

# SA15: Hydra URD Male Germline (Genome)

*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/MaleGenome/"), recursive = T)
```

## Load data

We load the genome-aligned Seurat object, convert it an URD object, and then subset to the male germline

```
# Load the Seurat object aligned to the genome and convert to an URD object
hydra.genome <- readRDS(file = paste0(main.path, "objects/Hydra_Seurat_Whole_Genome.rds"))
hydra.genome.urd <- seuratToURD(hydra.genome)

## [1] "Marchenko-Pastur eigenvalue null upper bound: 21.7607698992586"
## [1] "10 PCs have larger eigenvalues."

# Subset to cluster 18, which contains the male germline
hydra.male <- urdSubset(hydra.genome.urd, cells.keep = cellsInCluster(hydra.genome.urd,
  "res.1.5", "18"))

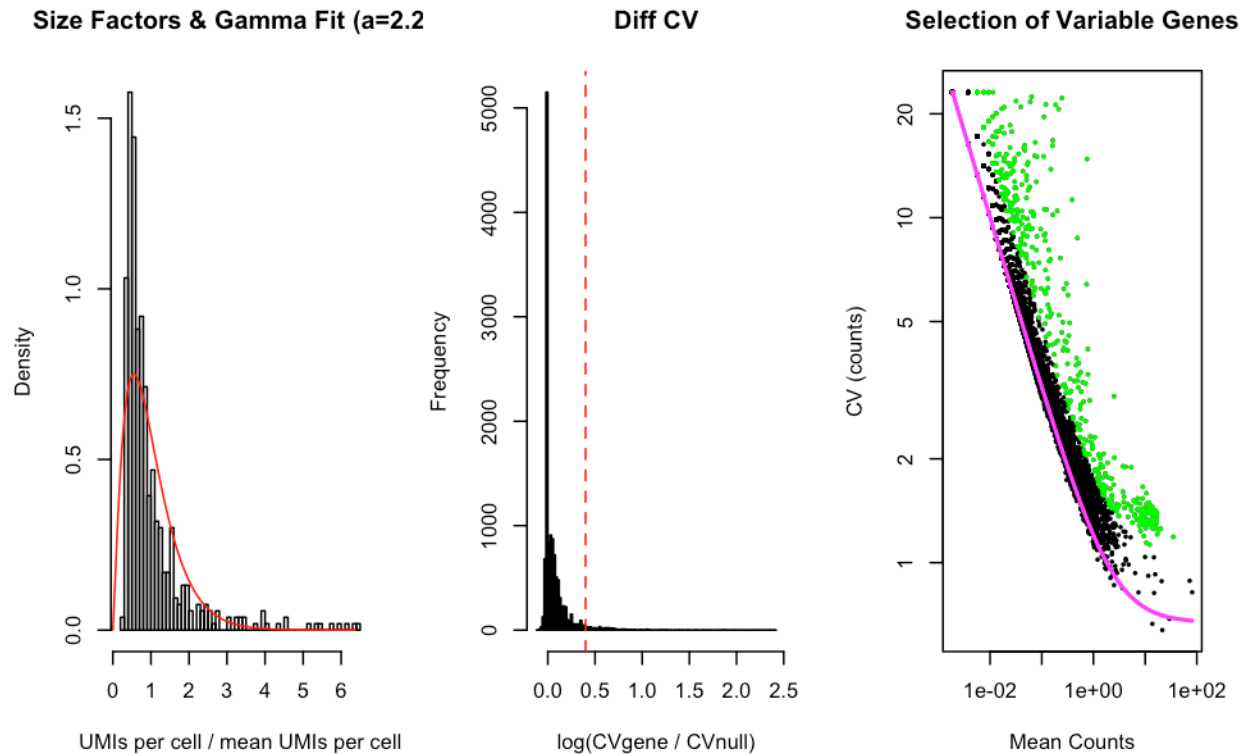
# Remove the Seurat and full genome URD objects, then garbage collect to keep RAM
# usage down
rm(list = c("hydra.genome.urd", "hydra.genome"))
shh <- gc()
```

## Variable genes

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

```
# Determine variable genes
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): 4584 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/MaleGenome/Clusters-MaleGenome-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 (9.62).

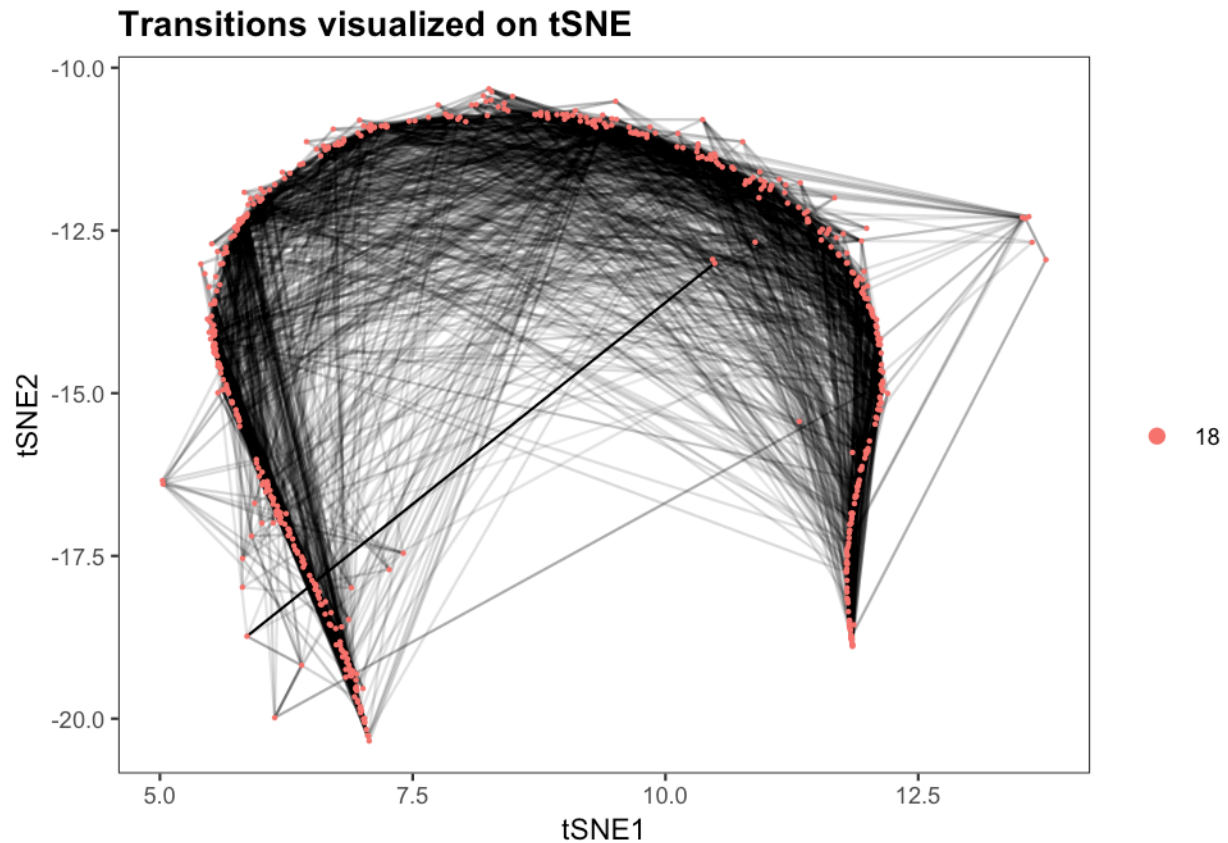
```
# Calculate diffusion map
hydra.male <- calcDM(hydra.male, knn = 75, sigma.use = 9.624249)

## [1] "Using provided global sigma 9.624249"
```

## Evaluation of diffusion map

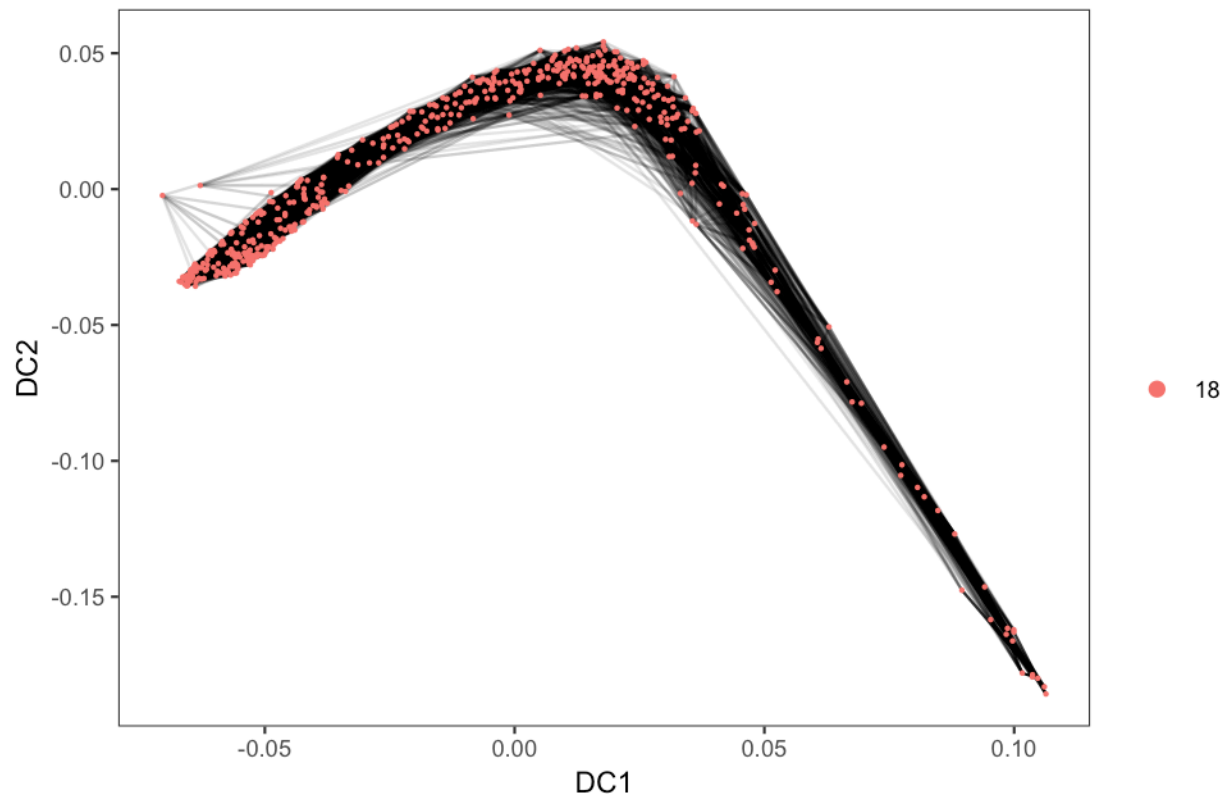
The diffusion map seems to strongly represent the differentiation process. Since this is a fairly simple trajectory, the trajectory is actually evidence in diffusion components 1 and 2 (DC1 vs. DC2).

```
# Make transition plots
plotDim(hydra.male, "res.1.5", transitions.plot = 3000, plot.title = "Transitions visualized on tSNE")
```



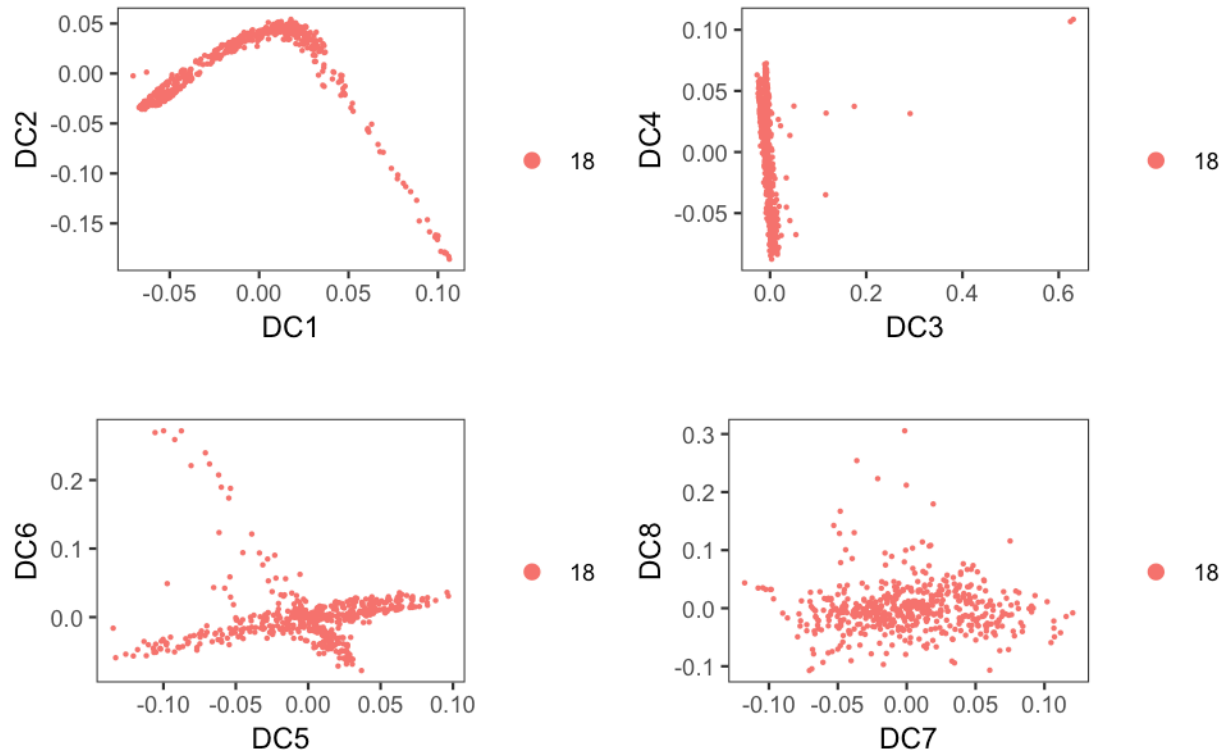
```
plotDim(hydra.male, "res.1.5", transitions.plot = 5000, reduction.use = "dm", plot.title = "Transitions visualized on diffusion")
```

## Transitions visualized on diffusion map



```
# 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")
```

## Pairs of Diffusion Components

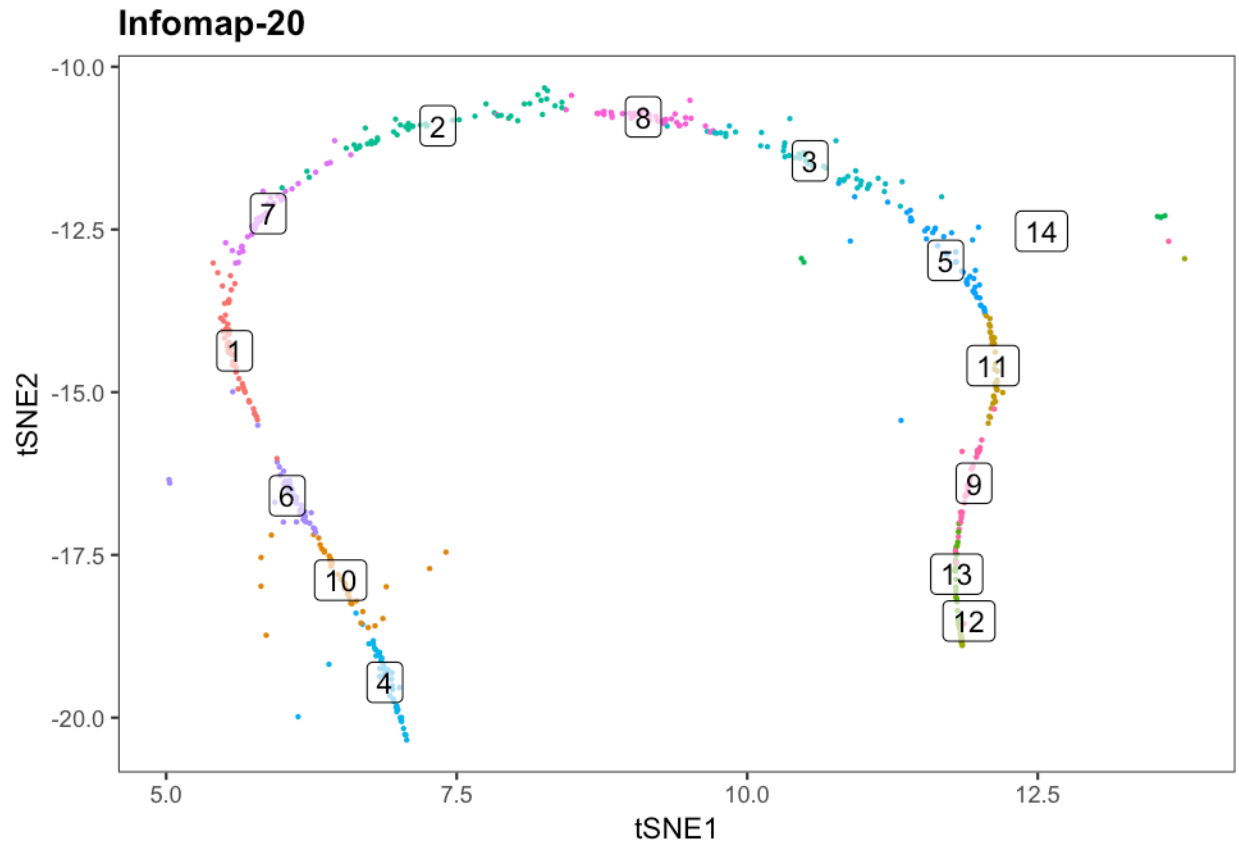


## Calculate pseudotime

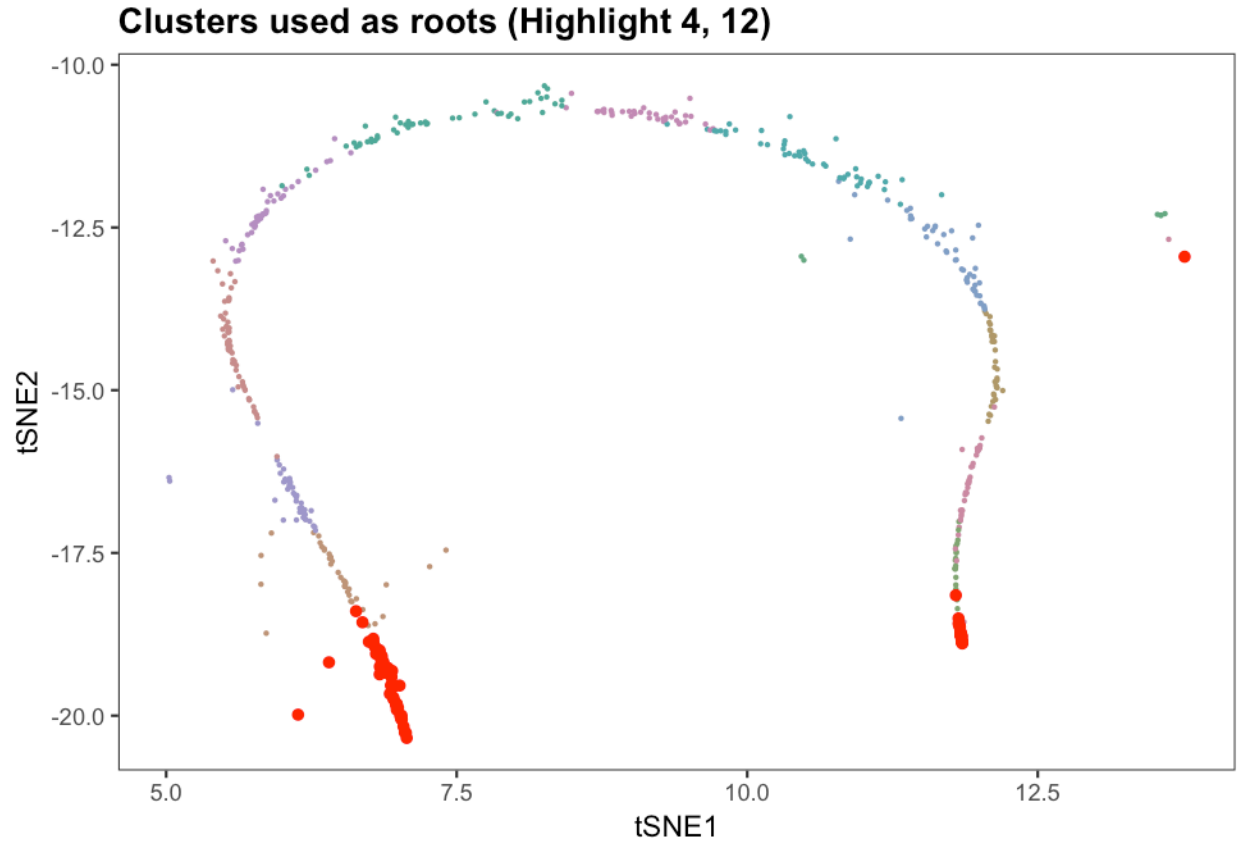
Here, we calculate pseudotime from both ends of the differentiation process (starting both at the beginning and at the end) and average the two pseudotimes to produce a single ordering across the entire male germline lineage.

We used clusters 4 and 12 from our clustering as the two ends.

```
plotDim(hydra.male, "Infomap-20", label.clusters = T, legend = F)
```



```
plotDimHighlight(hydra.male, "Infomap-20", cluster = c("4", "12"), plot.title = "Clusters used as roots",  
  legend = F)
```



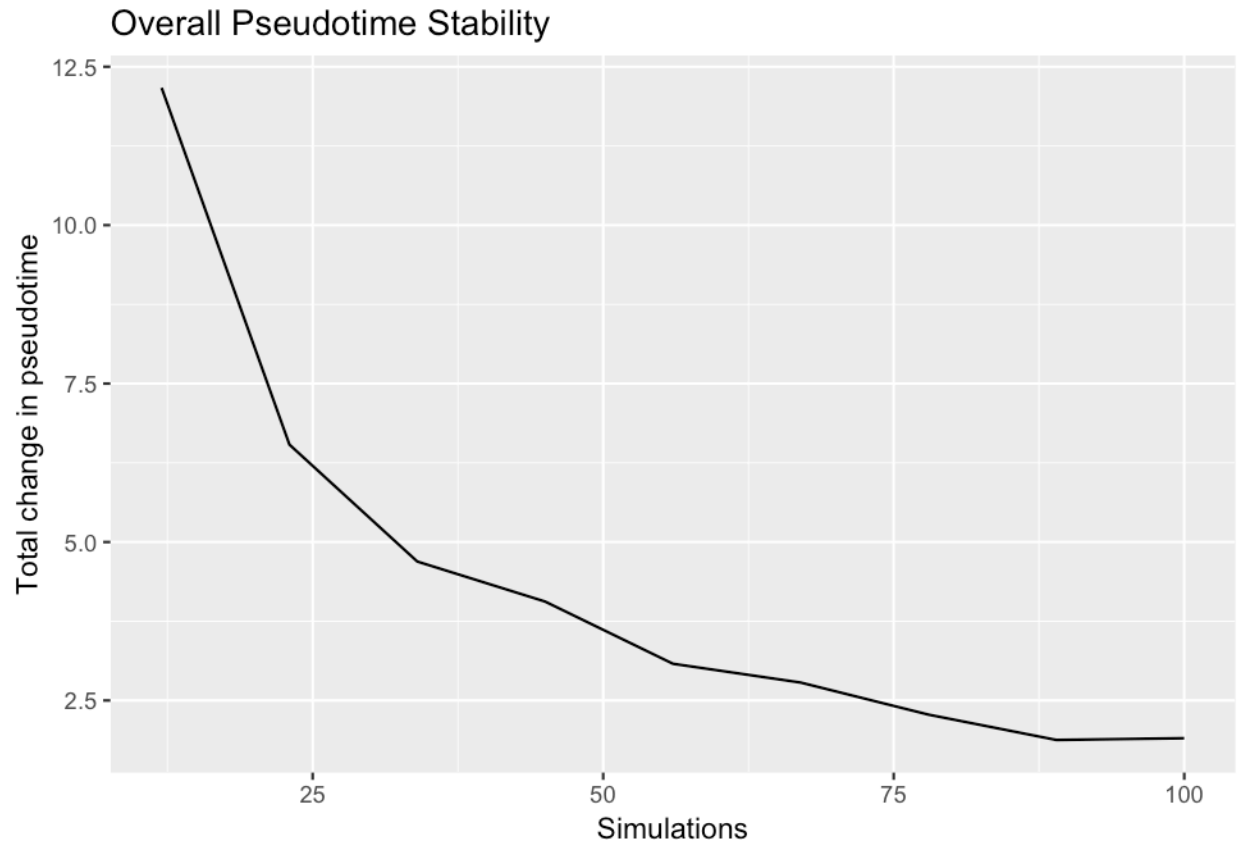
Since the simulation process is stochastic, we load our previously calculated results for consistency, but the following commands were run previously:

```
# Perform the 'flood' simulations to determine cells' pseudotime
male.flood.4 <- floodPseudotime(hydra.male, root.cells = cellsInCluster(hydra.male,
  "Infomap-20", "4"), n = 100, minimum.cells.flooded = 2, verbose = T)
male.flood.12 <- floodPseudotime(hydra.male, root.cells = cellsInCluster(hydra.male,
  "Infomap-20", "12"), n = 100, minimum.cells.flooded = 2, verbose = T)
```

We then 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.4, floods.name = "pseudotime.4")
hydra.male <- floodPseudotimeProcess(hydra.male, male.flood.12, floods.name = "pseudotime.12")

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

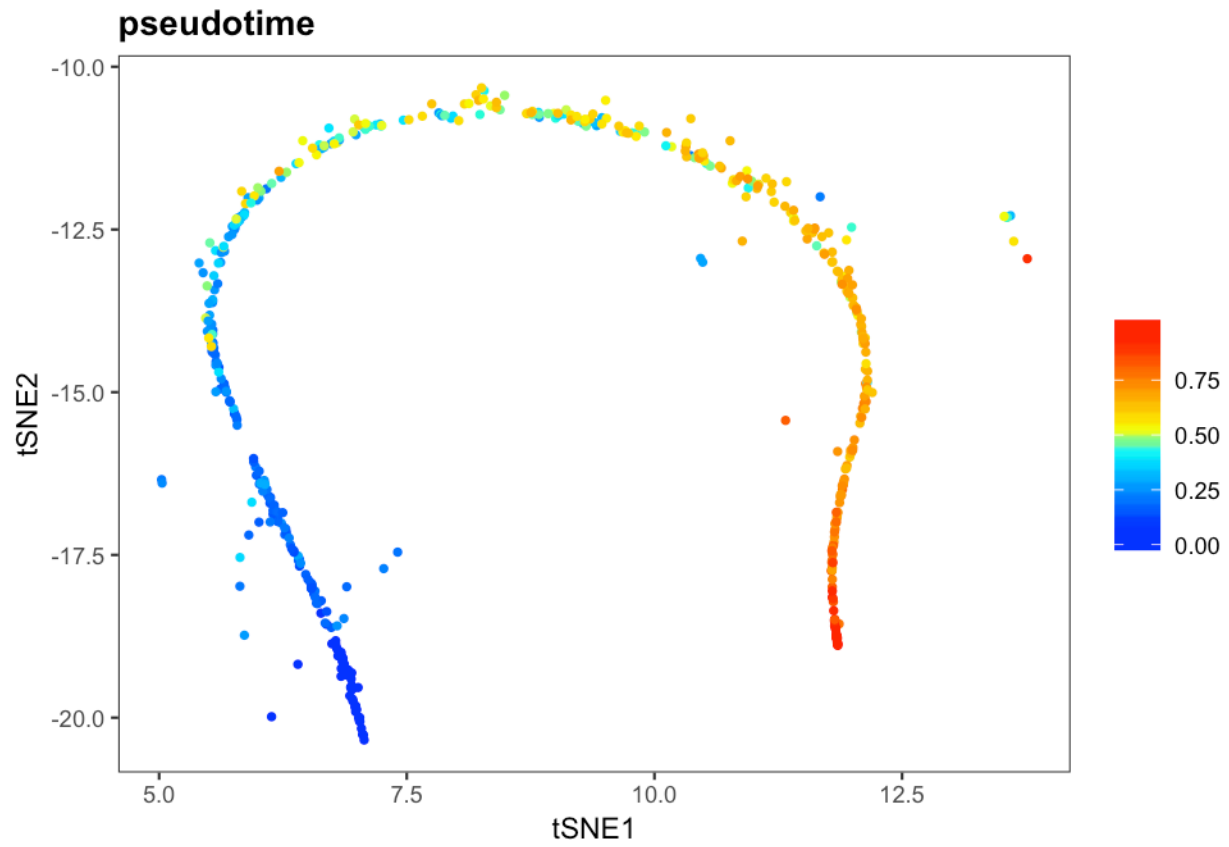


```
# Combine the two into a single pseudotime
hydra.male@pseudotime$pseudotime.4.norm <- hydra.male@pseudotime$pseudotime.4/max(hydra.male@pseudotime$pseudotime.4)
hydra.male@pseudotime$pseudotime.12.norm <- 1 - (hydra.male@pseudotime$pseudotime.12/max(hydra.male@pseudotime$pseudotime.12))
hydra.male@pseudotime$pseudotime <- rowMeans(hydra.male@pseudotime[, c("pseudotime.4.norm",
  "pseudotime.12.norm")])
```

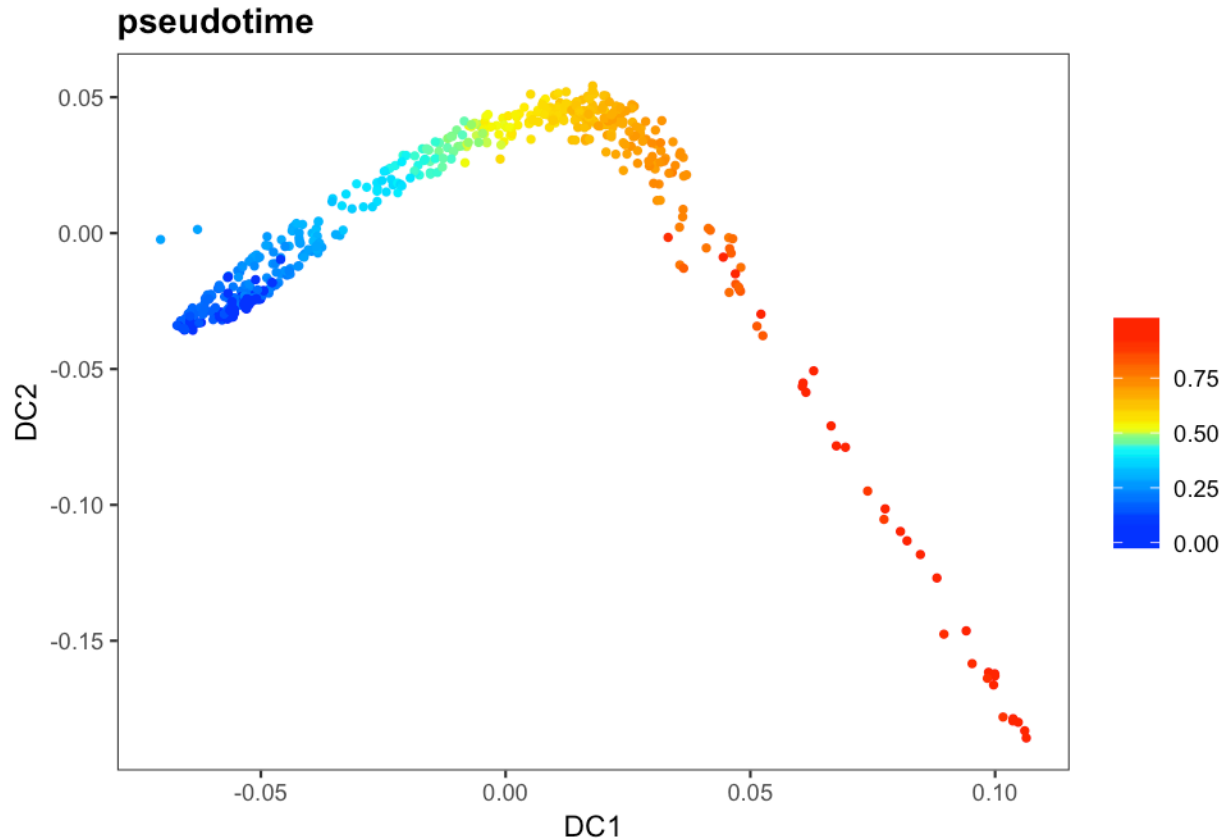
Pseudotime was calculated from early differentiation to late. It has good temporal ordering across this trajectory.

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





```
plotDim(hydra.male, "pseudotime", reduction.use = "dm")
```



## 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 <- Matrix::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 = 0.875) #.875 previously

## [1] "2018-11-01 14:50:18: Calculating moving window expression."
## [1] "2018-11-01 14:50:34: Generating un-scaled fits."
## [1] "2018-11-01 14:50:47: Generating scaled fits."
## [1] "2018-11-01 14:51:00: 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)

```

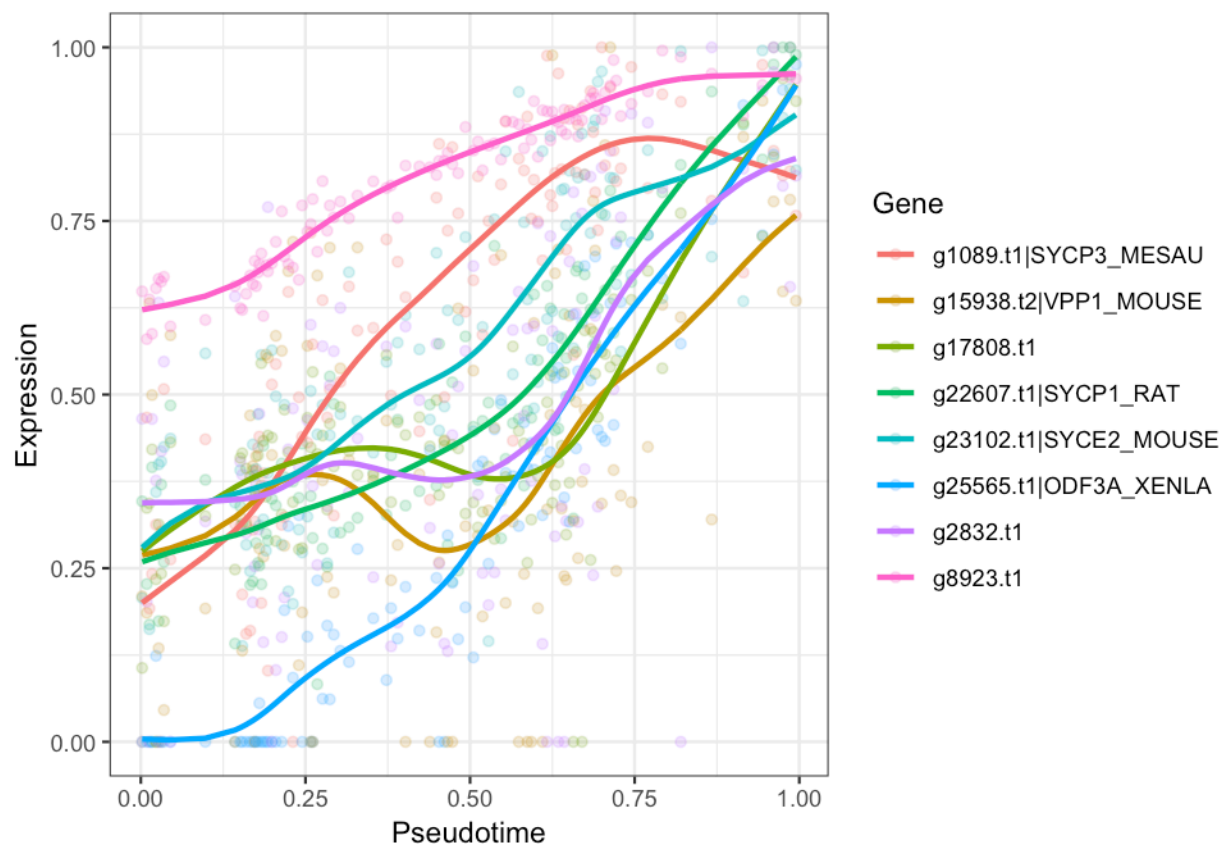
## Spline plots

We can then fit splines to genes' expression and plot their expression across the process of differentiation.

```

genes.spline.plot <- c("g22607.t1|SYCP1_RAT", "g23102.t1|SYCE2_MOUSE", "g1089.t1|SYCP3_MESAU",
" g25565.t1|ODF3A_XENLA", "g15938.t2|VPP1_MOUSE", "g2832.t1", "g17808.t1", "g8923.t1")
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/MaleGenome/MaleGenome-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 (Genome, 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/MaleGenome/Splines-MaleGenome.rds"))
write(varying.genes.5cell, file = paste0(main.path, "URD/MaleGenome/Genes-MaleGenome-Varying.txt"))
saveRDS(hydra.male, file = paste0(main.path, "URD/MaleGenome/Hydra_URD_MaleGenome.rds"))

```