Lung Cancer Data R-Vignette

Perry Haaland

5/27/2020

1.0 Introduction

The purpose of this R-Vignette is to help make the lung cancer data more easily accessible to someone who wants to analyze it using R. The intended consumer of this document is an experienced statistician or student who already has a working knowledge of R and a basic understanding of genomic statistics. If we are successful, there should be few surprises or difficulties getting started with the data analysis.

All of the explanations and background are taken from Chapter 4.1.1 of the book **Object Oriented Data Analysis** by J. S. Marron and Ian Dryden (OODA). These come from an RNAseq study of lung cancer, and in particular was an early data set collected as part of The Cancer Genome Atlas (John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. Nature genetics, 45(10):1113-1120, 2013).

This vignette can be viewed as a supplement to the **OODA** book that illustrates how to read and organize the data using the R environment. Consequently, we don't repeat the analysis or explanations provided in the book, but rather try to reduce barriers to a new user to get started working with the data to either replicate the OODA analysis or extend it.

Note that we are use the collection of `tidyverse` packages and capabilities for the analysis. We also use the package `factoextra` for visualizing the results of the PCA analysis.

```
library(tidyverse)
library(factoextra)
```

In the Section 2, we read and examine the data sets for each gene. After some reorganization, we combine the data into one data frame that is easy to use in subsequent analysis. In Section 3, we examine summary statistics for the counts, consider the need for a log transformation, and we plot counts against genome position for each gene. In Section 4, we consider the total number of counts by subject and gene with the intent to aid the user in considering whether or not there might be outliers in the data. In Section 5, we illustrate, for a single gene, how to organize the data for a Principal Components Analysis.

2.0 Read and Organize the Data

In this section, we are going to read each of the data files and examine and compare the structures of the resulting data frames. We are also going to do some renaming and reorganizing of the data to facilitate downstream analysis in R. In particular, we construct some new variables and we convert each data frame to a long form that we then combine into one combined data frame.

Early TCGA Data with RNAseq read depth curves are provided for 4 genes; namely,

- PIK3CA

- CDKN2A
- P10
- STK11

There is one file in the `DerivedData` folder for each of the four genes. We are concerned with the `csv` files. Each file contains a matrix of data. The first column is the Genomic Coordinate, which can be considered a row name. The remaining columns are subjects or case names (TCGA IDs). The data entries are read counts. The gene name is included in the name of the file.

2.1 Read and examine the data

We begin by reading each of the gene files into R as separate data frames.

```
cdkn2a_df <- read.csv("../DerivedData/LungCancer2011CDKN2A.csv",
                      header = TRUE) %>%
  as_tibble
p10_df <- read.csv("../DerivedData/LungCancer2011P10.csv",
                      header = TRUE) %>%
  as_tibble
pik3ca_df <- read.csv("../DerivedData/LungCancer2011PIK3CA.csv",
                      header = TRUE) %>%
  as_tibble
stk11_df <- read.csv("../DerivedData/LungCancer2011STK11.csv",
                      header = TRUE) %>%
  as_tibble
```

Next we compare the sizes of the imported data sets.

```
cat("CDNK2A dimensions =",dim(cdkn2a_df), "\n")
```

```
## CDNK2A dimensions = 1709 181
```

```
cat("P10 dimensions =",dim(p10_df), "\n")
```

```
## P10 dimensions = 5546 181
```

```
cat("PIK3CA dimensions =",dim(pik3ca_df), "\n")
```

```
## PIK3CA dimensions = 3707 181
```

```
cat("STK11 dimensions =",dim(stk11_df), "\n")
```

```
## STK11 dimensions = 3276 181
```

The data are organized so that the first column is the Genomic Coordinates and the remaining columns are the cases (observations). We next verify that the cases, that is, column names, are the same for each data frame.

```
all(colnames(cdkn2a_df)==colnames(p10_df))
```

```
## [1] TRUE
```

```
all(colnames(cdkn2a_df)==colnames(pik3ca_df))
```

```
## [1] TRUE
```

```
all(colnames(cdkn2a_df)==colnames(stk11_df))
```

```
## [1] TRUE
```

2.2 Rename and Reorganize the Data

Now that we have verified the cases, we are going to do a bit or reorganizing so that the four data sets can be concatenated into a single data frame. We are going to also do some work to make the data sets a bit easier to work with.

First we note that the case names are rather long and complicated.

```
colnames(cdkn2a_df)[1:5]
```

```
## [1] "Genomic.Coords"
## [2] "preview.20110919.110107_UNC9.SN296_0118_B815C0ABXX7TCGA.66.2756.11A.01R.A"
## [3] "preview.20110919.110107_UNC9.SN296_0118_B815C0ABXX8TCGA.66.2756.11A.01R.V"
## [4] "preview.20110919.110113_UNC10.SN254_0184_B802WAABXX2TCGA.66.2751.01A.02R.V"
## [5] "preview.20110919.110113_UNC10.SN254_0184_B802WAABXX3TCGA.66.2755.01A.02R.A"
```

So we are going to replace them with simpler versions.

```
case_names = paste0("Case", 1:(ncol(cdkn2a_df) - 1))
names(cdkn2a_df)[-1] <- case_names
names(p10_df)[-1]  <- case_names
names(pik3ca_df)[-1]  <- case_names
names(stk11_df)[-1]  <- case_names
```

2.3 Compute new variables

The data values for each gene are read counts at a particular genomic position, `Genomic.Coords`. These values are distinct for each gene. For subsequent analysis of each gene, we are primarily interested in the relative position in the genome for that particular gene. The relative positions are called Exonic NT Numbers. Next we add that variable to each data frame. We are also going to add the gene name to each data frame to facilitate combining the data.

```
cdkn2a_df <- cdkn2a_df %>%
  mutate(Gene.Name = "CDKN2A",
         NT.Number = 1:nrow(cdkn2a_df)) %>%
  select(Gene.Name, Genomic.Coords, NT.Number, starts_with("Case"))
p10_df <- p10_df %>%
```

```
  mutate(Gene.Name = "P10",
         NT.Number = 1:nrow(p10_df)) %>%
  select(Gene.Name, Genomic.Coords, NT.Number, starts_with("Case"))
pik3ca_df <- pik3ca_df %>%
  mutate(Gene.Name = "PIK3CA",
         NT.Number = 1:nrow(pik3ca_df)) %>%
  select(Gene.Name, Genomic.Coords, NT.Number, starts_with("Case"))
stk11_df <- stk11_df %>%
  mutate(Gene.Name = "STK11",
         NT.Number = 1:nrow(stk11_df)) %>%
  select(Gene.Name, Genomic.Coords, NT.Number, starts_with("Case"))
```

2.4 Combine into one data frame

Now we are ready to convert to a long form of each data set and combine them into one data frame.

In order to make the first series of graphs, we first pivot the data table to make the data format more amendable to standard methods in R.

```
long1_df <- cdkn2a_df %>%
  pivot_longer(starts_with("Case"),
               names_to = "Case",
               values_to = "Count")
long2_df <- p10_df %>%
  pivot_longer(starts_with("Case"),
               names_to = "Case",
               values_to = "Count")
long3_df <- pik3ca_df %>%
  pivot_longer(starts_with("Case"),
               names_to = "Case",
               values_to = "Count")
long4_df <- stk11_df %>%
  pivot_longer(starts_with("Case"),
               names_to = "Case",
               values_to = "Count")
long_df <- rbind(long1_df, long2_df, long3_df, long4_df)
head(long_df)
```

```
## # A tibble: 6 x 5
##   Gene.Name Genomic.Coords NT.Number Case  Count
##   <chr>              <int>     <int> <chr> <int>
## 1 CDKN2A          21967752         1 Case1     0
## 2 CDKN2A          21967752         1 Case2     0
## 3 CDKN2A          21967752         1 Case3     0
## 4 CDKN2A          21967752         1 Case4     0
## 5 CDKN2A          21967752         1 Case5     0
## 6 CDKN2A          21967752         1 Case6     0
```

To conclude this section, we examine and verify visually, by comparison to the results above, that the sizes of the data for each gene are correct.

```
long_df %>%
  group_by(Gene.Name) %>%
  summarize(n.coords = length(unique(Genomic.Coords)),
            n.cases = length(unique(Case)))
```

```
## # A tibble: 4 x 3
##   Gene.Name n.coords n.cases
##   <chr>        <int>   <int>
## 1 CDKN2A        1709     180
## 2 P10           5546     180
## 3 PIK3CA        3707     180
## 4 STK11         3276     180
```

3.0 Comparison of Counts Across Genes

In order to be confident that the data is comparable across the genes, we next look at the distribution of counts for each gene.

3.1 Summary statistics

The following result shows values of summary statistics computed on the raw counts for each gene.

```
long_df %>%
  group_by(Gene.Name) %>%
  summarize(length = length(unique(Genomic.Coords)),
         total = sum(Count),
         min = min(Count),
         max = max(Count),
         mean = round(mean(Count), 1),
         median = round(median(Count),1),
         std = round(sqrt(var(Count)),1),
         mad = round(mad(Count),1))
```

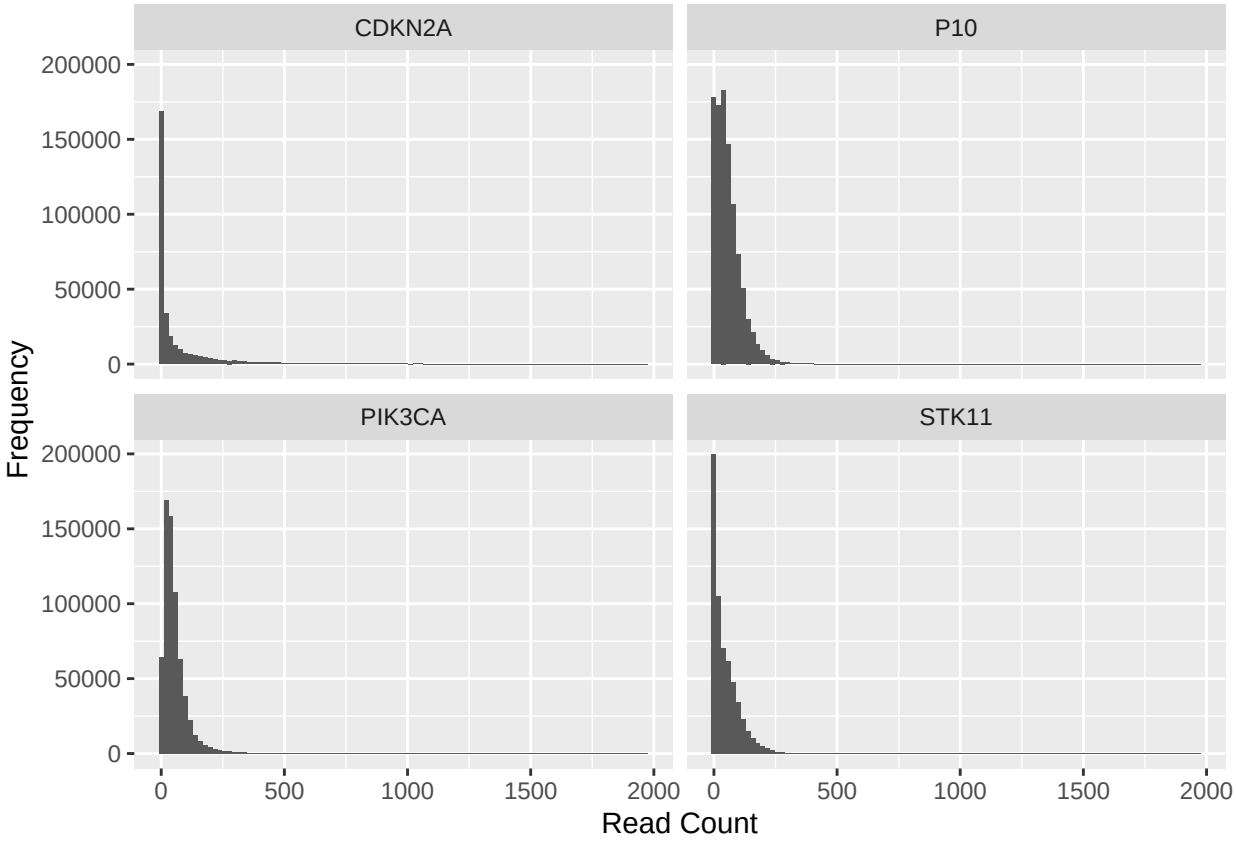
```
## # A tibble: 4 x 9
##   Gene.Name length    total   min   max  mean median   std   mad
##   <chr>      <int>    <int> <int> <int> <dbl>  <dbl> <dbl> <dbl>
## 1 CDKN2A      1709 22575562     0  1966  73.4      6 163     8.9
## 2 P10         5546 56769549     0   587  56.9     45  51.6  46
## 3 PIK3CA      3707 36009398     0   544  54       41  49.7  32.6
## 4 STK11       3276 26988161     0   410  45.8     27  52.6  40
```

Without commenting on the importance of this finding, we note that there are considerable differences in the number of reads (counts) for the different genes in total and also the average counts per position.

3.2 Skewness and log transform

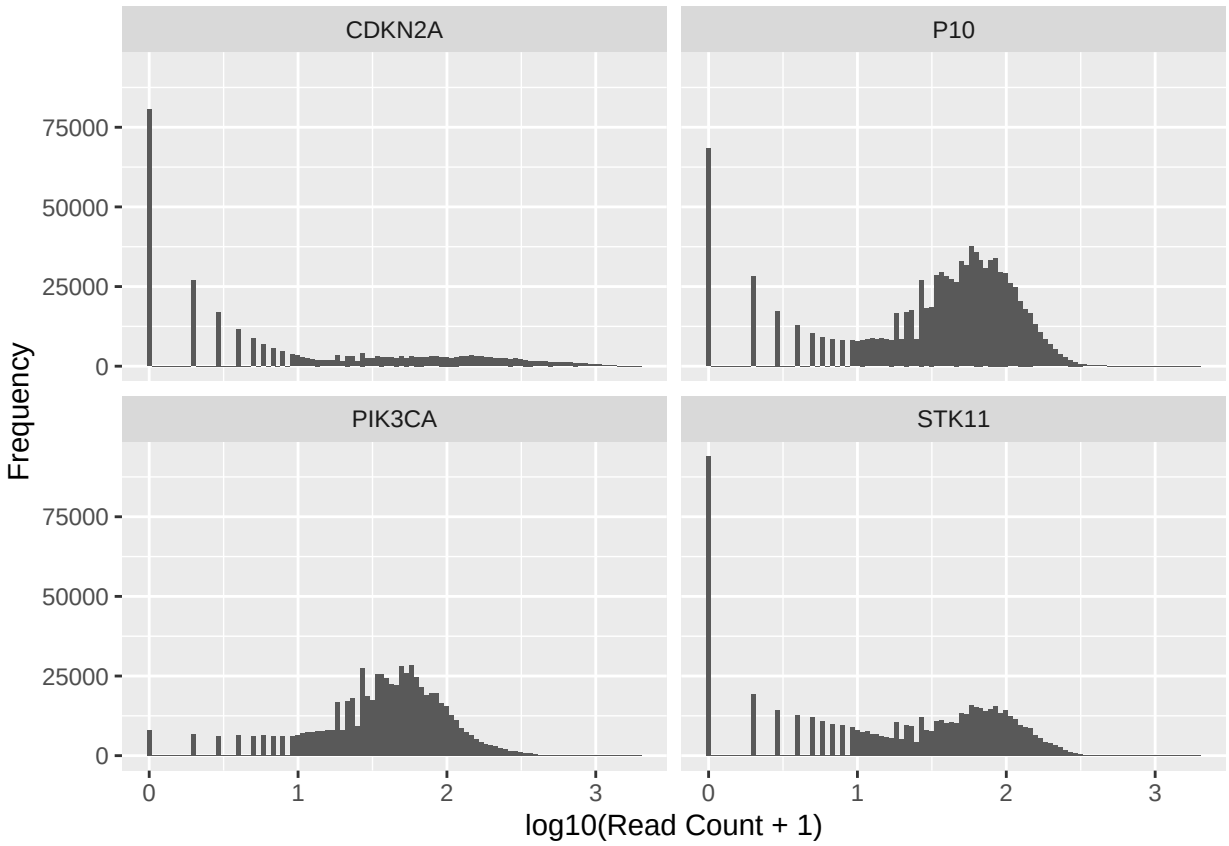
First we note that there is a very long right tail to the counts for each gene.

```
long_df %>%
  ggplot(aes(x = Count)) +
  geom_histogram(bins = 100) +
  facet_wrap(~ Gene.Name) +
  labs(x = "Read Count",
       y = "Frequency")
```



We readily see that most of the counts are 0 or close to 0. For subsequent analysis, we will typically consider the response to be `log10(count + 1)`, which we show below for comparison.

```
long_df %>%
  ggplot(aes(x = log10(Count + 1))) +
  geom_histogram(bins = 100) +
  facet_wrap(~ Gene.Name) +
  labs(x = "log10(Read Count + 1)",
       y = "Frequency")
```



3.3 Summary statistics on transformed data

The following result shows values of summary statistics computed on the log transformed counts for each gene.

```
long_df %>%
  group_by(Gene.Name) %>%
  mutate(logCount = log10(Count + 1)) %>%
  summarize(length = length(unique(Genomic.Coords)),
         min = min(logCount),
         max = max(logCount),
         mean = round(mean(logCount), 1),
         median = round(median(logCount),1),
         std = round(sqrt(var(logCount)),1),
         mad = round(mad(logCount),1))
```

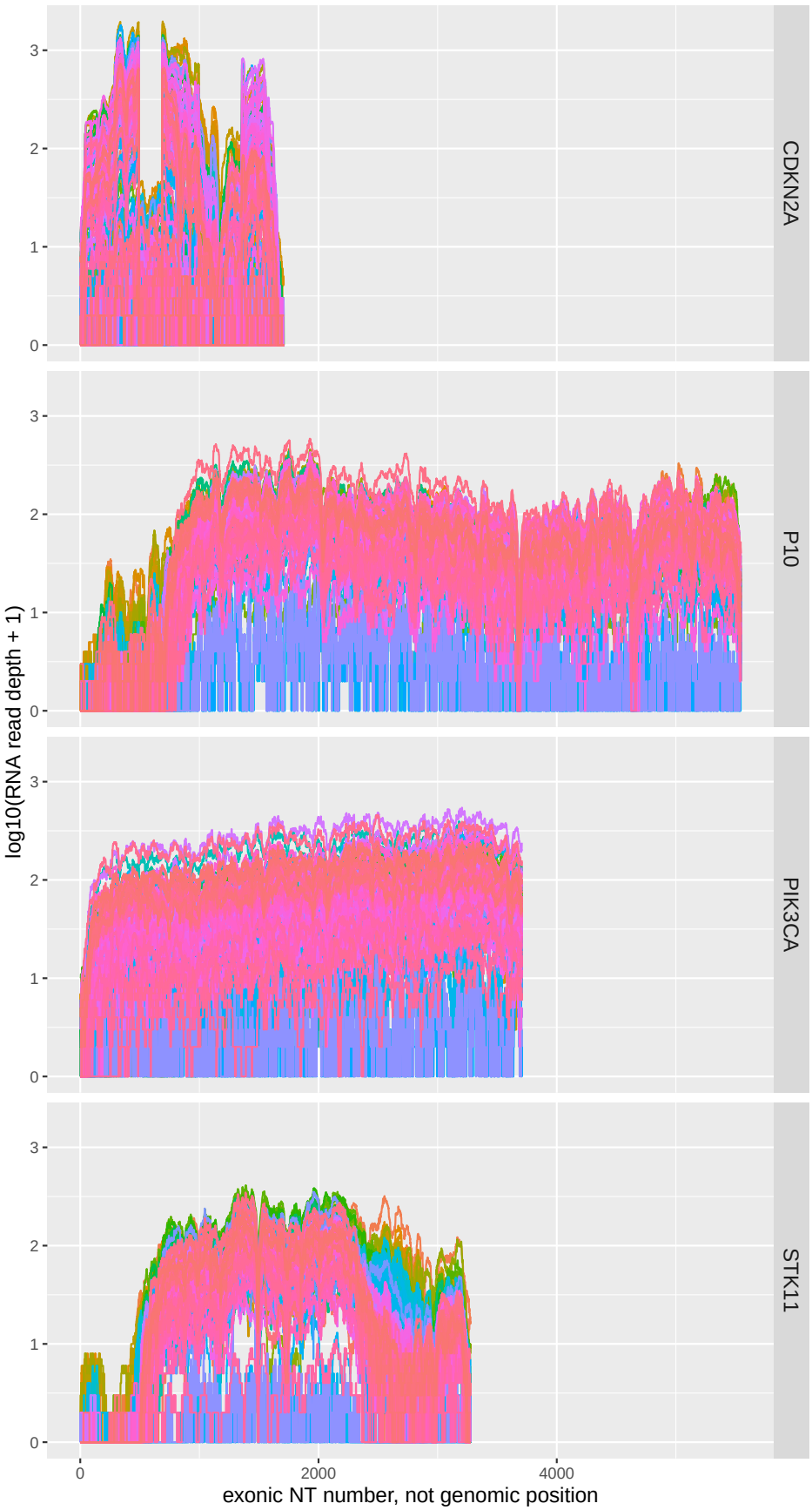
```
## # A tibble: 4 x 8
##   Gene.Name length   min   max  mean median   std   mad
##   <chr>      <int> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 CDKN2A      1709     0  3.29   1      0.8   0.9   1.3
## 2 P10         5546     0  2.77   1.5    1.7   0.6   0.4
## 3 PIK3CA      3707     0  2.74   1.6    1.6   0.4   0.4
## 4 STK11       3276     0  2.61   1.3    1.4   0.7   0.7
```

We note that the mean and median values for each gene are now much closer.

3.4 Plots of read counts versus position

We conclude this section with a plot of read counts versus position within each gene. We show each case with a different line. The y value, `Count` has been transformed by log10(y+1). The line colors are recycled and don't have any significance. We note that there are some regions of the genome that have very few or no reads. We are also reminded the the four genes have quite different lengths.

```
long_df %>%
  ggplot(aes(x= NT.Number, y = log10(Count+1), group = Case)) +
           geom_line(aes(color = Case)) +
  facet_grid(rows = vars(Gene.Name)) +
  theme(legend.position = "none") +
  labs(x = "exonic NT number, not genomic position",
       y = "log10(RNA read depth + 1)") +
  theme(axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        strip.text = element_text(size = 12))
```

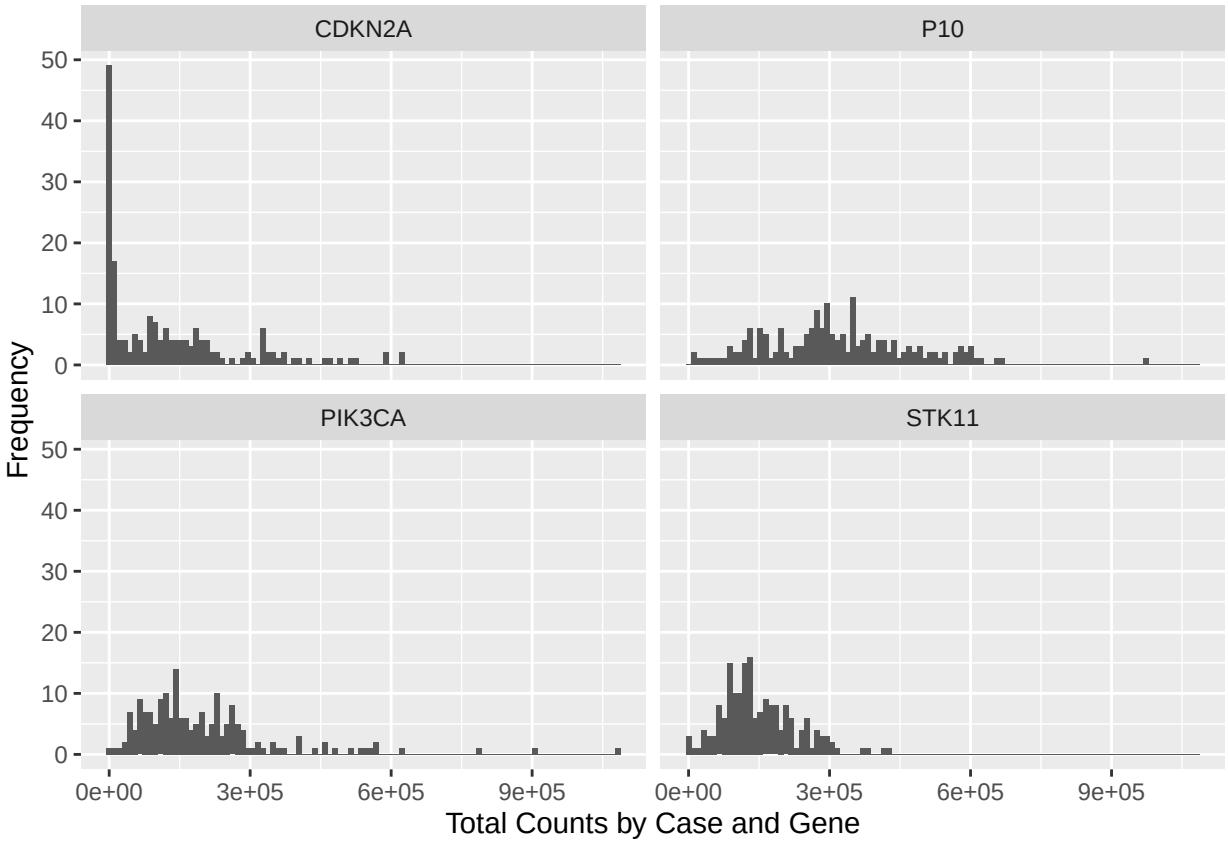

CDKN2A
P10
PIK3CA
STK11
log10(RNA read depth + 1)
0
1
2
3
0
2000
4000
exonic NT number, not genomic position

4.0 Consideration of Possible Outliers

In the figure above, we note that there are some curves that are very close to zero counts across each of the genes. This is itself is not a cause for concern as any of these genes could be expressed at low levels or very high levels. Without the intent of making a judgement about whether any particular cases are outliers, which we leave to the reader, we provide some sample computations to check for difference in read counts across subjects (`Cases`).

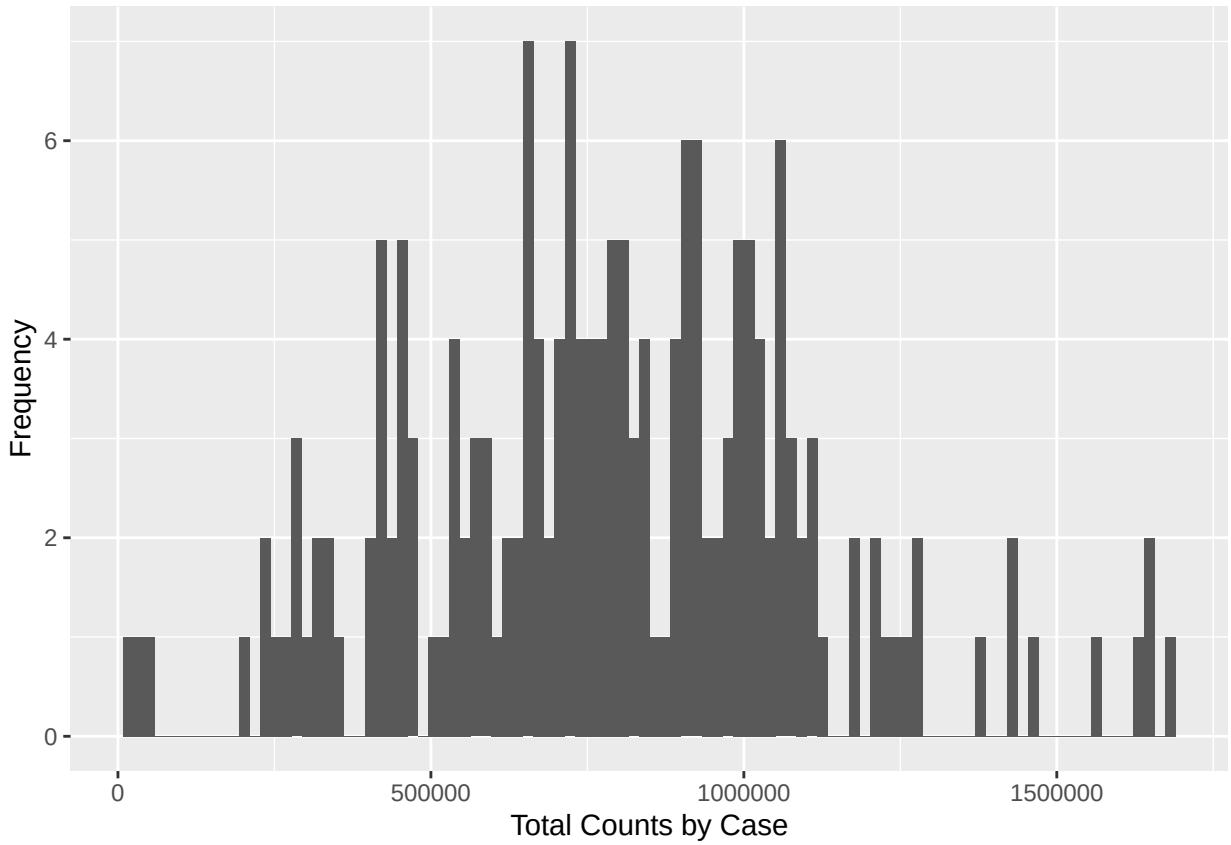
The figure below shows the total counts on a per case basis for each gene.

```
long_df %>%
  group_by(Gene.Name, Case) %>%
  summarize(total = sum(Count)) %>%
  ggplot(aes(x = total)) +
    geom_histogram(bins = 100) +
    facet_wrap(~ Gene.Name) +
    labs(x = "Total Counts by Case and Gene",
       y = "Frequency")
```



The figure below shows the total counts by `Case` across all genes.

```
long_df %>%
  group_by(Case) %>%
  summarize(total = sum(Count)) %>%
  ggplot(aes(x = total)) +
    geom_histogram(bins = 100) +
    labs(x = "Total Counts by Case",
       y = "Frequency")
```



From this point, the interested reader could invesigate further the cases at either ends of the extremes for total counts.

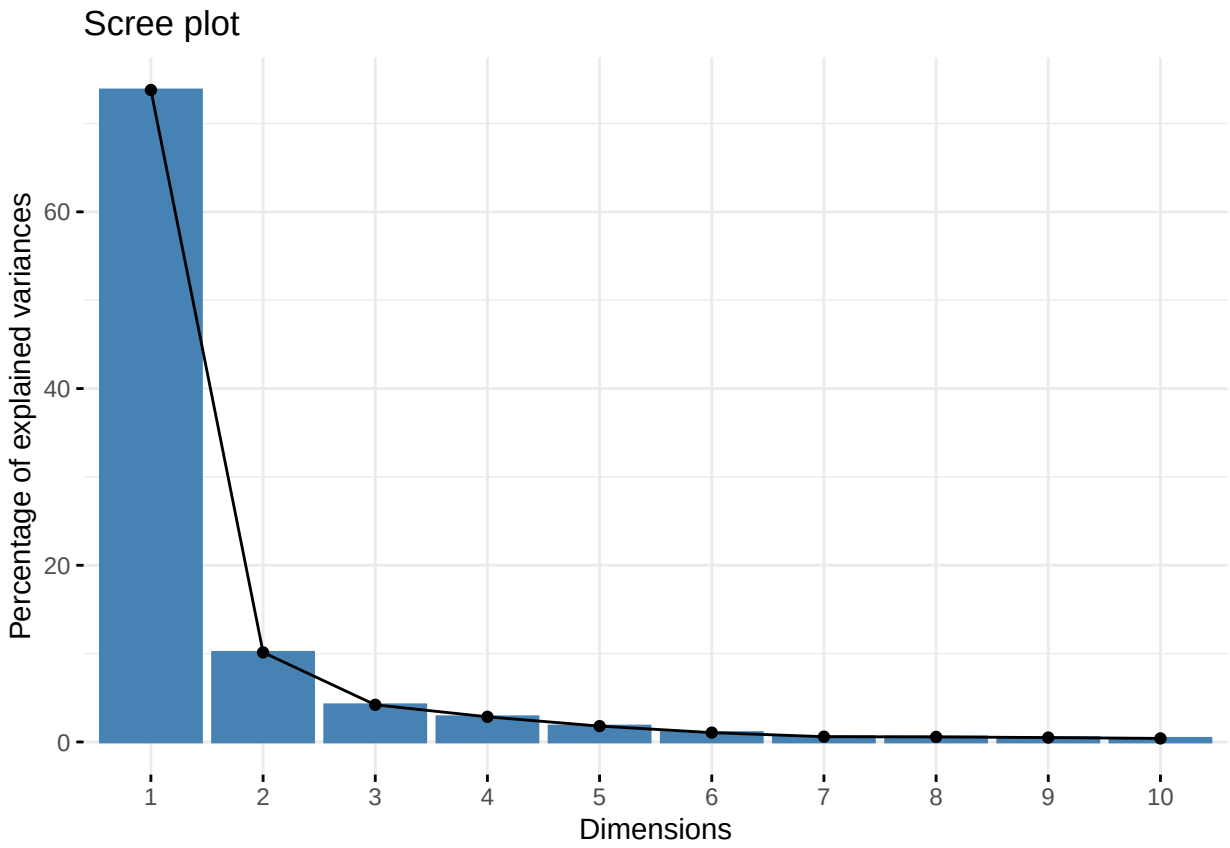
5.0 Principal Components Analysis

We conclude this vignette, with a example of a Principal Components Analysis (PCA) for one of the genes. In particular, we focus on the gene CDKN2A, which has long been known to be involved in many types of cancer. To do the PCA, we reformat the data for the gene so that the cases are in rows.

```
rows_df <- cdkn2a_df %>%
  select(-Gene.Name, -Genomic.Coords, -NT.Number) %>%
  t()
colnames(rows_df) <- cdkn2a_df$NT.Number
```

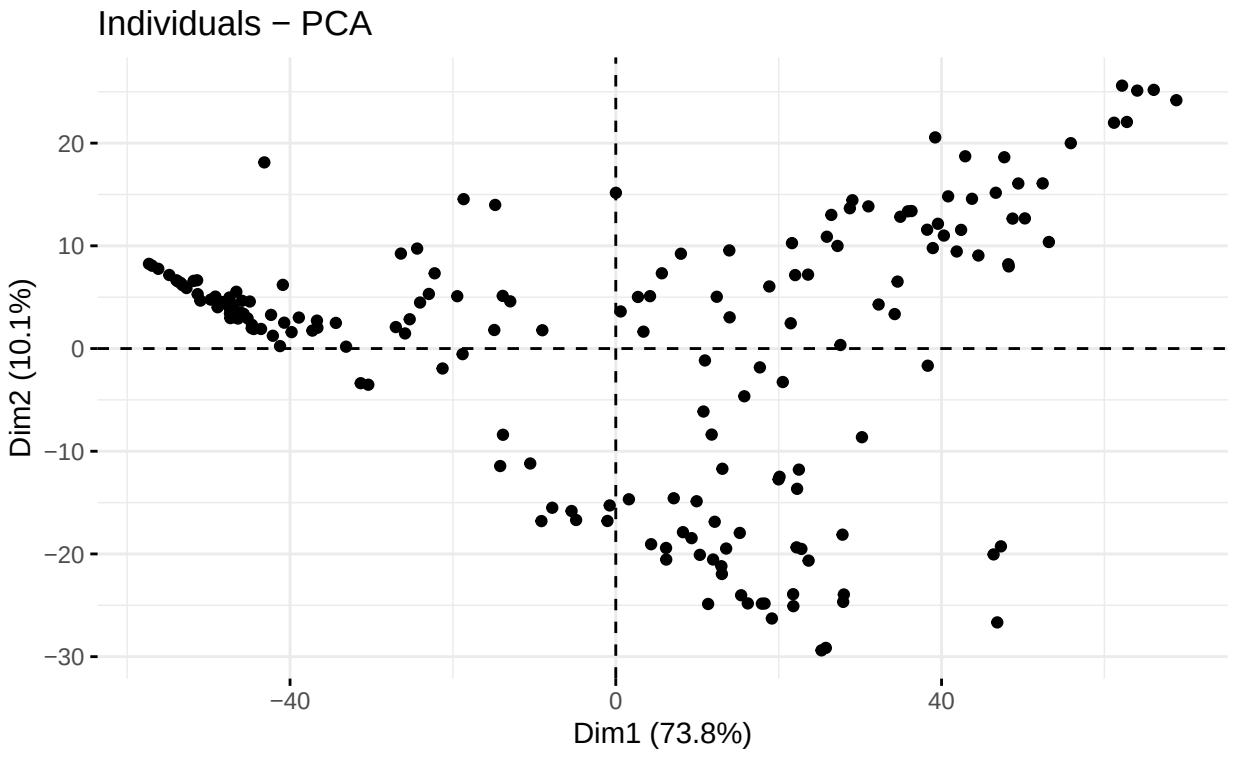
Now we are ready to do the PCA. Note that we standardize the counts at each position as part of the PCA. The first visualization is the standard scree plot. Note that we use the log transformation as above.

```
pca.out <- prcomp(log10(rows_df + 1), scale = TRUE)
fviz_eig(pca.out)
```

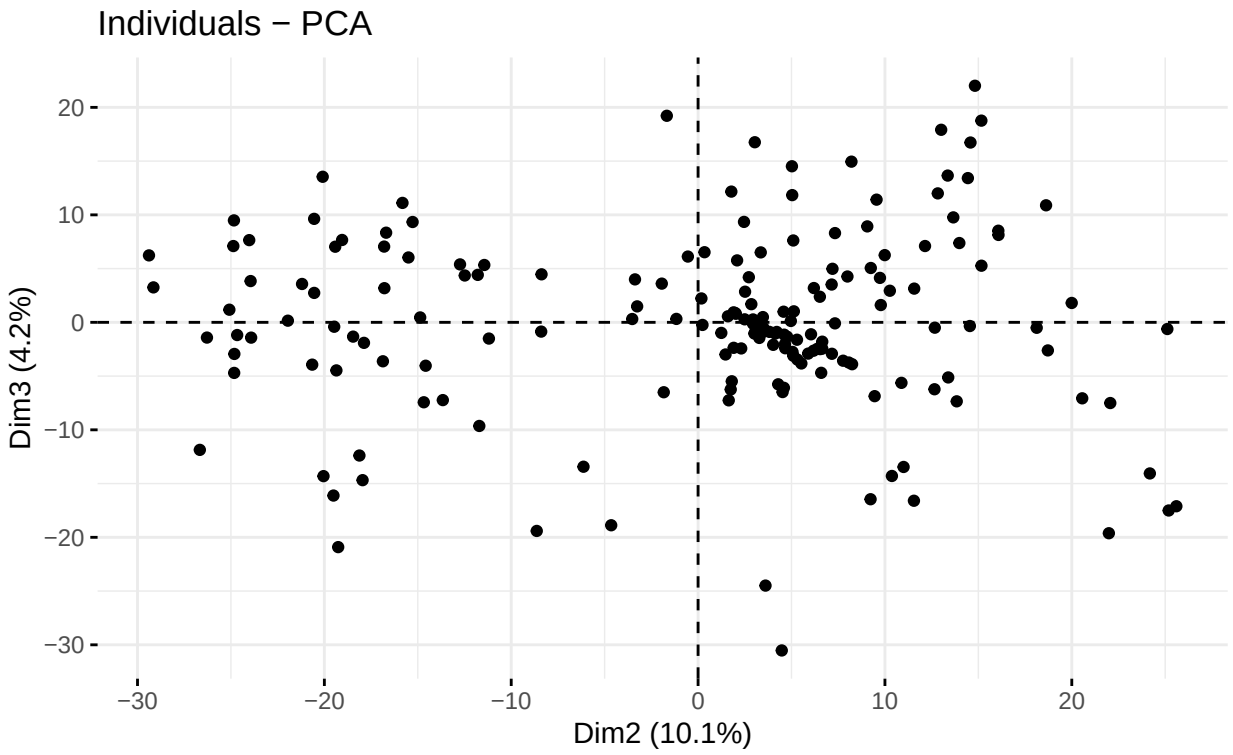


Next we look at the scatter plot of the first two dimensions.

```
fviz_pca_ind(pca.out,
             axes = c(1, 2),
             geom = "point")
```




```
fviz_pca_ind(pca.out,
             axes = c(2, 3),
             geom = "point")
```



5.0 Conclusion

In this vignette, we demonstrated how to the TCGA read count data for each gene. We examined the structure of each data file. We reviewed summary statistics. We noted the use of a log transformation to adjust for skewness. We did a brief examination of the patterns of counts across read positions in each gene. We showed some computations that could be used to evaluate the data for the presence of outliers. Finally, we demonstrated the application of PCA to one of the genes.

Hopefully, we accomplished our purpose of paving the way for an experienced statistician and user of R to quickly get up to speed on using the data set. We remind the reader that Chapter 4.1 of the **OODA** book provides a detailed presentation of the exporatory analysis of this data. A further discussion on confirmatory analysis can be found in Chapter 14.