scoring cells by differentiation

Andy Sposato

The purpose of this notebook is to add differentation scores to Seurat objects. We do this by converting Seurat objects to URD objects, running URD, and adding the pseudotime information to a Seurat object's metadata as diff.score.

Generate the Seurat objects

In this example, I'll be using an unedited sample (yr1.6 or 20 mo.) and a double-edited sample (yr1.8 or 22 mo.)

If you have already generated objects or have downloaded the 20 mo. and 22 mo. objects from Andy's Github/Google Drive, skip to Run URD section. You have likely already generated your objectsas you'll want to see quality of your atlas before you proceed with amplifying lineage barcodes from 10X cDNA, but here are some steps just in case.

```r
# load required packages
library(dplyr)
library(Seurat)
# uncomment the next line if you need to install URD
# source("https://raw.githubusercontent.com/farrellja/URD/master/URD-Install.R")
suppressPackageStartupMessages(library(rgl))
## Warning: package 'rgl' was built under R version 3.4.3
suppressPackageStartupMessages(library(URD))
## Warning: package 'Matrix' was built under R version 3.4.2
knitr::opts_chunk$set(echo = TRUE)
rgl::setupKnitr()
library(scales)
library(cowplot)
```

```r
# read the 10X feature barcode matrix data
# these are the CellRanger outputs you'll need:
# filtered_feature_bc_matrix
#   barcodes.tsv.gz
#   features.tsv.gz
#   matrix.mtx.gz
yr1.6_A.data <- Read10X(data.dir = 'data/18827X2/filtered_feature_bc_matrix/')
yr1.6_A <- CreateSeuratObject(counts = yr1.6_A.data, project = 'yr1.6_A', min.cells = 3)
# mark mt- RNAs as mitochondrial RNA reads
yr1.6_A[['percent.mt']] <- PercentageFeatureSet(yr1.6_A, pattern = "mt-")
# add helpful metadata to your object
yr1.6_A$sub.sample <- "20mo"
yr1.6_A$age <- "20 mo."
yr1.6_A$condition <- "unedited"
# nFeature_RNA is the number of unique genes
# because transcription is quite high in the testis, we kept cells with >200 genes
# but fewer than n genes, where n represents the largest value of the third quartile
# of each dataset
yr1.6_A <- subset(yr1.6_A, subset = nFeature_RNA > 200 & nFeature_RNA < 4999)
# eliminate cells with greater than 5% mitochondrial reads
# when cells are lysed during dissociation, mitochondrial reads make up a large proportion
# of the reads captured. We subset based on this metric to avoid grouping dying cells
# into our data.
yr1.6_A <- subset(yr1.6_A, subset = percent.mt < 5)
yr1.6_A<- NormalizeData(yr1.6_A)
yr1.6_A <- FindVariableFeatures(yr1.6_A)
yr1.6_A <- ScaleData(yr1.6_A)
yr1.6_A <- RunPCA(yr1.6_A)

yr1.6_B.data <- Read10X(data.dir = 'data/18827X4/filtered_feature_bc_matrix/')
yr1.6_B <- CreateSeuratObject(counts = yr1.6_B.data, project = 'yr1.6_B', min.cells = 3)
yr1.6_B <- subset(yr1.6_B, subset = nFeature_RNA > 200 & nFeature_RNA < 6373)
yr1.6_B[['percent.mt']] <- PercentageFeatureSet(yr1.6_B, pattern = "mt-")
yr1.6_B$sub.sample <- "20mo"
yr1.6_B$age <- "20 mo."
yr1.6_B$condition <- "unedited"
yr1.6_B <- subset(yr1.6_B, subset = percent.mt < 5)
yr1.6_B <- NormalizeData(yr1.6_B)
yr1.6_B <- FindVariableFeatures(yr1.6_B)
yr1.6_B <- ScaleData(yr1.6_B)
yr1.6_B <- RunPCA(yr1.6_B)

# I run UMAP after integrating samples
# here, yr1.6_A and yr1.6_B are cells from the same testis that were split across
# 2 channels of 10X
yr1.6.anchors <- FindIntegrationAnchors(object.list = list(yr1.6_A, yr1.6_B),
                                        reduction = "cca", dims = 1:10)
unedited <- IntegrateData(anchorset = yr1.6.anchors, dims = 1:10)
DefaultAssay(unedited) <- "integrated"
unedited <- ScaleData(unedited, assay = "integrated")
unedited <- ScaleData(unedited, assay = "RNA")
unedited <- RunPCA(unedited, features = VariableFeatures(object = unedited))
# You can use ElbowPlot() to determine a good range for dimensions to use.
# This will likely be between 10 and 20.
unedited <- RunUMAP(unedited, reduction = 'pca', dims = 1:10)
unedited <- FindNeighbors(unedited, reduction = 'pca', dims = 1:10)
unedited <- FindClusters(unedited, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2615
## Number of edges: 87303
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9067
## Number of communities: 12
## Elapsed time: 0 seconds
```

```
# I remove these intermediate objects since I no longer need them to keep the "mess"
# in Global Environment manageable
rm(yr1.6_A.data, yr1.6_B.data, yr1.6_A, yr1.6_B, yr1.6.anchors)
```

```r
yr1.8_A.data <- Read10X(data.dir = 'data/18827X1/filtered_feature_bc_matrix/')
# create Seurat object
yr1.8_A <- CreateSeuratObject(counts = yr1.8_A.data, project = 'yr1.8_A', min.cells = 3)
yr1.8_A[['percent.mt']] <- PercentageFeatureSet(yr1.8_A, pattern = "mt-")
yr1.8_A$sub.sample <- "22mo"
yr1.8_A$age <- "22 mo."
yr1.8_A$condition <- "edited"
yr1.8_A <- subset(yr1.8_A, subset = nFeature_RNA > 200 & nFeature_RNA < 6962)
yr1.8_A <- subset(yr1.8_A, subset = percent.mt < 5)
yr1.8_A <- NormalizeData(yr1.8_A)
yr1.8_A <- FindVariableFeatures(yr1.8_A)
yr1.8_A <- ScaleData(yr1.8_A)
yr1.8_A <- RunPCA(yr1.8_A)

yr1.8_B.data <- Read10X(data.dir = 'data/18827X3/filtered_feature_bc_matrix/')
yr1.8_B <- CreateSeuratObject(counts = yr1.8_B.data, project = 'yr1.8_B', min.cells = 3)
yr1.8_B[['percent.mt']] <- PercentageFeatureSet(yr1.8_B, pattern = "mt-")
yr1.8_B$sub.sample <- "22mo"
yr1.8_B$age <- "22 mo."
yr1.8_B$condition <- "edited"
yr1.8_B <- subset(yr1.8_B, subset = nFeature_RNA > 200 & nFeature_RNA < 7058)
yr1.8_B <- subset(yr1.8_B, subset = percent.mt < 5)
yr1.8_B <- NormalizeData(yr1.8_B)
yr1.8_B <- FindVariableFeatures(yr1.8_B)
yr1.8_B <- ScaleData(yr1.8_B)
yr1.8_B <- RunPCA(yr1.8_B)

yr1.8.anchors <- FindIntegrationAnchors(object.list = list(yr1.8_A, yr1.8_B),
                                        reduction = "cca", dims = 1:15)
edited <- IntegrateData(anchorset = yr1.8.anchors, dims = 1:15)
DefaultAssay(edited) <- "integrated"
edited <- ScaleData(edited, assay = "integrated")
edited <- ScaleData(edited, assay = "RNA")
edited <- RunPCA(edited, features = VariableFeatures(object = edited))
edited <- RunUMAP(edited, reduction = 'pca', dims = 1:15)
edited <- FindNeighbors(edited, reduction = 'pca', dims = 1:15)
edited <- FindClusters(edited, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3289
## Number of edges: 116393
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9113
## Number of communities: 11
## Elapsed time: 0 seconds
```

```r
rm(yr1.8_A.data, yr1.8_B.data, yr1.8_A, yr1.8_B, yr1.8.anchors)
```

Run URD

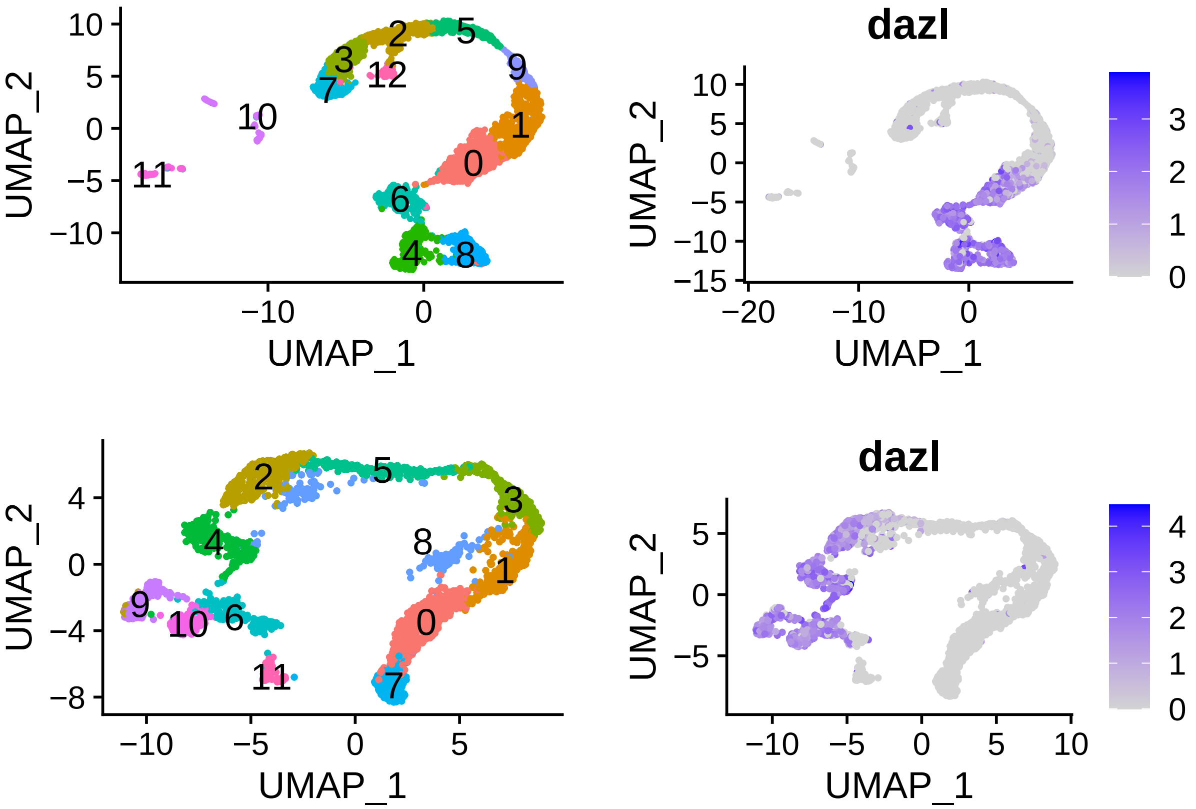
URD is a pseudotime approach that assigns a differentiation score to each individual cell in your object. It was developed by Jeff Farrell who helped us get started with this. We used an approach similar to the hydra male transcriptome project from Celina Juliano's lab.

The more cells you have, the longer URD takes to run. You may consider running this program as a scheduled job through CHPC.

```
unedited <- readRDS("data/objects/mo22.rds")
edited <- readRDS("data/objects/mo20.rds")
```

We're looking at ddx4 as a marker for spermatogonia. The most undifferentiated cells will express this marker well and help us determine where the "beginning" of differentiation is. This will be important for setting the root cells when running URD. It's worth noting that URD works best when a clear continuum of cells exists.

```
p1 <- DimPlot(unedited, label = T, label.size = 5) + NoLegend()
p2 <- FeaturePlot(unedited, "dazl")
p3 <- DimPlot(edited, label = T, label.size = 5) + NoLegend()
p4 <- FeaturePlot(edited, "dazl")
plot_grid(p1, p2, p3, p4, ncol = 2)
```



This seuratV3ToURD function was written by Jeff. It converts a Seurat object to an URD object. Feel free to open it up in a text editor to review it.

```
source("bin/SeuratV3ToURD.R")
unedited_URD <- seuratV3ToURD(unedited)
```

```
## [1] "Marchenko-Pastur eigenvalue null upper bound: 5.20659123106629"
## [1] "16 PCs have larger eigenvalues."
```

```r
edited_URD <- seuratV3ToURD(edited)
```

```
## [1] "Marchenko-Pastur eigenvalue null upper bound: 4.59441932520585"
## [1] "15 PCs have larger eigenvalues."
```

```r
clusters <- as.factor(unedited_URD@group.ids$seurat_clusters)
unedited_URD@meta$seurat_clusters <- clusters

var.genes <- findVariableGenes(unedited_URD, set.object.var.genes=F, diffCV.cutoff=0.3,
                               mean.min=.005, mean.max=100, main.use="", do.plot = F)
unedited_URD@var.genes <- var.genes

# the knn should be set to the square root of the total number of cells in your object
unedited_URD <- calcDM(unedited_URD, knn = 57)
```

```
## [1] "destiny determined an optimal global sigma of 17.51"
```

```
## Warning in DiffusionMap(data.use, sigma = sigma.use, k = knn, n_eigs =
## dcs.store, : You have 2924 genes. Consider passing e.g. n_pcs = 50 to speed up
## computation.
```

```r
# the root cells for the forward calculation should be your spermatogonia clusters
# you can also define a vector of cells instead of whole clusters
root.cells <- cellsInCluster(unedited_URD, "seurat_clusters", c("4", "8"))
unedited_URD.floods <- floodPseudotime(unedited_URD, root.cells = root.cells, n=20,
                                       minimum.cells.flooded = 2, verbose=F)
unedited_URD <- floodPseudotimeProcess(unedited_URD, unedited_URD.floods,
                                       floods.name="pseudotime.fwd")

# the root cells for the reverse calculation should be the most mature spermatids cluster
root.cells <- cellsInCluster(unedited_URD, "seurat_clusters", c("7"))
unedited_URD.floods <- floodPseudotime(unedited_URD, root.cells = root.cells, n=20,
                                       minimum.cells.flooded = 2, verbose=F)
unedited_URD <- floodPseudotimeProcess(unedited_URD, unedited_URD.floods,
                                       floods.name="pseudotime.rev")

# normalize values
unedited_URD@pseudotime$pseudotime.fwd.norm <- unedited_URD@pseudotime$pseudotime.fwd/
  max(unedited_URD@pseudotime$pseudotime.fwd)
# 1 - reverse scores
unedited_URD@pseudotime$pseudotime.rev.norm <- 1 - unedited_URD@pseudotime$pseudotime.rev/
  max(unedited_URD@pseudotime$pseudotime.rev)

# take the average of the normalized forward and reverse
unedited_URD@pseudotime$pseudotime <- rowMeans(unedited_URD@pseudotime
                                               [,c("pseudotime.rev.norm",
                                                   "pseudotime.fwd.norm")])

# rescale so differentiation scores are on a 0 to 1 range
unedited_URD@pseudotime$pseudotime.scaled <- rescale(unedited_URD@pseudotime$pseudotime,
                                                     to = c(0,1))

# save the URD object
# later we'll pull pseudotime.scaled from it for transfer to the Seurat object
```

```r
saveRDS(unedited_URD, "data/objects/unedited_URD.rds")
rm(unedited_URD.floods)
```

```r
clusters <- as.factor(edited_URD@group.ids$seurat_clusters)
edited_URD@meta$seurat_clusters <- clusters

var.genes <- findVariableGenes(edited_URD, set.object.var.genes=F, diffCV.cutoff=0.3,
                               mean.min=.005, mean.max=100, main.use="", do.plot = F)
edited_URD@var.genes <- var.genes

# the knn should be set to the square root of the total number of cells in your object
edited_URD <- calcDM(edited_URD, knn = 51)
```

```
## [1] "destiny determined an optimal global sigma of 16.927"
```

```
## Warning in DiffusionMap(data.use, sigma = sigma.use, k = knn, n_eigs =
## dcs.store, : You have 2503 genes. Consider passing e.g. n_pcs = 50 to speed up
## computation.
```

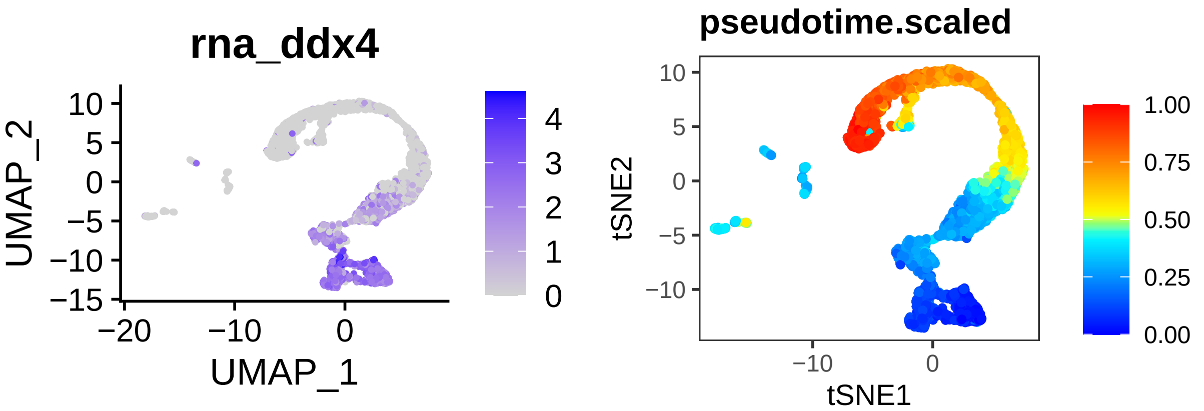
```r
# the root cells for the forward calculation should be your spermatogonia clusters
root.cells <- cellsInCluster(edited_URD, "seurat_clusters", c("9", "10"))
edited_URD.floods <- floodPseudotime(edited_URD, root.cells = root.cells, n=20,
                                     minimum.cells.flooded = 2, verbose=F)
edited_URD <- floodPseudotimeProcess(edited_URD, edited_URD.floods,
                                     floods.name="pseudotime.fwd")

# the root cells for the reverse calculation should be the most mature spermatids cluster
root.cells <- cellsInCluster(edited_URD, "seurat_clusters", c("7"))
edited_URD.floods <- floodPseudotime(edited_URD, root.cells = root.cells, n=20,
                                     minimum.cells.flooded = 2, verbose=F)
edited_URD <- floodPseudotimeProcess(edited_URD, edited_URD.floods,
                                     floods.name="pseudotime.rev")

# normalize values
edited_URD@pseudotime$pseudotime.fwd.norm <- edited_URD@pseudotime$pseudotime.fwd/
  max(edited_URD@pseudotime$pseudotime.fwd)
edited_URD@pseudotime$pseudotime.rev.norm <- 1 -  edited_URD@pseudotime$pseudotime.rev/
  max(edited_URD@pseudotime$pseudotime.rev)

# take the average of the normalized forward and reverse
edited_URD@pseudotime$pseudotime <- rowMeans(edited_URD@pseudotime
                                             [,c("pseudotime.rev.norm",
                                                 "pseudotime.fwd.norm")])

# rescale so differentiation scores are on a 0 to 1 range
edited_URD@pseudotime$pseudotime.scaled <- rescale(edited_URD@pseudotime$pseudotime,
                                                   to = c(0,1))

# save the object - later we'll pull pseudotime.scaled from it for transfer to the Seurat object
saveRDS(edited_URD, "data/objects/edited_URD.rds")
rm(edited_URD.floods)
```

```r
#p1 <- DimPlot(unedited, group.by = "seurat_clusters", label = T) + NoLegend()
p2 <- FeaturePlot(unedited, "rna_ddx4")
#p3 <- FeaturePlot(unedited, "rna_gsdf")
```

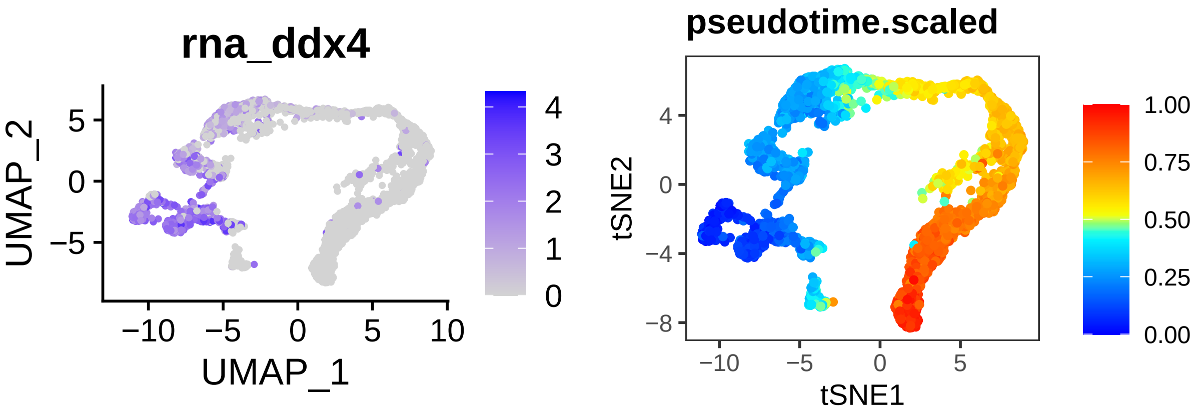
```r
#p4 <- FeaturePlot(unedited, "rna_tssk6")
#p5 <- plotDim(unedited_URD, "pseudotime.fwd")
#p6 <- plotDim(unedited_URD, "pseudotime.rev")
#p7 <- plotDim(unedited_URD, "pseudotime")
p8 <- plotDim(unedited_URD, "pseudotime.scaled")
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## i The deprecated feature was likely used in the URD package.
##   Please report the issue to the authors.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
plot_grid(p2, p8, ncol = 2, nrow = 2)
```



```r
# p1 <- DimPlot(edited, group.by = "seurat_clusters", label = T)
p2 <- FeaturePlot(edited, "rna_ddx4")
# p3 <- FeaturePlot(edited, "rna_gsdf")
# p4 <- FeaturePlot(edited, "rna_tssk6")
# p5 <- plotDim(edited_URD, "pseudotime.fwd")
# p6 <- plotDim(edited_URD, "pseudotime.rev")
# p7 <- plotDim(edited_URD, "pseudotime")
p8 <- plotDim(edited_URD, "pseudotime.scaled")
plot_grid(p2, p8, ncol = 2, nrow = 2)
```



Add pseudotime information from URD objects to Seurat objects

```r
unedited_diffs <- as.vector(unedited_URD@pseudotime$pseudotime.scaled)
edited_diffs <- as.vector(edited_URD@pseudotime$pseudotime.scaled)

unedited@meta.data$diff.score <- unedited_diffs
edited@meta.data$diff.score <- edited_diffs

saveRDS(unedited, "data/objects/unedited_w_diffscores.rds")
saveRDS(edited, "data/objects/edited_w_diffscores.rds")
```