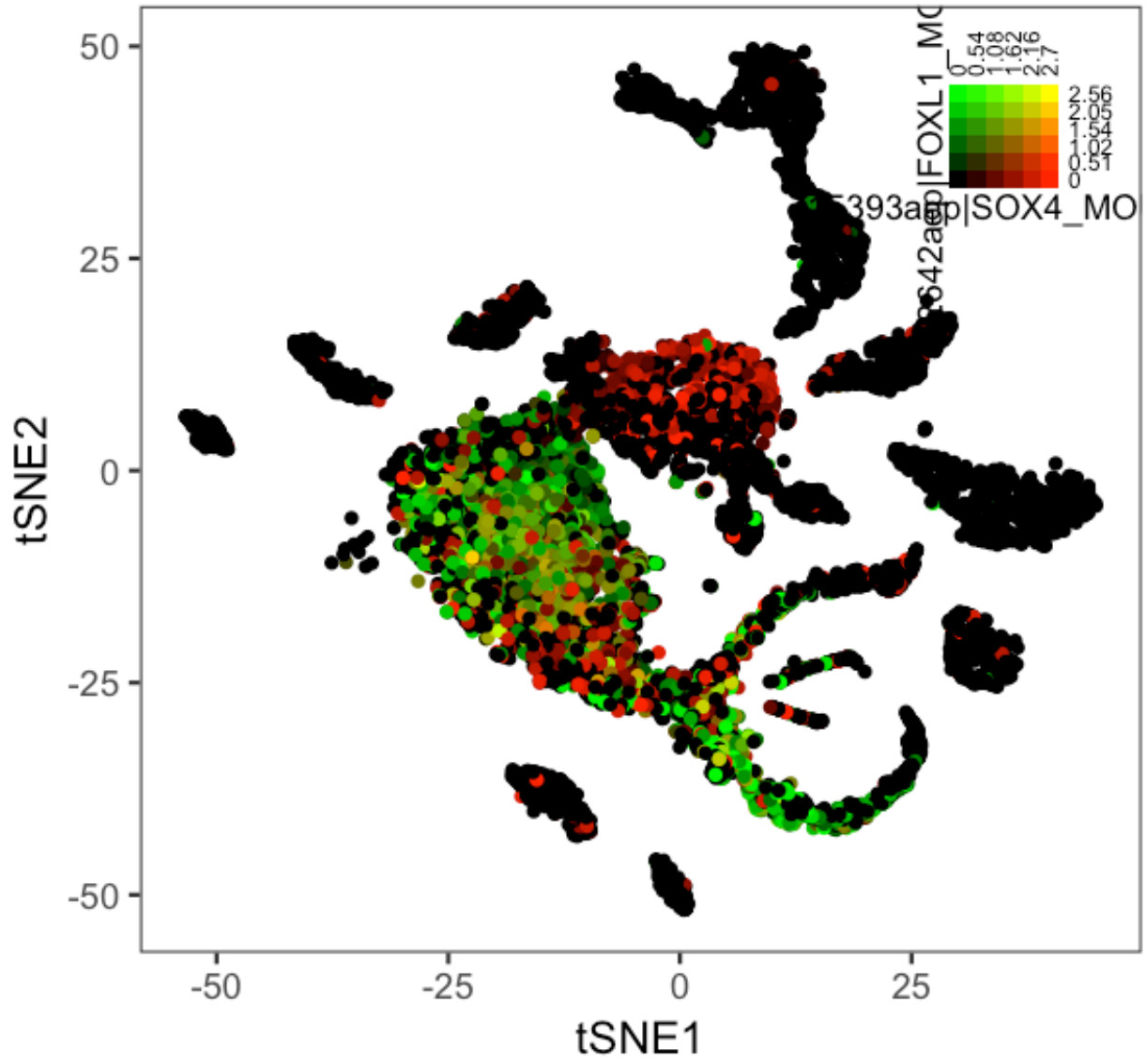
```
# Plot interstitial stem cell genes
plotDim(hydra.ic, "t15393aep|SOX4_MOUSE")
```



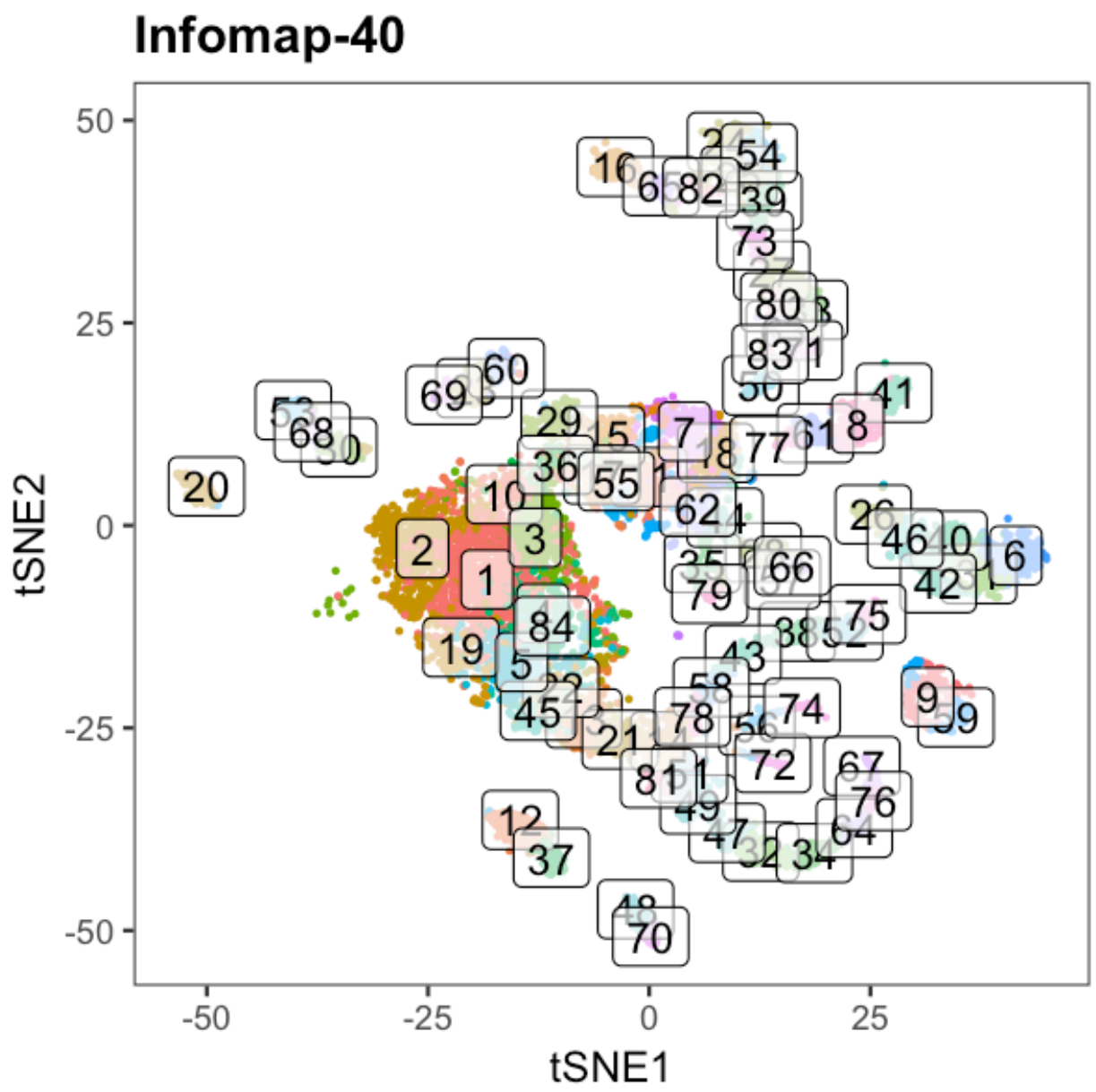

```
plotDim(hydra.ic, "t12642aep|FOXL1_MOUSE")
```



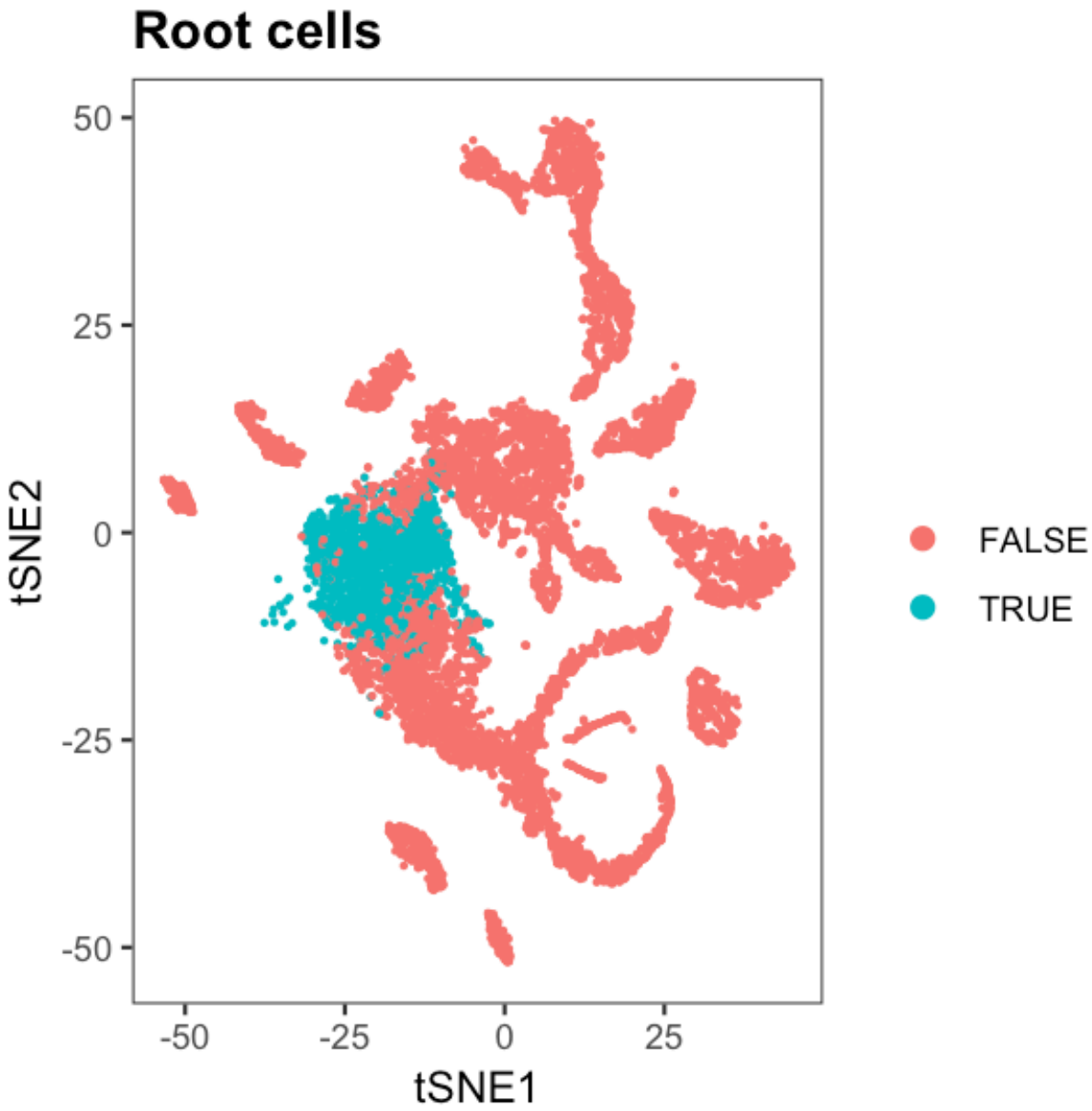

```
plotDimDual(hydra.ic, "t15393aep|SOX4_MOUSE", "t12642aep|FOXL1_MOUSE")
```




```
# Plot the clustering
plotDim(hydra.ic, "Infomap-40", label.clusters = T, legend = F)
```




```
# Define root cells
root.cells <- cellsInCluster(hydra.ic, "Infomap-40", c("1", "2", "3"))
hydra.ic <- groupFromCells(hydra.ic, "root", root.cells)

# Plot the root cells
plotDim(hydra.ic, "root", plot.title = "Root cells")
```



Tips

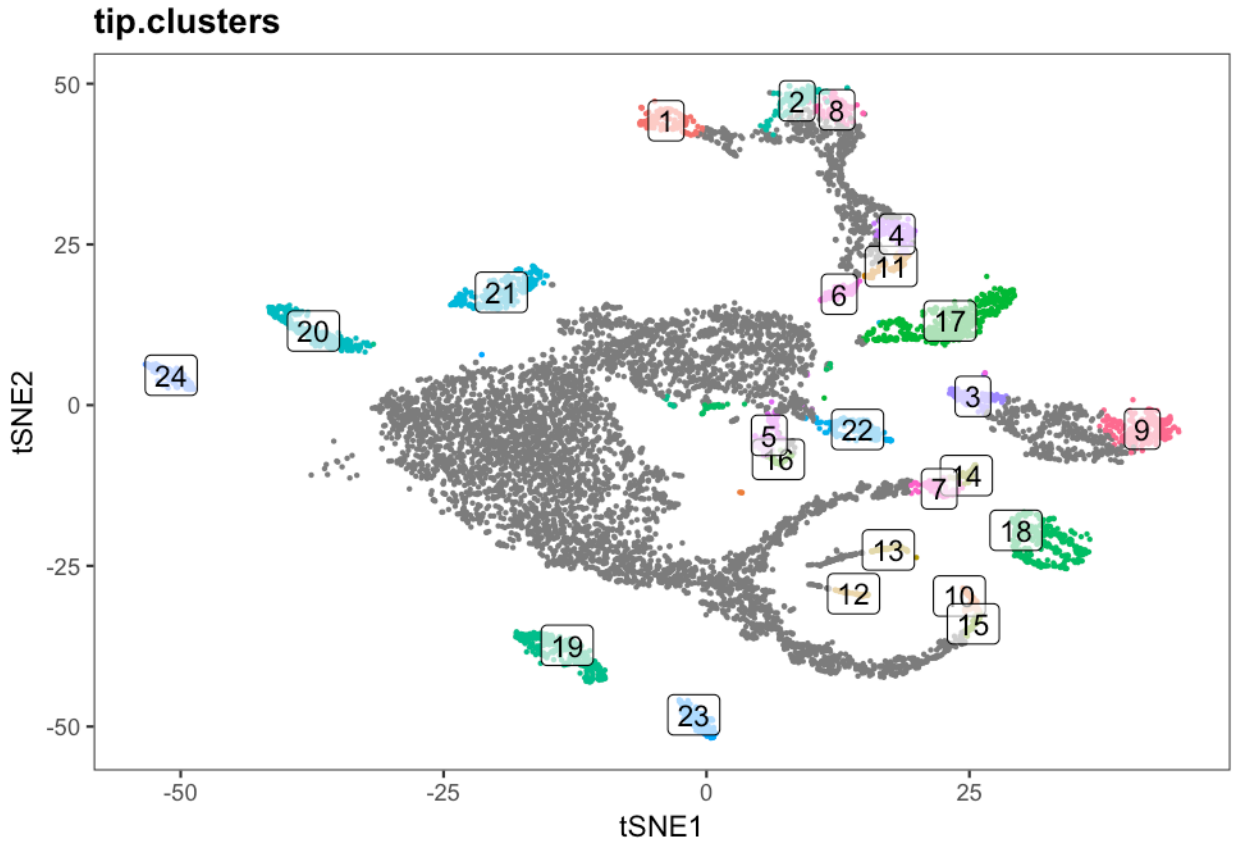
For tips, we performed random walks from the individual neuronal types identified in the Seurat clustering, and various clusters determined by the finer-grained graph clustering computed using URD. We performed walks from several clusters within the nematocyte and gland lineages so that we could ensure we chose tips that did a good job of visiting the rest of the data. (In some cases, a group of very differentiated cells can be poorly connected and random walks from those cells may not visit the rest of the cells well.)

```
# Create a new clustering for holding the tips you want to use
hydra.ic@group.ids$tip.clusters.raw <- NA

# Tips to try from Infomap-40 Choosing a couple of extra from the ends of some
# very differentiated populations just in case they end up being very poorly
# connected and not 100% sure where the pseudotime progression will go in the
# zymogen gland Nematocytes: 75, 52, 67, 76, 74, 72 Spumous: 26, 6 Zymogen: 16,
# 24, 54, 33, 71, 50 Neuronal: 79+35
infomap40.clusters.use <- c("75", "52", "67", "76", "74", "72", "26", "6", "16",
```

```
    "24", "54", "33", "71", "50", "79", "35")
# Copy selected Infomap-40 clusters over to the tip clustering
i40.tip.cells <- cellsInCluster(hydra.ic, "Infomap-40", infomap40.clusters.use)
hydra.ic@group.ids[i40.tip.cells, "tip.clusters.raw"] <- paste0("I40-", hydra.ic@group.ids[i40.tip.cells,
    "Infomap-40"])

# For neuronal clusters, mostly used clusters from 'clustering2' Neuronal: 17,
# 28, 16, 26, 25, 14, 12, 20 Replaced 21 with 81+38 from Infomap-40 because 21
# seems to extend into progenitor territory
res15.clusters.use <- c("17", "28", "16", "26", "25", "14", "12", "20")
# Copy selected 'clustering2' clusters over to the tip clustering
res15.tip.cells <- cellsInCluster(hydra.ic, "clustering2", res15.clusters.use)
hydra.ic@group.ids[res15.tip.cells, "tip.clusters.raw"] <- paste0("R15-", hydra.ic@group.ids[res15.tip.cells,
    "clustering2"])

# Renumber the clusters because there are overlapping cluster 'names' between the
# two clusterings.
hydra.ic@group.ids$tip.clusters <- as.factor(hydra.ic@group.ids$tip.clusters.raw)
levels(hydra.ic@group.ids$tip.clusters) <- as.character(seq_along(levels(hydra.ic@group.ids$tip.clusters)))
hydra.ic@group.ids$tip.clusters <- as.character(hydra.ic@group.ids$tip.clusters)
```

Visualize the chosen clusters to use for random walks.

```
plotDim(hydra.ic, "tip.clusters", label.clusters = T, legend = F)
```



Diffusion Map

Calculate the transition probabilities

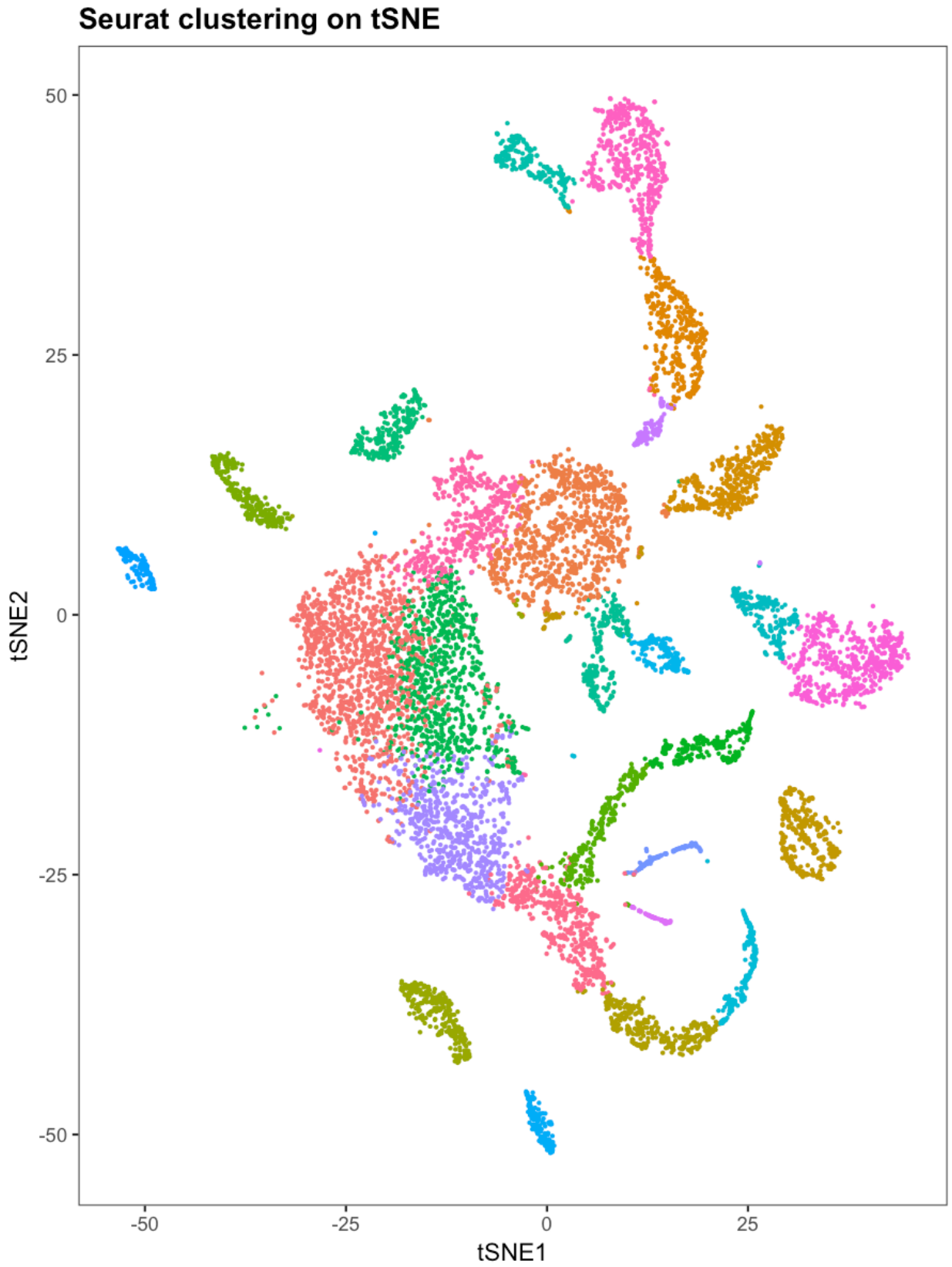
We calculated our diffusion map using 100 nearest neighbors (~ square root of number of cells in data), using sigma determine by *destiny*'s local sigma determination mode (i.e. based on the distance to the 5-7th nearest neighbors). This mode is not deterministic, so we load our previous result, but the command run previously was:

```
hydra.ic <- calcDM(hydra.ic, knn = 100, sigma.use = "local")
```

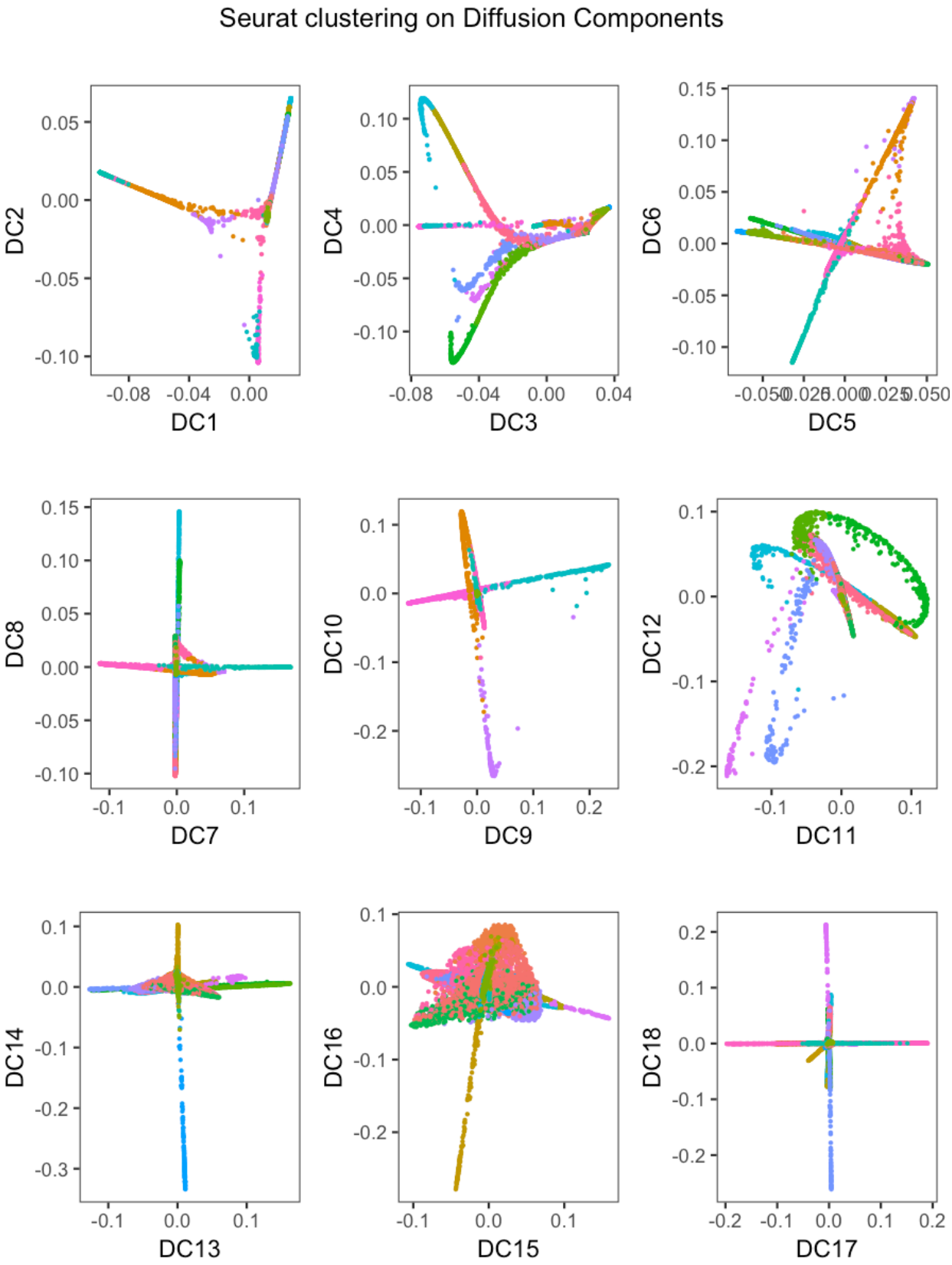
Evaluation of diffusion map

The diffusion map parameters seem fairly good. DCs 1/2 has a representation of the glands, DCs 3/4 seem to encode the nematocytes, and some neuron types become apparent in DCs 5/6 and DCs 13/14, 15/16.

```
plotDim(hydra.ic, label = "clustering2", legend = F, label.clusters = F, plot.title = "Seurat clustering on tSNE")
```




```
plotDimArray(hydra.ic, label = "clustering2", reduction.use = "dm", dims.to.plot = 1:18,
    legend = F, label.clusters = F, plot.title = "", outer.title = "Seurat clustering on Diffusion Components")
```


Seurat clustering on Diffusion Components
DC2
DC1
DC4
DC3
DC6
DC5
DC8
DC7
DC10
DC9
DC12
DC11
DC14
DC13
DC16
DC15
DC18
DC17

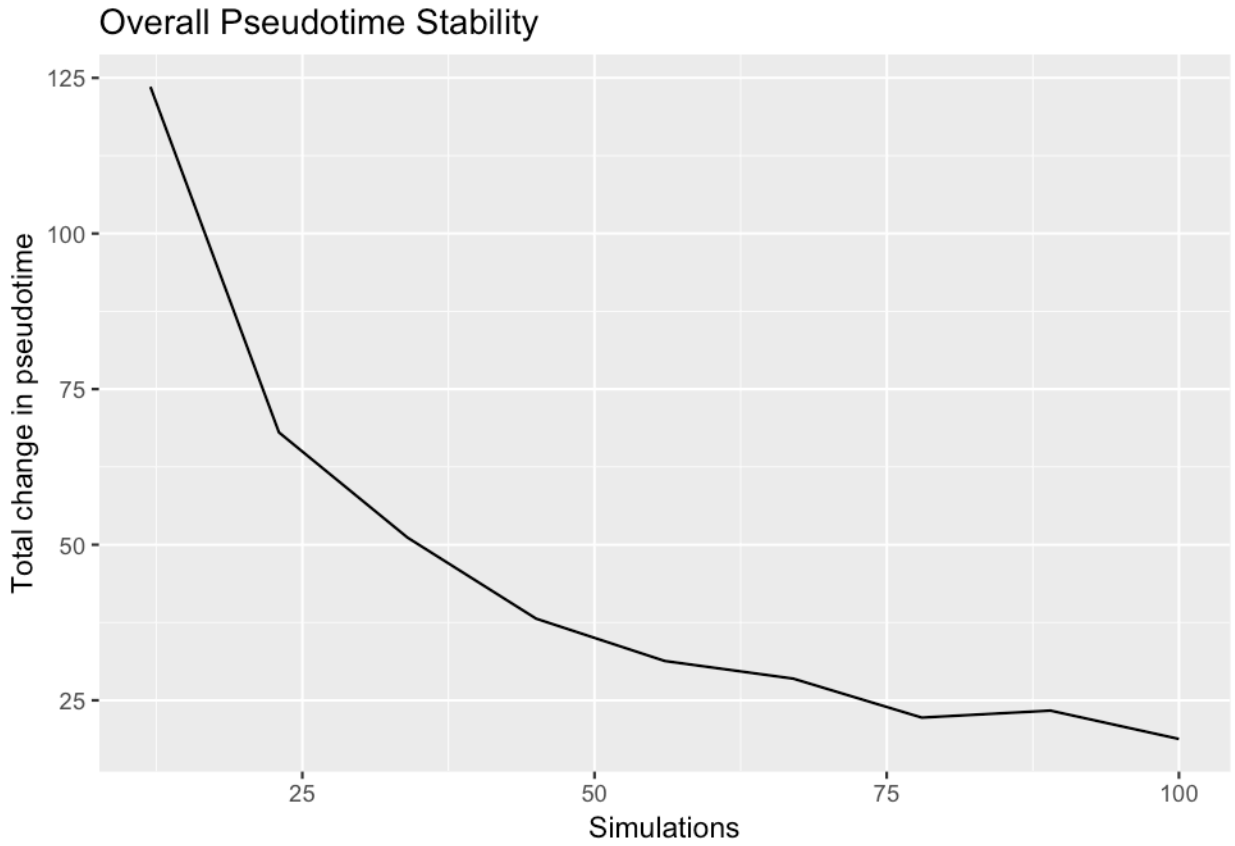
Calculate pseudotime

We perform random simulations to determine the approximate number of transitions to reach each cell in the data from the root and assign it a pseudotime. The random simulations are non-deterministic, so we load our previous result, but the commands run previous were:

```
# Calculate pseudotime floods from the root and load them into the object
flood.result <- floodPseudotime(object, root.cells = root.cells, n = 100, minimum.cells.flooded = 2,
    verbose = T)
```

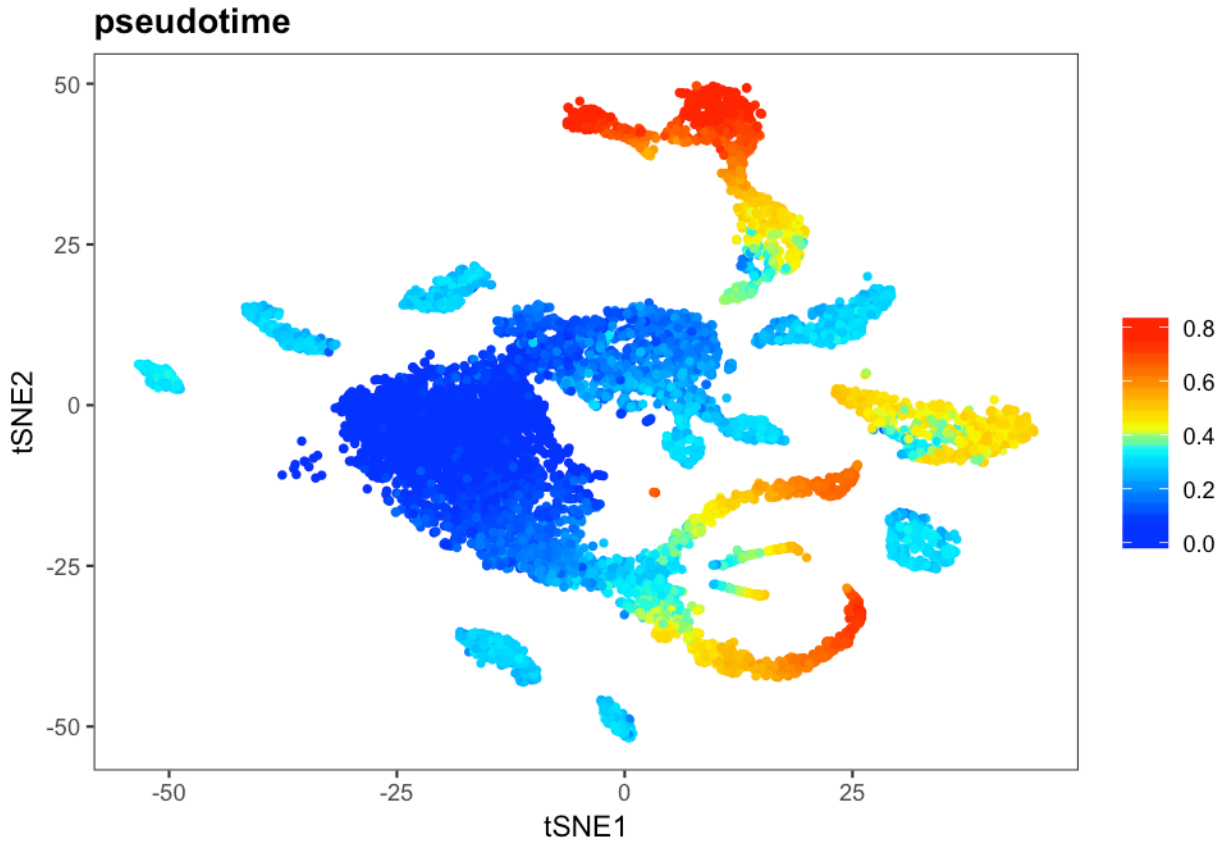
We the process the random simulations to convert them to a 0-1 range pseudotime and verify that enough simulations have been performed.

```
# Process the simulations to determine pseudotime
hydra.ic <- floodPseudotimeProcess(hydra.ic, flood.result, floods.name = "pseudotime")

# Check that enough pseudotime simulations were run -- is change in pseudotime
# reaching an asymptote?
pseudotimePlotStabilityOverall(hydra.ic)
```



This produces a nice pseudotime ordering in the data. For instance, a strong pseudotime gradient is observed in nematogenesis and zymogen/granular mucous glands, and a moderate pseudotime gradient is observed in the spumous mucous glands and neurons. It is important to note that not all terminal populations have the latest pseudotime. This is because this data is not a time-course, but a profile of a homeostatic animal. In this case, the length of each trajectory in pseudotime is more reflective of the transcriptional differences it exhibits compared to the stem cell population.

```
# Inspect pseudotime on the tSNE plot
plotDim(hydra.ic, "pseudotime")
```



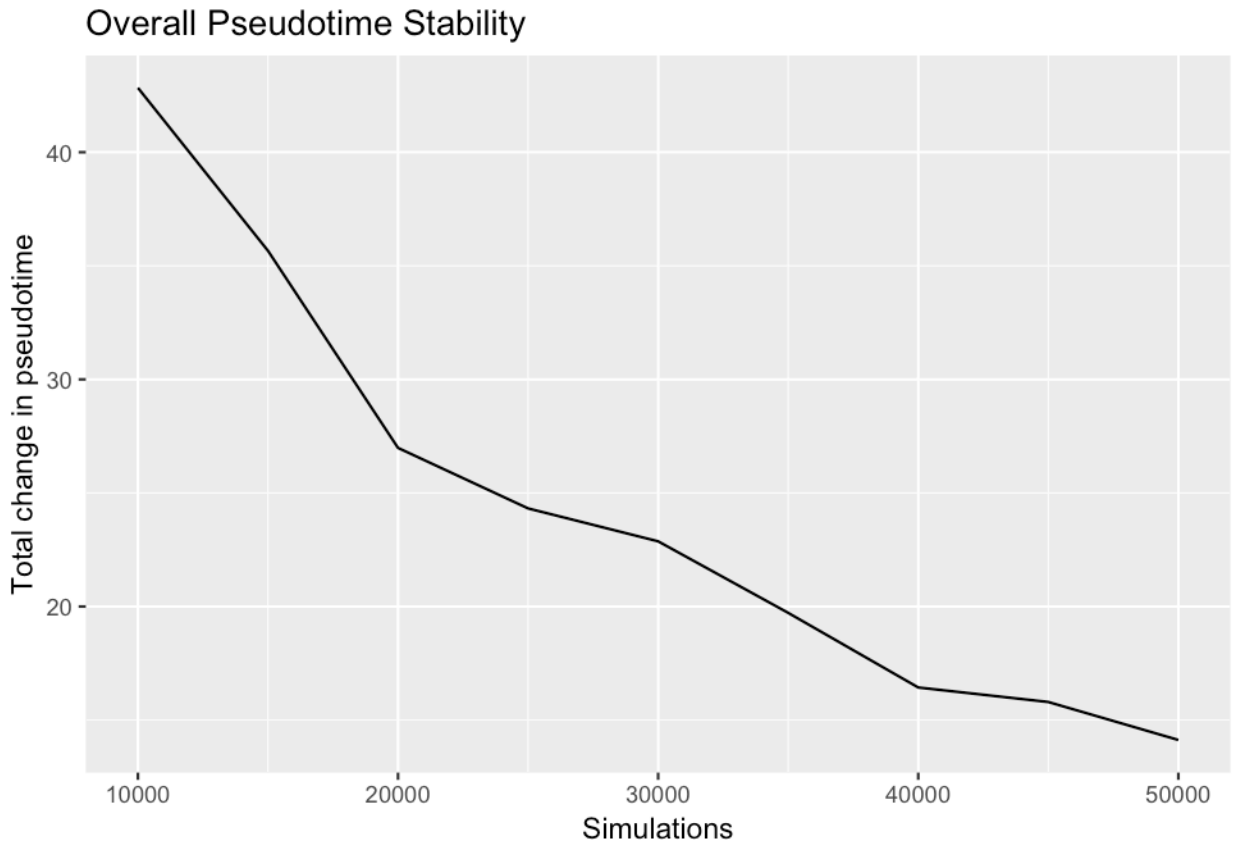
Biased random walks

We performed biased random walks from each of the selected tips. The random walk parameters were fairly standard (optimal 0 cells forward, maximum 100 cells back), with 50,000 walks per tip. The random simulations are non-deterministic, so we load our previous result, but the commands run previous were:

```
# Bias the transition probabilities by cellular pseudotime
pseudotime.logistic <- pseudotimeDetermineLogistic(hydra.ic, pseudotime = "pseudotime",
    optimal.cells.forward = 0, max.cells.back = 100, pseudotime.direction = "<")
tm.biased <- as.matrix(pseudotimeWeightTransitionMatrix(hydra.ic, pseudotime = "pseudotime",
    logistic.params = pseudotime.logistic, pseudotime.direction = "<"))

# Simulate biased random walks
walks.en <- simulateRandomWalksFromTips(hydra.ic, "tip.clusters", root.cells = root.cells,
    transition.matrix = tm.biased, n.per.tip = 50000, root.visits = 1)
```

Then we process the walks and check that enough simulations have been run by checking whether the total change in their assignment reaches an asymptote (which it does here).

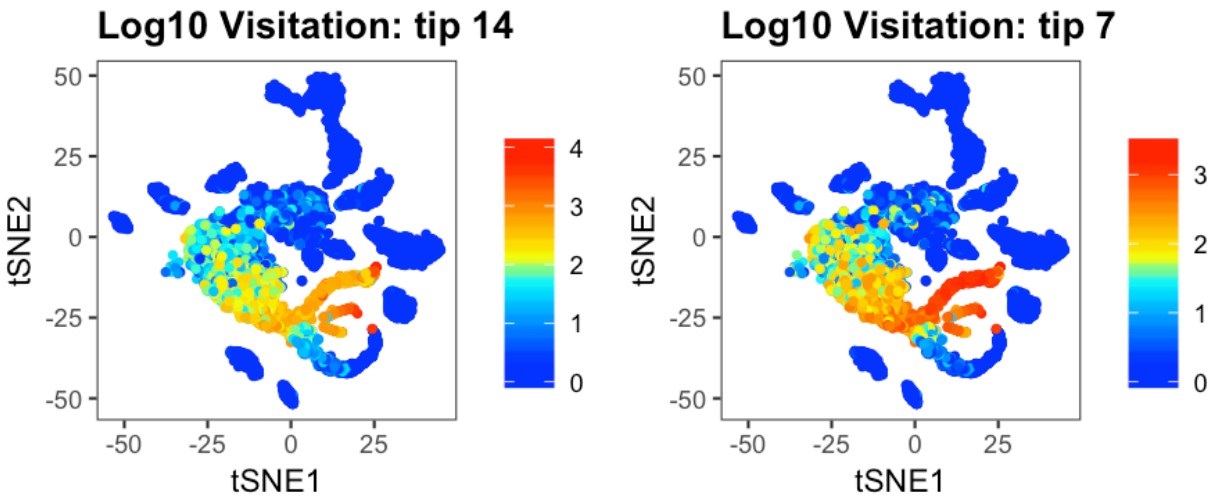
```
hydra.ic <- processRandomWalksFromTips(hydra.ic, walks.list = walks.ic, verbose = F)

# Check that enough random walk simulations were run -- is change in pseudotime
# reaching an asymptote?
pseudotimePlotStabilityOverall(hydra.ic)
```



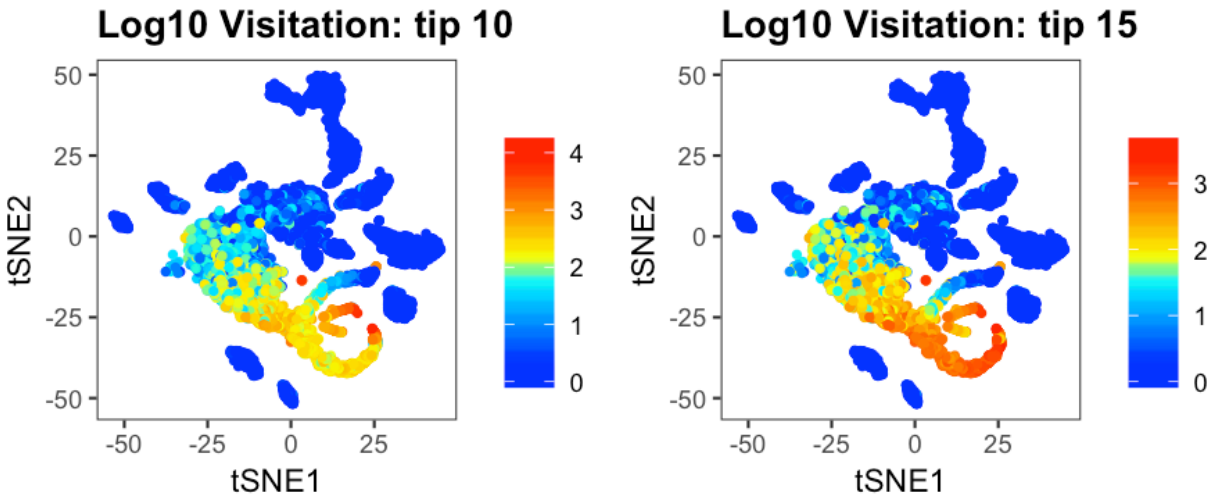
Evaluation of random walks

For some terminal populations, we tried random walks from multiple potential tips to make sure that they did a good job of visiting the rest of the data. To build the tree, we chose a single potential tip for each cell type. It seemed like for nematocytes, tip 14 does a better job visiting the rest of the nematocytes than 7, and either 10 or 15 would work. For zymogen glands on the aboral end, tip 1 does not visit the lower body, so also will use 2 as a tip. The rest of the zymogen/granular is visited, so can just use 6 at the oral end. In the neurons, seems like clusters 5 and 16 don't visit each other, so should use both combined as a single tip.

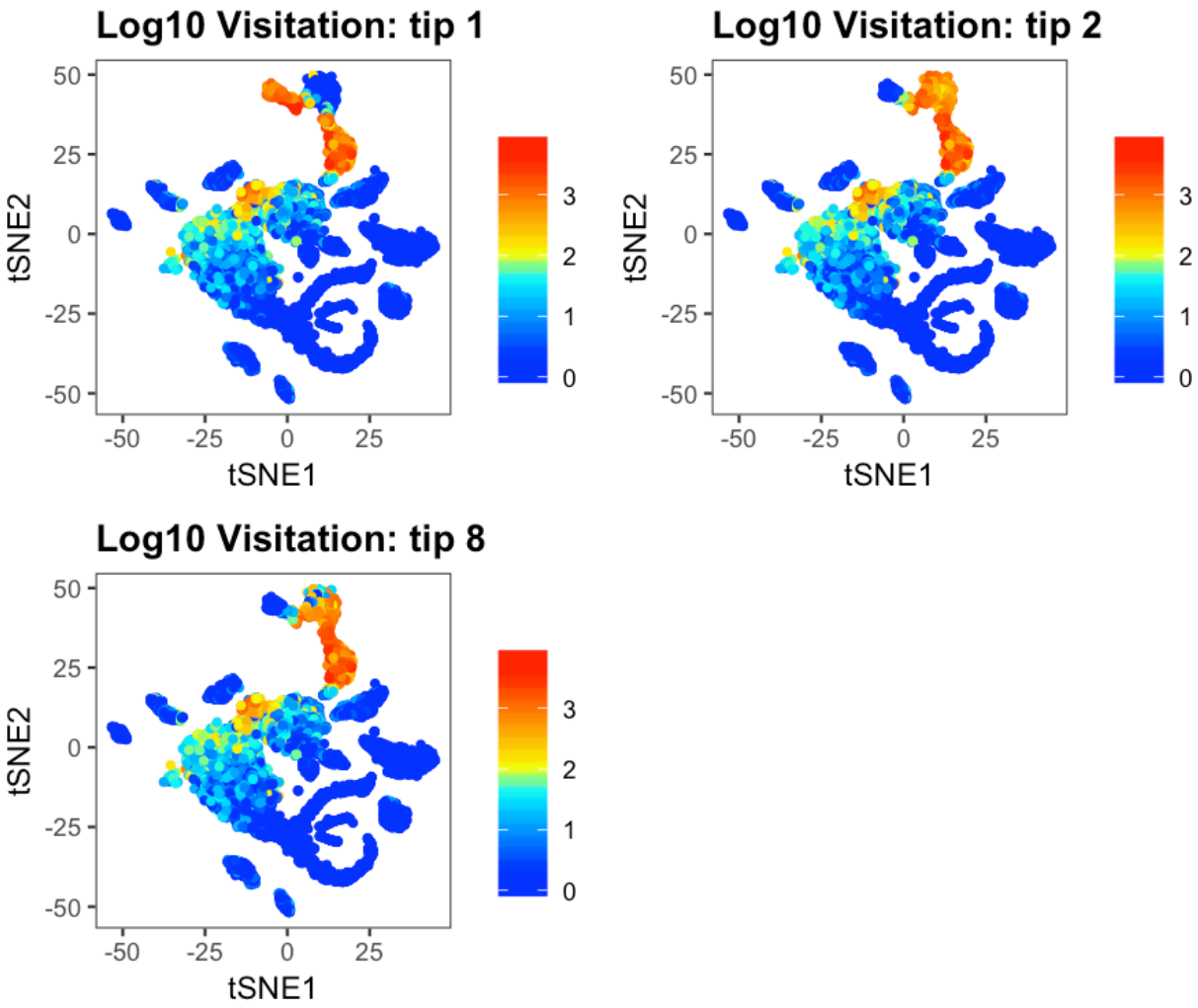
```
# Nematocyte A
gridExtra::grid.arrange(grobs = list(plotDim(hydra.ic, "visitfreq.log.14", plot.title = "Log10 Visitation: tip 14"),
    plotDim(hydra.ic, "visitfreq.log.7", plot.title = "Log10 Visitation: tip 7")),
    ncol = 2)
```



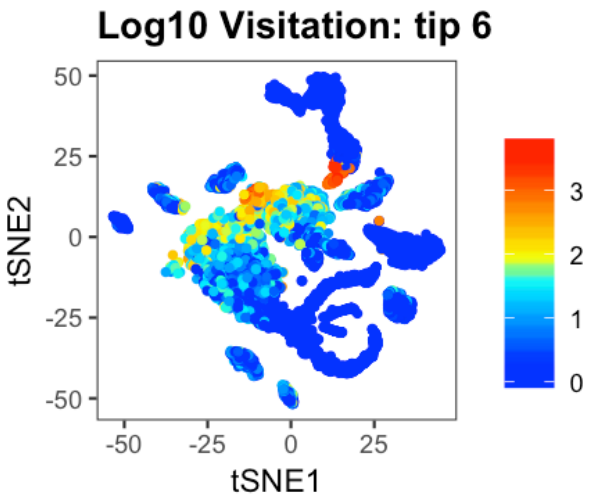

```
# Nematocyte B
gridExtra::grid.arrange(grobs = list(plotDim(hydra.ic, "visitfreq.log.10", plot.title = "Log10 Visitation: tip 10"),
    plotDim(hydra.ic, "visitfreq.log.15", plot.title = "Log10 Visitation: tip 15")),
    ncol = 2)
```

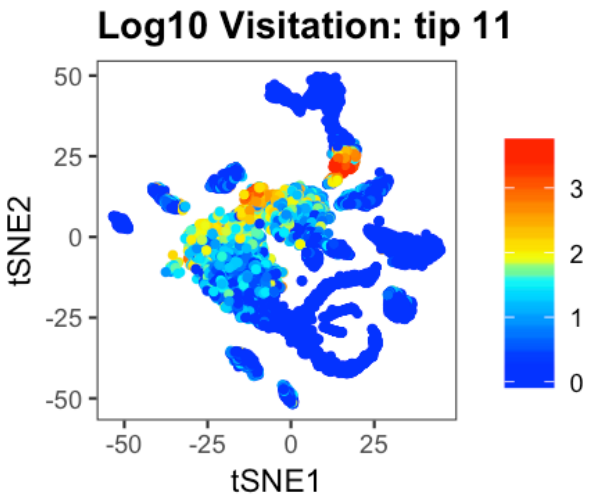


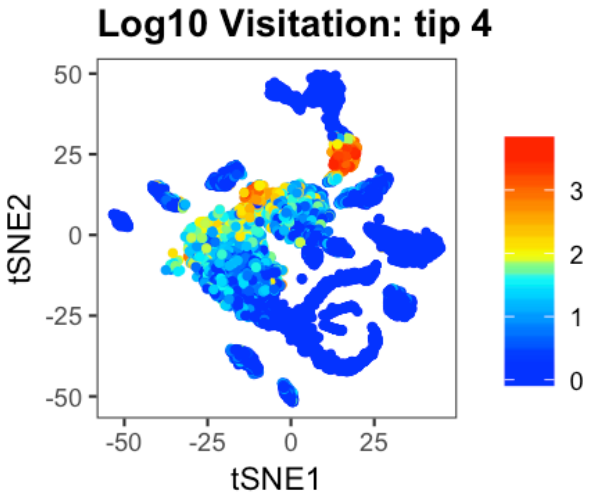

```
# Zymogen aboral
gridExtra::grid.arrange(grobs = list(plotDim(hydra.ic, "visitfreq.log.1", plot.title = "Log10 Visitation: tip 1"),
    plotDim(hydra.ic, "visitfreq.log.2", plot.title = "Log10 Visitation: tip 2"),
    plotDim(hydra.ic, "visitfreq.log.8", plot.title = "Log10 Visitation: tip 8")),
    ncol = 2)
```



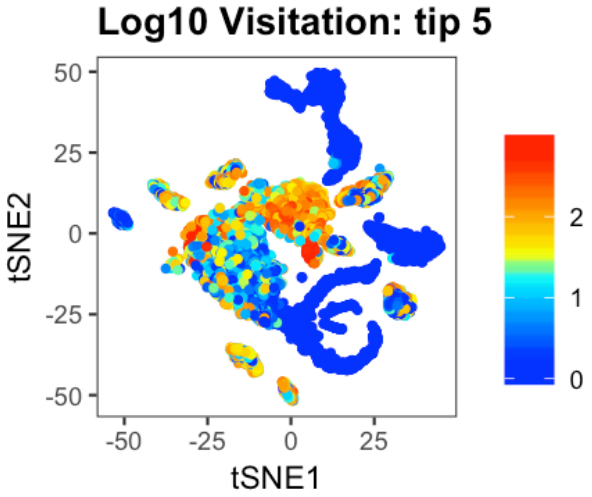

```
# Zymogen oral
gridExtra::grid.arrange(grobs = list(plotDim(hydra.ic, "visitfreq.log.6", plot.title = "Log10 Visitation: tip 6"),
    plotDim(hydra.ic, "visitfreq.log.11", plot.title = "Log10 Visitation: tip 11"),
    plotDim(hydra.ic, "visitfreq.log.4", plot.title = "Log10 Visitation: tip 4")),
    ncol = 2)
```

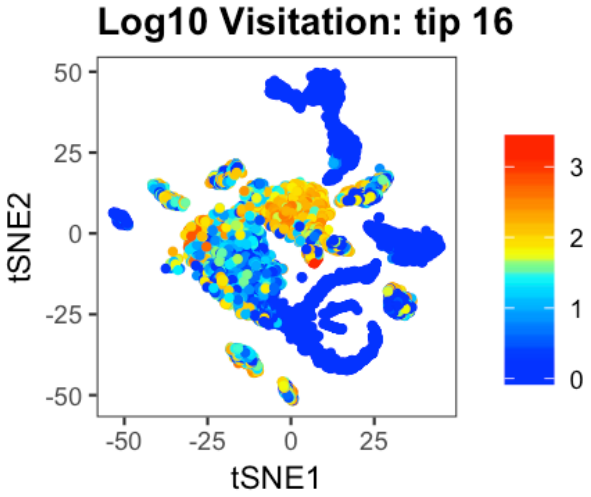







```
# Neuron
gridExtra::grid.arrange(grobs = list(plotDim(hydra.ic, "visitfreq.log.5", plot.title = "Log10 Visitation: tip 5"),
    plotDim(hydra.ic, "visitfreq.log.16", plot.title = "Log10 Visitation: tip 16")),
    ncol = 2)
```





Build the tree

Define tips to use

Based on the visitation above, we chose a cluster to represent each terminal population as a tip.

```
# Clusters chosen above as the best ones to use for building the tree Nemato: 14,
# 13, 12, 10 Neuro: 24, 20, 21, 19, 23, 18, 22, (5+16), 17 Spumous: 3, 9
# Granular: 6 Zymogen: 2, 1

# Load tip information into the tree
hydra.ic.tree <- loadTipCells(hydra.ic, "tip.clusters")

# Combine the two small clusters in a single neural population into one tip
hydra.ic.tree <- combineTipVisitation(hydra.ic.tree, "16", "5", new.tip = "16")

# Choose tips to use in building the tree
tips.to.build <- c("14", "13", "12", "10", "24", "20", "21", "19", "23", "18", "22",
    "16", "3", "9", "6", "2", "1", "17")
```

Build tree and name the tips

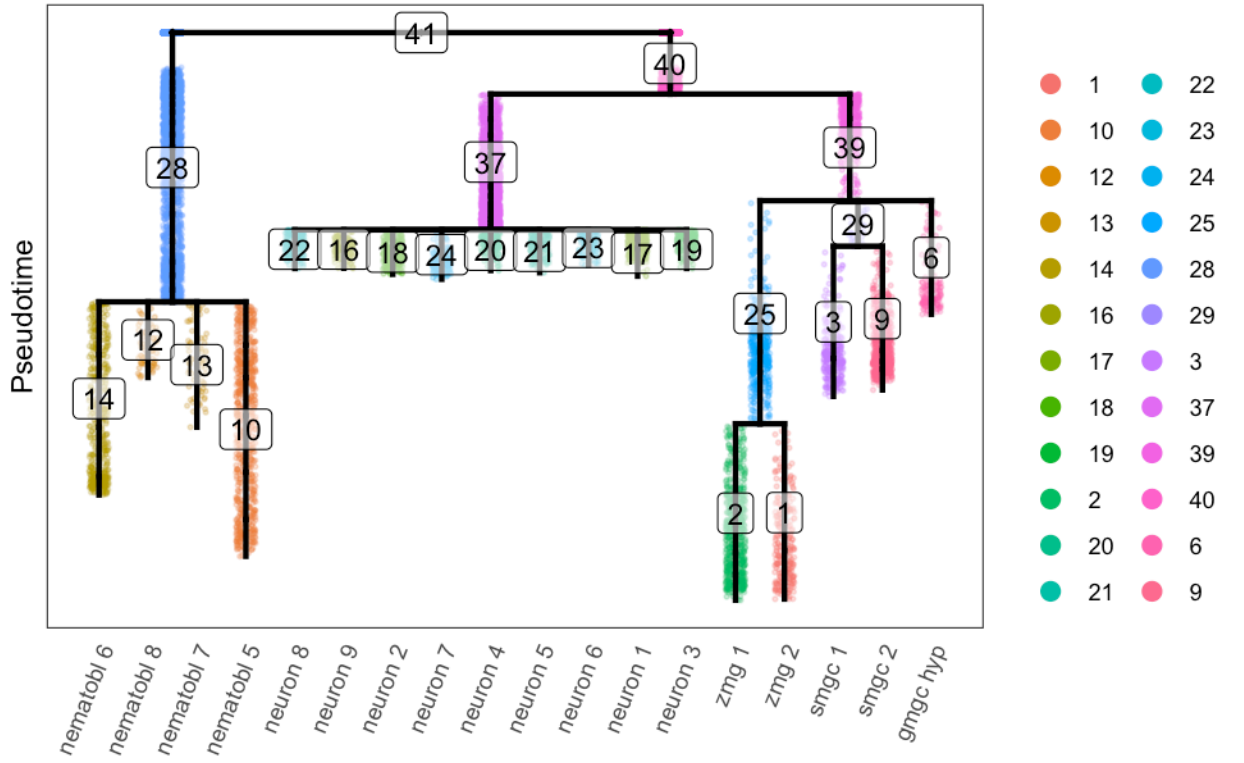
Then, we build the tree, and name the tips based on the cell population they represent.

```
# Build the tree
hydra.ic.tree <- buildTree(hydra.ic.tree, pseudotime = "pseudotime", divergence.method = "preference",
    save.all.breakpoint.info = T, tips.use = tips.to.build, cells.per.pseudotime.bin = 25,
    bins.per.pseudotime.window = 10, p.thresh = 0.001, min.cells.per.segment = 10,
    save.breakpoint.plots = NULL, verbose = F)

hydra.ic.tree <- nameSegments(object = hydra.ic.tree, segments = c("1", "2", "6",
    "3", "9", "14", "13", "12", "10", "24", "20", "21", "17", "16", "22", "18", "19",
    "23"), segment.names = c("zmg 2", "zmg 1", "gmgc hyp", "smgc 1", "smgc 2", "nematobl 6",
    "nematobl 7", "nematobl 8", "nematobl 5", "neuron 7", "neuron 4", "neuron 5",
    "neuron 1", "neuron 9", "neuron 8", "neuron 2", "neuron 3", "neuron 6"), short.names = c("ZymLower",
    "ZymUpper", "GranularUpper", "SpumLower", "SpumUpper", "NEM_6", "NEM_7", "NEM_8",
    "NEM_5", "NC_7", "NC_4", "NC_5", "NC_1", "NC_9", "NC_8", "NC_2", "NC_3", "NC_6"))
```

Inspect the tree

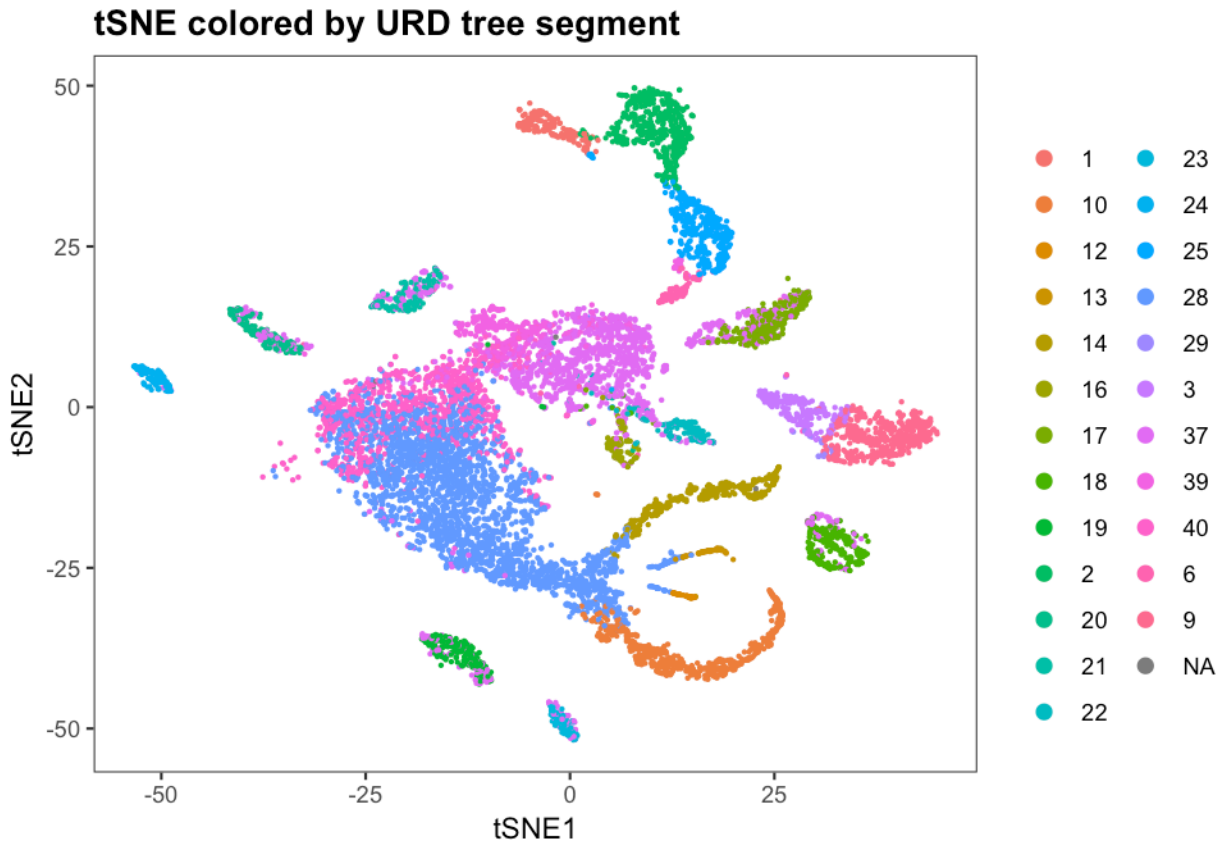
We recover a tree with major branches representing nematogenesis, neurogenesis, and gland development. A potential common progenitor state to neurogenesis and gland development (segment 40) is an unexpected and interesting finding. The tree combines differentiation progress in the nematogenesis, neurogenesis, and early gland development (i.e. segment 39) portions of the tree, with what is primarily spatial information in the later gland development regions of the tree (segments 25, 2, 1, 29, 3, 9, and 6).

```
plotTree(hydra.ic.tree, "segment", label.segments = T, title = "")
```



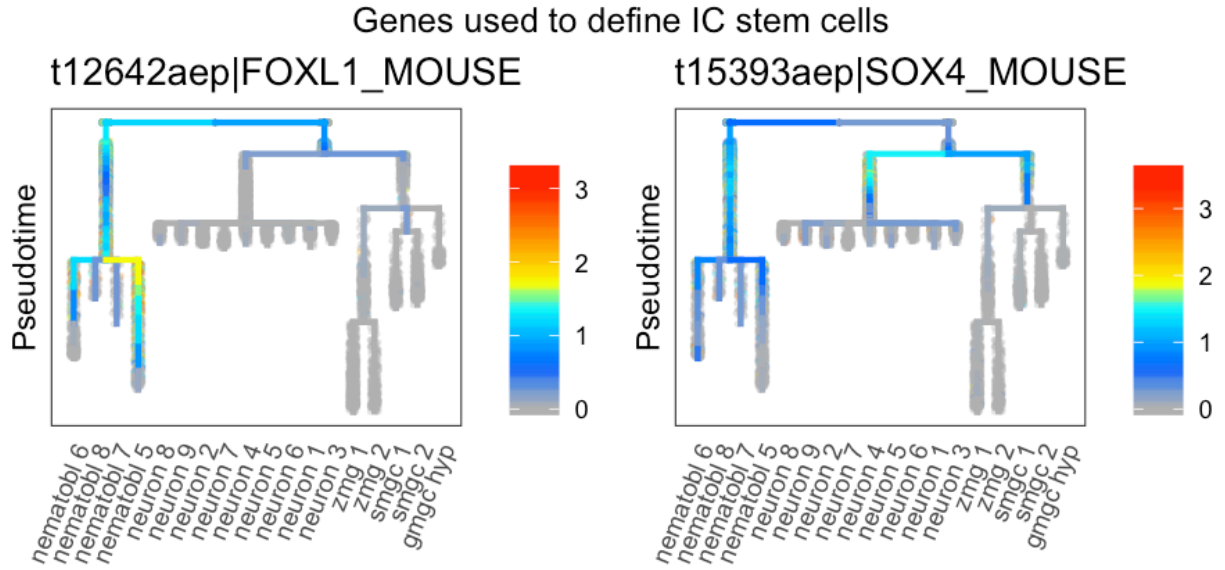
We can visualize the assignment of segment identities on the original tSNE. In this case, there is reasonable agreement between the structure observed in the tSNE and recovered by URD.

```
plotDim(hydra.ic.tree, "segment", plot.title = "tSNE colored by URD tree segment")
```

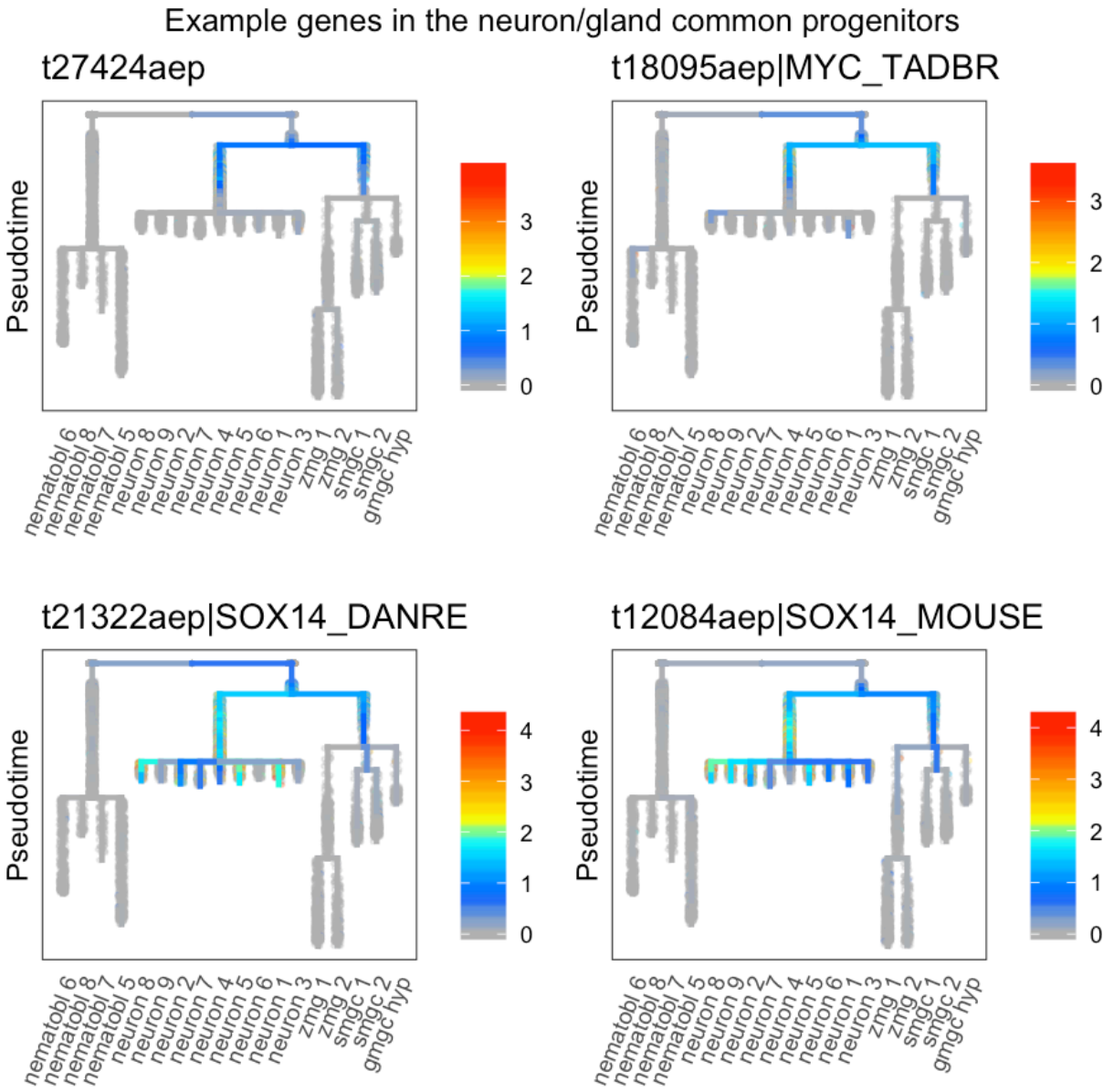


We can plot gene expression of various populations on the tree to see where they are expressed.

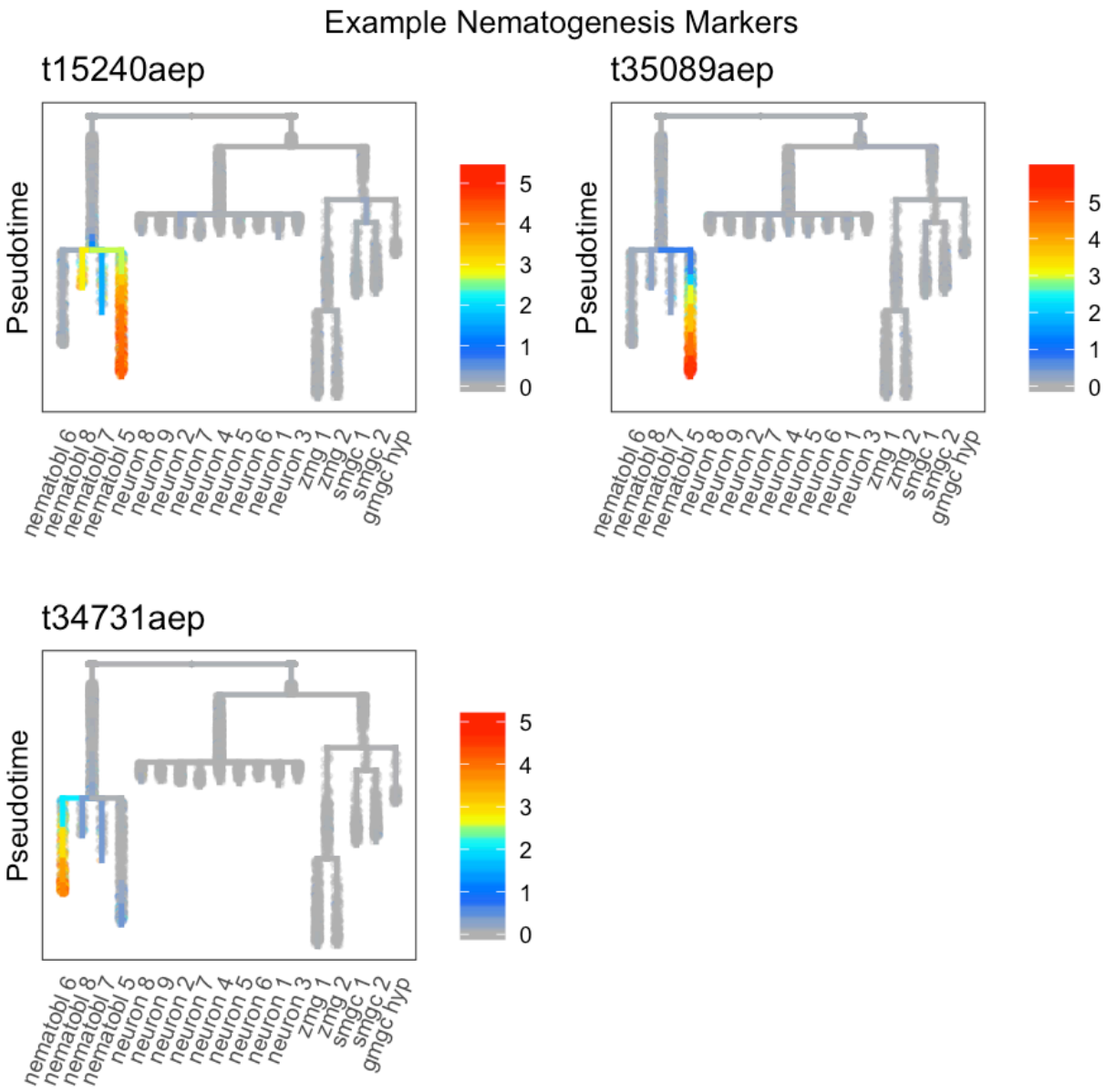
```
# Markers of the stem cells
gridExtra::grid.arrange(grobs = list(plotTree(hydra.ic.tree, "t12642aep|FOXL1_MOUSE"),
    plotTree(hydra.ic.tree, "t15393aep|SOX4_MOUSE")), ncol = 2, top = "Genes used to define IC stem cells")
```




```
# Markers of the common progenitors
gridExtra::grid.arrange(grobs = list(plotTree(hydra.ic.tree, "t27424aep"), plotTree(hydra.ic.tree,
    "t18095aep|MYC_TADBR"), plotTree(hydra.ic.tree, "t21322aep|SOX14_DANRE"), plotTree(hydra.ic.tree,
    "t12084aep|SOX14_MOUSE")), ncol = 2, top = "Example genes in the neuron/gland common progenitors")
```




```
# Markers in the nematocytes
gridExtra::grid.arrange(grobs = list(plotTree(hydra.ic.tree, "t15240aep"), plotTree(hydra.ic.tree,
    "t35089aep"), plotTree(hydra.ic.tree, "t34731aep")), ncol = 2, top = "Example Nematogenesis Markers")
```



Save results

```
saveRDS(hydra.ic.tree, file = paste0(main.path, "URD/IC/Hydra_URD_IC.rds"))
```