SA15: Hydra URD Male Germline (Genome)

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

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

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."
```

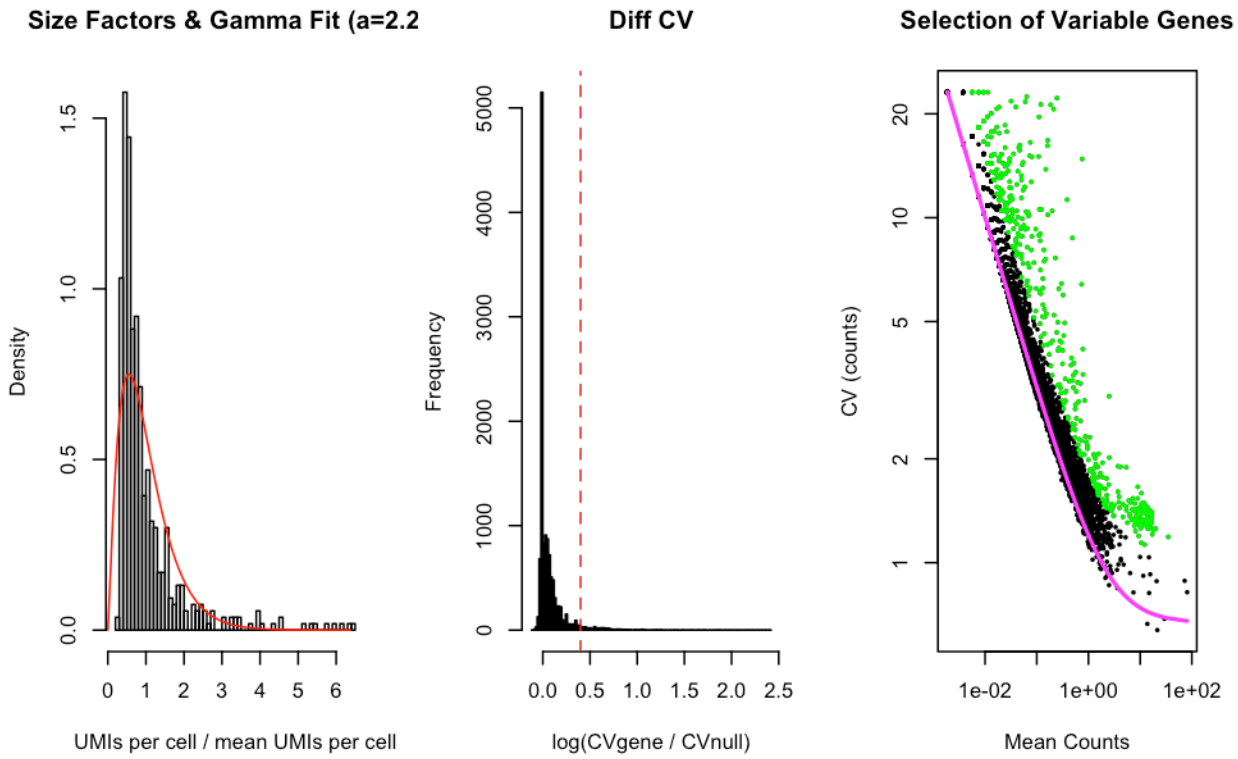
```
# Subset to cluster 18, which contains the male germline
hydra.male <- urdSubset(hydra.genome.urd, cells.keep = cellsInCluster(hydra.genome.urd,
    "res.1.5", "18"))

# 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 subsetted, 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).

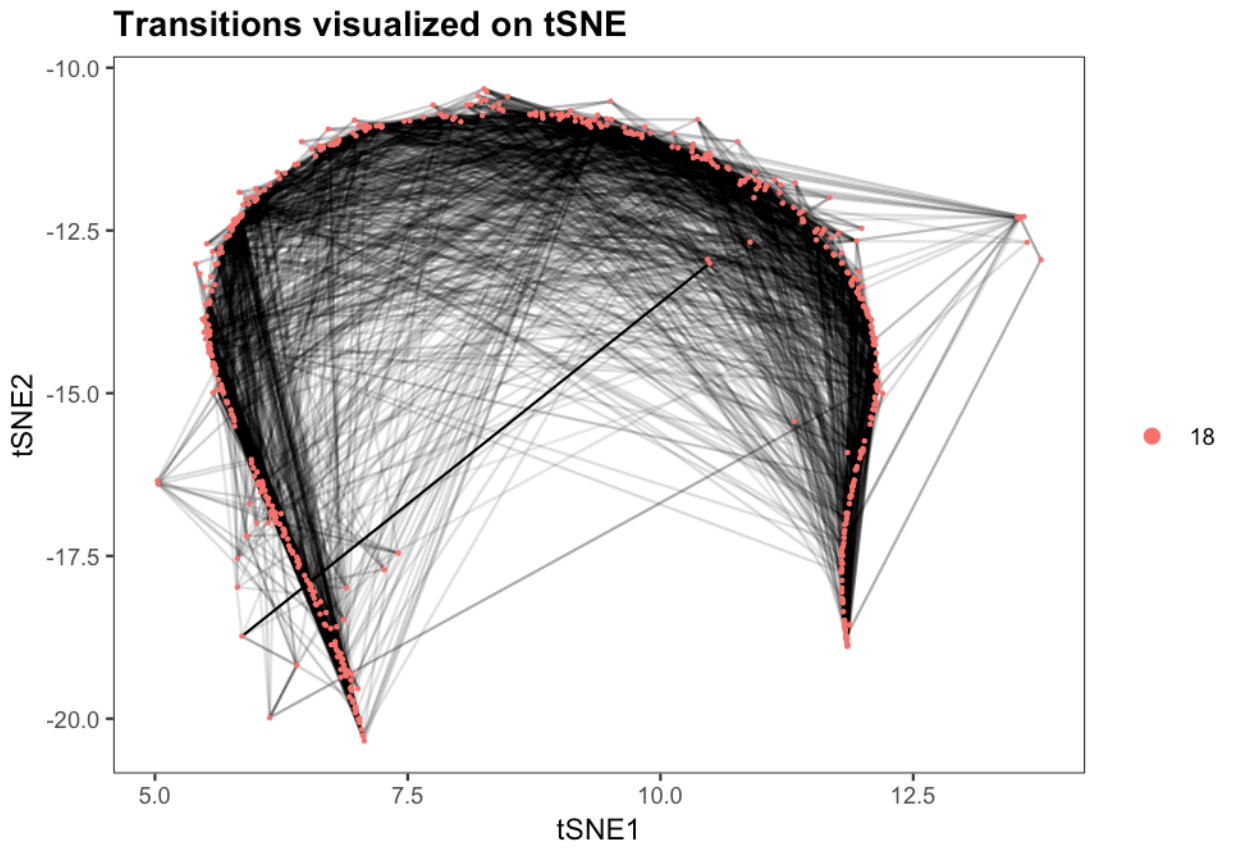
```
# Calculate diffusion map
hydra.male <- calcDM(hydra.male, knn = 75, sigma.use = 9.624249)
```

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

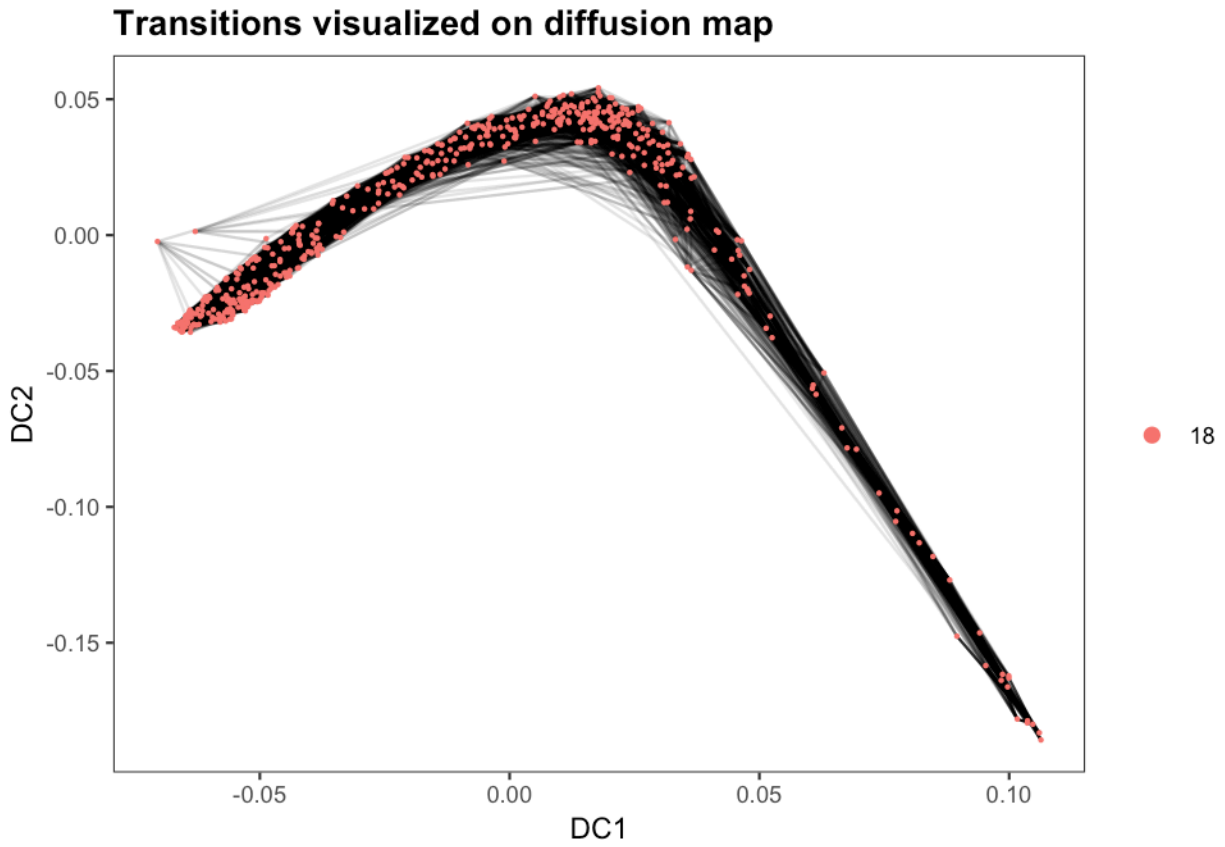
Evaluation of diffusion map

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).

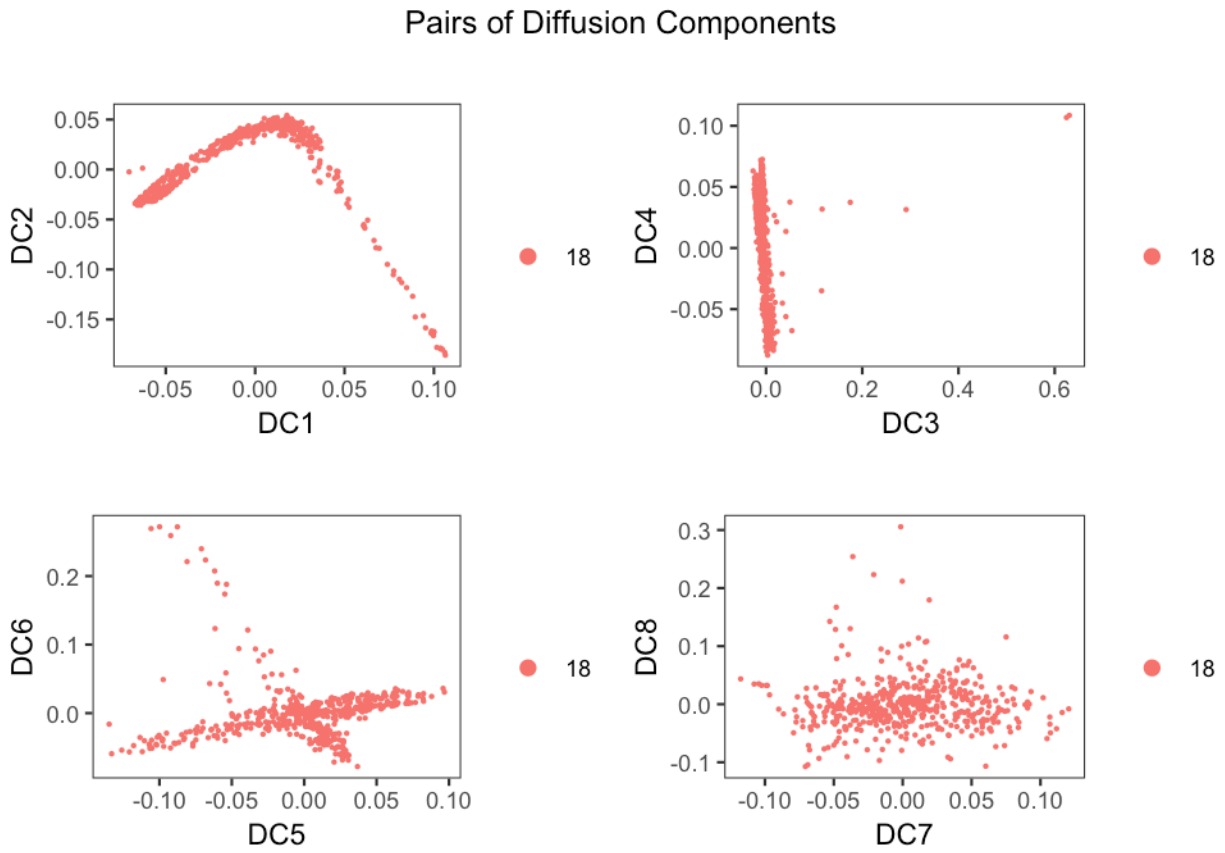
```
# Make transition plots
plotDim(hydra.male, "res.1.5", transitions.plot = 3000, plot.title = "Transitions visualized on tSNE")
```




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




```
# 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")
```

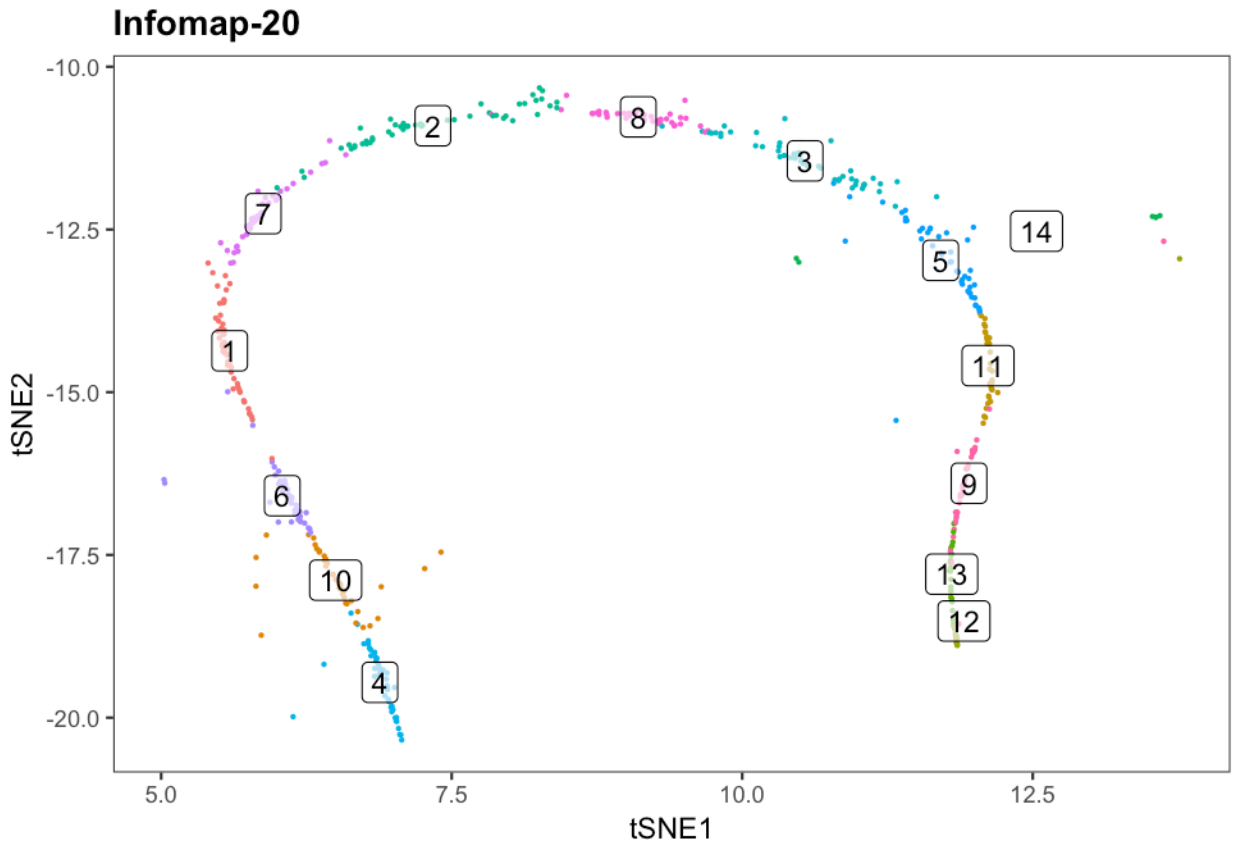


Calculate pseudotime

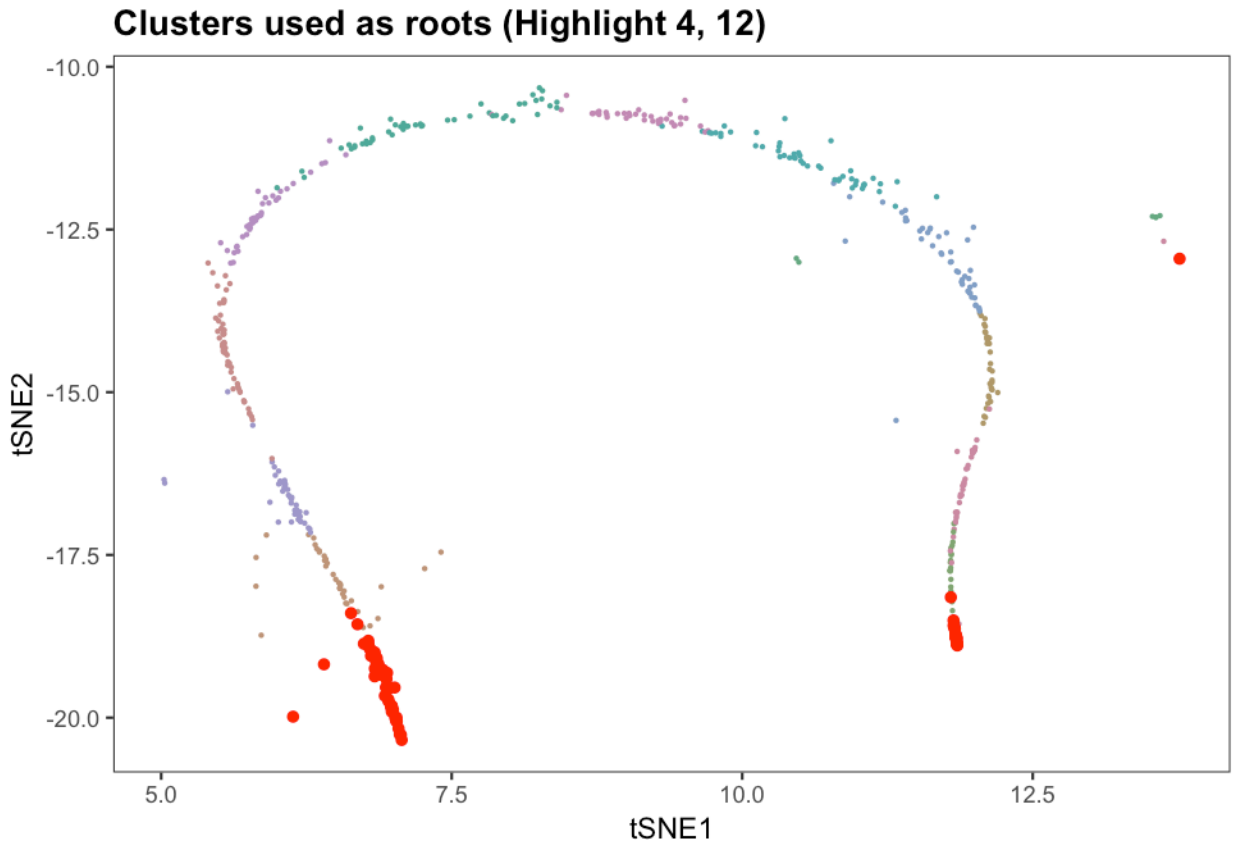
Here, we calculate pseudotime from both ends of the differentiation process (starting both at the beginning and at the end) and average the two pseudotimes to produce a single ordering across the entire male germline lineage.

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)
```

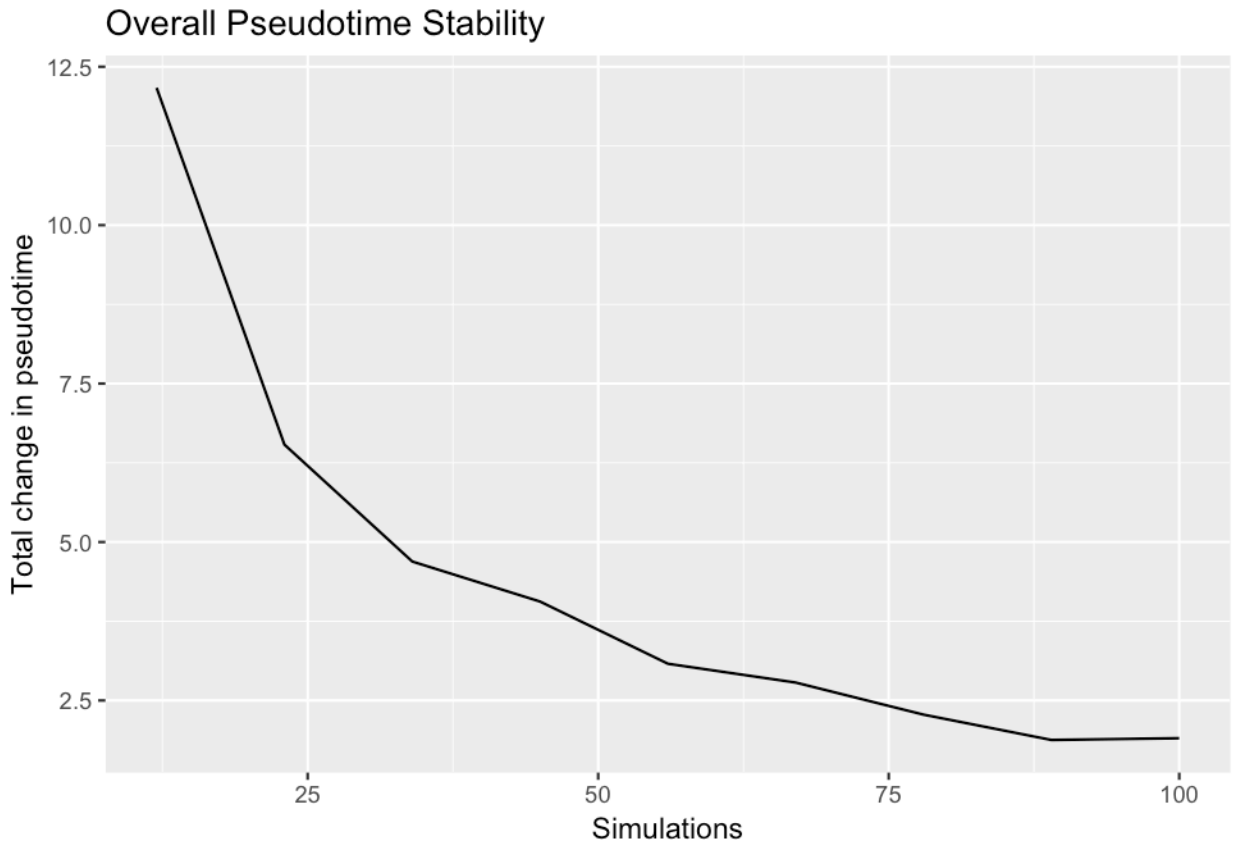


Since the simulation process is stochastic, we load our previously calculated results for consistency, but the following commands were run previously:

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

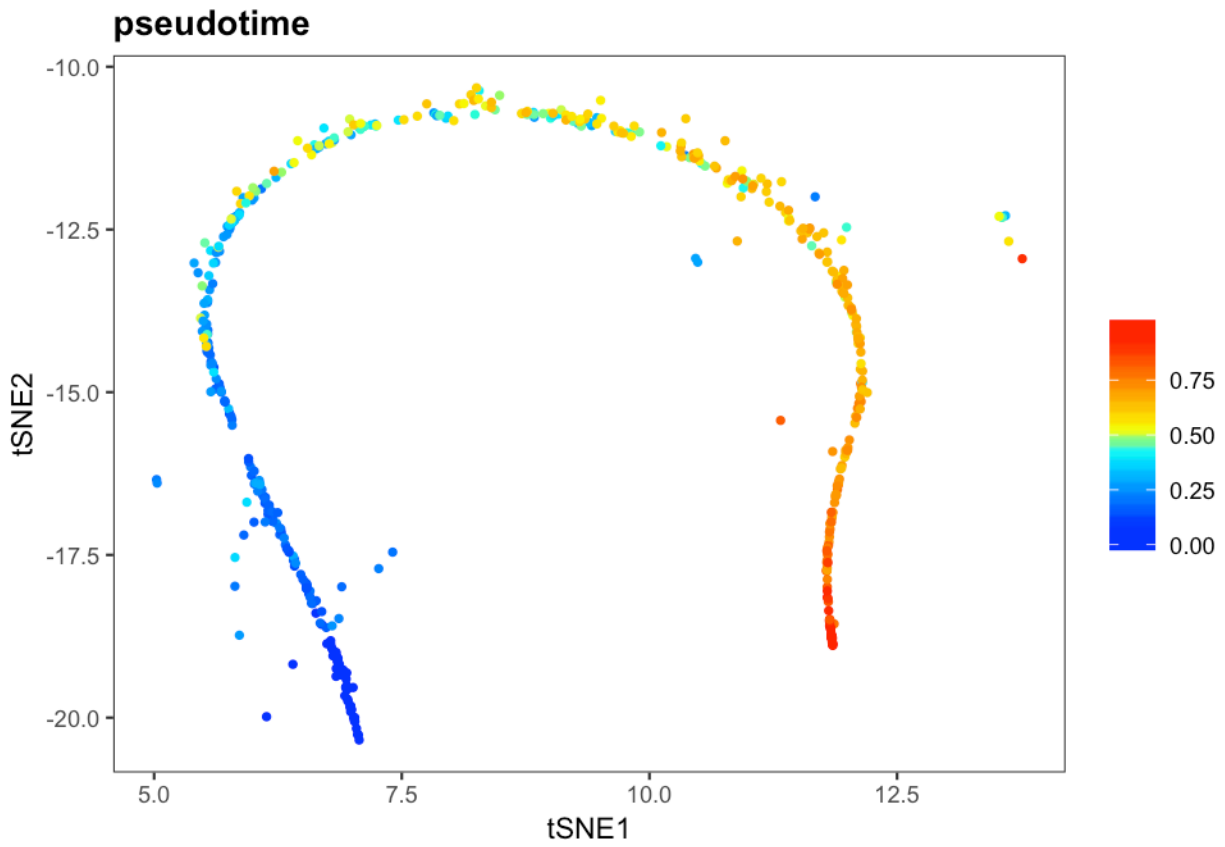
```
# Process the floods to derive pseudotime
hydra.male <- floodPseudotimeProcess(hydra.male, male.flood.4, floods.name = "pseudotime.4")
hydra.male <- floodPseudotimeProcess(hydra.male, male.flood.12, floods.name = "pseudotime.12")

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



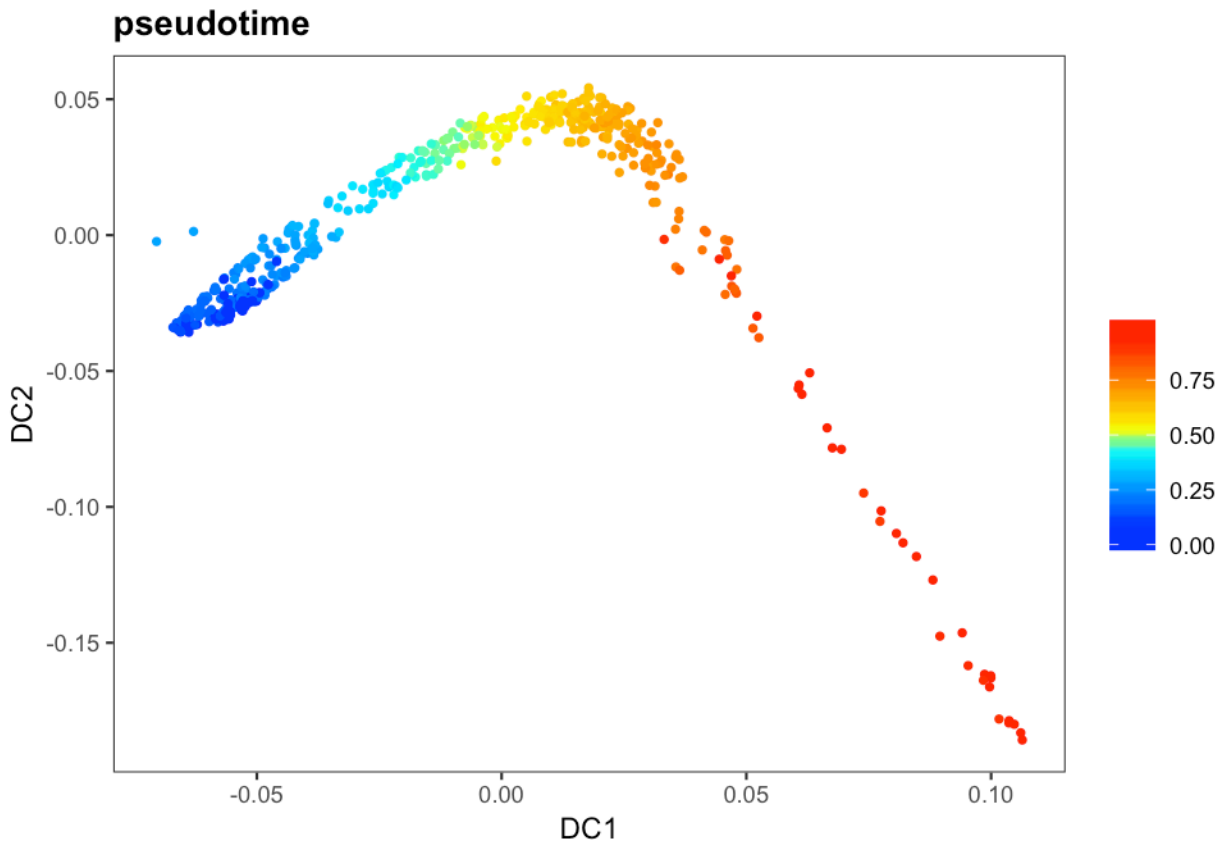

```
# Combine the two into a single pseudotime
hydra.male@pseudotime$pseudotime.4.norm <- hydra.male@pseudotime$pseudotime.4/max(hydra.male@pseudotime$pseudotime.4)
hydra.male@pseudotime$pseudotime.12.norm <- 1 - (hydra.male@pseudotime$pseudotime.12/max(hydra.male@pseudotime$pseudotime.12))
hydra.male@pseudotime$pseudotime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

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

```
## [1] "2019-03-20 06:05:00: Calculating moving window expression."
## [1] "2019-03-20 06:05:14: Generating un-scaled fits."
## [1] "2019-03-20 06:05:24: Generating scaled fits."
## [1] "2019-03-20 06:05:35: Reducing mean expression data to same dimensions as spline fits."
```

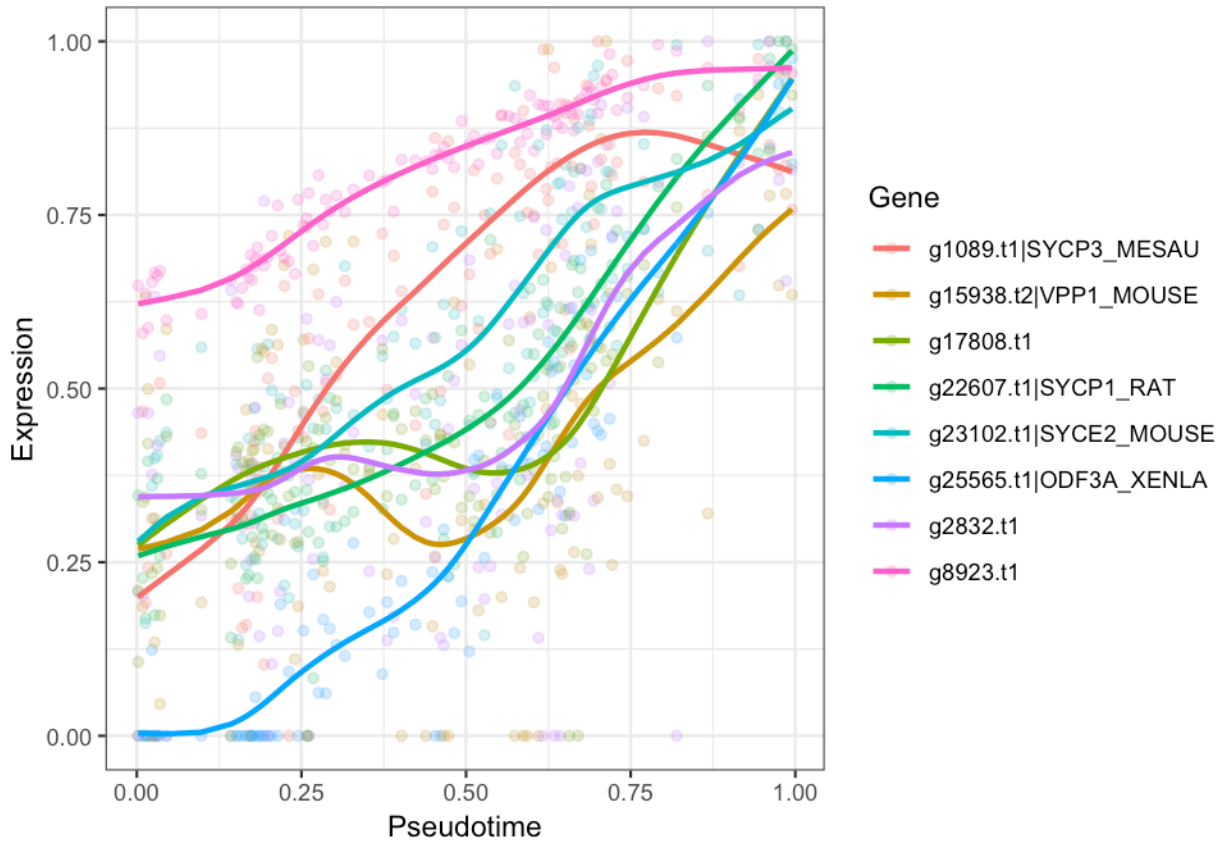
```
# Which genes change in their actual mean expression value by at least 0.5?
spline.change.real <- apply(expressed.spline.5cell$mean.smooth, 1, function(x) diff(range(x)))
genes.change.real <- names(which(spline.change.real >= 0.5))
```

```
# 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 <= 0.06))

# 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 * 0.1/0.06 +
    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 > 0.19))

# 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 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

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[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/MaleGenome/MaleGenome-Varying.pdf"), width = 17, height = 22)
# Plot the heatmap
gplots::heatmap.2(x = as.matrix(ss[varying.genes.5cell[h.ss$order], ]), Rowv = F,
    RowSideColors = h.ss.clust.col[h.ss$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("Male Germline (Genome) Expression", line = -1, adj = 0.48, cex.main = 4)
# Add tissue labels to bottom
title("Early", line = -103, adj = 0.08, cex.main = 2)
title("Late", 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/MaleGenome/MaleGenome-Cluster-Profiles.pdf"), width = 8.5,
    height = 11)
plotSmoothFit(so, sort(rownames(so$mean.expression)), scaled = T, plot.data = T,
    multiplot = T) + ggtitle("Male Germline (Genome) 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, 7, 9, 11, 13, 15)
cluster.starts <- c(1, head(cluster.ends, -1) + 1)

pdf(paste0(main.path, "URD/MaleGenome/MaleGenome-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 <- h.ss$order[which(h.ss.clust.ord[h.ss$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 = 0.75, margins = c(8, 15), lwid = c(0.3, 4),
        lhei = c(0.3, 4), labCol = NA)
    # Put a title on it
    title(paste0("Male Germline (Genome): Clusters ", cluster.starts[c], " to ",
        cluster.ends[c]), line = -1, adj = 0.48, cex.main = 4)
    # Add tissue labels to bottom
    title("Early", line = -103, adj = 0.075, cex.main = 2)
    title("Late", line = -103, adj = 0.9, cex.main = 2)
}
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/MaleGenome/MaleGenome-Genes-Varying.txt"))
# Save objects
saveRDS(expressed.spline.5cell, file = paste0(main.path, "URD/MaleGenome/Splines-MaleGenome.rds"))
saveRDS(hydra.male, file = paste0(main.path, "URD/MaleGenome/Hydra_URD_MaleGenome.rds"))
```