SA13: Hydra URD Spumous Mucous

Jeff Farrell

October 1, 2018

```r
# 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
```

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

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

Load subsetted URD data for this lineage

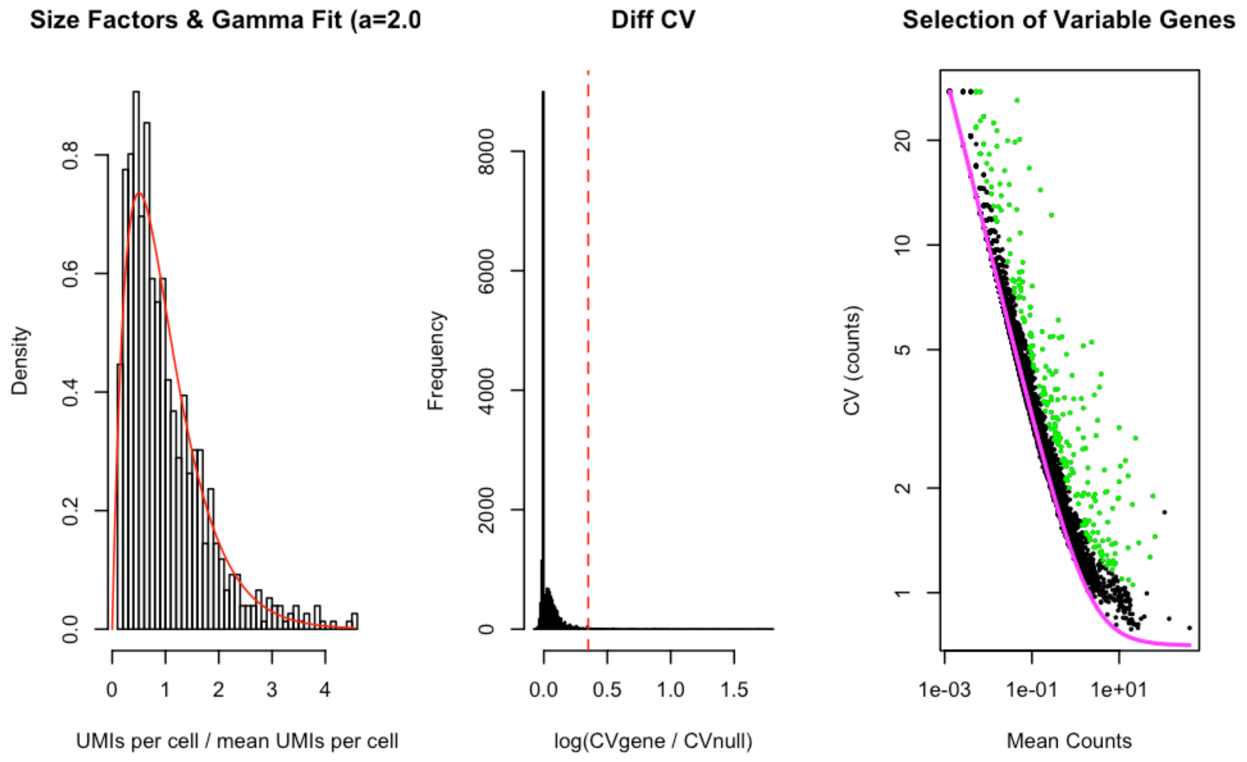
```r
hydra.spumous <- readRDS(file = paste0(main.path, "objects/Hydra_URD_Input_Spumous.rds"))

# Get rid of a few loser cells that are way on the outskirts of the tSNE
loser.cells <- c("02-C0_GATGTCTTGCTT", "12-N1_TTCACTTGCAGT", "12-N2_ACAGCATGGCAT",
    "11-BU_CAACTTCCGGAT", "11-BU_GTCGGCGGACGA")
hydra.spumous <- urdSubset(hydra.spumous, cells.keep = setdiff(colnames(hydra.spumous@logupx.data),
    loser.cells))
```

Variable genes

Since we've subsetted, it's best to calculate a list of variable genes specific to this lineage.

```r
# Determine variable genes (251 genes, 192 were variable across whole IC lineage)
hydra.spumous <- findVariableGenes(object = hydra.spumous, set.object.var.genes = T,
    diffCV.cutoff = 0.35, do.plot = T)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 15309 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.

```
# Set random seed so clusters get the same names every time
set.seed(19)
# Graph clustering with various parameters
hydra.spumous <- graphClustering(hydra.spumous, num.nn = c(20, 30, 40, 50), do.jaccard = T,
    method = "Infomap")
hydra.spumous <- graphClustering(hydra.spumous, num.nn = c(5, 10, 15, 20, 30), do.jaccard = T,
    method = "Louvain")
hydra.spumous <- graphClustering(hydra.spumous, num.nn = c(6, 7, 8), do.jaccard = T,
    method = "Louvain")
# Record the clustering
pdf(paste0(main.path, "URD/SpumousMucous/Clusters-SpumousMucous-Infomap20.pdf"))
plotDim(hydra.spumous, "Infomap-20", label.clusters = T, legend = F)
dev.off()
```

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

Diffusion maps

Calculate the transition probabilities

Calculate a diffusion map specific to the spumous mucous trajectory. In these very specific datasets (i.e. those cropped to a single lineage), using more nearest neighbors than the square root of the data seems to work better, probably because there are fewer cell states to consider. (Here we use 75 NNs, versus prediction of 28.) We used a global sigma determined by destiny (sigma=NULL), which is 5.86.

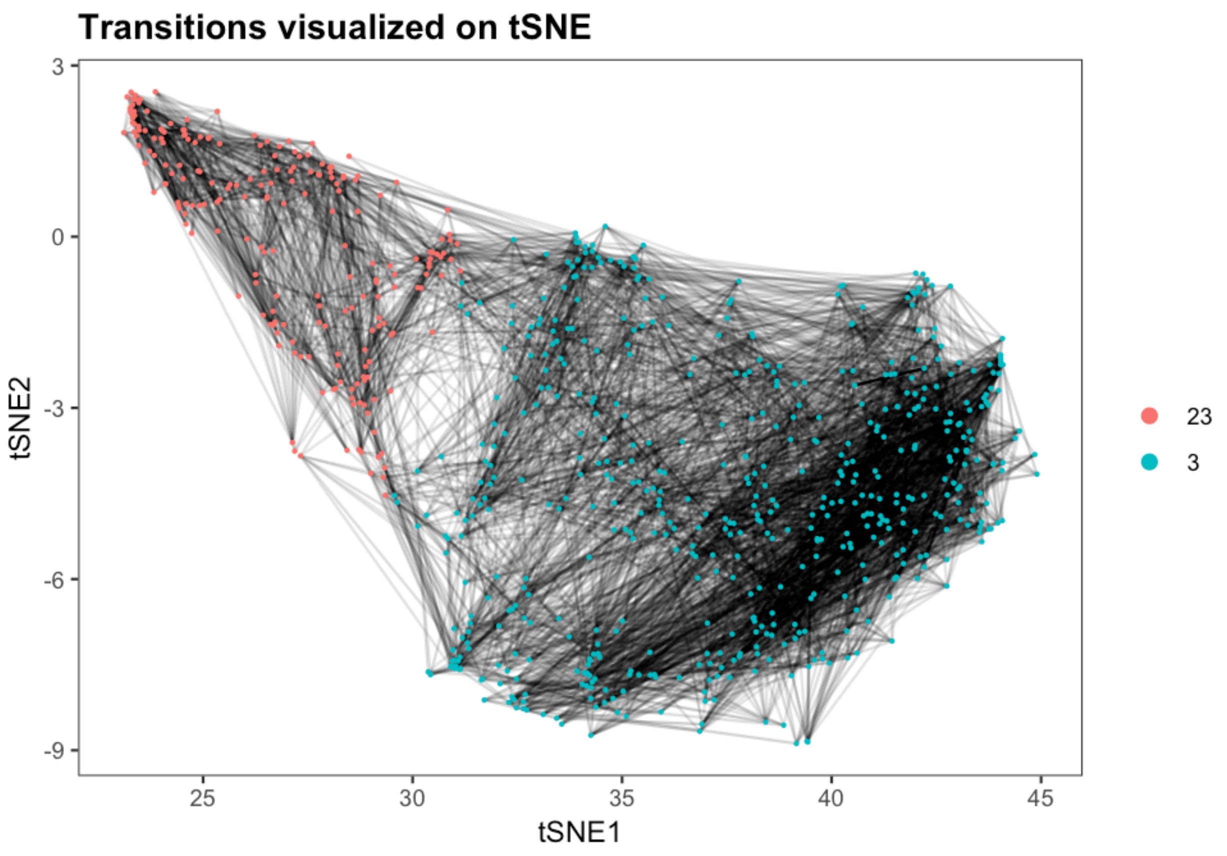
```r
# Calculate the diffusion map
hydra.spumous <- calcDM(hydra.spumous, knn = 75, sigma.use = 5.86)  # Was suggested by destiny sigma=NULL previously
```

```
## [1] "Using provided global sigma 5.86"
```

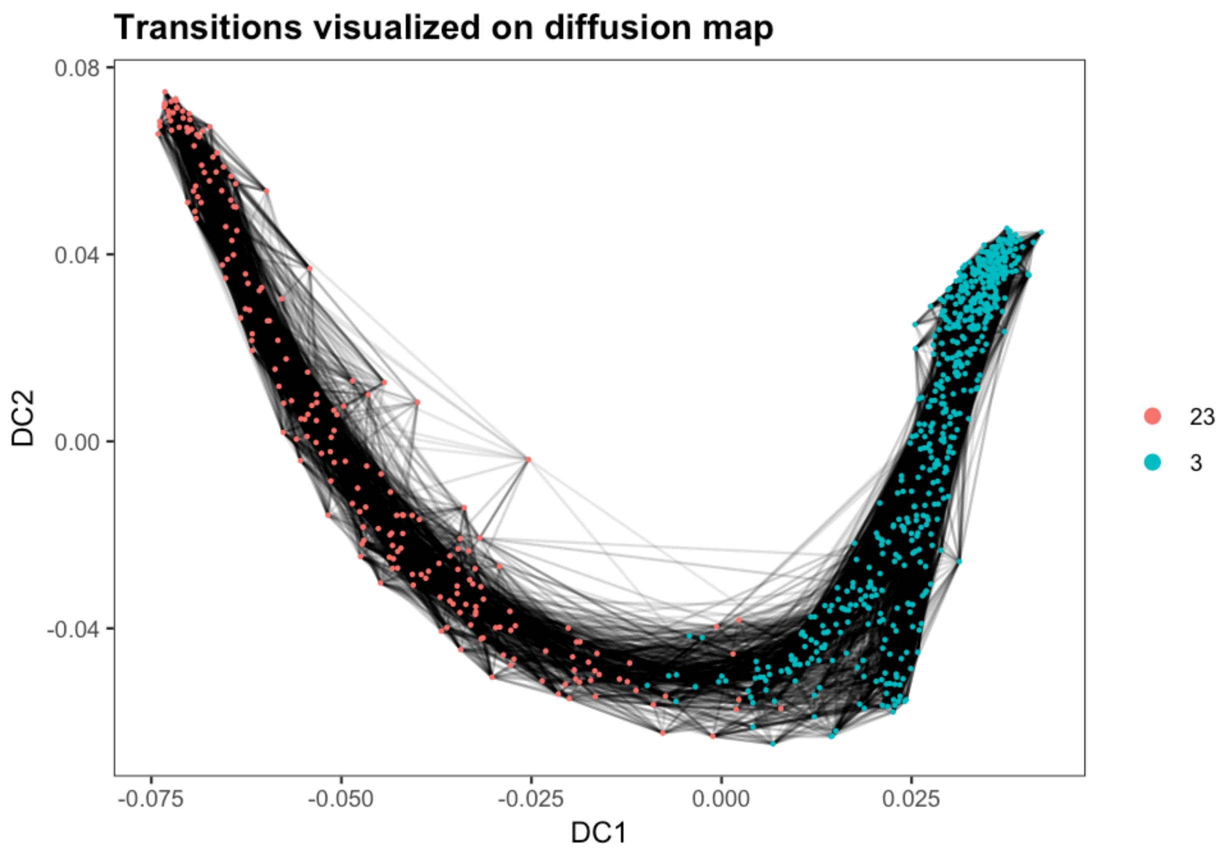
Evaluation of diffusion map

The diffusion map looks good and represents the ends of the differentiation process strongly as tips. Since this is a simple trajectory, it is evident in DC1 and DC2. The transitions seem make sense in the diffusion map and tSNE.

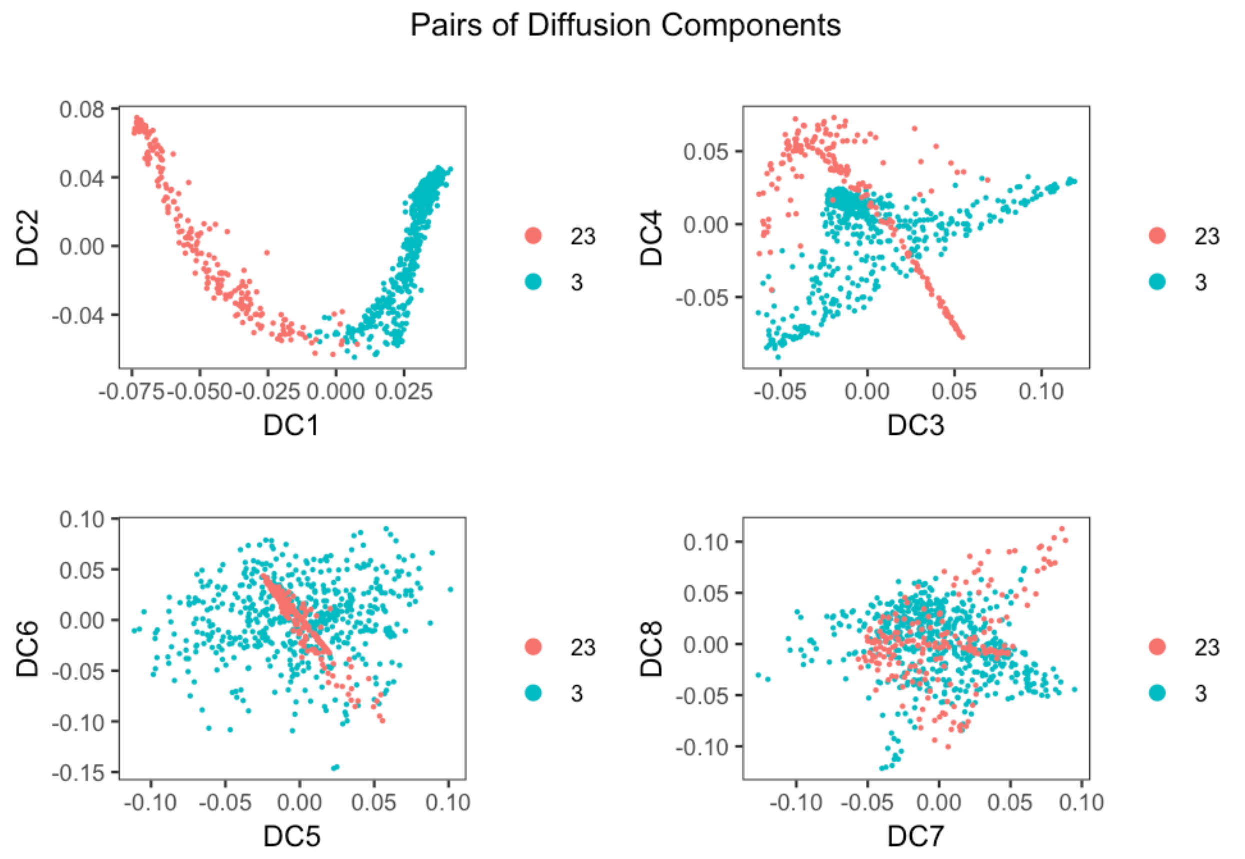
```r
# Make transition plots
plotDim(hydra.spumous, "res.1.5", transitions.plot = 5000, plot.title = "Transitions visualized on tSNE")
```



```r
plotDim(hydra.spumous, "res.1.5", transitions.plot = 10000, reduction.use = "dm",
    plot.title = "Transitions visualized on diffusion map")
```



```
# Make array plots
plotDimArray(hydra.spumous, label = "res.1.5", reduction.use = "dm", dims.to.plot = 1:8,
    plot.title = "", outer.title = "Pairs of Diffusion Components")
```



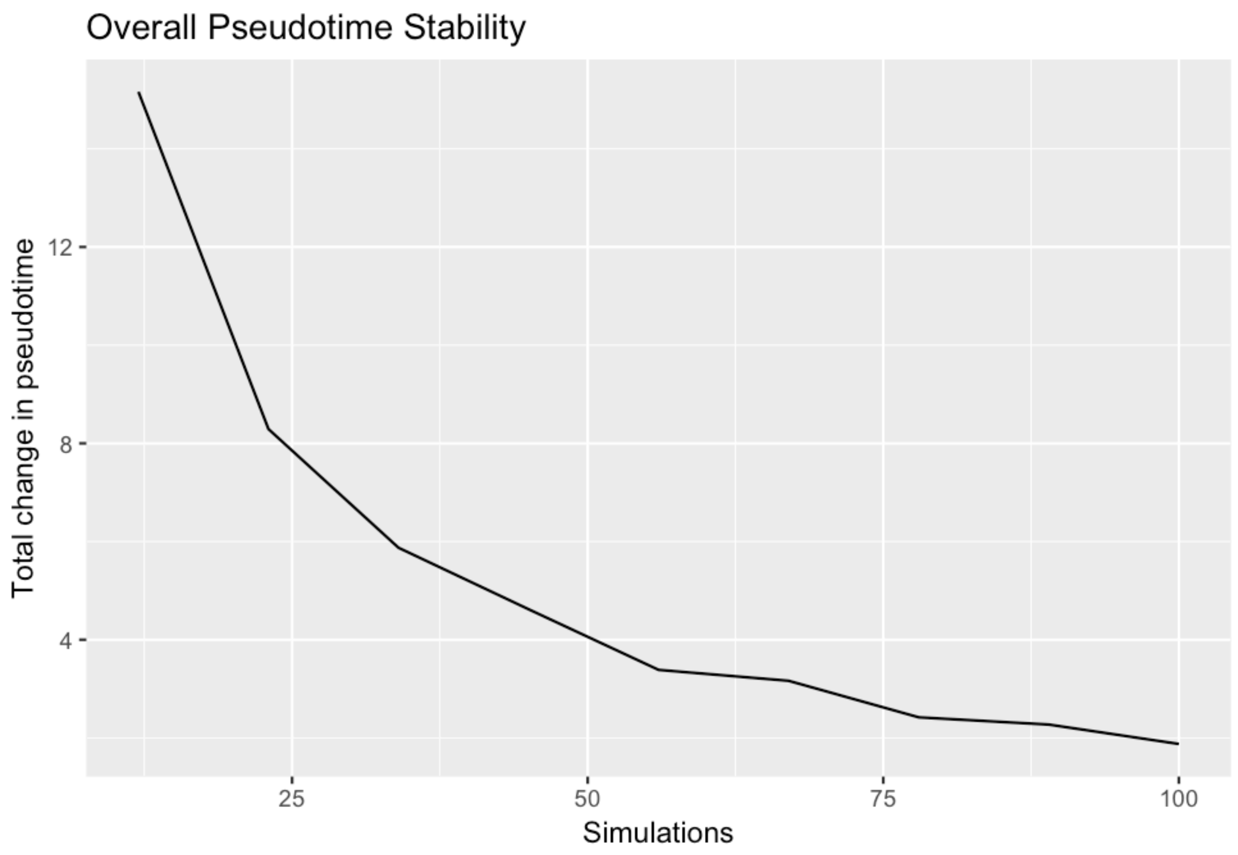
Calculate pseudotime

Here, we treat the process as a continuous head to food set of transitions. We calculated pseudotime starting from the oral end.
Since the simulation process is stochastic, we load our previously calculated results for consistency, but the following commands were run previously:

```r
# Perform the 'flood' simulations to determine cells' pseudotime
spumous.flood <- floodPseudotime(hydra.spumous, 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.

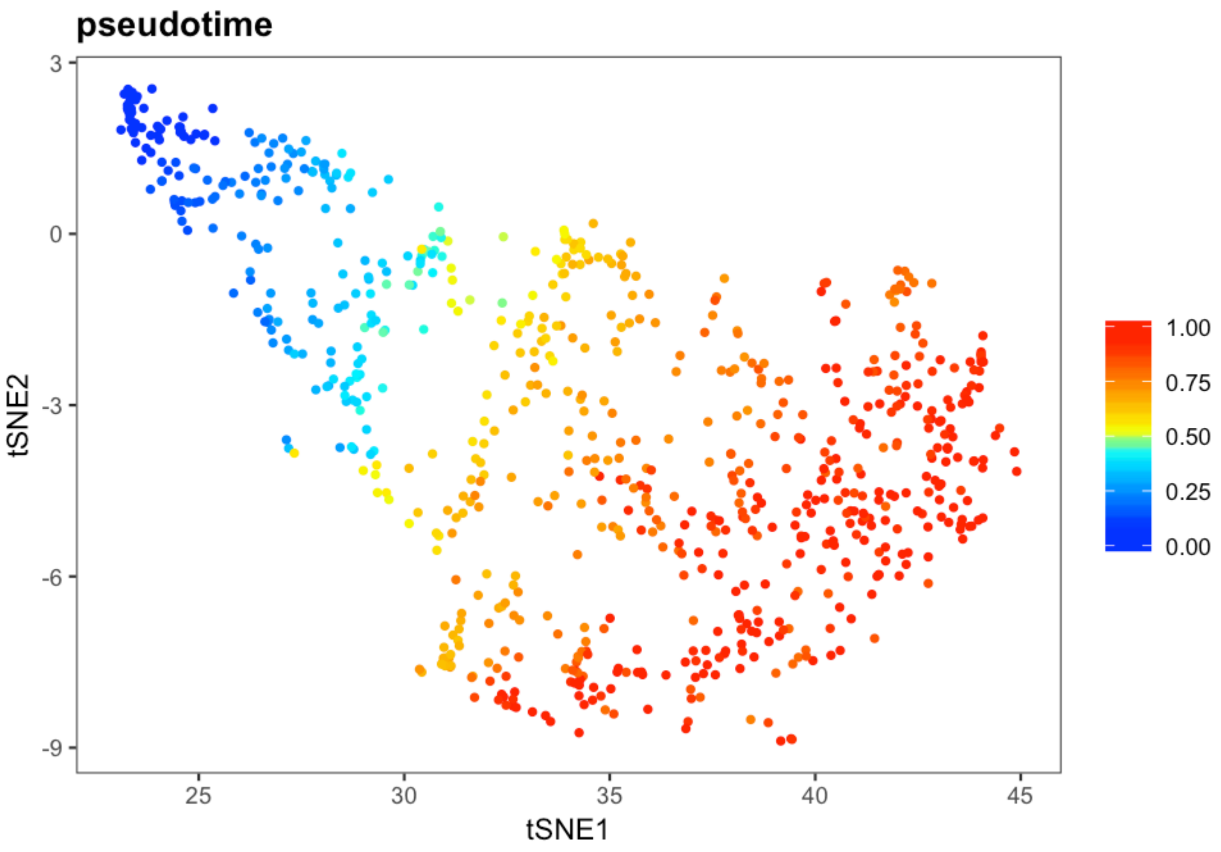
```r
# Process the floods to derive pseudotime
hydra.spumous <- floodPseudotimeProcess(hydra.spumous, spumous.flood, floods.name = "pseudotime")

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



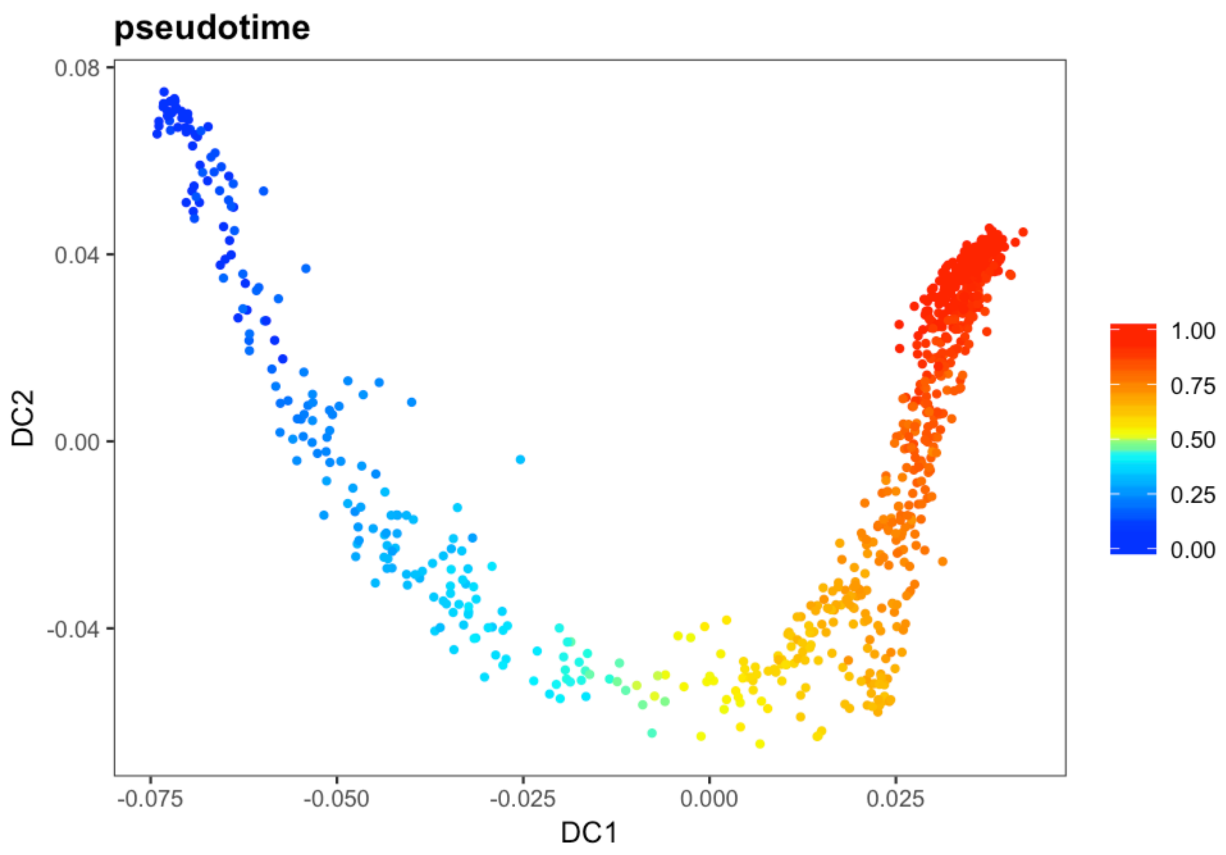
```r
# Normalize pseudotime length to 1
hydra.spumous@pseudotime$pseudotime <- hydra.spumous@pseudotime$pseudotime/max(hydra.spumous@pseudotime$pseudotime)
```

Pseudotime was calculated starting at 0 at the oral end. It seems to agree well with the tSNE and diffusion map and provides smooth ordering.

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



```
# And on the diffusion map
plotDim(hydra.spumous, "pseudotime", reduction.use = "dm")
```



Find spatially varying genes

Since in this trajectory, “pseudotime” is really a proxy for spatial location (oral to aboral), 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 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 varying genes

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

# Calculate spline fit
expressed.spline.5cell <- geneSmoothFit(hydra.spumous, method = "spline", pseudotime = "pseudotime",
    cells = colnames(hydra.spumous@logupx.data), genes = expressed.genes, moving.window = 1,
    cells.per.window = 5, spar = 0.875)
```

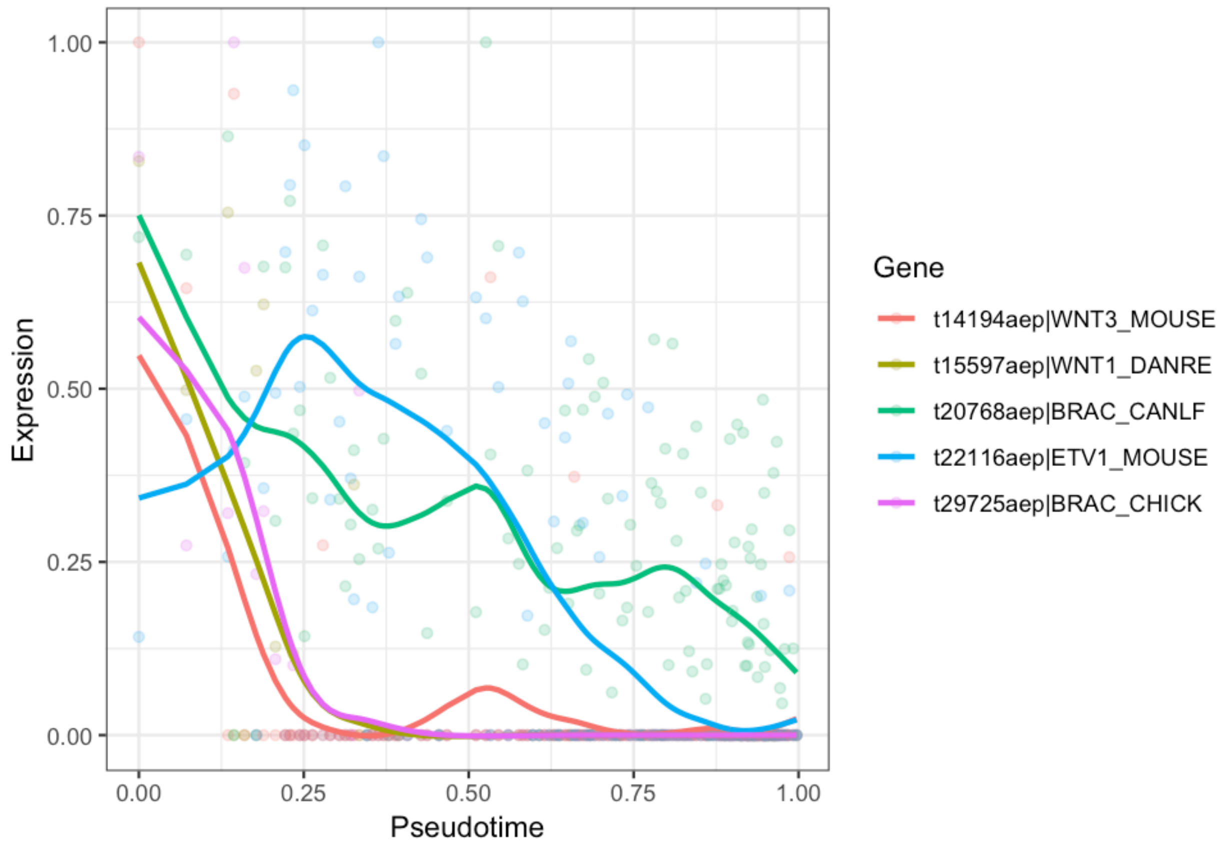
```
## [1] "2018-11-01 14:42:25: Calculating moving window expression."
## [1] "2018-11-01 14:42:47: Generating un-scaled fits."
## [1] "2018-11-01 14:43:06: Generating scaled fits."
## [1] "2018-11-01 14:43:24: Reducing mean expression data to same dimensions as spline fits."
```

```
# Which genes have a spline curve that crosses 0.5?
max.mean.spline.5cell <- apply(expressed.spline.5cell$mean.smooth, 1, max)
genes.wellexp.5cell <- names(which(max.mean.spline.5cell > 0.5))

# Look at how good the spline fits are
spline.fit.5cell <- apply(expressed.spline.5cell$scaled.smooth - expressed.spline.5cell$scaled.expression.red,
    1, function(i) sum(i^2))
spline.fit.norm.5cell <- spline.fit.5cell/ncol(expressed.spline.5cell$scaled.expression.red)
genes.wellfit.5cell <- names(which(spline.fit.norm.5cell <= 0.045))

# Which genes change in their scaled log2 mean expression value by at least 33%?
change.scale.5cell <- apply(expressed.spline.5cell$scaled.smooth, 1, function(x) diff(range(x)))
genes.scale.5cell <- names(which(change.scale.5cell >= 0.33))

# Take the intersection of those genes and use them in the heatmap & analysis
varying.genes.5cell <- intersect(intersect(genes.wellexp.5cell, genes.scale.5cell),
    genes.wellfit.5cell)
```

Spline plots

We can then plot the expression of genes and their fit splines.

```
genes.spline.plot <- c("t14194aep|WNT3_MOUSE", "t15597aep|WNT1_DANRE", "t20768aep|BRAC_CANLF",
    "t29725aep|BRAC_CHICK", "t22116aep|ETV1_MOUSE")
plotSmoothFit(expressed.spline.5cell, genes = genes.spline.plot, scaled = T)
```



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 <- expressed.spline.5cell$scaled.expression[varying.genes.5cell, ]
se.sd <- apply(se, 1, sd)
```

```r
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
pt.wm <- apply(se.z, 1, function(w) {
    weighted.mean(x = as.numeric(colnames(se.z)), w = w)
})

# Make some heatmaps
row.font.size = 0.3
pdf(paste0(main.path, "URD/SpumousMucous/SpumousMucous-Varying.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 = row.font.size, margins = c(5, 8), lwid = c(0.3,
        4), lhei = c(0.4, 4), labCol = NA)
title("Spumous Mucous (Oral to Aboral)", line = -1, adj = 0.48, cex.main = 4)
dev.off()
```

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

Save results

```r
# Save the results
saveRDS(expressed.spline.5cell, file = paste0(main.path, "URD/SpumousMucous/Splines-SpumousMucous.rds"))
write(varying.genes.5cell, file = paste0(main.path, "URD/SpumousMucous/Genes-SpumousMucous-Varying.txt"))
saveRDS(hydra.spumous, file = paste0(main.path, "URD/SpumousMucous/Hydra_URD_SpumousMucous.rds"))
```