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

Vincent Knight-Schrijver

25 January 2023

Introduction

This report file goes through the process of reading the data ready for further analysis. Our data is sourced from out in-house experiments here at the University of Cambridge and can be accessed from the from GEO (Foetal heart dataset: GSE216019, hESC-Epicardium differentiation dataset: GSE216177). The adult hearts dataset was acquired from the Heart Cell Atlas (HCA) https://www.heartcellatlas.org/#DataSources.

Foetal data

Our datasets were stored locally and were processed prior to the initiation of this project. These datasets were first aligned in CellRanger version 6.1 and the filtered output was placed into separate directories. A preliminary analysis was carried our before this project, which was used to construct an annotations dataframe. We will use this for initial direction in the analysis.

Read data

For reading the foetal data we will read in the 12 later samples, and the 1 pilot sample. We will then process the data to standardise the features and cell names.

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

```r
######################################################################
# Foetal data - Apex samples

# APEX
 # BRC2251 = G4
 # BRC2252 = D8
 # BRC2256 = F8
 # BRC2260 = F2
 # BRC2262 = B1
 # BRC2263 = G7

  G4.apex.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/G4/filtered_feature_bc_matrix")
  D8.apex.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/D8/filtered_feature_bc_matrix")
  F8.apex.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/F8/filtered_feature_bc_matrix")
  F2.apex.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/F2/filtered_feature_bc_matrix")
  B1.apex.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/B1/filtered_feature_bc_matrix")
  G7.apex.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/G7/filtered_feature_bc_matrix")
  alexsc.apex.countsdata = read10X(path = "data/Alex10X")

# Foetal data - Base samples
# AORTA
 # BRC2251 = F4
 # BRC2252 = C8
 # BRC2256 = E8
 # BRC2260 = E2
 # BRC2262 = A1
 # BRC2263 = F7

  F4.aorta.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/F4/filtered_feature_bc_matrix")
  C8.aorta.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/C8/filtered_feature_bc_matrix")
  E8.aorta.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/E8/filtered_feature_bc_matrix")
  E2.aorta.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/E2/filtered_feature_bc_matrix")
  A1.aorta.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/A1/filtered_feature_bc_matrix")
  F7.aorta.countsdata <- read10X(path = "data/Cellranger6.1_filtered_barcodes_Semih/F7/filtered_feature_bc_matrix")

  gc()
########################################################################
```

```
########################################################################
# ### Updating and merging rownames between cellranger versions
# We have two versions of the HGNC annotations as our pilot sample was sequenced and aligned many years ago.
# First I will collect both features dataframes
  genes1 = read.table("data/Alex10X/genes.tsv")
  genes2 = read.table("data/Cellranger6.1_filtered_barcodes_Semih/G4/filtered_feature_bc_matrix/features.tsv.gz")

# Then I will subset both genes dataframes by the intersection of EnsemblID
  genes3 = genes1[genes1[,1] %in% genes2[,1],]

# Add new column for genes 2 genes that match genes3[,1] genes
  genes3[,c("G3.ENS", "G3.SYMBOL")] = genes2[match(genes3[,1], genes2[,1]),1:2]

# Prepare matrices rows - we also deal with duplicates here
# locate our objects for processing
  objs = ls(pattern="\\.countsdata$")
    # Quickly rearranging the obj order manually
  obj.order = c(12, 9, 5, 4, 11, 7, 8, 6, 3, 1, 13, 10, 2) # 2 at the end for alexsc dataset
  objs = objs[obj.order]

# Map for sample code and donor name across dissected region
  sample.codes = data.frame(
    "BRC" = c(rep(c("BRC2251", "BRC2252", "BRC2256", "BRC2260", "BRC2262", "BRC2263"),each=2),"alexsc"),
    "lane" = c("G4", "F4", "D8", "C8", "F8", "E8", "F2", "E2", "B1", "A1", "G7", "F7", "alexsc"),
    "position" = 1:13,
    "location" = c(rep(c("apex", "base"),6),"apex")
  )

# Apply new matched rownames across all objects
  for(obj in objs){
      x = get(obj)
    # subset genes
      x = x[genes3[,1],]

    # Deal with those aggregates
    # We pass these genes through to be aggregated as they are duplicates
      genes4 = genes3[,"G3.SYMBOL"]
      dup.genes = genes4[duplicated(genes4)]
      duplicates2remove = genes4 %in% dup.genes & (1:length(genes4) %in% which(duplicated(genes4)))

    # aggregate those duplicates:
      for(i in 1:length(dup.genes)){
        # Sum together genes of same name into one row to keep
          x$counts[which(genes4 == dup.genes[i])[1],] = colSums(x$counts[genes4 == dup.genes[i],])
      }
    # Then we subset the matrix removing the redundant duplicates
       x$counts = x$counts[!duplicates2remove,]

    # remove genes from genes list:
       genes4 = genes4[!duplicates2remove]

    # Edit rownames from ENS to HGNC
      rownames(x$counts) = genes4

    # assign object and return
      assign(obj, x)

      rm(x)
      gc()
  }

# Now re-label the barcodes / cell names to be unique between experiments
```

```
  for(obj in objs){
    x = get(obj)
  # rename columns
    colnames(x$counts) = gsub("-1$", paste("-", gsub("\\..*", "", obj), sep=""), x$samples[,"Barcode"]) # rename columns while we're at it...
  # assign object and return
    assign(obj, x)

    rm(x)
    gc()
  }

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

Lastly, we will take some important variables for each cell, namely those which will be altered after the processing step. We will take library size and number of genes.

```
#####################################################################
# ### Annotations of foetal data
# Preliminary annotations
  cell.barcodes = as.character(unlist(sapply(objs, function(i){colnames(get(i)$counts)})))
  genes = rownames(get(objs[1])) # any matrix will work, the genes have been subset and aggregated
  nCount_RNA = unlist(sapply(objs, function(i){get(i)$samples$lib.size}))


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

Pre-processing and QC of foetal data

After we have standardised the formatting of our foetal data we will quickly process the matrices to remove any extreme outliers. Downstream of this we will further subset our data to lie within the boundaries of the adult dataset. The criteria at this stage to keep a cell is based on the following:

1. Depth of above 500, but below 70000 UMIs
2. Complexity of above 400 genes expressed
3. Fraction of mitochondrial RNA must be less that 15 %

The following loop applies this to all of our formatted objects.

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

  # Plots
    pdf(file.path(script.dir, "figures", paste(k,"_QC_initial.pdf",sep="")), height=10, width=9)
    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(400))

    # High mitochondrial mRNA - remove above 10%
```

```
      plot(nMito); abline(h=15)

      cMito = nMito > 15
      cGenes = nGenes < 400
      cReads = nReads < 500 | nReads > 70000 # arbitrary threshold
      assign(paste(obj, ".cells2remove",sep=""), (cMito | cGenes | cReads))
      x2remove = get(paste(obj, ".cells2remove",sep=""))

    # 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(x2remove), main="cells removed (red)", pch=16)
    dev.off()

  # Matrix cut
    assign(paste(obj, ".cut.colnames", sep=""), colnames(x)[!x2remove])
  }

# ### Create new objects of cut matrices by defining cells for removing
# These are the vectors of cell names that pass these basic QC thresholds
  names.objs = ls()[grep(".cut.colnames", ls())][obj.order]

# These are stuck together for easily cutting our datasets
  cut.vector = as.character(unlist(sapply(names.objs, function(i){get(i)})))

# Cut these matrices
 for(obj in objs){
    x = get(obj)
    y = gsub("\\..*$", "", obj)
  # rename columns
    x = x$counts[,colnames(x$counts) %in% cut.vector]
  # assign object and return
    assign(paste(y,".cut",sep=""), x)

    rm(x)
    gc()
  }

#  leaving us now with a list of objects with the suffix ".cut".
######################################################################
```

Foetal initial output

From here we can free up our RAM, and combine the cut objects into a single matrix for exporting.

```
######################################################################
# ### Exporting a single object of 13 foetal hearts.
# We will now remove the initial objects and use these QC cut objects to save RAM before the next step(s)
  rm(G4.apex.countsdata)
  rm(F4.aorta.countsdata)
  rm(D8.apex.countsdata)
  rm(C8.aorta.countsdata)
  rm(F8.apex.countsdata)
  rm(E8.aorta.countsdata)
  rm(F2.apex.countsdata)
  rm(E2.aorta.countsdata)
  rm(B1.apex.countsdata)
  rm(A1.aorta.countsdata)
  rm(G7.apex.countsdata)
  rm(F7.aorta.countsdata)
  rm(alexsc.apex.countsdata)

  gc(full=T)

# Combine all ls() ".cut$" objects into an ordered vector and cbind to form a dataframe / matrix
  objs.cut = ls()[grep(".cut$", ls())][obj.order]
  foetal.all.matrix = cbind(
    get(objs.cut[1]),
    get(objs.cut[2]),
    get(objs.cut[3]),
    get(objs.cut[4]),
    get(objs.cut[5]),
    get(objs.cut[6]),
    get(objs.cut[7]),
    get(objs.cut[8]),
    get(objs.cut[9]),
    get(objs.cut[10]),
    get(objs.cut[11]),
    get(objs.cut[12]),
    get(objs.cut[13])
    )

# Write sparse matrix
  library(DropletUtils)
  library(Matrix)
# sparse Matrix
  foetal.all.matrix.sparse <- Matrix(as.matrix(foetal.all.matrix), sparse = T)

# Write output as a 10x sparse format - we're currently in "../G_Foetal_Epicardium/"
  write10xCounts(
    "data/foetal_all/",
    foetal.all.matrix.sparse,
    barcodes = colnames(foetal.all.matrix),
    gene.id = rownames(foetal.all.matrix),
    gene.symbol = rownames(foetal.all.matrix),
    gene.type = "Gene Expression",
    overwrite = FALSE,
    type = c("auto", "sparse", "HDF5"),
    genome = "unknown",
    version = c("3"),
    chemistry = "Single Cell 3' v3",
    original.gem.groups = 1L,
    library.ids = "custom"
  )
########################################################################
```

```
########################################################################
  annotations = readRDS("../G_Foetal_Epicardium/output/v_2/annotations/foetal_all_annotations_version4.rds")
  foetal.all.matrix = Read10X(data.dir = "../G_Foetal_Epicardium/data/foetal_all")
########################################################################
```

That is it for the initial reading in of foetal data. We will load this object in later on in the analysis. In the time being we will load in the adult data and start downsampling.

Adult dataset

We will first convert the downloaded adult dataset into an R compatible format using Seurat. The adult dataset was taken from the Heart Cell Atlas [2020] (https://cellgeni.cog.sanger.ac.uk/heartcellatlas/data/global_raw.h5ad). We will then examine the number of epicardial cells and see that only 6 donors have sufficient cells to include in the analysis, warranting downsampling of the adult dataset.

```
# ### convert the h5ad from the HCA using Seurat
  library(Seurat)
  library(SeuratDisk)

# First convert the raw dataset h5ad into Seurath5
  Convert("global_raw.h5ad", "global_raw.h5seurat")

# then we load in the h5ad Seurat object
  seuratObject <- LoadH5Seurat("HCA2020/global_raw.h5seurat")

# Taking a look at the object we can see the spread of Epicardial cells in the adult data
  epi.cells = seuratObject$cell_states == "Meso"
   bar.matrix(
    seuratObject$cell_source[epi.cells],
    seuratObject$donor[epi.cells],
    plot=T, scale=F, label="# cells"
   )
```

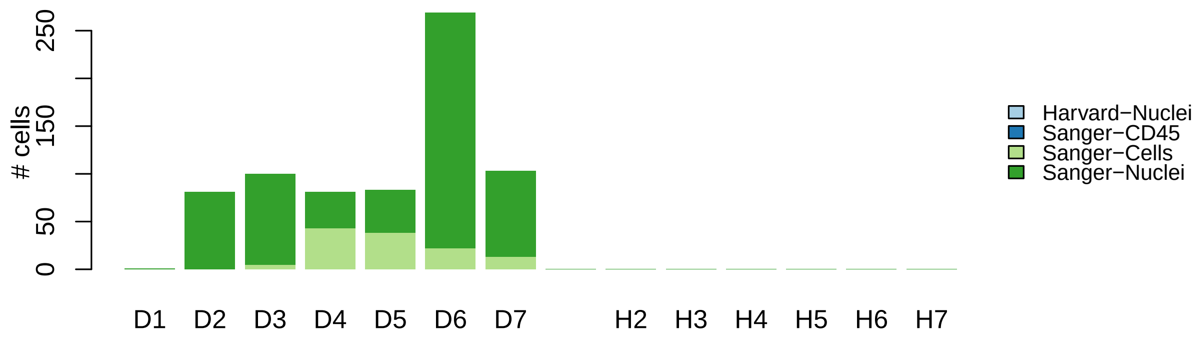

Figure 1: Only 6 donors in the adult dataset have over 3 epicardial cells.

Subsampling of datasets

We will use a sub-sampling strategy to reduce the dataset size and to control for variance between experimental conditions. For the adult data we can remove all donors that had fewer than 3 epicardial cells, since these are not informative for our analysis into our cell population of interest (the epicardium).

We can view the dataset distribution of cell types etc across various factors and are interested in sampling evenly across both donor and cell type. We use the previous annotations ("cell_states") to determine the categories and my approach is to sample half from cells and half from Nuclei if possible, for each cluster. The remaining barcodes will be made up from either source until the size of cell type meets, or exceeds N.

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

# Firstly, we will subset this dataset by donor, allowing us to form sampling stratifications
# We can subset the dataset by donors - those which have EPI labelled cells
  epi.donors = table(seuratObject$donor[seuratObject$cell_states == "Meso"]) > 1
  epi.donors = seuratObject$donor %in% names(epi.donors)[epi.donors]

# we are only interested in nuclei OR single cells from these donors...
  epi.cell.types = seuratObject$cell_source %in% c("Sanger-Cells", "Sanger-Nuclei")

# Put this together for a sample selection across donors:
  cells2sample = epi.cell.types & epi.donors

# Plotting figure
      par(oma=c(2,0,0,6))
      pre.ss.donor = bar.matrix(
        seuratObject$donor[cells2sample],
        seuratObject$cell_states[cells2sample],
        plot=T,
        scale=F,
        label="# cells",
        colours=Discrete.U2,
        las=3
      )

      text(
        pre.ss.donor$barx,
        pre.ss.donor$bary+max(pre.ss.donor$bary)*0.01,
        pre.ss.donor$bary,
        xpd=NA,
        cex=0.8,
        adj=0,
        srt=90
      )

    # pre_sampling all
      par(oma=c(2,0,0,6))
      pre.ss.donor = bar.matrix(
        seuratObject$cell_source[cells2sample],
        seuratObject$cell_states[cells2sample],
        plot=T,
        scale=F,
        label="# cells",
        colours=Discrete.U2,
        las=3
      )

      text(
        pre.ss.donor$barx,
        pre.ss.donor$bary+max(pre.ss.donor$bary)*0.01,
        pre.ss.donor$bary,
        xpd=NA,
        cex=0.8,
        adj=0,
        srt=90
      )
```

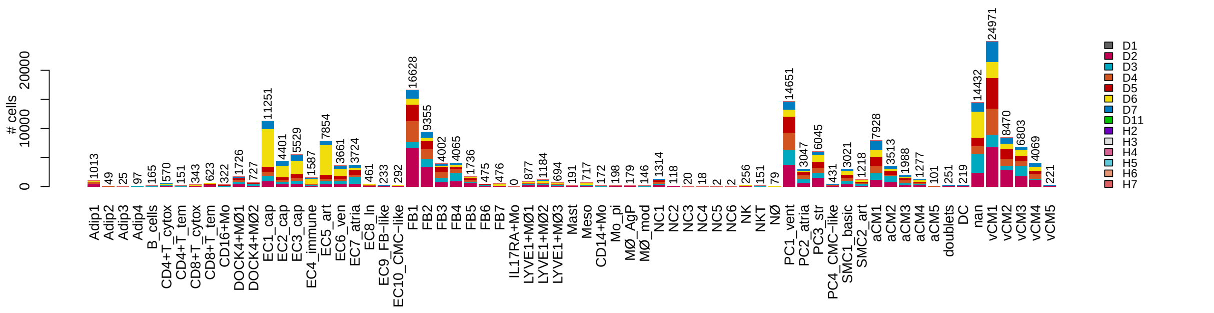

Figure 2: Cell donors across cell states show that each cell state is comprised of a different composition of donors (note that we only have donors d2-d7 here). Subsampling can balance this.

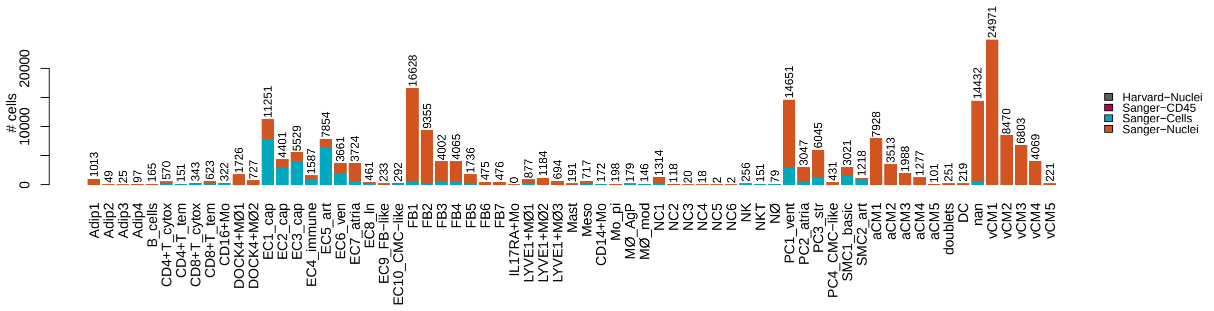

Figure 3: Cell sources across cell states show that each cell state is comprised of a different composition of nuclei or cells (note that we only have donors d2-d7 here). Subsampling can balance this.

We can see that there are 718 "Meso" (Epicardial) cells (I had an additional one from the donor D1, this value remained from a previous run of this analysis). We'll begin by taking all cells and then subsampling to the number of epicardial cells. I choose 597 based on the number of nuclei in the dataset, as we will end up with a 50 / 50 split between cells and nuclei before downsampling to the size of the epicardial cell cluster.

```
##########################################################################
  epi.cell.cells = seuratObject$cell_source %in% c("Sanger-Cells")
```

```
# Put this together for a sample selection:
  cells2sample = epi.cell.cells & epi.donors

# subsampling cells from these donors:
# We take the maximum number of epicardial nuclei: 597 (718 epi cells in total)
  N = 597
  cells.ss = stratified_subsample(
    f1 = seuratObject$cell_states[cells2sample],
    f2 = seuratObject$donor[cells2sample],
    cell_IDs = colnames(seuratObject)[cells2sample],
    N = N,
    seed = 1
  )


    par(oma=c(3,0,0,5), mfrow=c(2,1), mar=c(1,4,2,0))
    pre.ss.cells = bar.matrix(
      seuratObject$donor[cells2sample],
      seuratObject$cell_states[cells2sample],
      plot=T,
      scale=F,
      label="# cells",
      colours=Discrete.U2,
      las=3,
      main="",
      xaxt="n"
    )

    text(
      pre.ss.cells$barx,
      pre.ss.cells$bary+max(pre.ss.cells$bary)*0.01,
      pre.ss.cells$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
    mtext("before subsampling",2, padj=-5)
    par(mar=c(4.3,4,1.6,0))
    post.ss.cells = bar.matrix(
      seuratObject$donor[cells.ss],
      seuratObject$cell_states[cells.ss],
      plot=T,
      scale=F,
      label="# cells",
      colours=Discrete.U2,
      las=3,
      main="",
      leg=F
    )

    text(
      post.ss.cells$barx,
      post.ss.cells$bary+max(post.ss.cells$bary)*0.01,
      post.ss.cells$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
  mtext("after subsampling",2, padj=-5)
######################################################################
```

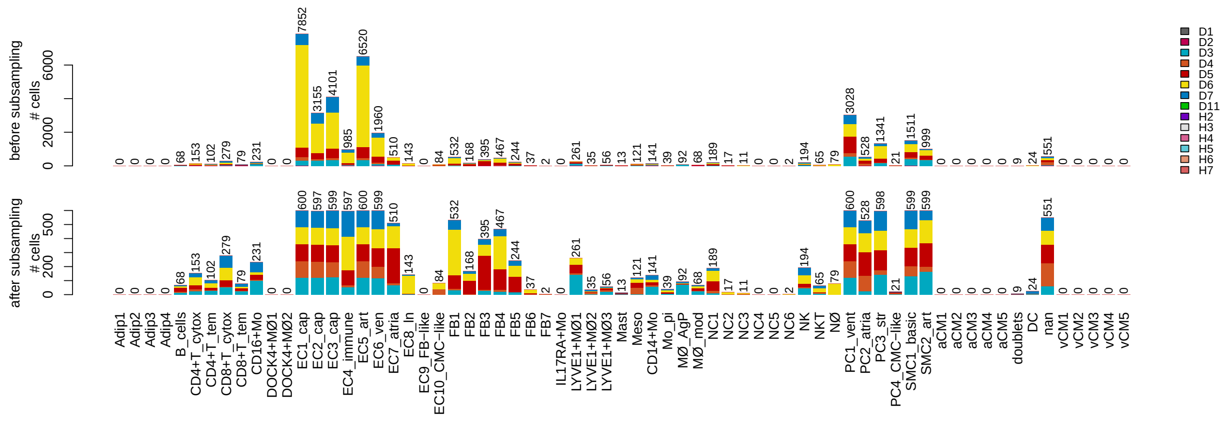

Figure 4: Subsampling of adult cell state annotations to make cell state clusters of a maximum of 597 cells each.

We will then repeat this step for the nuclei, again using a value of 597 for that 50 / 50 split between cells and nuclei before further downsampling downsampling to the size of the epicardial cell cluster.

```
##########################################################################
# We'll next take all nuclei and then subsampling to the number of epicardial nuclei.
  epi.cell.nuclei = seuratObject$cell_source %in% c("Sanger-Nuclei")

# Put this together for a sample selection:
  cells2sample = epi.cell.nuclei & epi.donors

# subsampling nuclei from these donors:
  N = 597
  nuclei.ss = stratified_subsample(
    f1 = seuratObject$cell_states[cells2sample],
    f2 = seuratObject$donor[cells2sample],
    cell_IDs = colnames(seuratObject)[cells2sample],
    N = N,
    seed = 1
  )

# pre_sampling_nuclei
    par(oma=c(3,0,0,5), mfrow=c(2,1), mar=c(1,4,2,0))
    pre.ss.nuclei = bar.matrix(
      seuratObject$donor[cells2sample],
      seuratObject$cell_states[cells2sample],
      plot=T,
      scale=F,
      label="# nuclei",
      colours=Discrete.U2,
      las=3,
      main="",
      xaxt="n"
    )

    text(
      pre.ss.nuclei$barx,
      pre.ss.nuclei$bary+max(pre.ss.nuclei$bary)*0.01,
      pre.ss.nuclei$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
    mtext("before subsampling",2, padj=-5)
    par(mar=c(4.3,4,1.6,0))
    post.ss.nuclei = bar.matrix(
      seuratObject$donor[nuclei.ss],
      seuratObject$cell_states[nuclei.ss],
      plot=T,
      scale=F,
      label="# nuclei",
      colours=Discrete.U2,
      las=3,
      main="",
      leg=F
    )
    text(
      post.ss.nuclei$barx,
      post.ss.nuclei$bary+max(post.ss.nuclei$bary)*0.01,
      post.ss.nuclei$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
    mtext("after subsampling",2, padj=-5)

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

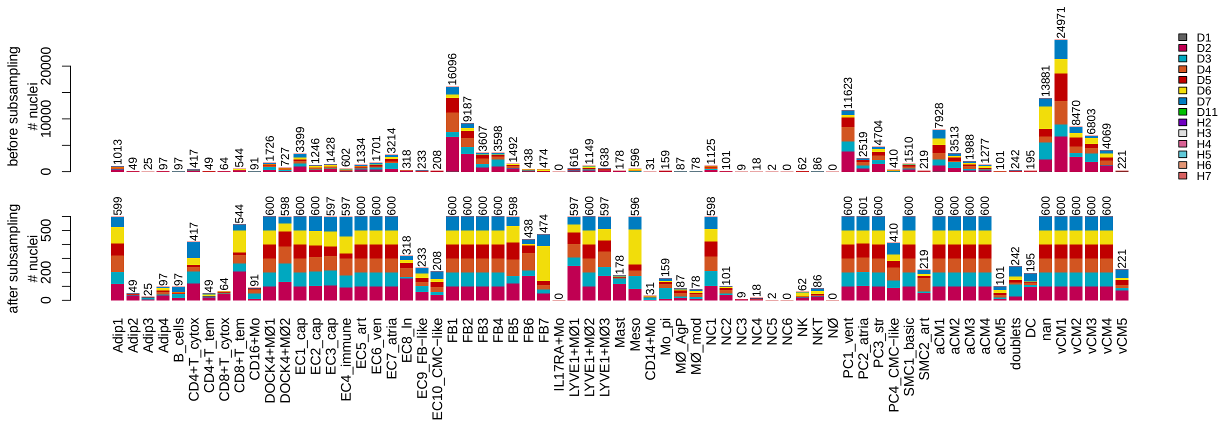

Figure 5: Subsampling of adult cell state annotations to make cell state clusters of a maximum of 597 cells each.

Then, we approach the last step of downsampling the adult data. This is where we group cells and nuclei together followed by a final sub-sampling of cell state clusters to the size of overall epicardial cell population 718.

```
######################################################################
# ### Sample donor across cell clusters after balancing source
Then we paste the cell sources together and re-sample to make 718 epi cells:
  cells2sample = c(cells.ss, nuclei.ss)
  N = 718
  all.ss = stratified_subsample(
    f1 = seuratObject$cell_states[cells2sample],
    f2 = seuratObject$donor[cells2sample],
    cell_IDs = cells2sample,
    N = N,
    seed = 1
  )

  par(oma=c(3,0,0,7), mfrow=c(2,1), mar=c(1,4,2,0))
     pre.ss.all = bar.matrix(
      seuratObject$cell_source[cells2sample],
      seuratObject$cell_states[cells2sample],
      plot=T,
      scale=F,
      label="# cells",
      colours=Discrete.U2,
      las=3,
      main="",
      xaxt="n"
    )

    text(
      pre.ss.all$barx,
      pre.ss.all$bary+max(pre.ss.all$bary)*0.01,
      pre.ss.all$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
   mtext("before subsampling",2, padj=-5)
   par(mar=c(4.3,4,1.6,0))
    post.ss.all = bar.matrix(
      seuratObject$cell_source[all.ss],
      seuratObject$cell_states[all.ss],
      plot=T,
      scale=F,
      label="# cells",
      colours=Discrete.U2,
      las=3,
      main="",
      leg=F
    )

    text(
      post.ss.all$barx,
      post.ss.all$bary+max(post.ss.all$bary)*0.01,
      post.ss.all$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
   mtext("after subsampling",2, padj=-5)

   par(oma=c(3,0,0,5), mfrow=c(2,1), mar=c(1,4,2,0))
     pre.ss.all = bar.matrix(
      seuratObject$donor[cells2sample],
      seuratObject$cell_states[cells2sample],
      plot=T,
      scale=F,
      label="# cells",
      colours=Discrete.U2,
      las=3,
      main="",
      xaxt="n"
    )

    text(
      pre.ss.all$barx,
      pre.ss.all$bary+max(pre.ss.all$bary)*0.01,
      pre.ss.all$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
   mtext("before subsampling",2, padj=-5)
   par(mar=c(4.3,4,1.6,0))
    post.ss.all = bar.matrix(
      seuratObject$donor[all.ss],
      seuratObject$cell_states[all.ss],
      plot=T,
      scale=F,
      label="# cells",
      colours=Discrete.U2,
      las=3,
      main="",
      leg=F
    )

    text(
      post.ss.all$barx,
      post.ss.all$bary+max(post.ss.all$bary)*0.01,
      post.ss.all$bary,
      xpd=NA,
      cex=0.8,
      adj=0,
      srt=90
    )
   mtext("after subsampling",2, padj=-5)

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

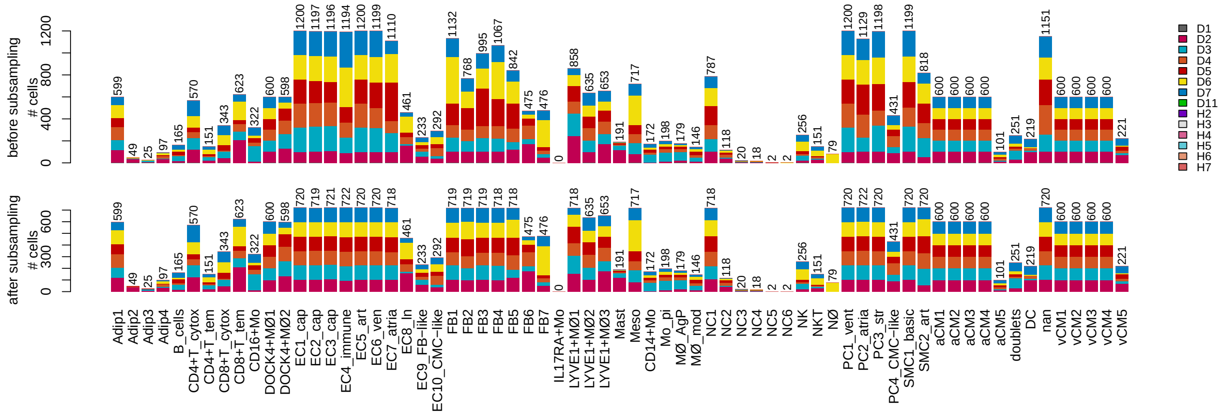

Figure 6: Subsampling of adult cell states by donor to make 718 cells in each cluster after balancing the nuclei and cell compositions results in even donor distribution within clusters.

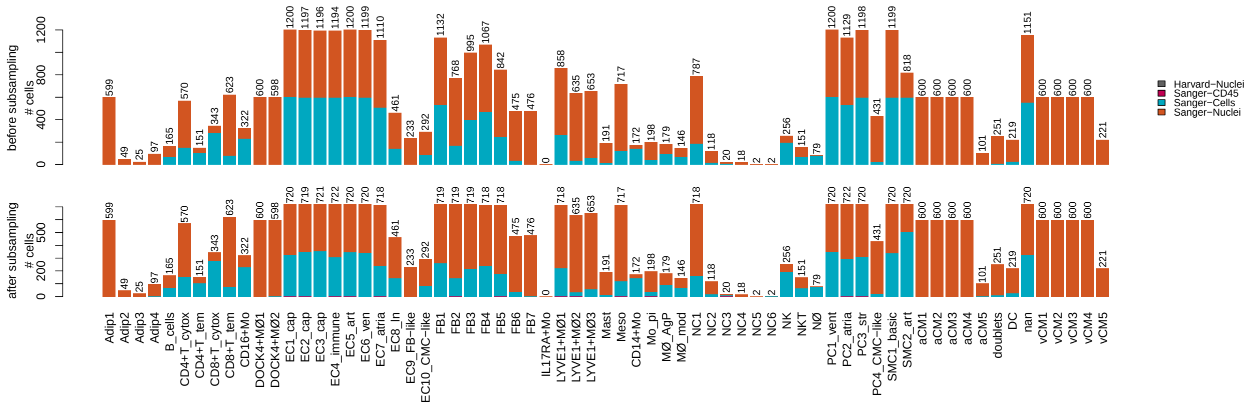

Figure 7: Subsampling of adult cell states by donor to make 718 cells in each cluster after balancing the nuclei and cell compositions results in even cell source distribution within clusters.

We will write this subsampled vector as a csv for calling later if necessary, and write the whole object as a Seurat object for the analysis.

```
########################################################################
# ### Saving our objects
# save our sub-sampled barcodes
  write.table(all.ss, file=file.path(script.dir,"all.ss.csv"), sep=",", col.names=F, row.names=F, quote=F)

# finally we will subset our SeuratObject with this sample
  adult.ss.so = subset(seuratObject, cells=all.ss)
  saveRDS(adult.ss.so, "RDS_files_V3/adult.ss.so.rds")
########################################################################
```

Summary: reading data and adult data subsampling

This finishes our adult dataset subsampling. We have combined approximately 600 cells with 600 nuclei from each cluster, and re-sampled down to the size of the epicardial cell population (717) within the entire dataset using donor as a stratification. This should regulate a bit of the donor to donor variation in downstream integration and analysis. The next code report, report B, documents the processing of foetal data, including removal of red blood cells. Save the following objects:

```
######################################################################
# ### objects here for further analysis
  annotations
  foetal.all.matrix
  adult.ss.so
  all.ss
######################################################################
```