final_project_sc_rna_analysis

Yuki Ito

Load in data

```
# Load in scRNA-seq data
HB17_bg_initial <- Read10X(data.dir = "/projectnb/bf528/students/yuki/final-project-yukiito0914/refs/HB17_background_filtered_feature_bc_matrix")
HB17_bg <- CreateSeuratObject(counts = HB17_bg_initial, project = "HB17_bg", min.cells = 3, min.features = 200)

HB17_PDX_initial <- Read10X(data.dir = "/projectnb/bf528/students/yuki/final-project-yukiito0914/refs/HB17_PDX_filtered_feature_bc_matrix")
HB17_PDX <- CreateSeuratObject(counts = HB17_PDX_initial, project = "HB17_PDX", min.cells = 3, min.features = 200)

HB17_tumor_initial <- Read10X(data.dir = "/projectnb/bf528/students/yuki/final-project-yukiito0914/refs/HB17_tumor_filtered_feature_bc_matrix")
HB17_tumor <- CreateSeuratObject(counts = HB17_tumor_initial, project = "HB17_tumor", min.cells = 3, min.features = 200)

HB30_PDX_initial <- Read10X(data.dir = "/projectnb/bf528/students/yuki/final-project-yukiito0914/refs/HB30_PDX_filtered_feature_bc_matrix")
HB30_PDX <- CreateSeuratObject(counts = HB30_PDX_initial, project = "HB30_PDX", min.cells = 3, min.features = 200)

HB30_tumor_initial <- Read10X(data.dir = "/projectnb/bf528/students/yuki/final-project-yukiito0914/refs/HB30_tumor_filtered_feature_bc_matrix")
HB30_tumor <- CreateSeuratObject(counts = HB30_tumor_initial, project = "HB30_tumor", min.cells = 3, min.features = 200)

HB53_bg_initial <- Read10X(data.dir = "/projectnb/bf528/students/yuki/final-project-yukiito0914/refs/HB53_background_filtered_feature_bc_matrix")
HB53_bg <- CreateSeuratObject(counts = HB53_bg_initial, project = "HB53_background", min.cells = 3, min.features = 200)

HB53_tumor_initial <- Read10X(data.dir = "/projectnb/bf528/students/yuki/final-project-yukiito0914/refs/HB53_tumor_filtered_feature_bc_matrix")
HB53_tumor <- CreateSeuratObject(counts = HB53_tumor_initial, project = "HB53_tumor", min.cells = 3, min.features = 200)
```

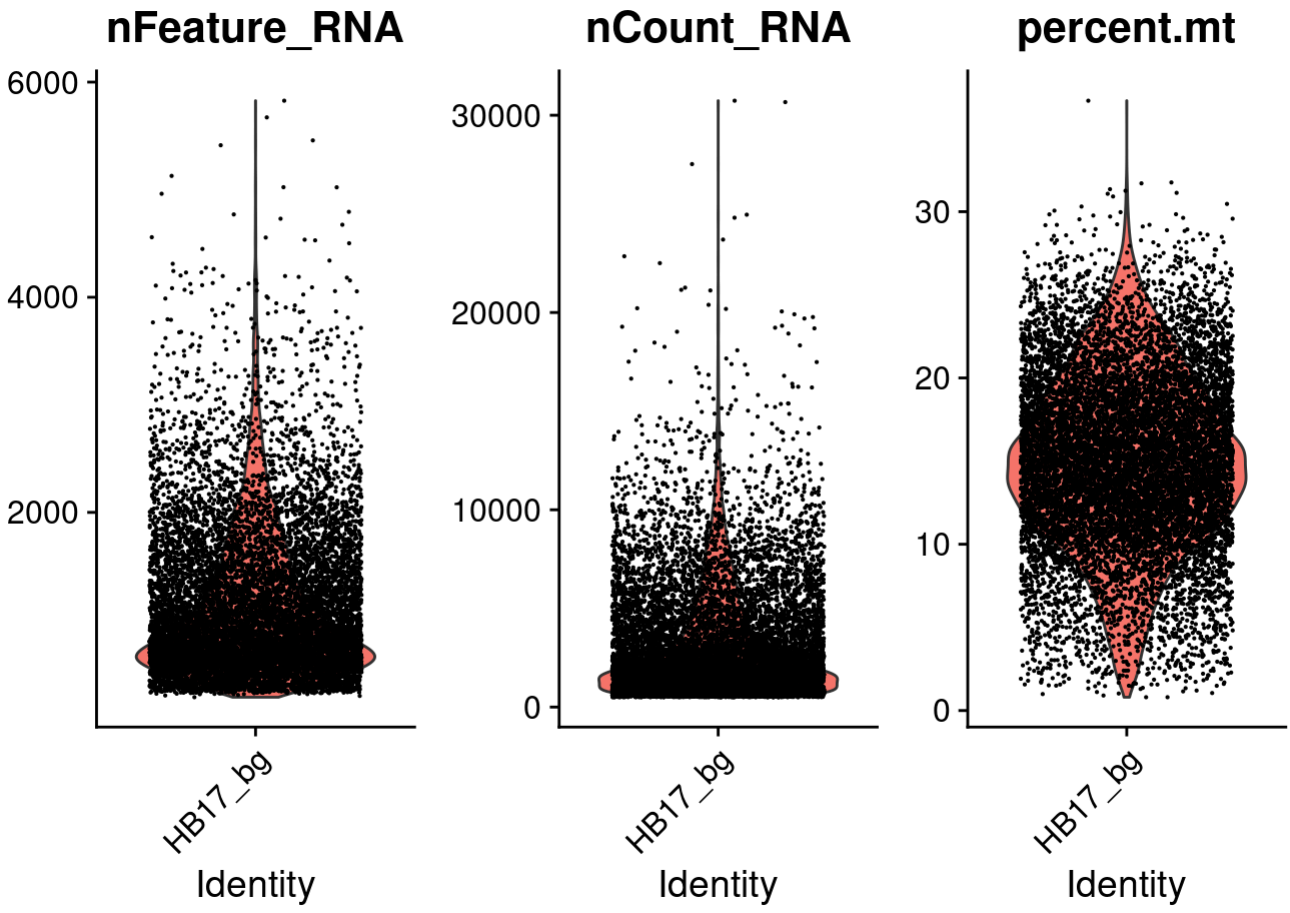
```
# Add metadata
HB17_bg$sample <- "HB17_background"
HB17_PDX$sample <- "HB17_PDX"
HB17_tumor$sample <- "HB17_tumor"
HB30_PDX$sample <- "HB30_PDX"
HB30_tumor$sample <- "HB30_tumor"
HB53_bg$sample <- "HB53_background"
HB53_tumor$sample <- "HB53_tumor"
```

Preprocessing

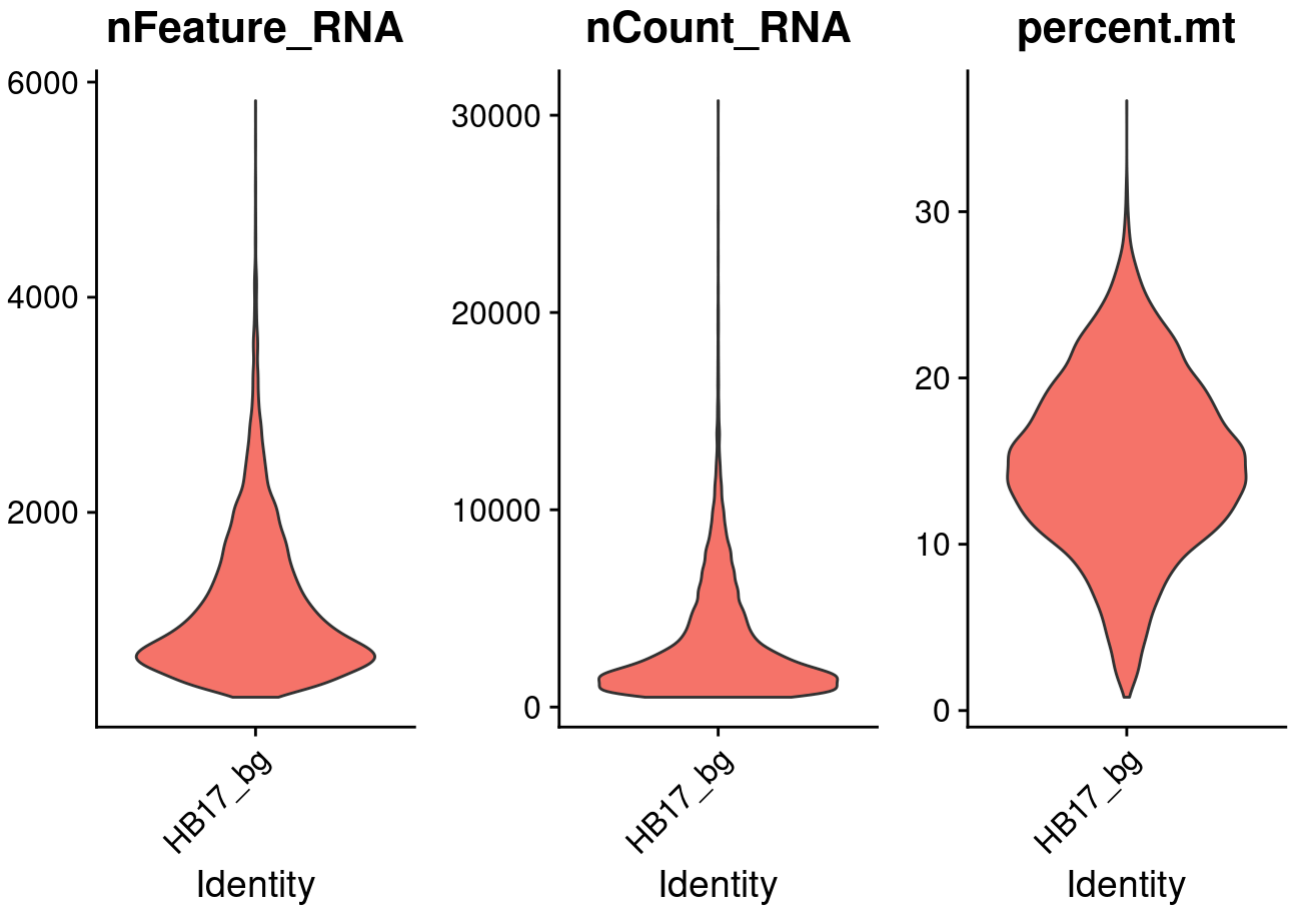
```
# Calculate mitochondrial genes rate
HB17_bg[["percent.mt"]] <- PercentageFeatureSet(HB17_bg, pattern = "^MT-")
HB17_PDX[["percent.mt"]] <- PercentageFeatureSet(HB17_PDX, pattern = "^MT-")
HB17_tumor[["percent.mt"]] <- PercentageFeatureSet(HB17_tumor, pattern = "^MT-")
HB30_PDX[["percent.mt"]] <- PercentageFeatureSet(HB30_PDX, pattern = "^MT-")
HB30_tumor[["percent.mt"]] <- PercentageFeatureSet(HB30_tumor, pattern = "^MT-")
HB53_bg[["percent.mt"]] <- PercentageFeatureSet(HB53_bg, pattern = "^MT-")
HB53_tumor[["percent.mt"]] <- PercentageFeatureSet(HB53_tumor, pattern = "^MT-")
```

Visualize QC metrics as a violin plot

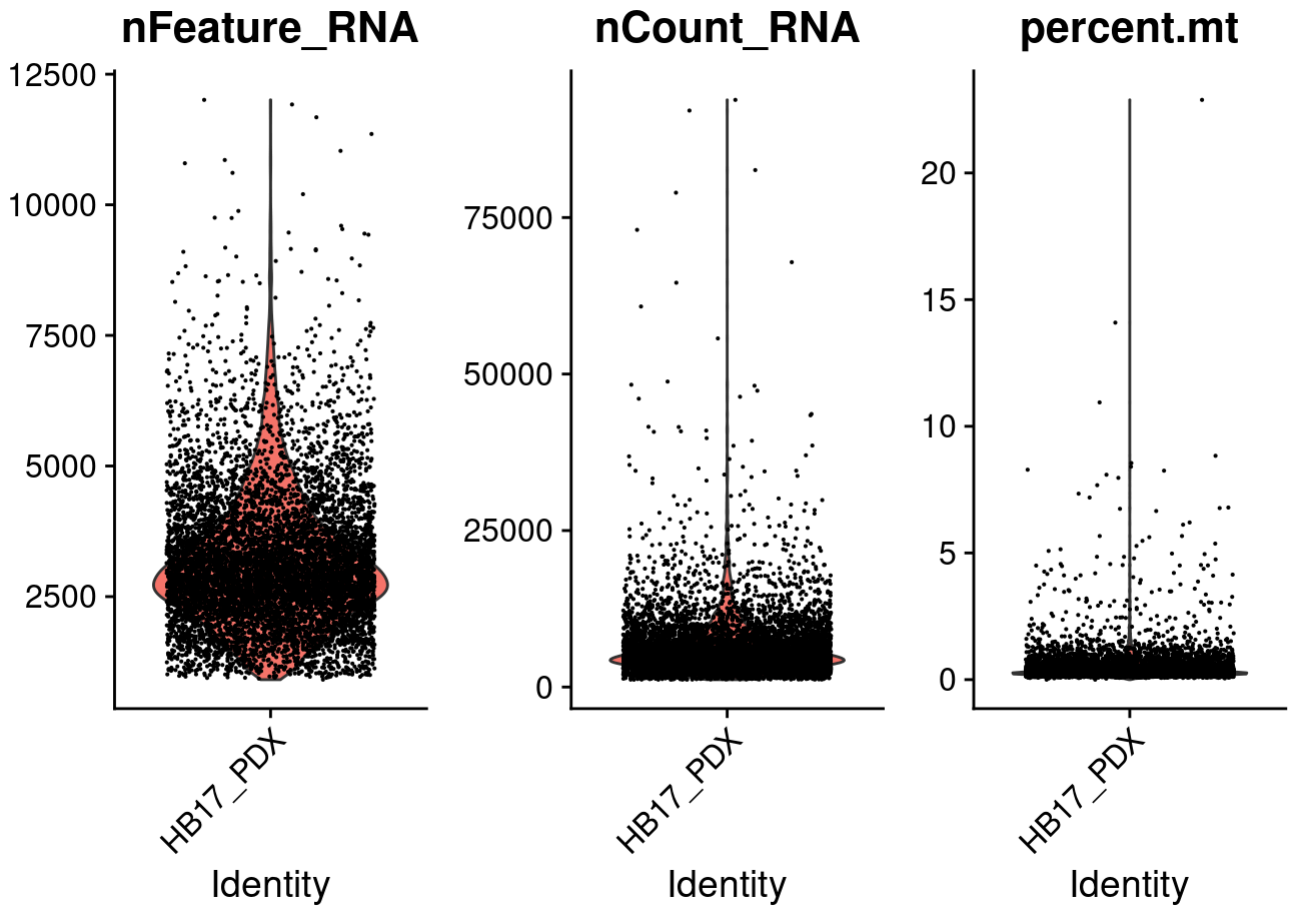
```
VlnPlot(HB17_bg, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



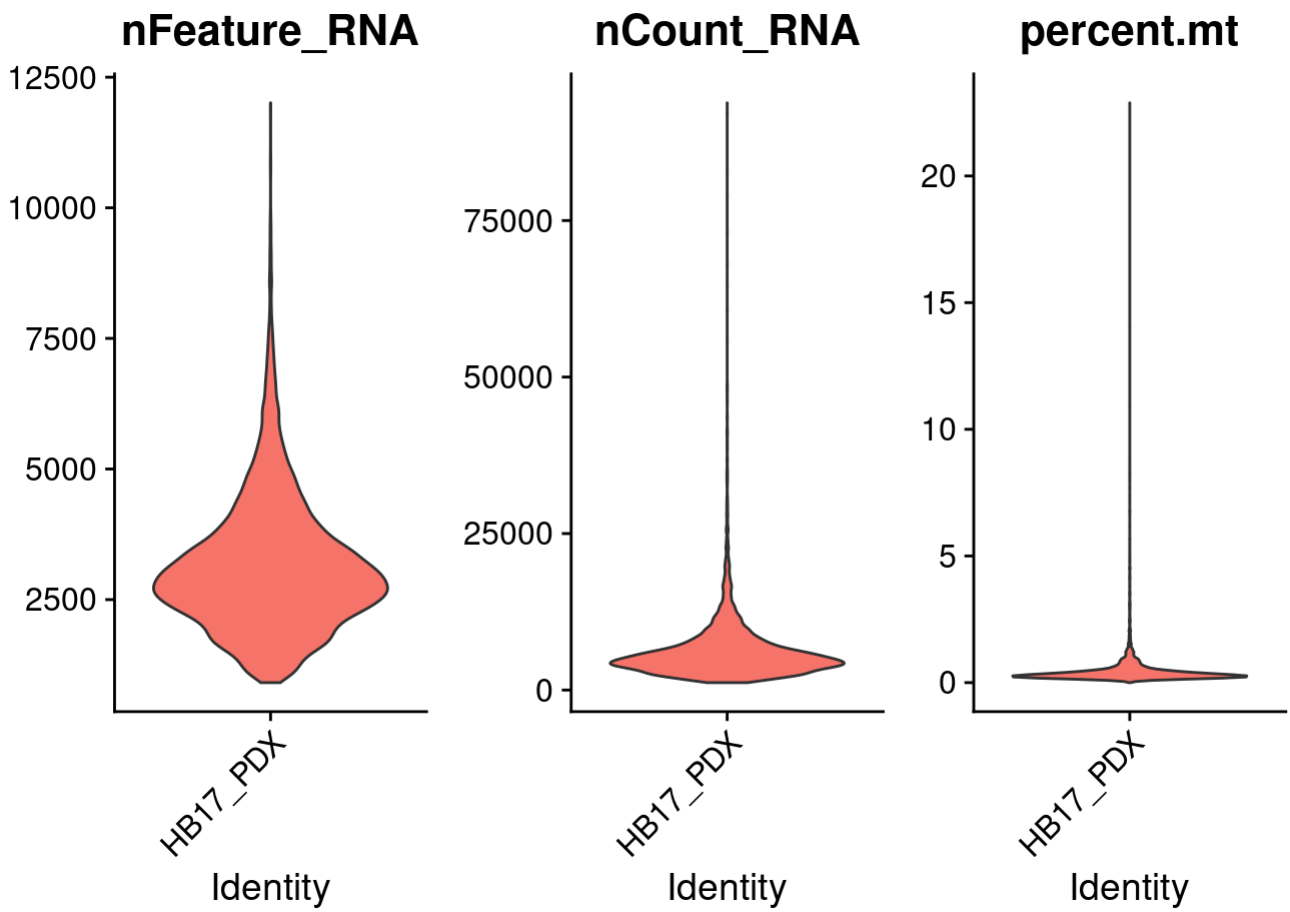

```
VlnPlot(HB17_bg, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size = 0) # Remove dots
```



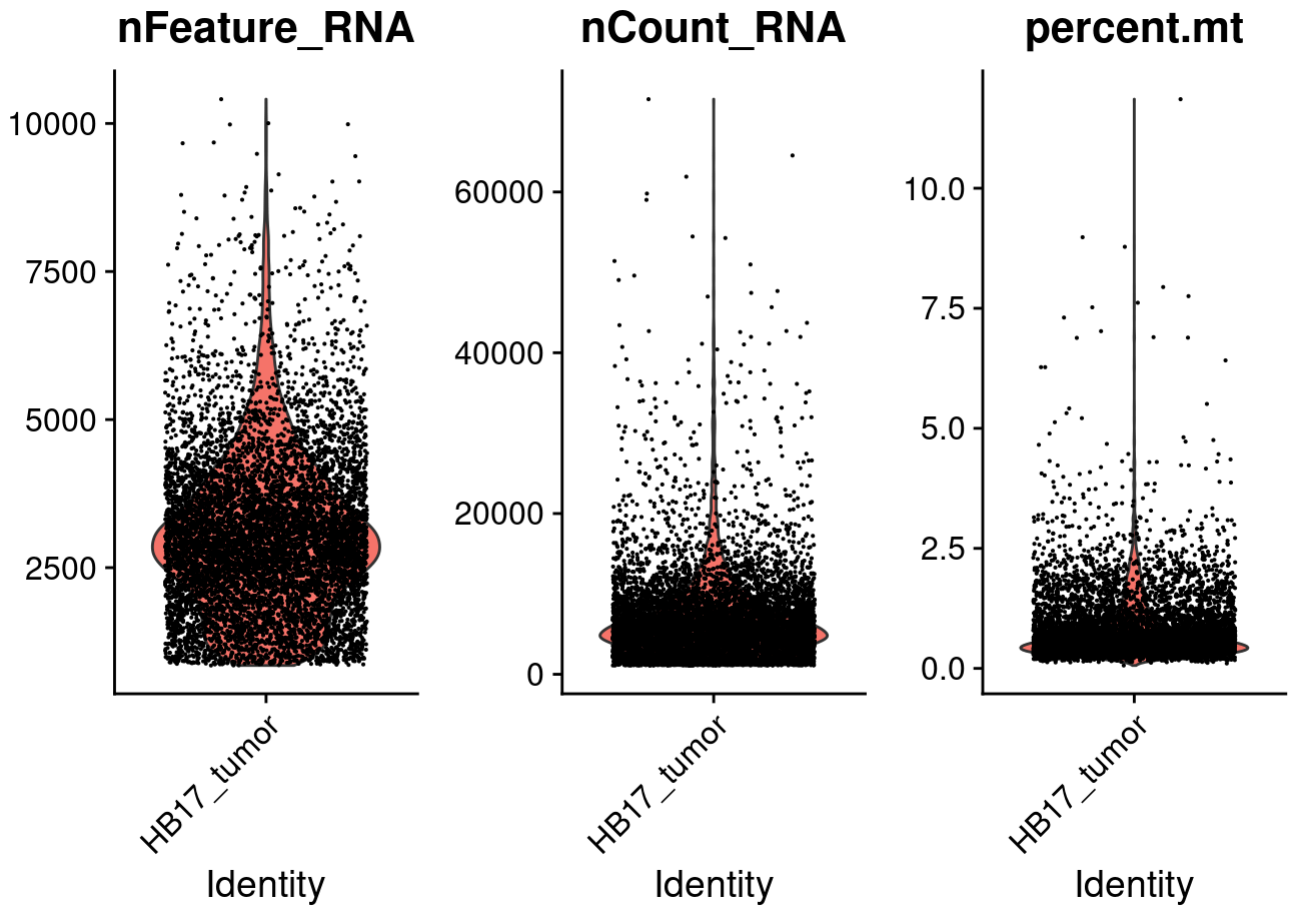

```
VlnPlot(HB17_PDX, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



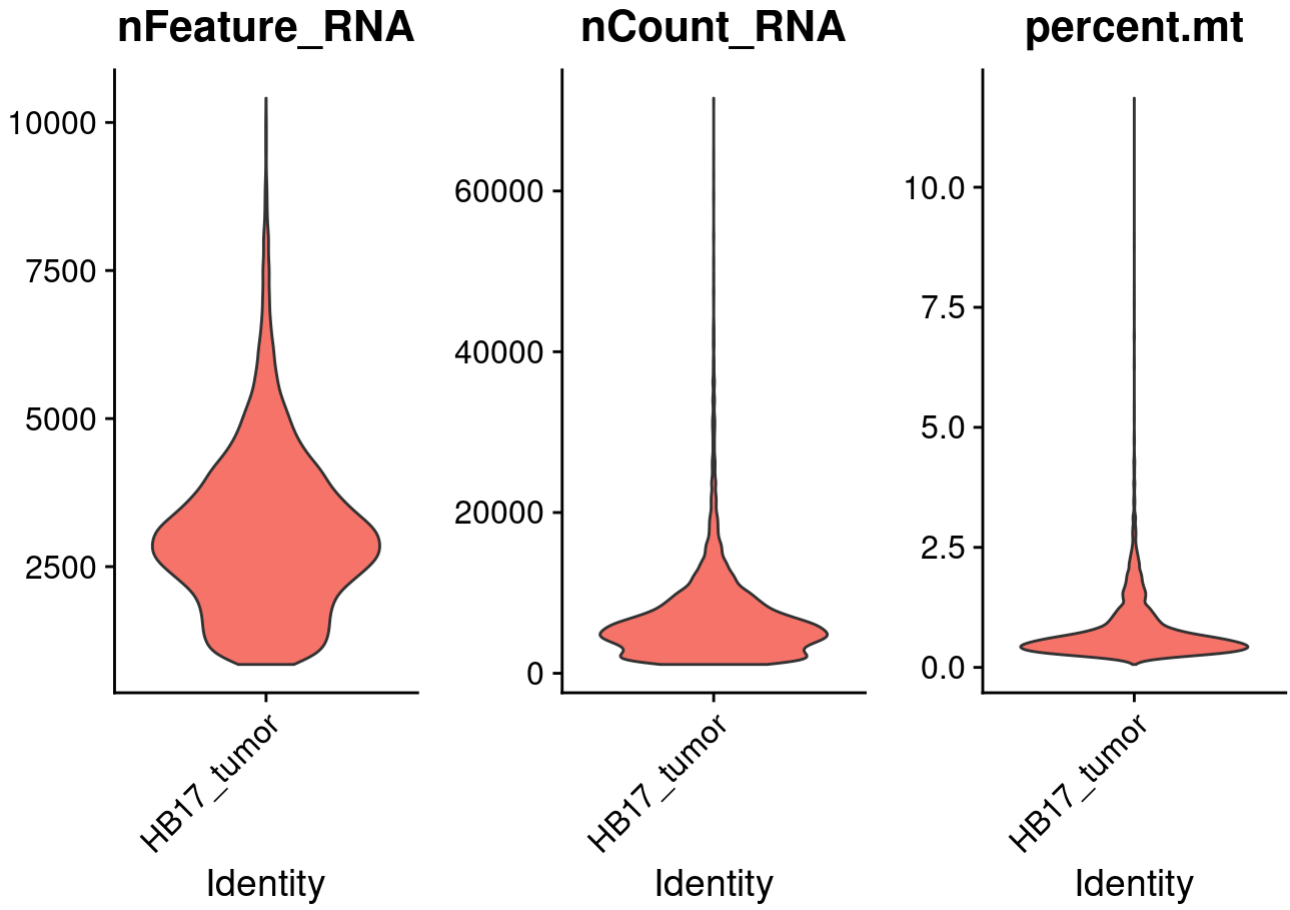
```
VlnPlot(HB17_PDX, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size = 0) # Remove dots
```



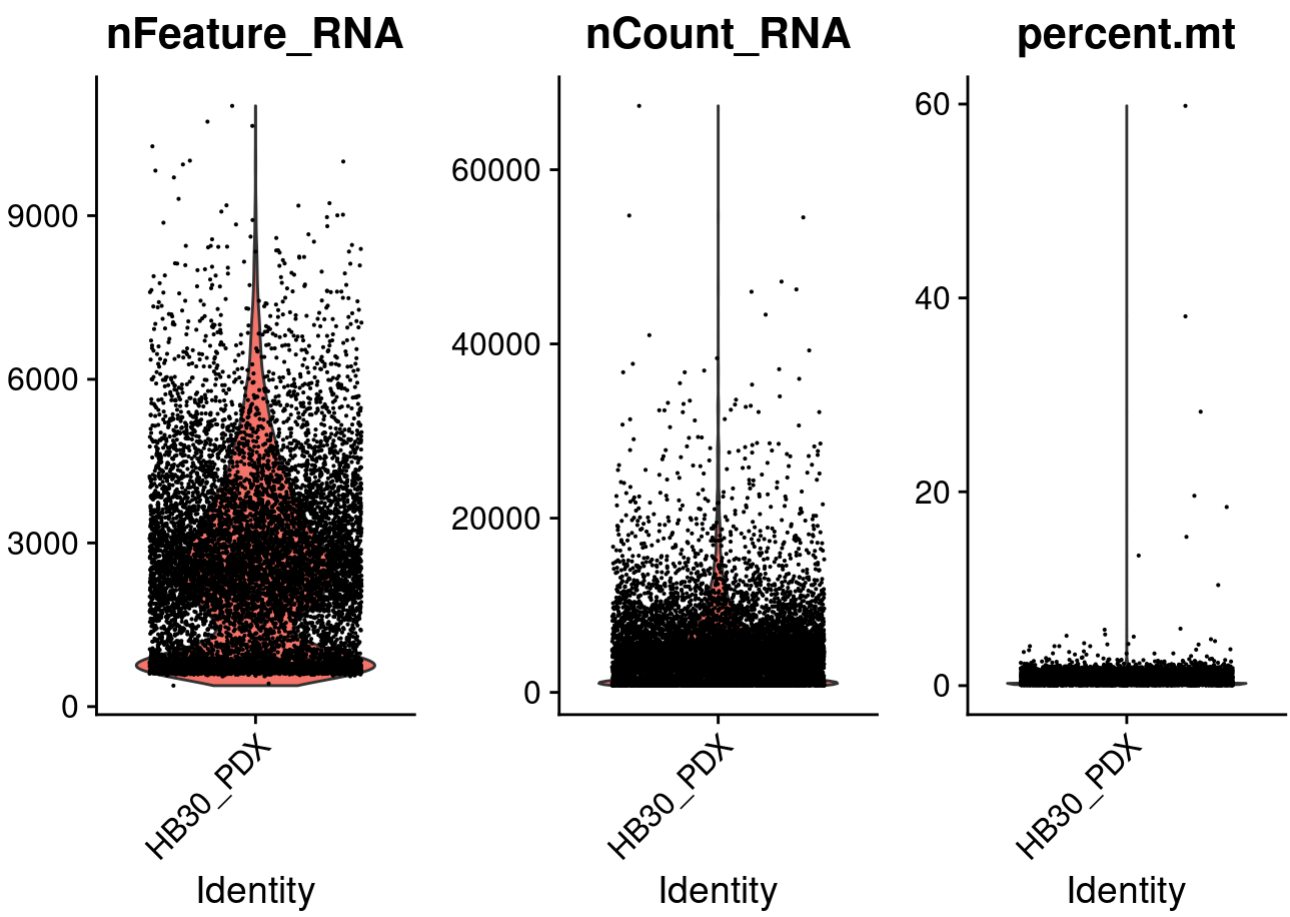

```
VlnPlot(HB17_tumor, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



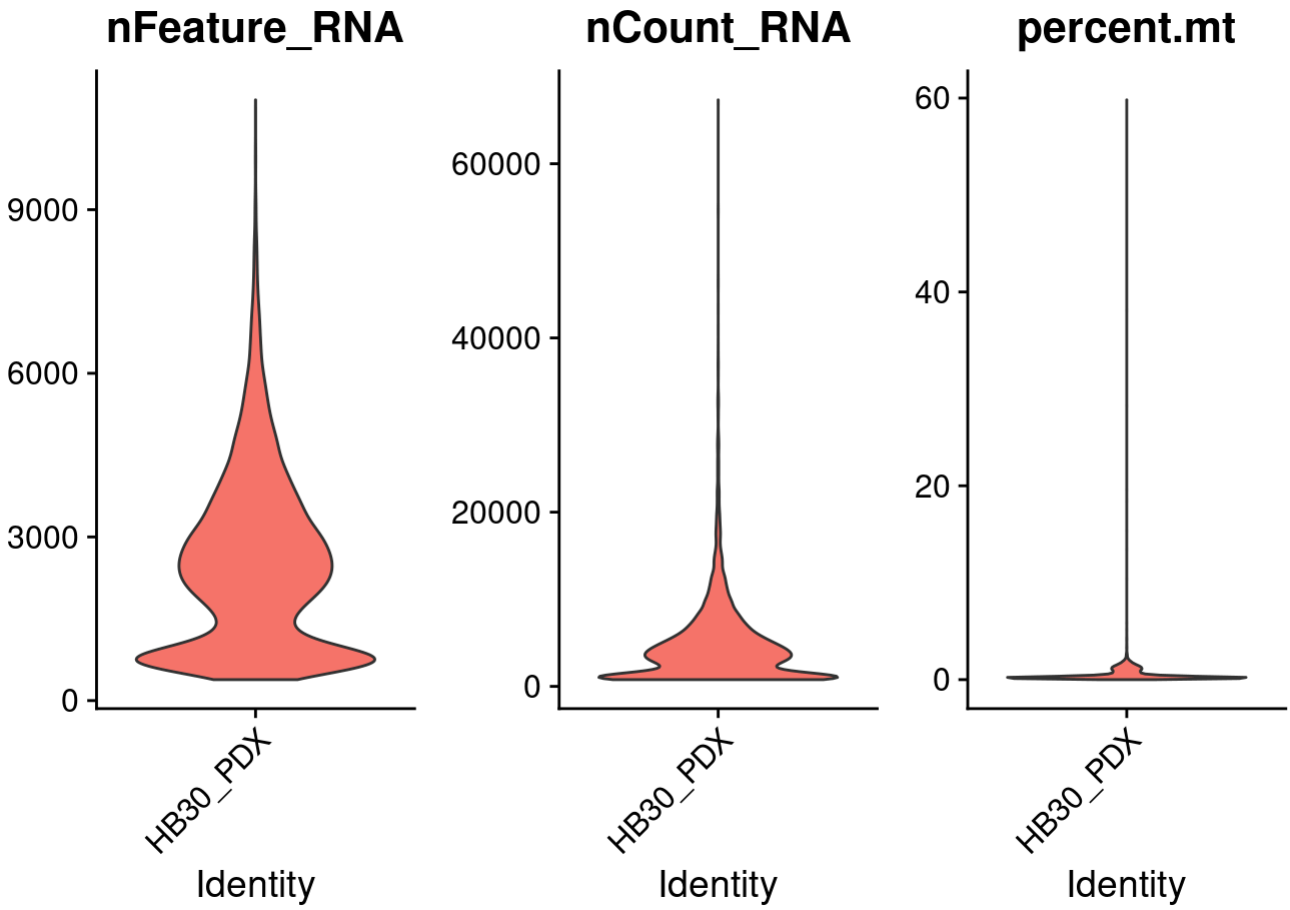

```
VlnPlot(HB17_tumor, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size = 0) # Remove dots
```



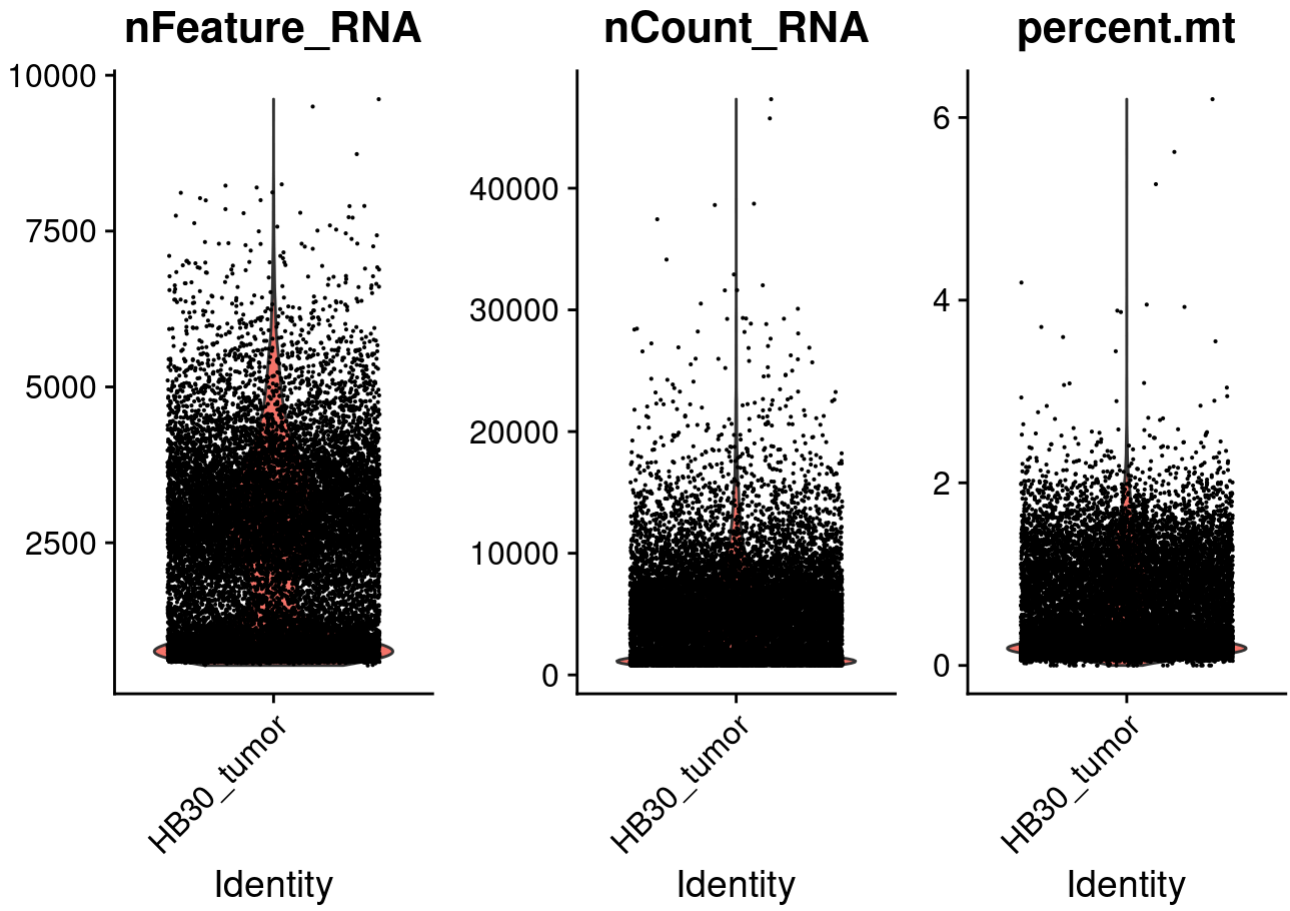
```
VlnPlot(HB30_PDX, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



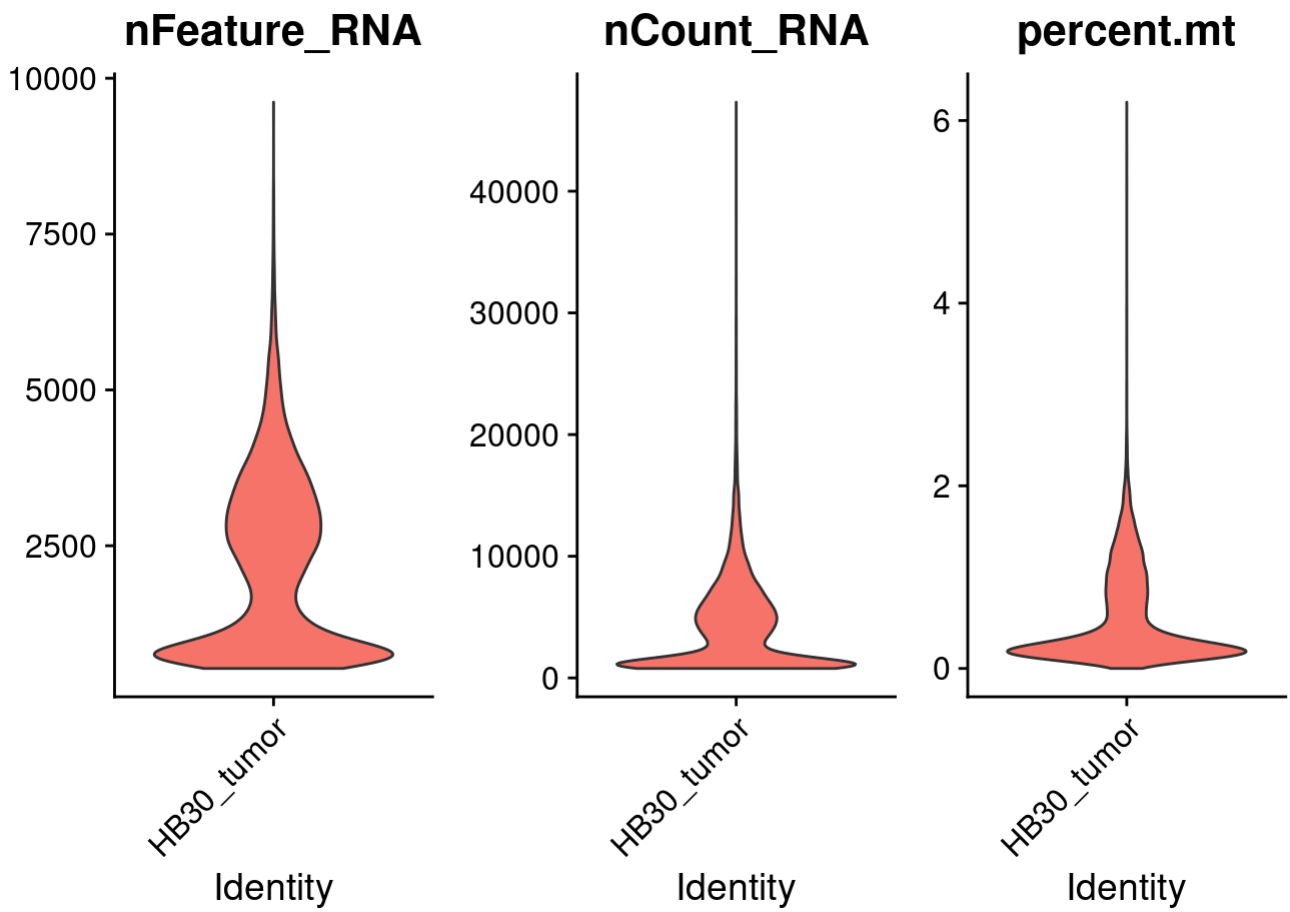

```
VlnPlot(HB30_PDX, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size = 0) # Remove dots
```



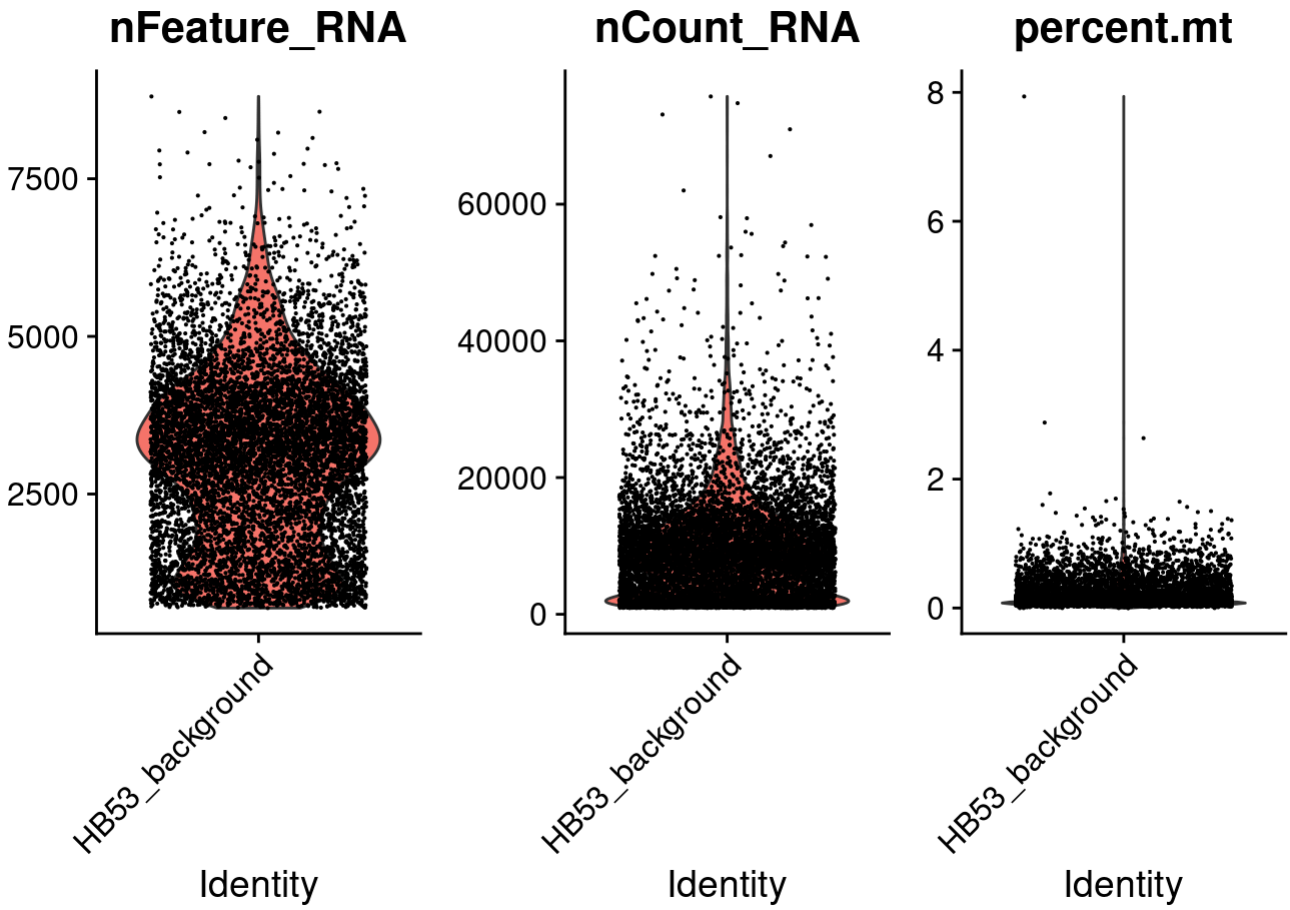

```
VlnPlot(HB30_tumor, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



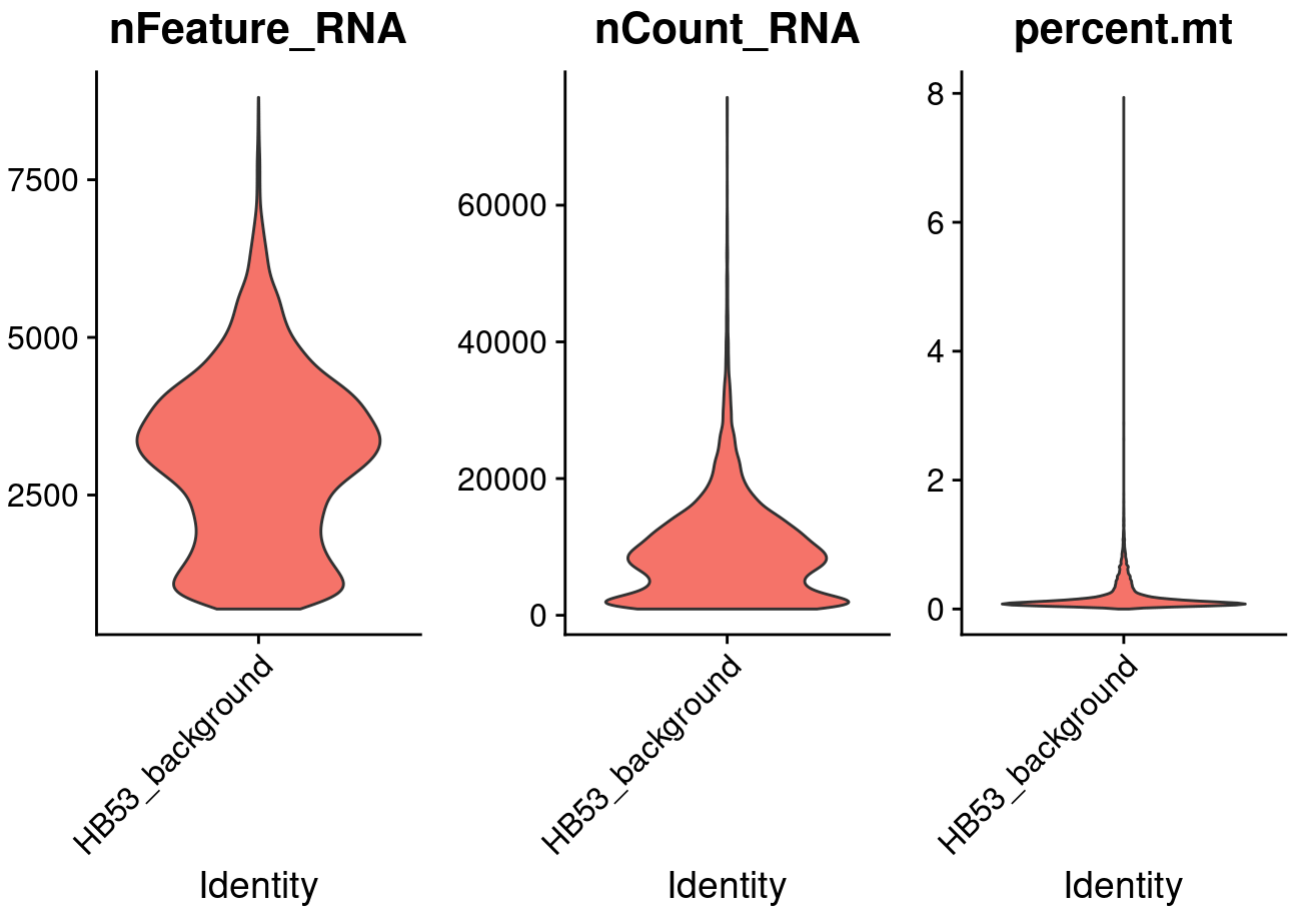
```
VlnPlot(HB30_tumor, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size = 0) # Remove dots
```



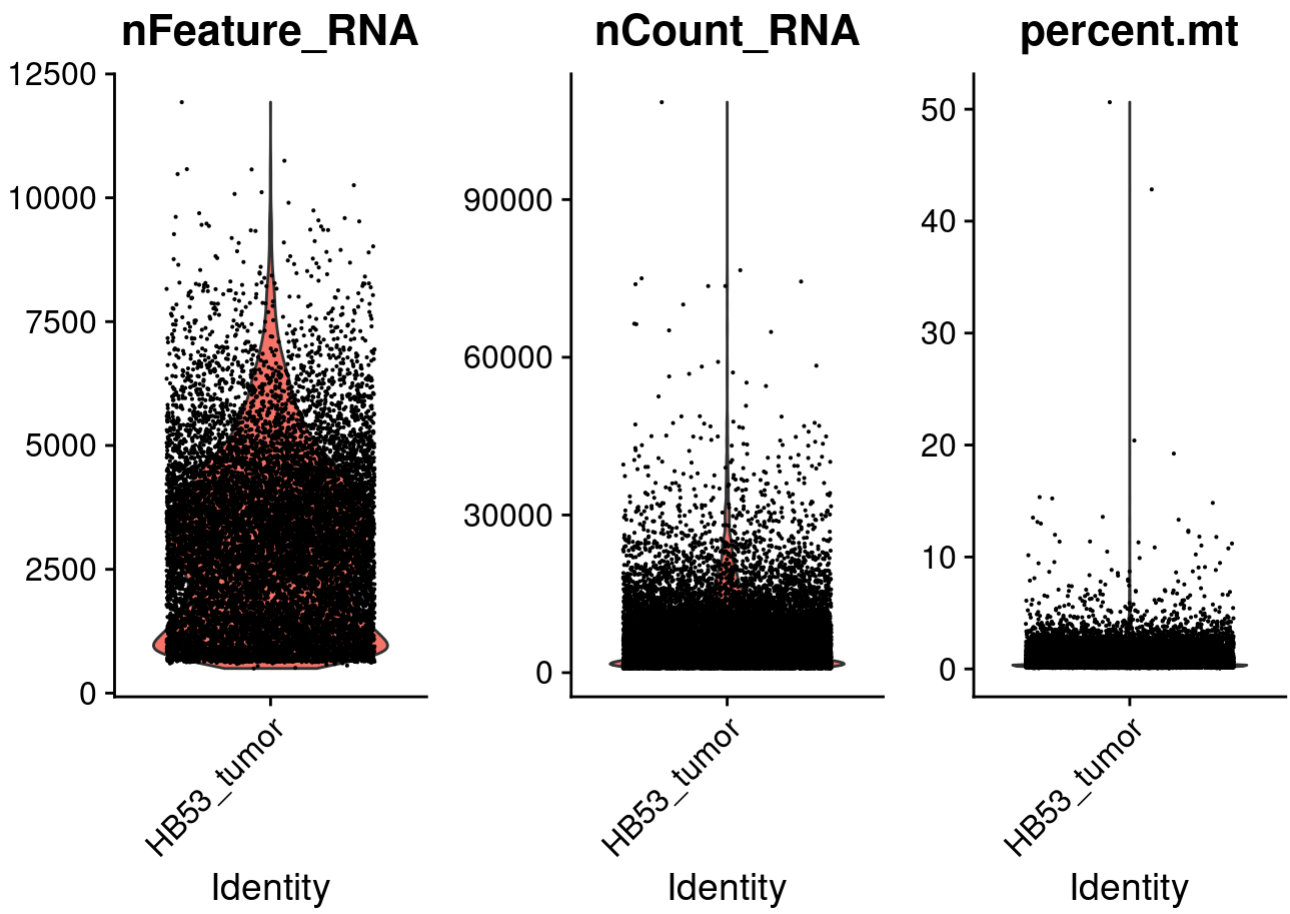

```
VlnPlot(HB53_bg, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



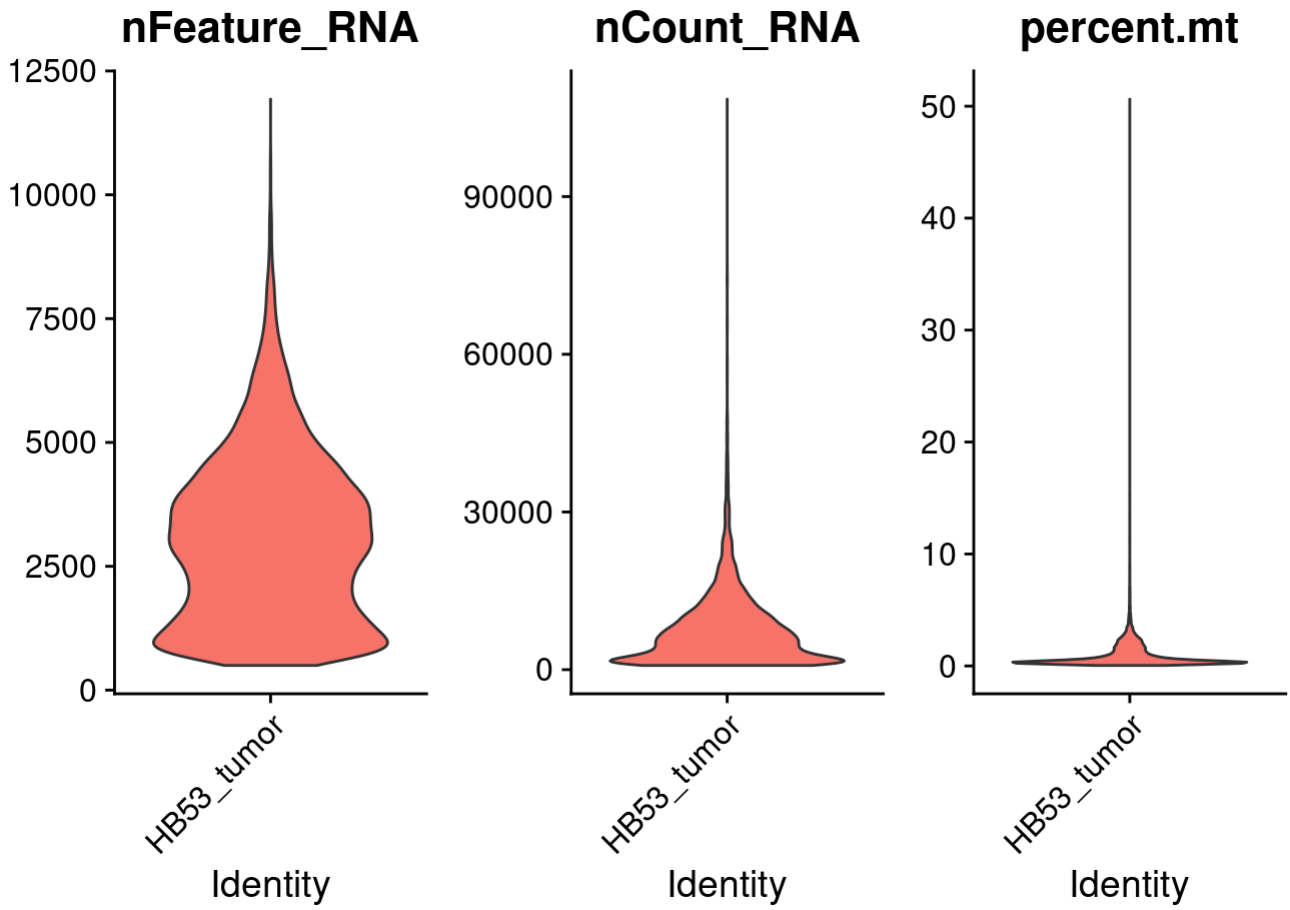

```
VlnPlot(HB53_bg, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size = 0) # Remove dots
```



```
VlnPlot(HB53_tumor, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```

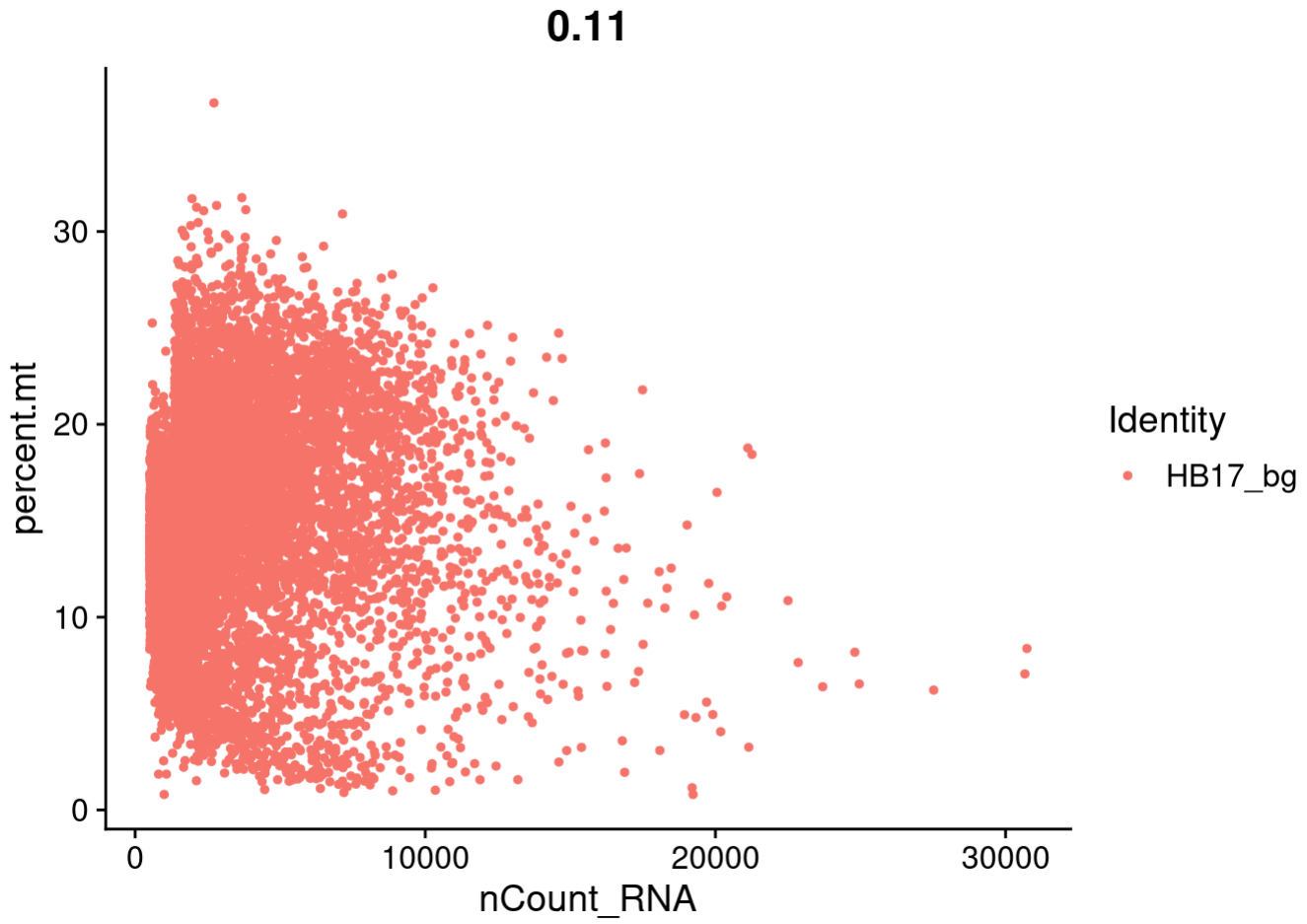



```
VlnPlot(HB53_tumor, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3, pt.size = 0) # Remove dots
```

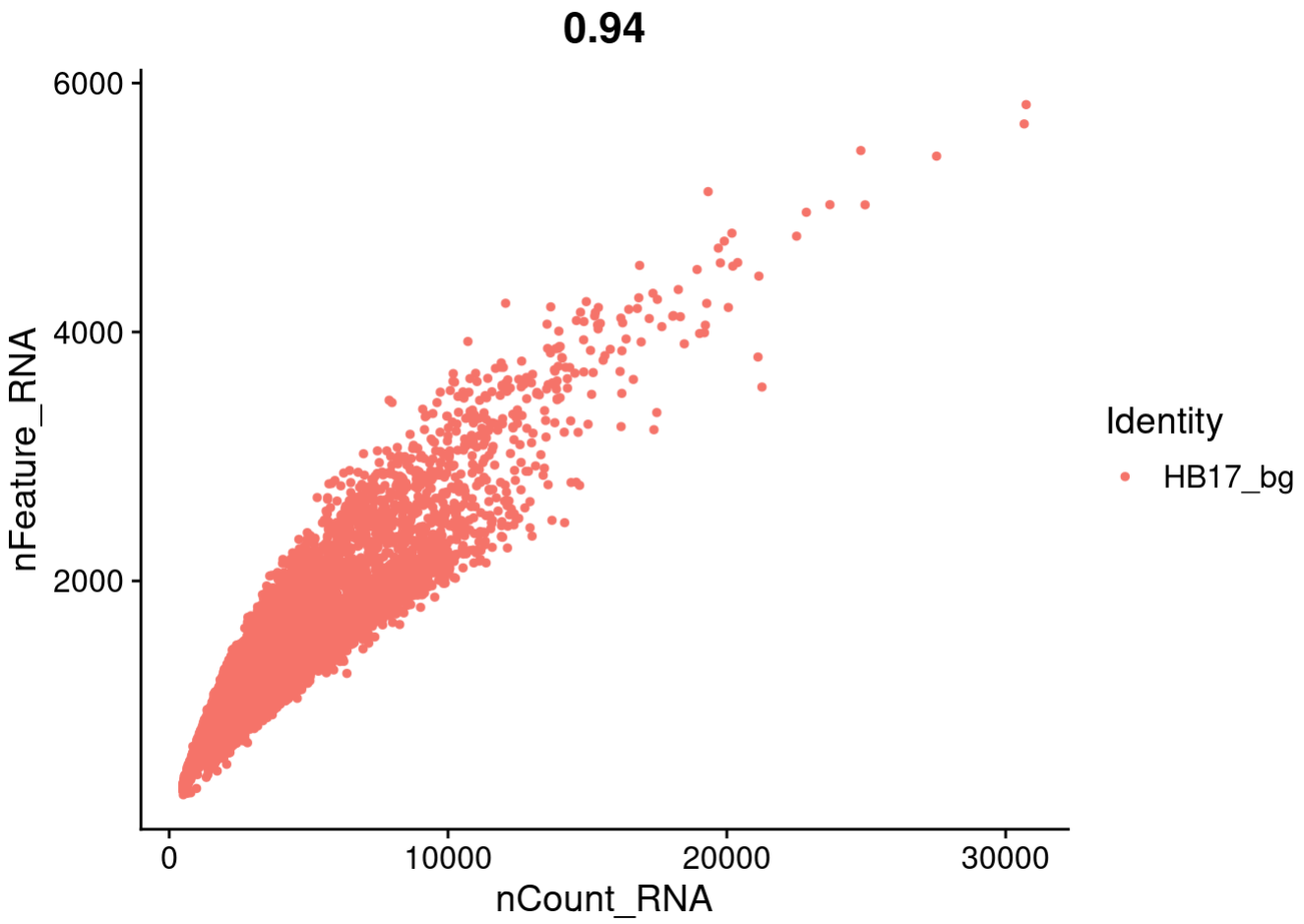


Visualize feature-feature relationships

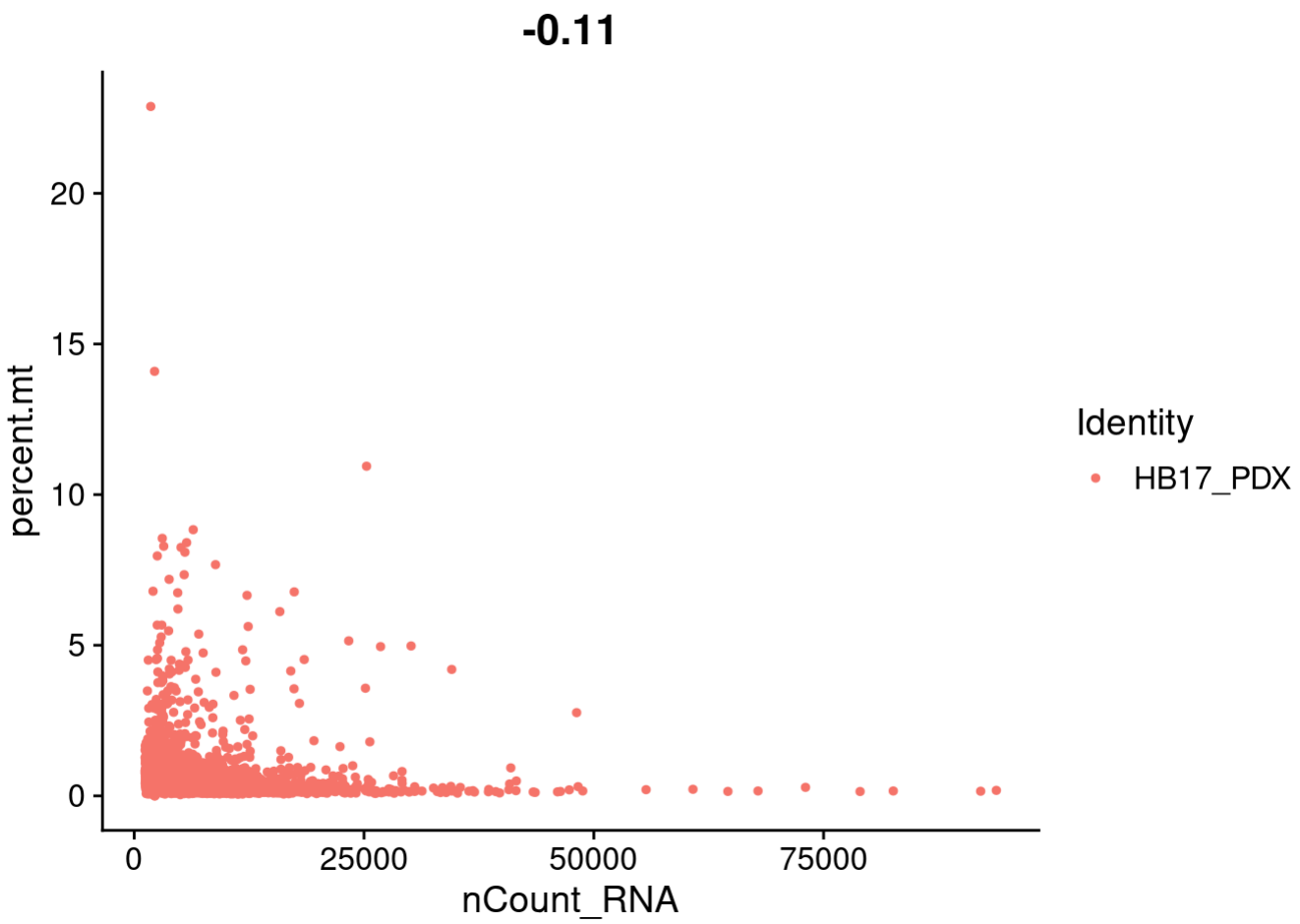
```
FeatureScatter(HB17_bg, feature1 = "nCount_RNA", feature2 = "percent.mt")
```



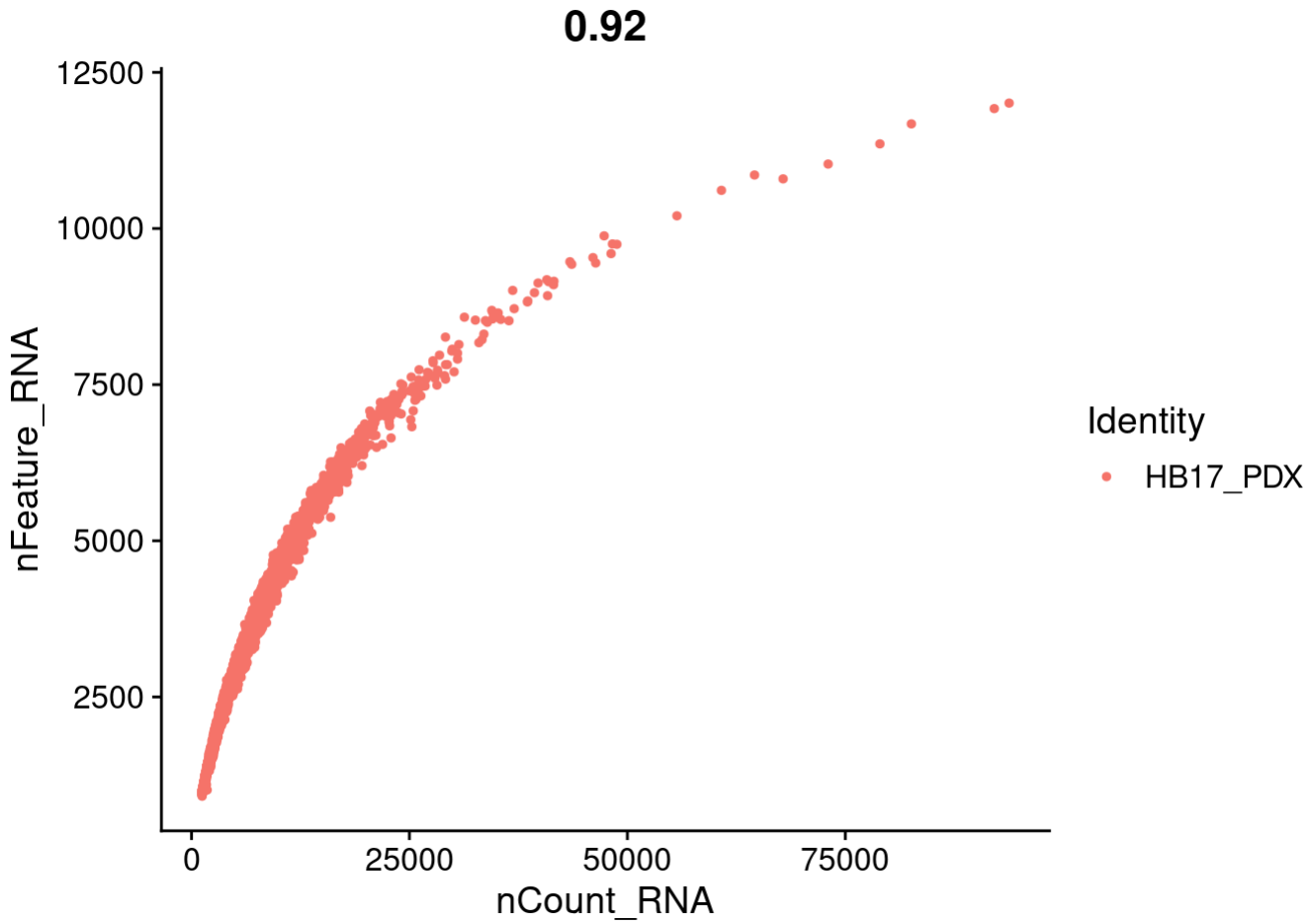

```
FeatureScatter(HB17_bg, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



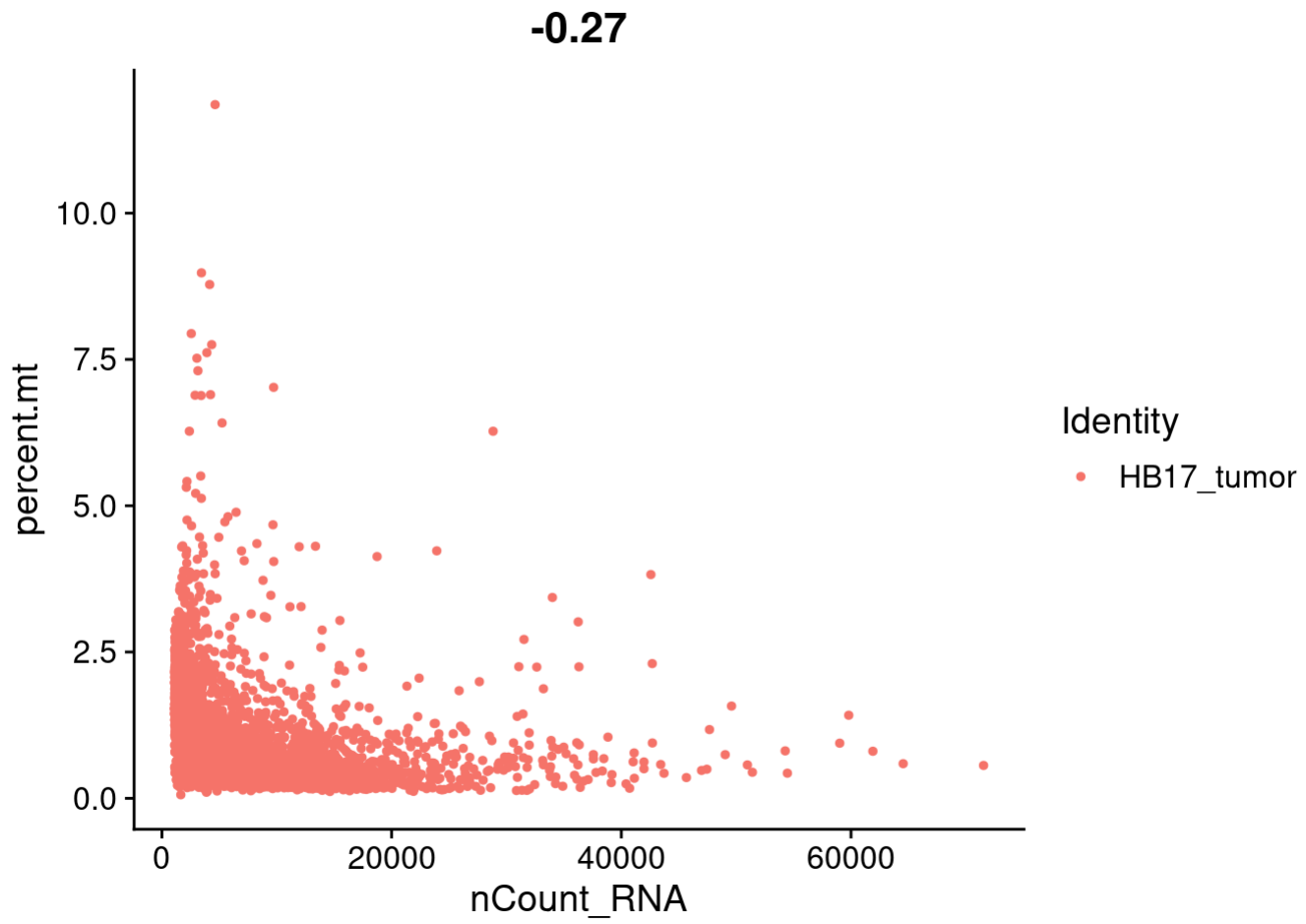
```
FeatureScatter(HB17_PDX, feature1 = "nCount_RNA", feature2 = "percent.mt")
```



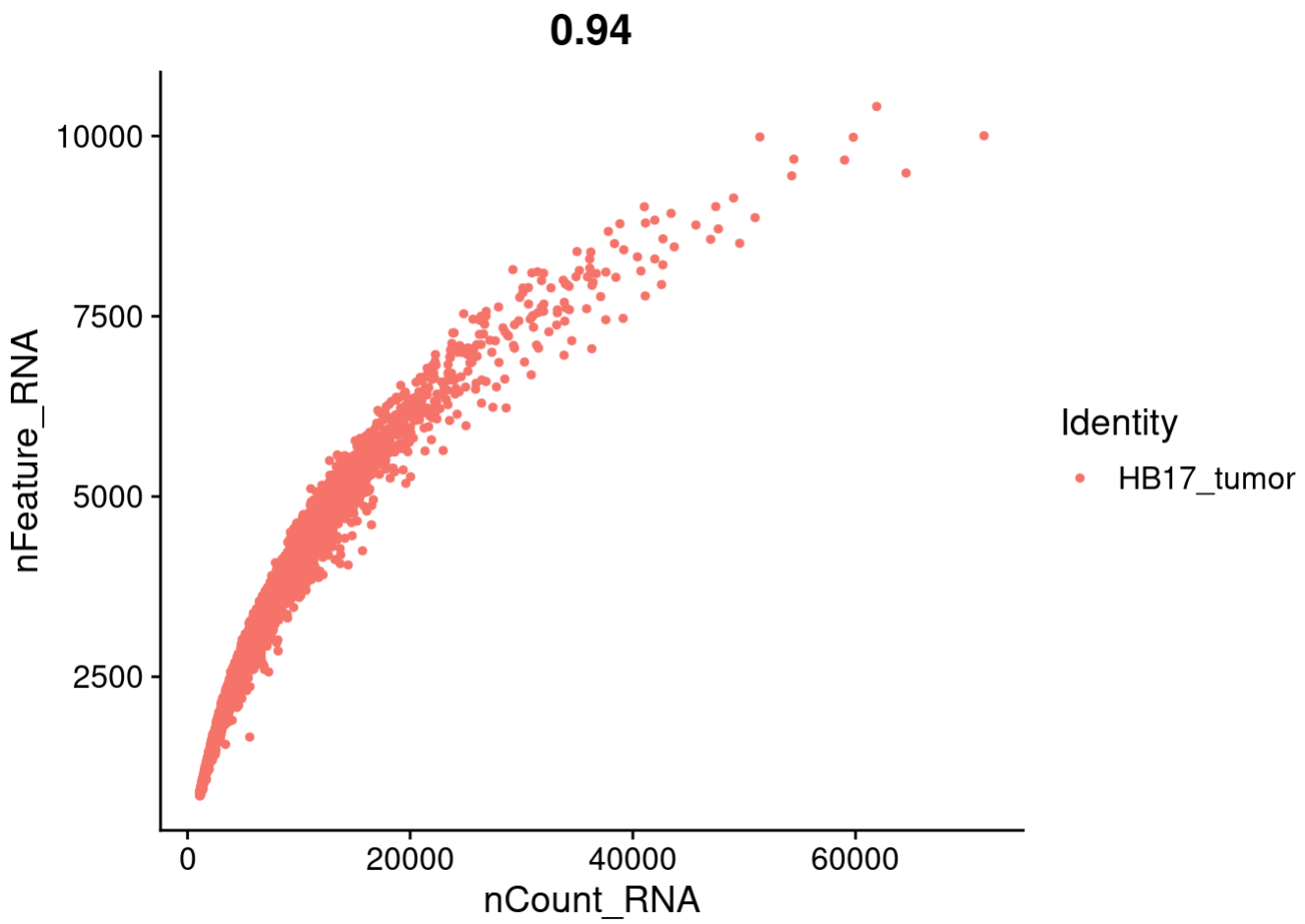

```
FeatureScatter(HB17_PDX, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



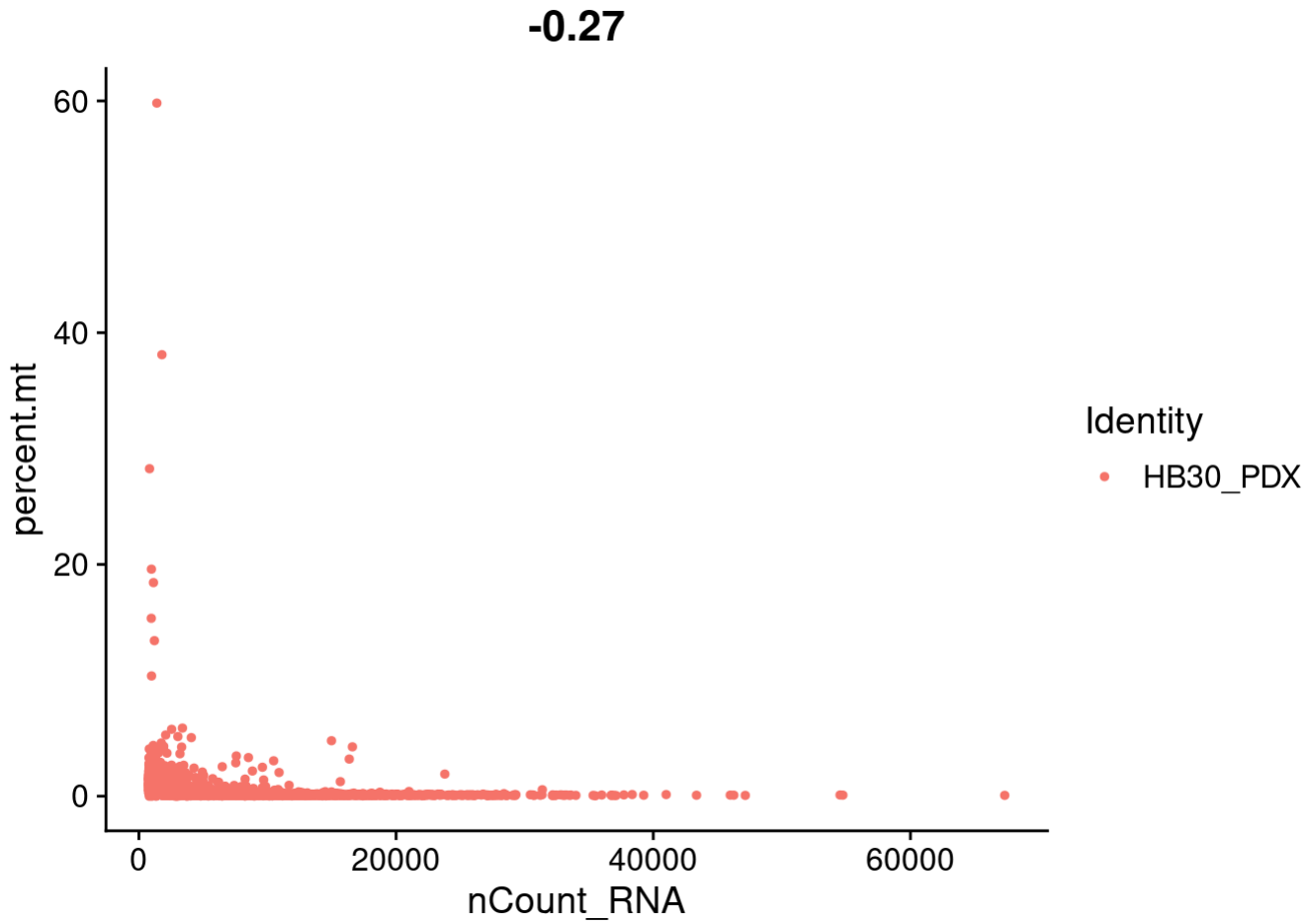

```
FeatureScatter(HB17_tumor, feature1 = "nCount_RNA", feature2 = "percent.mt")
```



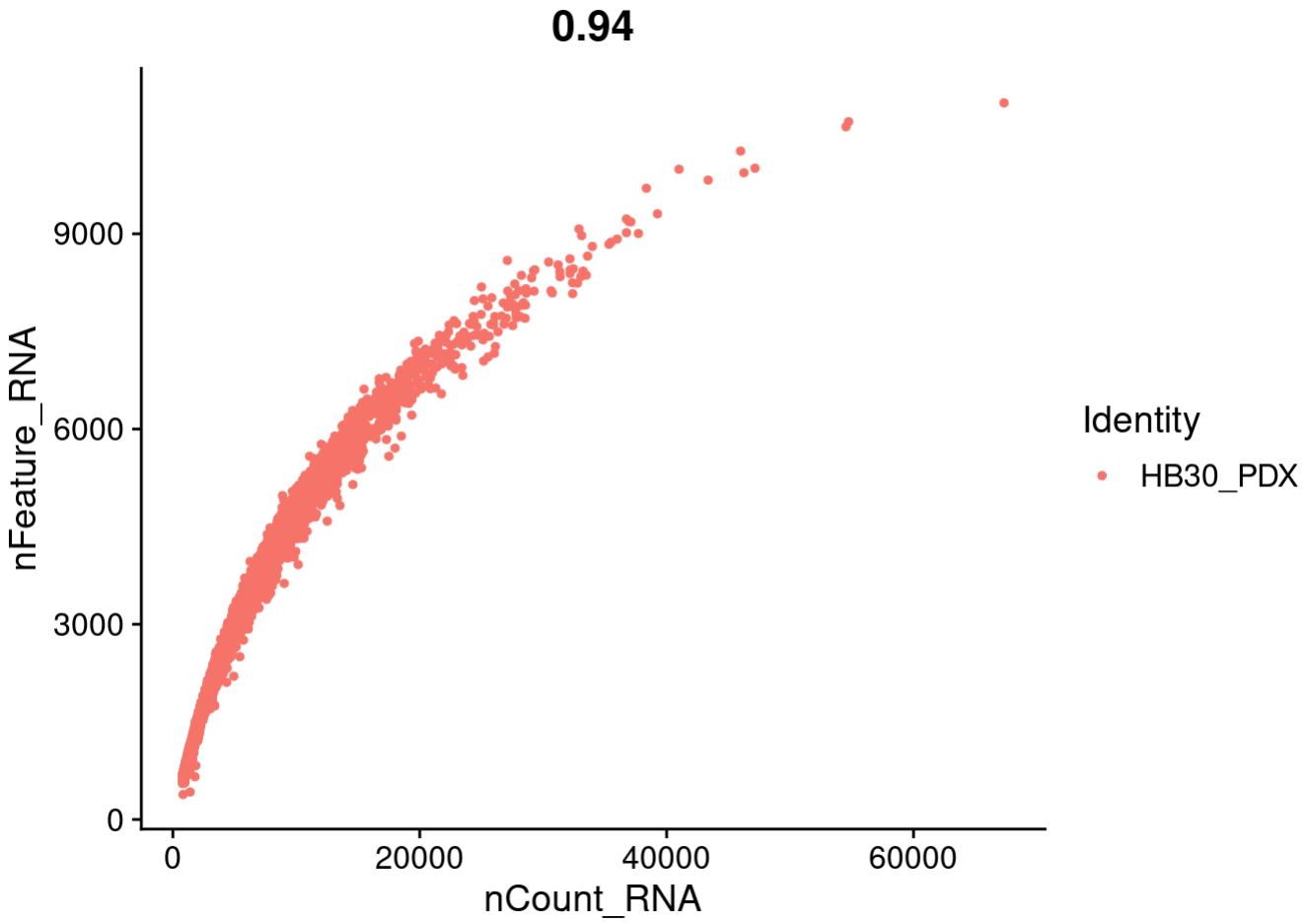
```
FeatureScatter(HB17_tumor, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



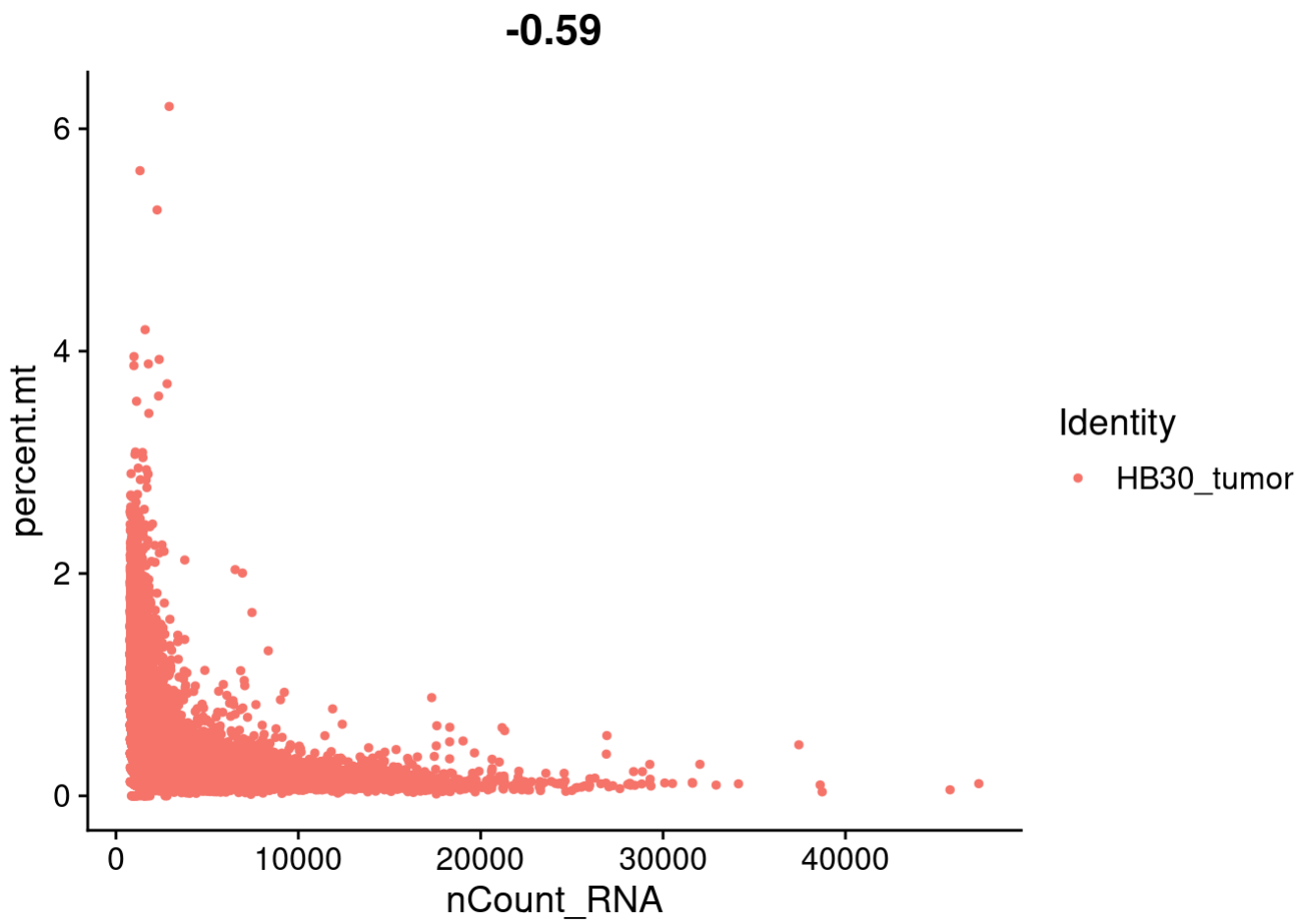

```
FeatureScatter(HB30_PDX, feature1 = "nCount_RNA", feature2 = "percent.mt")
```



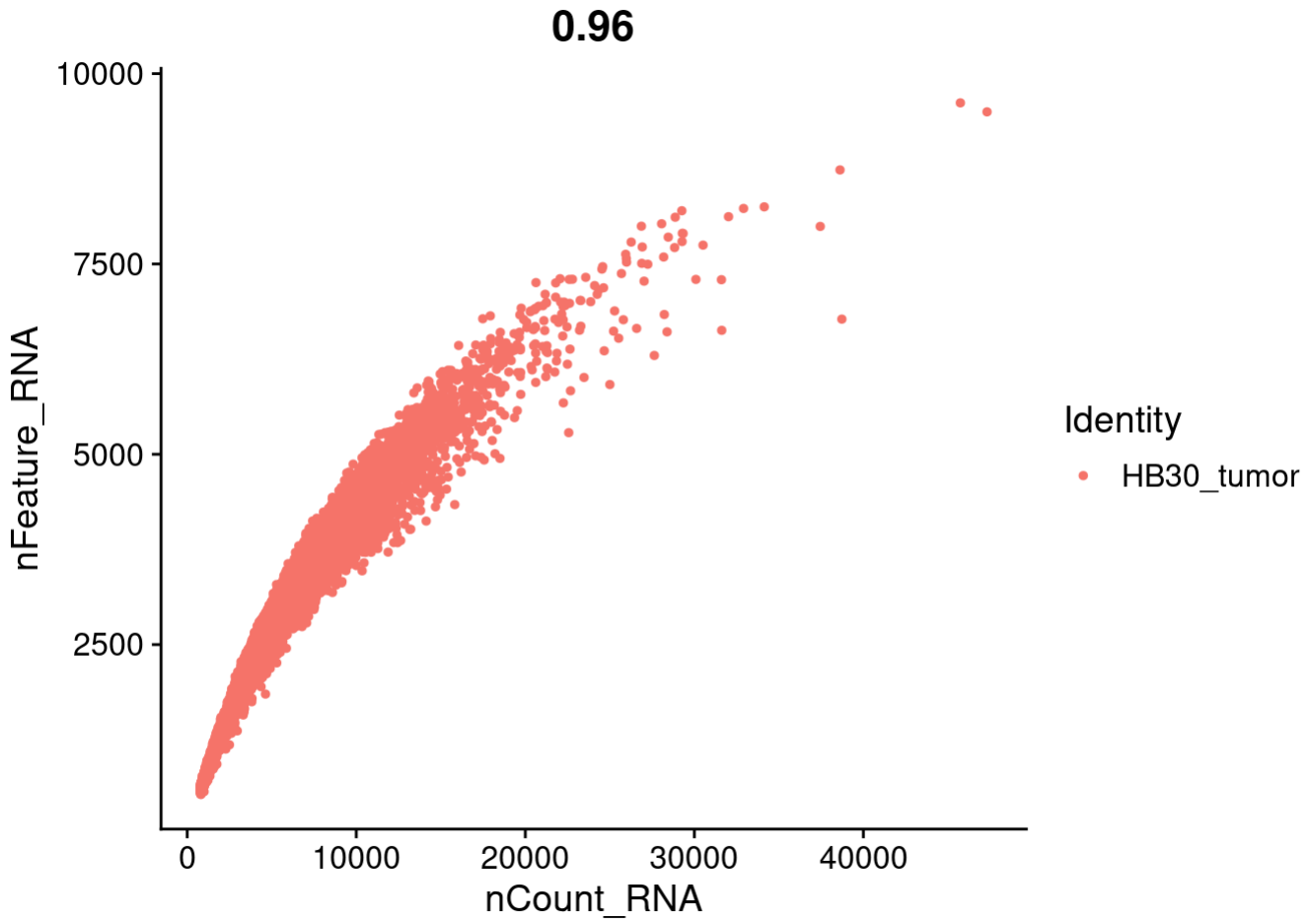

```
FeatureScatter(HB30_PDX, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



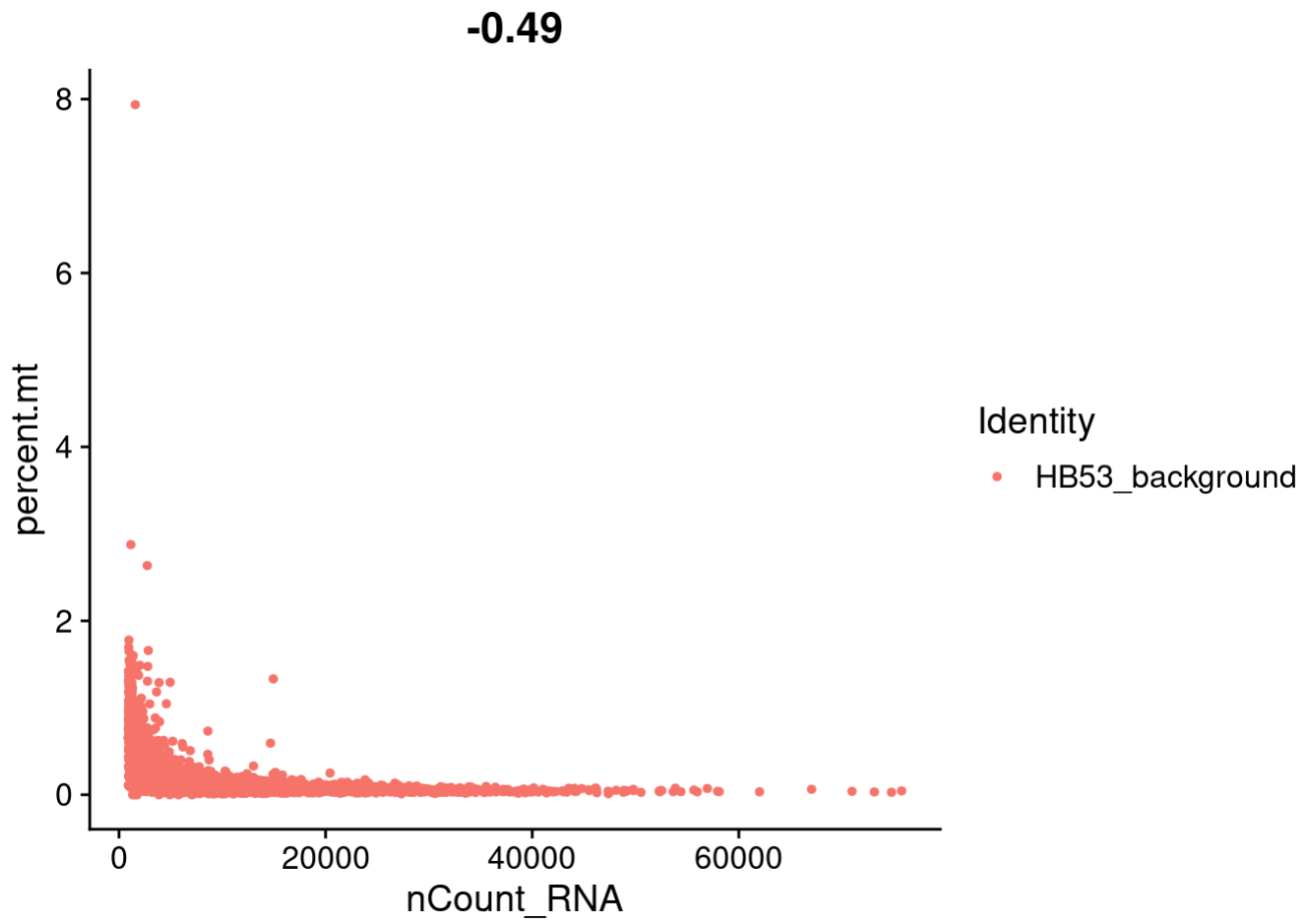
```
FeatureScatter(HB30_tumor, feature1 = "nCount_RNA", feature2 = "percent.mt")
```



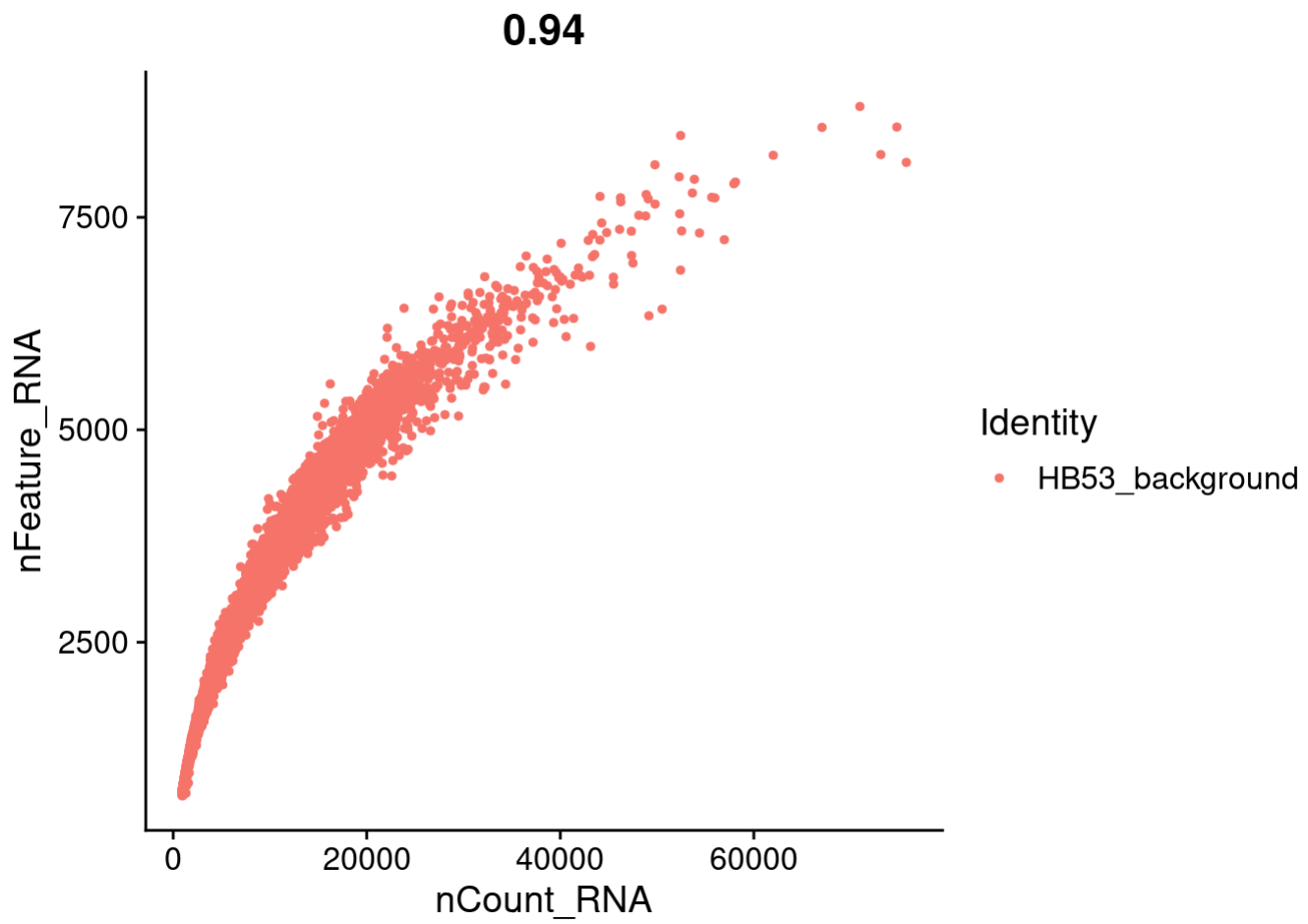

```
FeatureScatter(HB30_tumor, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



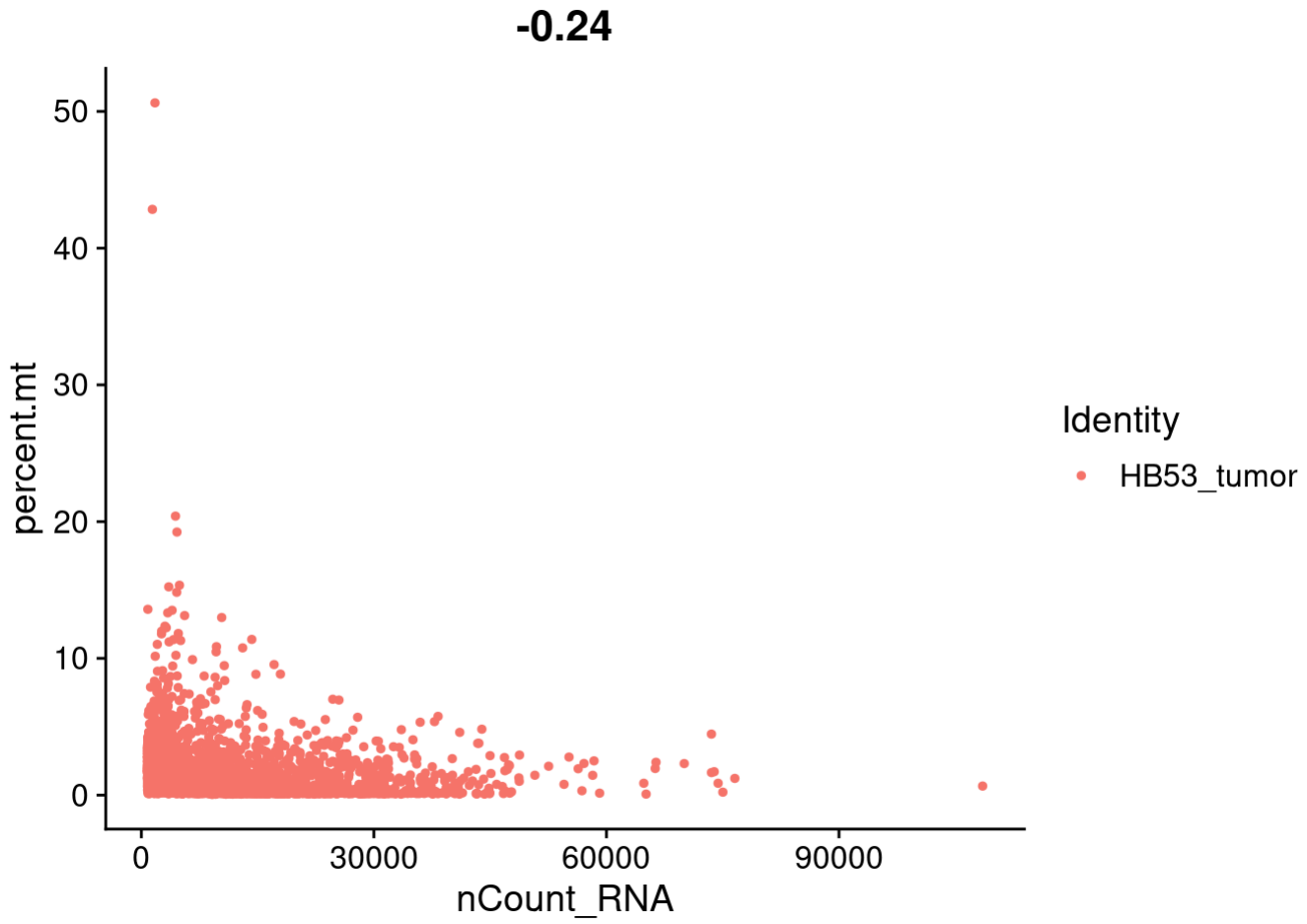

```
FeatureScatter(HB53_bg, feature1 = "nCount_RNA", feature2 = "percent.mt")
```



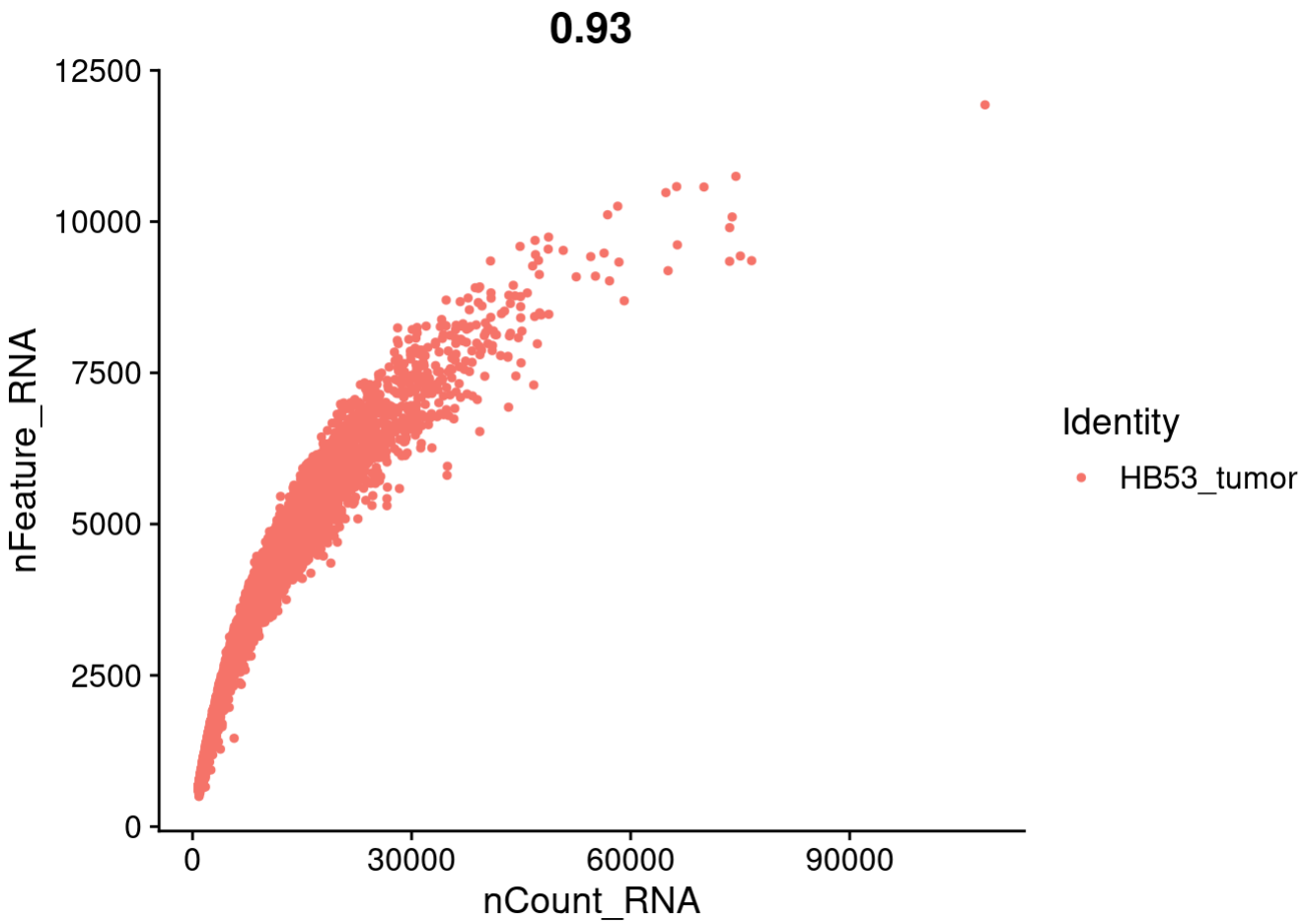
```
FeatureScatter(HB53_bg, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```




```
FeatureScatter(HB53_tumor, feature1 = "nCount_RNA", feature2 = "percent.mt")
```




```
FeatureScatter(HB53_tumor, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



Filtering and Doublet detection

HB17_background

```
# Filtering
HB17_bg <- subset(HB17_bg, subset = nFeature_RNA > 500 & nFeature_RNA < 5000 & nCount_RNA < 25000 & pe
rcent.mt < 20)
# Doublet detection
HB17_bg_sce <- as.SingleCellExperiment(HB17_bg)
HB17_bg_sce <- scDblFinder(HB17_bg_sce)
HB17_bg$scDblFinder_class <- HB17_bg_sce$scDblFinder.class
HB17_bg_filtered <- subset(HB17_bg, subset = scDblFinder_class == "singlet")
table(HB17_bg$scDblFinder_class)
```

```
##
## singlet doublet
##    6867    1137
```

HB17_PDX

```
# Filtering
HB17_PDX <- subset(HB17_PDX, subset = nFeature_RNA >= 750 & nFeature_RNA < 7500 & nCount_RNA < 40000 &
percent.mt < 5)
# Doublet detection
HB17_PDX_sce <- as.SingleCellExperiment(HB17_PDX)
HB17_PDX_sce <- scDblFinder(HB17_PDX_sce)
HB17_PDX$scDblFinder_class <- HB17_PDX_sce$scDblFinder.class
HB17_PDX_filtered <- subset(HB17_PDX, subset = scDblFinder_class == "singlet")
table(HB17_PDX$scDblFinder_class)
```

```
##
## singlet doublet
##    7375     558
```

HB17_tumor

```
# Filtering
HB17_tumor <- subset(HB17_tumor, subset = nFeature_RNA > 750 & nFeature_RNA < 7500 & nCount_RNA < 4000
0 & percent.mt < 5)
# Doublet detection
HB17_tumor_sce <- as.SingleCellExperiment(HB17_tumor)
HB17_tumor_sce <- scDblFinder(HB17_tumor_sce)
HB17_tumor$scDblFinder_class <- HB17_tumor_sce$scDblFinder.class
HB17_tumor_filtered <- subset(HB17_tumor, subset = scDblFinder_class == "singlet")
table(HB17_tumor$scDblFinder_class)
```

```
##
## singlet doublet
##    7079     817
```

HB30_PDX

```
# Filtering
HB30_PDX <- subset(HB30_PDX, subset = nFeature_RNA >= 750 & nFeature_RNA < 9000 & nCount_RNA < 40000 &
percent.mt < 10)
# Doublet detection
HB30_PDX_sce <- as.SingleCellExperiment(HB30_PDX)
HB30_PDX_sce <- scDblFinder(HB30_PDX_sce)
HB30_PDX$scDblFinder_class <- HB30_PDX_sce$scDblFinder.class
HB30_PDX_filtered <- subset(HB30_PDX, subset = scDblFinder_class == "singlet")
table(HB30_PDX$scDblFinder_class)
```

```
##
## singlet doublet
##    8337     416
```

HB30_tumor

```
# Filtering
HB30_tumor <- subset(HB30_tumor, subset = nFeature_RNA > 1000 & nFeature_RNA < 7500 & nCount_RNA < 400
00 & percent.mt < 5)
# Doublet detection
HB30_tumor_sce <- as.SingleCellExperiment(HB30_tumor)
HB30_tumor_sce <- scDblFinder(HB30_tumor_sce)
HB30_tumor$scDblFinder_class <- HB30_tumor_sce$scDblFinder.class
HB30_tumor_filtered <- subset(HB30_tumor, subset = scDblFinder_class == "singlet")
table(HB30_tumor$scDblFinder_class)
```

```
##
## singlet doublet
##   11667    1223
```

HB53_background

```
# Filtering
HB53_bg <- subset(HB53_bg, subset = nFeature_RNA > 1000 & nFeature_RNA < 7500 & nCount_RNA < 40000 & p
ercent.mt < 5)
# Doublet detection
HB53_bg_sce <- as.SingleCellExperiment(HB53_bg)
HB53_bg_sce <- scDblFinder(HB53_bg_sce)
HB53_bg$scDblFinder_class <- HB53_bg_sce$scDblFinder.class
HB53_bg_filtered <- subset(HB53_bg, subset = scDblFinder_class == "singlet")
table(HB53_bg$scDblFinder_class)
```

```
##
## singlet doublet
##    6954     929
```

HB53_tumor

```
# Filtering
HB53_tumor <- subset(HB53_tumor, subset = nFeature_RNA > 1000 & nFeature_RNA < 10000 & nCount_RNA < 40
000 & percent.mt < 5)
# Doublet detection
HB53_tumor_sce <- as.SingleCellExperiment(HB53_tumor)
HB53_tumor_sce <- scDblFinder(HB53_tumor_sce)
HB53_tumor$scDblFinder_class <- HB53_tumor_sce$scDblFinder.class
HB53_tumor_filtered <- subset(HB53_tumor, subset = scDblFinder_class == "singlet")
table(HB53_tumor$scDblFinder_class)
```

```
##
## singlet doublet
##    9500    1606
```

Filtering results

```
# Define sample IDs corresponding to Seurat object names
sample_ids <- c("HB17_bg", "HB17_PDX", "HB17_tumor",
                "HB30_PDX", "HB30_tumor", "HB53_bg", "HB53_tumor")

# Initialize a list to store summary results
summary_list <- list()

for (id in sample_ids) {
  # Retrieve the raw (pre-filtering) and processed (post-filtering) Seurat objects
  initial_obj <- get(paste0(id, "_initial"))
  filtered_obj <- get(id)

  # Get cell and gene counts before filtering
  cells_before <- ncol(initial_obj)
  genes_before <- nrow(initial_obj)

  # Get cell and gene counts after doublet removal (if available)
  if (!is.null(filtered_obj$scDblFinder_class)) {
    singlets <- subset(filtered_obj, subset = scDblFinder_class == "singlet")
  } else {
    singlets <- filtered_obj  # Use the object as-is if doublet classification is not present
  }
  cells_after <- ncol(singlets)
  genes_after <- nrow(singlets)

  # Store the result as a one-row data frame
  summary_list[[id]] <- data.frame(
    Cells_Before = cells_before,
    Cells_After = cells_after,
    Genes_Before = genes_before,
    Genes_After = genes_after
  )
}

# Combine all rows into a single data frame
summary_table <- do.call(rbind, summary_list)
print(summary_table)
```

```
##            Cells_Before Cells_After Genes_Before Genes_After
## HB17_bg           11197        6867        33538       20519
## HB17_PDX           8027        7375        33538       25234
## HB17_tumor         7995        7079        33538       25334
## HB30_PDX          10303        8337        33538       25871
## HB30_tumor        19042       11667        33538       26902
## HB53_bg            8540        6954        33538       25204
## HB53_tumor        12832        9500        33538       26845
```

Discussion

I set sample-specific QC cutoffs by inspecting violin plots of the number of unique genes detected per barcode, the total number of molecules per barcode, and the percent mitochondrial reads. For each sample, I removed cells with very low feature counts (nFeature_RNA < 500–1000), indicative of dead or empty droplets; cells with abnormally high feature or molecule counts (nFeature_RNA > 5000–10000 or nCount_RNA > 25000–40000), suggestive of doublets; and cells with elevated mitochondrial fraction (percent.mt > 5–20%), reflecting compromised integrity. Before filtering, the dataset comprised 77,936 cells, which was reduced to 57,779 cells after QC. Likewise, out of an average of 33,538 genes per sample, a mean of 25,130 genes remained following filtering. A common strategy for setting thresholds without relying on visual inspection of plots is to calculate each QC metric's median (M) and median absolute deviation (MAD), then flag as outliers any cells with values outside the interval M±k×MAD (typically k=3).

Count Normalization

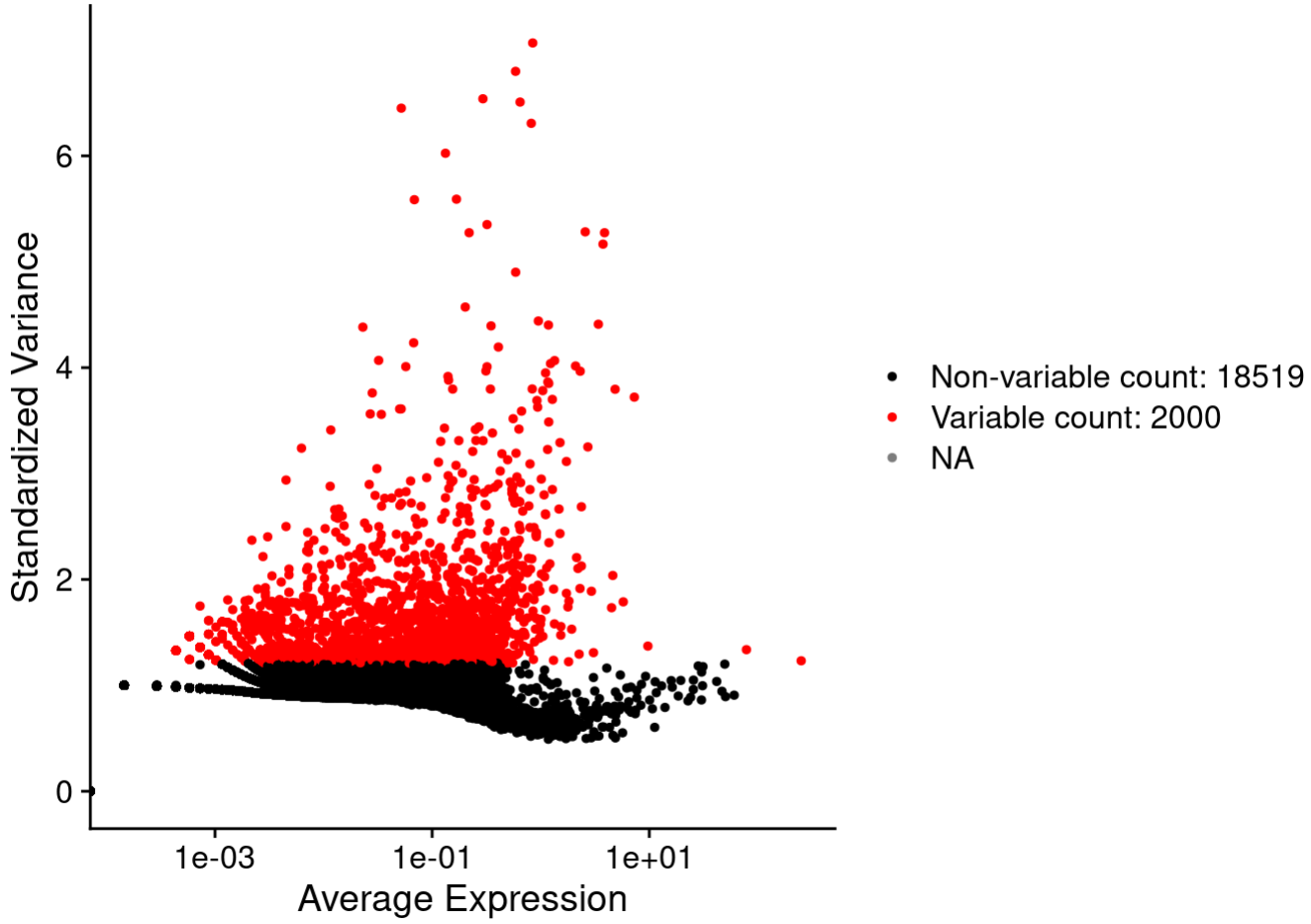
```
# Merge filtered data
combined <- merge(
  x = HB17_bg_filtered,
  y = list(HB17_PDX_filtered, HB17_tumor_filtered, HB30_PDX_filtered, HB30_tumor_filtered, HB53_bg_fil
tered, HB53_tumor_filtered),
  add.cell.ids = c("HB17_bg", "HB17_PDX", "HB17_tumor", "HB30_PDX", "HB30_tumor", "HB53_bg", "HB53_tum
or")
)
# Normalize the data
combined <- NormalizeData(combined)
```

Discussion

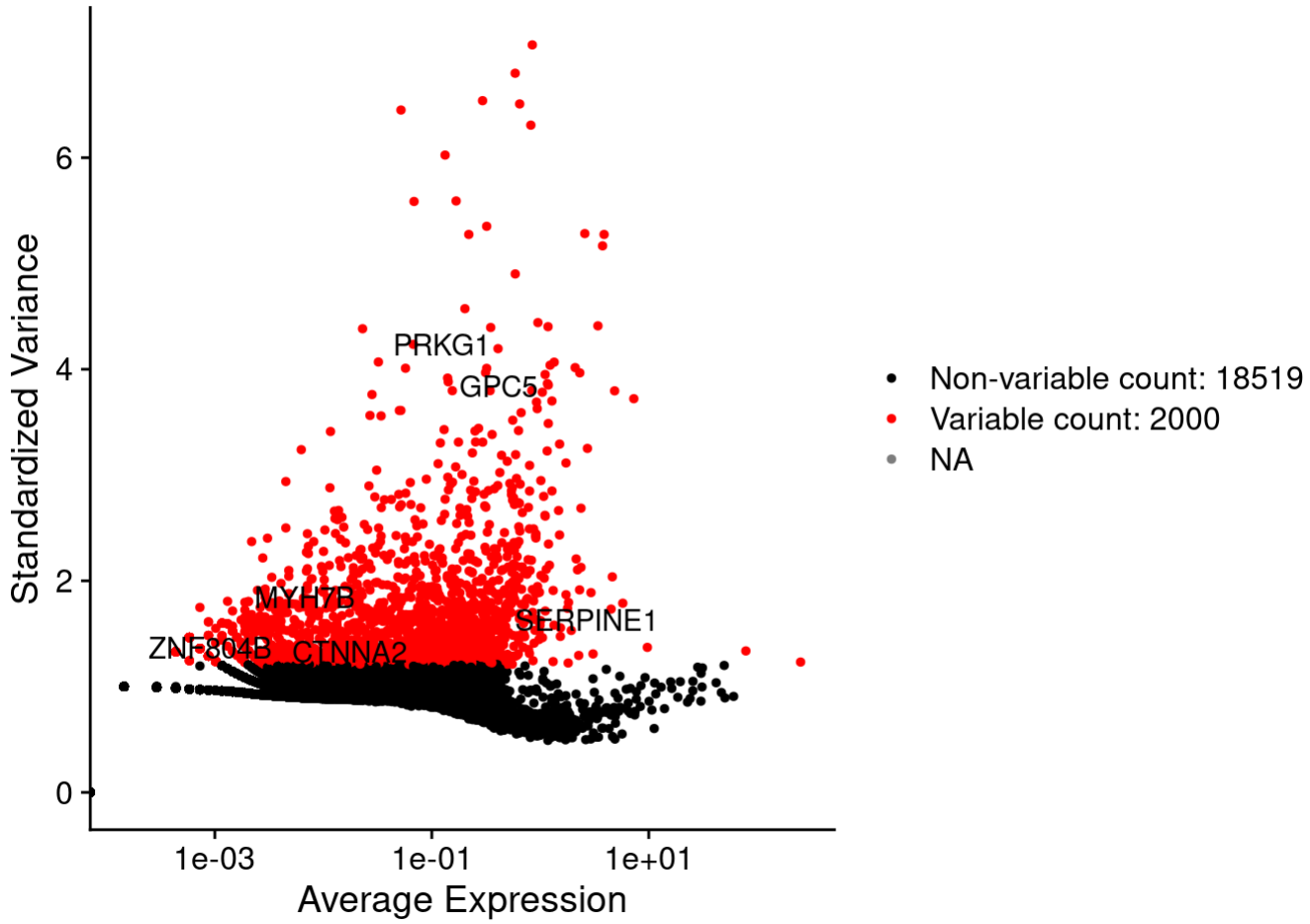
I applied Seurat's LogNormalize method. For each cell, raw UMI counts are divided by the cell's total counts, multiplied by a scale factor of 10000, and then transformed as log(x+1). This ensures all cells are brought to a common scale and reduces the impact of library-size differences.

Feature Selection

```
combined <- FindVariableFeatures(combined, selection.method = "vst", nfeatures = 2000)
# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(combined), 10)
# plot variable features with and without labels
VariableFeaturePlot(combined)
```




```
plot1 <- VariableFeaturePlot(combined)
LabelPoints(plot = plot1, points = top10, repel = TRUE)
```

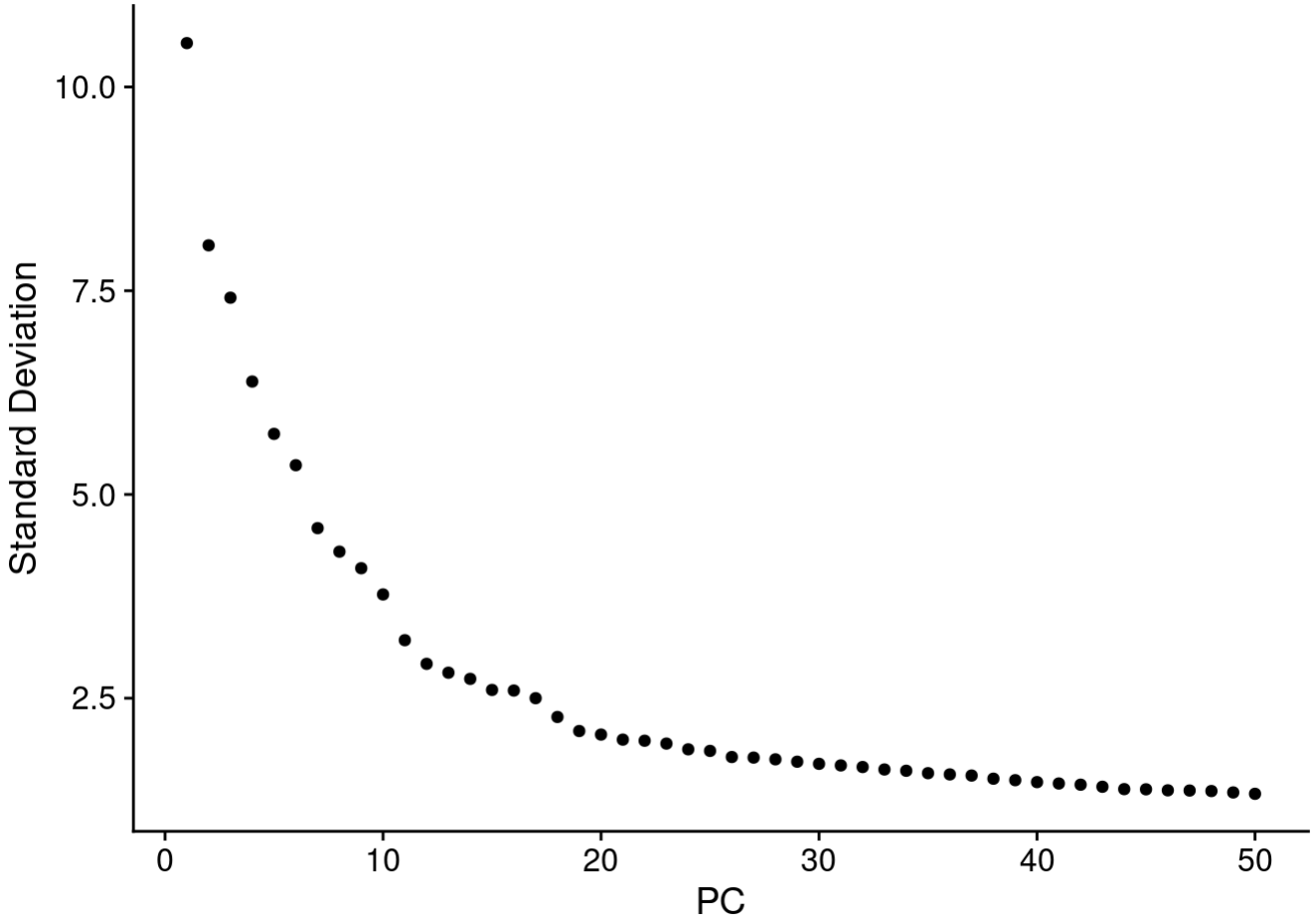


Discussion

I used vst to fit a mean–variance trend and select the top 2,000 genes by standardized variance for downstream PCA and UMAP. Out of roughly 20,519 detected genes, 2,000 exceeded the variability threshold and were retained as “highly variable,” while the remaining 18,519 did not meet this cutoff.

PCA

```
# Scaling the data
all.genes <- rownames(combined)
combined <- ScaleData(combined, features = all.genes)
# Perform PCA on the scaled data
combined <- RunPCA(combined, features = VariableFeatures(object = combined))
# Visualize PCA results
ElbowPlot(combined, ndims = 50)
```



Discussion

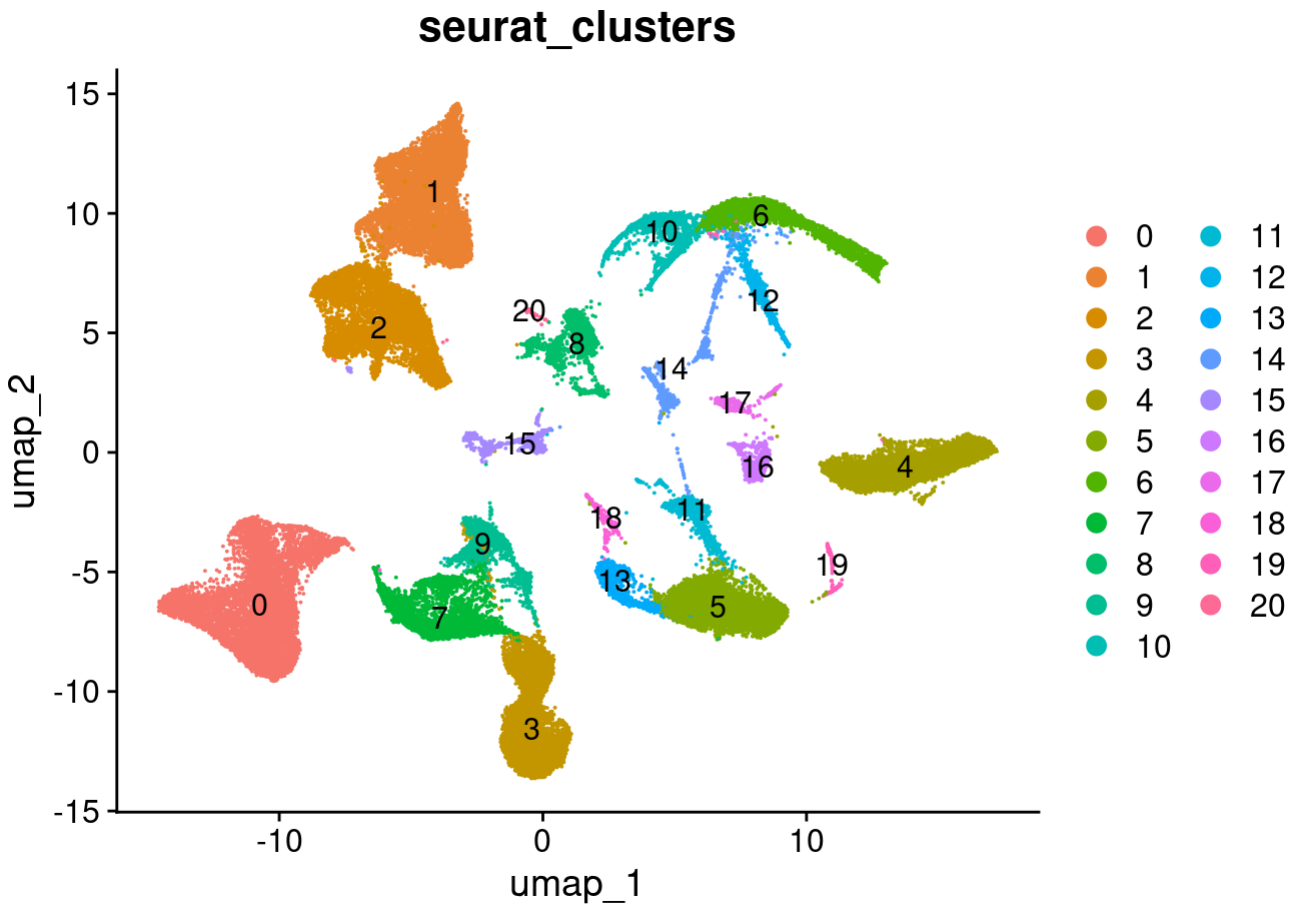
I selected the first 25 principal components because the elbow plot shows that the standard deviation stabilizes at PC25, indicating that additional components add minimal new information. Using PCs 1–25 thus preserves most biological signal while reducing noise.

Clustering and visualization

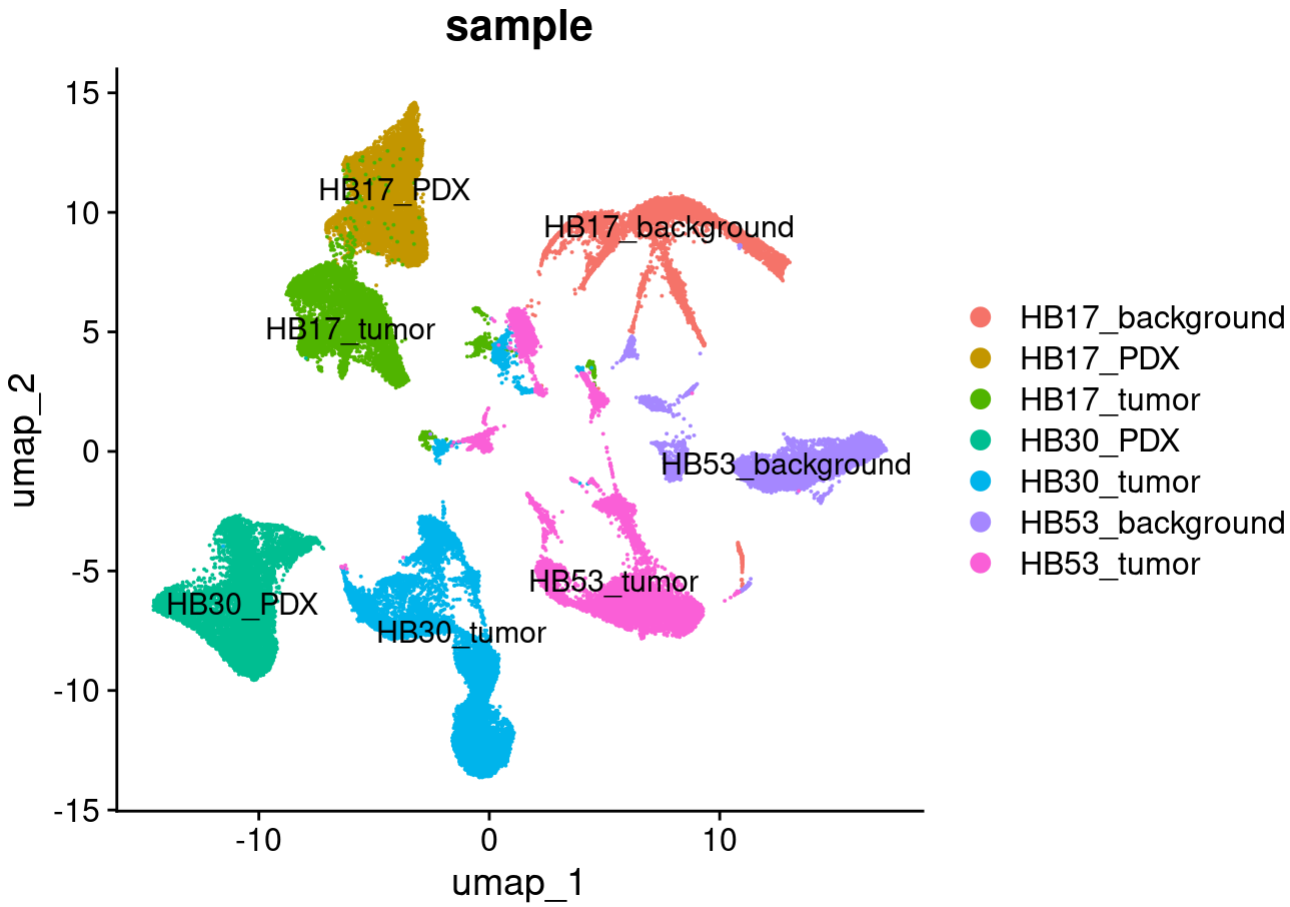
```
# Find neighbors with top 25 components
combined <- FindNeighbors(combined, dims = 1:25)
# Perform clustering
combined <- FindClusters(combined, resolution = 0.2)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 57779
## Number of edges: 2188224
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9779
## Number of communities: 21
## Elapsed time: 16 seconds
```

```
# UMAP
combined <- RunUMAP(combined, dims = 1:25)
# Visualization
DimPlot(combined, reduction = "umap", group.by = "seurat_clusters", label = TRUE)
```




```
DimPlot(combined, reduction = "umap", group.by = "sample", label = TRUE)
```



Discussion

The per-sample cell counts were 6867 (HB17_bg), 7375 (HB17_PDX), 7079 (HB17_tumor), 8337 (HB30_PDX), 11667 (HB30_tumor), 6954 (HB53_bg), and 9500 (HB53_tumor), for a total of 57779 cells across all seven samples. Unsupervised clustering at a resolution of 0.2 yielded 20 distinct clusters. However, when we colored the UMAP embedding by sample origin, cells from each sample formed largely separate islands, indicating the presence of batch effects. Consequently, we proceeded to apply data integration (Harmony) before downstream analysis.

Integration

```
# Integration of data from different samples
combined_harmony <- RunHarmony(combined, group.by.vars = "sample")
# Find neighbors with top 25 components of Harmony-corrected cell embeddings
combined_harmony <- FindNeighbors(combined_harmony, reduction = "harmony", dims = 1:25)
```

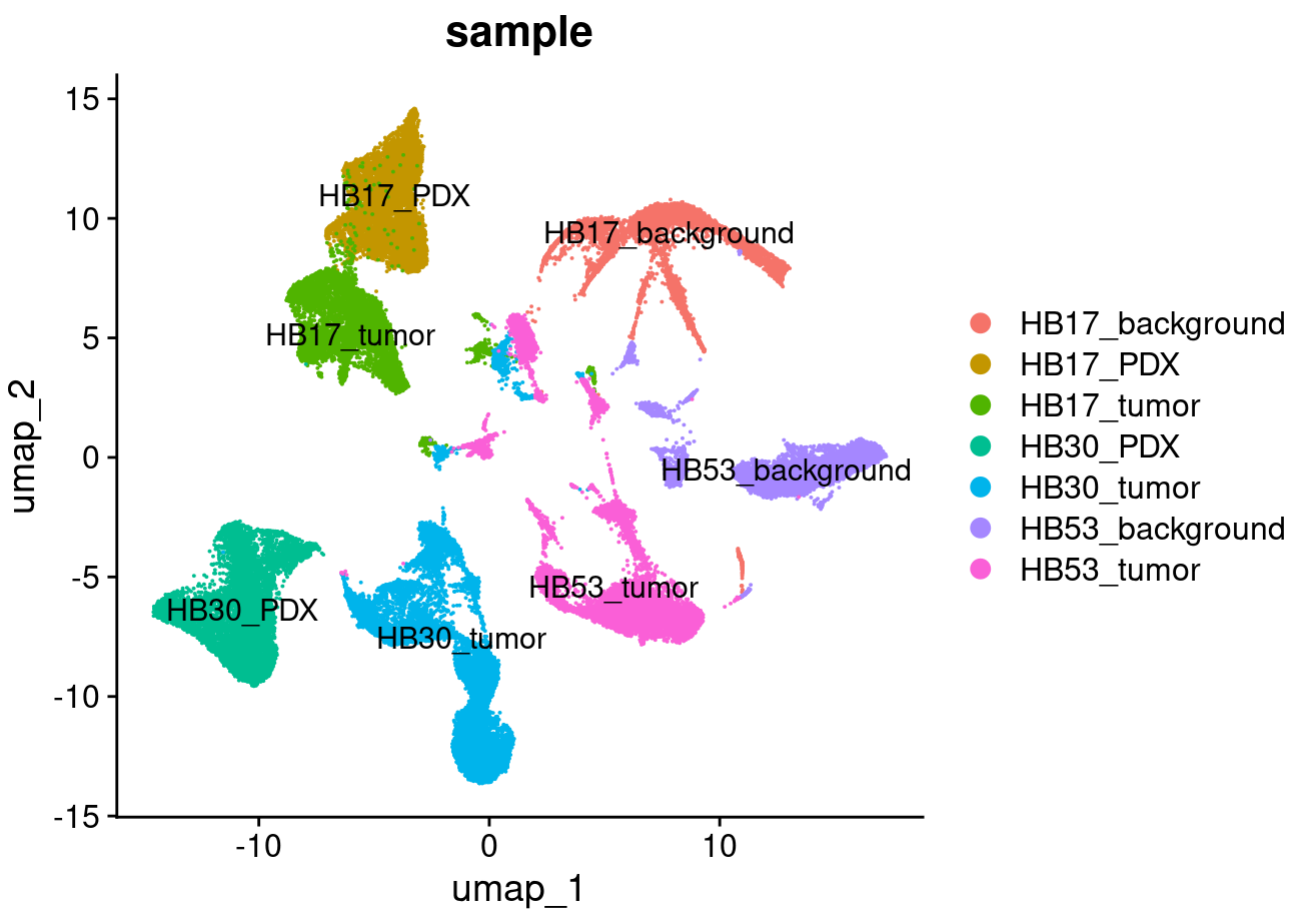
```
# Perform clustering
combined_harmony <- FindClusters(combined_harmony, resolution = 0.2)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 57779
## Number of edges: 1955298
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9572
## Number of communities: 12
## Elapsed time: 17 seconds
```

```
# UMAP
combined_harmony <- RunUMAP(combined_harmony, reduction = "harmony", dims = 1:25)
# Visualization
print("Before Integration")
```

```
## [1] "Before Integration"
```

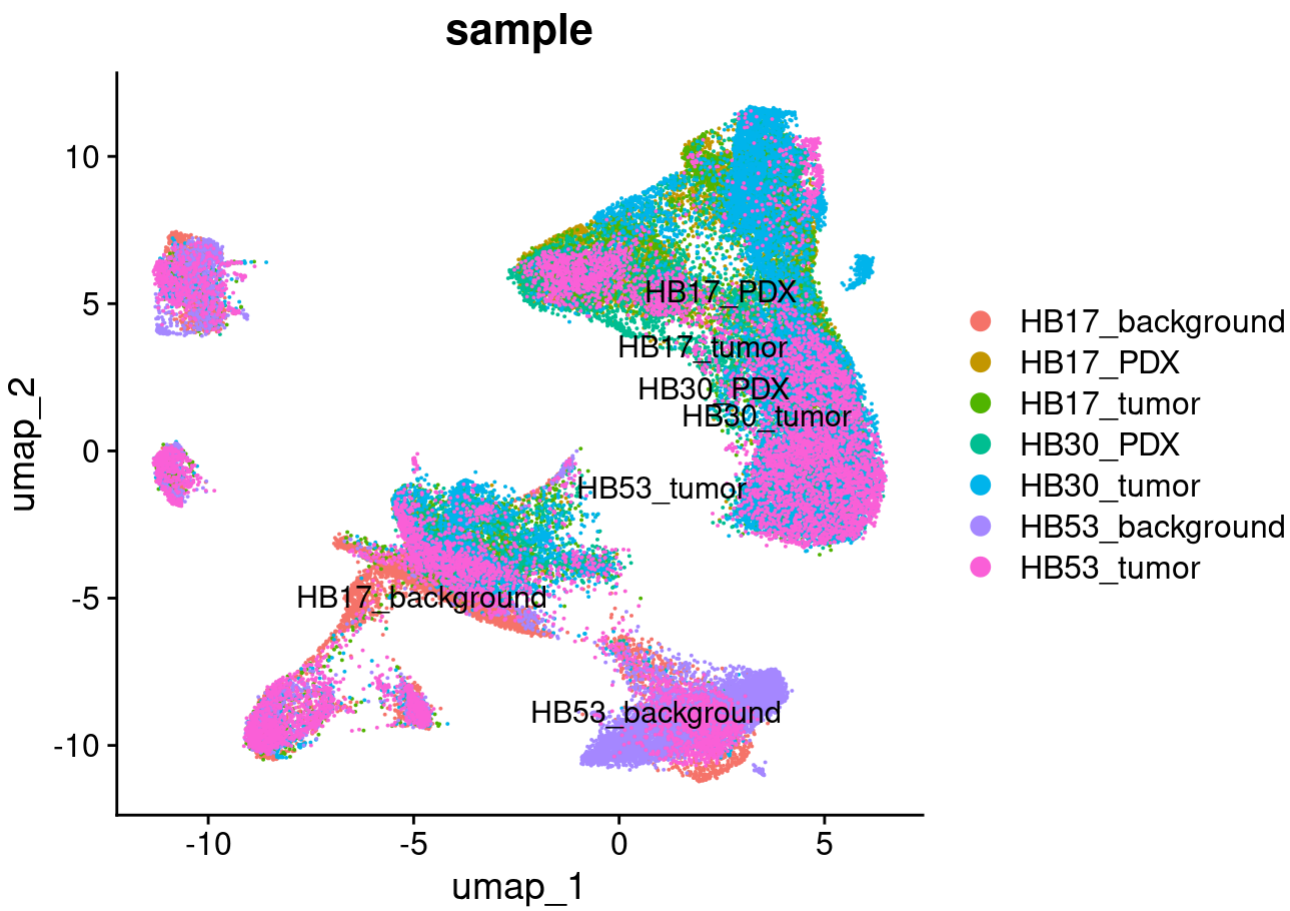
```
DimPlot(combined, reduction = "umap", group.by = "sample", label = TRUE) # before integration
```



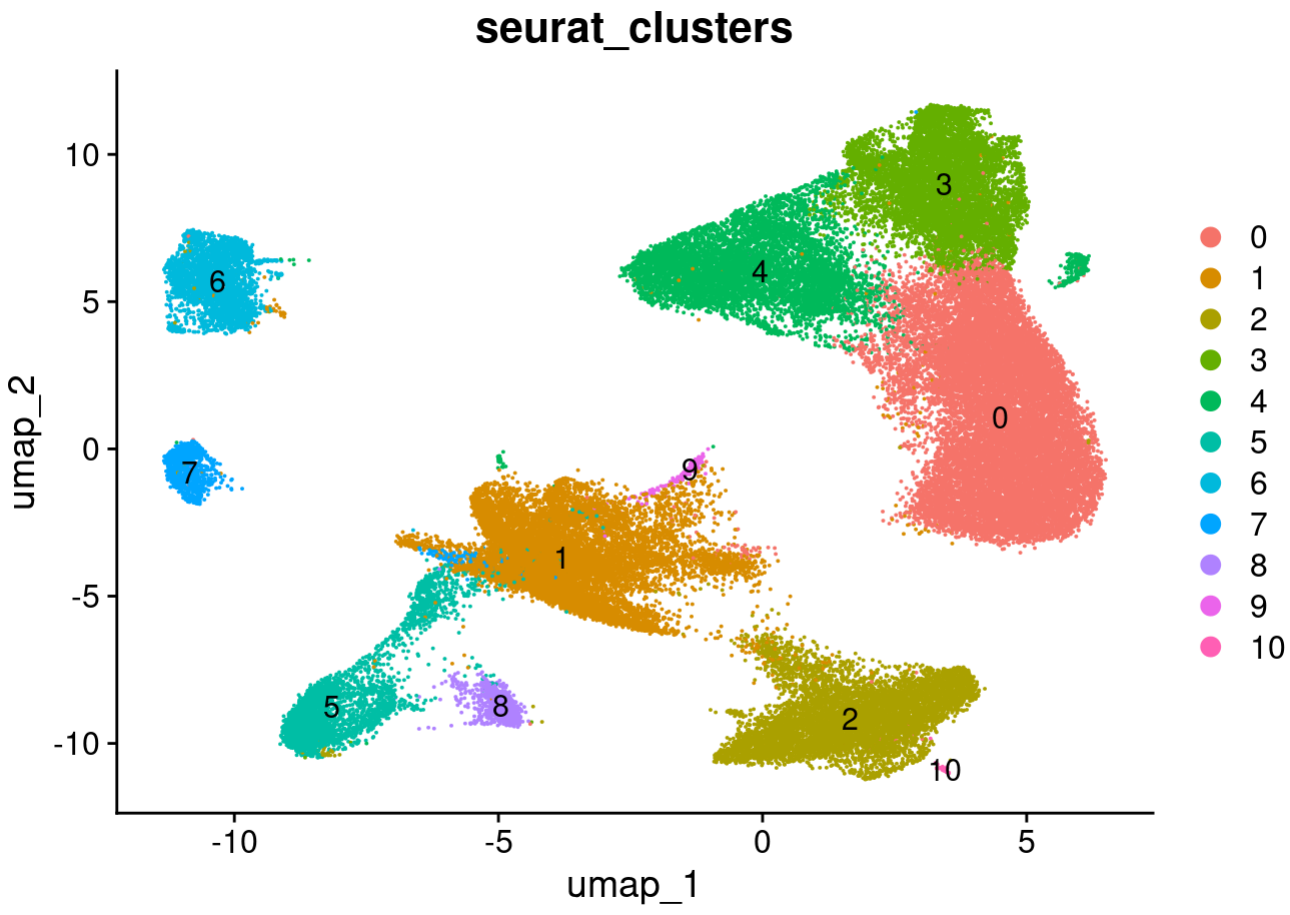

```
print("After Integration")
```

```
## [1] "After Integration"
```

```
DimPlot(combined_harmony, reduction = "umap", group.by = "sample", label = TRUE) # after integration
```




```
DimPlot(combined_harmony, reduction = "umap", group.by = "seurat_clusters", label = TRUE)
```



Discussion

Before integration, cells segregate almost entirely by sample, with each dataset forming its own island in UMAP space, providing evidence of batch-driven separation. After running Harmony, cells from different samples are well intermixed within shared clusters, indicating that technical differences have been effectively removed and that true biological variation is now the primary driver of the embedding.

Marker Gene Analysis

```
# Find markers for all clusters
combined_harmony <- JoinLayers(combined_harmony)
all.markers <- FindAllMarkers(object = combined_harmony, only.pos = TRUE)

# Listing the top five marker genes for each of the clusters
all.markers <- readRDS("all_markers.rds")
top5 <- all.markers %>%
  group_by(cluster) %>%
  top_n(5, avg_log2FC) %>%
  select(cluster, gene, avg_log2FC, p_val_adj, pct.1, pct.2)
print(top5, n = Inf)
```

```
## # A tibble: 55 × 6
## # Groups:   cluster [11]
##    cluster gene       avg_log2FC p_val_adj pct.1 pct.2
##    <fct>   <chr>           <dbl>     <dbl> <dbl> <dbl>
##  1 0       AC008991.1       2.89 0         0.084 0.01
##  2 0       AL360007.1       2.77 0         0.077 0.013
##  3 0       AC104809.1       2.74 1.98e-139 0.025 0.003
##  4 0       AL133330.2       3.21 1.05e- 72 0.013 0.001
##  5 0       AC129492.2       2.94 2.43e- 71 0.013 0.001
##  6 1       HRG              2.26 0         0.265 0.139
##  7 1       SAA4             2.28 0         0.145 0.043
##  8 1       APOF             2.28 0         0.121 0.031
##  9 1       CFHR2            2.27 1.30e- 97 0.033 0.008
## 10 1       MT1HL1           2.52 7.40e- 66 0.019 0.004
## 11 2       LINC02197        5.72 0         0.62  0.016
## 12 2       AC007221.2       5.53 0         0.285 0.006
## 13 2       AC025810.1       5.56 0         0.083 0.001
## 14 2       SLITRK3          6.01 0         0.05  0
## 15 2       FGF14-AS1        5.52 2.46e-242 0.024 0
## 16 3       AC090138.1       6.71 9.71e-258 0.026 0
## 17 3       AC079117.1       5.46 1.22e-180 0.021 0.001
## 18 3       HTN1             4.83 5.46e-150 0.021 0.001
## 19 3       AC007405.2       4.94 4.19e-103 0.013 0.001
## 20 3       AC016716.2       4.87 2.24e- 85 0.011 0
## 21 4       KIF18B           4.05 0         0.604 0.037
## 22 4       PIF1             4.04 0         0.335 0.019
## 23 4       AL138789.1       4.22 0         0.043 0.001
## 24 4       AC002563.1       4.21 9.45e-177 0.022 0.001
## 25 4       HMMR-AS1         4.04 2.18e-126 0.016 0.001
## 26 5       IGSF21           6.48 0         0.29  0.013
## 27 5       AC079015.1       6.89 6.10e-274 0.031 0.001
## 28 5       AC138207.5       7.12 1.03e-163 0.017 0
## 29 5       LINC01645        6.48 1.09e-119 0.012 0
## 30 5       IGSF21-AS1       6.60 1.39e- 79 0.01  0
## 31 6       PTPRB            6.35 0         0.839 0.065
## 32 6       BTNL9            6.36 0         0.332 0.01
## 33 6       NOTCH4           6.29 0         0.321 0.01
## 34 6       AC010737.1       6.28 0         0.194 0.006
## 35 6       AL021937.3       6.22 0         0.052 0.001
## 36 7       TRPA1            7.92 0         0.062 0
## 37 7       FENDRR           8.06 4.66e-252 0.03  0
## 38 7       AC019270.1      10.2  8.66e-240 0.021 0
## 39 7       LINC00473        8.56 1.41e-159 0.016 0
## 40 7       SLC26A10         7.96 1.14e- 92 0.012 0
## 41 8       CD247            7.68 0         0.608 0.019
## 42 8       PTGDR            7.93 0         0.164 0.002
## 43 8       AC090627.1       8.53 2.67e-227 0.03  0
## 44 8       AL356276.1       8.43 1.01e-109 0.013 0
## 45 8       KIR2DL3          7.87 9.01e- 67 0.01  0
## 46 9       SFRP5            8.74 0         0.372 0.003
## 47 9       CXCL6            8.88 0         0.241 0.002
## 48 9       LINP1            9.57 0         0.036 0
## 49 9       ARL14            8.71 6.80e-216 0.059 0.001
## 50 9       AC073218.2       8.85 7.41e-110 0.016 0
## 51 10      LYPD4           10.2  6.60e-131 0.021 0
```

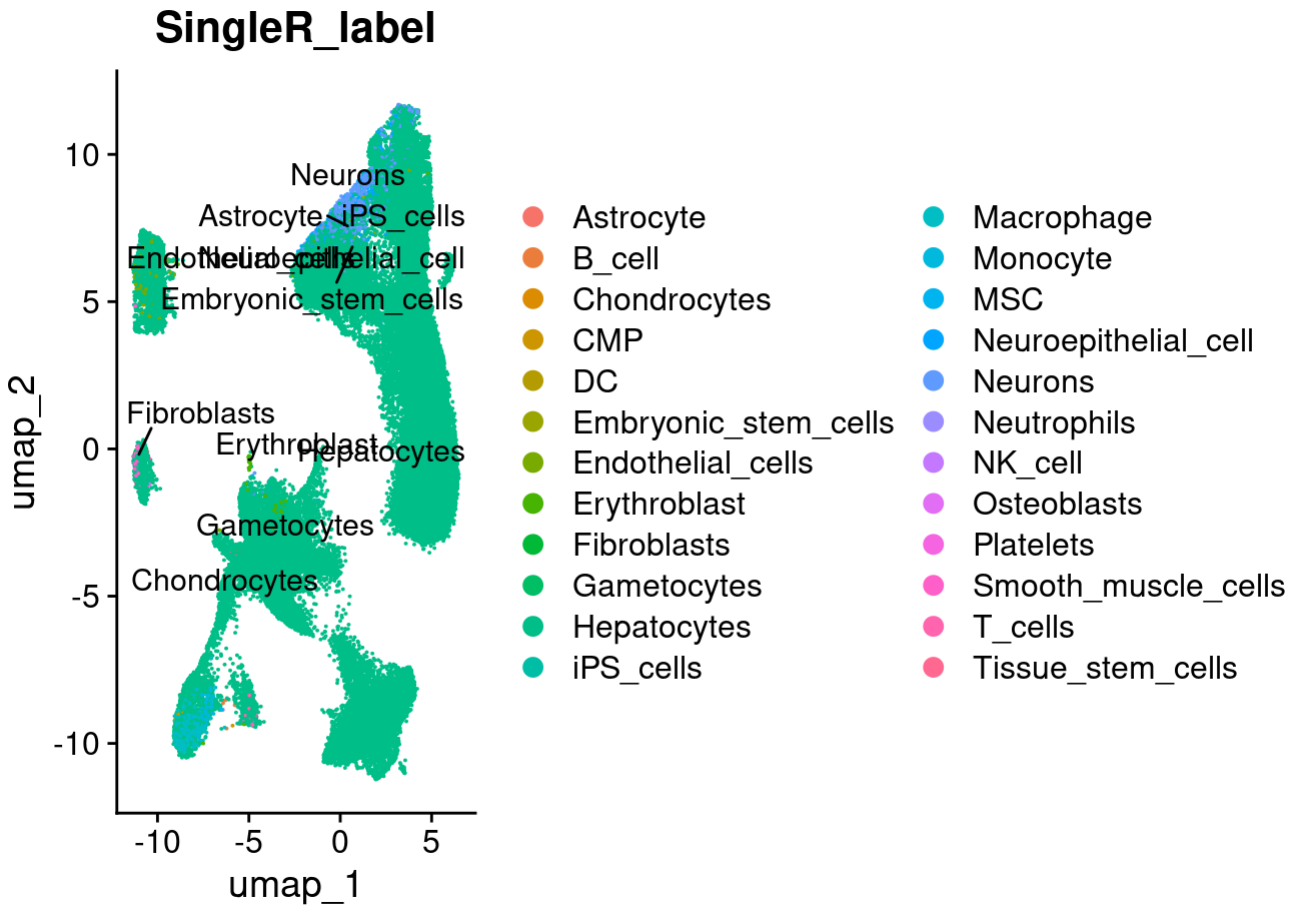
```
## 52 10      KCND3-AS1       8.81 5.75e- 64 0.021 0
## 53 10      AC012506.3      9.21 5.75e- 64 0.021 0
## 54 10      AC068228.1      8.91 3.08e- 35 0.021 0
## 55 10      PRAMEF33        8.52 1.94e- 30 0.021 0
```

Discussion

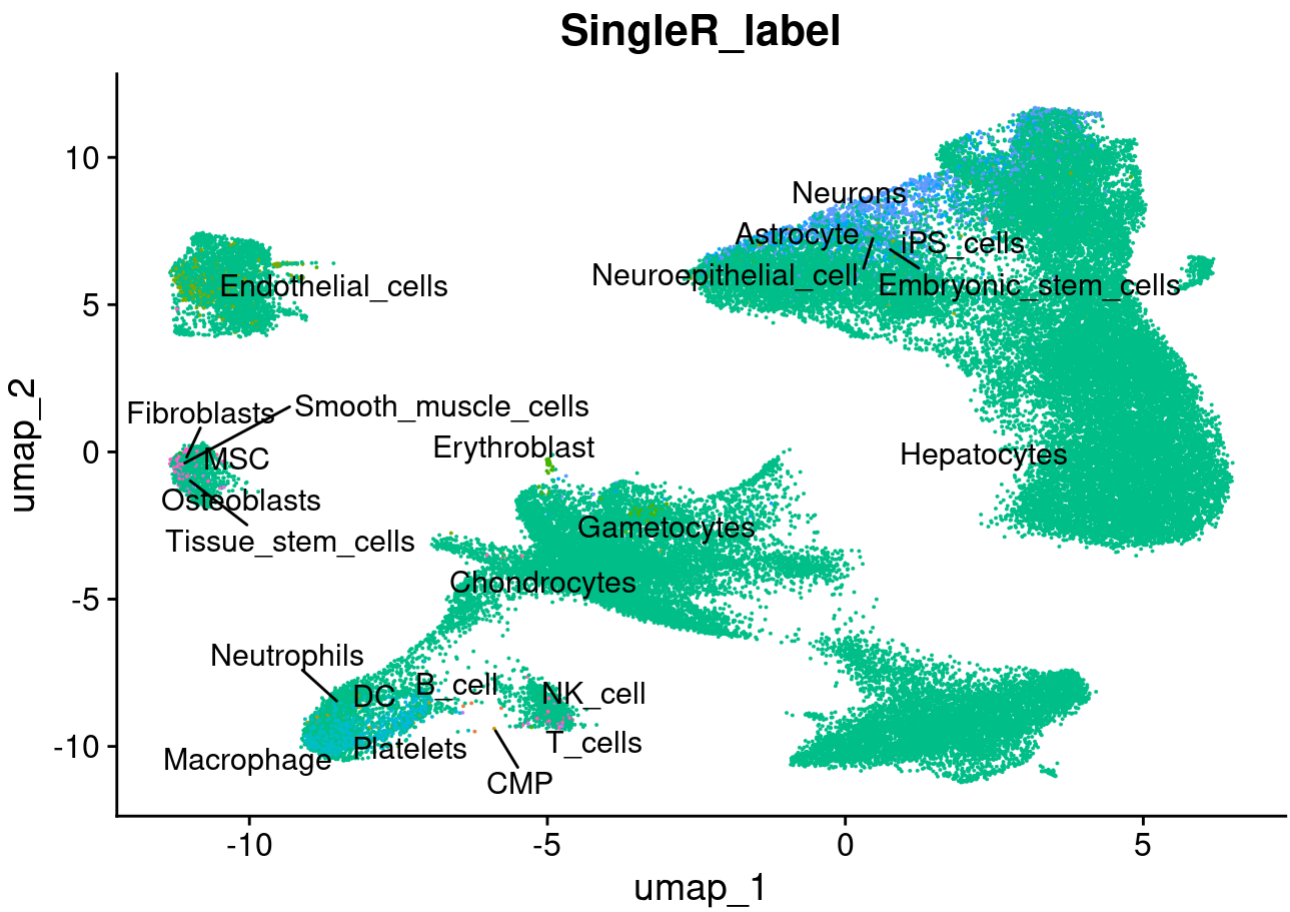
I used FindAllMarkers function in Seurat with the only.pos = TRUE option for merker gene analysis. This function performs a gene-by-gene comparison between each cluster and all other cells using a nonparametric Wilcoxon rank-sum test. By setting only.pos = TRUE, it returns only those genes that are significantly more highly expressed in a given cluster than elsewhere, along with p-values, adjusted p-values, average log fold-changes and detection percentages. This method integrates seamlessly into the Seurat workflow, makes minimal distributional assumptions, and directly identifies cluster-specific markers. However, it can be slow on large datasets, its results depend strongly on parameter settings (e.g., detection fraction, log-fold change threshold), and it remains sensitive to dropout noise and residual batch effects despite upstream correction.

Automatic Annotation of Cell labels

```
combined_harmony <- JoinLayers(combined_harmony)
ref  <- HumanPrimaryCellAtlasData()
sce  <- as.SingleCellExperiment(combined_harmony)
pred <- SingleR(test = sce, ref = ref, labels = ref$label.main)
combined_harmony$SingleR_label <- pred$labels
DimPlot(combined_harmony, group.by = "SingleR_label", label = TRUE, repel = TRUE) # with legend
```



```
DimPlot(combined_harmony, group.by = "SingleR_label", label = TRUE, repel = TRUE) + NoLegend() # without legend
```



Discussion

SingleR assigns each query cell a label by comparing its expression profile to reference “pseudo-bulk” profiles from pure cell types. For each cell, it computes similarity scores to every reference label, initially assigns the best-matching label, and then refines assignments by aggregating across the dataset to improve robustness.

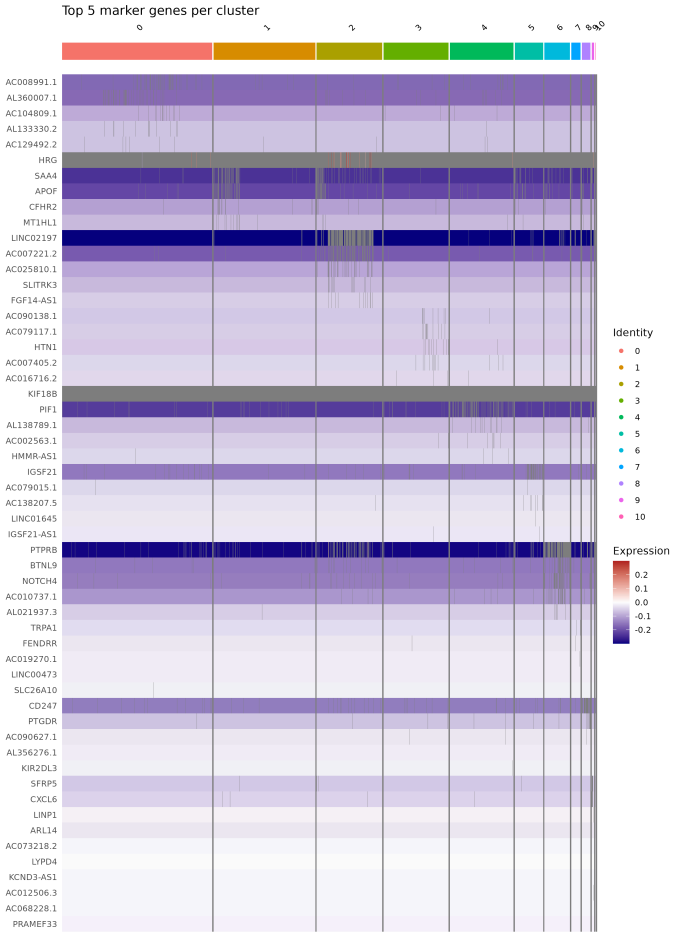
Comparing the results from marker gene analysis and singleR, clusters2 and 10, which derive almost exclusively from the HB53_background sample, express canonical hepatocyte markers, confirming they represent normal liver parenchyma. In contrast, the remaining clusters (0,1,3,4,5,6,7,8,9) originate predominantly from tumor or PDX samples and comprise a mixture of cell types—tumor-like epithelial cells, various immune lineages (neutrophils, macrophages, T/NK cells), endothelial cells, and stromal populations (fibroblasts, stellate cell–like), as well as a small subset labeled by SingleR as developmental/neuronal. This heterogeneity reflects the complex tumor microenvironment with both malignant cells and infiltrating non-parenchymal cell types.

Reference: Aran D, Looney AP, Liu L, Wu E, Fong V, Hsu A, Chak S, Naikawadi RP, Wolters PJ, Abate AR, Butte AJ, Bhattacharya M (2019). “Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage.” Nat. Immunol., 20, 163-172. doi:10.1038/s41590-018-0276-y (doi:10.1038/s41590-018-0276-y).

Manual Cluster Labeling

Heatmap for top marker genes

```
genes_all <- unique(top5$gene)
p1 <- DoHeatmap(
  object = combined_harmony,
  features = genes_all,
  assay = "RNA",
  slot = "scale.data",
  group.by = "seurat_clusters",
  size = 3
) +
  scale_fill_gradientn(colors = c("navy", "white", "firebrick"), limits = c(-0.3, 0.3)) +
  ggtitle("Top 5 marker genes per cluster")

ggsave("heatmap_top5_markers.png", plot = p1, width = 10, height = 14, dpi = 300)
```

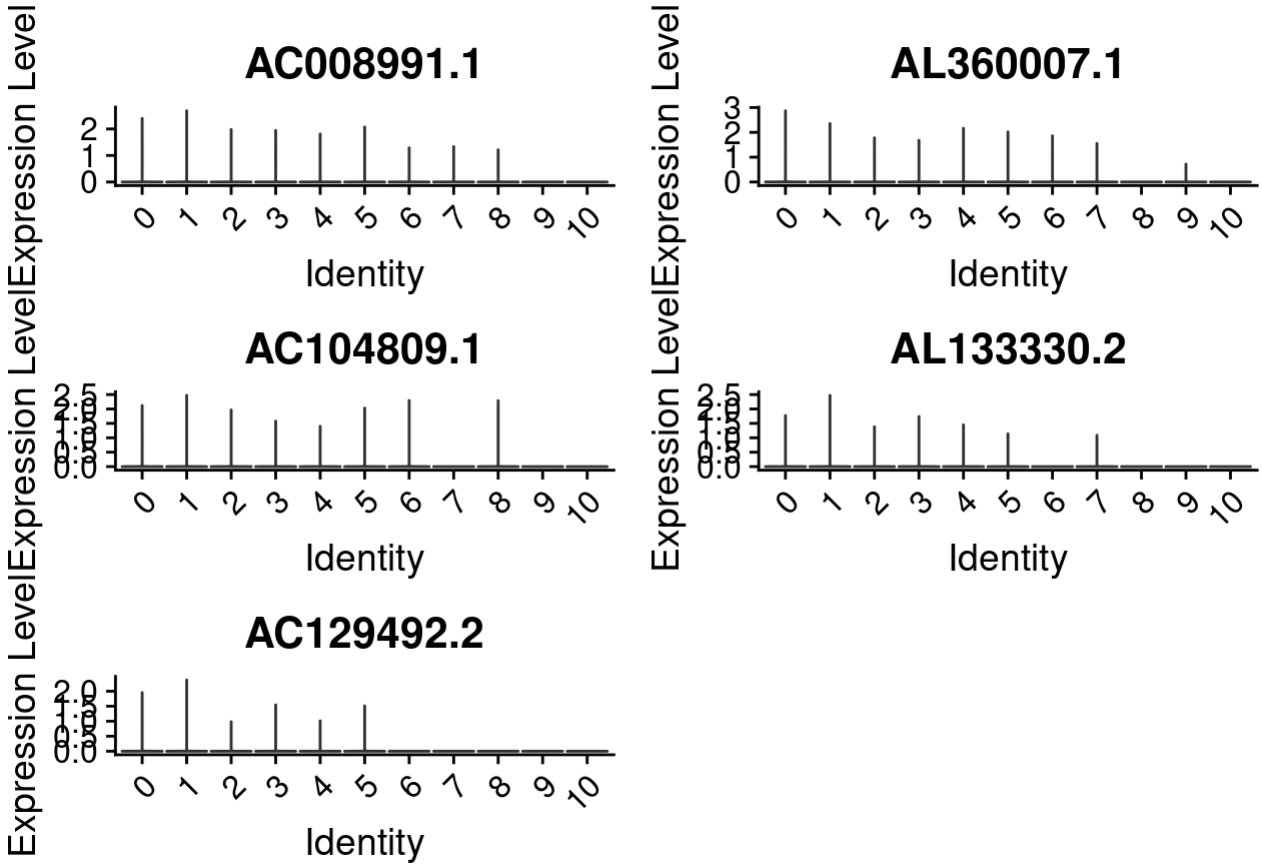


Violin plot of expression of top marker genes in

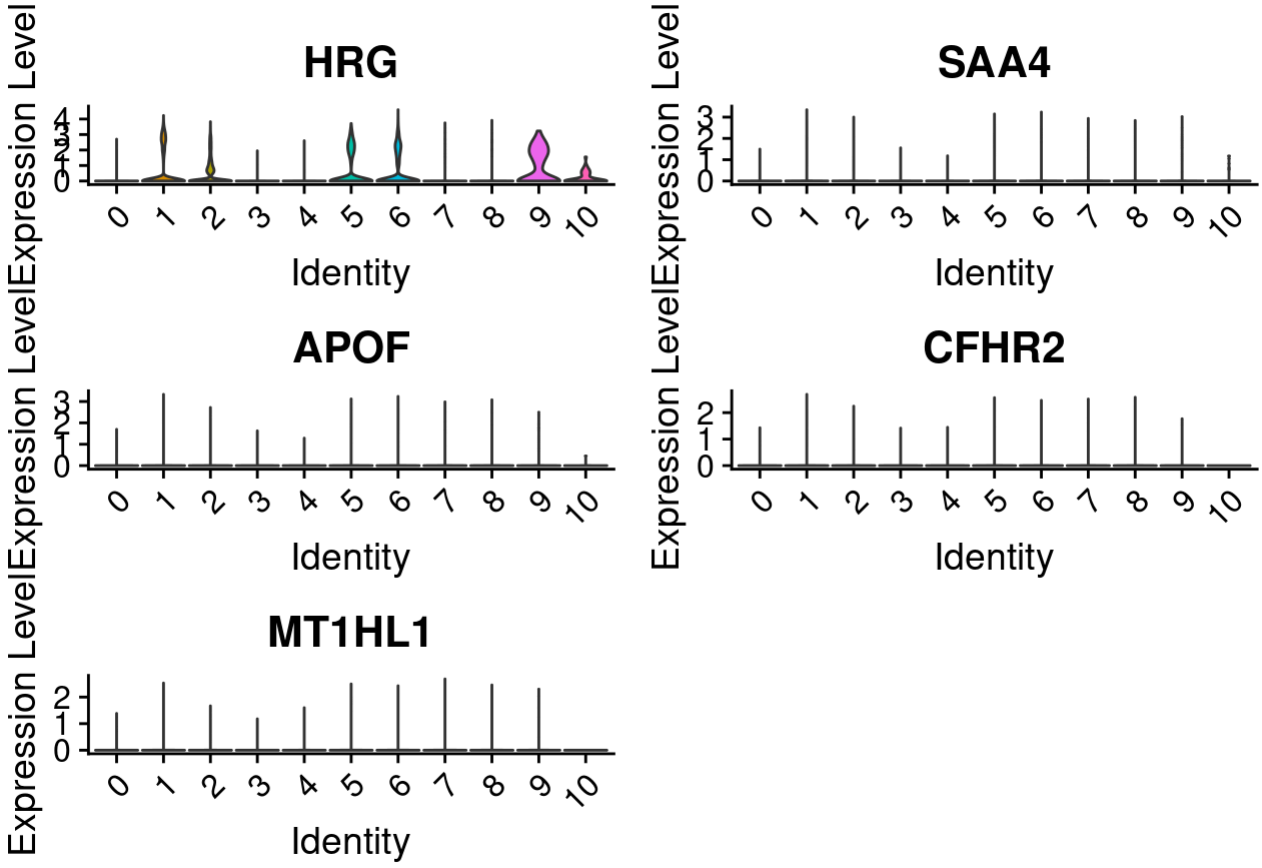
top three clusters

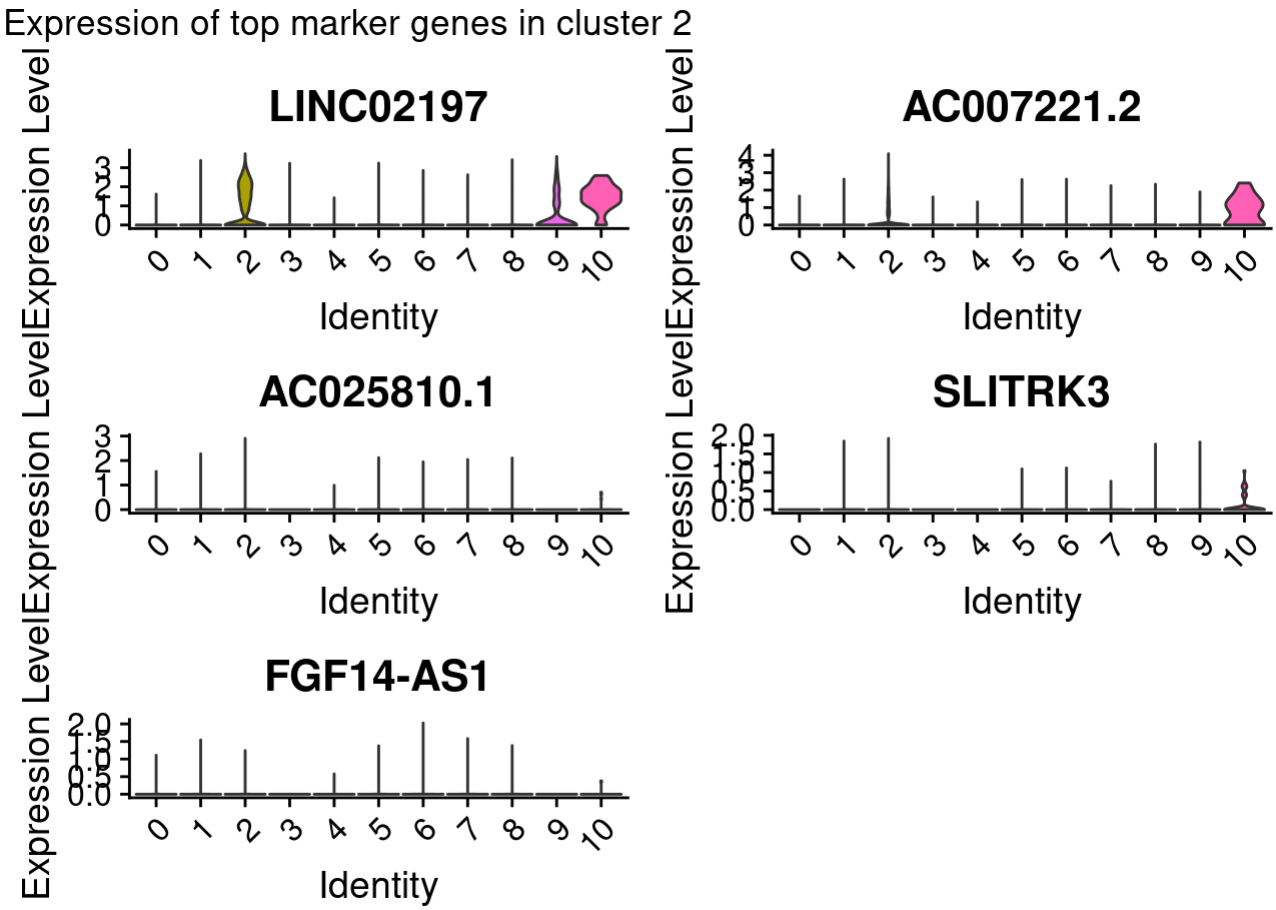
```
# Count the number of cells in each cluster
cluster_counts <- table(combined_harmony$seurat_clusters)
# Identify the top 3 clusters with the most cells
top_clusters <- names(sort(cluster_counts, decreasing = TRUE))[1:3]
# Extract top 5 marker genes per cluster (based on avg_log2FC)
top_markers_per_cluster <- all.markers %>%
  filter(cluster %in% top_clusters) %>%
  group_by(cluster) %>%
  top_n(5, avg_log2FC) %>%
  ungroup()
# Generate violin plots for each top cluster
for (clust in top_clusters) {

  genes <- top_markers_per_cluster %>%
    filter(cluster == clust) %>%
    pull(gene)

  p <- VlnPlot(
    object = combined_harmony,
    features = genes,
    group.by = "seurat_clusters",
    pt.size = 0,      # remove jittered dots
    ncol = 2          # arrange plots in 2 columns
  )

  final_plot <- wrap_elements(full = p) +
    plot_annotation(title = paste("Expression of top marker genes in cluster", clust))

  print(final_plot)
}
```

Expression of top marker genes in cluster 0



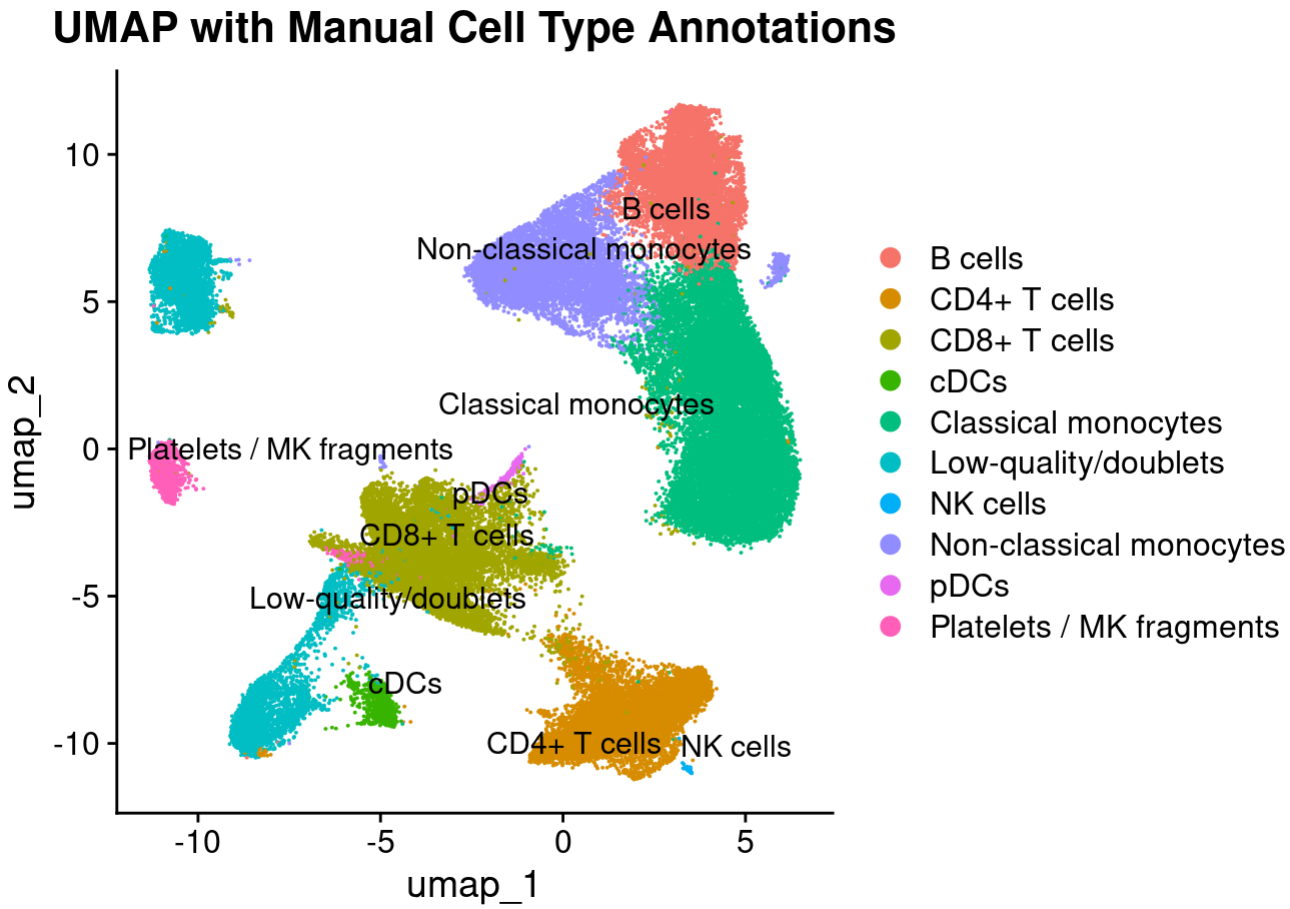
Expression of top marker genes in cluster 1




Expression of top marker genes in cluster 2
LINC02197
AC007221.2
AC025810.1
SLITRK3
FGF14-AS1
Expression Level
Identity
0 1 2 3 4 5 6 7 8 9 10

UMAP with Manual Cell Type Annotations

```
# Define manual annotations
manual_labels <- c(
  "Classical monocytes",             # cluster 0
  "CD8+ T cells",                    # cluster 1
  "CD4+ T cells",                    # cluster 2
  "B cells",                         # cluster 3
  "Non-classical monocytes",         # cluster 4
  "Low-quality/doublets",            # cluster 5
  "Low-quality/doublets",            # cluster 6
  "Platelets / MK fragments",        # cluster 7
  "cDCs",                            # cluster 8
  "pDCs",                            # cluster 9
  "NK cells"                         # cluster 10
)
names(manual_labels) <- as.character(0:10)

# Convert cluster IDs to character
cluster_ids <- as.character(combined_harmony$seurat_clusters)

# Create named metadata vector
annot_vector <- manual_labels[cluster_ids]
names(annot_vector) <- colnames(combined_harmony)

# Add metadata
combined_harmony <- AddMetaData(
  object   = combined_harmony,
  metadata = annot_vector,
  col.name = "manual_celltype"
)

DimPlot(
  combined_harmony,
  group.by = "manual_celltype",
  label = TRUE,
  repel = TRUE
) + ggtitle("UMAP with Manual Cell Type Annotations")
```



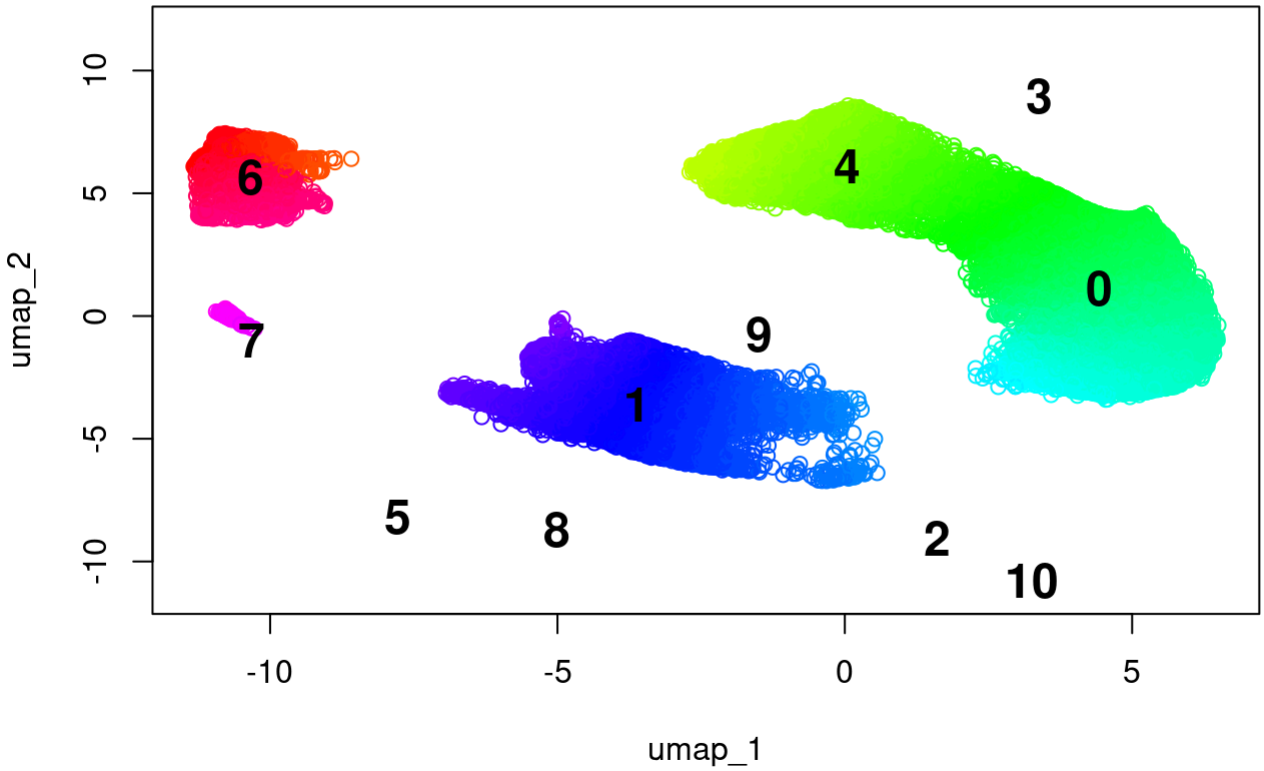
Discussion

Guided by SingleR predictions and cluster-specific marker genes, I annotated the 11 clusters as follows: cluster0 (CD14, LYZ,S100A8/9) and cluster4 (FCGR3A, MS4A7) were classified as classical and non-classical monocytes, respectively[1]; cluster1 (CD8A, GZMB, NKG7) and cluster2 (IL7R, MAL, CD69) correspond to cytotoxic $CD8^+$ and helper $CD4^+$ T cells[2]; cluster3 (MS4A1,CD79A) represents circulating B cells[3]; cluster7 (PF4, PPBP) consists of platelet/megakaryocyte fragments; cluster8 (HLA-DRA, FCER1A, CST3) matches conventional myeloid dendritic cells[1]; cluster9 (LILRA4) defines plasmacytoid dendritic cells; cluster10 (GNLY, NKG7, PRF1) contains natural killer cells. The residual small clusters5 and 6 express mostly housekeeping transcripts (NEAT1, XIST) and lack lineage markers, so they were deemed low-quality/doublet artefacts.

Reference. 1. Villani, A. C., Satija, R., Reynolds, G., Sarkizova, S., Shekhar, K., Fletcher, J., … & Hacohen, N. (2017). Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. Science, 356(6335), eaah4573.
2. Szabo, P. A., Levitin, H. M., Miron, M., Snyder, M. E., Senda, T., Yuan, J., … & Sims, P. A. (2019). Single-cell transcriptomics of human T cells reveals tissue and activation signatures in health and disease. Nature communications, 10(1), 4706.
3. Schep, A. N., Wu, B., Buenrostro, J. D., & Greenleaf, W. J. (2017). chromVAR: inferring transcription-factor-associated accessibility from single-cell epigenomic data. Nature methods, 14(10), 975-978.

Pseudotime analysis

```
sce <- as.SingleCellExperiment(combined_harmony)
sce <- slingshot(sce, clusterLabels = 'seurat_clusters',
                 reducedDim = 'UMAP')
plot(reducedDims(sce)$UMAP, col = rainbow(100)[cut(sce$slingPseudotime_1, 100)])
centers <- aggregate(reducedDims(sce)$UMAP, by = list(cluster = sce$seurat_clusters), FUN = mean)
text(centers[,2], centers[,3], labels = centers$cluster, cex = 1.5, font = 2)
```



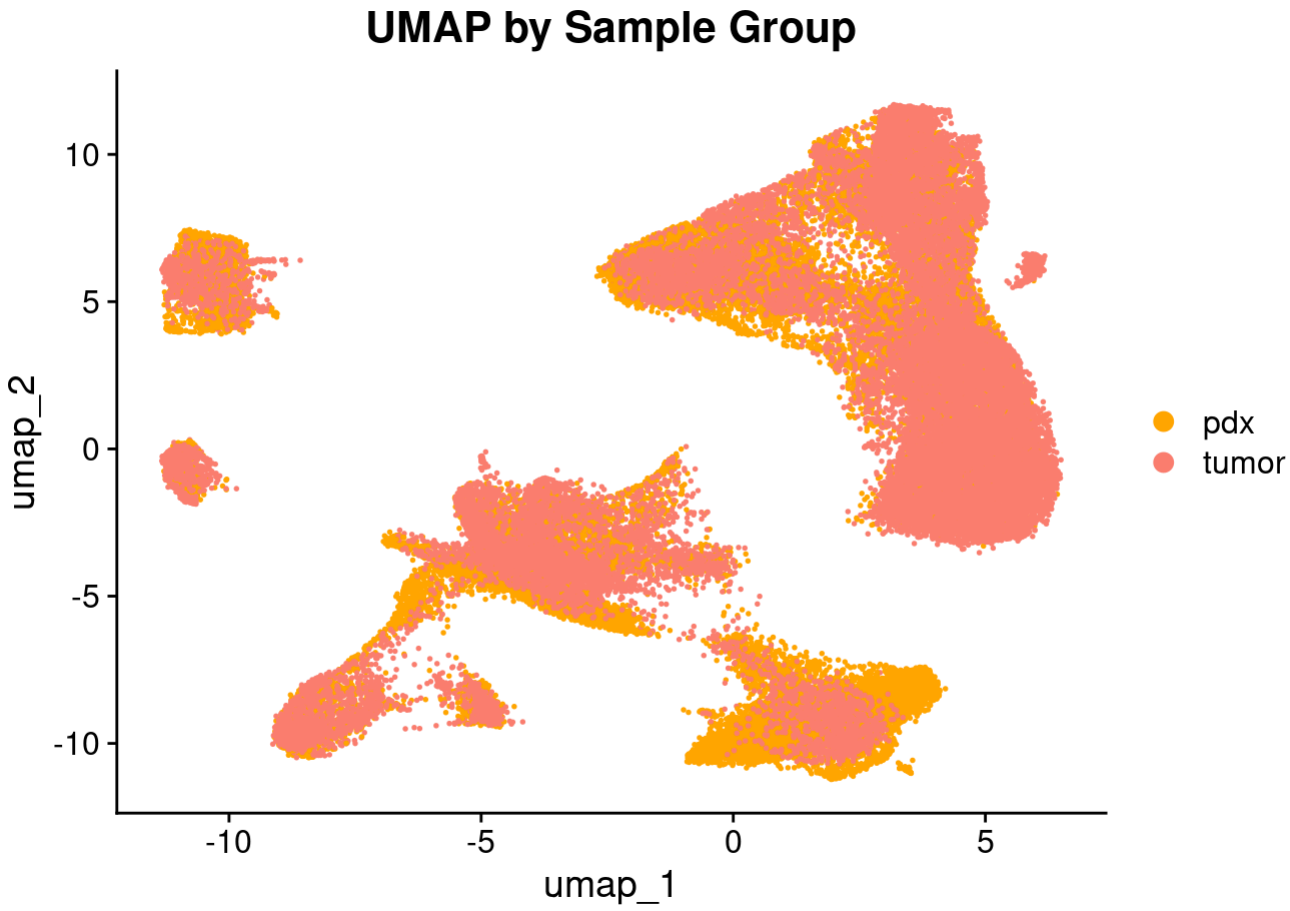
Discussion

Using Slingshot, I inferred pseudotime trajectories based on Seurat clustering and UMAP embeddings. The gradient of colors represents the inferred pseudotime, highlighting potential differentiation paths. For example, cells in cluster 6 appear to precede those in clusters 4, 0, and eventually cluster 1 along the inferred trajectory. Some clusters, such as 5 and 7, lack a pseudotime assignment, suggesting they may not participate in the main differentiation paths. These findings are consistent with previous results, where Slingshot successfully reconstructed lineage trajectories from scRNA-seq data[1].

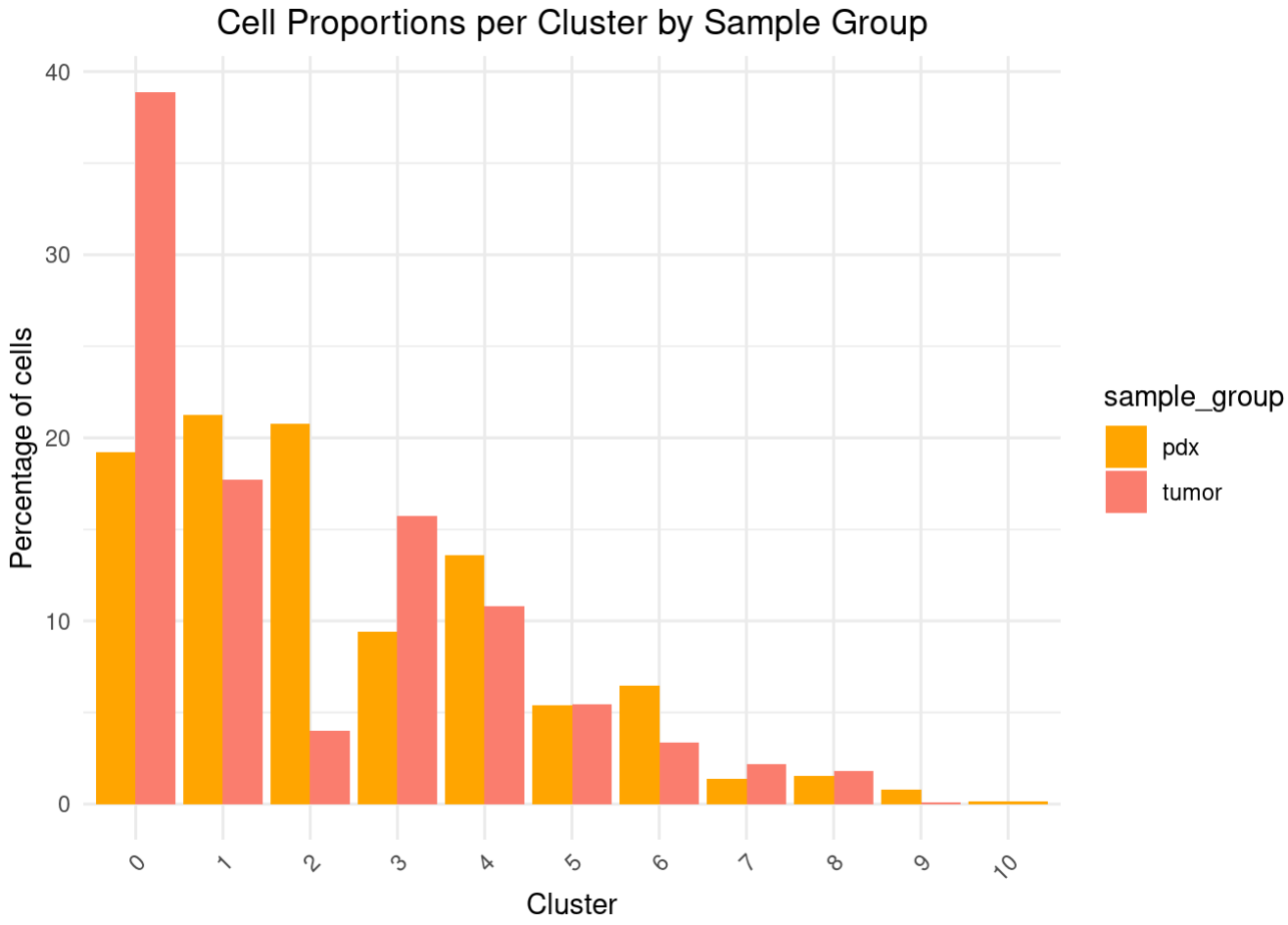
Reference: 1. Street, K., Risso, D., Fletcher, R. B., Das, D., Ngai, J., Yosef, N., … & Dudoit, S. (2018). Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. BMC genomics, 19, 1-16.

Cell Proportion analysis

```
# Create a new column classifying each sample as "tumor" or "pdx"
combined_harmony$sample_group <- ifelse(
  grepl("tumor", combined_harmony$sample, ignore.case = TRUE),
  "tumor",
  "pdx"
)

# UMAP colored by sample group
DimPlot(
  combined_harmony,
  group.by = "sample_group",
  pt.size = 0.3,
  cols = c("tumor" = "salmon", "pdx" = "orange")
) +
  ggtitle("UMAP by Sample Group") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
# Bar plot
combined_harmony@meta.data %>%
  count(sample_group, seurat_clusters) %>%
  group_by(sample_group) %>%
  mutate(percentage = n / sum(n) * 100) %>%
  ggplot(aes(x = seurat_clusters, y = percentage, fill = sample_group)) +
  geom_bar(stat = "identity", position = "dodge") +
  ylab("Percentage of cells") +
  xlab("Cluster") +
  scale_fill_manual(values = c("tumor" = "salmon", "pdx" = "orange")) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(hjust = 0.5)
  ) +
  ggtitle("Cell Proportions per Cluster by Sample Group")
```



Discussion

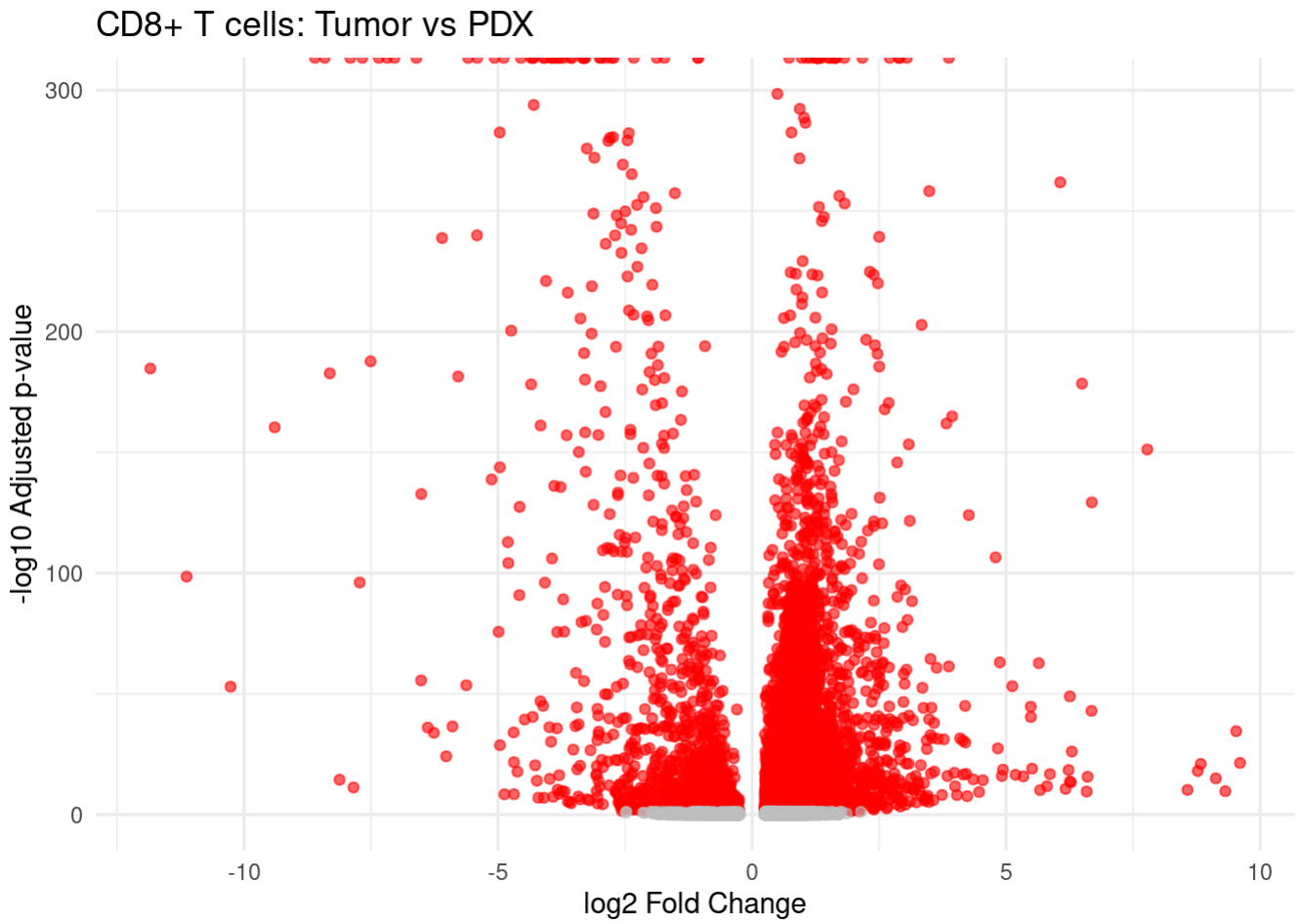
I compared the cell type composition between tumor and PDX samples. Most clusters were represented in both groups, though cluster 0 (classical monocytes) was more abundant in tumor, while clusters such as 2 (CD4⁺ T cells) and 4 (non-classical monocytes) were enriched in PDX. This suggests selective expansion or depletion of immune populations during engraftment, consistent with prior observations that PDX models partially recapitulate the original tumor microenvironment[1].

Reference: 1. Ben-David, U., Ha, G., Tseng, Y. Y., Greenwald, N. F., Oh, C., Shih, J., … & Golub, T. R. (2017). Patient-derived xenografts undergo mouse-specific tumor evolution. Nature genetics, 49(11), 1567-1575.

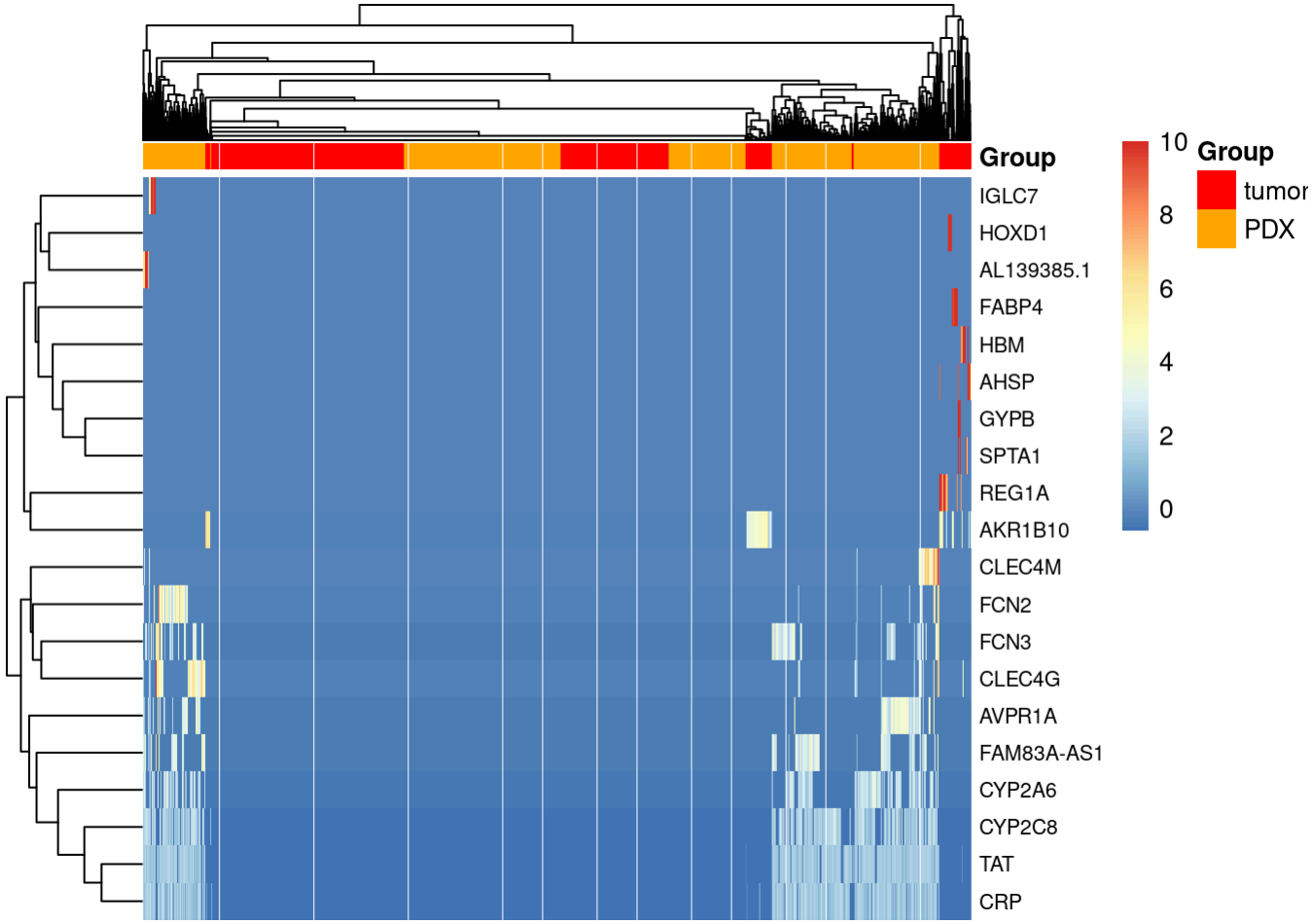
Additional analysis

Differential Expression and Visualization of CD8+ T Cells Between Tumor and PDX Conditions

```
# Subset Seurat object to CD8+ T cells
cd8_cells <- subset(combined_harmony, subset = manual_celltype == "CD8+ T cells")

# Create tumor vs PDX label based on sample metadata
cd8_cells$group <- ifelse(grepl("tumor", cd8_cells$sample, ignore.case = TRUE), "tumor", "PDX")

# Set identities for differential expression
Idents(cd8_cells) <- "group"

# Perform differential expression analysis between tumor and PDX within CD8+ T cells
cd8_de <- FindMarkers(cd8_cells, ident.1 = "tumor", ident.2 = "PDX", logfc.threshold = 0.25)
cd8_de$gene <- rownames(cd8_de)

# Add column indicating significance for volcano plot
cd8_de$significant <- cd8_de$p_val_adj < 0.05 & abs(cd8_de$avg_log2FC) > 0.25

# Volcano plot
ggplot(cd8_de, aes(x = avg_log2FC, y = -log10(p_val_adj), color = significant)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = c("grey", "red")) +
  theme_minimal() +
  labs(title = "CD8+ T cells: Tumor vs PDX",
       x = "log2 Fold Change",
       y = "-log10 Adjusted p-value") +
  theme(legend.position = "none")
```




```
# Select top 20 DE genes by absolute logFC and p-value
top_genes <- cd8_de %>%
  filter(p_val_adj < 0.05) %>%
  arrange(desc(abs(avg_log2FC))) %>%
  slice_head(n = 20) %>%
  pull(gene)

# Re-scale expression data for the selected genes
cd8_cells <- ScaleData(cd8_cells, features = top_genes)

# Extract scaled expression matrix for heatmap
scaled_expr <- GetAssayData(cd8_cells, slot = "scale.data")[top_genes, ]

# Create annotation for sample group (tumor/PDX)
annotation_col <- data.frame(Group = cd8_cells$group)
rownames(annotation_col) <- colnames(cd8_cells)

# Draw heatmap of top DE genes
pheatmap(scaled_expr,
         annotation_col = annotation_col,
         show_colnames = FALSE,
         cluster_cols = TRUE,
         cluster_rows = TRUE,
         fontsize_row = 8,
         annotation_colors = list(Group = c(tumor = "red", PDX = "orange")))
```



Discussion

I subset the dataset to focus on CD8⁺ T cells, which play a central role in antitumor immunity and were well represented in both tumor and PDX conditions. To investigate context-specific transcriptional changes, I performed differential expression (DE) analysis using Seurat's FindMarkers function (log fold change threshold = 0.25) comparing tumor- versus PDX-derived CD8⁺ T cells. The resulting volcano plot reveals widespread gene expression differences, including a subset of genes strongly upregulated in tumor samples. The accompanying heatmap highlights top DE genes such as FABP4, AHSP, and CRP, which were enriched in tumor-derived cells but nearly absent in PDX. These findings suggest that the tumor microenvironment induces distinct transcriptional programs in CD8⁺ T cells not recapitulated in PDX. This supports prior observations that PDX models often diverge from their source tumors due to murine-specific selective pressures[1].

Reference: 1. Ben-David, U., Ha, G., Tseng, Y. Y., Greenwald, N. F., Oh, C., Shih, J., … & Golub, T. R. (2017). Patient-derived xenografts undergo mouse-specific tumor evolution. Nature genetics, 49(11), 1567-1575.