SA14: Hydra URD Male Germline (Transcriptome)

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/MaleTranscriptome/"), recursive = T)
```

Load subsetted URD data for this lineage

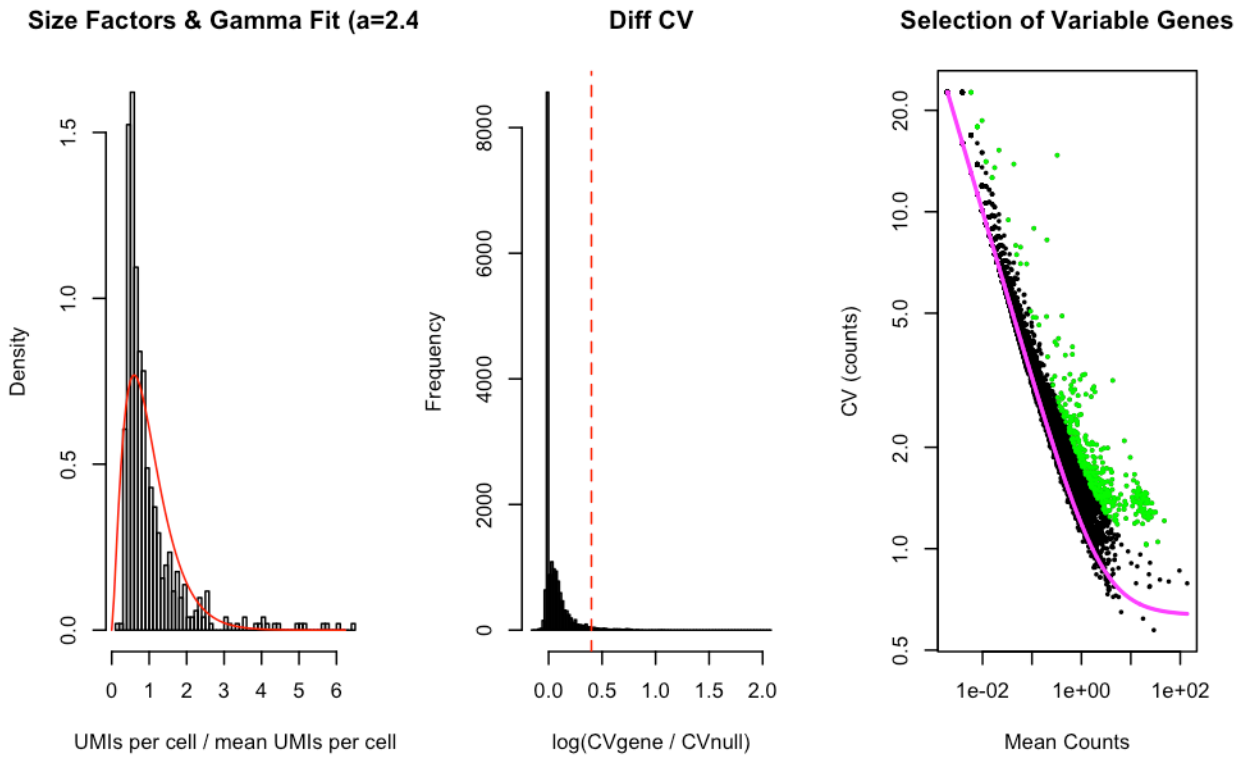
```
hydra.male <- readRDS(file = paste0(main.path, "objects/Hydra_URD_Input_Male.rds"))
```

Variable genes

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

```
# Determine variable genes (398 genes, 205 were variable across whole IC lineage)
hydra.male <- findVariableGenes(object = hydra.male, set.object.var.genes = T, diffCV.cutoff = 0.4,
    do.plot = T)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 16025 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.male <- graphClustering(hydra.male, num.nn = c(20, 30, 40, 50), do.jaccard = T,
    method = "Infomap")
hydra.male <- graphClustering(hydra.male, num.nn = c(5, 10, 15, 20, 30), do.jaccard = T,
    method = "Louvain")
hydra.male <- graphClustering(hydra.male, num.nn = c(6, 7, 8), do.jaccard = T, method = "Louvain")
# Record the clustering
pdf(paste0(main.path, "URD/MaleTranscriptome/Clusters-MaleTranscriptome-Infomap20.pdf"))
plotDim(hydra.male, "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 male germline 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 23.) We used a global sigma autodetected by *destiny* here (7.04).

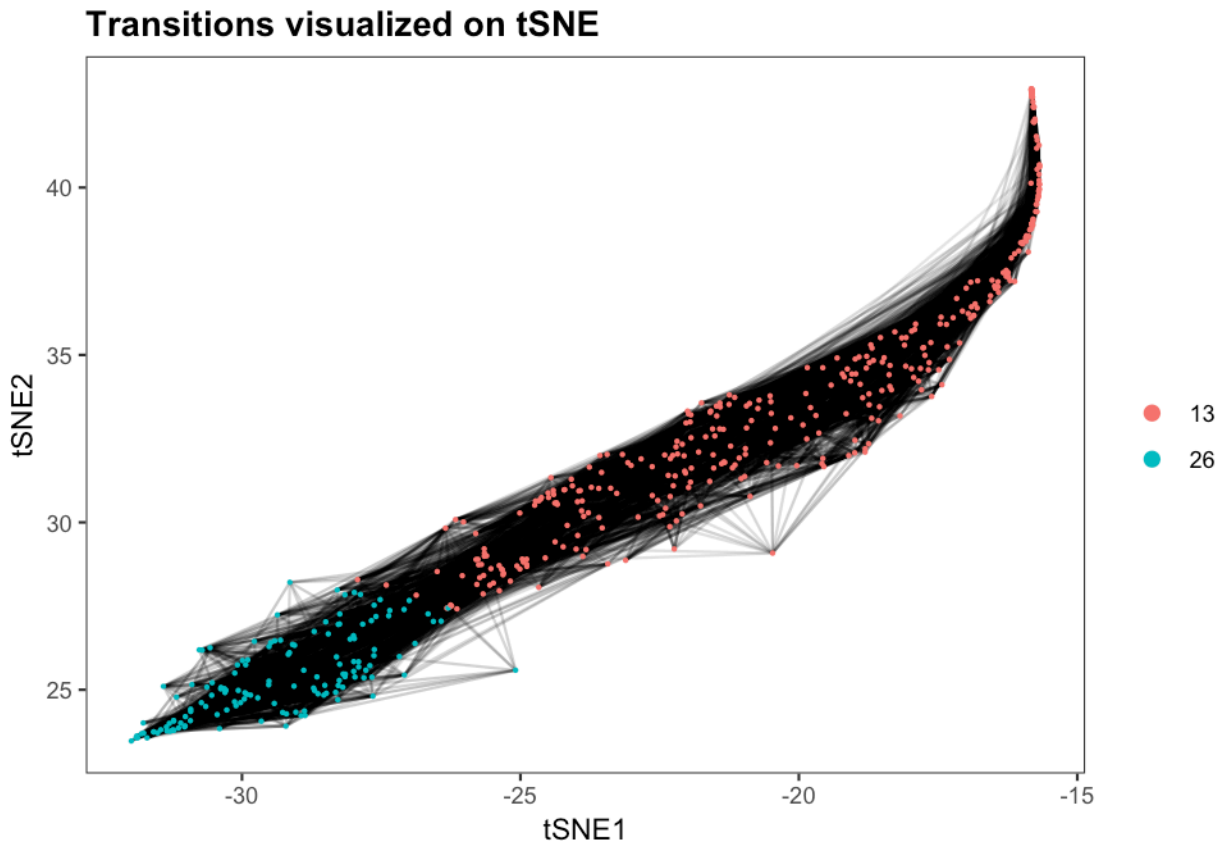
```
# Calculate diffusion map
hydra.male <- calcDM(hydra.male, knn = 75, sigma.use = 7.04)
```

```
## [1] "Using provided global sigma 7.04"
```

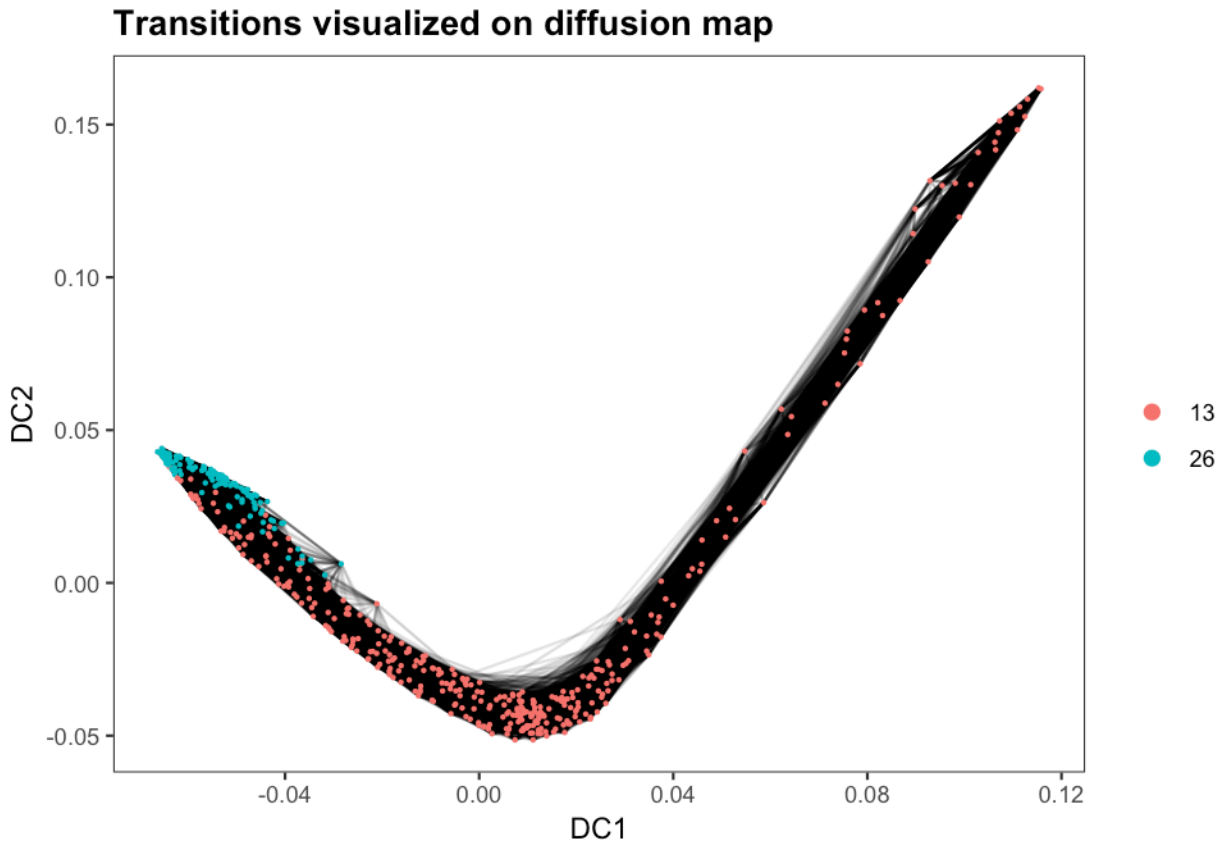
Evaluation of diffusion map

The diffusion map looks good and represents the ends of the differentiation process strongly as tips. This is a simple trajectory, so it's reflected pretty clearly solely in the diffusion map in DCs 1 and 2.

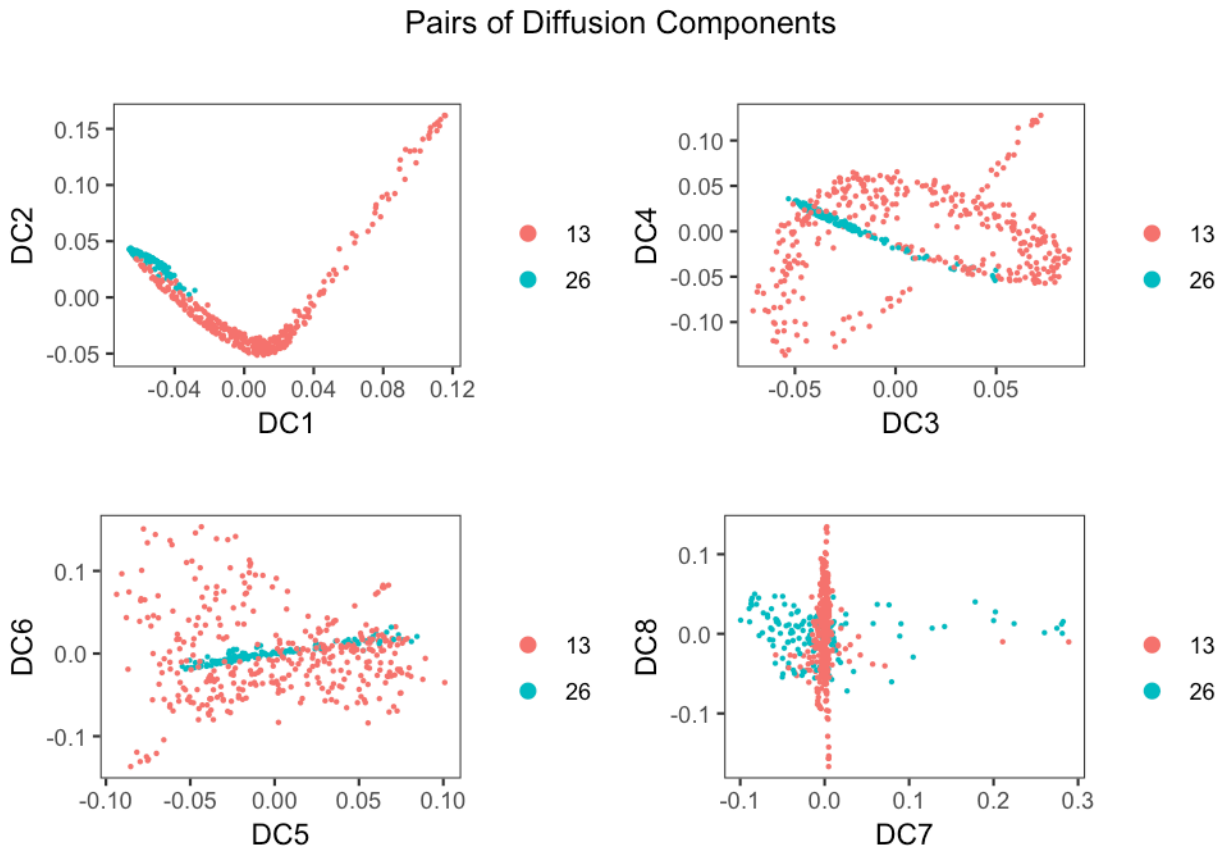
```
# Make transition plots
plotDim(hydra.male, "res.1.5", transitions.plot = 5000, plot.title = "Transitions visualized on tSNE")
```




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




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



Calculate pseudotime

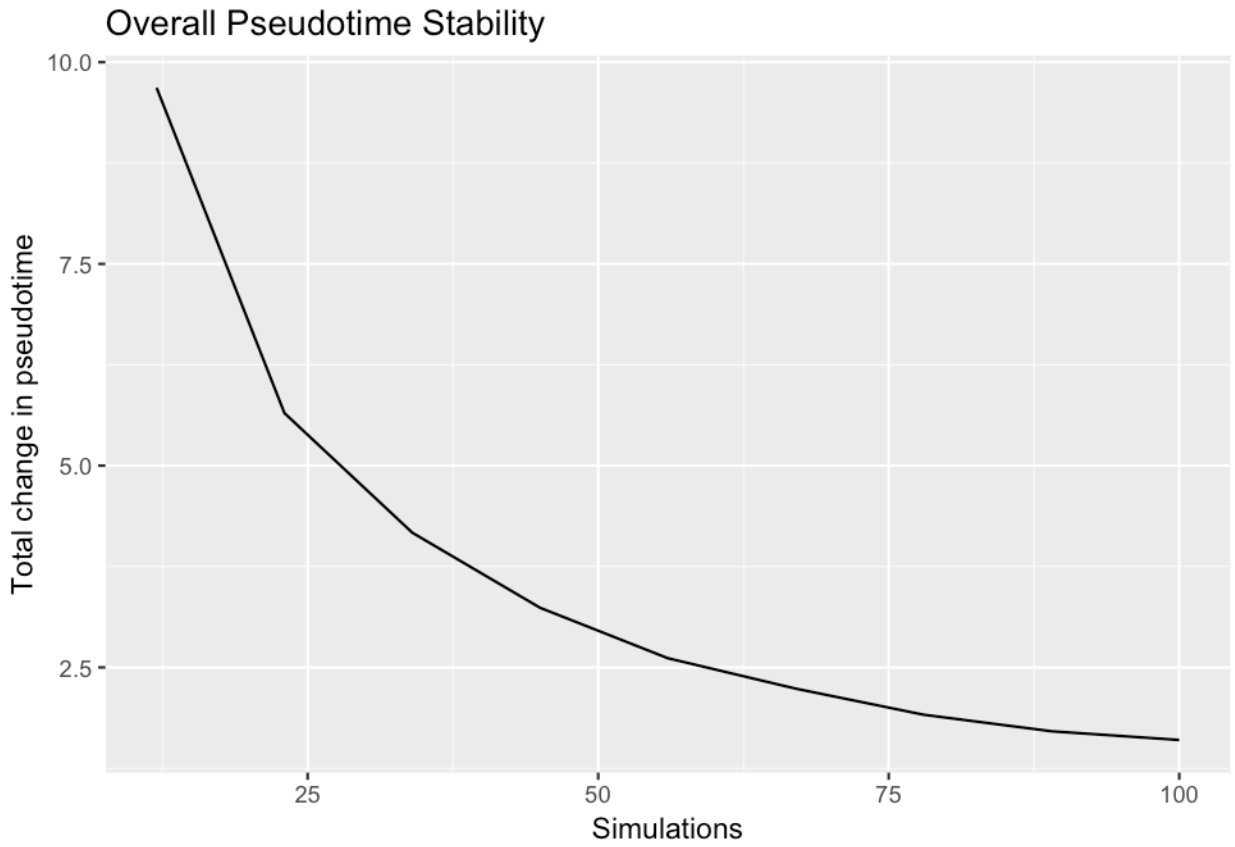
Here, we calculated pseudotime from each end of the germline trajectory (starting at the beginning and starting at the end) and then average the two pseudotimes to produce a single ordering across the entire male germline lineage.

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
male.flood.9 <- floodPseudotime(hydra.male, root.cells = cellsInCluster(hydra.male,
    "Infomap-20", "9"), n = 100, minimum.cells.flooded = 2, verbose = T)
male.flood.7 <- floodPseudotime(hydra.male, root.cells = cellsInCluster(hydra.male,
    "Infomap-20", "7"), 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.

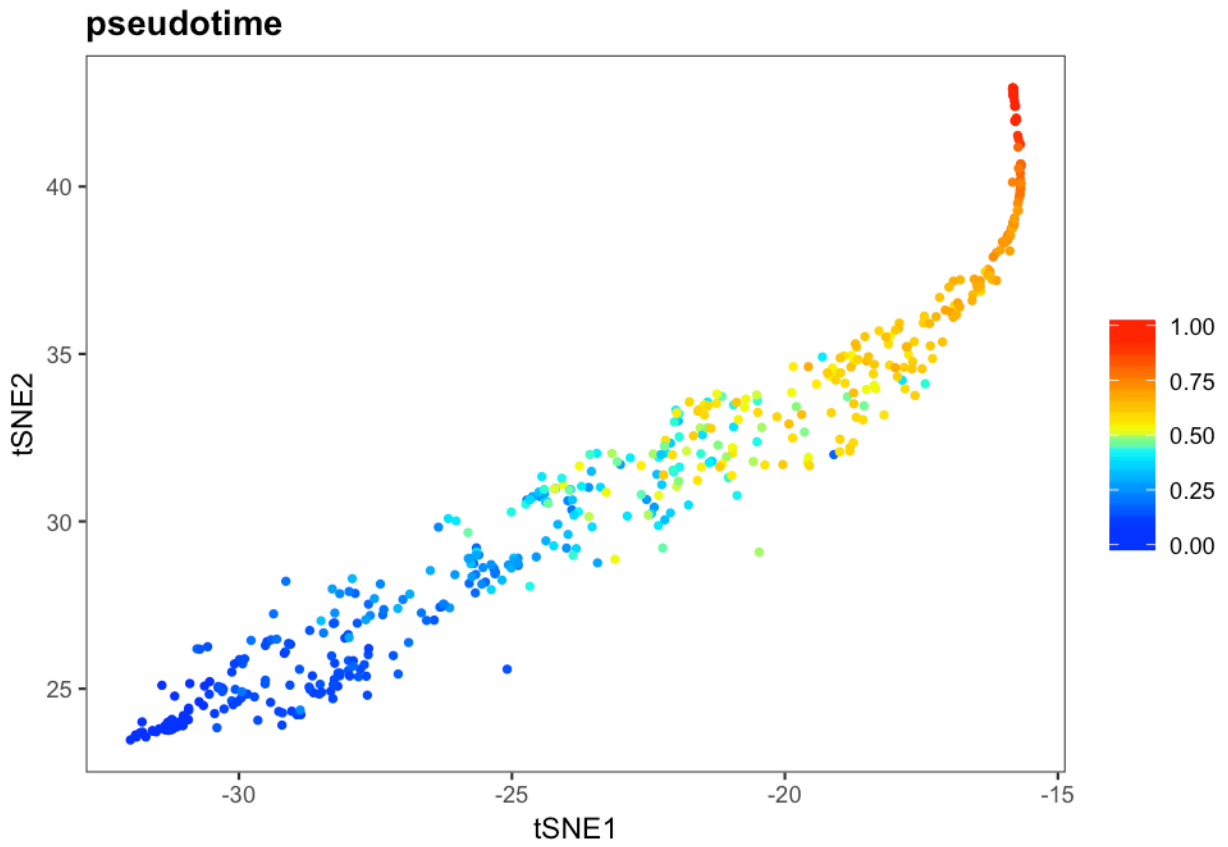
```
# Process the floods to derive pseudotime
hydra.male <- floodPseudotimeProcess(hydra.male, male.flood.9, floods.name = "pseudotime.9")
hydra.male <- floodPseudotimeProcess(hydra.male, male.flood.7, floods.name = "pseudotime.7")

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




```
# Combine the two into a single pseudotime so that you don't get a bunch of cells
# squished up at 0.
hydra.male@pseudotime$pseudotime.9.norm <- hydra.male@pseudotime$pseudotime.9/max(hydra.male@pseudotime$pseudotime.9)
hydra.male@pseudotime$pseudotime.7.norm <- 1 - (hydra.male@pseudotime$pseudotime.7/max(hydra.male@pseudotime$pseudotime.7))
hydra.male@pseudotime$pseudotime <- rowMeans(hydra.male@pseudotime[, c("pseudotime.7.norm",
    "pseudotime.9.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.male, "pseudotime")
```



Find genes that vary in pseudotime

To find the genes that vary in pseudotime (i.e. that are differential in the male germline as it differentiates), 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.male@logupx.data > 0)/ncol(hydra.male@logupx.data)
expressed.genes <- names(frac.exp)[which(frac.exp > 0.01)]

# Calculate spline fit
expressed.spline.5cell <- geneSmoothFit(hydra.male, method = "spline", pseudotime = "pseudotime",
    cells = colnames(hydra.male@logupx.data), genes = expressed.genes, moving.window = 1,
    cells.per.window = 5, spar = 0.8)  #.875 previously
```

```
## [1] "2018-11-01 14:44:57: Calculating moving window expression."
## [1] "2018-11-01 14:45:19: Generating un-scaled fits."
## [1] "2018-11-01 14:45:36: Generating scaled fits."
## [1] "2018-11-01 14:45:53: Reducing mean expression data to same dimensions as spline fits."
```

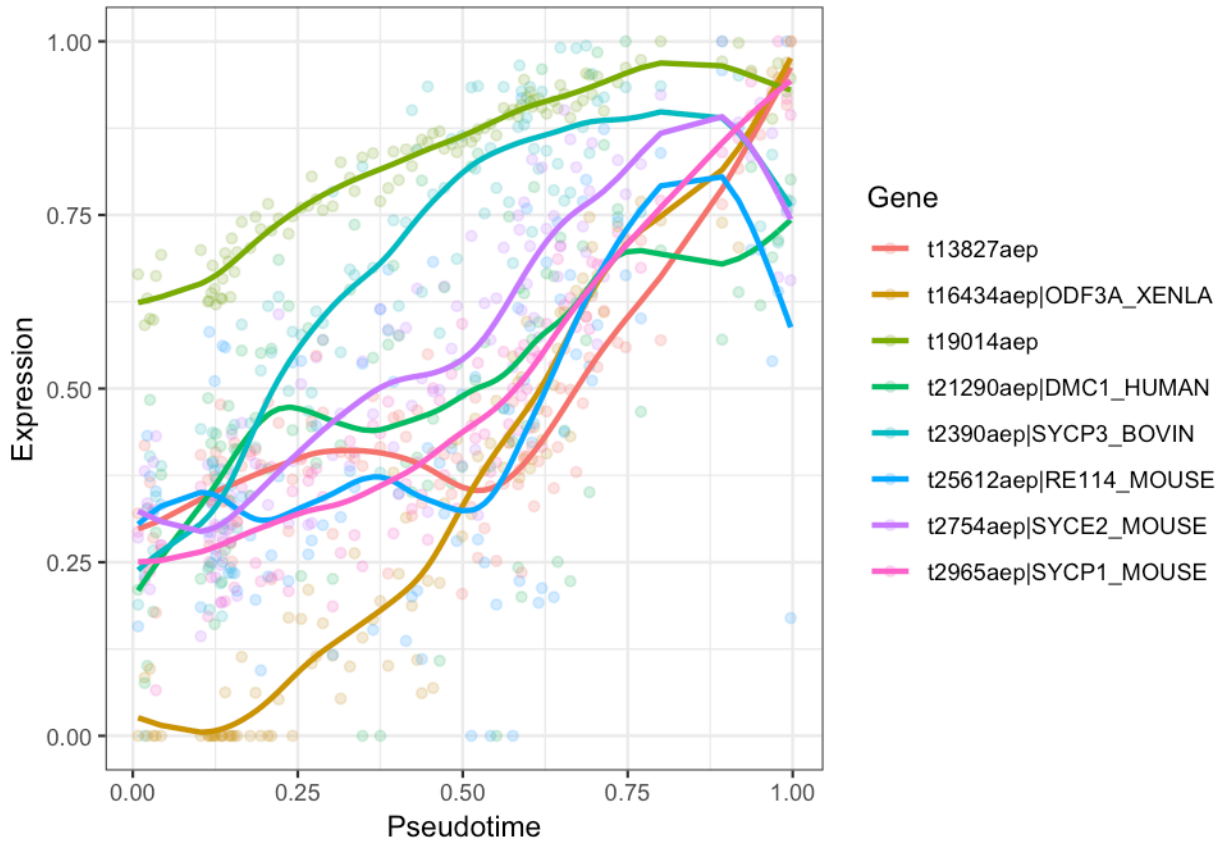
```
# Which genes have a spline curve that crosses 0.5?
max.mean.spline.5cell <- apply(expressed.spline.5cell$mean.smooth, 1, max)
genes.wellexp.5cell <- names(which(max.mean.spline.5cell > 0.5))

# 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 <= 0.06))

# 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("t2965aep|SYCP1_MOUSE", "t2754aep|SYCE2_MOUSE", "t2390aep|SYCP3_BOVIN",
    "t16434aep|ODF3A_XENLA", "t21290aep|DMC1_HUMAN", "t25612aep|RE114_MOUSE", "t13827aep",
    "t19014aep")
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)
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 = 0.06
pdf(paste0(main.path, "URD/MaleTranscriptome/MaleTranscriptome-Varying.pdf"), width = 25.5,
    height = 33)
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("Male Germline (Transcriptome, Pseudotime)", line = -1, adj = 0.48, cex.main = 4)
dev.off()
```

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

Save results

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