SA10: Hydra URD Ectoderm

Jeff Farrell

October 1, 2018

Setup

```
# 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/Ectoderm/"), recursive = T)
```

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

Load Data

```
# Load subsetted ectoderm Seurat object and convert to an URD object
hydra.seurat <- readRDS(paste0(main.path, "/objects/Hydra_Seurat_Ecto.rds"))
hydra.ec <- seuratToURD(hydra.seurat)
rm(hydra.seurat)
shh <- gc()

# Load full-data transcriptome Seurat object and cover to an URD object.
hydra.seurat <- readRDS(paste0(main.path, "objects/Hydra_Seurat_Whole_Transcriptome.rds"))
hydra.full <- seuratToURD(hydra.seurat)
```

```
## [1] "Marchenko-Pastur eigenvalue null upper bound: 18.6992548391687"
## [1] "11 PCs have larger eigenvalues."
```

```
rm(hydra.seurat)
shh <- gc()

# Load full-data NMF results
nmf.full.cells <- as.data.frame(t(read.csv(paste0(main.path, "nmf/wt_K96/GoodMeta_CellScores.csv"),
    row.names = 1)))
nmf.ecto.cells <- as.data.frame(t(read.csv(paste0(main.path, "nmf/ec_K76/GoodMeta_CellScores.csv"),
    row.names = 1)))
rownames(nmf.ecto.cells) <- gsub("X", "", gsub("\\.", "-", rownames(nmf.ecto.cells)))
colnames(nmf.ecto.cells) <- paste0("ec", colnames(nmf.ecto.cells))
rownames(nmf.full.cells) <- gsub("X", "", gsub("\\.", "-", rownames(nmf.full.cells)))

# Scale NMF results to 0-1
nmf.full.cells <- sweep(nmf.full.cells, 2, apply(nmf.full.cells, 2, max), "/")

# Put NMF results into URD object
hydra.ec@nmf.c1 <- as(as.matrix(cbind(nmf.full.cells[colnames(hydra.ec@logupx.data),
    ], nmf.ecto.cells[colnames(hydra.ec@logupx.data), ])), "dgCMatrix")
hydra.full@nmf.c1 <- as(as.matrix(nmf.full.cells[colnames(hydra.full@logupx.data),
    ]), "dgCMatrix")
```

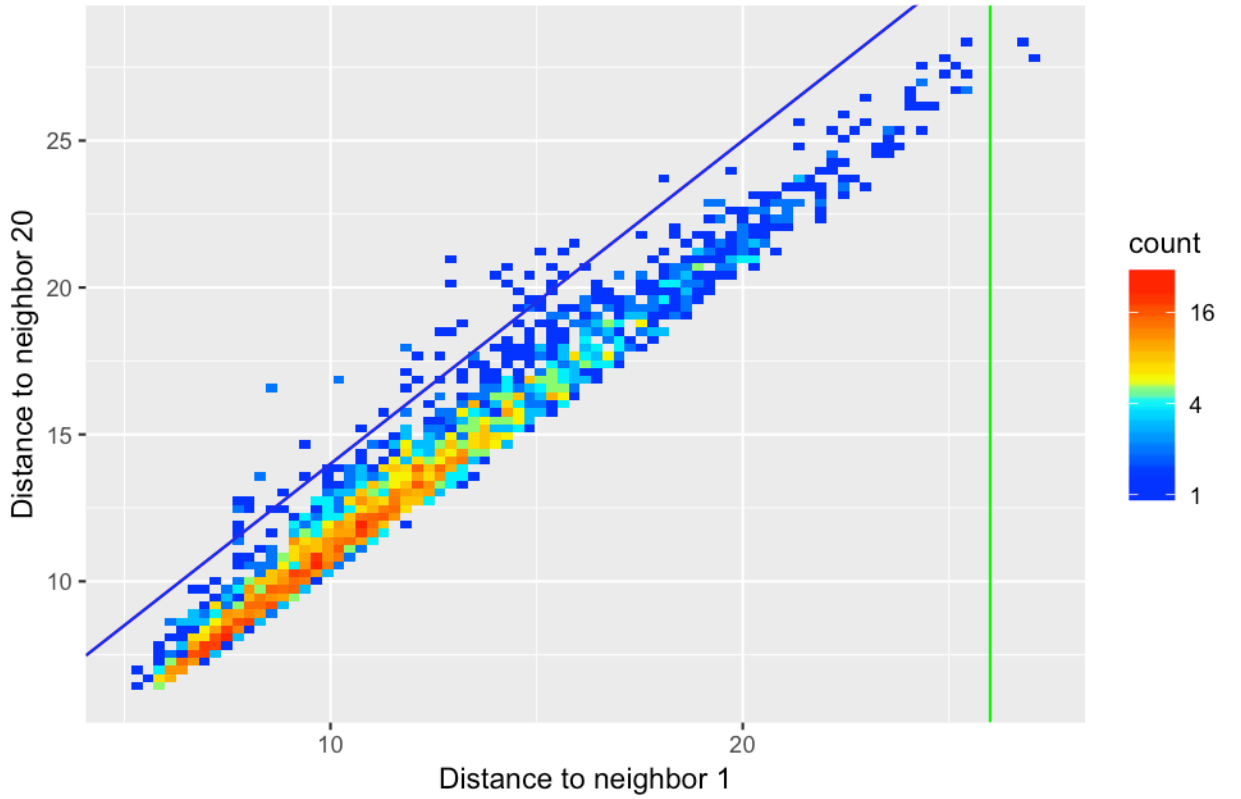
Remove outlier cells and doublets

Before attempting to build trajectories, it's important to clean the data by removing outlier cells that will confound the calculation of transition probabilities and removing doublets. Due to the phagocytic behavior of the dissociated ectodermal cells, many cells that exhibit non-ectodermal gene expression have to be removed.

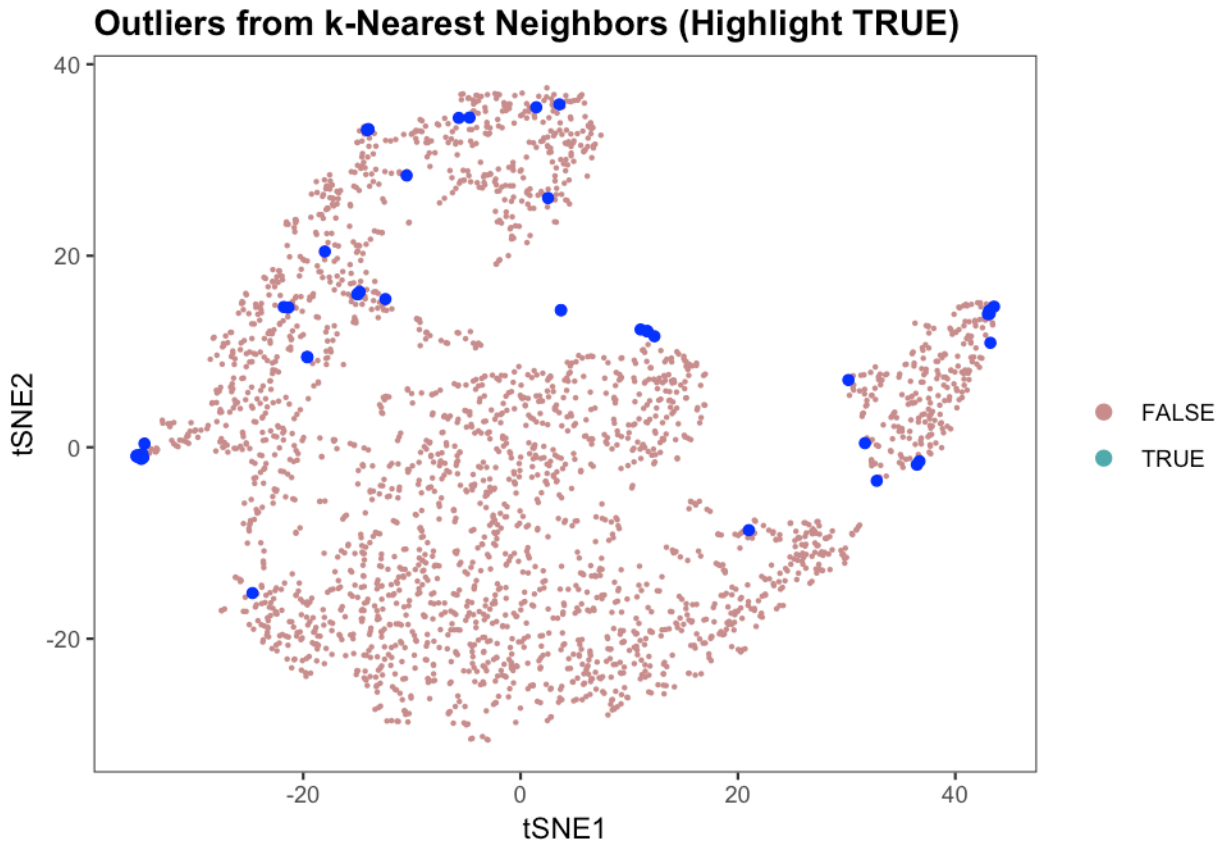
Calculate k-nearest neighbor networks

A few cells have unusually large distances to their nearest neighbors and will perform poorly in the diffusion map, so it is good to remove those.

```
# Calculate k-nearest neighbor network
hydra.ec <- calcKNN(hydra.ec, nn = 200)

# Choose outliers that have very unusual nearest neighbor distances
knn.outliers <- knnOutliers(hydra.ec, x.max = 26, slope.r = 1.1, int.r = 3, slope.b = 1.1,
    int.b = 3)
```



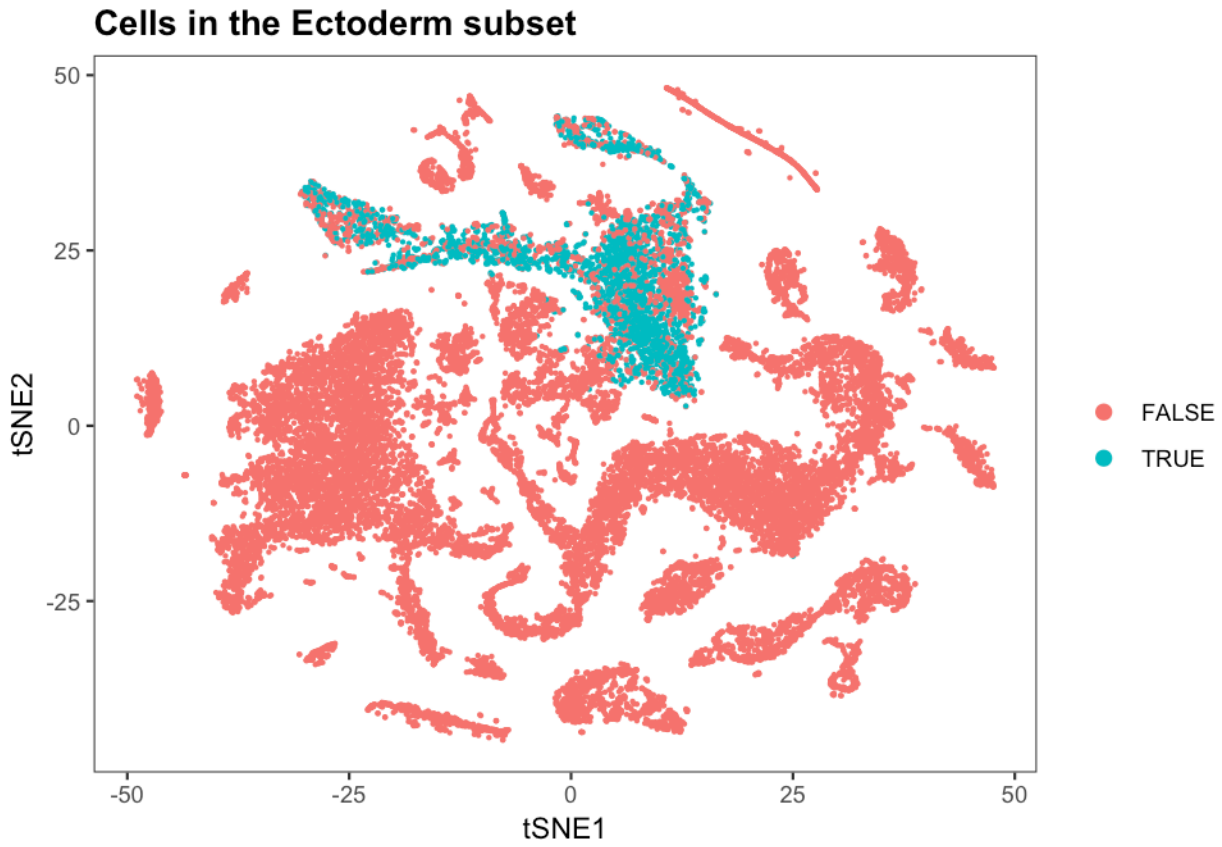

```
# Check out who those outliers are to make sure you're not cropping out all
# differentiated cells or something
hydra.ec <- groupFromCells(hydra.ec, group.id = "knn.outliers", cells = knn.outliers)
plotDimHighlight(hydra.ec, "knn.outliers", "TRUE", highlight.color = "blue", plot.title = "Outliers from k-Nearest Neighbors")
```



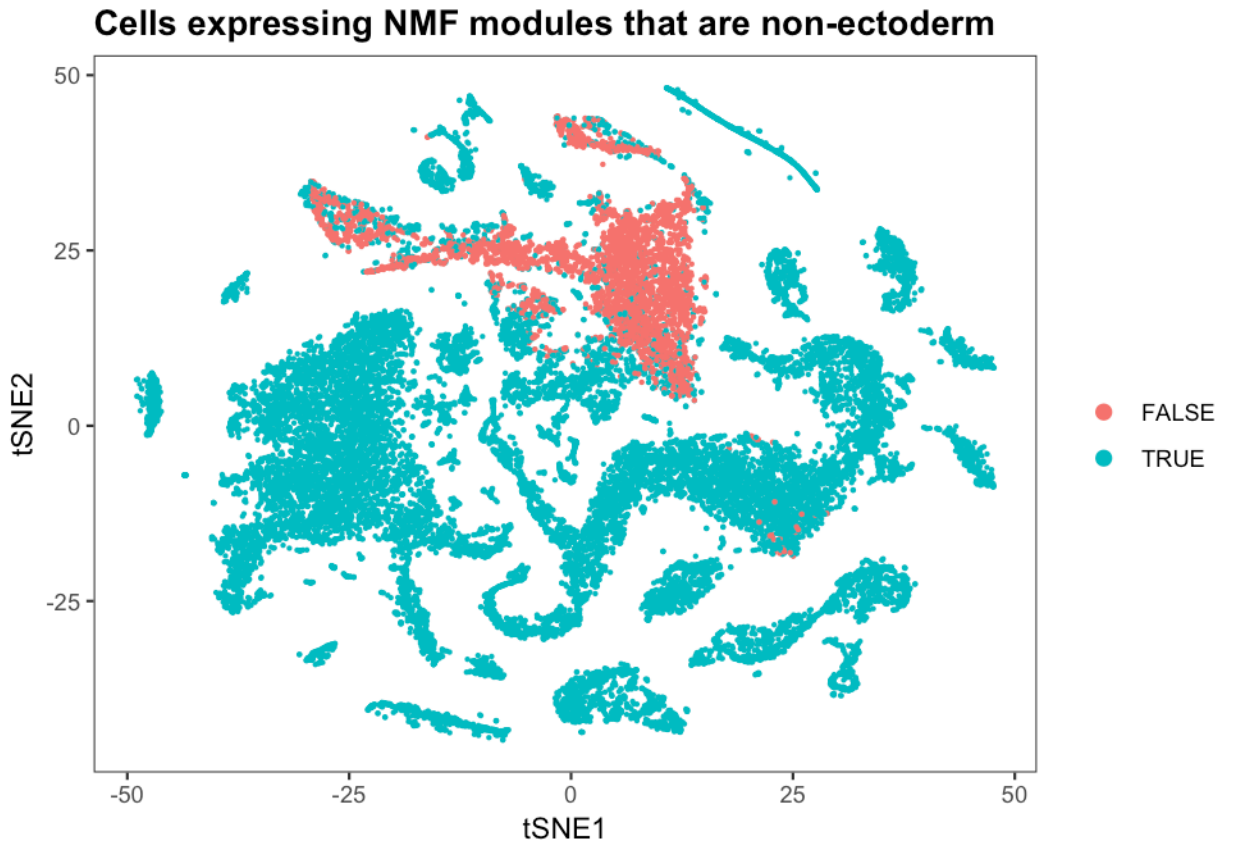
Non-ectodermal contaminating cells

Many cells in the data represent biological doublets, since cells of the ectodermal lineage readily engulf other cells once dissociated. In order to remove these cells, we used a non-negative matrix factorization (NMF) decomposition of the full data set, and identified those modules that are not primarily expressed in the ectoderm. Any cells in the 'ectoderm' lineage that had significant expression of modules primarily associated with other cell types were removed.

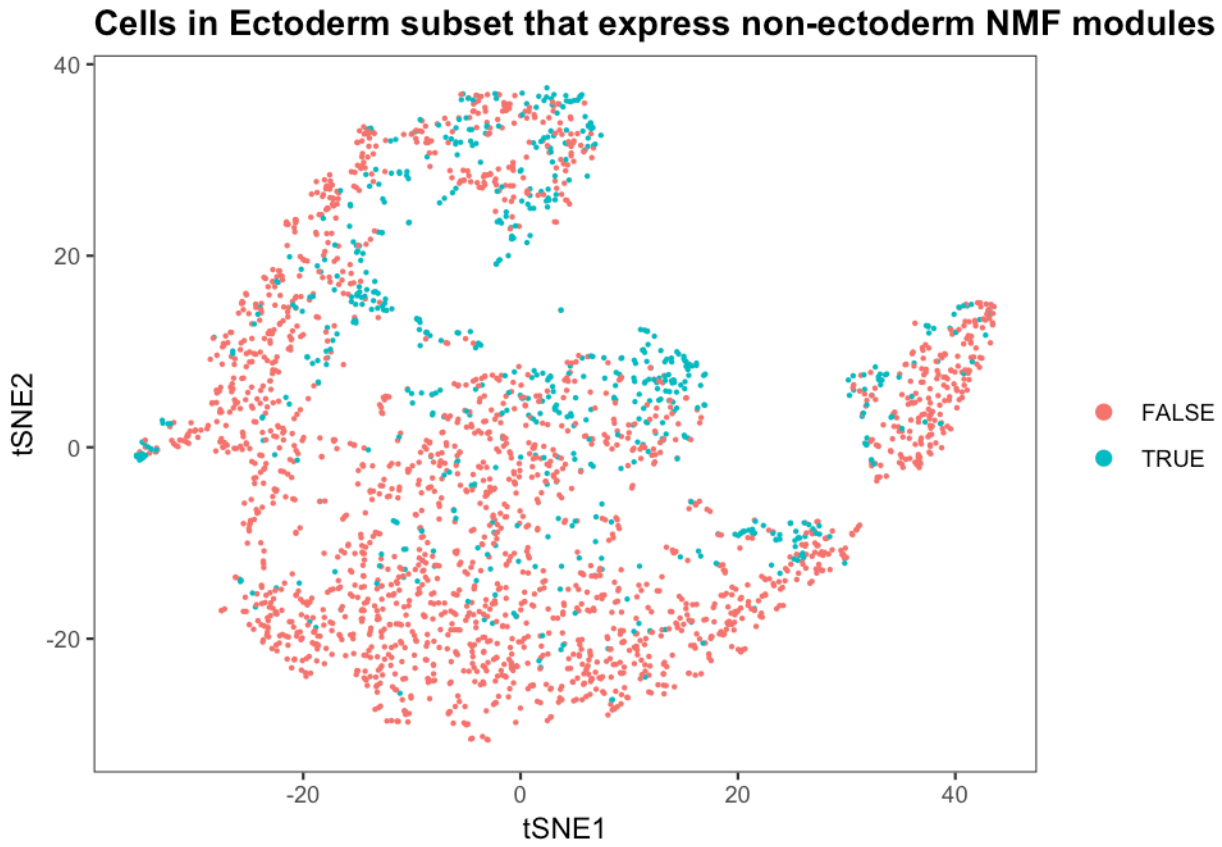
```
# Modules from NMF calculated on entire data that are representative of
# non-endodermal cells.
nmf.full.nonecto <- c("wt2", "wt4", "wt7", "wt8", "wt9", "wt12", "wt13", "wt15",
    "wt16", "wt20", "wt22", "wt23", "wt25", "wt28", "wt30", "wt31", "wt32", "wt34",
    "wt37", "wt38", "wt39", "wt41", "wt43", "wt45", "wt46", "wt47", "wt48", "wt52",
    "wt53", "wt54", "wt55", "wt57", "wt59", "wt61", "wt64", "wt65", "wt66", "wt69",
    "wt70", "wt71", "wt72", "wt73", "wt74", "wt75", "wt76", "wt79", "wt81", "wt82",
    "wt83", "wt84", "wt85", "wt89", "wt92", "wt94", "wt95")

# Identify the cells in the full-data object that are in the endoderm object
hydra.full <- groupFromCells(hydra.full, "ectoderm", colnames(hydra.ec@logupx.data))
plotDim(hydra.full, "ectoderm", plot.title = "Cells in the Ectoderm subset")
```




```
# Which cells express NMF modules that are not associated with the endoderm?
# Here, expression is defined cells that express one or more modules at a level >
# 0.125 (scaled from 0-1)
cells.full.nmfnonecto <- names(which(Matrix::rowSums(hydra.full@nmf.c1[, nmf.full.nonecto] >
    0.125) > 0))
hydra.full <- groupFromCells(hydra.full, "nonecto", cells.full.nmfnonecto)
plotDim(hydra.full, "nonecto", plot.title = "Cells expressing NMF modules that are non-ectoderm")
```




```
# Annotate the ectoderm subset with cells that express non-ectoderm NMF modules
outliers.nmf.full.nonecto <- intersect(cells.full.nmfnonecto, colnames(hydra.ec@logupx.data))
hydra.ec <- groupFromCells(hydra.ec, "outliers.nmf.full.nonecto", outliers.nmf.full.nonecto)
plotDim(hydra.ec, "outliers.nmf.full.nonecto", plot.title = "Cells in Ectoderm subset that express non-ectoderm NMF modules")
```

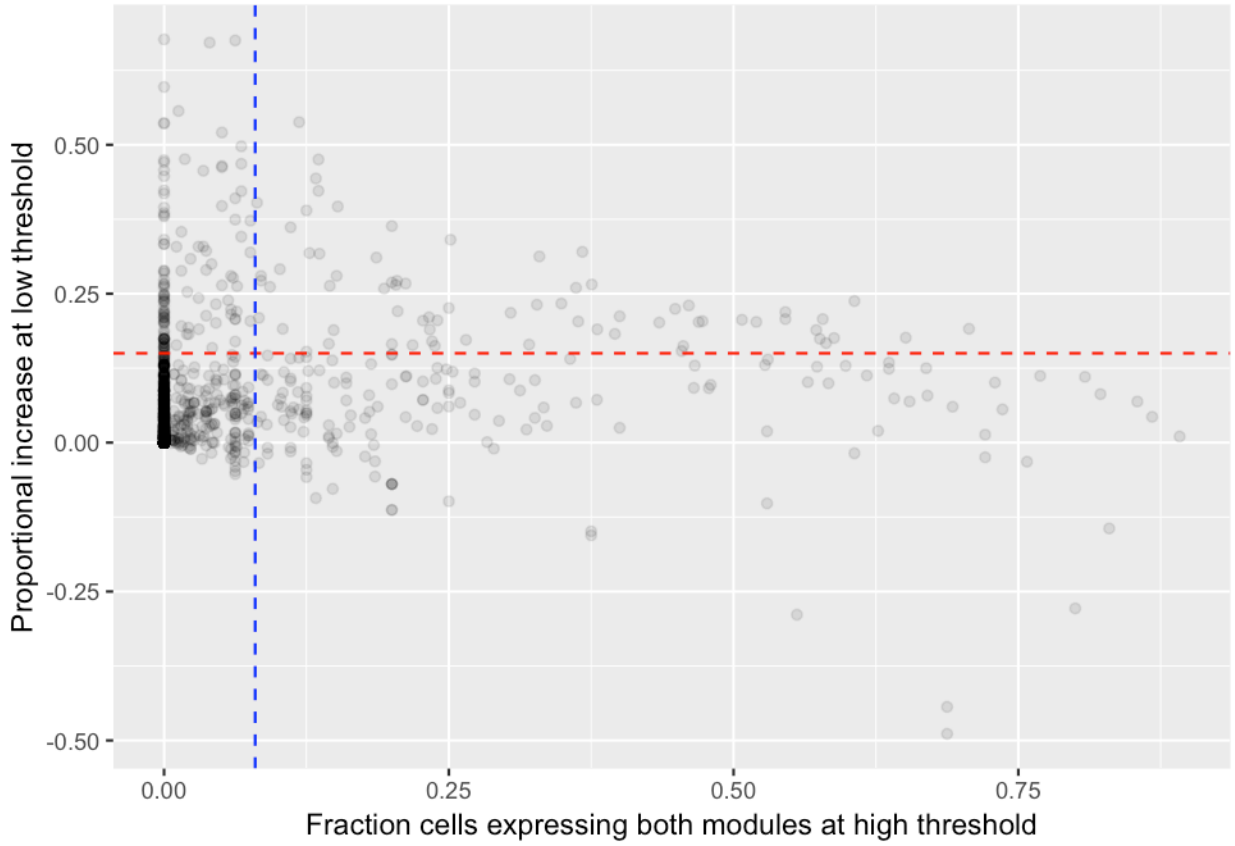


Doublets via NMF

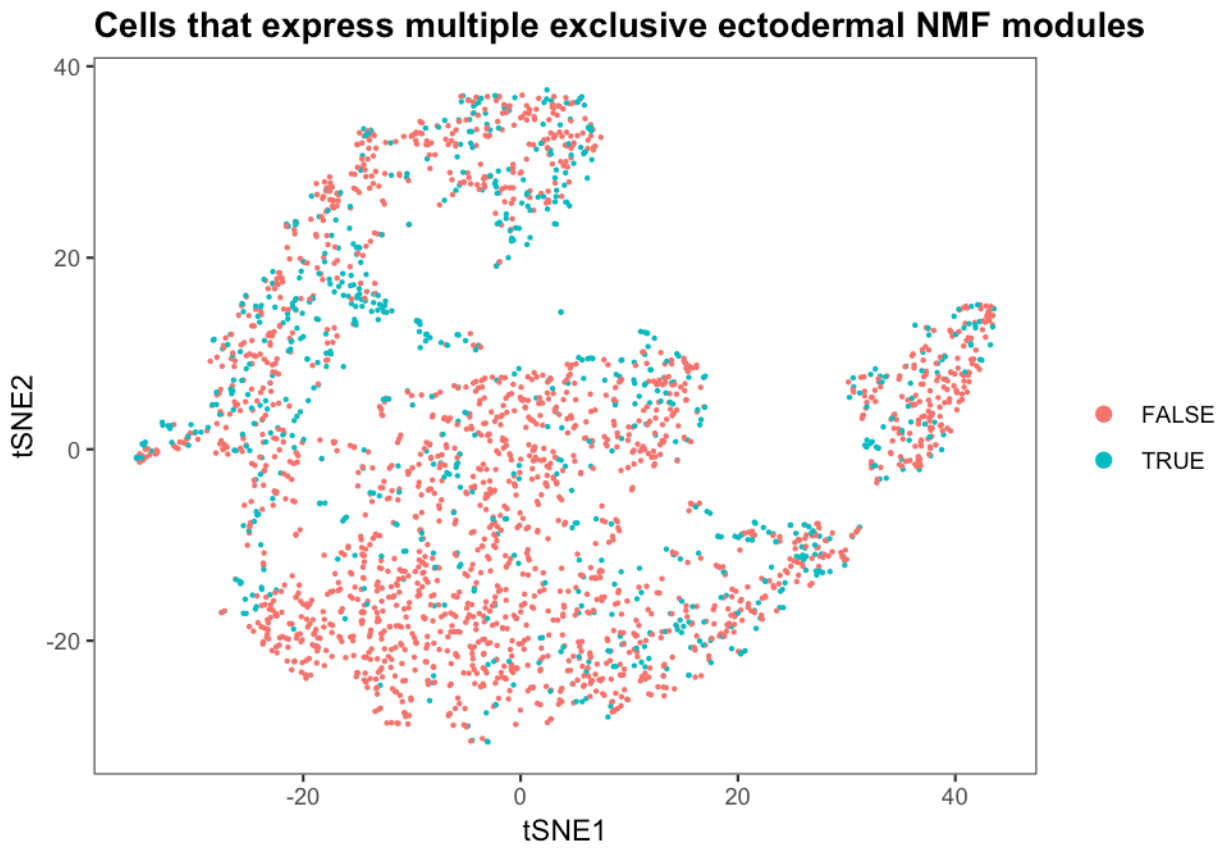
Additionally, some ectodermal cells are doublets of multiple ectodermal cell types. These are detected using NMF decomposition of the ectoderm lineage. These doublets are cells that express multiple NMF modules that are non-overlapping in the data. (In other words, NMF modules that are mutually exclusive in the majority of the data.)

```
# Determine overlaps between module pairs
nmf.mods.ec.use <- c("ec0", "ec4", "ec8", "ec12", "ec13", "ec14", "ec16", "ec17",
    "ec20", "ec21", "ec22", "ec24", "ec25", "ec26", "ec27", "ec29", "ec30", "ec31",
    "ec32", "ec33", "ec34", "ec35", "ec36", "ec38", "ec39", "ec40", "ec41", "ec42",
    "ec44", "ec45", "ec47", "ec48", "ec49", "ec50", "ec51", "ec52", "ec53", "ec54",
    "ec56", "ec57", "ec58", "ec59", "ec60", "ec61", "ec62", "ec63", "ec64", "ec66",
    "ec67", "ec71", "ec72", "ec73", "ec75")
nmf.doublet.combos <- NMFDoubletsDefineModules(hydra.ec, modules.use = nmf.mods.ec.use,
    module.thresh.high = 0.3, module.thresh.low = 0.15)

# Determine which module pairs to use for doublets
NMFDoubletsPlotModuleThresholds(nmf.doublet.combos, frac.overlap.max = 0.08, frac.overlap.diff.max = 0.15)
```

```
## Warning: Removed 103 rows containing missing values (geom_point).
```

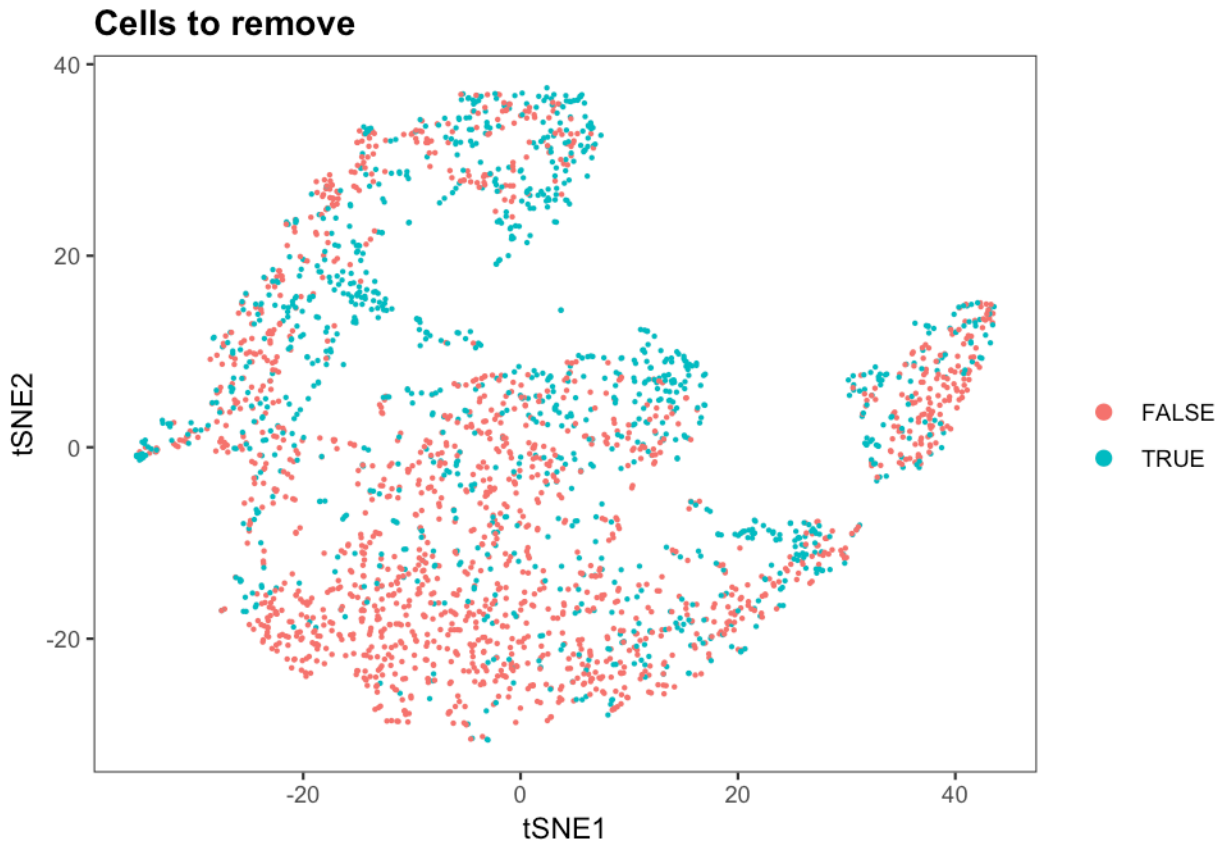



```
# Define doublet cells
nmf.doublets <- NMFDoubletsDetermineCells(hydra.ec, nmf.doublet.combos, module.expressed.thresh = 0.2,
    frac.overlap.max = 0.08, frac.overlap.diff.max = 0.15)  # 735 cells

# Visualize doublets
hydra.ec <- groupFromCells(hydra.ec, "nmf.doublets", nmf.doublets)
plotDim(hydra.ec, "nmf.doublets", plot.title = "Cells that express multiple exclusive ectodermal NMF modules")
```



Generate subsetted data

```
# All cropped cells
cropped.cells <- unique(c(knn.outliers, outliers.nmf.full.nonecto, nmf.doublets))
hydra.ec <- groupFromCells(hydra.ec, "cropped.cells", cropped.cells)
plotDim(hydra.ec, "cropped.cells", plot.title = "Cells to remove")
```



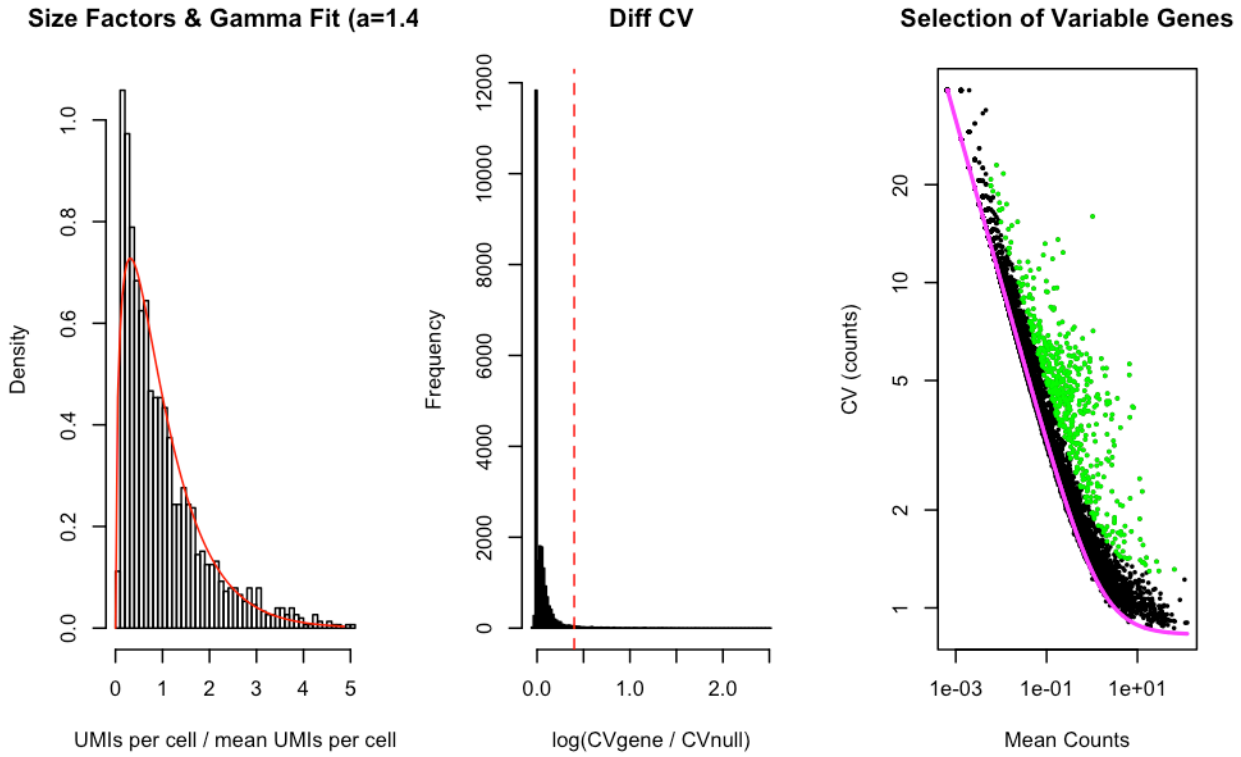

```
# Define new cell populations
cells.no.outliers <- setdiff(colnames(hydra.ec@logupx.data), cropped.cells)

# Subset the URD object
hydra.ec <- urdSubset(hydra.ec, cells.no.outliers)
```

Variable genes

We perform the analysis on genes that exhibit more variability than other genes of similar expression levels in order to focus on the genes that are most likely to encode biological information (as opposed to just technical). Since we've subsetted the object and also removed lots of contaminating cells, we should re-calculate the variable genes to focus just on the ectoderm.

```
# A more restrictive list of variable genes
hydra.ec <- findVariableGenes(object = hydra.ec, set.object.var.genes = T, diffCV.cutoff = 0.4,
    do.plot = T)  # 533 genes
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 10481 x values <= 0
## omitted from logarithmic plot
```



Graph Clustering

Generated several more fine-grained clusterings in order to have better options for choosing root and tip cells.

```
# Calculate PCA because doing spectral clustering
hydra.ec <- calcPCA(hydra.ec, mp.factor = 2)
```

```
## [1] "2018-11-01 14:15:49: Centering and scaling data."
## [1] "2018-11-01 14:15:50: Removing genes with no variation."
## [1] "2018-11-01 14:15:50: Calculating PCA."
## [1] "2018-11-01 14:15:52: Estimating significant PCs."
## [1] "Marchenko-Pastur eigenvalue null upper bound: 2.53436544074686"
## [1] "9 PCs have eigenvalues larger than 2 times null upper bound."
## [1] "Storing 18 PCs."
```

```
# Set random seed so clusters get the same names every time
set.seed(19)
# Graph clustering with various parameters
hydra.ec <- graphClustering(hydra.ec, num.nn = c(20, 30, 40, 50), do.jaccard = T,
    method = "Infomap")
hydra.ec <- graphClustering(hydra.ec, num.nn = c(5, 10, 15, 20, 30), do.jaccard = T,
    method = "Louvain")
hydra.ec <- graphClustering(hydra.ec, num.nn = c(6, 7, 8), do.jaccard = T, method = "Louvain")
# Record the clustering
pdf(paste0(main.path, "URD/Ectoderm/Clusters-Ectoderm-Infomap30.pdf"))
plotDim(hydra.ec, "Infomap-30", label.clusters = T, legend = F)
dev.off()
```

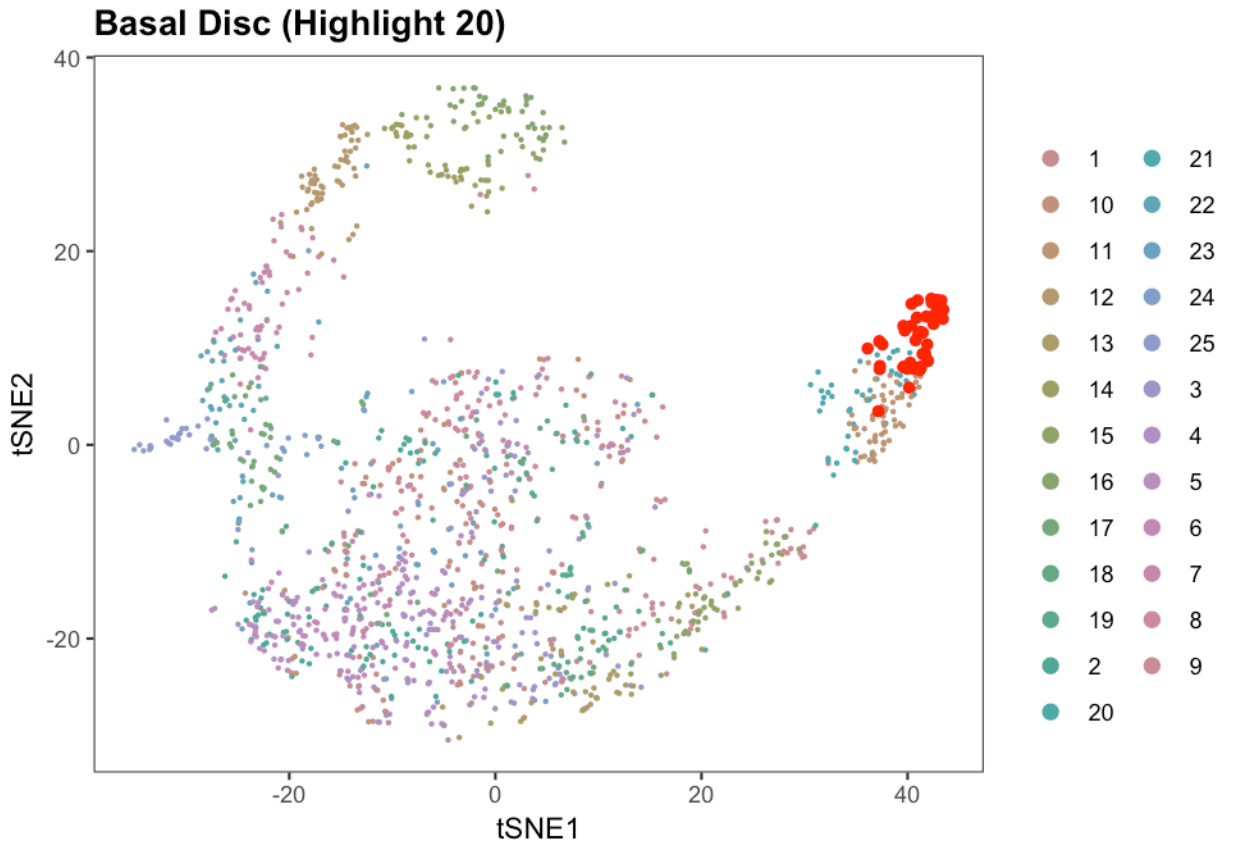
```
## quartz_off_screen
##                 2
```

Define root and tips

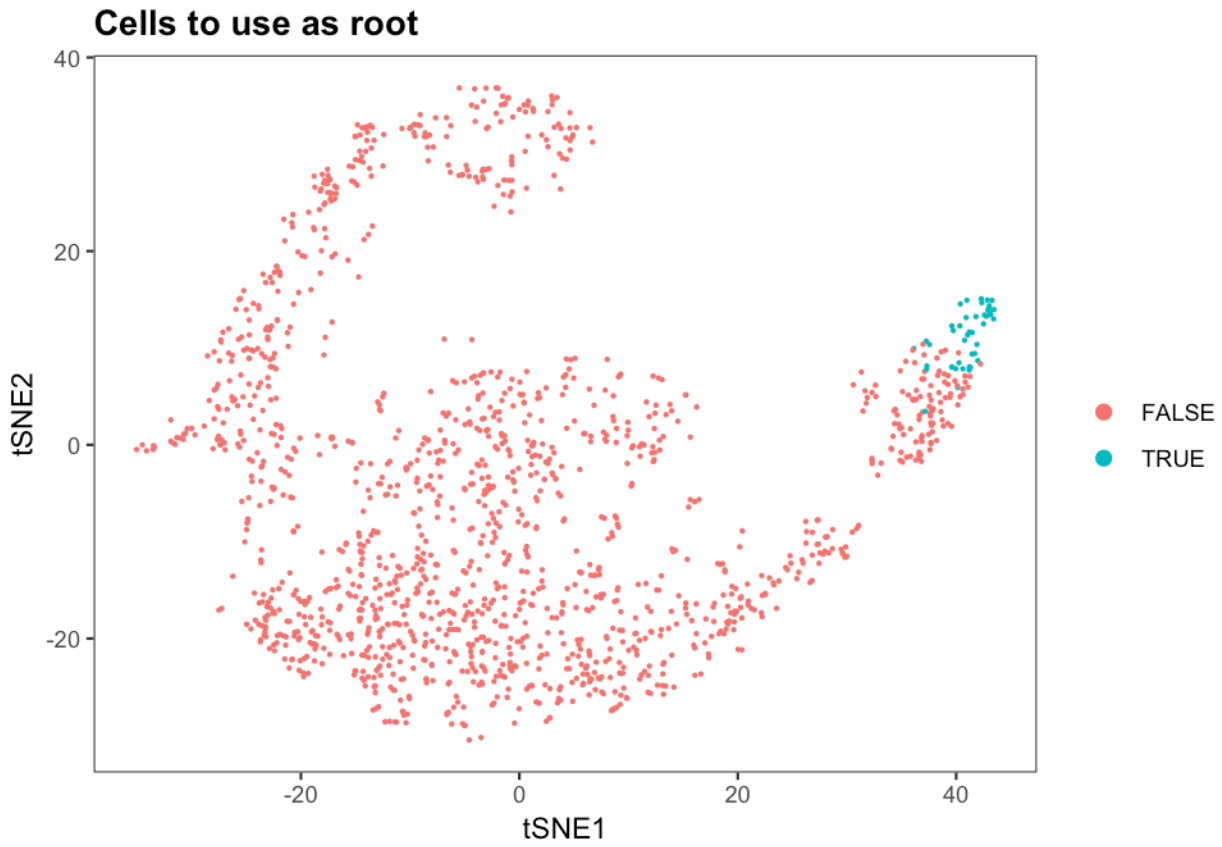
Root

Here, we are building spatial trajectories in the ectoderm, so going to use the basal disc (cluster 20) as the root and then head towards the hypostome and tentacles to build a braching trajectory.

```
# Using Infomap, 30 NN clustering to define roots and tips
plotDimHighlight(hydra.ec, clustering = "Infomap-30", cluster = "20", plot.title = "Basal Disc")  # basal disc
```



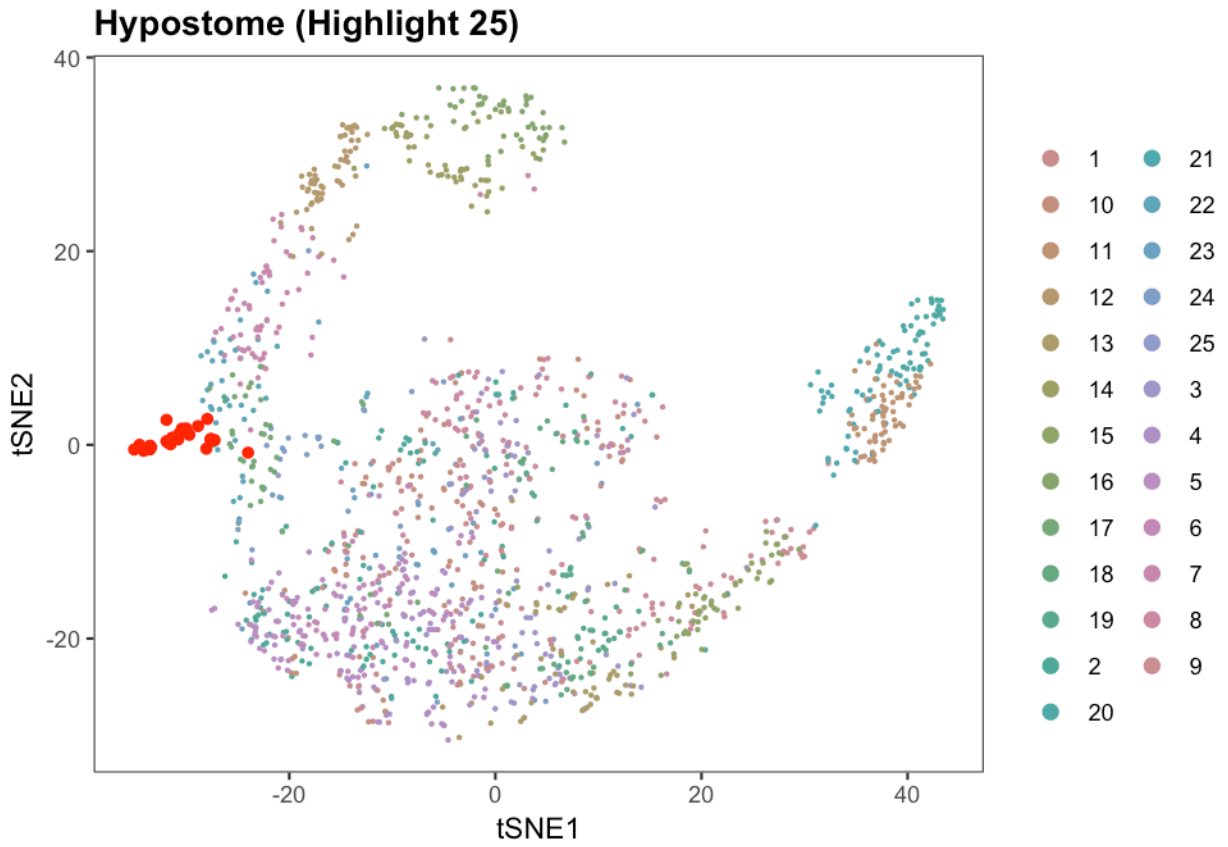

```
# Define root cells
root.cells <- cellsInCluster(hydra.ec, "Infomap-30", "20")
hydra.ec <- groupFromCells(hydra.ec, "root", root.cells)

plotDim(hydra.ec, "root", plot.title = "Cells to use as root")
```



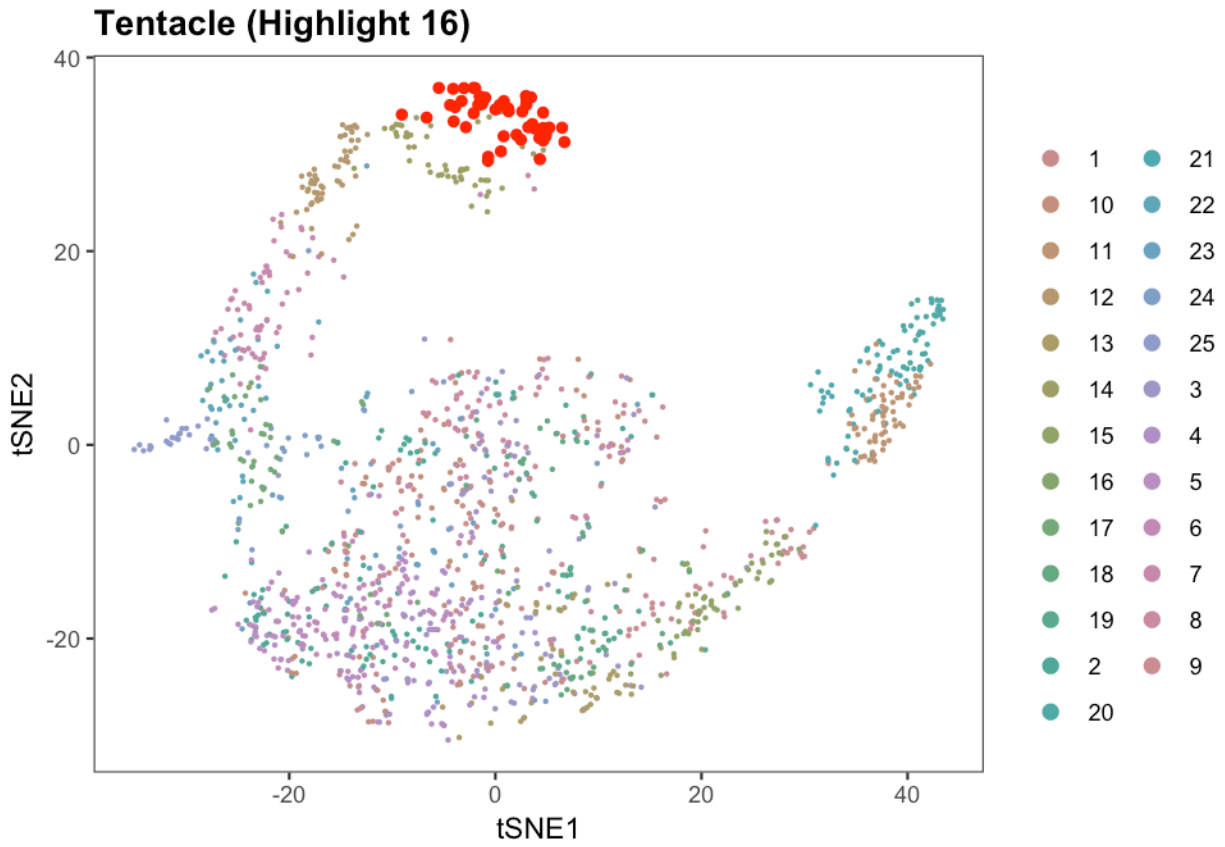
Tips

Using clusters in the head as tips: hypostome (25) and tentacle (16).

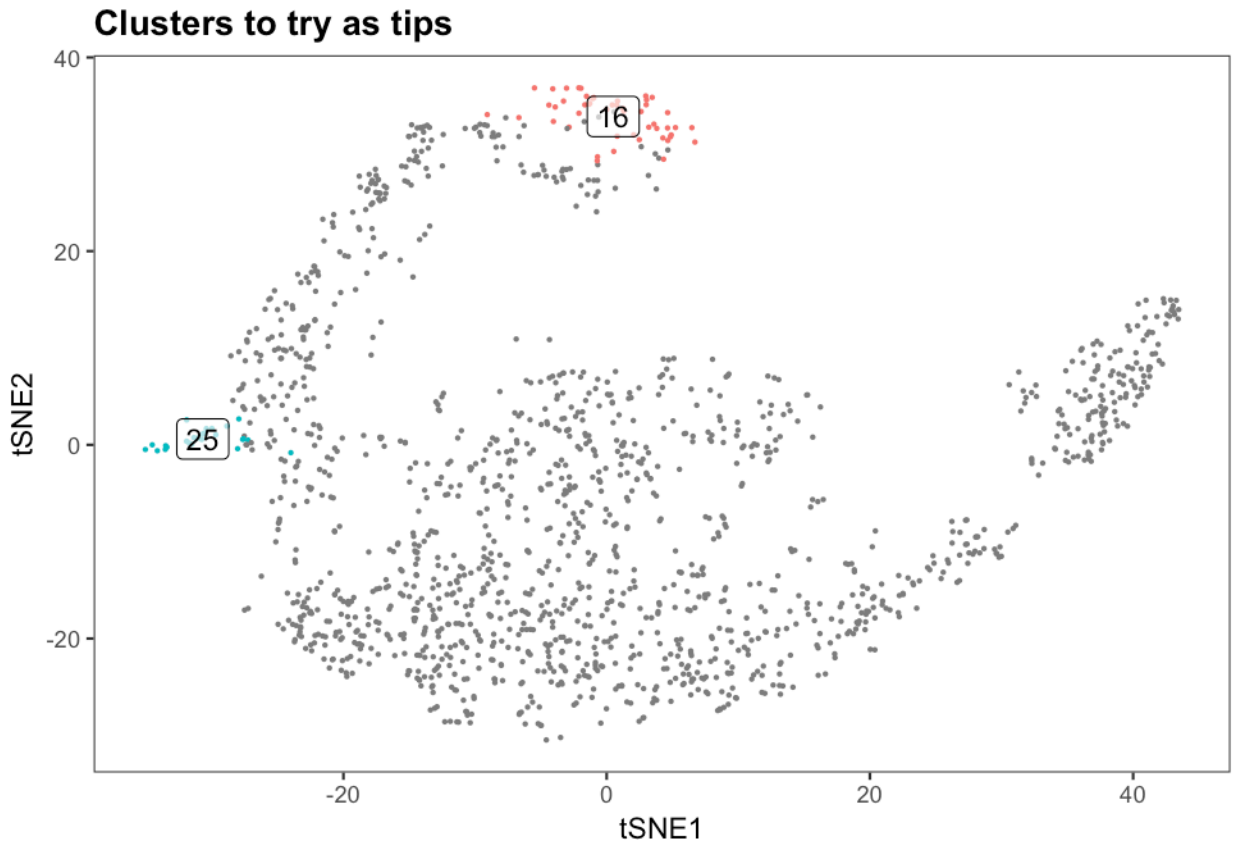
```
# Using Infomap, 30 NN clustering to define roots and tips
plotDimHighlight(hydra.ec, clustering = "Infomap-30", cluster = "25", plot.title = "Hypostome")  # hypostome
```




```
plotDimHighlight(hydra.ec, clustering = "Infomap-30", cluster = "16", plot.title = "Tentacle")  # battery
```




```
# Establish a tip clustering
hydra.ec@group.ids$tip.clusters <- NA
infomap30.clusters.use <- c("25", "16")
i30.tip.cells <- cellsInCluster(hydra.ec, "Infomap-30", infomap30.clusters.use)
hydra.ec@group.ids[i30.tip.cells, "tip.clusters"] <- hydra.ec@group.ids[i30.tip.cells,
    "Infomap-30"]

plotDim(hydra.ec, "tip.clusters", label.clusters = T, legend = F, plot.title = "Clusters to try as tips")
```



Diffusion maps

Calculate the transition probabilities

We calculated our diffusion map using 40 nearest neighbors (~ square root of number of cells in data), using *destiny*'s local sigma mode, which determines a sigma for each cell based on the distance to its nearest neighbors.

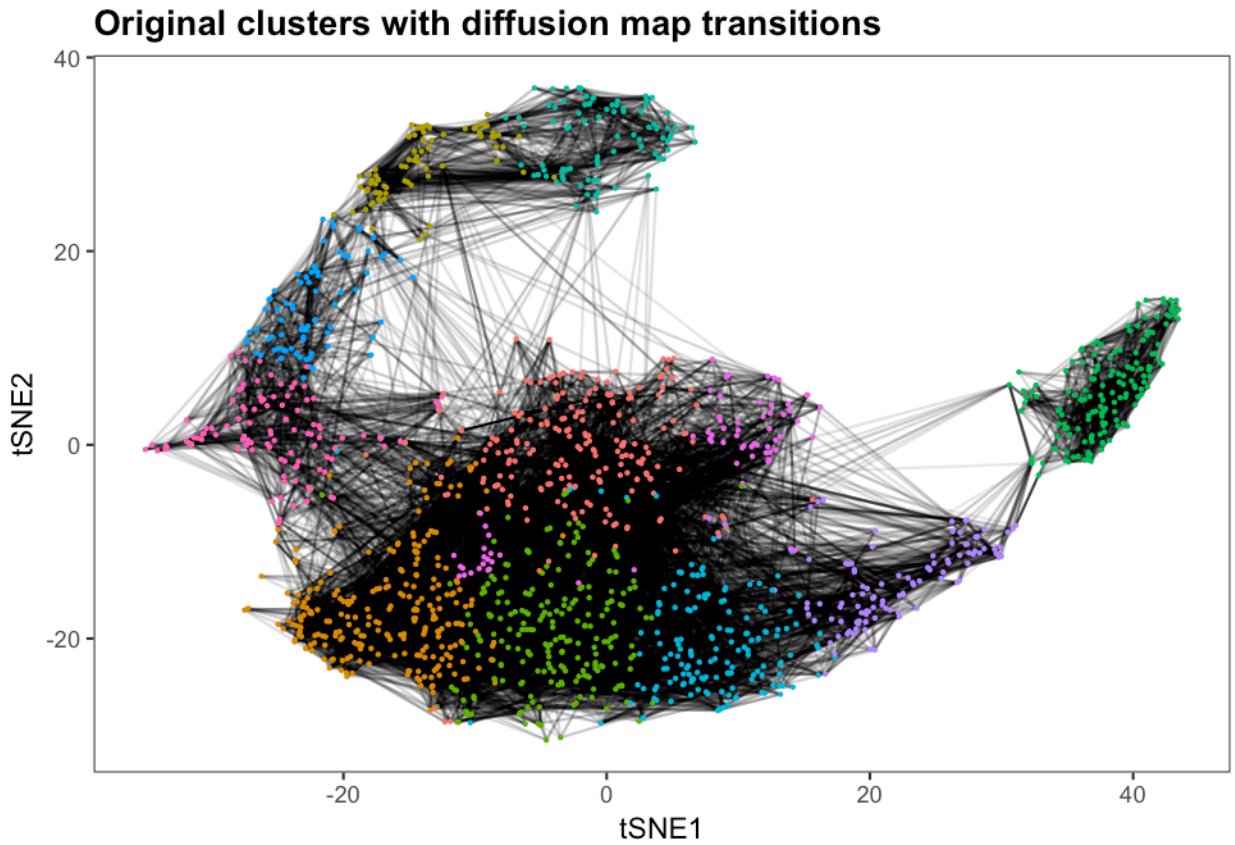
```
# Calculate diffusion map 39 NN = sqrt(1521 cells) Using local sigma
hydra.ec <- calcDM(hydra.ec, knn = 40, sigma.use = "local")
```

```
## [1] "Using local sigma."
```

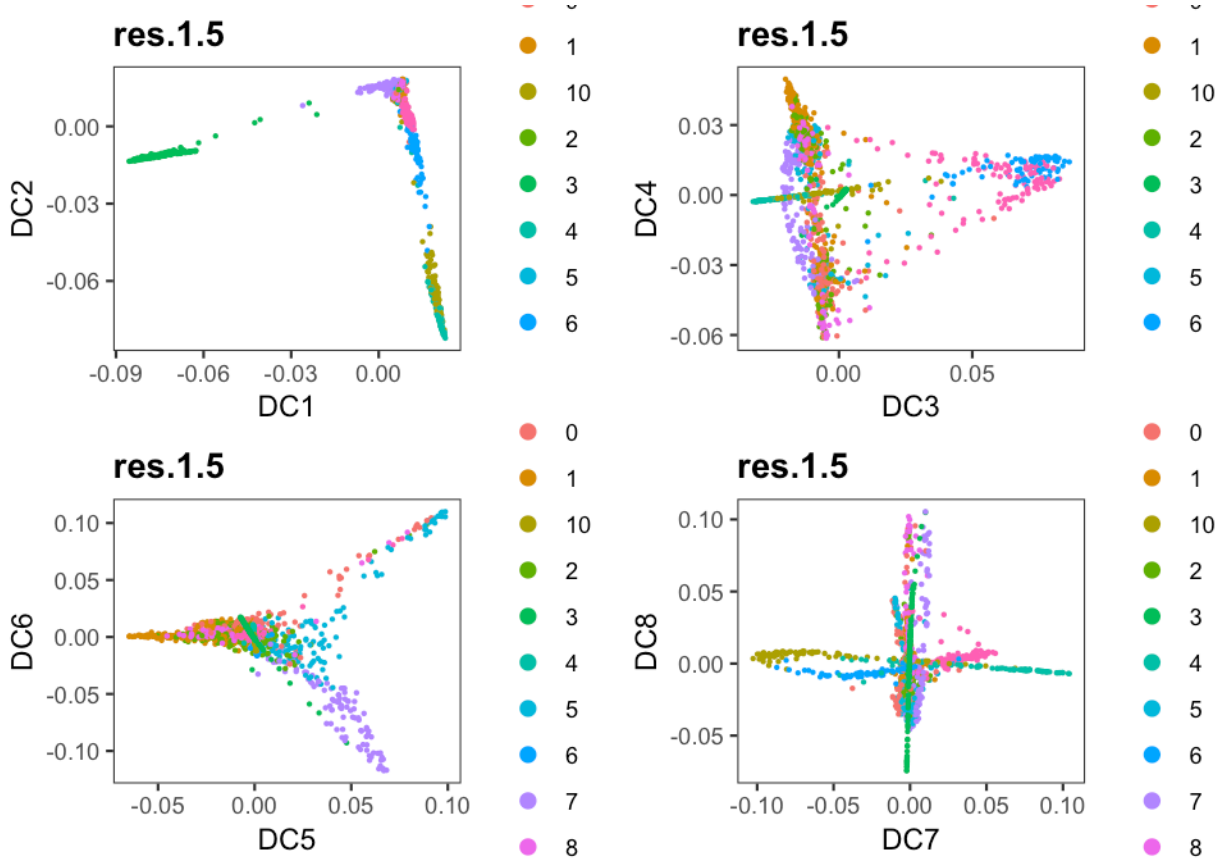
Evaluation of diffusion map

Diffusion map parameters seem good in that tentacle, hypostome, and foot are all nicely represented as spikes in the diffusion map and the connections in the diffusion map seem to do a good job of evenly connecting cells in the body column.

```
# Test how the connections are in the data
plotDim(hydra.ec, "res.1.5", transitions.plot = 10000, plot.title = "Original clusters with diffusion map transitions",
    legend = F)
```




```
# These look good. Transitions from the basal disc to the peduncle seem rare, but
# that is expected (as the transition is thought to be very rapid).

# The hypostome, tentacle, and foot are well-represented as tips in the diffusion
# map, so parameters are probably fairly good.
plotDimArray(hydra.ec, label = "res.1.5", reduction.use = "dm", dims.to.plot = 1:8)
```



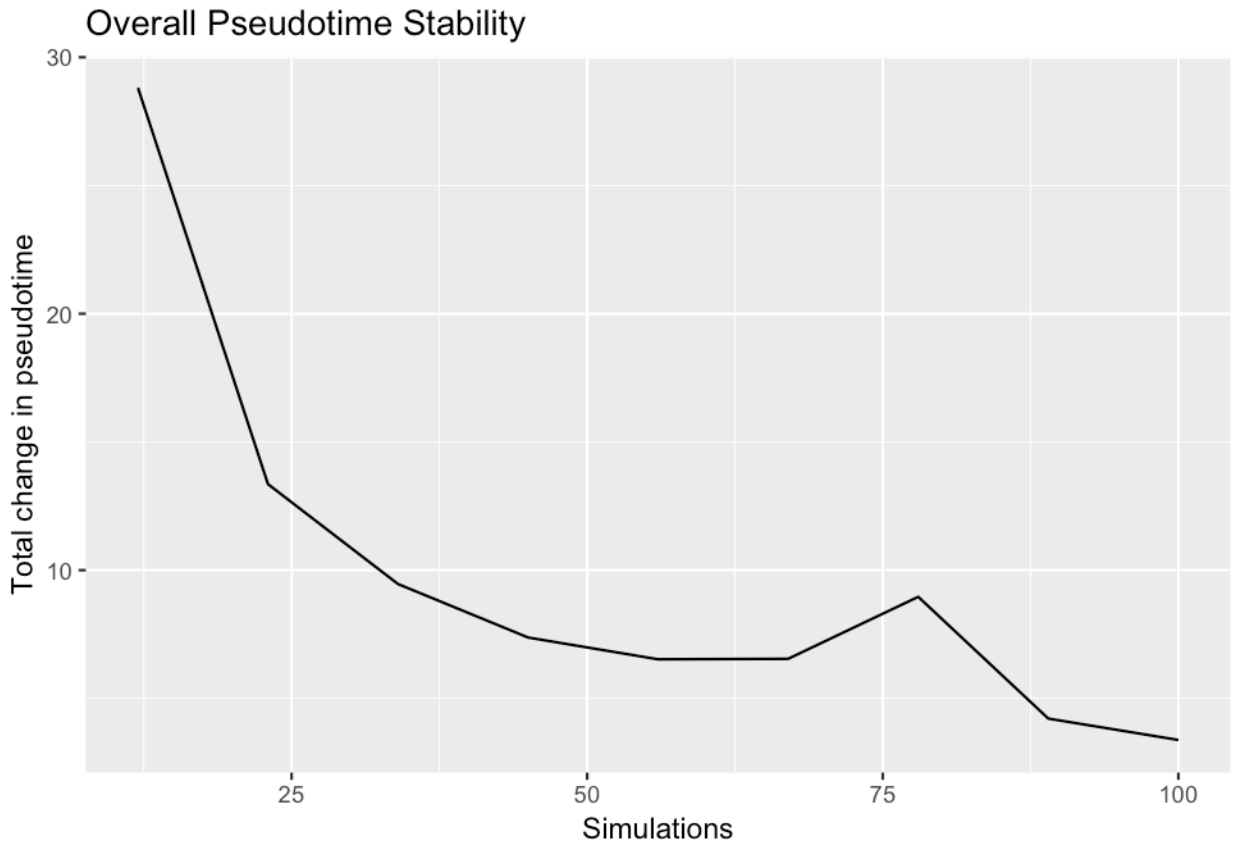
Pseudotime

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)
```

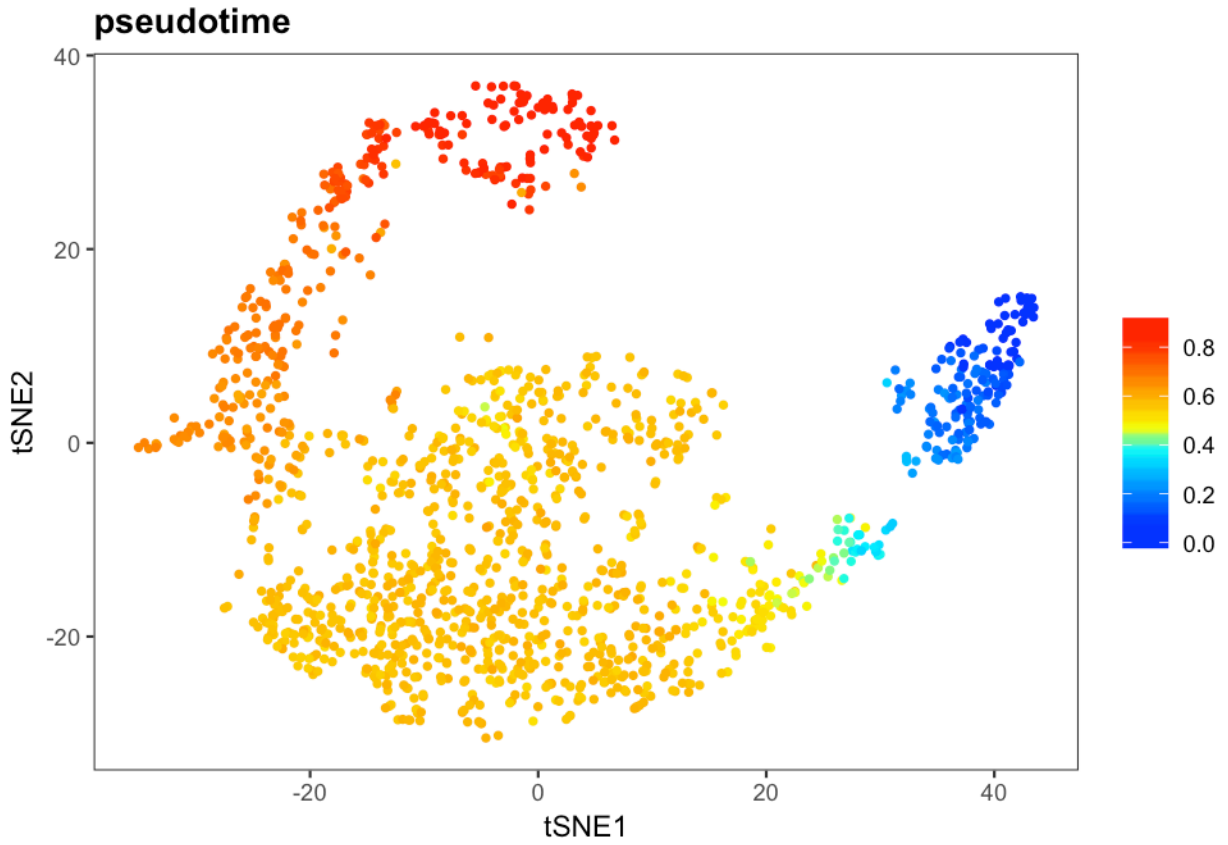
We the process the random simulations to convert them to a 0-1 range pseudotime and verify that enough simulations have been performed.

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

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



Pseudotime was calculated starting at the basal disc. It has a smooth temporal ordering towards the head.

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



Biased random walks

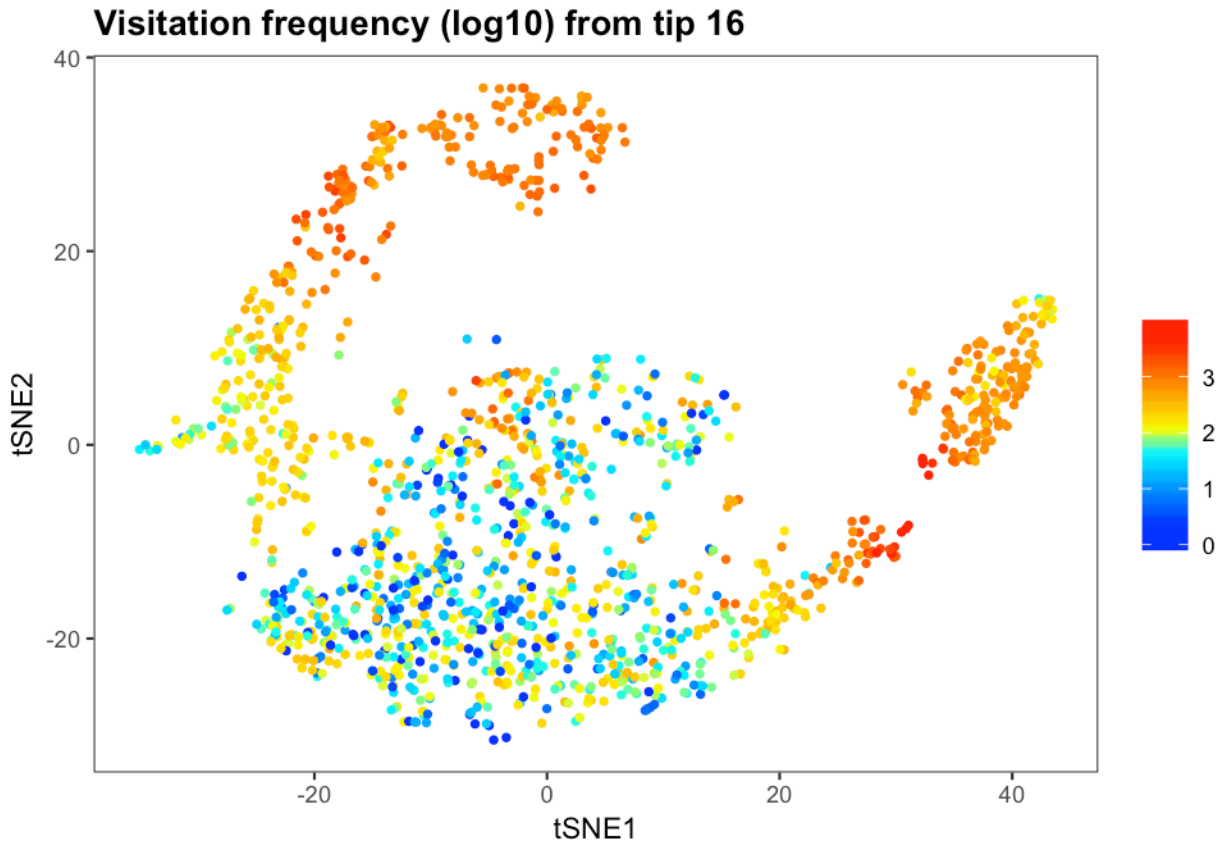
We tried biased random walks from a tip in the hypostome and tentacle. The random walk parameters were fairly standard (optimal 20 cells forward, maximum 40 cells back). The random simulations are non-deterministic, so we load our previous result, but the commands run previous were:

```
# Bias the transition probabilities by cellular pseudotime
pseudotime.logistic <- pseudotimeDetermineLogistic(hydra.ec, pseudotime = "pseudotime",
    optimal.cells.forward = 20, max.cells.back = 40, pseudotime.direction = "<")
tm.biased <- as.matrix(pseudotimeWeightTransitionMatrix(hydra.ec, pseudotime = "pseudotime",
    logistic.params = pseudotime.logistic, pseudotime.direction = "<"))

# Simulate biased random walks and load them into the object
walks.ec <- simulateRandomWalksFromTips(hydra.ec, "tip.clusters", root.cells = root.cells,
    transition.matrix = tm.biased, n.per.tip = 20000, root.visits = 1)

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

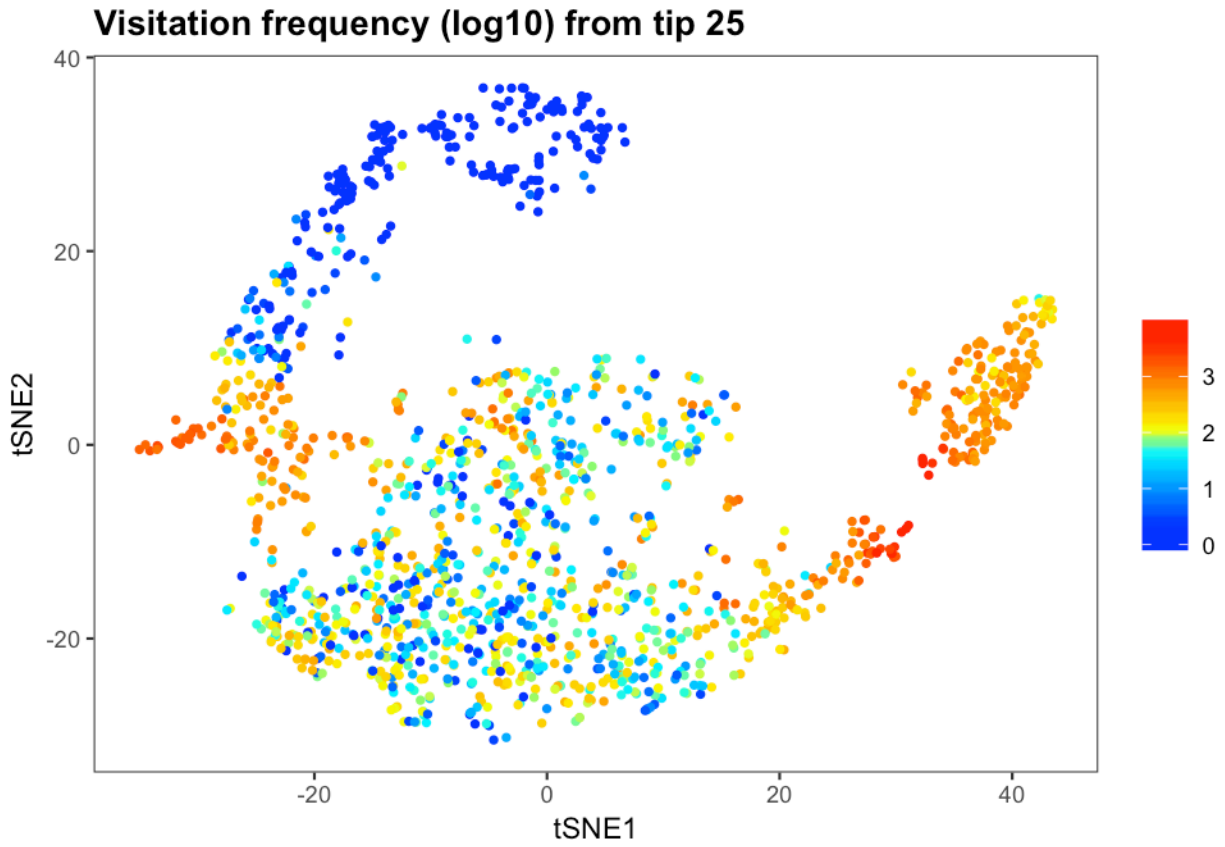
Evaluation of random walks

The random walks from the two tips did a good job of visiting the majority of the data.

```
# Plot visitation from individual tips
plotDim(hydra.ec, "visitfreq.log.16", plot.title = "Visitation frequency (log10) from tip 16")
```




```
plotDim(hydra.ec, "visitfreq.log.25", plot.title = "Visitation frequency (log10) from tip 25")
```



Build the tree

Since we have walks with good visitation, we can now build a tree to describe the spatial localization of cells in the ectoderm, branching in the head into the hypostome and tentacle.

```
# Load the tips into the tree
hydra.ec.tree <- loadTipCells(hydra.ec, "tip.clusters")

# Build the tree
hydra.ec.tree <- buildTree(hydra.ec.tree, pseudotime = "pseudotime", divergence.method = "preference",
    save.all.breakpoint.info = T, tips.use = c("16", "25"), cells.per.pseudotime.bin = 25,
    bins.per.pseudotime.window = 8, p.thresh = 0.05, min.cells.per.segment = 10,
    save.breakpoint.plots = NULL)
```

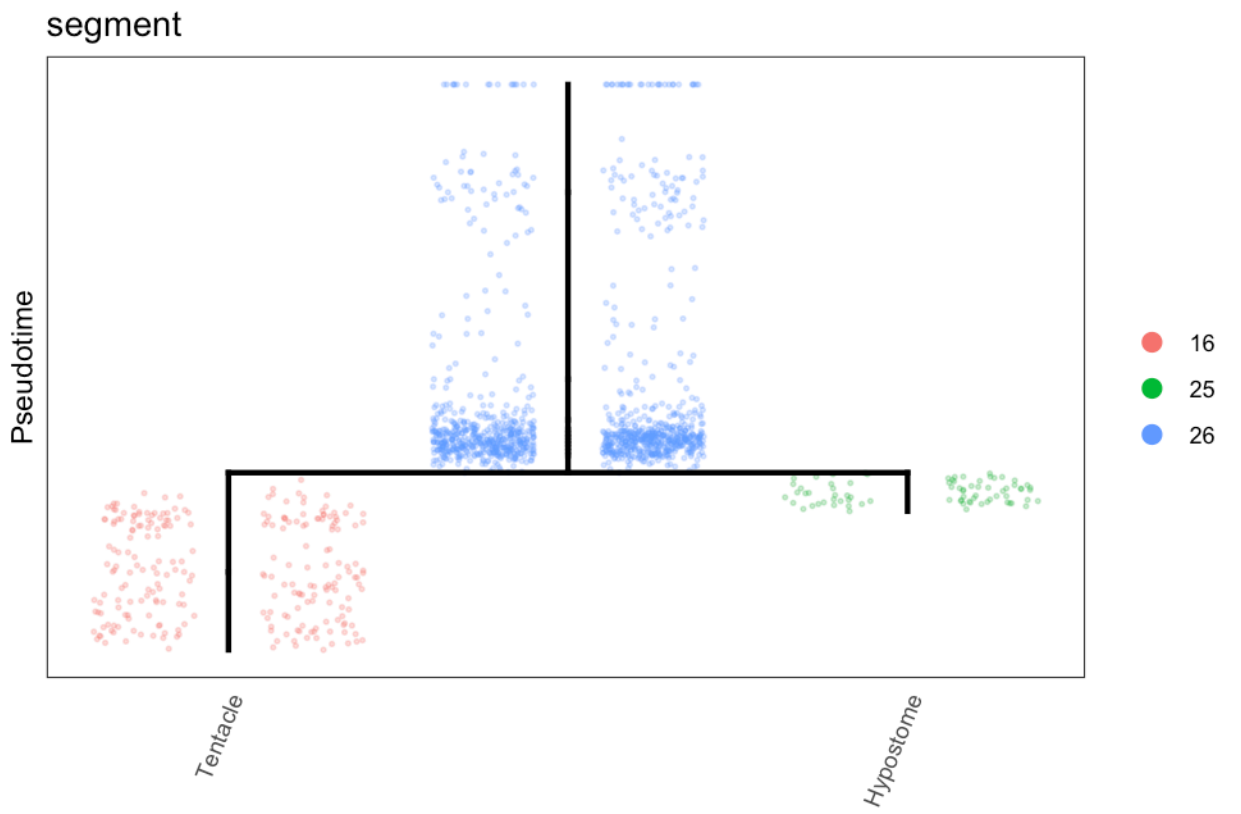
```
## [1] "Calculating divergence between 16 and 25 (Pseudotime 0 to 0.677)"
## [1] "Joining segments 16 and 25 at pseudotime 0.616 to create segment 26"
## [1] "Assigning cells to segments."
## Warning in assignCellsToSegments(object, pseudotime, verbose): 23 cells
## were not visited by a branch that exists at their pseudotime and were not
## assigned.
## [1] "Collapsing short segments."
## [1] "Removing singleton segments."
## [1] "Reassigning cells to segments."
## Warning in assignCellsToSegments(object, pseudotime, verbose): 23 cells
## were not visited by a branch that exists at their pseudotime and were not
## assigned.
## [1] "Assigning cells to nodes."
## [1] "Laying out tree."
## [1] "Adding cells to tree."
```

```
# Name the segments to reflect their tissue (identified by external knowledge of
# gene expression)
```

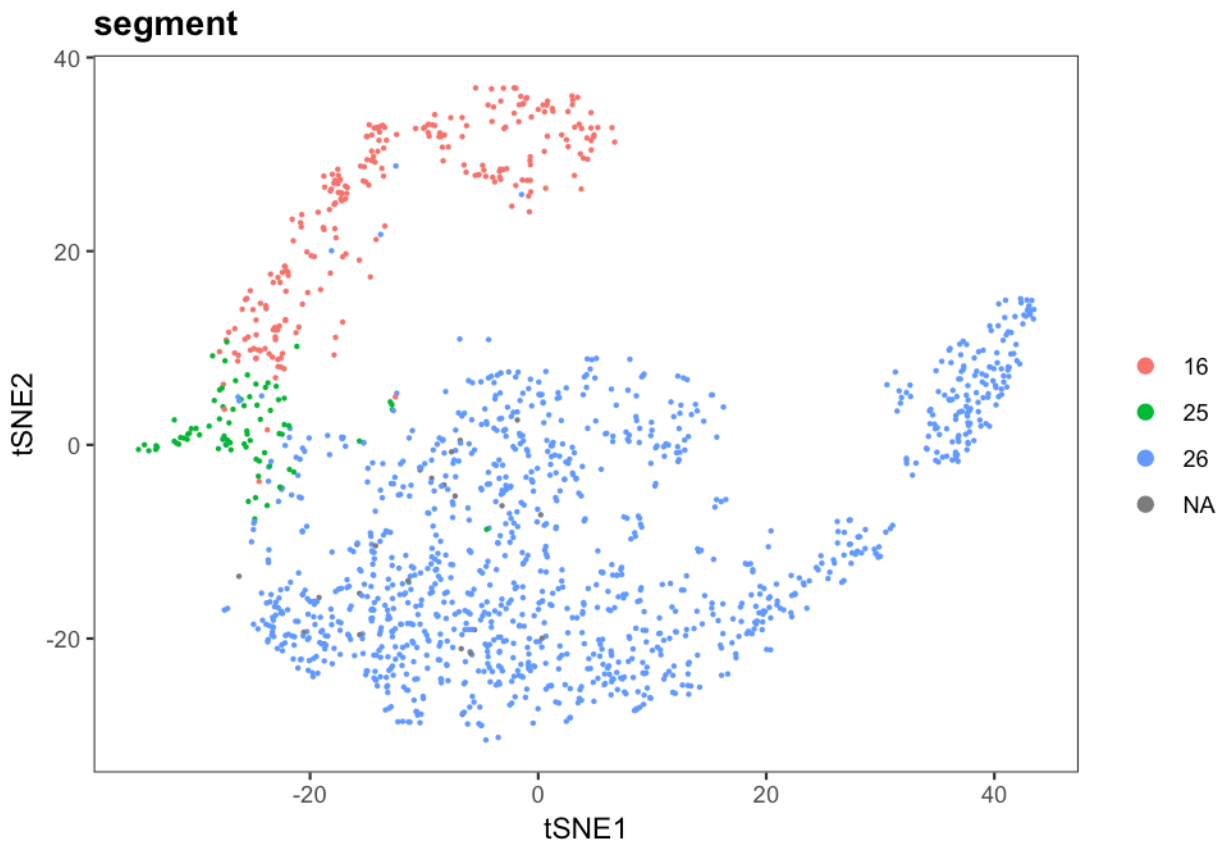
```
hydra.ec.tree <- nameSegments(object = hydra.ec.tree, segments = c("25", "16"), segment.names = c("Hypostome",
    "Tentacle"), short.names = c("Hyp", "Tent"))
```

Visualize the tree

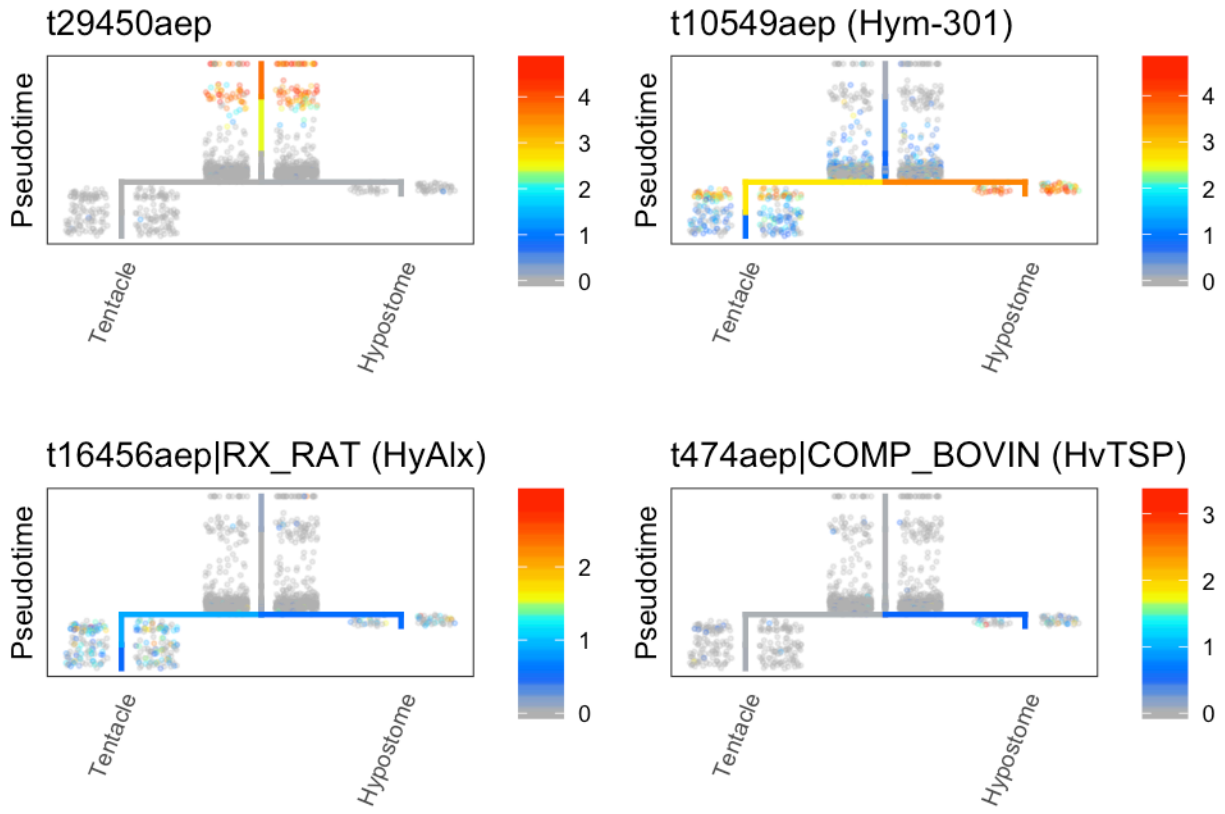
```
plotTree(hydra.ec.tree, "segment")
```




```
plotDim(hydra.ec.tree, "segment")
```




```
gridExtra::grid.arrange(grobs = list(plotTree(hydra.ec.tree, "t29450aep", title = "t29450aep"),
    plotTree(hydra.ec.tree, "t10549aep", title = "t10549aep (Hym-301)"), plotTree(hydra.ec.tree,
        "t16456aep|RX_RAT", title = "t16456aep|RX_RAT (HyAlx)"), plotTree(hydra.ec.tree,
        "t474aep|COMP_BOVIN", title = "t474aep|COMP_BOVIN (HvTSP)")))
```



Finding spatially varying genes

Since in this trajectory, "pseudotime" is really a proxy for spatial location (foot to head), we can find the spatially varying genes by finding those that vary in pseudotime. We group cells 5 at a time and calculate a spline curve that fits the mean expression (vs. pseudotime) of each group of 5 cells. We then consider those genes spatially varying that are: (1) expressed (present in at least 1% of cells, mean expression > 0.5 in at least one group of 5 cells, spline curve > 0.5 at some point), (2) vary significantly (their spline curve varies at least 33%), (3) are well fit (noise is usually poorly fit, here we threshold on the sum of squared residuals).

Calculate spline curves

```
# Consider all genes expressed in 1% of endodermal cells
frac.exp <- Matrix::rowSums(hydra.ec.tree@logupx.data > 0)/ncol(hydra.ec.tree@logupx.data)
ecto.genes <- names(frac.exp)[which(frac.exp > 0.01)]

# Identify cell populations: basal disc, body column + tentacle,
basal.disc.cells <- cellsInCluster(hydra.ec.tree, "res.1.5", "3")
body.column.and.battery.cells <- setdiff(cellsInCluster(hydra.ec.tree, "segment",
    c("16", "26")), basal.disc.cells)
hypo.trajectory.cells <- c(cellsInCluster(hydra.ec.tree, "segment", "25"), cellsInCluster(hydra.ec.tree,
    "segment", "26")[order(hydra.ec.tree@pseudotime[cellsInCluster(hydra.ec.tree,
    "segment", "26"), "pseudotime"], decreasing = T)[1:250]]  # Include the 250 'oldest' cells from before the branchpoint
)

# Calculate smoothed spline fits of the expression within each population
basal.disc.spline.single <- geneSmoothFit(hydra.ec.tree, method = "spline", pseudotime = "pseudotime",
    cells = basal.disc.cells, genes = ecto.genes, moving.window = 1, cells.per.window = 5,
    spar = 0.875)  # Basal disc only
```

```
## [1] "2018-11-01 14:17:09: Calculating moving window expression."
```

```
## [1] "2018-11-01 14:17:17: Generating un-scaled fits."
## [1] "2018-11-01 14:17:31: Generating scaled fits."
## [1] "2018-11-01 14:17:46: Reducing mean expression data to same dimensions as spline fits."
```

```
battery.spline.single <- geneSmoothFit(hydra.ec.tree, method = "spline", pseudotime = "pseudotime",
    cells = body.column.and.battery.cells, genes = ecto.genes, moving.window = 1,
    cells.per.window = 5, spar = 0.875)  # Body column and tentacles (which contain battery cells)
```

```
## [1] "2018-11-01 14:17:46: Calculating moving window expression."
## [1] "2018-11-01 14:18:47: Generating un-scaled fits."
## [1] "2018-11-01 14:19:26: Generating scaled fits."
## [1] "2018-11-01 14:20:02: Reducing mean expression data to same dimensions as spline fits."
```

```
head.spline.single <- geneSmoothFit(hydra.ec.tree, method = "spline", pseudotime = "pseudotime",
    cells = hypo.trajectory.cells, genes = ecto.genes, moving.window = 1, cells.per.window = 5,
    spar = 0.875)  # Hypostome
```

```
## [1] "2018-11-01 14:20:18: Calculating moving window expression."
## [1] "2018-11-01 14:20:32: Generating un-scaled fits."
## [1] "2018-11-01 14:20:46: Generating scaled fits."
## [1] "2018-11-01 14:21:01: Reducing mean expression data to same dimensions as spline fits."
```

```
## Combine pieces of spline fits into a single list for multi-plotting Identify
## the pseudotime of the branchpoint / breakpoint between basal disc and body
## column
pt.crop <- as.numeric(unlist(hydra.ec.tree@tree$segment.pseudotime.limits)[1])
basal.crop <- max(hydra.ec.tree@pseudotime[cellsInCluster(hydra.ec.tree, "res.1.5",
    "3"), "pseudotime"])
# Crop according to the pseudotime of the branchpoint
body.only.spline.single <- cropSmoothFit(battery.spline.single, pt.max = pt.crop)
battery.only.spline.single <- cropSmoothFit(battery.spline.single, pt.min = pt.crop)
head.only.spline.single <- cropSmoothFit(head.spline.single, pt.min = pt.crop)
# Combine into a list; Names in the plots are determined by names of the smooth
# objects in the list
splines <- list(basal.disc.spline.single, body.only.spline.single, battery.only.spline.single,
    head.only.spline.single)
names(splines) <- c("Basal Disc", "Body Column", "Tentacle", "Hypostome")
```

Calculate varying genes

Since there is such a dramatic transcriptional change between the basal disc and the body column, going to calculate a separate spline curve for the basal disc; also calculate two spline curves for body column to tentacle and body column to hypostome.

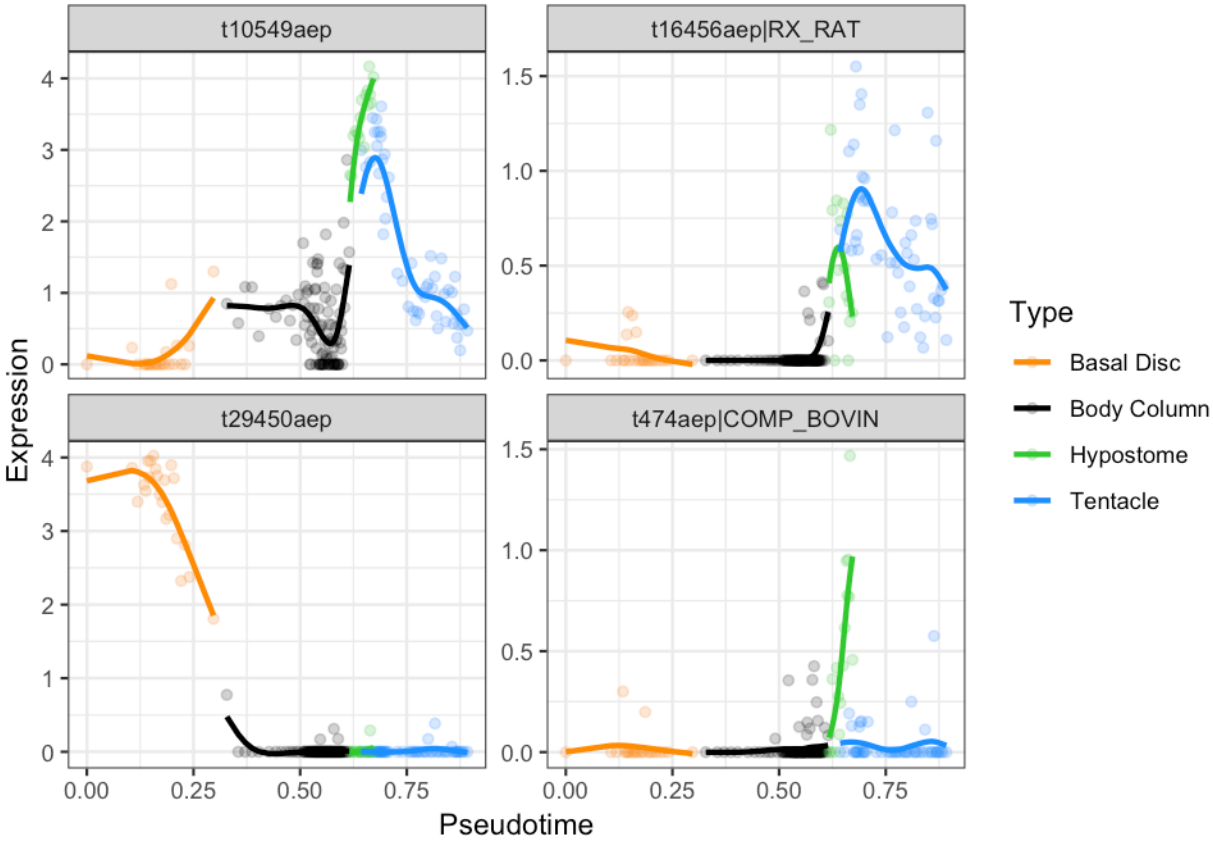
```
# Which genes have a spline curve that crosses 0.5?
basal.disc.max.mean.spline <- apply(splines$`Basal Disc`$mean.smooth, 1, max)
body.max.mean.spline <- apply(splines$`Body Column`$mean.smooth, 1, max)
tentacle.max.mean.spline <- apply(splines$Tentacle$mean.smooth, 1, max)
hypo.max.mean.spline <- apply(splines$Hypostome$mean.smooth, 1, max)
tentacle.genes.wellexpressed <- names(which(tentacle.max.mean.spline > 0.5 | basal.disc.max.mean.spline >
    0.5 | body.max.mean.spline > 0.5))
hypo.genes.wellexpressed <- names(which(hypo.max.mean.spline > 0.5 | basal.disc.max.mean.spline >
    0.5 | body.max.mean.spline > 0.5))

# Combine into heatmaps of basal-body-tentacle and basal-body-hypostome
tentacle.spline <- combineSmoothFit(splines, fits.use = c("Basal Disc", "Body Column",
    "Tentacle"))
hypo.spline <- combineSmoothFit(splines, fits.use = c("Basal Disc", "Body Column",
    "Hypostome"))

# Which genes change in their scaled log2 mean expression value by at least 33%?
tentacle.change.scale <- apply(tentacle.spline$scaled.smooth, 1, function(x) diff(range(x)))
hypo.change.scale <- apply(hypo.spline$scaled.smooth, 1, function(x) diff(range(x)))
tentacle.genes.scale <- names(which(tentacle.change.scale >= 0.33))
hypo.genes.scale <- names(which(hypo.change.scale >= 0.33))

# Which genes are well fit by the spline curves? (Noise is usually poorly fit by
# a curve)
tentacle.spline.fit <- apply(tentacle.spline$scaled.smooth - tentacle.spline$scaled.expression.red,
    1, function(i) sum(i^2))
hypo.spline.fit <- apply(hypo.spline$scaled.smooth - hypo.spline$scaled.expression.red,
    1, function(i) sum(i^2))
tentacle.spline.fit.norm <- tentacle.spline.fit/ncol(tentacle.spline$scaled.expression.red)
```

```
hypo.spline.fit.norm <- hypo.spline.fit/ncol(hypo.spline$scaled.expression.red)
tentacle.wellfit <- names(which(tentacle.spline.fit.norm <= 0.02))
hypo.wellfit <- names(which(hypo.spline.fit.norm <= 0.02))

# Take the intersection of those genes and use them in the heatmap & analysis
tentacle.genes <- intersect(intersect(tentacle.genes.scale, tentacle.wellfit), tentacle.genes.wellexpressed)
hypo.genes <- intersect(intersect(hypo.genes.scale, hypo.wellfit), hypo.genes.wellexpressed)
```

Spline plots

We can then plot the expression of genes in their splines.

```
ecto.plot.genes <- c("t29450aep", "t10549aep", "t16456aep|RX_RAT", "t474aep|COMP_BOVIN")

plotSmoothFitMultiCascade(splines, genes = ecto.plot.genes, scaled = F, colors = c(`Basal Disc` = "#FF8C00",
    `Body Column` = "#000000", Tentacle = "#1E90FF", Hypostome = "#32CD32"))
```



Heatmaps

Finally, we can make heatmaps of the expression of all of the genes that we found to vary spatially. These we save as PDF output because they do not perform well inside of R Markdown.

```
# Heatmap Basics
cols <- (scales::gradient_n_pal(RColorBrewer::brewer.pal(9, "YlOrRd")))(seq(0, 1,
    length.out = 50))

# Get the scaled data, z-score it, and set values <0 to 0, the hierarchical
# cluster.  This emphasizes clustering on regions of peak expression for a gene.
se <- tentacle.spline$scaled.expression[tentacle.genes, ]
se.sd <- apply(se, 1, sd)
se.mean <- apply(se, 1, mean)
se.z <- sweep(sweep(se, 1, se.mean, "-"), 1, se.sd, "/")
se.z[se.z < 0] <- 0
```

```
h.sez <- as.dendrogram(hclust(dist(se.z)))

# Find a 'peak pseudotime' for each gene by taking the weighted average of each
# window's pseudotime, weighted by the expression z-score (>0) within it Can use
# this to reorder the dendrogram to try to put it in pseudotime order.
pt.wm <- apply(se.z, 1, function(w) {
    weighted.mean(x = as.numeric(colnames(se.z)), w = w)
})

# Generate the actual heatmap and save as a PDF.
font.size <- 0.16
pdf(paste0(main.path, "URD/Ectoderm/Ectoderm-Varying-Tentacle.pdf"), width = 17,
    height = 22)
gplots::heatmap.2(as.matrix(se), Rowv = reorder(h.sez, max(pt.wm) - pt.wm, agglo.FUN = median),
    Colv = F, dendrogram = "none", col = cols, trace = "none", density.info = "none",
    key = F, cexCol = 0.8, cexRow = font.size, margins = c(5, 8), lwid = c(0.3, 4),
    lhei = c(0.4, 4), labCol = NA)
title("Ectoderm: Foot to Tentacle", line = -1, adj = 0.48, cex.main = 4)
dev.off()
```

```
## quartz_off_screen
##                 2
```

```
# Heatmap Basics
cols <- (scales::gradient_n_pal(RColorBrewer::brewer.pal(9, "YlOrRd")))(seq(0, 1,
    length.out = 50))

# Get the scaled data, z-score it, and set values <0 to 0, the hierarchical
# cluster.  This emphasizes clustering on regions of peak expression for a gene.
se <- hypo.spline$scaled.expression[hypo.genes, ]
se.sd <- apply(se, 1, sd)
se.mean <- apply(se, 1, mean)
se.z <- sweep(sweep(se, 1, se.mean, "-"), 1, se.sd, "/")
se.z[se.z < 0] <- 0
h.sez <- as.dendrogram(hclust(dist(se.z)))

# Find a 'peak pseudotime' for each gene by taking the weighted average of each
# window's pseudotime, weighted by the expression z-score (>0) within it Can use
# this to reorder the dendrogram to try to put it in pseudotime order.
pt.wm <- apply(se.z, 1, function(w) {
    weighted.mean(x = as.numeric(colnames(se.z)), w = w)
})

# Generate the actual heatmap and save as a PDF.
font.size <- 0.25
pdf(paste0(main.path, "URD/Ectoderm/Ectoderm-Varying-Hypostome.pdf"), width = 17,
    height = 22)
gplots::heatmap.2(as.matrix(se), Rowv = reorder(h.sez, max(pt.wm) - pt.wm, agglo.FUN = median),
    Colv = F, dendrogram = "none", col = cols, trace = "none", density.info = "none",
    key = F, cexCol = 0.8, cexRow = font.size, margins = c(5, 8), lwid = c(0.3, 4),
    lhei = c(0.4, 4), labCol = NA)
title("Ectoderm: Foot to Hypostome", line = -1, adj = 0.48, cex.main = 4)
dev.off()
```

```
## quartz_off_screen
##                 2
```

Save results

```
write(tentacle.genes, file = paste0(main.path, "URD/Ectoderm/Genes-Ectoderm-Varying-Tentacle.txt"))
write(hypo.genes, file = paste0(main.path, "URD/Ectoderm/Genes-Ectoderm-Varying-Hypostome.txt"))
saveRDS(splines, file = paste0(main.path, "URD/Ectoderm/Splines-Ectoderm.rds"))
saveRDS(hydra.ec.tree, file = paste0(main.path, "URD/Ectoderm/Hydra_URD_Ectoderm.rds"))
```