SA12: Hydra URD Granular Mucous & Zymogen Gland Cells

Jeff Farrell

October 1, 2018, updated March 19, 2019

```
# 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 3.4.4
```

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

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

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

Load subsetted URD data for this lineage

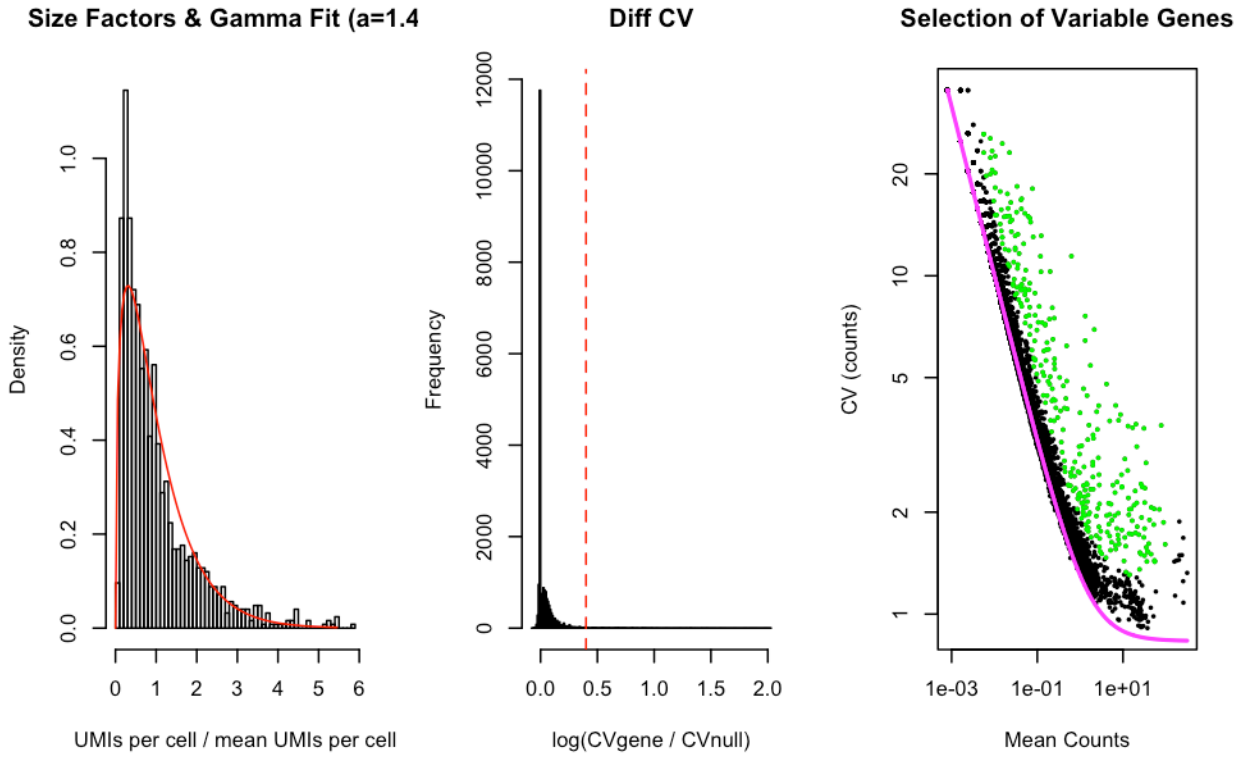
```
hydra.zymogen <- readRDS(file = paste0(main.path, "objects/Hydra_URD_Input_Zymogen.rds"))
```

Variable genes

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

```
# Determine variable genes (315 genes, 288 were variable across whole IC lineage)
hydra.zymogen <- findVariableGenes(object = hydra.zymogen, set.object.var.genes = T,
    diffCV.cutoff = 0.4, do.plot = T)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 11693 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.zymogen <- graphClustering(hydra.zymogen, num.nn = c(20, 30, 40, 50), do.jaccard = T,
    method = "Infomap")
hydra.zymogen <- graphClustering(hydra.zymogen, num.nn = c(5, 10, 15, 20, 30), do.jaccard = T,
    method = "Louvain")
hydra.zymogen <- graphClustering(hydra.zymogen, num.nn = c(6, 7, 8), do.jaccard = T,
    method = "Louvain")
# Record the clustering
pdf(paste0(main.path, "URD/GranularZymogen/Clusters-GranularZymogen-Infomap20.pdf"))
plotDim(hydra.zymogen, "Infomap-20", label.clusters = T, legend = F)
dev.off()
```

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

Diffusion map

Calculate the transition probabilities

Calculate a diffusion map specific to the granular 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 35.) We used a global sigma that is slightly smaller than the 'autodetected' one (here, 6).

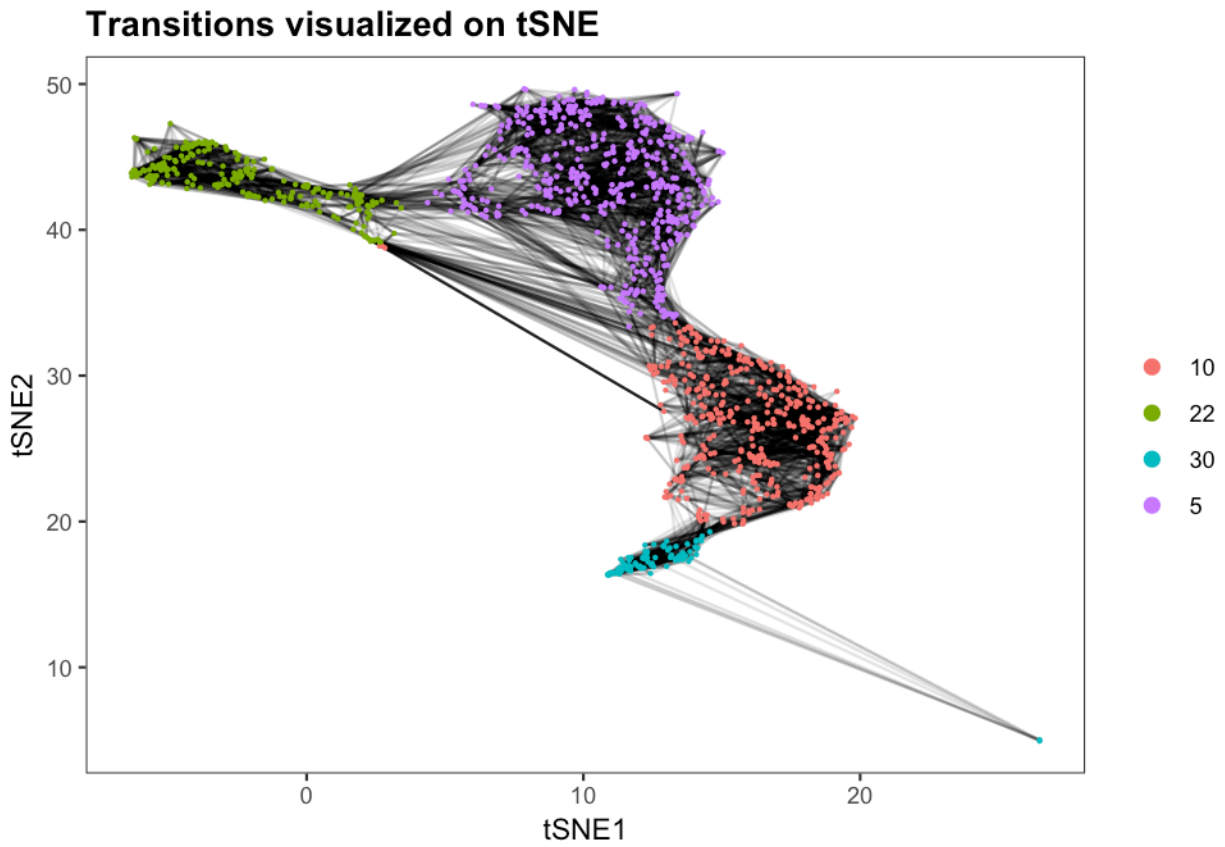
```
# Calculate diffusion maps without doublets or terminal nematocytes, using the
# more restricted variable gene list that lacks CCA genes
hydra.zymogen <- calcDM(hydra.zymogen, knn = 75, sigma.use = 6)
```

```
## [1] "Using provided global sigma 6"
```

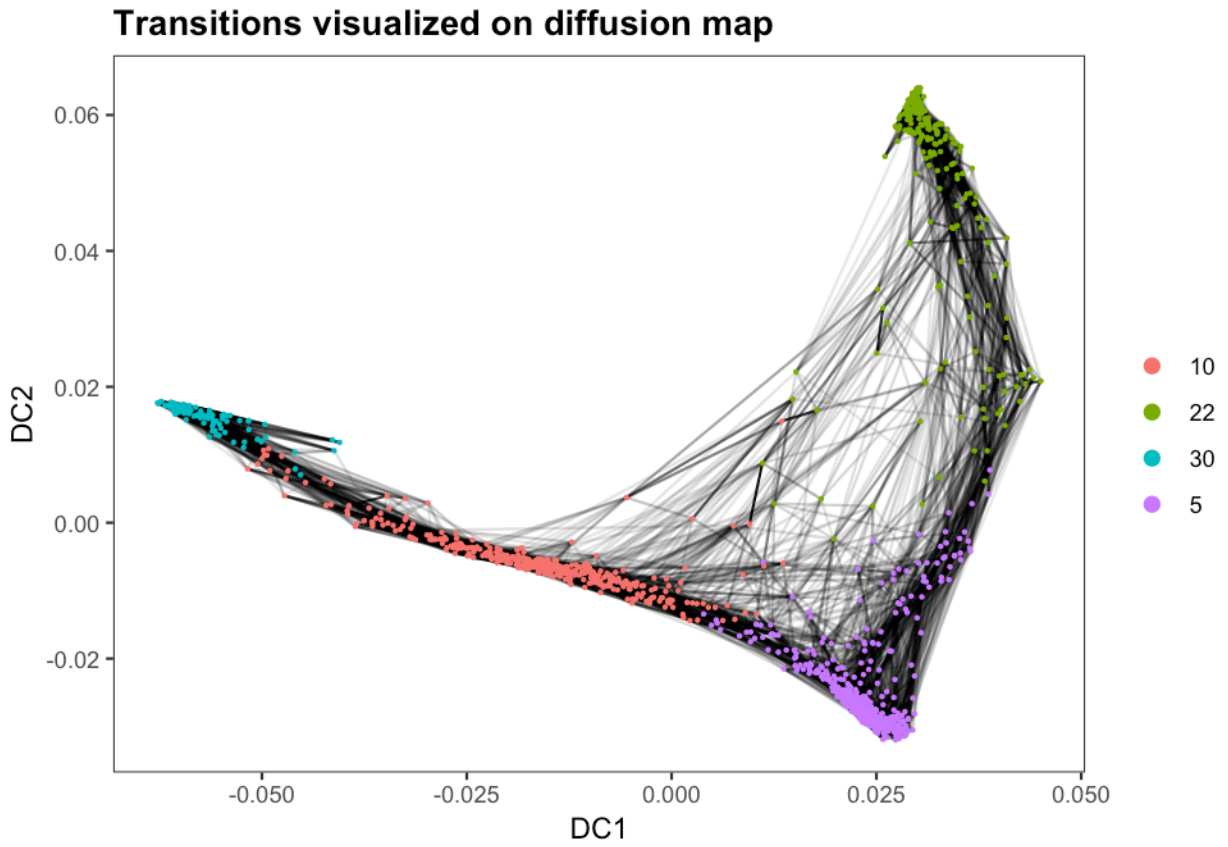
Evaluation of diffusion map

The diffusion map looks good and represents the ends of the differentiation process strongly as tips. In DC1 and DC2, it is evident that there are multiple pathways through the data – both a continuous head to foot differentiation that we aim to capture here, and also separate paths to multiple parts of the body column that are captured as a branching structure in the interstitial tree, and can be seen in DCs 3 and 4.

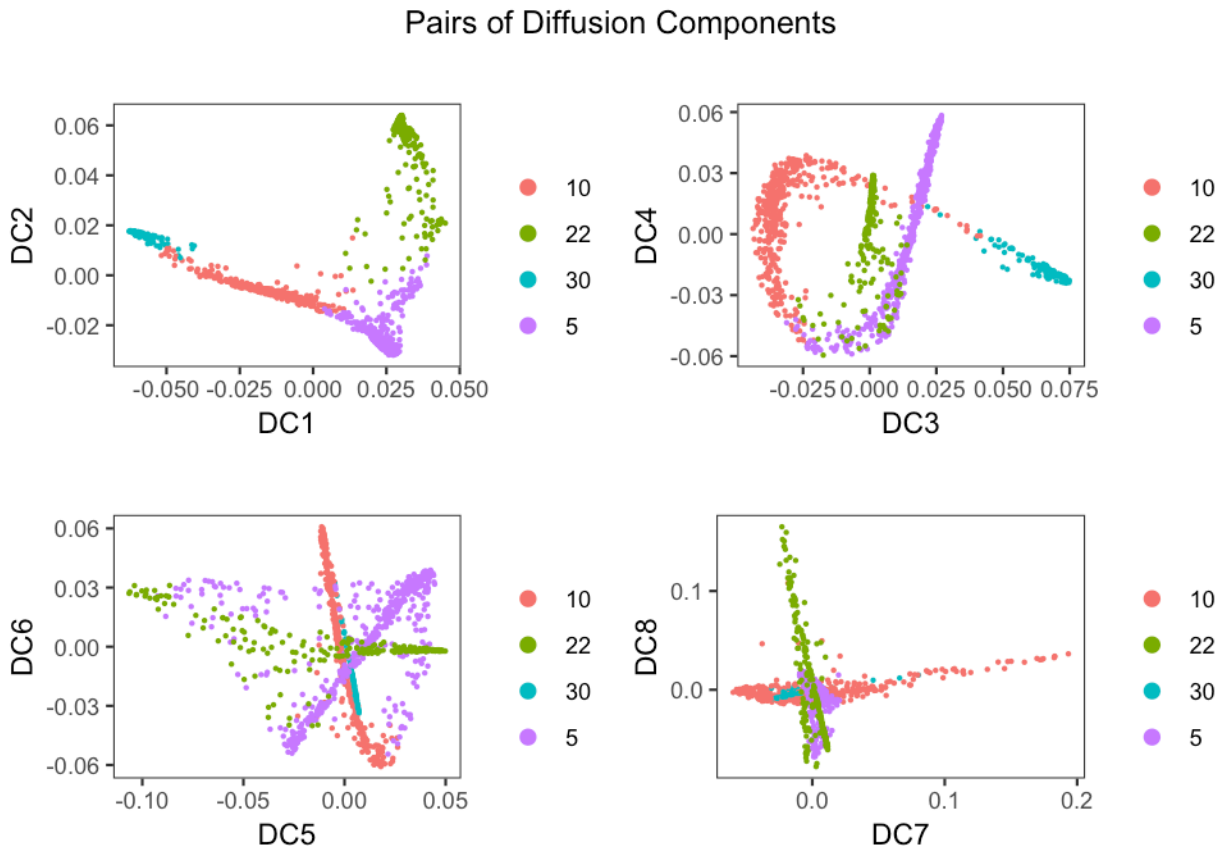
```
# Make transition plots
plotDim(hydra.zymogen, "res.1.5", transitions.plot = 5000, plot.title = "Transitions visualized on tSNE")
```




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




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



Calculate pseudotime

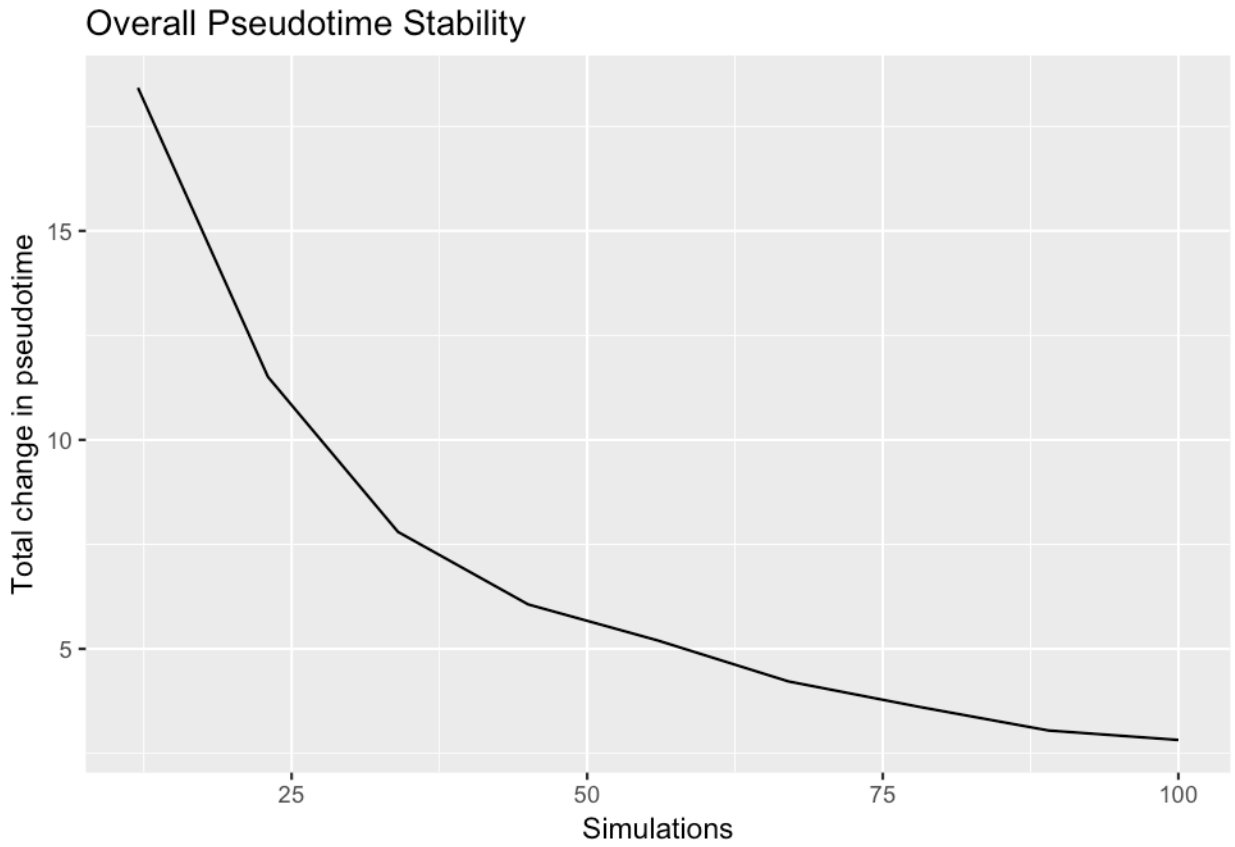
Here, we treat the process as a continuous head to food set of transitions. We calculate pseudotime from each end (starting with the head as root, then starting with the foot as root) and average the two pseudotimes to produce a single ordering across the entire granular mucous gland cell and zymogen gland cell lineage.

Since the simulation process is stochastic, we load our previously calculated results for consistency, but the following commands were run previously:

```
# Perform the 'flood' simulations to determine cells' pseudotime
zymogen.flood.13 <- floodPseudotime(hydra.zymogen, root.cells = cellsInCluster(hydra.zymogen,
    "Infomap-20", "13"), n = 100, minimum.cells.flooded = 2, verbose = T)
zymogen.flood.14 <- floodPseudotime(hydra.zymogen, root.cells = cellsInCluster(hydra.zymogen,
    "Infomap-20", "14"), 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.

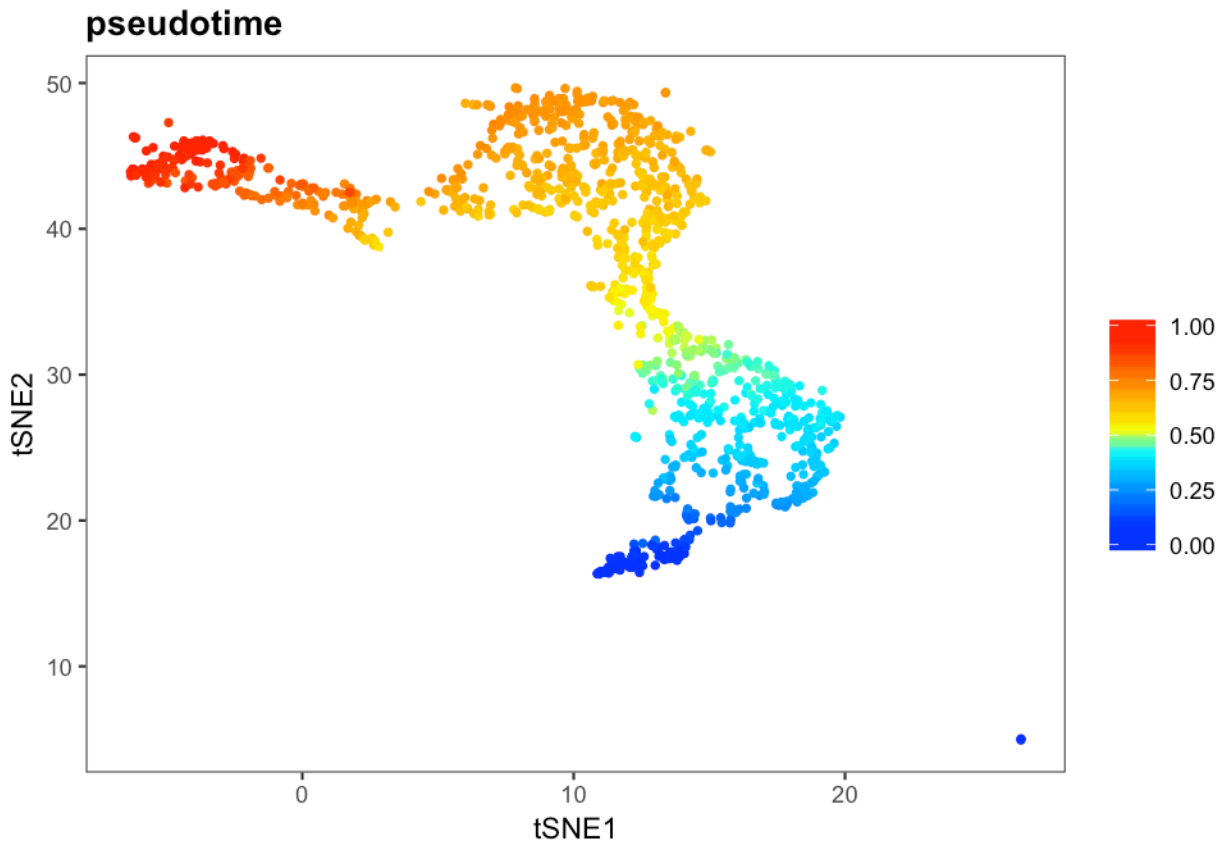
```
# Process the floods to derive pseudotime
hydra.zymogen <- floodPseudotimeProcess(hydra.zymogen, zymogen.flood.13, floods.name = "pseudotime.13")
hydra.zymogen <- floodPseudotimeProcess(hydra.zymogen, zymogen.flood.14, floods.name = "pseudotime.14")

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




```
# Combine the two into a single pseudotime so that you don't get a bunch of cells
# squished up at 0.
hydra.zymogen@pseudotime$pseudotime.13.norm <- hydra.zymogen@pseudotime$pseudotime.13/max(hydra.zymogen@pseudotime$pseudotime.13
hydra.zymogen@pseudotime$pseudotime.14.norm <- 1 - (hydra.zymogen@pseudotime$pseudotime.14/max(hydra.zymogen@pseudotime$pseudoti
hydra.zymogen@pseudotime$pseudotime <- rowMeans(hydra.zymogen@pseudotime[, c("pseudotime.13.norm",
    "pseudotime.14.norm")])
```

Pseudotime was calculated from the head to the foot. It has good temporal ordering across this trajectory.

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



Find spatially varying genes

Since in this trajectory, "pseudotime" is really a proxy for spatial location (head to foot), 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: (1) are well fit (noise is usually poorly fit, here we threshold on the sum of squared residuals), (2) vary significantly (their spline curve changes at least 0.5 in actual expression and their spline curve varies 30-40% depending on tight the fit is), and (3) are fit significantly better by the spline curve than a straight line with slope 0.

Calculate varying genes

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

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

```
## Warning in as.POSIXlt.POSIXct(x, tz): unknown timezone 'zone/tz/2018i.1.0/
## zoneinfo/America/New_York'
```

```
## [1] "2019-03-20 05:16:32: Calculating moving window expression."
## [1] "2019-03-20 05:17:10: Generating un-scaled fits."
## [1] "2019-03-20 05:17:36: Generating scaled fits."
## [1] "2019-03-20 05:18:03: Reducing mean expression data to same dimensions as spline fits."
```

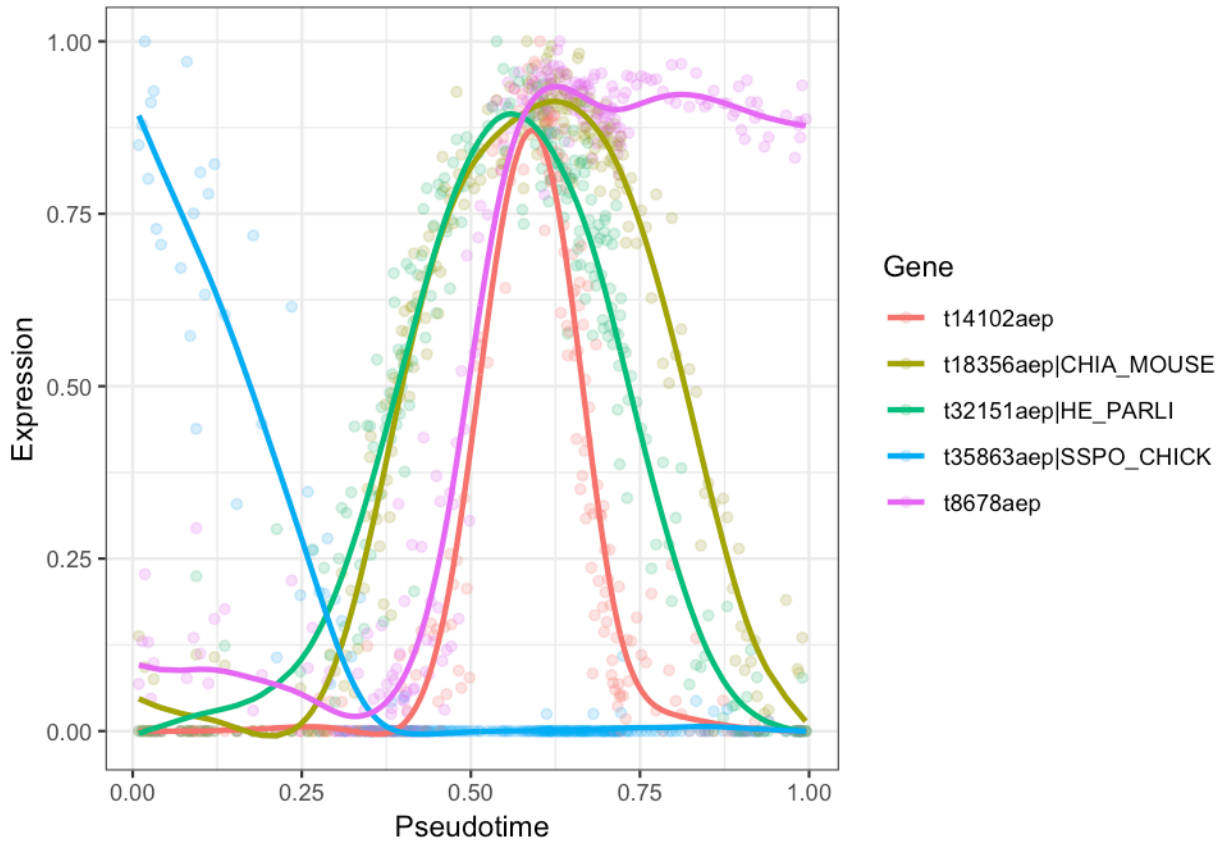
```
# Which genes change in their actual mean expression value by at least 0.5?
spline.change.real <- apply(expressed.spline.5cell$mean.smooth, 1, function(x) diff(range(x)))
```

```
genes.change.real <- names(which(spline.change.real >= 0.5))

# Which genes are well fit by the spline curves? (Noise is usually poorly fit by
# a curve)
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 sufficiently?  At
# least 30%, and requiring more change (up to 40%) as the data is less well fit
# by its spline curve.
change.scale.5cell <- apply(expressed.spline.5cell$scaled.smooth, 1, function(x) diff(range(x)))
genes.scale.5cell <- names(which(change.scale.5cell >= spline.fit.norm.5cell * 0.1/0.045 +
    0.3))

# Ensure that genes are fit by the spline curve significantly better than a flat
# line of slope 0.  (Weighted by distance to next point in pseudotime to
# compensate for point density)
w <- ncol(expressed.spline.5cell$scaled.smooth) - 1
weight <- diff(as.numeric(colnames(expressed.spline.5cell$scaled.smooth))) * 1000
spline.fit.weighted <- apply(expressed.spline.5cell$scaled.smooth[, 1:w] - expressed.spline.5cell$scaled.expression.red[,
    1:w], 1, function(i) sum(weight * i^2))
spline.flat.fit.weighted <- apply(expressed.spline.5cell$scaled.expression.red[,
    1:w], 1, function(x) sum(weight * (x - mean(x))^2))
spline.fit.ratio <- log2(spline.flat.fit.weighted/spline.fit.weighted)
spline.fit.betterthanflat <- names(which(spline.fit.ratio > 0.25))

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

Spline plots

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

```
genes.spline.plot <- c("t14102aep", "t8678aep", "t35863aep|SSPO_CHICK", "t18356aep|CHIA_MOUSE",
    "t32151aep|HE_PARLI")
plotSmoothFit(expressed.spline.5cell, genes = genes.spline.plot, scaled = T)
```



Heatmaps

Heatmap of all significantly spatially varying genes

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))

# Reduce the data in the splines, such that each block is minimum 0.01
# pseudotime; in this case, each block will be >= 5 cells, >= pseudotime 0.01
s <- expressed.spline.5cell
colnames(s$mean.expression) <- as.character(round(as.numeric(colnames(s$mean.expression)),
    digits = 2))
s$mean.expression <- matrixReduce(s$mean.expression)
colnames(s$mean.smooth) <- as.character(round(as.numeric(colnames(s$mean.smooth)),
    digits = 2))
s$mean.smooth <- matrixReduce(s$mean.smooth)

# Re-scale spline curves min/max for the heatmap
ss <- sweep(s$mean.smooth, 1, apply(s$mean.smooth, 1, min), "-")
ss <- sweep(ss, 1, apply(ss, 1, max), "/")

# Hierarchical cluster based on smoothed expression
h.ss <- hclust(dist(as.matrix(ss[varying.genes.5cell, ])), method = "ward.D2")
h.ss.d <- as.dendrogram(h.ss)
k = 15  # Cluster number chosen by eye.
h.ss.clust <- cutree(h.ss, k = k)

# Get cluster order as it will be in the heatmap
clust.order <- unique(h.ss.clust[rev(h.ss$order)])
h.ss.clust.ord <- plyr::mapvalues(from = clust.order, to = 1:k, x = h.ss.clust)
```

```
# Generate cluster color vector
cluster.h <- seq(0, 1, length.out = k + 1)
cluster.s <- rep(c(1, 0.75), length = k)
cluster.v <- rep(c(1, 0.75), length = k)
cluster.colors <- hsv(h = cluster.h[1:k], s = cluster.s, v = cluster.v)
h.ss.clust.col <- plyr::mapvalues(from = as.character(1:k), to = cluster.colors,
    x = h.ss.clust.ord)

# Generate the actual heatmap and save as a PDF.
pdf(paste0(main.path, "URD/GranularZymogen/GranularZymogen-Varying.pdf"), width = 17,
    height = 22)
# Plot the heatmap
gplots::heatmap.2(x = as.matrix(ss[varying.genes.5cell[rev(h.ss$order)], ]), Rowv = F,
    RowSideColors = h.ss.clust.col[rev(h.ss$order)], Colv = F, dendrogram = "none",
    col = cols, trace = "none", density.info = "none", key = F, cexCol = 0.8, cexRow = 0.08,
    margins = c(8, 8), lwid = c(0.3, 4), lhei = c(0.3, 4), labCol = NA)
# Put a title on it
title("Granular/Zymogen Expression", line = -1, adj = 0.48, cex.main = 4)
# Add tissue labels to bottom
title("Head", line = -103, adj = 0.08, cex.main = 2)
title("Foot", line = -103, adj = 0.95, cex.main = 2)
dev.off()
```

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

Heatmap cluster expression profiles

We clustered the expression of these spatially varying genes in order to describe the overall common spatial expression patterns.

```
# Make a spline object that aggregates the expression of genes from each cluster
# as their mean

# Make fake spline object by running aggregate on all its slots
so <- expressed.spline.5cell
# Aggregate by mean across clusters
so$scaled.expression <- stats::aggregate(s$scaled.expression[varying.genes.5cell,
    ], by = list(h.ss.clust.ord[varying.genes.5cell]), FUN = mean)
# Rename rows to add a leading 0 to 1-9 so ggplot2 will sort correctly
rownames(so$scaled.expression) <- sprintf("%02d", as.numeric(rownames(so$scaled.expression)))
so$mean.expression <- stats::aggregate(s$mean.expression[varying.genes.5cell, ],
    by = list(h.ss.clust.ord[varying.genes.5cell]), FUN = mean)
rownames(so$mean.expression) <- sprintf("%02d", as.numeric(rownames(so$mean.expression)))
so$scaled.smooth <- stats::aggregate(s$scaled.smooth[varying.genes.5cell, ], by = list(h.ss.clust.ord[varying.genes.5cell]),
    FUN = mean)
rownames(so$scaled.smooth) <- sprintf("%02d", as.numeric(rownames(so$scaled.smooth)))
so$mean.smooth <- stats::aggregate(s$mean.smooth[varying.genes.5cell, ], by = list(h.ss.clust.ord[varying.genes.5cell]),
    FUN = mean)
rownames(so$mean.smooth) <- sprintf("%02d", as.numeric(rownames(so$mean.smooth)))
so$mean.expression.red <- stats::aggregate(s$mean.expression.red[varying.genes.5cell,
    ], by = list(h.ss.clust.ord[varying.genes.5cell]), FUN = mean)
rownames(so$mean.expression.red) <- sprintf("%02d", as.numeric(rownames(so$mean.expression.red)))
so$scaled.expression.red <- stats::aggregate(s$scaled.expression.red[varying.genes.5cell,
    ], by = list(h.ss.clust.ord[varying.genes.5cell]), FUN = mean)
rownames(so$scaled.expression.red) <- sprintf("%02d", as.numeric(rownames(so$scaled.expression.red)))

# Plot expression profiles of each cluster
pdf(paste0(main.path, "URD/GranularZymogen/GranularZymogen-Cluster-Profiles.pdf"),
    width = 8.5, height = 11)
plotSmoothFit(so, sort(rownames(so$mean.expression)), scaled = T, plot.data = T,
    multiplot = T) + ggtitle("Granular/Zymogen Cluster Expression Profiles") + ylim(0,
    1)
```

```
## Warning in plotSmoothFit(so, sort(rownames(so$mean.expression)), scaled =
## T, : NAs introduced by coercion

## Warning in plotSmoothFit(so, sort(rownames(so$mean.expression)), scaled =
## T, : NAs introduced by coercion

## Warning: Removed 15 rows containing missing values (geom_point).

## Warning: Removed 15 rows containing missing values (geom_path).
```

```
dev.off()
```

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

"Zoom-in" heatmaps

We also present a zoom-in of the spatial expression heatmaps for readability.

```
## Make booklet of heatmaps for each cluster

# Choose clusters for each page
cluster.ends <- c(2, 5, 9, 15)
cluster.starts <- c(1, head(cluster.ends, -1) + 1)

pdf(paste0(main.path, "URD/GranularZymogen/GranularZymogen-Cluster-Heatmaps.pdf"),
    width = 17, height = 22)
for (c in 1:length(cluster.ends)) {
    # Which clusters to put in this heatmap
    c.use <- cluster.starts[c]:cluster.ends[c]
    # Determine which rows to plot
    rp <- rev(h.ss$order)[which(h.ss.clust.ord[rev(h.ss$order)] %in% c.use)]
    # Make the heatmap
    gplots::heatmap.2(x = as.matrix(ss[varying.genes.5cell[rp], ]), Rowv = F, RowSideColors = h.ss.clust.col[rp],
        Colv = F, dendrogram = "none", col = cols, trace = "none", density.info = "none",
        key = F, cexCol = 0.8, cexRow = 0.8, margins = c(8, 15), lwid = c(0.3, 4),
        lhei = c(0.3, 4), labCol = NA)
    # Put a title on it
    title(paste0("Granular/Zymogen: Clusters ", cluster.starts[c], " to ", cluster.ends[c]),
        line = -1, adj = 0.48, cex.main = 4)
    # Add tissue labels to bottom
    title("Head", line = -103, adj = 0.075, cex.main = 2)
    title("Foot", line = -103, adj = 0.9, cex.main = 2)
}
dev.off()
```

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

Transcription factor heatmaps

And, finally, we isolated the transcription factors that vary spatially in the tissue, as potential candidates for important regulators of the differentiation of particular tissues.

```
# Load list of Hydra TFs
tfs <- read.table(paste0(main.path, "objects/aepLRv2_TFs.txt"), header = T, stringsAsFactors = F)
tf.genes <- tfs$ID

# Get ectoderm varying genes that are also TFs
varying.tfs <- intersect(tf.genes, varying.genes.5cell)

# Figure out how they should be ordered on the plot
tf.order <- t(apply(ss[varying.tfs, ] > 0.75, 1, function(x) {
    y <- which(x)
    return(c(min(y), max(y)))
}))
tfs.ordered <- varying.tfs[order(tf.order[, 1], tf.order[, 2], decreasing = c(F,
    F), method = "radix")]

# Make a heatmap of just the TFs
pdf(paste0(main.path, "URD/GranularZymogen/GranularZymogen-SpatialTFs.pdf"), width = 8.5,
    height = 11)
gplots::heatmap.2(as.matrix(ss[tfs.ordered, ]), Rowv = F, Colv = F, dendrogram = "none",
    col = cols, trace = "none", density.info = "none", key = F, cexCol = 0.8, cexRow = 1,
    margins = c(5, 13), lwid = c(0.3, 4), lhei = c(0.35, 4), labCol = NA)
title("Granular/Zymogen Varying TFs", line = 1.5, adj = 0.5, cex.main = 1.5)
title("Head", line = -49, adj = 0, cex.main = 1.5)
title("Foot", line = -49, adj = 0.7, cex.main = 1.5)
dev.off()
```

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

Save results

```
# Make data frame of genes and their cluster identities
write.table(data.frame(gene = varying.genes.5cell[rev(h.ss$order)], cluster = h.ss.clust.ord[rev(h.ss$order)]),
    quote = F, row.names = F, sep = "\t", file = paste0(main.path, "URD/GranularZymogen/GranularZymogen-Genes-Varying.txt"))
# Save objects
saveRDS(expressed.spline.5cell, file = paste0(main.path, "URD/GranularZymogen/Splines-GranularZymogen.rds"))
saveRDS(hydra.zymogen, file = paste0(main.path, "URD/GranularZymogen/Hydra_URD_GranularZymogen.rds"))
```