scRNA-seq analysis for manuscript: Global Inhibition of Myeloperoxidase Improves Survival in a Novel Mouse Model of Coronary Atherosclerosis, Spontaneous Myocardial Infarction, and Stroke

Anita Salamon

2022-15-09

step: QC Filter Cells

Intruction

This notebook provides an overview of the scRNA-seq analysis workflow for the manuscript in preparation for submission at *Nature* in 2023

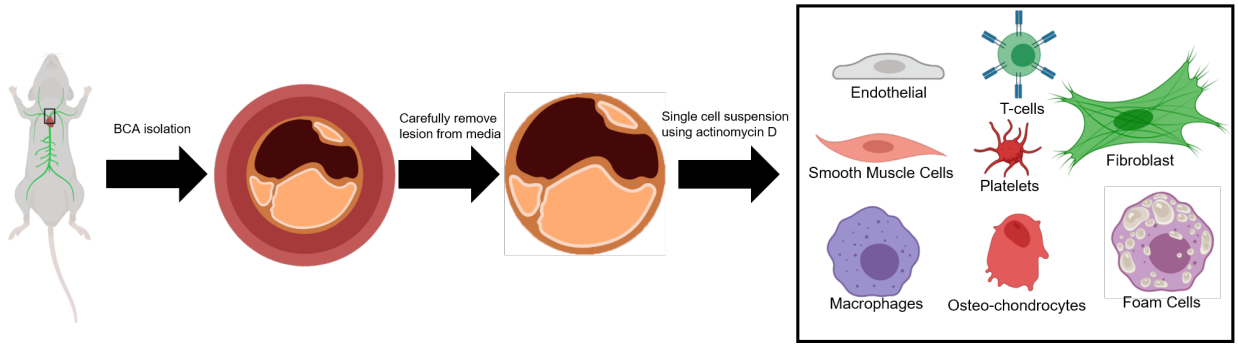


Figure 1: Workflow showing the steps necessary to get single cell isolation from advanced BCA lesions

Experimental Design

SMC lineage tracing Myh11-Cre^{ERT2}/Rosa-eYFP/*SR-B1*$^{\Delta CT/\Delta CT}$/LDLR$^{-/-}$ mice treated AZM198 for 24 weeks on WD vs 24 weeks WD control. First, let's load the necessary packages for the analysis. For installation, please refer to package instructions.

The following code block sets three variables that you must update to run this code successfully on your own experiment. These variables determine what your experiment is named and where R looks for the Cell Ranger output. The "ids" variable should be your sample names and **must** be the names of directories that contain the "outs" folders created by Cell Ranger.

```
experiment_name = "AZM"
dataset_loc <- "/project/Owens_Rivanna/01.Data_Analysis/2022.09.scRNA-seq_Sohel/"
ids <- c("WT_ALL", "AZM_ALL", "WT_YFP", "AZM_YFP")
```

Sequencing metrics

Summary metrics for each sample, similar output to what Cell Ranger generates but show all libraries together.

```
d10x.metrics <- lapply(ids, function(i){
  metrics <- read.csv(file.path(dataset_loc, paste0(i,"_CountMatrix/outs"),"metrics_summary.csv"), colC
})
experiment.metrics <- do.call("rbind", d10x.metrics)
rownames(experiment.metrics) <- ids

sequencing.metrics <- data.frame(t(experiment.metrics[,c(1:19)]))
rownames(sequencing.metrics) <- gsub("\\.", " ", rownames(sequencing.metrics))

sequencing.metrics %>%
  kable(caption = "<center> <b> Cell Ranger Results </b>") %>%
  pack_rows("Overview", 1, 3, label_row_css = "background-color: #666; color: #fff;") %>%
  pack_rows("Sequencing Characteristics", 4, 9, label_row_css = "background-color: #666; color: #fff;")
  pack_rows("Mapping Characteristics", 10, 19, label_row_css = "background-color: #666; color: #fff;") %
  kable_classic_2()
```

Load the Cell Ranger Matrix Data

To use the Cell Ranger defined cells, changed "raw_feature_bc_matrix.h5" to "filtered_feature_bc_matrix.h5". Here, I am using the raw data.

```
d10x.data <- lapply(ids, function(i){
  d10x <- Read10X_h5(file.path(dataset_loc, paste0(i, "_CountMatrix/outs"),"raw_feature_bc_matrix.h5"))
  colnames(d10x) <- paste(sapply(strsplit(colnames(d10x),split="-"),'[[',1L),i,sep="-")
  d10x
})
names(d10x.data) <- ids
```

Let's recreate the barcode rank plot from the Cell Ranger web summary file.

Barcode Rank Plot for the gene expression data enables one to assess library quality. Ideally there is a steep drop-off separating high UMI count cells from low UMI count background noise.

```
plot_cellranger_cells <- function(ind){
  xbreaks = c(1,1e1,1e2,1e3,1e4,1e5,1e6)
  xlabels = c("1","10","100","1000","10k","100K","1M")
  ybreaks = c(1,2,5,10,20,50,100,200,500,1000,2000,5000,10000,20000,50000,100000,200000,500000,1000000)
  ylabels = c("1","2","5","10","2","5","100","2","5","1000","2","5","10k","2","5","100K","2","5","1M")

  pl1 <- data.frame(index=seq.int(1,ncol(d10x.data[[ind]])),
```

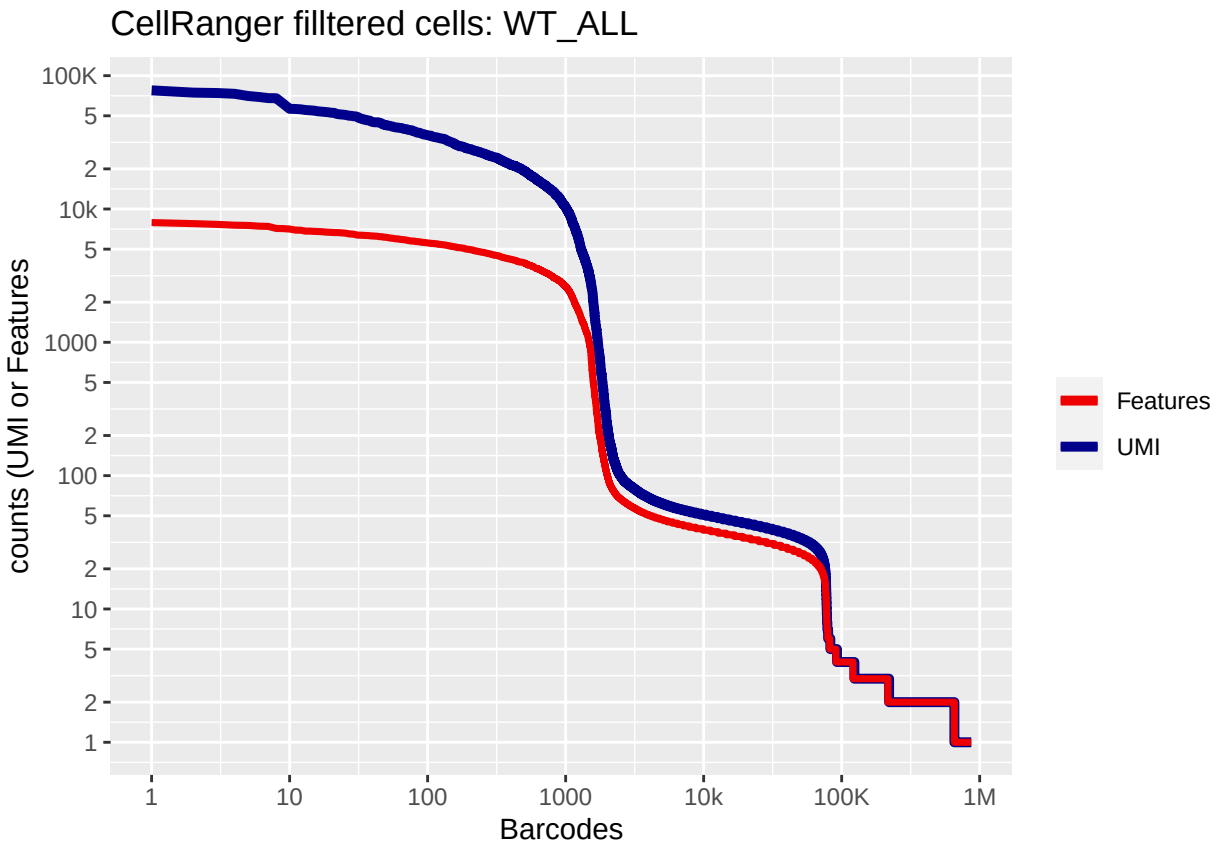
Table 1:   Cell Ranger Results

	WT_ALL	AZM_ALL	WT_YFP	AZM_YFP
Overview				
Estimated Number of Cells	1,813	3,406	2,497	2,090
Mean Reads per Cell	238,688	107,509	162,141	224,135
Median Genes per Cell	2,877	1,967	3,620	3,836
Sequencing Characteristics				
Number of Reads	432,740,609	366,176,600	404,865,040	468,442,701
Valid Barcodes	97.9%	97.9%	98.0%	97.9%
Sequencing Saturation	89.9%	83.6%	85.3%	87.1%
Q30 Bases in Barcode	96.7%	96.2%	96.2%	96.2%
Q30 Bases in RNA Read	92.7%	92.8%	93.5%	92.9%
Q30 Bases in UMI	93.7%	92.2%	93.5%	93.6%
Mapping Characteristics				
Reads Mapped to Genome	93.4%	94.6%	94.6%	94.7%
Reads Mapped Confidently to Genome	91.1%	92.2%	92.4%	92.8%
Reads Mapped Confidently to Intergenic Regions	6.1%	4.6%	3.3%	3.2%
Reads Mapped Confidently to Intronic Regions	13.9%	12.4%	10.9%	10.7%
Reads Mapped Confidently to Exonic Regions	71.1%	75.2%	78.2%	79.0%
Reads Mapped Confidently to Transcriptome	67.7%	72.0%	74.9%	75.6%
Reads Mapped Antisense to Gene	1.3%	1.3%	1.5%	1.5%
Fraction Reads in Cells	90.7%	94.1%	95.3%	94.3%
Total Genes Detected	19,843	20,641	19,116	18,917
Median UMI Counts per Cell	11,949	6,922	15,300	18,919

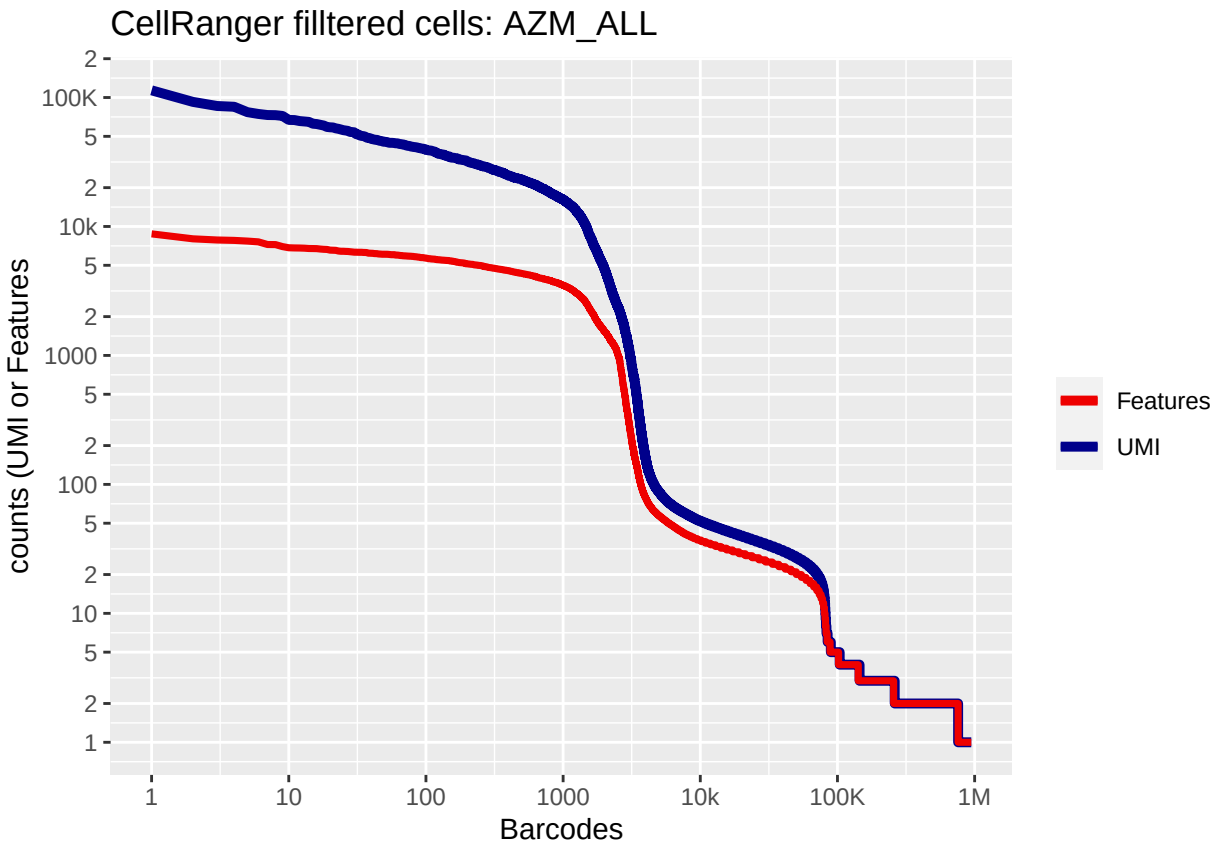
```
                    nCount_RNA = sort(Matrix:::colSums(d10x.data[[ind]])+1,decreasing=T),
                    nFeature_RNA = sort(Matrix:::colSums(d10x.data[[ind]]>0)+1,decreasing=T)) %>%
    ggplot() +
    scale_color_manual(values=c("red2","blue4"), labels=c("Features", "UMI"), name=NULL) +
    ggtitle(paste("CellRanger filltered cells:",ids[ind],sep=" ")) + xlab("Barcodes") + ylab("counts (U
    scale_x_continuous(trans = 'log2', breaks=xbreaks, labels = xlabels) +
    scale_y_continuous(trans = 'log2', breaks=ybreaks, labels = ylabels) +
    geom_line(aes(x=index, y=nCount_RNA, color = "UMI"), size=1.75) +
    geom_line(aes(x=index, y=nFeature_RNA, color = "Features"), size=1.25)

  return(pl1)
}
plot_cellranger_cells(1)
```

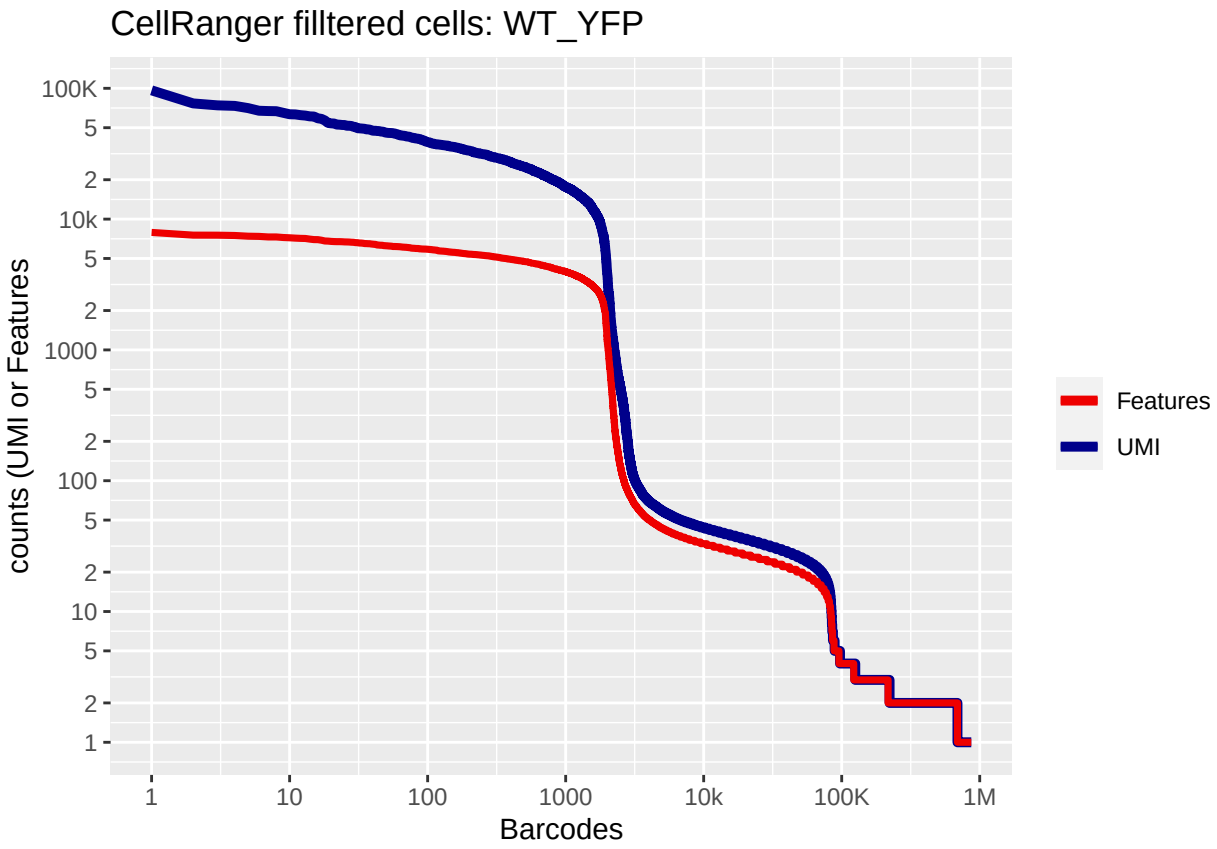
```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
```



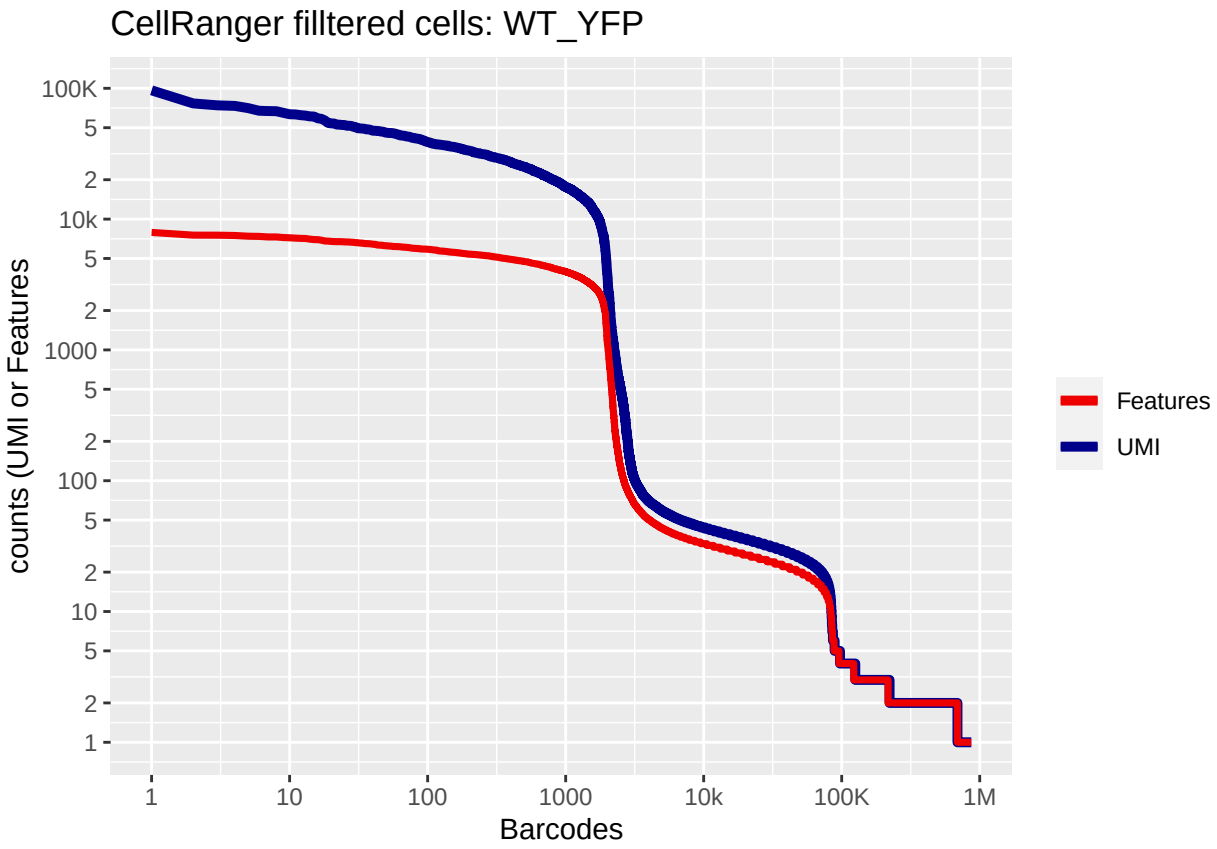

```
plot_cellranger_cells(2)
```




```
plot_cellranger_cells(3)
```




```
plot_cellranger_cells(3)
```



Create Seurat Object

```
#combine list
experiment.data <- do.call("cbind", d10x.data)
## Basic filtering. Filter criteria: remove genes that do not occur in a minimum of 0 cells and remove
 experiment.aggregate <- CreateSeuratObject(
  experiment.data,
  project = experiment_name,
  min.cells = 0,
  min.features = 300,
  names.field = 2,
  names.delim = "\\-")
str(experiment.aggregate)
```

```
## Formal class 'Seurat' [package "SeuratObject"] with 13 slots
##   ..@ assays      :List of 1
##   .. ..$ RNA:Formal class 'Assay' [package "SeuratObject"] with 8 slots
##   .. .. .. .. ..@ counts       :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. .. .. .. .. .. ..@ i       : int [1:28451868] 63 120 149 194 206 214 219 233 296 356 ...
##   .. .. .. .. .. .. ..@ p       : int [1:8866] 0 496 3758 6338 8131 9260 13718 17849 20892 21829 ...
##   .. .. .. .. .. .. ..@ Dim     : int [1:2] 32286 8865
##   .. .. .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. .. .. ..$ : chr [1:32286] "Xkr4" "Gm1992" "Gm19938" "Gm37381" ...
##   .. .. .. .. .. .. .. ..$ : chr [1:8865] "AAACGAAAGAGATTCA-WT_ALL" "AAACGAAAGTAACCGG-WT_ALL" "AAACGAAG
##   .. .. .. .. .. .. ..@ x       : num [1:28451868] 1 1 1 1 1 1 1 1 2 1 ...
##   .. .. .. .. .. .. ..@ factors : list()
##   .. .. .. ..@ data         :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
```

```
##   .. .. .. .. .. ..@ i       : int [1:28451868] 63 120 149 194 206 214 219 233 296 356 ...
##   .. .. .. .. .. ..@ p       : int [1:8866] 0 496 3758 6338 8131 9260 13718 17849 20892 21829 ...
##   .. .. .. .. .. ..@ Dim     : int [1:2] 32286 8865
##   .. .. .. .. .. ..@ Dimnames:List of 2
##   .. .. .. .. .. .. ..$ : chr [1:32286] "Xkr4" "Gm1992" "Gm19938" "Gm37381" ...
##   .. .. .. .. .. .. ..$ : chr [1:8865] "AAACGAAAGAGATTCA-WT_ALL" "AAACGAAAGTAACCGG-WT_ALL" "AAACGAAG
##   .. .. .. .. .. ..@ x       : num [1:28451868] 1 1 1 1 1 1 1 1 2 1 ...
##   .. .. .. .. .. ..@ factors : list()
##   .. .. .. ..@ scale.data   : num[0 , 0 ]
##   .. .. .. ..@ key          : chr "rna_"
##   .. .. .. ..@ assay.orig   : NULL
##   .. .. .. ..@ var.features : logi(0)
##   .. .. .. ..@ meta.features:'data.frame':   32286 obs. of  0 variables
##   .. .. .. ..@ misc         : list()
##   ..@ meta.data   :'data.frame': 8865 obs. of  3 variables:
##   .. ..$ orig.ident  : Factor w/ 4 levels "AZM_ALL","AZM_YFP",..: 3 3 3 3 3 3 3 3 3 3 ...
##   .. ..$ nCount_RNA  : num [1:8865] 2568 15859 9460 6730 4029 ...
##   .. ..$ nFeature_RNA: int [1:8865] 496 3262 2580 1793 1129 4458 4131 3043 937 5528 ...
##   ..@ active.assay: chr "RNA"
##   ..@ active.ident: Factor w/ 4 levels "AZM_ALL","AZM_YFP",..: 3 3 3 3 3 3 3 3 3 3 ...
##   .. ..- attr(*, "names")= chr [1:8865] "AAACGAAAGAGATTCA-WT_ALL" "AAACGAAAGTAACCGG-WT_ALL" "AAACGAA
##   ..@ graphs      : list()
##   ..@ neighbors   : list()
##   ..@ reductions  : list()
##   ..@ images      : list()
##   ..@ project.name: chr "AZM"
##   ..@ misc        : list()
##   ..@ version     :Classes 'package_version', 'numeric_version'  hidden list of 1
##   .. ..$ : int [1:3] 4 1 3
##   ..@ commands    : list()
##   ..@ tools       : list()
```

```
#Lets spend a little time getting to know the Seurat object
slotNames(experiment.aggregate)
```

```
##  [1] "assays"       "meta.data"    "active.assay" "active.ident" "graphs"
##  [6] "neighbors"    "reductions"   "images"       "project.name" "misc"
## [11] "version"      "commands"     "tools"
```

```
head(experiment.aggregate[[]])
```

```
##                         orig.ident nCount_RNA nFeature_RNA
## AAACGAAAGAGATTCA-WT_ALL     WT_ALL       2568          496
## AAACGAAAGTAACCGG-WT_ALL     WT_ALL      15859         3262
## AAACGAAGTAGCTCGC-WT_ALL     WT_ALL       9460         2580
## AAACGCTCATCACGGC-WT_ALL     WT_ALL       6730         1793
## AAACGCTTCACGAGGA-WT_ALL     WT_ALL       4029         1129
## AAAGAACAGAGATGCC-WT_ALL     WT_ALL      17640         4458
```

```
str(d10x.data)
```

```
## List of 4
```

```
##  $ WT_ALL :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:8060844] 32207 11490 18209 26794 6647 32204 32200 1626 11003 23205 ...
##   .. ..@ p       : int [1:869119] 0 1 2 3 4 4 6 7 8 9 ...
##   .. ..@ Dim     : int [1:2] 32286 869118
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:32286] "Xkr4" "Gm1992" "Gm19938" "Gm37381" ...
##   .. .. ..$ : chr [1:869118] "AAACCCAAGAAACCCA-WT_ALL" "AAACCCAAGAAACTAC-WT_ALL" "AAACCCAAGAACAAGG-W
##   .. ..@ x       : num [1:8060844] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..@ factors : list()
##  $ AZM_ALL:Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:10814232] 22233 6972 11868 19974 27851 32200 30425 4163 15214 27809 ...
##   .. ..@ p       : int [1:945024] 0 1 1 6 7 7 7 7 8 8 ...
##   .. ..@ Dim     : int [1:2] 32286 945023
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:32286] "Xkr4" "Gm1992" "Gm19938" "Gm37381" ...
##   .. .. ..$ : chr [1:945023] "AAACCCAAGAAACACT-AZM_ALL" "AAACCCAAGAAACCAT-AZM_ALL" "AAACCCAAGAAACCCA
##   .. ..@ x       : num [1:10814232] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..@ factors : list()
##  $ WT_YFP :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:10754324] 4873 18209 16973 27555 29613 2088 1029 5450 4946 4358 ...
##   .. ..@ p       : int [1:868532] 0 0 2 2 2 4 5 5 6 7 ...
##   .. ..@ Dim     : int [1:2] 32286 868531
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:32286] "Xkr4" "Gm1992" "Gm19938" "Gm37381" ...
##   .. .. ..$ : chr [1:868531] "AAACCCAAGAAACCCA-WT_YFP" "AAACCCAAGAAACTGT-WT_YFP" "AAACCCAAGAAAGCGA-W
##   .. ..@ x       : num [1:10754324] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..@ factors : list()
##  $ AZM_YFP:Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:10485445] 11003 5626 1752 4163 4604 7070 14959 19426 22952 23253 ...
##   .. ..@ p       : int [1:890421] 0 1 2 12 12 13 13 14 15 16 ...
##   .. ..@ Dim     : int [1:2] 32286 890420
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:32286] "Xkr4" "Gm1992" "Gm19938" "Gm37381" ...
##   .. .. ..$ : chr [1:890420] "AAACCCAAGAAACACT-AZM_YFP" "AAACCCAAGAAACCAT-AZM_YFP" "AAACCCAAGAAACCCA
##   .. ..@ x       : num [1:10485445] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..@ factors : list()
```

```
str(experiment.data)
```

```
## Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   ..@ i       : int [1:40114845] 32207 11490 18209 26794 6647 32204 32200 1626 11003 23205 ...
##   ..@ p       : int [1:3573093] 0 1 2 3 4 4 6 7 8 9 ...
##   ..@ Dim     : int [1:2] 32286 3573092
##   ..@ Dimnames:List of 2
##   .. ..$ : chr [1:32286] "Xkr4" "Gm1992" "Gm19938" "Gm37381" ...
##   .. ..$ : chr [1:3573092] "AAACCCAAGAAACCCA-WT_ALL" "AAACCCAAGAAACTAC-WT_ALL" "AAACCCAAGAACAAGG-WT_
##   ..@ x       : num [1:40114845] 1 1 1 1 1 1 1 1 1 1 ...
##   ..@ factors : list()
```

```
tail(colnames(experiment.data))
```

```
## [1] "TTTGTTGTCTTTACAC-AZM_YFP" "TTTGTTGTCTTTCCGG-AZM_YFP"
## [3] "TTTGTTGTCTTTCGAT-AZM_YFP" "TTTGTTGTCTTTCTAG-AZM_YFP"
## [5] "TTTGTTGTCTTTGATC-AZM_YFP" "TTTGTTGTCTTTGGAG-AZM_YFP"
```

Calculate percentage of mitochondrial transcripts

```
experiment.aggregate$percent.mito <- PercentageFeatureSet(experiment.aggregate, pattern = "^mt-")
summary(experiment.aggregate$percent.mito)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.068   3.008   5.197   4.560  85.160
```

Calculate percentage of hemoglobin transcripts

```
experiment.aggregate$percent.hemo <- PercentageFeatureSet(experiment.aggregate, pattern = "^Hb[^(p)]")
summary(experiment.aggregate$percent.hemo)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0064  0.1906  0.0205 94.1771
```

Basic QA/QC

Let's examine the distribution of features (genes) per cell, number of UMIs per cell, and percent mitochondrial and hemoglobin reads per cell in each of the samples ### Quantile tables

```
kable(do.call("cbind", tapply(experiment.aggregate$nFeature_RNA,
                      Idents(experiment.aggregate),quantile,probs=seq(0,1,0.05))),
      caption = "<center> <b> 5% Quantiles of Genes/Cell by Sample </b>") %>% kable_classic_2()
```

\begin{table}

\caption{

5% Quantiles of Genes/Cell by Sample }

	AZM_ALL	AZM_YFP	WT_ALL	WT_YFP
0%	300.00	301.00	300.0	300.00
5%	426.00	715.45	488.8	579.75
10%	625.90	1699.60	903.6	1074.10
15%	944.40	2449.10	1170.0	2069.95
20%	1136.00	2789.80	1390.6	2597.60
25%	1266.00	3100.00	1658.0	2887.75
30%	1421.70	3299.00	1933.0	3085.10
35%	1586.00	3482.80	2311.8	3290.65
40%	1768.00	3638.20	2616.0	3468.80
45%	2080.20	3805.30	2833.6	3634.55
50%	2455.50	3956.00	3019.0	3804.00
55%	2827.45	4115.00	3249.4	3951.00
60%	3127.20	4235.00	3443.0	4086.80
65%	3409.35	4404.75	3678.2	4259.70
70%	3648.30	4560.00	3928.0	4442.60
75%	3872.25	4769.75	4153.0	4637.50
80%	4136.00	4962.60	4399.8	4830.40
85%	4399.00	5176.95	4726.2	5059.60
90%	4773.10	5428.50	5129.4	5351.10
95%	5397.10	5861.55	5693.0	5802.45
100%	8763.00	7712.00	7910.0	7912.00

\end{table}

```
kable(do.call("cbind", tapply(experiment.aggregate$nCount_RNA,
                              Idents(experiment.aggregate),quantile,probs=seq(0,1,0.05))),
      caption = "<center> <b> 5% Quantiles of UMI/Cell by Sample </b>") %>% kable_classic_2()
```

\begin{table}

\caption{

5% Quantiles of UMI/Cell by Sample }

	AZM_ALL	AZM_YFP	WT_ALL	WT_YFP
0%	474.00	441.00	368.0	462.00
5%	1212.00	2267.50	1604.0	1533.90
10%	1743.80	5487.40	2816.8	3207.30
15%	2158.70	8869.25	3704.4	6624.00
20%	2564.40	11194.80	4459.6	8925.20
25%	3119.75	12783.75	5560.0	10660.00
30%	3883.50	14442.90	6964.4	11984.20
35%	4910.20	15908.35	8606.2	13360.60
40%	5863.60	17308.60	10065.2	14376.80
45%	7172.40	18691.25	11467.8	15463.10
50%	9101.50	19907.00	12936.0	16636.00
55%	11548.15	21143.40	14203.6	17508.50
60%	13585.20	22535.60	15557.6	18976.20
65%	15328.75	24038.50	17158.4	20306.65
70%	16971.50	25956.60	19210.4	22021.20
75%	18952.75	27850.50	21156.0	24048.50
80%	21267.40	30231.80	23397.4	26070.20
85%	23742.90	32963.85	26233.8	28895.35
90%	27693.80	37452.00	29647.6	32109.30
95%	34268.40	43465.60	37374.0	37561.85
100%	113692.00	95830.00	77637.0	96509.00

\end{table}

```
kable(round(do.call("cbind", tapply(experiment.aggregate$percent.mito, Idents(experiment.aggregate),qua
      caption = "<center> <b> 5% Quantiles of Percent Mitochondria by Sample </b>") %>% kable_classic_2
```

\begin{table}

\caption{

5% Quantiles of Percent Mitochondria by Sample }

	AZM_ALL	AZM_YFP	WT_ALL	WT_YFP
0%	0.000	0.000	0.000	0.000
5%	1.344	1.001	0.480	1.054
10%	1.796	1.300	2.021	1.328
15%	2.051	1.485	2.468	1.469
20%	2.304	1.649	2.848	1.596
25%	2.495	1.774	3.140	1.740
30%	2.661	1.922	3.390	1.868
35%	2.824	2.043	3.623	1.972
40%	2.997	2.182	3.860	2.080
45%	3.165	2.313	4.075	2.203
50%	3.344	2.446	4.361	2.328
55%	3.543	2.597	4.614	2.468
60%	3.766	2.791	4.880	2.588
65%	4.029	2.959	5.239	2.797
70%	4.345	3.180	5.551	3.024
75%	4.730	3.457	6.119	3.334
80%	5.226	3.859	6.908	3.733
85%	6.198	4.687	7.954	4.429
90%	9.267	6.316	11.061	5.839
95%	21.893	11.888	26.333	10.706
100%	85.160	74.256	81.148	60.248

\end{table}

```
kable(do.call("cbind", tapply(experiment.aggregate$percent.hemo,
                              Idents(experiment.aggregate),quantile,probs=seq(0,1,0.05))),
      caption = "<center> <b> 5% Quantiles of Percent Hemoglobin by Sample </b>") %>% kable_classic_2()
```

\begin{table}

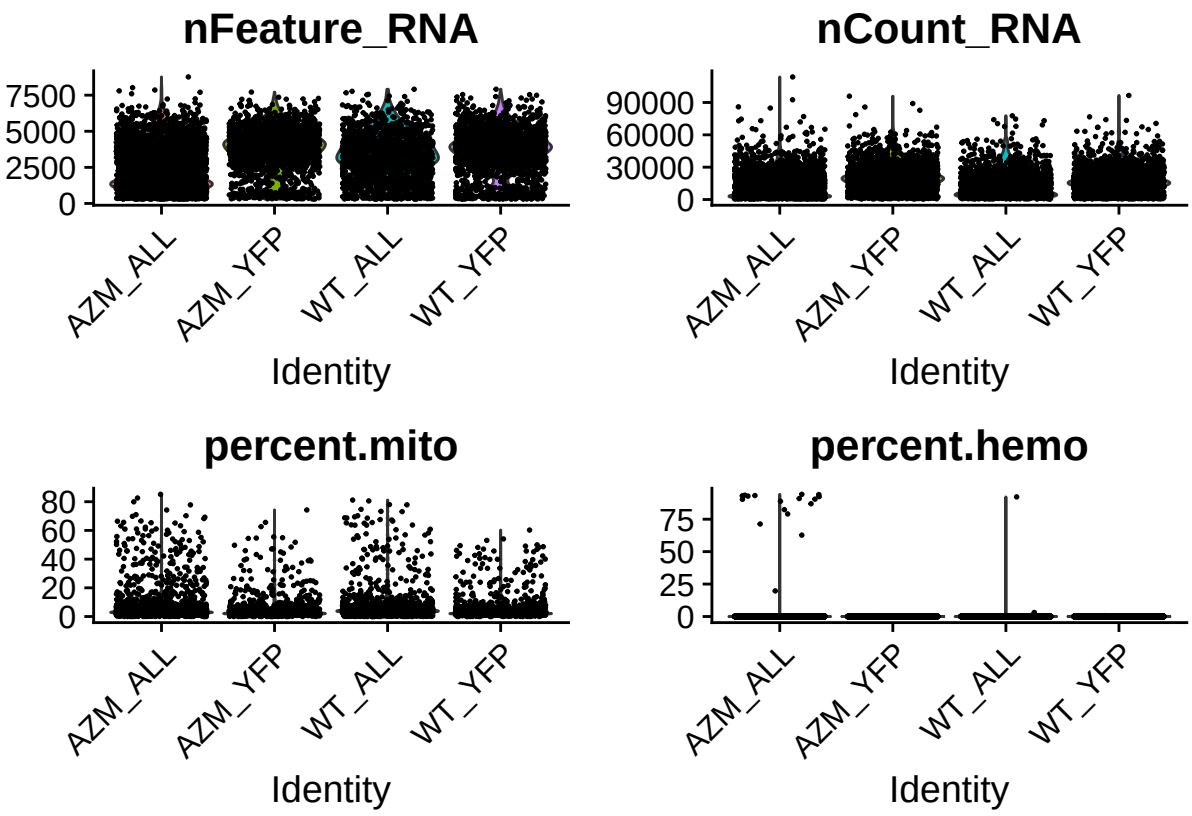
\caption{

5% Quantiles of Percent Hemoglobin by Sample }

	AZM_ALL	AZM_YFP	WT_ALL	WT_YFP
0%	0.0000000	0.0000000	0.0000000	0.0000000
5%	0.0000000	0.0000000	0.0000000	0.0000000
10%	0.0000000	0.0000000	0.0000000	0.0000000
15%	0.0000000	0.0000000	0.0000000	0.0000000
20%	0.0000000	0.0000000	0.0000000	0.0000000
25%	0.0000000	0.0000000	0.0000000	0.0000000
30%	0.0000000	0.0024203	0.0000000	0.0000000
35%	0.0000000	0.0043749	0.0000000	0.0000000
40%	0.0000000	0.0055584	0.0000000	0.0040137
45%	0.0039054	0.0072373	0.0000000	0.0054779
50%	0.0060842	0.0087624	0.0030650	0.0066279
55%	0.0089107	0.0109030	0.0053554	0.0079413
60%	0.0117669	0.0135244	0.0069437	0.0095793
65%	0.0150830	0.0168756	0.0082516	0.0116957
70%	0.0192597	0.0210349	0.0102726	0.0140230
75%	0.0253478	0.0281033	0.0130890	0.0176155
80%	0.0337321	0.0375098	0.0168501	0.0227149
85%	0.0429298	0.0523367	0.0225146	0.0280358
90%	0.0623908	0.0773273	0.0289955	0.0382037
95%	0.1085659	0.1372864	0.0449906	0.0622437
100%	94.1770648	0.6525231	92.1786321	0.6607804

\end{table} ## Violin Plots of 1) number of genes, 2) number of UMI, 3) percent mitochondrial genes and 4) percent hemoglobin genes

```
VlnPlot(
  experiment.aggregate,
  features = c("nFeature_RNA", "nCount_RNA","percent.mito", "percent.hemo"),
  ncol = 2, pt.size = 0.3)
```



Ridge Plot of 1) number of genes, 2) number of UMI, 3) percent mitochondrial genes and 4) percent hemoglobin genes

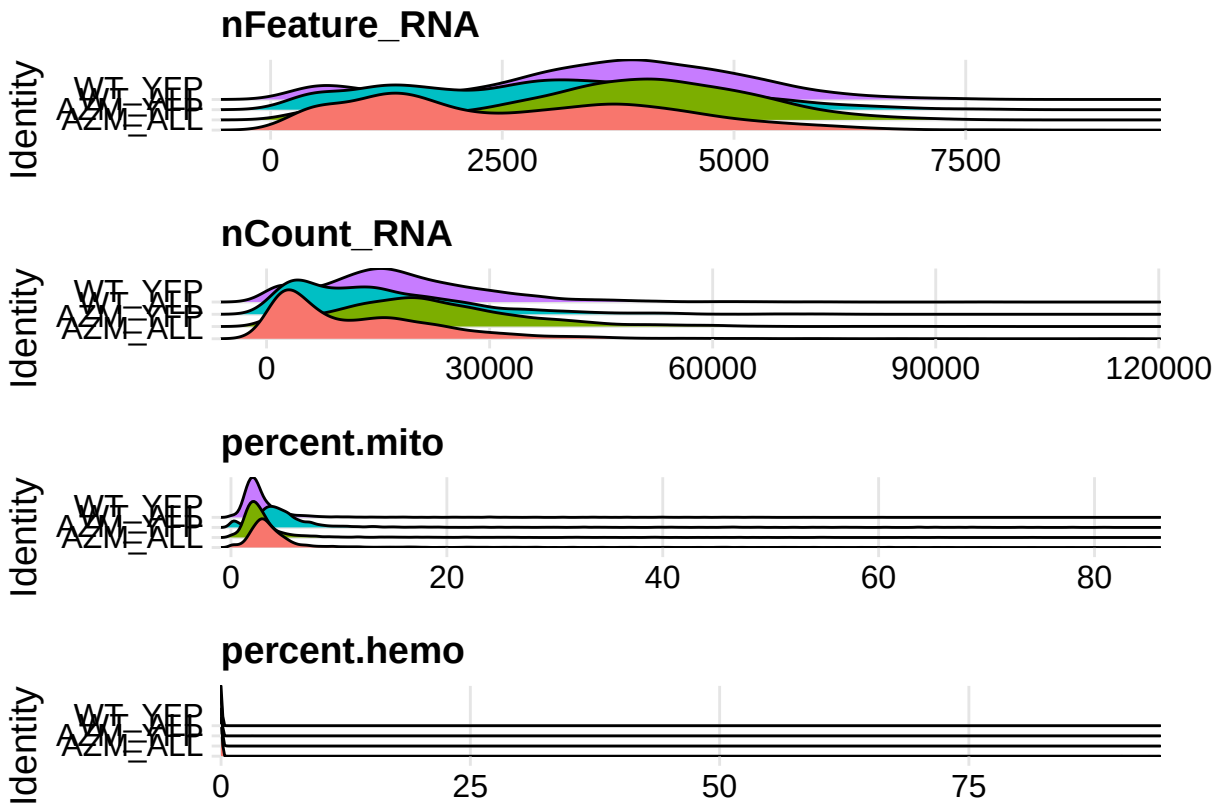
```
RidgePlot(experiment.aggregate, features=c("nFeature_RNA","nCount_RNA", "percent.mito", "percent.hemo")
```

```
## Picking joint bandwidth of 278
```

```
## Picking joint bandwidth of 2170
```

```
## Picking joint bandwidth of 0.308
```

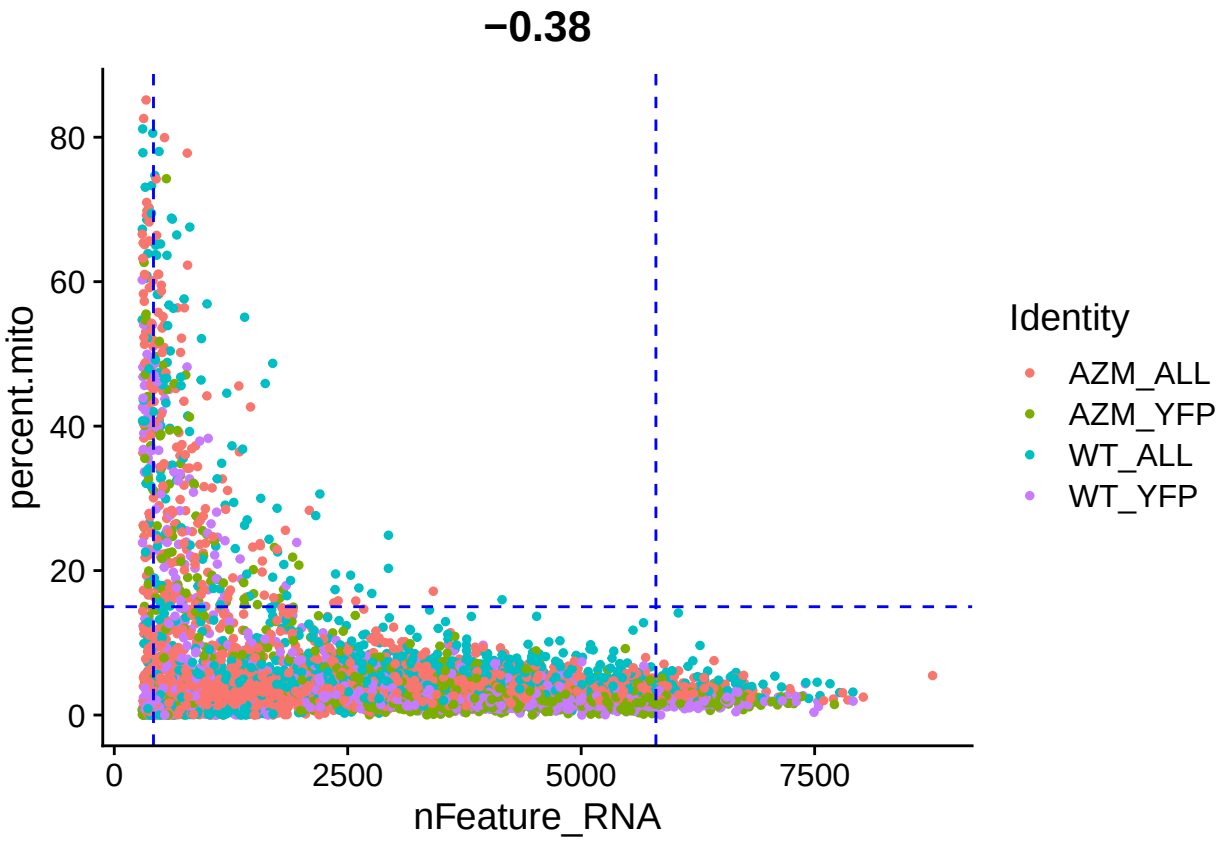
```
## Picking joint bandwidth of 0.00303
```



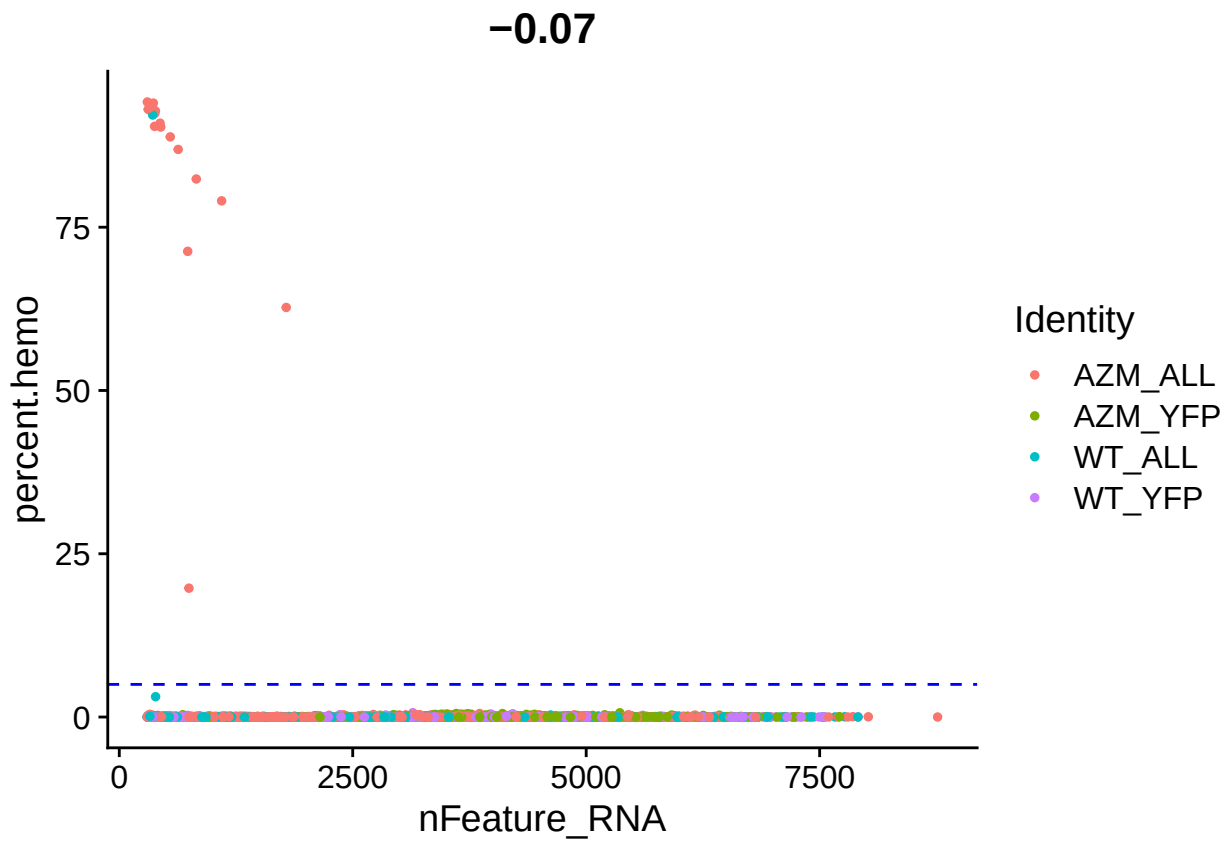
Scatter plot of gene expression across cells, (colored by sample), drawing horizontal and verticale lines at proposed filtering cutoffs.

```
png(file=".png",
width=600, height=350)
FeatureScatter(experiment.aggregate,
               feature1 = "nCount_RNA",
               feature2 = "nFeature_RNA",
               shuffle = TRUE) +
    geom_hline(yintercept = c(420,5800), linetype = 2, color="blue")
```

```
  FeatureScatter(experiment.aggregate,
               feature1 = "nFeature_RNA",
               feature2 = "percent.mito",
               shuffle = TRUE) +
    geom_hline(yintercept = 15, linetype = 2, color="blue") +
   geom_vline(xintercept = c(420,5800), linetype = 2, color="blue" )
```




```
FeatureScatter(experiment.aggregate,
               feature1 = "nFeature_RNA",
               feature2 = "percent.hemo",
               shuffle = TRUE) +
  geom_hline(yintercept = 5, linetype = 32, color="blue")
```



how many cells before QC selection?

```
table(experiment.aggregate$orig.ident)
```

```
##
## AZM_ALL AZM_YFP  WT_ALL  WT_YFP
##    3020    1928    1677    2240
```

QC selection:

-We define poor quality samples for mitochondrial content as cells which has more than 15% of cell reads originating from the mitochondrial genes and more than 5% of hemoglobin genes -We selected the cut-offs of removing genes below 5th percentile and genes above 95th percentile

```
experiment.aggregate.Anita <- experiment.aggregate
#these filtering cutoffs are a bit arbitrary and it might be different for different dataset, cell line
experiment.aggregate.Anita <- subset(experiment.aggregate, percent.mito <= 15.0)
experiment.aggregate.Anita <- subset(experiment.aggregate.Anita, percent.hemo <= 5.0)
experiment.aggregate.Anita <- subset(experiment.aggregate.Anita, nFeature_RNA >= 420 & nFeature_RNA <= 5
experiment.aggregate.Anita
```

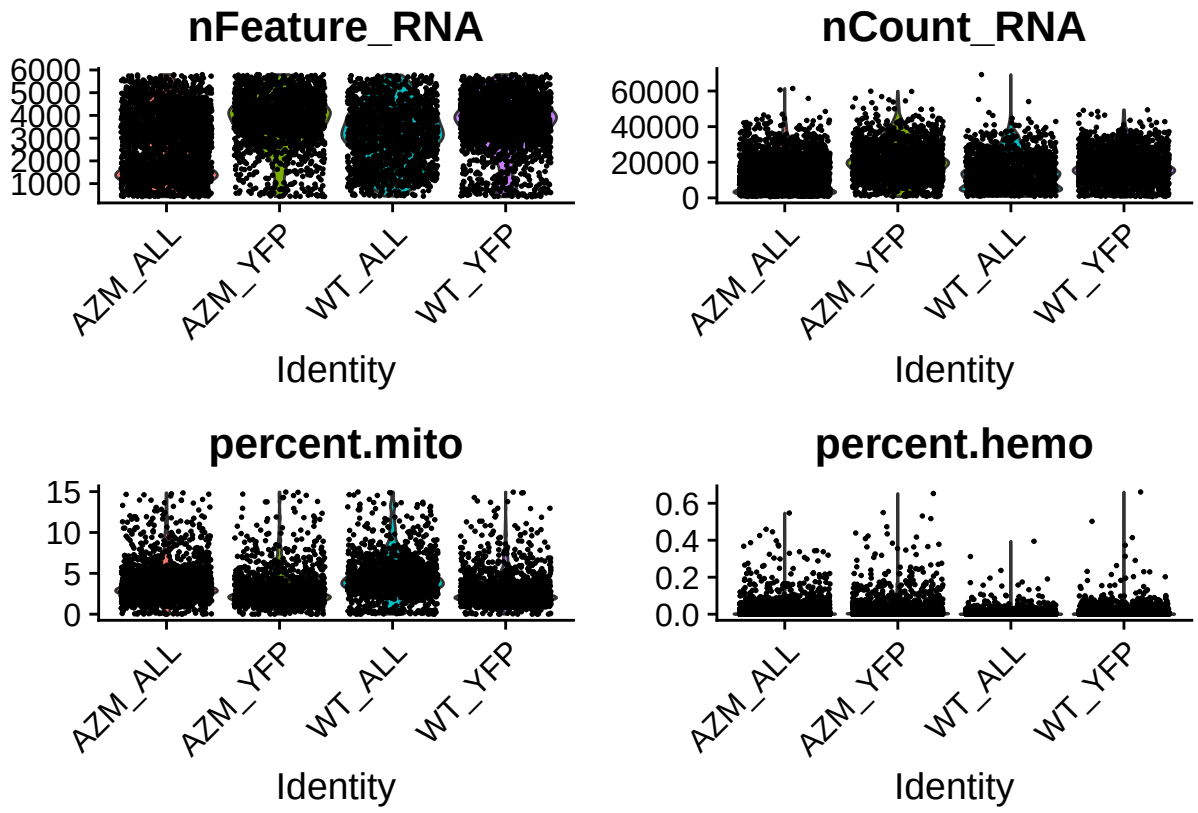
```
## An object of class Seurat
## 32286 features across 7830 samples within 1 assay
## Active assay: RNA (32286 features, 0 variable features)
```

```
#how many cells after QC selection?
table(experiment.aggregate.Anita$orig.ident)
```

```
##
## AZM_ALL AZM_YFP  WT_ALL  WT_YFP
##    2644    1730    1447    2009
```

Explore plots after filtering

```
VlnPlot(
  experiment.aggregate.Anita,
  features = c("nFeature_RNA", "nCount_RNA","percent.mito", "percent.hemo"),
  ncol = 2, pt.size = 0.3)
```



Ridge plot after QC selection

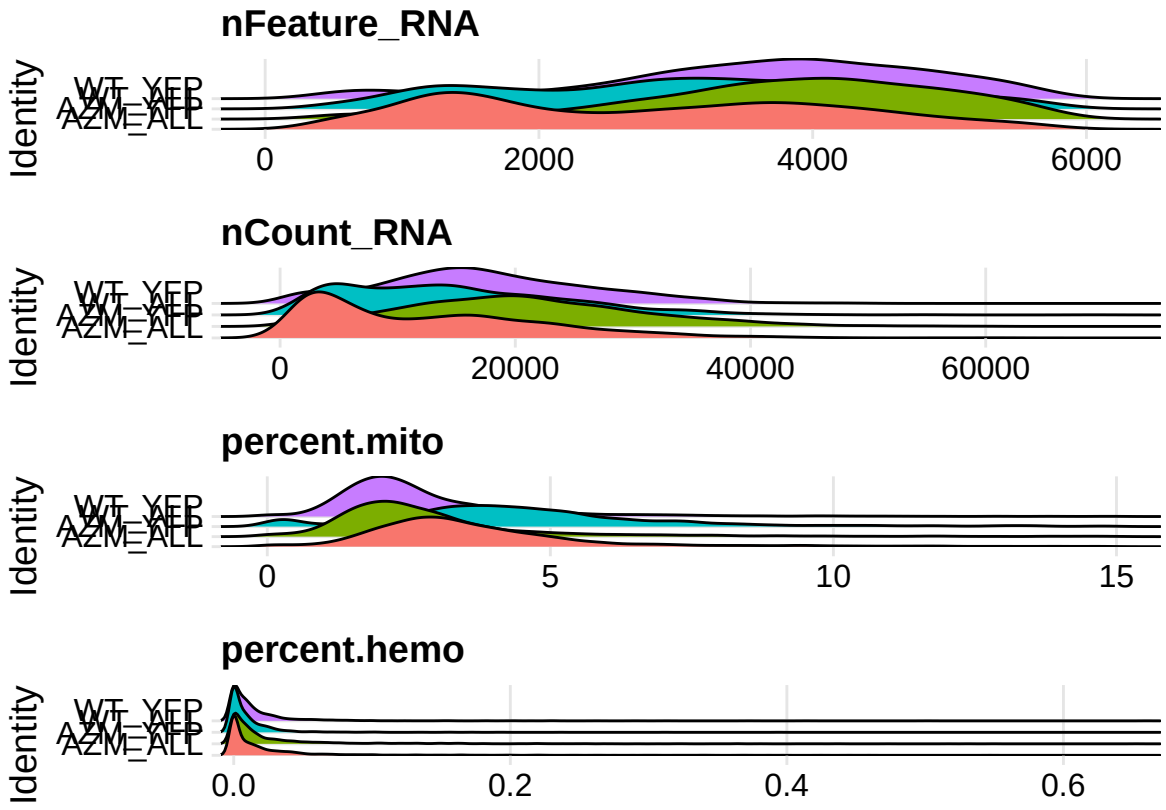
```
RidgePlot(experiment.aggregate.Anita, features=c("nFeature_RNA","nCount_RNA", "percent.mito", "percent.h
```

```
## Picking joint bandwidth of 248
```

```
## Picking joint bandwidth of 1870
```

```
## Picking joint bandwidth of 0.273
```

```
## Picking joint bandwidth of 0.00306
```



Finally, save the Seurat object and view the object.

```
save(experiment.aggregate.Anita,file="pre_sample_corrected-Anita.RData")
```

Session Information

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.5 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
```

```
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices datasets  utils     methods
## [8] base
##
## other attached packages:
##  [1] tinytex_0.42                scran_1.22.1
##  [3] scuttle_1.4.0               SingleCellExperiment_1.16.0
##  [5] SummarizedExperiment_1.24.0 GenomicRanges_1.46.1
##  [7] GenomeInfoDb_1.30.1         MatrixGenerics_1.6.0
##  [9] matrixStats_0.63.0          sva_3.42.0
## [11] BiocParallel_1.28.3         genefilter_1.76.0
## [13] mgcv_1.8-36                 nlme_3.1-152
## [15] org.Hs.eg.db_3.14.0         topGO_2.46.0
## [17] SparseM_1.81                GO.db_3.14.0
## [19] AnnotationDbi_1.56.2        IRanges_2.28.0
## [21] S4Vectors_0.32.4            Biobase_2.54.0
## [23] graph_1.72.0                BiocGenerics_0.40.0
## [25] limma_3.50.3                biomaRt_2.50.3
## [27] reshape2_1.4.4              dplyr_1.0.10
## [29] ggplot2_3.4.0               kableExtra_1.3.4
## [31] knitr_1.42                  hdf5r_1.3.7
## [33] SeuratObject_4.1.3          Seurat_4.3.0
## [35] rmarkdown_2.18              magrittr_2.0.3
##
## loaded via a namespace (and not attached):
##   [1] utf8_1.2.2                spatstat.explore_3.0-5
##   [3] reticulate_1.26           tidyselect_1.2.0
##   [5] RSQLite_2.2.19            htmlwidgets_1.5.4
##   [7] grid_4.1.1                Rtsne_0.16
##   [9] ScaledMatrix_1.2.0        munsell_0.5.0
##  [11] codetools_0.2-18          ica_1.0-3
##  [13] statmod_1.4.37            future_1.29.0
##  [15] miniUI_0.1.1.1            withr_2.5.0
##  [17] spatstat.random_3.0-1     colorspace_2.0-3
##  [19] progressr_0.11.0          filelock_1.0.2
##  [21] highr_0.9                 rstudioapi_0.14
##  [23] ROCR_1.0-11               tensor_1.5
##  [25] listenv_0.8.0             labeling_0.4.2
##  [27] GenomeInfoDbData_1.2.7    polyclip_1.10-4
##  [29] farver_2.1.1              bit64_4.0.5
##  [31] parallelly_1.32.1         vctrs_0.5.1
##  [33] generics_0.1.3            xfun_0.36
##  [35] BiocFileCache_2.2.1       R6_2.5.1
##  [37] rsvd_1.0.5                locfit_1.5-9.6
##  [39] DelayedArray_0.20.0       bitops_1.0-7
##  [41] spatstat.utils_3.0-1      cachem_1.0.6
##  [43] assertthat_0.2.1          promises_1.2.0.1
##  [45] scales_1.2.1              gtable_0.3.1
##  [47] beachmat_2.10.0           globals_0.16.2
##  [49] goftest_1.2-3             rlang_1.0.6
##  [51] systemfonts_1.0.4         splines_4.1.1
##  [53] lazyeval_0.2.2            spatstat.geom_3.0-3
```

```
##  [55] yaml_2.3.7               abind_1.4-5
##  [57] httpuv_1.6.6             tools_4.1.1
##  [59] ellipsis_0.3.2           RColorBrewer_1.1-3
##  [61] ggridges_0.5.4           Rcpp_1.0.9
##  [63] plyr_1.8.8               sparseMatrixStats_1.6.0
##  [65] progress_1.2.2           zlibbioc_1.40.0
##  [67] purrr_0.3.5              RCurl_1.98-1.9
##  [69] prettyunits_1.1.1        deldir_1.0-6
##  [71] pbapply_1.6-0            cowplot_1.1.1
##  [73] zoo_1.8-11               ggrepel_0.9.2
##  [75] cluster_2.1.2            data.table_1.14.6
##  [77] scattermore_0.8          lmtest_0.9-40
##  [79] RANN_2.6.1               fitdistrplus_1.1-8
##  [81] hms_1.1.2                patchwork_1.1.2
##  [83] mime_0.12                evaluate_0.20
##  [85] xtable_1.8-4             XML_3.99-0.12
##  [87] gridExtra_2.3            compiler_4.1.1
##  [89] tibble_3.1.8             KernSmooth_2.23-20
##  [91] crayon_1.5.2             htmltools_0.5.4
##  [93] later_1.3.0              tidyr_1.2.1
##  [95] DBI_1.1.3                dbplyr_2.2.1
##  [97] MASS_7.3-54              rappdirs_0.3.3
##  [99] Matrix_1.5-3             cli_3.6.0
## [101] metapod_1.2.0            parallel_4.1.1
## [103] igraph_1.3.5             pkgconfig_2.0.3
## [105] sp_1.5-1                 plotly_4.10.1
## [107] spatstat.sparse_3.0-0    xml2_1.3.3
## [109] svglite_2.1.0            annotate_1.72.0
## [111] dqrng_0.3.0              webshot_0.5.4
## [113] XVector_0.34.0           rvest_1.0.3
## [115] stringr_1.4.1            digest_0.6.31
## [117] sctransform_0.3.5        RcppAnnoy_0.0.20
## [119] spatstat.data_3.0-0      Biostrings_2.62.0
## [121] leiden_0.4.3             uwot_0.1.14
## [123] edgeR_3.36.0             DelayedMatrixStats_1.16.0
## [125] curl_4.3.3               shiny_1.7.3
## [127] lifecycle_1.0.3          jsonlite_1.8.3
## [129] BiocNeighbors_1.12.0     viridisLite_0.4.1
## [131] fansi_1.0.3              pillar_1.8.1
## [133] lattice_0.20-44          KEGGREST_1.34.0
## [135] fastmap_1.1.0            httr_1.4.4
## [137] survival_3.2-13          glue_1.6.2
## [139] png_0.1-8                bluster_1.4.0
## [141] bit_4.0.5                stringi_1.7.8
## [143] blob_1.2.3               BiocSingular_1.10.0
## [145] memoise_2.0.1            renv_0.16.0
## [147] irlba_2.3.5.1            future.apply_1.10.0
```