Long Covid study scRNA-seq, 8 month time point, data preprocessing

Emily Kibbler

2025-08-10

This pipeline is for analysis of scRNA-seq data from a publicly available data set.

Citation:

Phetsouphanh C, [. . .] Matthews GV. Improvement of immune dysregulation in individuals with long COVID at 24-months following SARS-CoV-2 infection. Nat Commun. 2024 Apr 17;15(1):3315. doi: 10.1038/s41467-024-47720-8. PMID: 38632311; PMCID: PMC11024141.

[Click here to access the data from GEO](https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE262861

This specific script reads in the libraries from samples collected at 8 months post-infection (10 patients with long COVID; 10 matched control patients who had COVID infections at similar times but did not develop long COVID)

```
library(Seurat)
```

```
## Loading required package: SeuratObject
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'SeuratObject'
```

```
## The following objects are masked from 'package:base':
##
##     intersect, t
```

```
library(ggpubr)
```

```
## Loading required package: ggplot2
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.4     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# This function takes a data set (Seurat object) and a list of desired features and returns a ggarrange
# Features must be something that can be retrieved by Seurat's FetchData function
# Heavy lifting performed by Seurat's VlnPlot function; formatting and tweaking by E. Kibbler
```

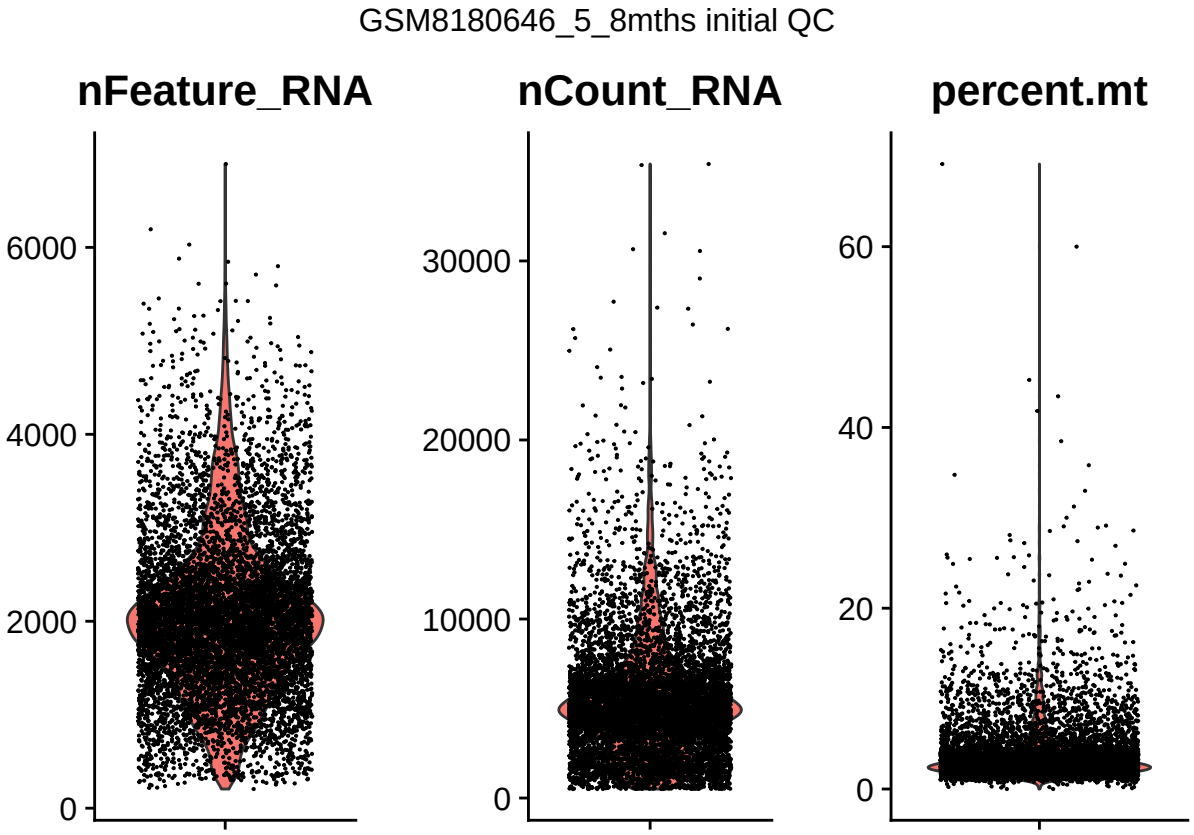
```r
myVlnPlot <- function(dat, feats) {
  # Structure as a list even if only one feature is given
  if (length(feats) == 1) {
    feats <- c(feats)
  }
  # Initiate empty list to put the plot(s) in
  plots <- c()
  # Plot each feature separately
  for (i in 1:length(feats)) {
    # Use suppressWarnings because the VlnPlot() has a default argument which uses a deprecated argumen
    temp <- suppressWarnings(VlnPlot(dat,
                    features = feats[i])) +
      theme(axis.text.x = element_blank(),
            axis.title.x = element_blank(),
            legend.position = "none")
    plots <- c(plots, list(temp))
  }
  # Generate and return a plot of panel(s)
  res <- ggarrange(plotlist = plots, nrow = 1)
  return(res)
}
```

```r
# Define libraries to read in
ids <- c("GSM8180646_5_8mths",
         "GSM8180647_6_8mths",
         "GSM8180648_7_8mths",
         "GSM8180649_8_8mths")
mywd <- "./data/covid/GSE262861_RAW/"
```

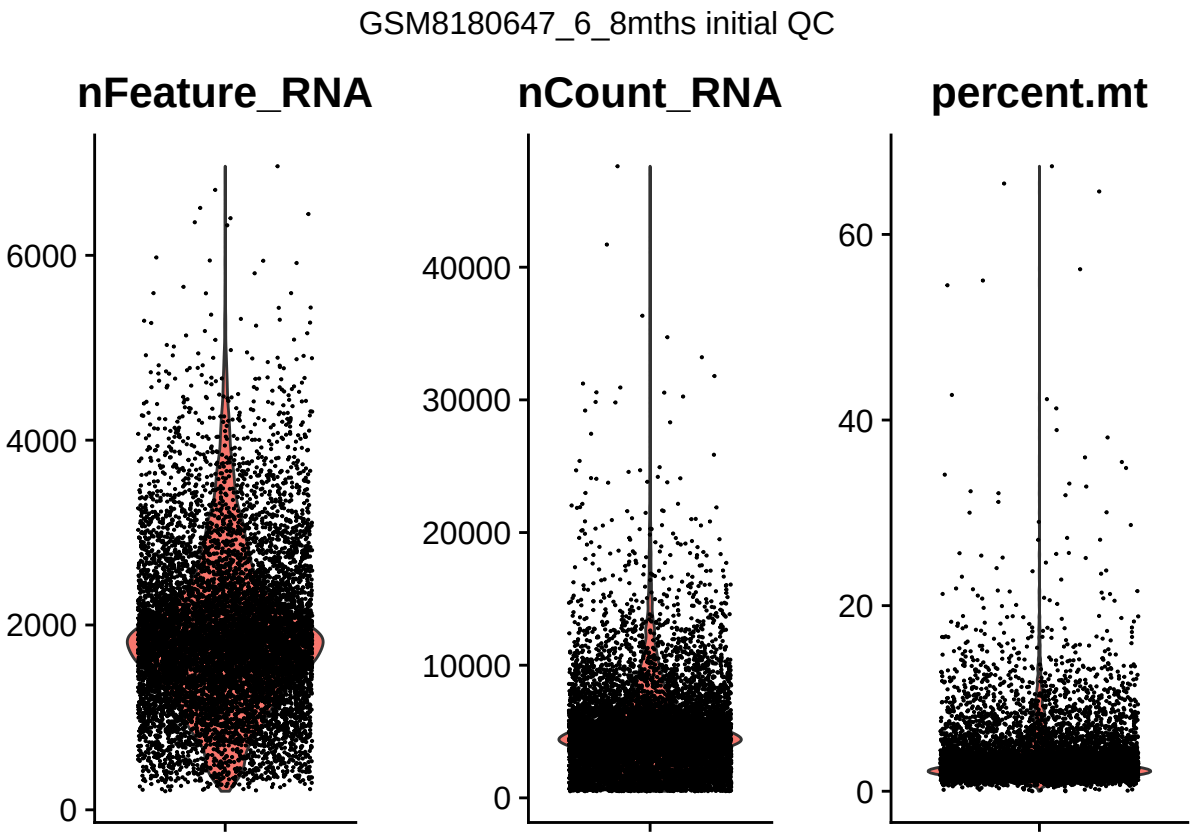
Read in data

```r
for (library_id in ids) {
    results <- read.table(paste0(mywd,
                                 library_id,
                                 "/",
                                 library_id,
                                 "_combined_results_w_combined_assignments.tsv.gz"),
                          header = T)
    pbmc.data <- Read10X(data.dir = paste0(mywd, library_id, "/10x_dat"))
    pbmc <- CreateSeuratObject(counts = pbmc.data,
                               project = "longcovid",
                               min.cells = 3,
                               min.features = 200)
    pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")
    pbmc[["UMI"]]  <- rownames(pbmc@meta.data)

    # Assign timepoints
    if (grepl("4mths", library_id)) {timepoint = "4months"}
    if (grepl("8mths", library_id)) {timepoint = "8months"}
    if (grepl("^ES", library_id)) {timepoint = "24months"}
    pbmc[["timepoint"]]  <- timepoint

    # Plot initial QC
    p <- myVlnPlot(pbmc, feats = c("nFeature_RNA",
```

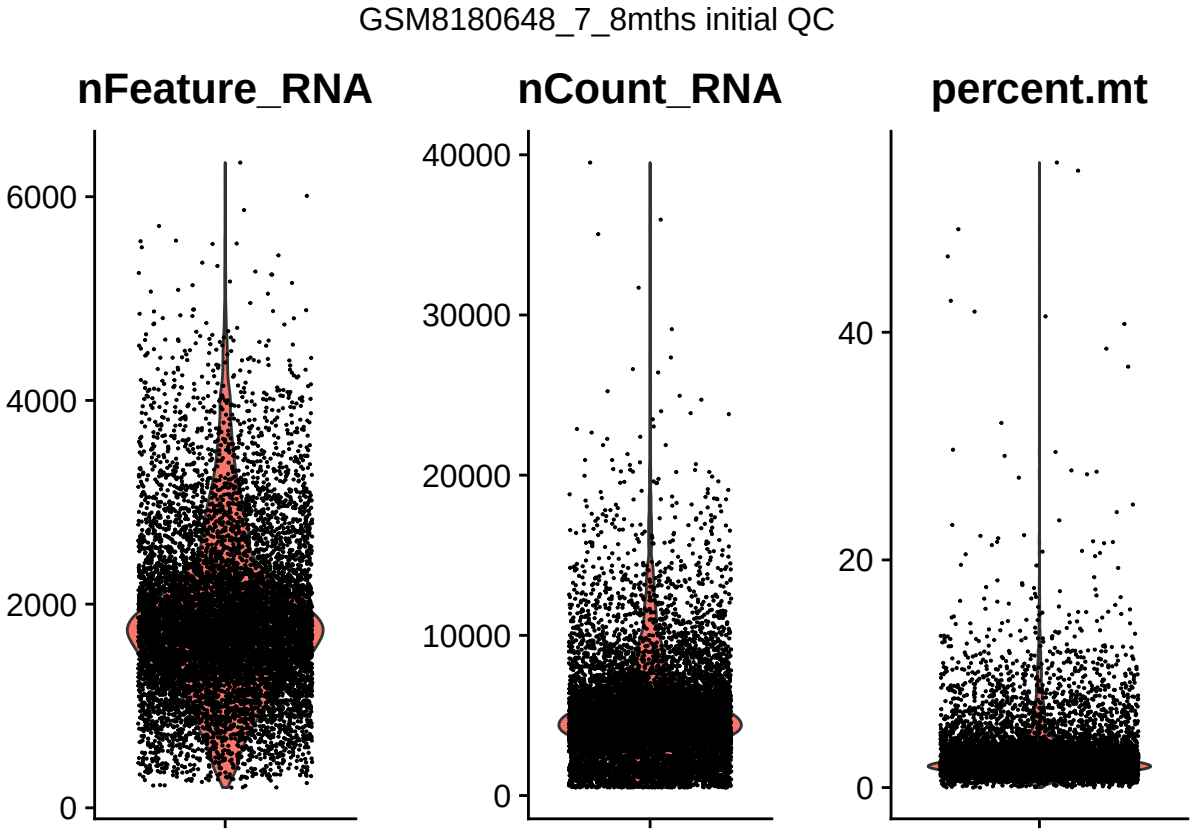
```
                                  "nCount_RNA",
                                  "percent.mt")) %>%
      annotate_figure(top = text_grob(paste(library_id, "initial QC")))
    print(p)
    ggsave(filename = paste0(mywd, library_id, "/qc.png"), p)

    # Assign cells
    pbmc@meta.data <- left_join(pbmc@meta.data,
                                results,
                                by = join_by(UMI == Barcode))
    rownames(pbmc@meta.data) <- pbmc@meta.data$UMI

    pool <- str_split_i(library_id, "_", 2)
    # Even-numbered pools are the second replicate
    if (pool %in% c(4, 6, 8)) {
      pbmc[["batch"]] <- "rep 2"
    } else {pbmc[["batch"]] <- "rep 1"}

    pbmc[["library"]] <- library_id

    write_rds(pbmc, file = paste0(mywd, library_id, "/", library_id, "_EK.rds"))
}
```



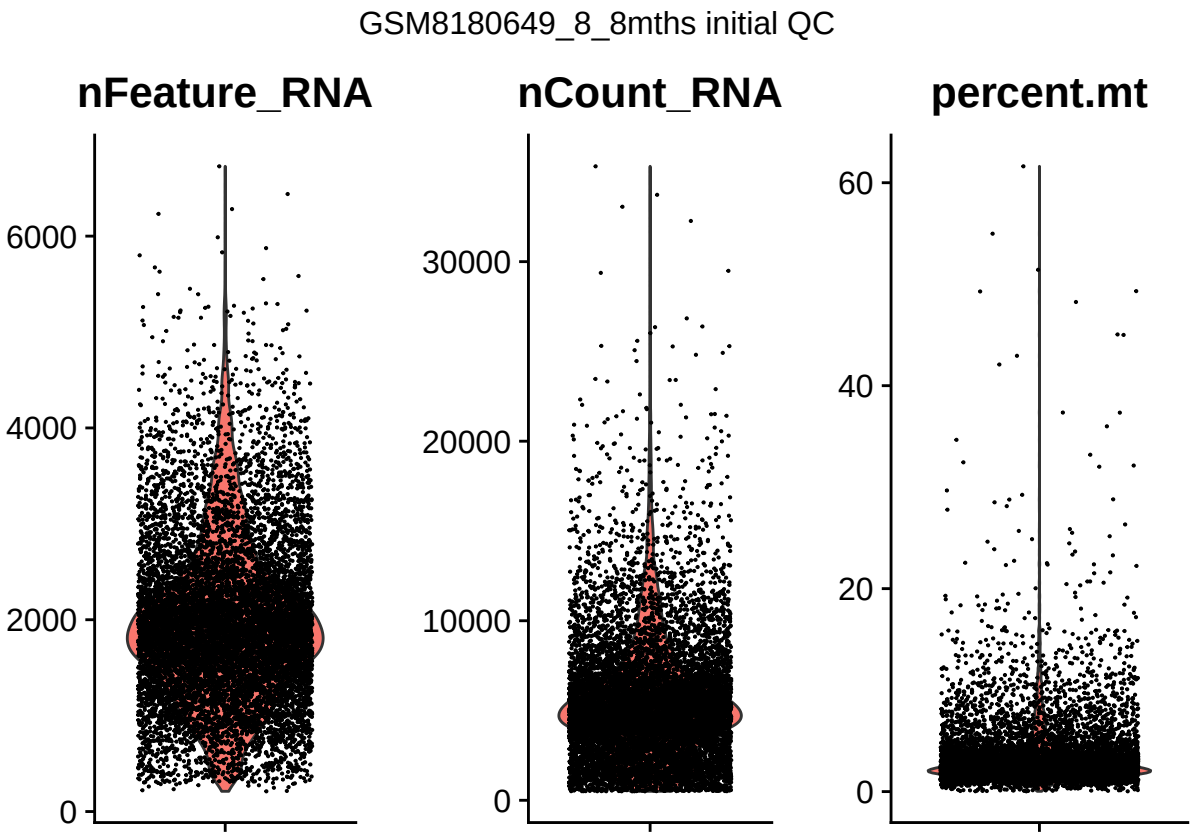

```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```



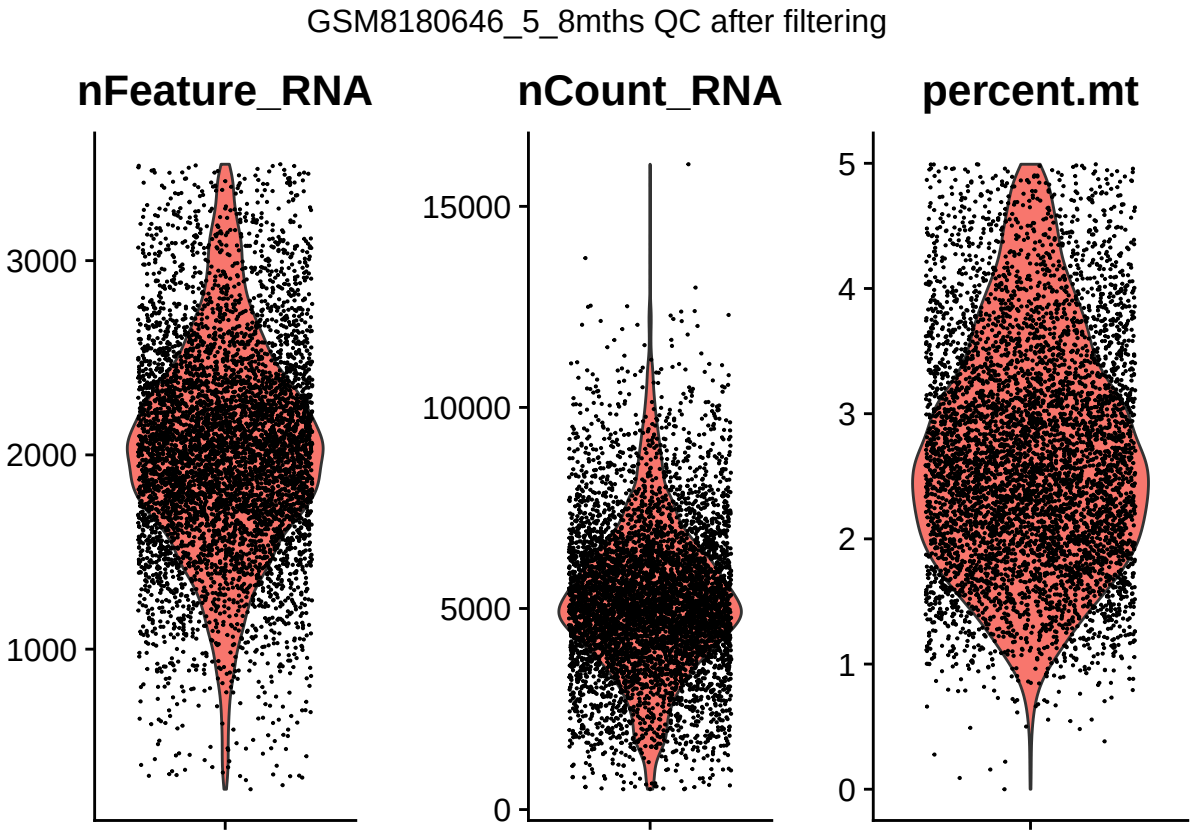
```
## Saving 6.5 x 4.5 in image
```



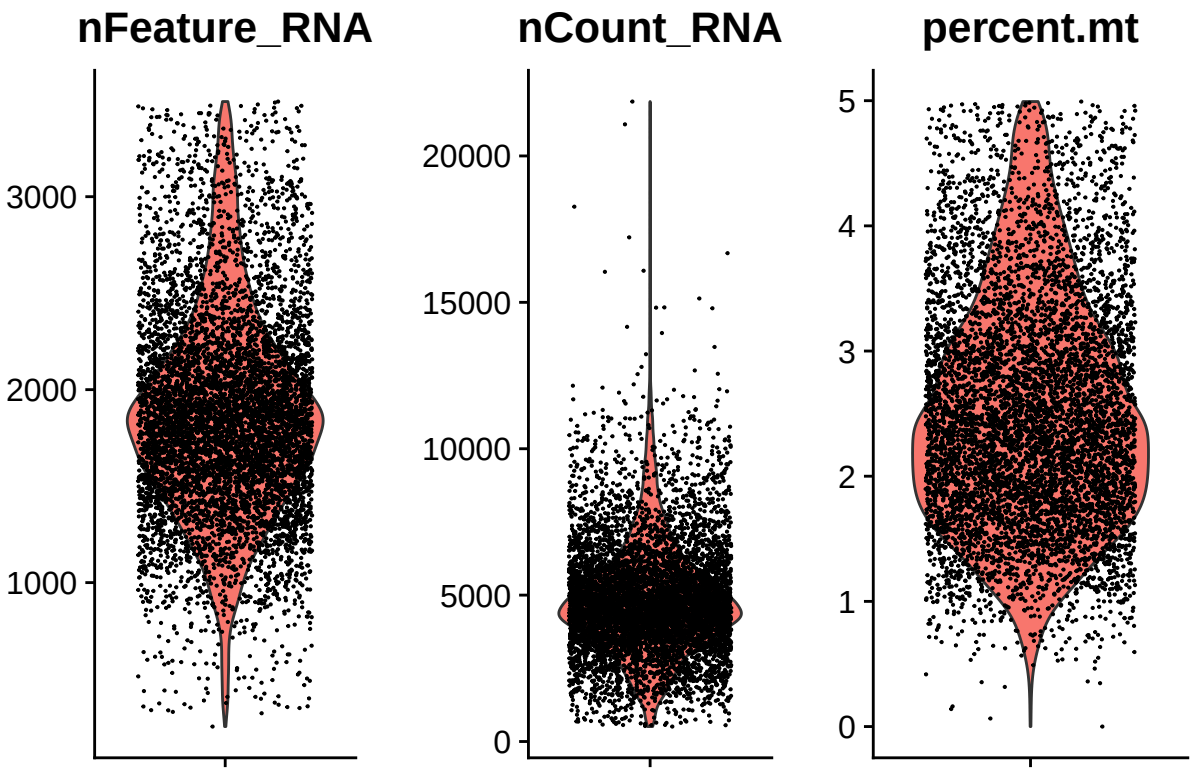

```
## Saving 6.5 x 4.5 in image
```

Filter and prepare individual data sets

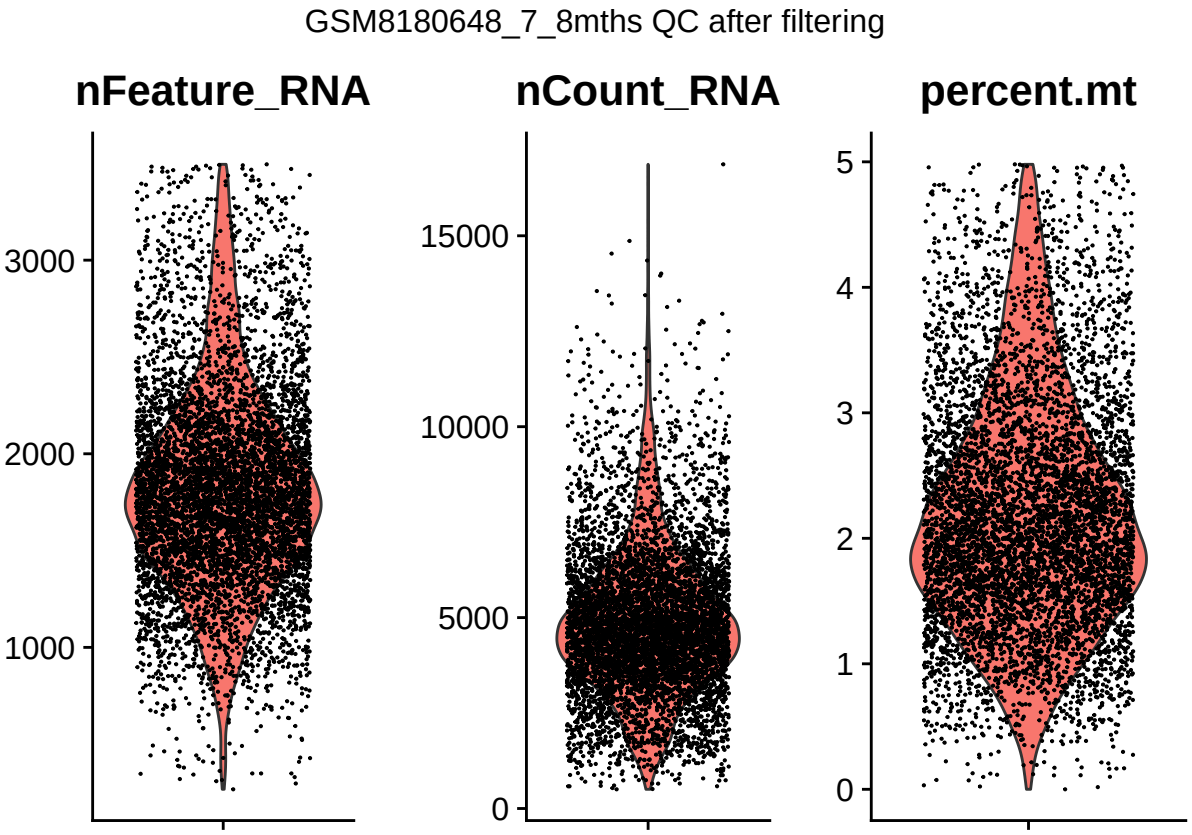
```
# Empty list, for each batch to be added to
pbmc.list <- list()
# Load and process all data
for (library_id in ids) {
  pbmc <- readRDS(file = paste0(mywd, library_id, "/", library_id, "_EK.rds"))
  pbmc <- subset(pbmc,
                 subset = nFeature_RNA > 200 &
                   nFeature_RNA < 3500 &
                   percent.mt < 5 &
                   Demuxlet_DropletType == "singlet")
  p <- myVlnPlot(pbmc, feats = c("nFeature_RNA",
                                 "nCount_RNA",
                                 "percent.mt")) %>%
    annotate_figure(top = text_grob(paste(library_id, "QC after filtering")))
  print(p)
  ggsave(filename = paste0(mywd, library_id, "/", library_id, "_after_filter_qc.png"), p)
  pbmc <- SCTransform(pbmc, verbose = FALSE)
  pbmc <- RunPCA(pbmc, verbose = FALSE)
  pbmc <- RunUMAP(pbmc, dims = 1:30, verbose = FALSE)
  pbmc <- FindNeighbors(pbmc, dims = 1:30, verbose = FALSE)
  pbmc <- FindClusters(pbmc, verbose = FALSE)
  pbmc.list[[library_id]] = pbmc
}
```



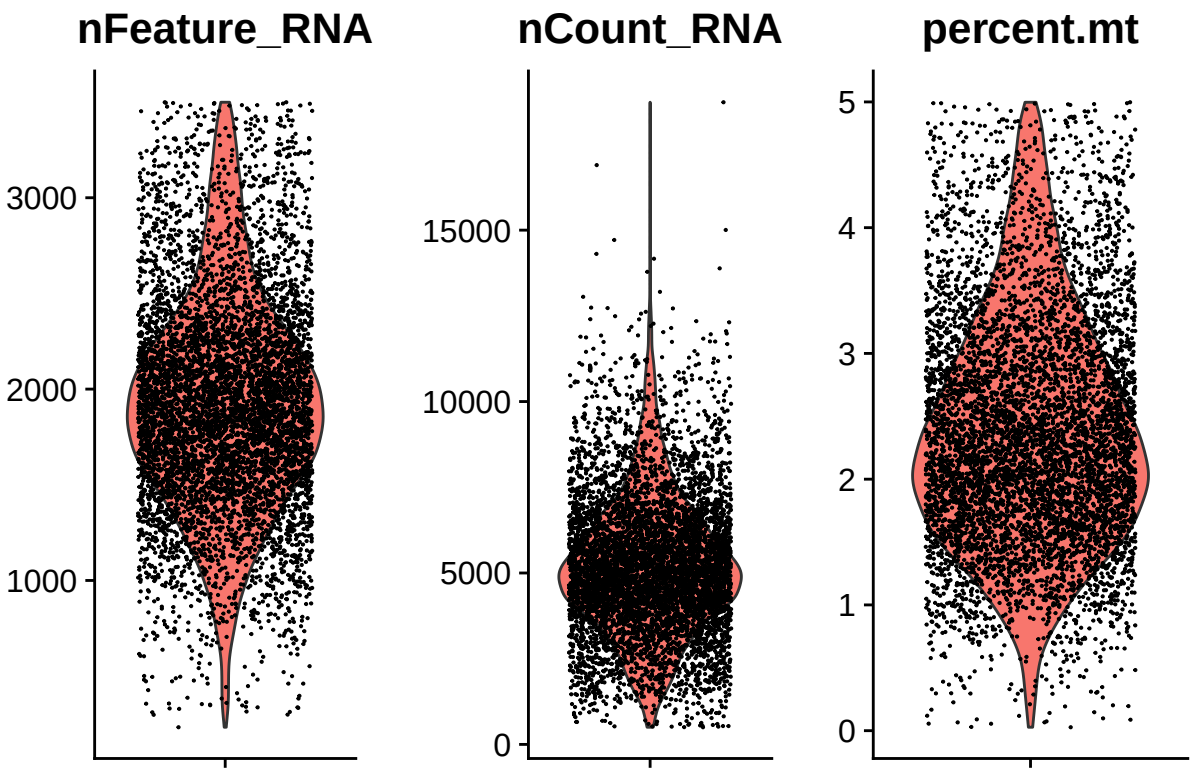

```
## Saving 6.5 x 4.5 in image
```



```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```

Integrate data

```
features <- SelectIntegrationFeatures(object.list = pbmc.list,
                                      nfeatures = 3000)
pbmc.list <- PrepSCTIntegration(object.list = pbmc.list,
                                anchor.features = features)
anchors <- FindIntegrationAnchors(object.list = pbmc.list,
                                  normalization.method = "SCT",
                                  anchor.features = features)
```

```
## Finding all pairwise anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 15107 anchors
## Filtering anchors
##  Retained 13077 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
```

```
## Finding anchors
##  Found 13747 anchors
## Filtering anchors
##  Retained 11595 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 14542 anchors
## Filtering anchors
##  Retained 12531 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 13893 anchors
## Filtering anchors
##  Retained 11605 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 14970 anchors
## Filtering anchors
##  Retained 12564 anchors
## Running CCA
## Merging objects
## Finding neighborhoods
## Finding anchors
##  Found 15586 anchors
## Filtering anchors
##  Retained 13400 anchors
```

```
combo <- IntegrateData(anchorset = anchors,
                                    normalization.method = "SCT")
```

```
## [1] 1
## [1] 2
```

```
## [1] 3
## [1] 4
## Merging dataset 1 into 2
## Extracting anchors for merged samples
## Finding integration vectors
## Finding integration vector weights
## Integrating data
## Merging dataset 3 into 4
## Extracting anchors for merged samples
## Finding integration vectors
## Finding integration vector weights
## Integrating data
## Merging dataset 2 1 into 4 3
## Extracting anchors for merged samples
## Finding integration vectors
## Finding integration vector weights
## Integrating data
```

```r
# Export, to be used by future scripts
saveRDS(combo, "./data/covid/initial_combo_8mths.rds")
```