Process Foetal data: A single-cell comparison of adult and foetal human epicardium defines the age-associated changes in epicardial activity

Vincent Knight-Schrijver

31 January 2023

Introduction

We will continue the analysis here by loading in the foetal dataset generated in the previous report. This step is focussed on aligning foetal data characteristics with those found in the adult data ready for integrating our human heart stages by going through some quality control steps.

1. Trimming and removing bad cells
2. Modelling and removal of erythrocytes
3. Clustering and subsampling

As usual, let's pick up our functions and packages.

```
source("config.r")
library(rmarkdown)
    GetPackages=F
func.dir="~/Documents/Rlib/functions"
source("scripts/packages.r")
```

Processing and bad cell calling

We will remove all barcodes with a depth of above 15,000 as given by the boundary of depth in the adult dataset. First we need to load in our previously defined objects as well as the annotations dataframe we put together prior to this project.

```
########################################################################
# ### this is the sparse matrix we made earlier
########################################################################
# ### Reading in previous data for decontamination
  foetal.all.matrix = Read10X(data.dir = "../G_Foetal_Epicardium/data/foetal_all")

# Annotations for this project
  annotations = readRDS("../G_Foetal_Epicardium/output/v_2/annotations/foetal_all_annotations_version4.rds")

# QC filter boolean vector from foetal_preprocess script
  foetal.cells2remove = read.csv(
    file.path("output/v_3/foetal_RBC_removal", "../foetal_preprocess", "preprocess_cells2remove.csv"
  ), head=F)[,2]
  names(foetal.cells2remove) = read.csv(
    file.path("output/v_3/foetal_RBC_removal", "../foetal_preprocess", "preprocess_cells2remove.csv"
  ), head=F)[,1]
########################################################################
```

Then we can begin cutting down the barcodes.

```
########################################################################
# ### Basic QC in line with adult - let's remove those high depth cells.
  x = foetal.all.matrix

  # take reads
```

```
    nReads <- colSums(x)
  # cell gene expressions
    nGenes <- colSums(x > 0)
  # Mitochondrial Gene expression
    mito.genes <- grep(pattern = "^MT-", rownames(x))
    nMito <- colSums(x[mito.genes,])/nReads*100
  # Top influencial genes as cell reads fraction?
    g.influencial.50 <- colSums(x[order(-rowSums(x))[1:50],]) / nReads * 100
  # Quality control data.frame
    QC.df <- data.frame("nReads"=normalise(log(nReads+1)), # total reads QC
                        "nGenes"=normalise(log(nGenes+1)), # Features QC
                        "percent.mito "=normalise(log(nMito+1)),  # Erroneous feature enrichment
                        "%.reads.in.top.50.exp.genes"=normalise(log(g.influencial.50+1))    # Fraction of reads in the top expression genes
                        )
  # Make PCA of QC parameters
     QC.PCA <- prcomp(QC.df)
  # SCoring
     QC.df$QC.score = rowSums(QC.df[,1:2])-rowSums(QC.df[,3:4])
     QC.df$rank = order(-QC.df$QC.score)

# plots
   par(mfrow=c(3,3), mar=c(4,4,3,0.2))
  # Low nReads
    plot(nReads,nGenes)

  # Low gene expression - remove below
    plot(log(nGenes)); abline(h=log(300))

  # High mitochondrial mRNA - remove above 10%
    plot(nMito); abline(h=15)

    cMito = nMito > 15
    cGenes = nGenes < 300
    cReads = nReads < 1000 | nReads > 15000
    foetal.cells2remove = (cMito | cGenes | cReads)

  # QC plots

    for(i in 1:4){plot(QC.PCA$x[,1], QC.PCA$x[,2], col=plotcols(QC.df[,i]), main=colnames(QC.df)[i], pch=16)}
  # and sum of QC criteria (good - bad)
    plot(QC.PCA$x[,1:2], col=plotcols(rowSums(QC.df[,1:2])-rowSums(QC.df[,3:4])), main="QC.score", pch=16)
    points(QC.PCA$x[,1:2], col=(0:1)[as.factor(QC.df$QC.score > 0)], main="Failure.criteria", pch=16, cex=0.6)
    plot(QC.PCA$x[,1:2], col=as.factor(foetal.cells2remove), main="cells removed (red)", pch=16)

  write.table(data.frame(foetal.cells2remove),
    file=file.path("output/v_3/foetal_preprocess",  "preprocess_cells2remove.csv"),
    sep=",", row.names=T, col.names=F, quote=F
  )
  foetal.all.matrix = foetal.all.matrix[,!foetal.cells2remove]

#####################################################################
```

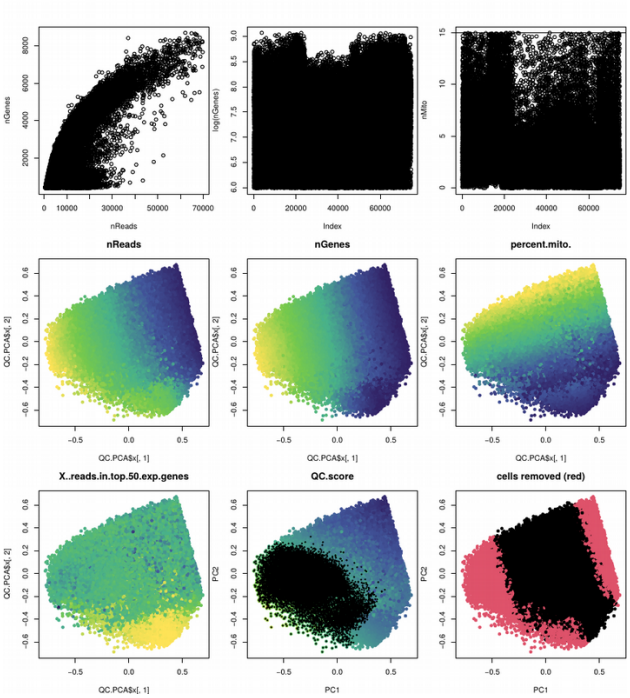

Figure 1: QC metrics across all foetal barcodes, QC PCA (PC1 / PC2), and adult-aligned cutoffs.

In figure 1 we see that we remove a lot of potentially high-quality foetal cells by aligning QC thresholds to the adult dataset. We also see in panel one, a distribution of highish UMI, but low gene expression content cells. This is the first indication of a strange cell type that we will highlight specifically as erythrocytes in the next step. Feel free to restart R as we'll be relabelling objects in the next step and pulling this vector back in.

Erythrocyte modelling and removal

We can quite evaluate erythrocytes on a single-cell basis using asimple classification model. In a previous analysis I experienced this aspect of our foetal heart scRNA-seq datasets and came up with this solution using the mean expression of a set of haemoglobulin genes as a variable for a two-compartment gaussian-mixture model (GMM). We can read in our datasets again as well as the annotations and previously filtered cells to apply onto our matrix.

```
######################################################################
# ### Reading in previous data for decontamination
  foetal.all.matrix = Read10X(data.dir = "../G_Foetal_Epicardium/data/foetal_all")

# Annotations for this project
  annotations = readRDS("../G_Foetal_Epicardium/output/v_2/annotations/foetal_all_annotations_version4.rds")

# QC filter boolean vector from foetal_preprocess script
  foetal.cells2remove = read.csv('
    file.path("output/v_3/foetal_RBC_removal", "../foetal_preprocess", "preprocess_cells2remove.csv"), head=F
  )[,2]
  names(foetal.cells2remove) = read.csv(
    file.path("output/v_3/foetal_RBC_removal", "../foetal_preprocess", "preprocess_cells2remove.csv"
  ), head=F)[,1]
```

From here will will follow the standard Seurat pipeline, including a quick integration across samples in order to generate clusters for removal.

```
######################################################################
# ### Firstly, reducing the matrix to "good cells" only
  foetal.all.matrix = foetal.all.matrix[,!foetal.cells2remove]

# Create a Seurat Object for integrationg and clustering of RBCs
  foetal.all.so = CreateSeuratObject(foetal.all.matrix)
```

```
# Assign meta data from our annotations
  foetal.all.so@meta.data = cbind(
    foetal.all.so@meta.data, annotations[
      rownames(foetal.all.so@meta.data),
    ][
      ,
    !colnames(annotations) %in% colnames(foetal.all.so@meta.data),drop=F
    ]
  )

# Split this large foetal object (of all experiments) by each sample
  foetal.all.list = SplitObject(foetal.all.so, split.by="sample")

# And for each experiment, we normalise the data and calculate variable features.
  for(i in 1:length(foetal.all.list)){
    foetal.all.list[[i]] = NormalizeData(foetal.all.list[[i]])
    foetal.all.list[[i]] = FindVariableFeatures(foetal.all.list[[i]])
  }
  gc()

# Integration anchors
  foetal.anchors <- FindIntegrationAnchors(object.list = foetal.all.list, dims = 1:30, anchor.features=2000)
  gc()

# Run integration
  foetal.integrated.RBCs <- IntegrateData(anchorset = foetal.anchors, dims = 1:30)
  gc()

# Add metadata
  foetal.integrated.RBCs@meta.data = cbind(
    foetal.integrated.RBCs@meta.data, annotations[
      rownames(foetal.integrated.RBCs@meta.data),
    ][
      ,
      !colnames(annotations) %in% colnames(foetal.integrated.RBCs@meta.data),drop=F
    ]
  )

# Deal with the metadata again... re-levelling factors from our RDS object
  foetal.integrated.RBCs@meta.data[,colnames(annotations)] = annotations[rownames(foetal.integrated.RBCs@meta.data),]

# ZERO-CENTRE
  foetal.integrated.RBCs <- ScaleData(foetal.integrated.RBCs, verbose = FALSE)
  gc()

# PCA
  foetal.integrated.RBCs <- RunPCA(foetal.integrated.RBCs, verbose = FALSE, npcs = 50)
  gc()

# UMAP
  foetal.integrated.RBCs <- RunUMAP(foetal.integrated.RBCs, dims = 1:30, umap.method = "uwot", n.components=2)
  gc()

# NEIGHBOURS
  foetal.integrated.RBCs = FindNeighbors(foetal.integrated.RBCs, k=30)
  gc()

# CLUSTERS
  foetal.integrated.RBCs = FindClusters(foetal.integrated.RBCs, res=0.1)
  foetal.integrated.RBCs = FindClusters(foetal.integrated.RBCs, res=0.3)
  foetal.integrated.RBCs = FindClusters(foetal.integrated.RBCs, res=1)
  foetal.integrated.RBCs = FindClusters(foetal.integrated.RBCs, res=3)
  foetal.integrated.RBCs = FindClusters(foetal.integrated.RBCs, res=10)
  gc()
#####################################################################
```

This is followed by the creation of an RBC score, calculated by the summed expression of a set of haemoglobulin genes. We then use a two compartment GMM in the hopes of classifying cells into high and low log-normal distributions of haemoglobulin expression. This is simple and effective in doing so. This score is added to the Seurat object metadata. This can then be visualised in our data (alongside a myriad of other variables) which suggests localisation to a particular group of cells across donors (Figure 2).

```
#####################################################################
# ### RBC removal
# Pull up RBC associated genes
  haem.genes = c("HBB", "HBG1", "HBG2", "HBM",  "HBA2", "HBA1", "HBQ1", "ALAS2")
  haem.genes = haem.genes[haem.genes %in% rownames(foetal.all.list[[1]])]
```

```
# Across whole dataset after integration and scaling
  rbc.score  = colSums(as.matrix(GetAssayData(foetal.integrated.RBCs, slot="scale.data")[haem.genes,]))
  rbc.mclust = Mclust(rbc.score, G=2)$classification

# Add vectors to metadata of object
  foetal.integrated.RBCs$rbc.mclust = rbc.mclust[colnames(foetal.integrated.RBCs)]
  foetal.integrated.RBCs$rbc.score = rbc.score[colnames(foetal.integrated.RBCs)]
  foetal.integrated.RBCs$rbc.mclust[foetal.integrated.RBCs$rbc.mclust == 2] = "RBC"
  foetal.integrated.RBCs$rbc.mclust[foetal.integrated.RBCs$rbc.mclust != "RBC"] = ""
  foetal.integrated.RBCs$rbc.mclust = as.factor(foetal.integrated.RBCs$rbc.mclust)
##########################################################
# ### Figure of the UMAP dims 1 and 2 with a few variables (including RBC calls from the GMM)
  par(mfrow=c(2,3), oma=c(0, 0, 0, 5), mar=c(4,4,2,10), cex=1)

  plot.so(foetal.integrated.RBCs, "lib.size", col=alt.cols, leg=T, xlab="UMAP 1", ylab="UMAP 2", pch=".")

  rbc.collected = collect.points(
    Embeddings(foetal.integrated.RBCs, reduction="umap")[,1],
    Embeddings(foetal.integrated.RBCs, reduction="umap")[,2],
    weight=rbc.score,
    colours=alt.cols,
    bin.dist=c(1,1),
    method="linear"
    )

  plot(rbc.collected[,1:2], col=rbc.collected[,4], pch=15, cex=1.4, xlab="UMAP 1", ylab="UMAP 2")
    gradient.legend(rbc.collected[,3], colours=alt.cols, scale=0.8, label="mean RBC score")

  plot.so(foetal.integrated.RBCs, "HCC", col=rep(Discrete12), leg=T, xlab="UMAP 1", ylab="UMAP 2", pch=".")

  plot.so(foetal.integrated.RBCs, "integrated_snn_res.0.3", col=rep(Discrete.U,2), label=T, xlab="UMAP 1", ylab="UMAP 2", pch=".", leg=T)

  plot.so(foetal.integrated.RBCs, "rbc.mclust", col=rep(Discrete), leg=T, xlab="UMAP 1", ylab="UMAP 2", pch=".")

  plot.so(foetal.integrated.RBCs, "scrublet", col=rep(Discrete), leg=T, xlab="UMAP 1", ylab="UMAP 2", pch=".")
#####################################################################
```

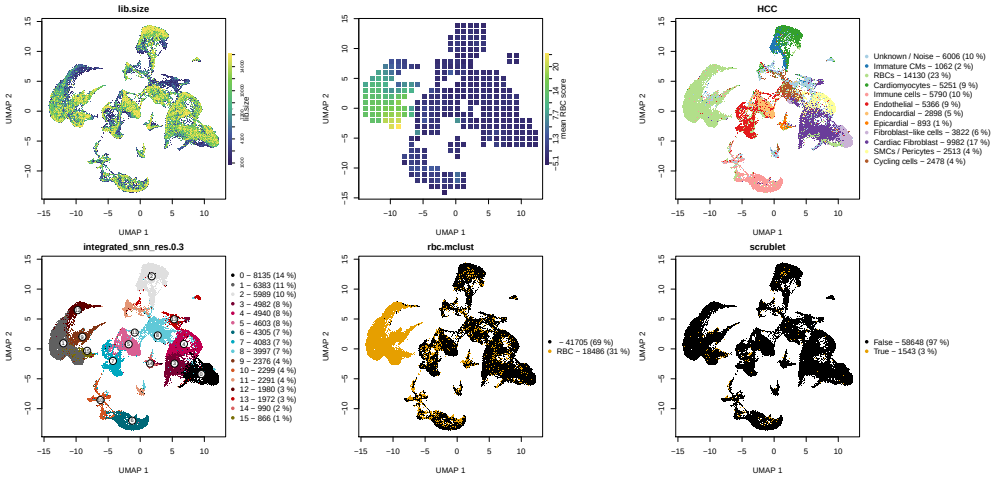

Figure 2: UMAPs of the integrated foetal datasets overlayed with depth, RBC score density, heart cell classification, clustering, GMM results, and doublet calls.

Finally, we use the binary classification of erythrocytes as defined above and combine this with clustering across multiple dimensions to define our red blood cell clusters for removal. In this heuristical approach we call all clusters with over 50 % cells classified as RBCs as a RBC cluster.

```
#####################################################################
# ### Clustering and RBC cluster calculation using RBC fraction
# Bring up clusters to annotate as RBC clusters - here we use res 0.3
  clusters = foetal.integrated.RBCs$integrated_snn_res.0.3

# For each cluster, generate aggregate RBC scores
  clusters.RBC = sapply(levels(clusters) , function(i) {table(foetal.integrated.RBCs$rbc.mclust[clusters==i])})
  RBC.HCC.score = round(100*clusters.RBC["RBC",] / colSums(clusters.RBC), 2)
  RBC.clusters = names(RBC.HCC.score)[which(RBC.HCC.score > 50)]
```

```
  foetal.integrated.RBCs$RBC.clusters = "non-RBC"
  foetal.integrated.RBCs$RBC.clusters[foetal.integrated.RBCs$integrated_snn_res.0.3 %in% RBC.clusters] = "RBC"

# ### Make figure using our plot.so function...
    par(oma=c(0,0,0,35))
    plot.so(foetal.integrated.RBCs, "RBC.clusters", col=Discrete.U2, leg=T)
########################################################################
```

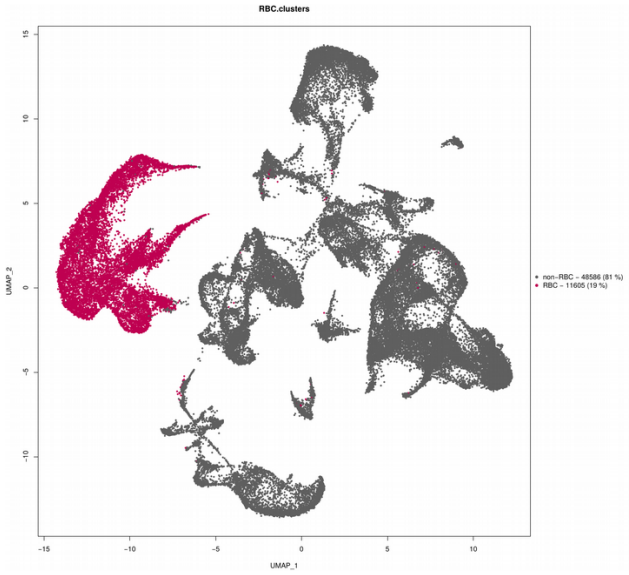

Figure 3: A final UMAP showing the integrated foetal datasets before RBC removal highlighted by the cluster-defined erythrocytes.

Finally, we will write our data objects for collection later on in the analysis.

```
########################################################################
# ### Write our datasets and key objects

# Write a new good cells object, additionally removing both doublets and RBCs from the mixture
  good.cells = names(foetal.integrated.RBCs$RBC.clusters)[
    foetal.integrated.RBCs$RBC.clusters == "non-RBC" &
    foetal.integrated.RBCs$scrublet == "False"
  ]
  write.table(good.cells, file=file.path("output/v_3/foetal_RBC_removal", "foetal.good.cells.csv"),
    sep=",", col.names=F, quote=F, row.names=F
  )
#
  saveRDS(foetal.integrated.RBCs, file.path("output/v_3/foetal_RBC_removal", "foetal.integrated.RBCs.rds"))
########################################################################
```

Removing all bad cells from foetal datasets

We will subset the data now, just taking the new sets of good cells and move forward into integrating foetal and adult datasets. The raw scripts during analysis are slightly different here, I ended up creating a pre-integrated object after removing the red blood cells. We will call this "foetal.integrated" here. However, the object does not need to be integrated and I only use this object going forward to derive the initial annotations for our next object. To advance from here we will simply subset our object.

```
########################################################################
# Annotations for this project
  annotations = readRDS("../G_Foetal_Epicardium/output/v_2/annotations/foetal_all_annotations_version4.rds")

# We have previously compiled the foetal dataset into a sparse matrix (QC cut quite loose)
  foetal.all.matrix = Read10X(data.dir = "../G_Foetal_Epicardium/data/foetal_all")

# RBC removed and doublet cells from this script
```

```
  good.cells = read.csv(file.path("output/v_3/foetal_RBC_removal", "foetal.good.cells.csv"), head=F)[,1]

# sample matrix
  foetal.all.matrix = foetal.all.matrix[,good.cells]
#####################################################################
```

The next thing we'll do is to run Seurat's integration pipeline on the foetal data.

```
#####################################################################
#  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #
# ### Firstly, create a Seurat Object
  foetal.all.so = CreateSeuratObject(foetal.all.matrix)

# Assign meta data from our annotations
  foetal.all.so@meta.data = cbind(
    foetal.all.so@meta.data, annotations[
      rownames(foetal.all.so@meta.data),
    ][
      ,
      !colnames(annotations) %in% colnames(foetal.all.so@meta.data),drop=F
    ]
  )

# Split this large foetal object (of all experiments) by each sample
  foetal.all.list = SplitObject(foetal.all.so, split.by="sample")

# And for each experiment, we normalise the data and calculate variable features.
  for(i in 1:length(foetal.all.list)){
    foetal.all.list[[i]] = NormalizeData(foetal.all.list[[i]])
    foetal.all.list[[i]] = FindVariableFeatures(foetal.all.list[[i]])
  }
  gc()

# Integration anchors
  foetal.anchors <- FindIntegrationAnchors(object.list = foetal.all.list, dims = 1:30, anchor.features=4000)
  gc()

# Run integration
  foetal.integrated <- IntegrateData(anchorset = foetal.anchors, dims = 1:30)
  gc()

# Add metadata
  foetal.integrated@meta.data = cbind(
    foetal.integrated@meta.data, annotations[
      rownames(foetal.integrated@meta.data),
    ][
      ,
      !colnames(annotations) %in% colnames(foetal.integrated@meta.data),drop=F
    ]
  )

# Deal with the metadata again... re-levelling factors from our RDS object
  foetal.integrated@meta.data[,colnames(annotations)] = annotations[rownames(foetal.integrated@meta.data),]

# This are the minimum steps for the object we need for the next stage of the analysis
  saveRDS(foetal.integrated, file.path("output/v_3/foetal_and_results", "foetal.integrated.rds"))

#####################################################################
```

Summary

In this second report, we have processed our foetal data, bringing the depth in line with the adult data. We have also removed the red blood cells and droplets, giving us a vector of good cells for use in stratified sampling of the foetal dataset in the next report, and for integration with the adult data. We can now combine the adult and foetal data and begin the integration process.