Project 1

Rohan Patel

2023-11-21

Importing the data

```
setwd("..")
data <- read_csv('Data/Exasens.csv')
```

```
## New names:
## Rows: 401 Columns: 13
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," chr
## (8): Diagnosis, ID, Imaginary Part, ...4, Real Part, ...6, ...12, ...13 dbl
## (3): Gender, Age, Smoking lgl (2): ...10, ...11
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...4`
## • `` -> `...6`
## • `` -> `...10`
## • `` -> `...11`
## • `` -> `...12`
## • `` -> `...13`
```

```
data <- data[-c(1, 2), ] #Removing Repeating and blank Headers
data <- data[, 1:9] #Removing Extra columns at the end
data <- data[,-2] #Removing ID Column
data <- data %>% setNames(c('Diagnosis', 'Imaginary Part: Min', 'Imaginary Part: Avg', 'Real
Part: Min', 'Real Part: Avg', 'Gender', 'Age', 'Smoking'))
data <- data[, c(2:length(data), 1)]
head(data)
```

```
## # A tibble: 6 × 8
##   `Imaginary Part: Min` `Imaginary Part: Avg` `Real Part: Min` `Real Part: Avg`
##   <chr>                 <chr>                 <chr>            <chr>
## 1 -320.61               -300.5635307          -495.26          -464.1719907
## 2 -325.39               -314.7503595          -473.73          -469.2631404
## 3 -323                  -317.4360556          -476.12          -471.8976667
## 4 -327.78               -317.3996698          -473.73          -468.856388
## 5 -325.39               -316.1557853          -478.52          -472.8697828
## 6 -327.78               -318.6775535          -507.23          -469.0241943
## # ℹ 4 more variables: Gender <dbl>, Age <dbl>, Smoking <dbl>, Diagnosis <chr>
```

Data Cleaning

Data Cleaning and Balancing

```
str(data)
```

```
## tibble [399 × 8] (S3: tbl_df/tbl/data.frame)
##  $ Imaginary Part: Min: chr [1:399] "-320.61" "-325.39" "-323" "-327.78" ...
##  $ Imaginary Part: Avg: chr [1:399] "-300.5635307" "-314.7503595" "-317.4360556" "-317.399
6698" ...
##  $ Real Part: Min     : chr [1:399] "-495.26" "-473.73" "-476.12" "-473.73" ...
##  $ Real Part: Avg     : chr [1:399] "-464.1719907" "-469.2631404" "-471.8976667" "-468.856
388" ...
##  $ Gender             : num [1:399] 1 0 1 1 0 1 1 1 1 1 ...
##  $ Age                : num [1:399] 77 72 73 76 65 60 76 77 74 67 ...
##  $ Smoking            : num [1:399] 2 2 3 2 2 2 2 2 2 2 ...
##  $ Diagnosis          : chr [1:399] "COPD" "COPD" "COPD" "COPD" ...
```

```
data[c('Imaginary Part: Min', 'Imaginary Part: Avg', 'Real Part: Min', 'Real Part: Avg')] <-
lapply(data[c('Imaginary Part: Min', 'Imaginary Part: Avg', 'Real Part: Min', 'Real Part: Av
g')], as.numeric)
str(data)
```

```
## tibble [399 × 8] (S3: tbl_df/tbl/data.frame)
##  $ Imaginary Part: Min: num [1:399] -321 -325 -323 -328 -325 ...
##  $ Imaginary Part: Avg: num [1:399] -301 -315 -317 -317 -316 ...
##  $ Real Part: Min     : num [1:399] -495 -474 -476 -474 -479 ...
##  $ Real Part: Avg     : num [1:399] -464 -469 -472 -469 -473 ...
##  $ Gender             : num [1:399] 1 0 1 1 0 1 1 1 1 1 ...
##  $ Age                : num [1:399] 77 72 73 76 65 60 76 77 74 67 ...
##  $ Smoking            : num [1:399] 2 2 3 2 2 2 2 2 2 2 ...
##  $ Diagnosis          : chr [1:399] "COPD" "COPD" "COPD" "COPD" ...
```

```
summary(data)
```

```
##  Imaginary Part: Min Imaginary Part: Avg Real Part: Min   Real Part: Avg
##  Min.   :-337.4      Min.   :-328.3      Min.   :-626.9   Min.   :-473.9
##  1st Qu.:-327.8      1st Qu.:-318.5      1st Qu.:-476.1   1st Qu.:-470.0
##  Median :-323.0      Median :-314.3      Median :-473.7   Median :-467.1
##  Mean   :-314.9      Mean   :-304.8      Mean   :-473.0   Mean   :-458.7
##  3rd Qu.:-320.6      3rd Qu.:-305.1      3rd Qu.:-468.9   3rd Qu.:-462.8
##  Max.   :-225.0      Max.   :-225.0      Max.   : -44.0   Max.   : -44.0
##  NA's   :299         NA's   :299         NA's   :299      NA's   :299
##      Gender            Age           Smoking       Diagnosis
##  Min.   :0.0000   Min.   :17.00   Min.   :1.000   Length:399
##  1st Qu.:0.0000   1st Qu.:31.00   1st Qu.:1.000   Class :character
##  Median :0.0000   Median :49.00   Median :2.000   Mode  :character
##  Mean   :0.3985   Mean   :48.74   Mean   :1.727
##  3rd Qu.:1.0000   3rd Qu.:64.00   3rd Qu.:2.000
##  Max.   :1.0000   Max.   :93.00   Max.   :3.000
##
```

What is this Imaginary Part/Real Part?

```
data <- na.omit(data)
```

Checking for class imbalance

```
table(data$Diagnosis)
```

```
##
##   Asthma     COPD       HC Infected
##       10       40       40       10
```

Label Encoding Diagnosis

```
data$Diagnosis <- factor(data$Diagnosis, levels = c("Asthma", "COPD", "HC", "Infected"), labe
ls = 0:3)
```

Saving Cleaned data

```
setwd("..")
write.csv(data, 'Data/cleaned_data.csv', row.names = FALSE)
```

EDA

Importing the Clean Data

```
setwd("..")
data <- read_csv('Data/cleaned_data.csv')
```

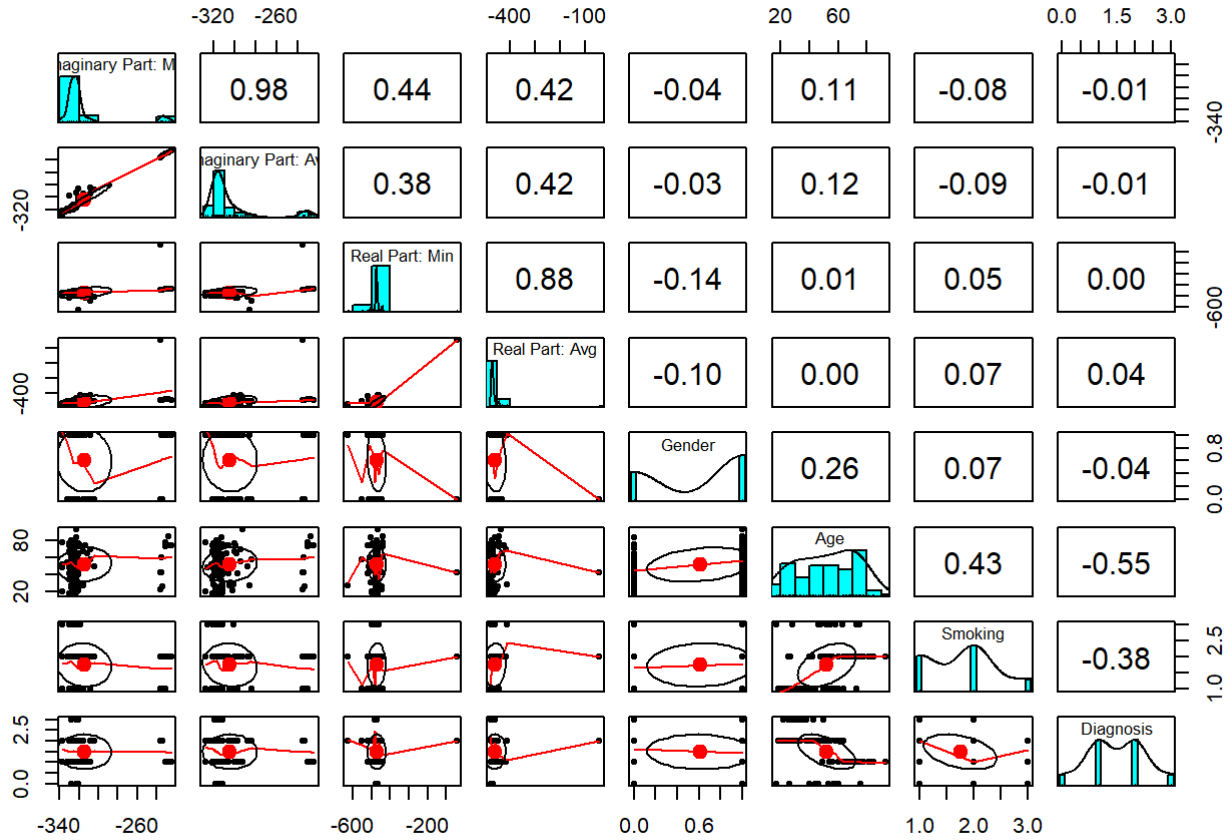
```
## Rows: 100 Columns: 8
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (8): Imaginary Part: Min, Imaginary Part: Avg, Real Part: Min, Real Part...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(data)
```

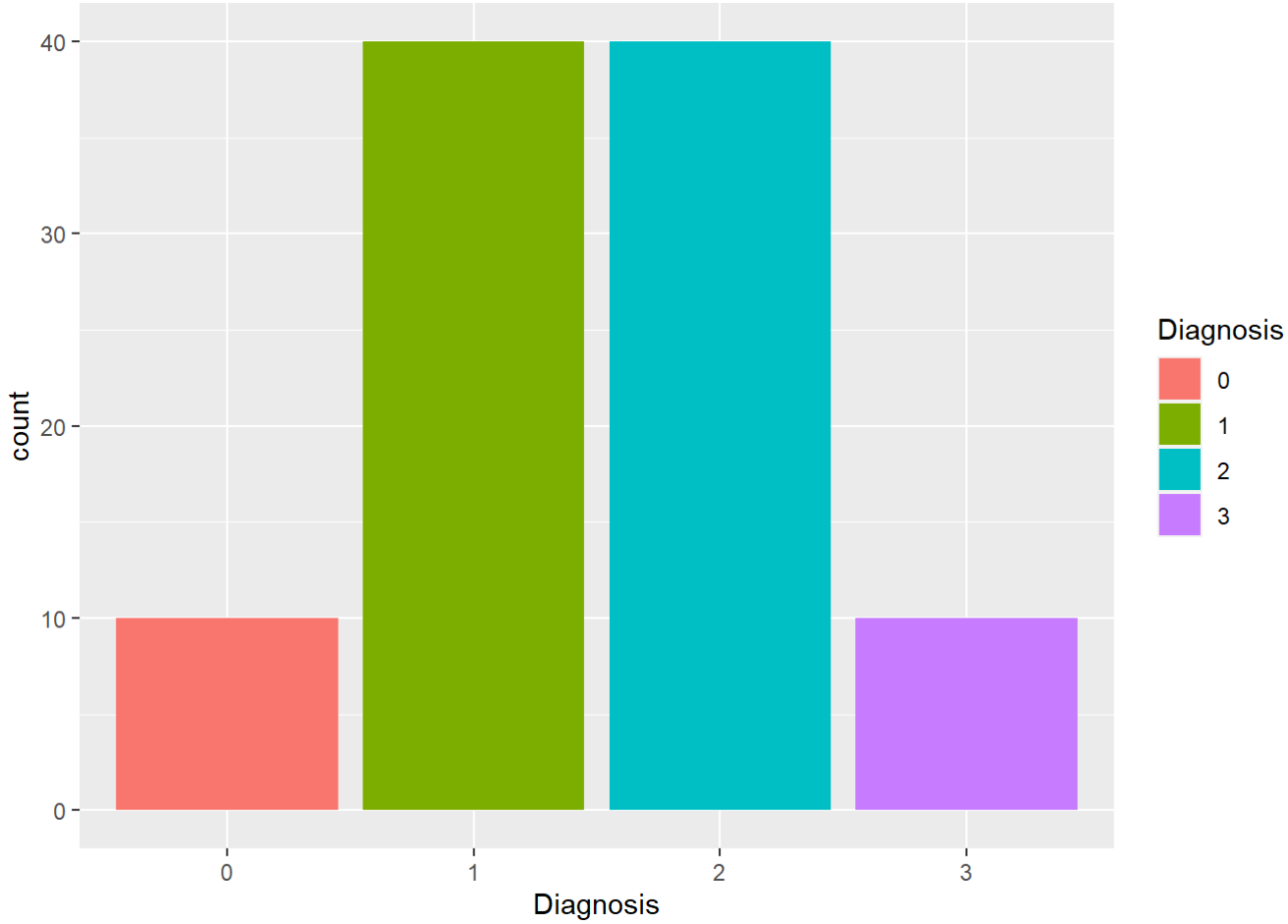
```
## # A tibble: 6 × 8
##   `Imaginary Part: Min` `Imaginary Part: Avg` `Real Part: Min` `Real Part: Avg`
##                   <dbl>                 <dbl>            <dbl>            <dbl>
## 1                 -321.                 -301.            -495.            -464.
## 2                 -325.                 -315.            -474.            -469.
## 3                 -323                  -317.            -476.            -472.
## 4                 -328.                 -317.            -474.            -469.
## 5                 -325.                 -316.            -479.            -473.
## 6                 -328.                 -319.            -507.            -469.
## # ℹ 4 more variables: Gender <dbl>, Age <dbl>, Smoking <dbl>, Diagnosis <dbl>
```

Exploring the Data

```
pairs.panels(data)
```

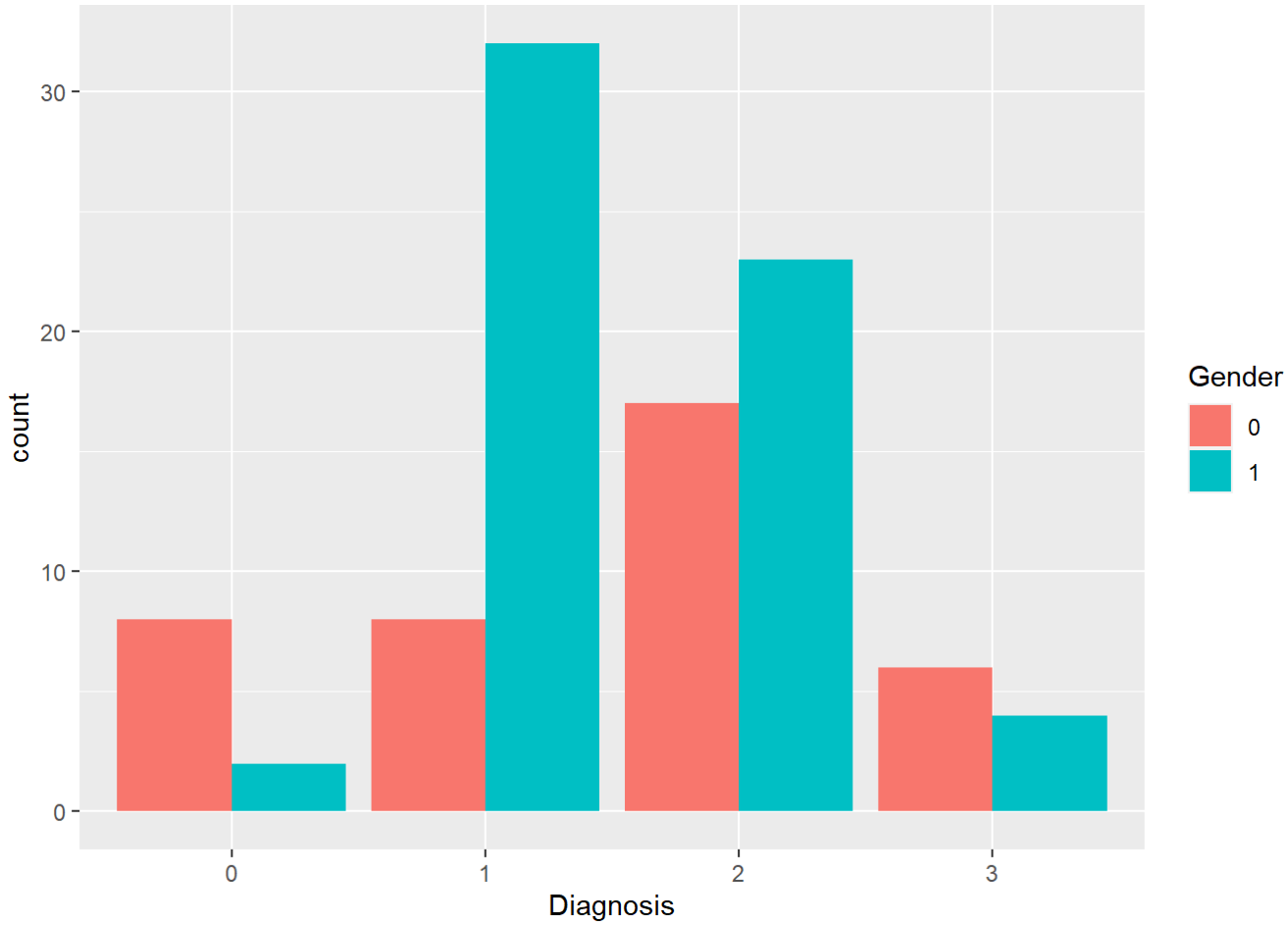



```
ggplot(data, aes(x = as.factor(Diagnosis), fill = as.factor(Diagnosis))) +
  geom_bar() +
  labs("Countplot of Diagnosis", x = "Diagnosis") +
  scale_fill_discrete(name = "Diagnosis")
```



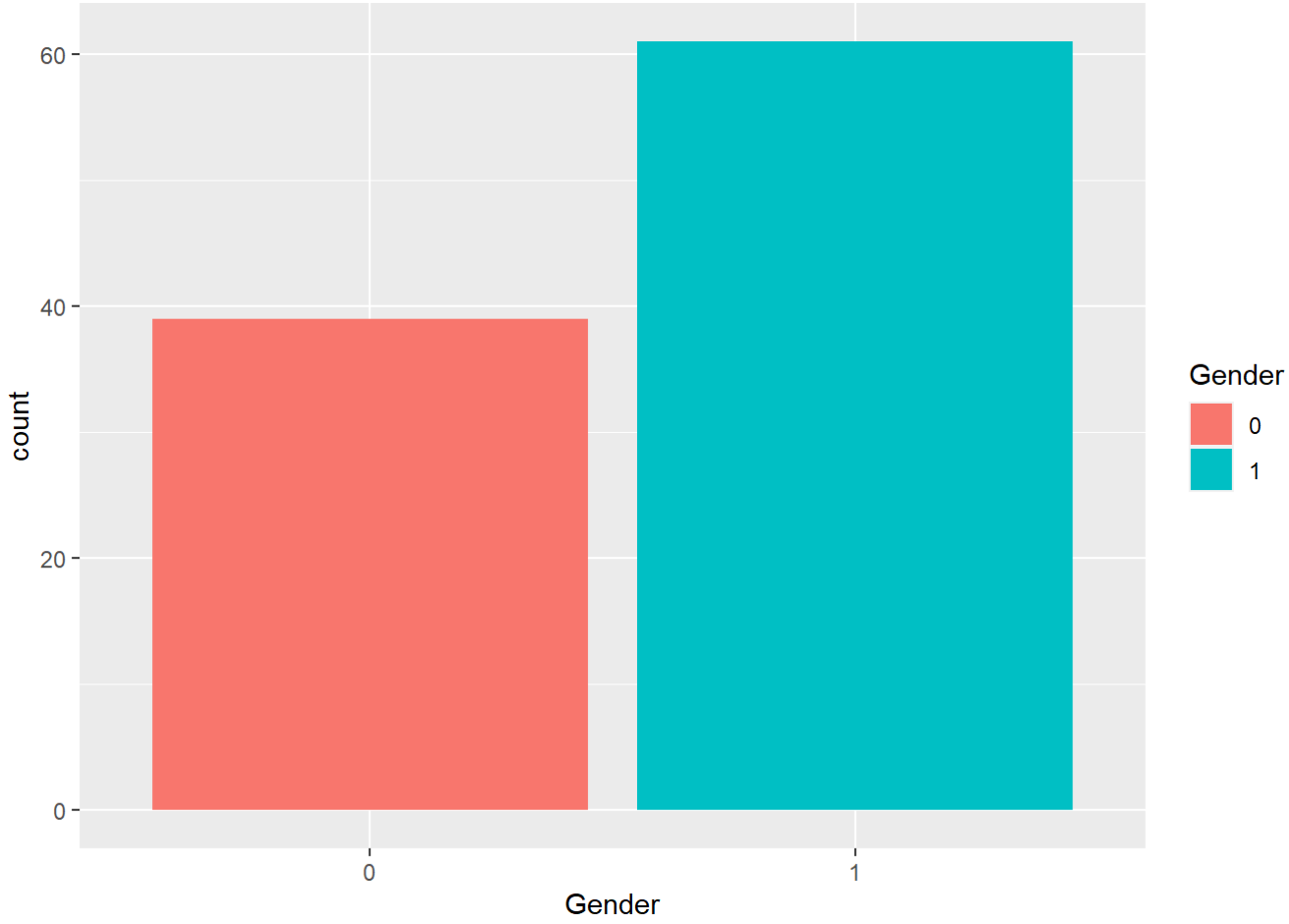
Majority of patients have COPD or are parts of Healthy control groups, data is imbalanced

```
ggplot(data, aes(x = as.factor(Diagnosis), fill = as.factor(Gender))) +
  geom_bar(position = "dodge") +
  labs("Countplot of Diagnosis with Hue Gender", x = "Diagnosis") +
  scale_fill_discrete(name = "Gender")
```



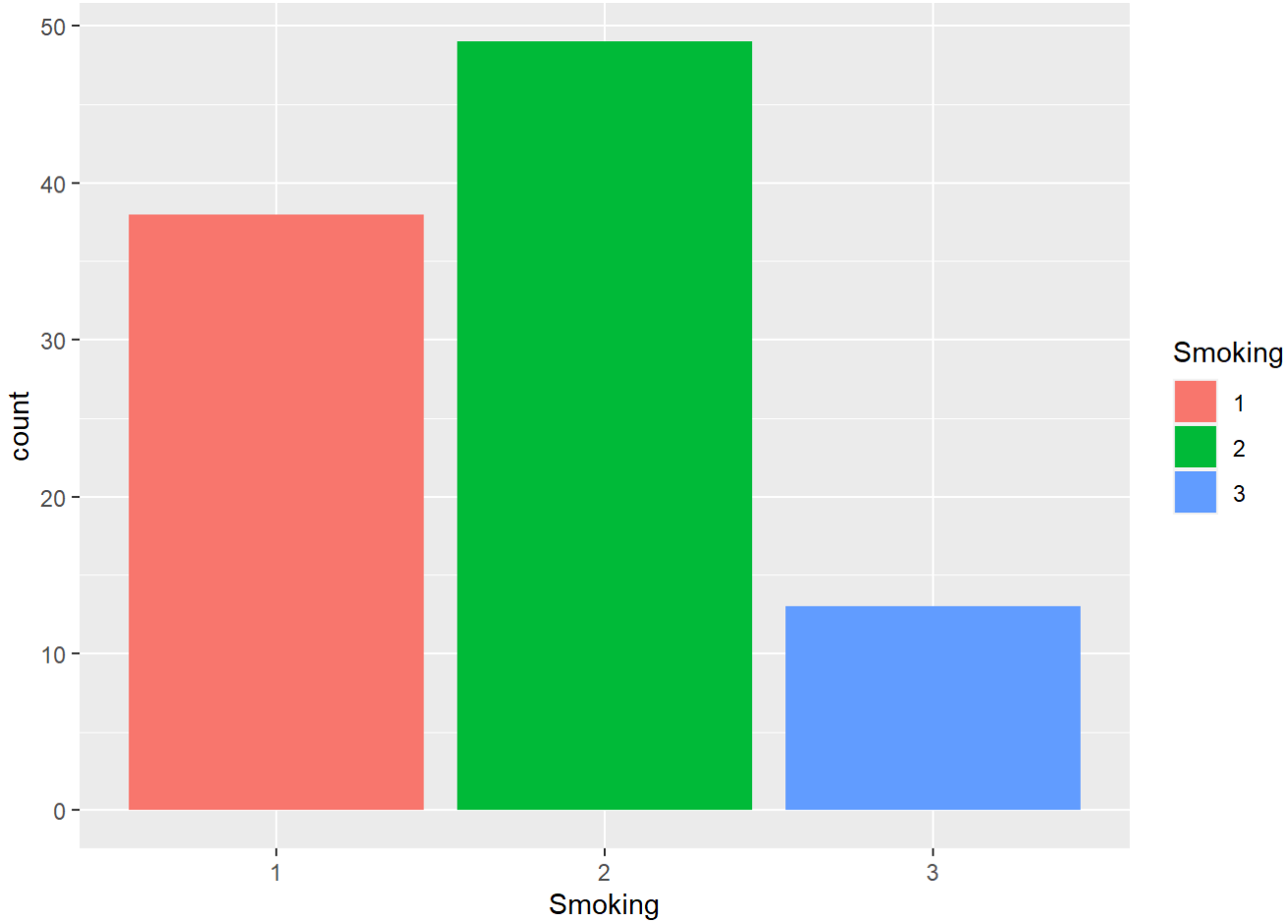
More Female patients have Asthma and Infections then males

```
ggplot(data, aes(x = as.factor(Gender), fill = as.factor(Gender))) +
  geom_bar() +
  labs("Countplot of Gender", x = "Gender") +
  scale_fill_discrete(name = "Gender")
```



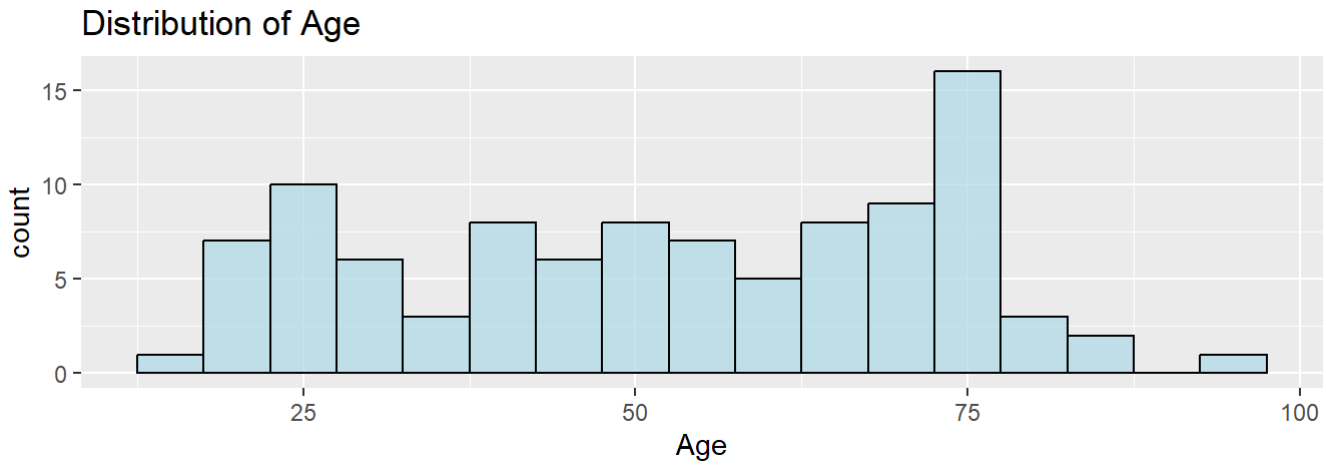
As always majority of patients are males

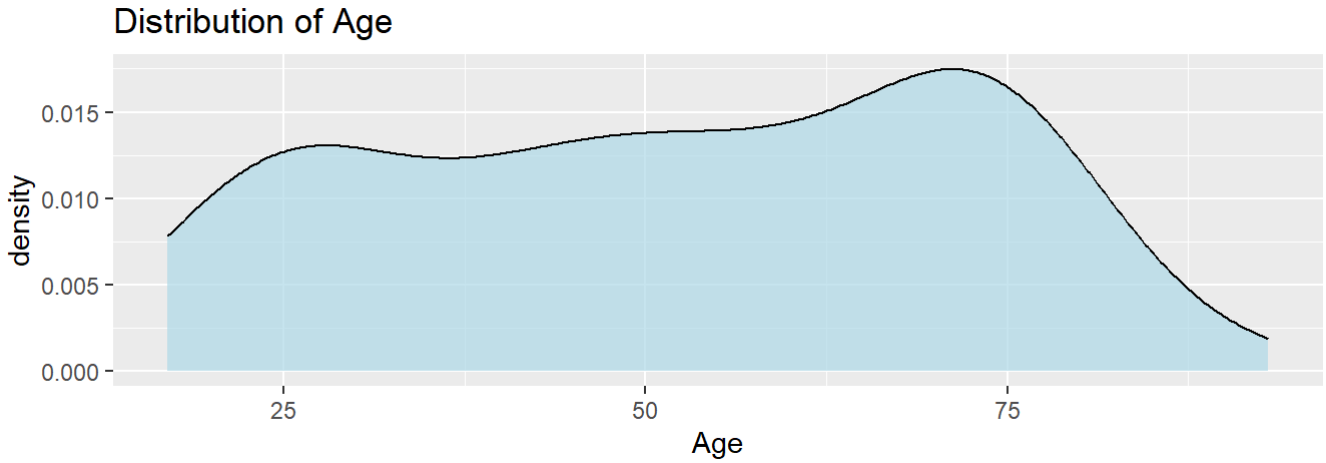
```
ggplot(data, aes(x = as.factor(Smoking), fill = as.factor(Smoking))) +
  geom_bar() +
  labs("Countplot of Smoking", x = "Smoking") +
  scale_fill_discrete(name = "Smoking")
```



Most patients are ex-smokers and non smokers, a few active smokers.

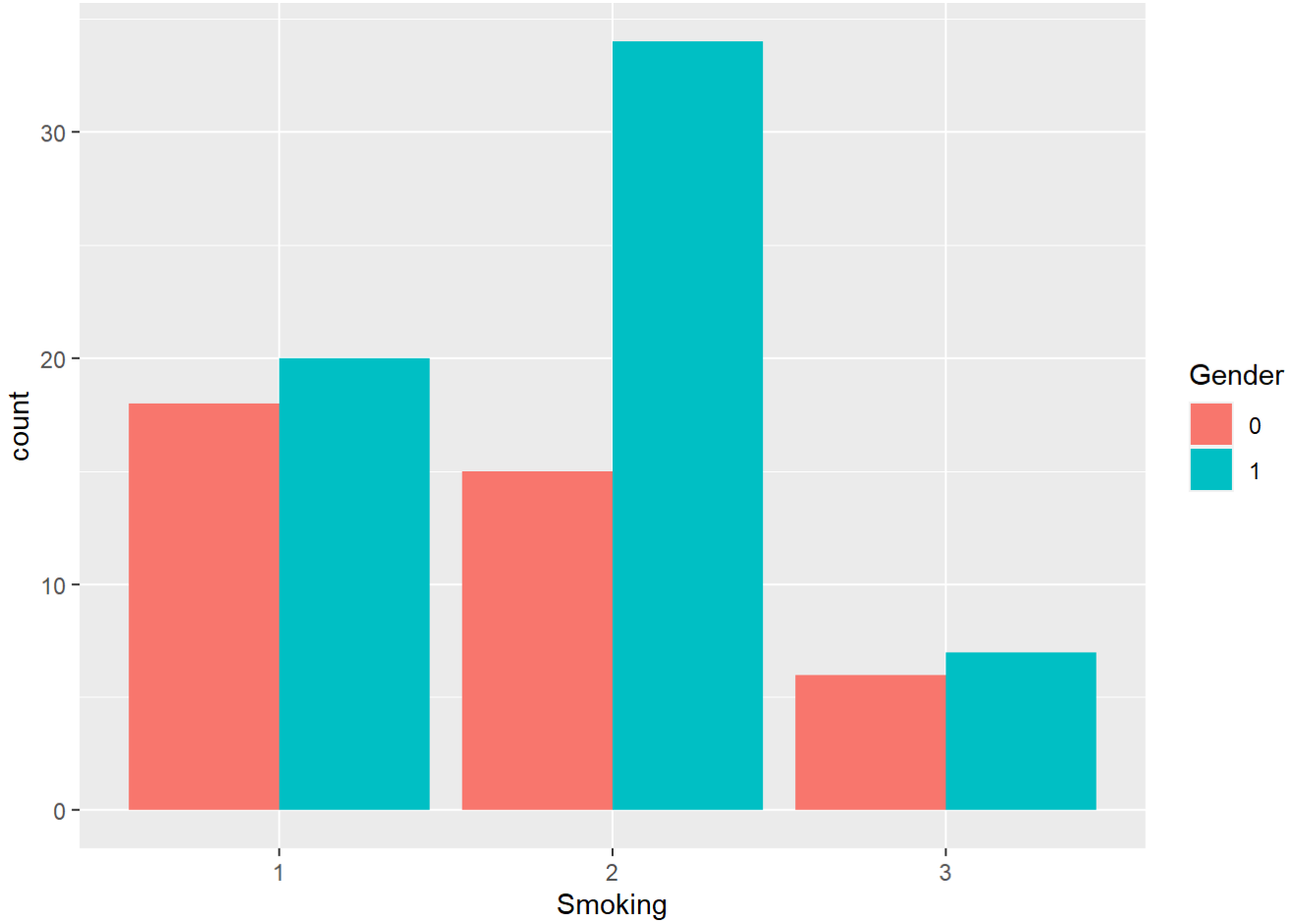
```
plot1 = ggplot(data, aes(x = Age)) +
  geom_histogram(binwidth = 5, fill = "lightblue", color = "black", alpha = 0.7) +
  ggtitle("Distribution of Age") +
  xlab("Age")

plot2 = ggplot(data, aes(x = Age)) +
  geom_density(fill = "lightblue", color = "black", alpha = 0.7) +
  ggtitle("Distribution of Age") +
  xlab("Age")

grid.arrange(plot1, plot2, ncol = 1)
```



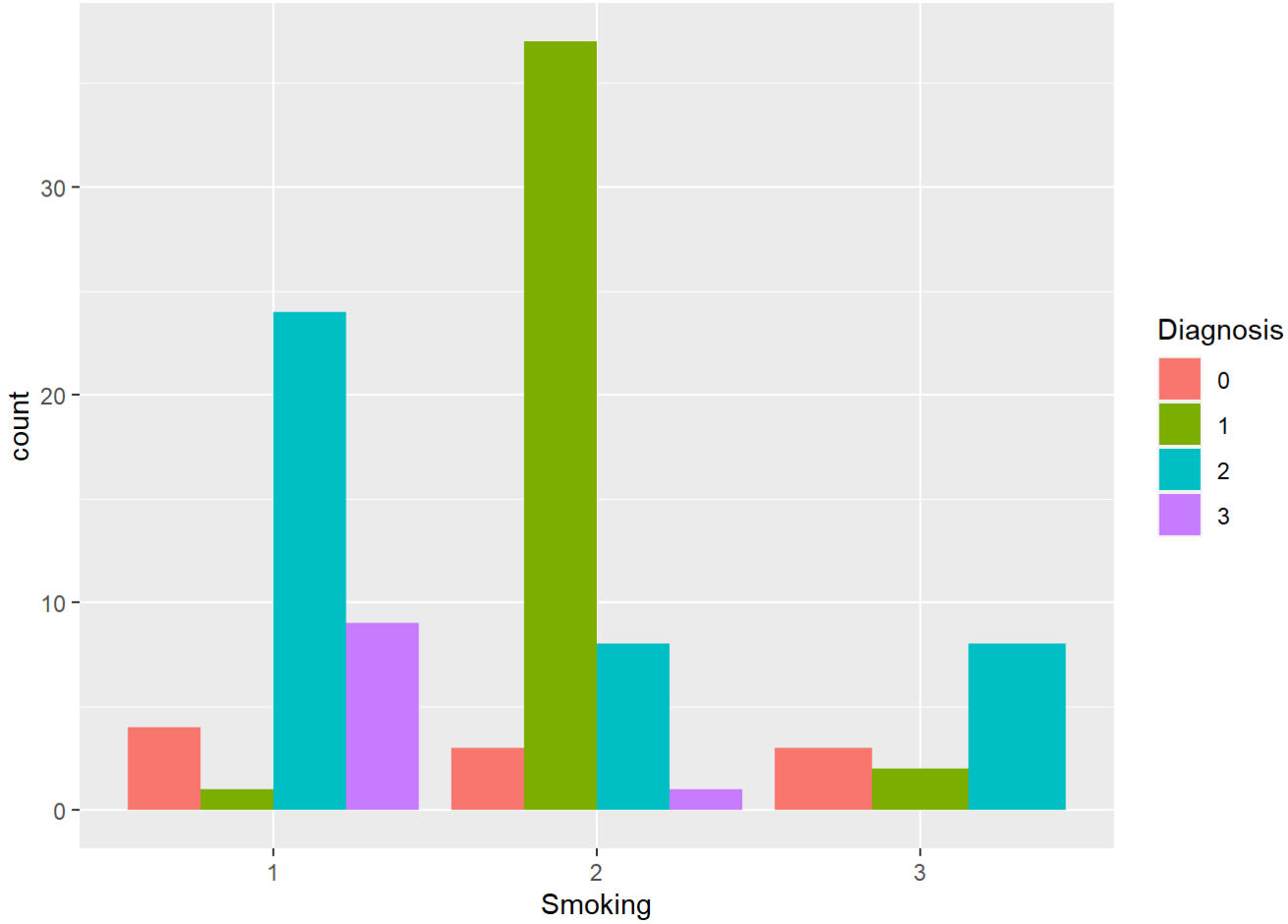


All patients are adults, ranging from young adults in the 20's to senior citizens upto their late seventies, very few paitents in their eighties and beyond

```
ggplot(data, aes(x = as.factor(Smoking), fill = as.factor(Gender))) +
  geom_bar(position = "dodge") +
  labs("Countplot of Smoking with Hue Gender", x = "Smoking") +
  scale_fill_discrete(name = "Gender")
```




```
# Disease by non smokers
ggplot(data, aes(x = as.factor(Smoking), fill = as.factor(Diagnosis))) +
  geom_bar(position = "dodge") +
  labs("Countplot of Smoking with Hue Diagnosis", x = "Smoking") +
  scale_fill_discrete(name = "Diagnosis")
```

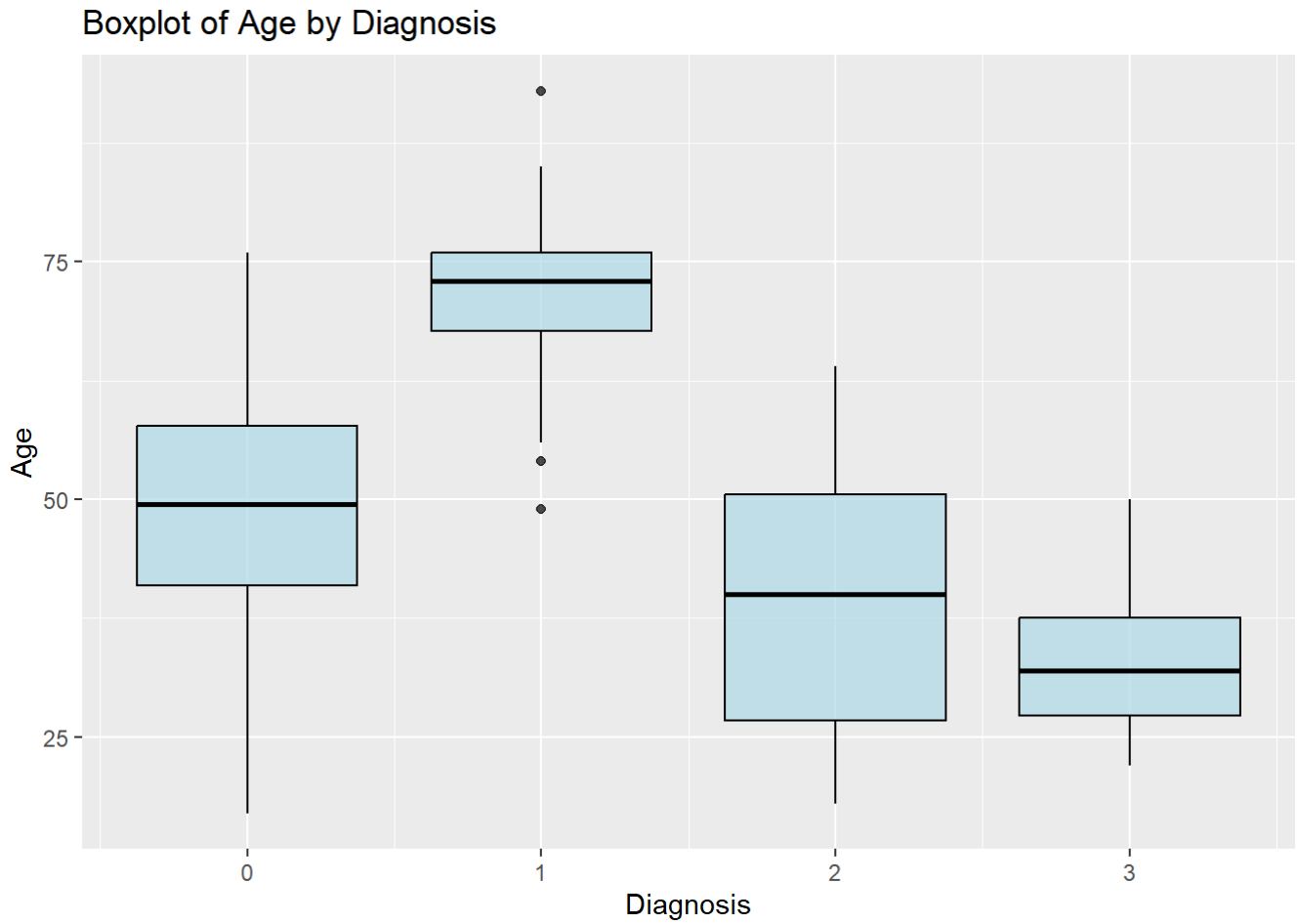


Non-Smokers: most are healthy, rest have infections and asthma, very few with COPD

Ex-Smokers: Majority have COPD.

Active Smokers: Not cases with infections, most are healthy with few having asthama and COPD

```
# Disease by non smokers
ggplot(data, aes(x = Diagnosis, y = Age, group = Diagnosis)) +
  geom_boxplot(fill = "lightblue", color = "black", alpha = 0.7) +
  ggtitle("Boxplot of Age by Diagnosis") +
  xlab("Diagnosis") +
  ylab("Age")
```



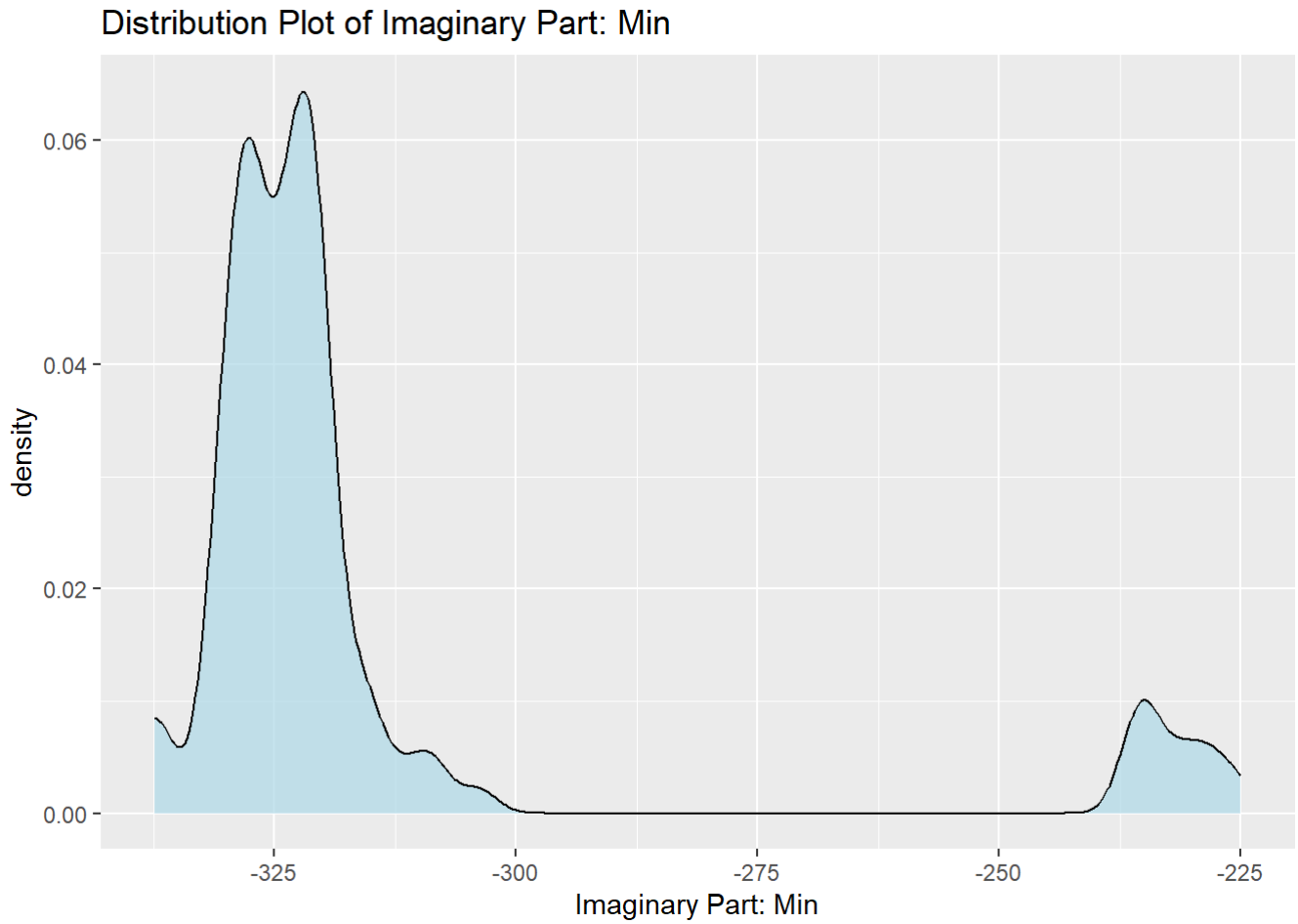
COPD: Seen mostly in middle aged to senior citizens

Asthama: Obsevered across almost all age groups

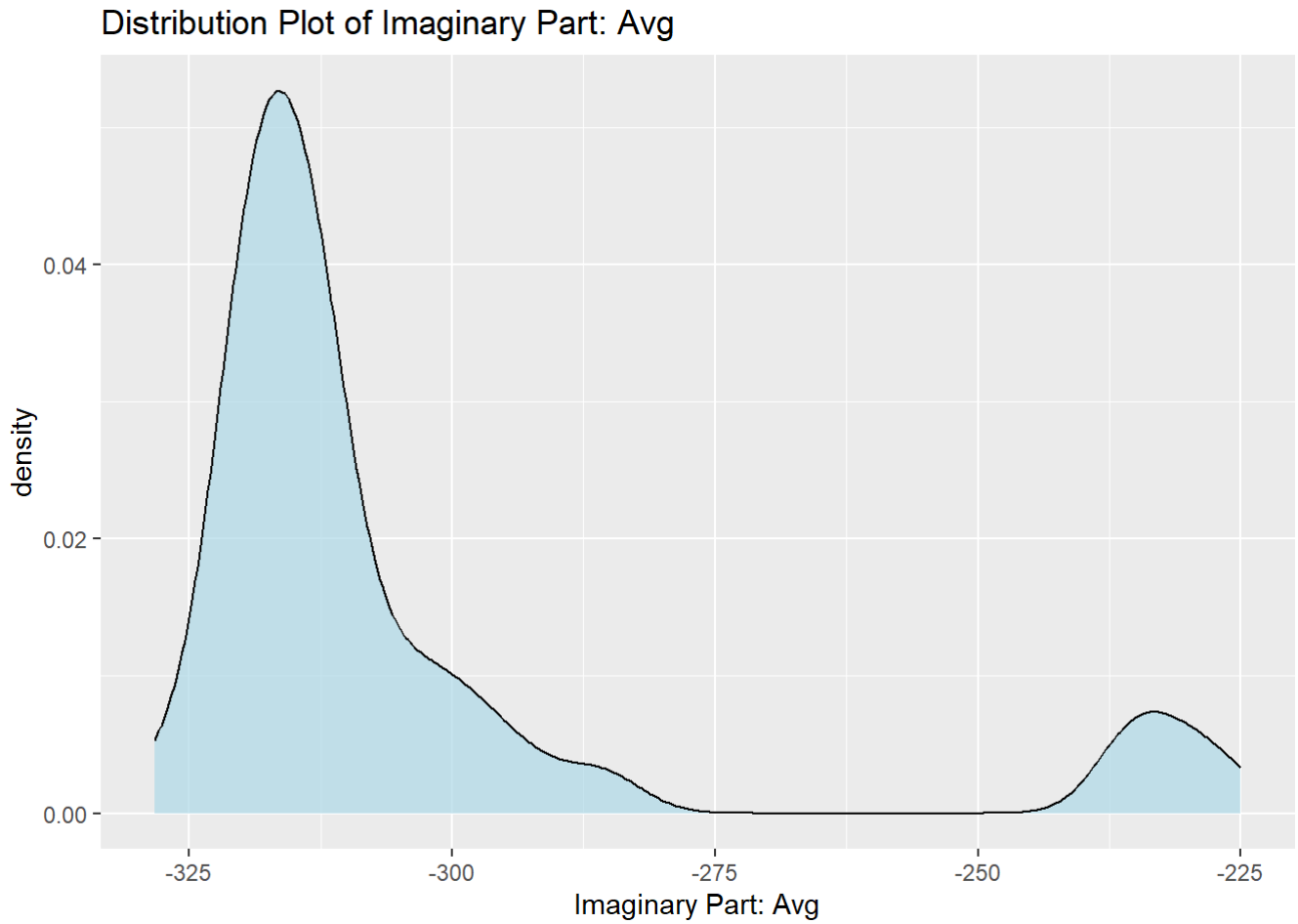
Infections: Not seen in older paients beyond 50-55

Healthy Groups: Oldest paients are upto the age of 60/65

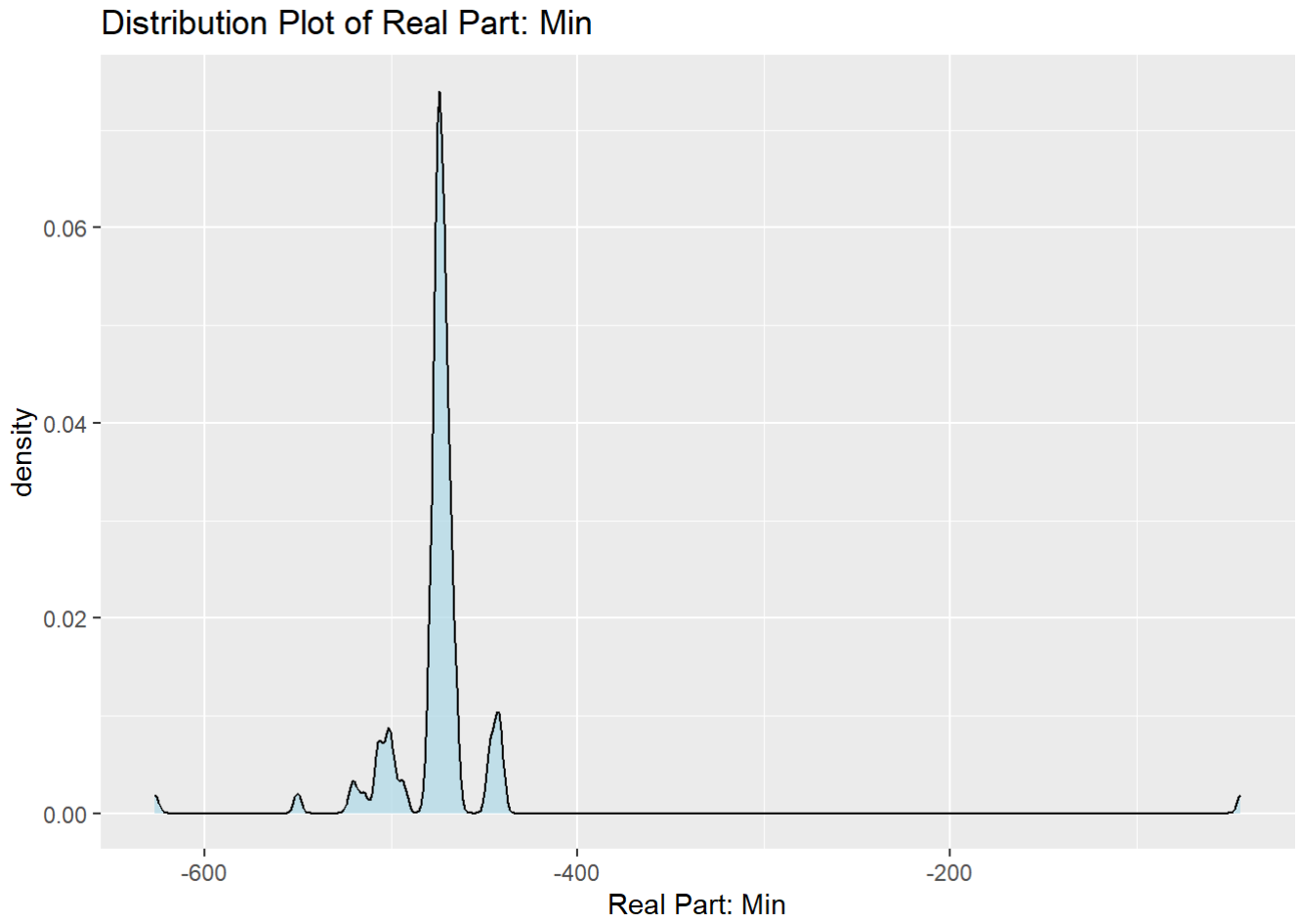
```
ggplot(data, aes(x = `Imaginary Part: Min`)) +
  geom_density(fill = "lightblue", color = "black", alpha = 0.7) +
  ggtitle("Distribution Plot of Imaginary Part: Min") +
  xlab("Imaginary Part: Min")
```



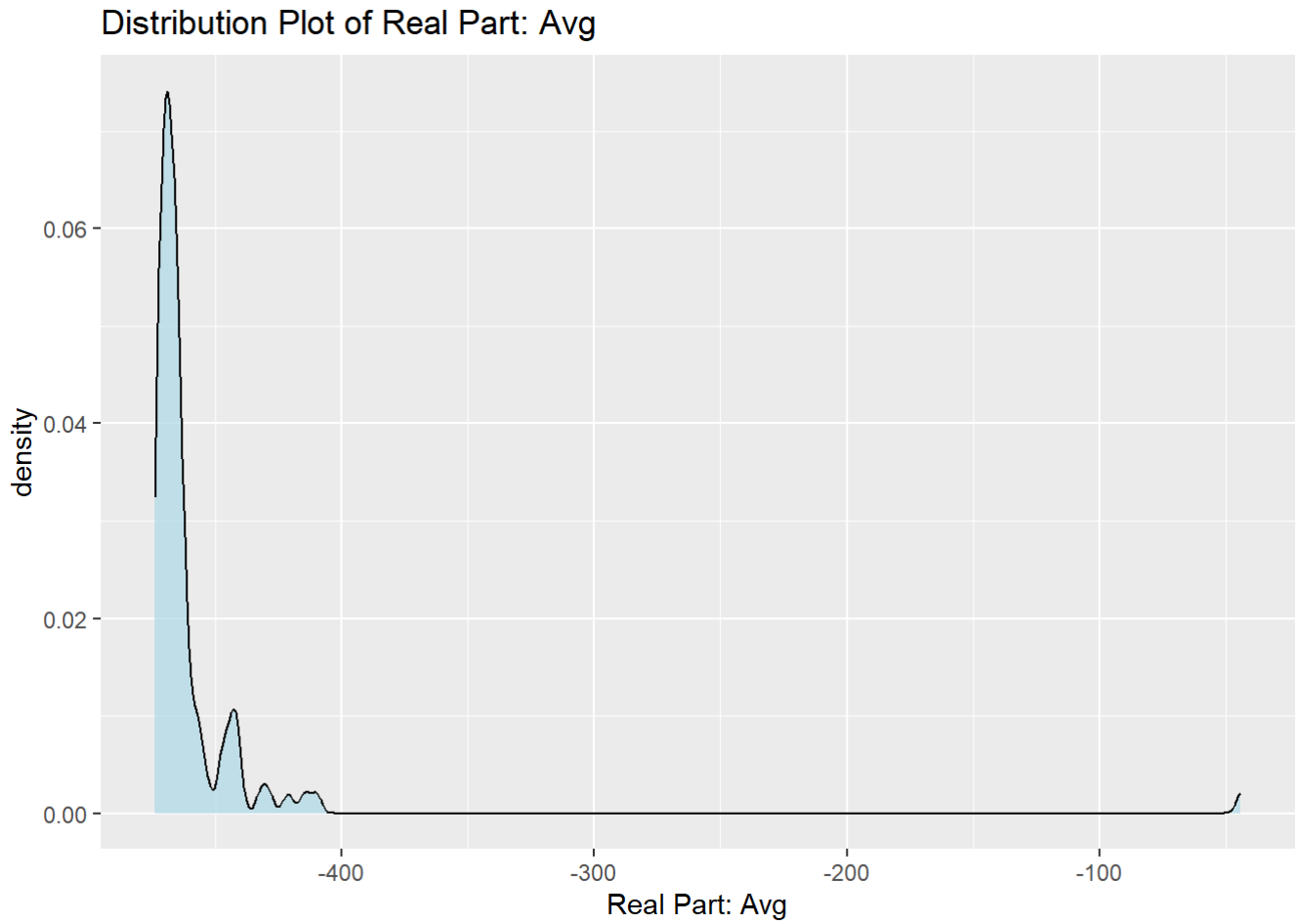

```
ggplot(data, aes(x = `Imaginary Part: Avg`)) +
  geom_density(fill = "lightblue", color = "black", alpha = 0.7) +
  ggtitle("Distribution Plot of Imaginary Part: Avg") +
  xlab("Imaginary Part: Avg")
```




```
ggplot(data, aes(x = `Real Part: Min`)) +
  geom_density(fill = "lightblue", color = "black", alpha = 0.7) +
  ggtitle("Distribution Plot of Real Part: Min") +
  xlab("Real Part: Min")
```

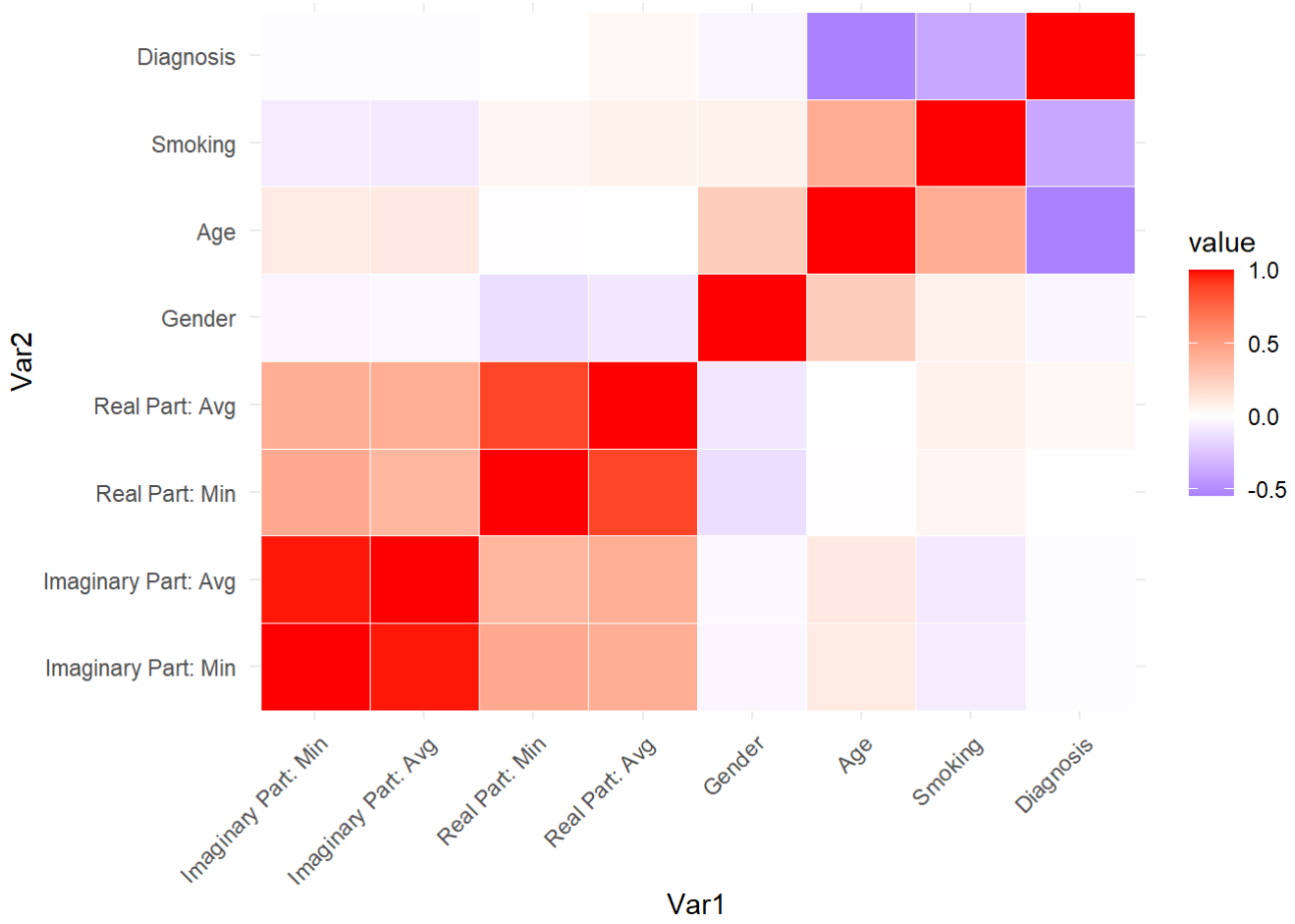



```
ggplot(data, aes(x = `Real Part: Avg`)) +
  geom_density(fill = "lightblue", color = "black", alpha = 0.7) +
  ggtitle("Distribution Plot of Real Part: Avg") +
  xlab("Real Part: Avg")
```



Imaginary part and real part data has a wide distribution with outlier values

```
correlation_long <- melt(cor(data))
ggplot(correlation_long, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Unsupervised Modelling

Importing data for clustering and PCA for plotting

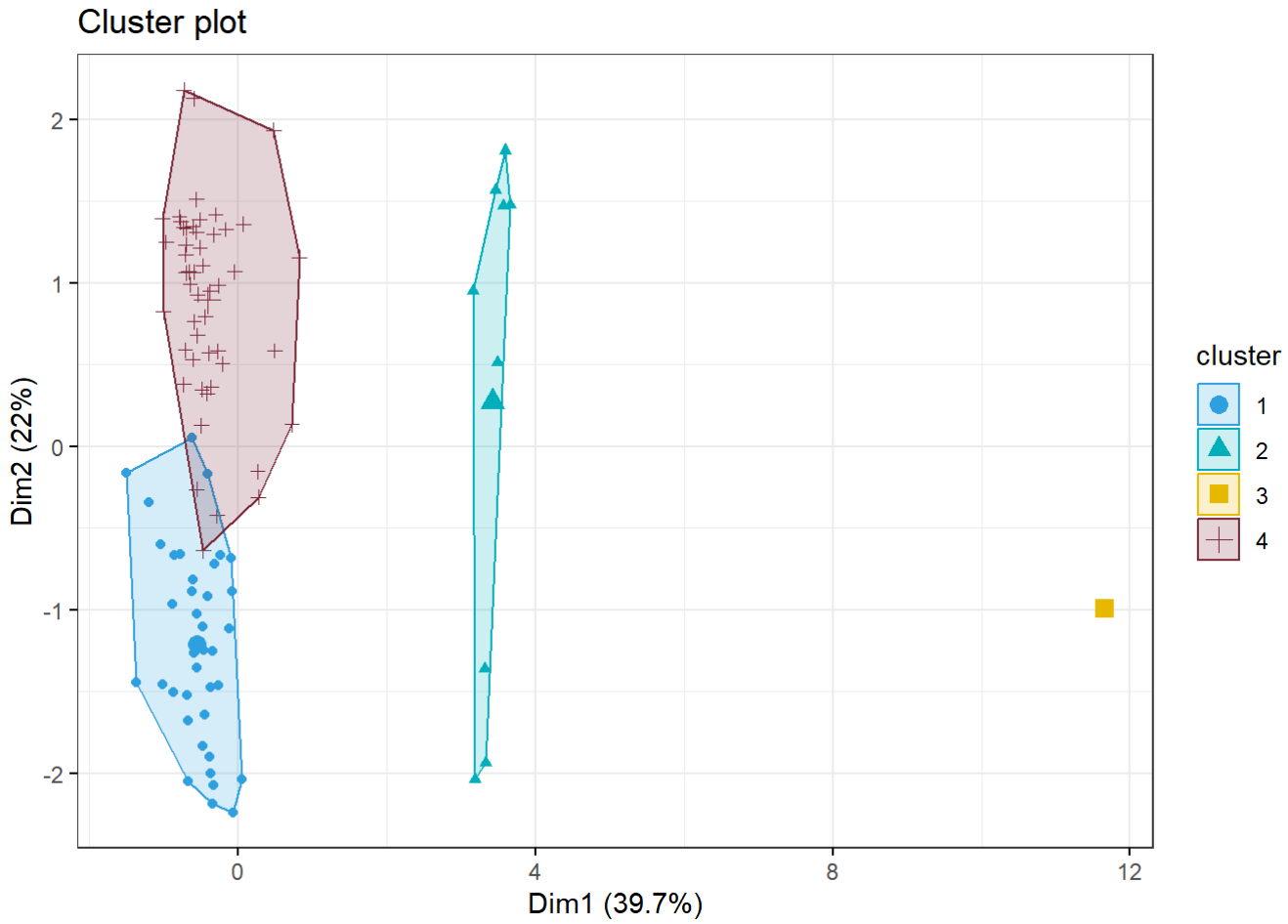
```
setwd("..")
data <- read_csv('Data/cleaned_data.csv')
```

```
## Rows: 100 Columns: 8
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (8): Imaginary Part: Min, Imaginary Part: Avg, Real Part: Min, Real Part...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

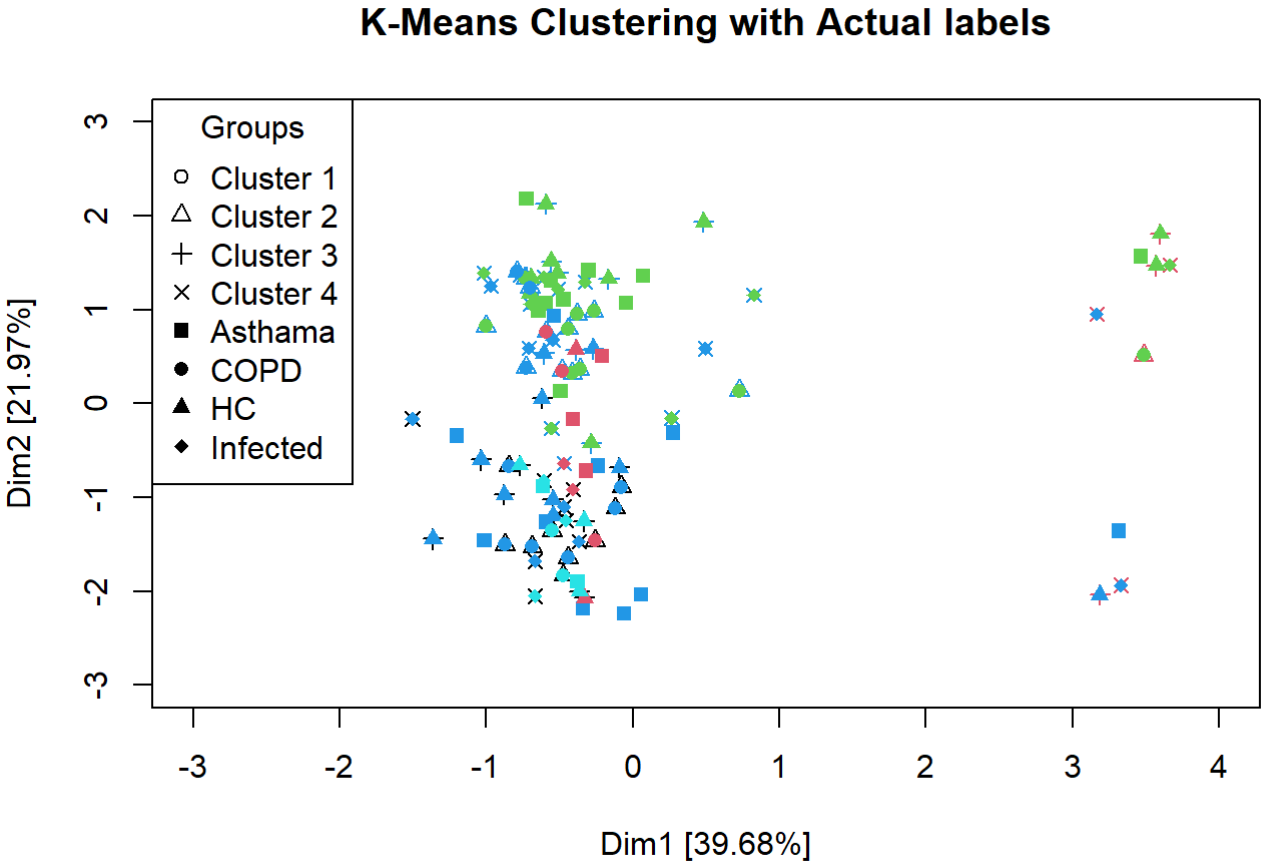
```
diagnosis_column_index <- which(colnames(data) == "Diagnosis")
x_data = data[, -diagnosis_column_index]
y_data = data[, diagnosis_column_index]

# Scaling
x_data_scaled <- scale(x_data)
#PCA
pca_result <- prcomp(x_data_scaled)
variance_explained <- round((pca_result$sdev^2) / sum(pca_result$sdev^2) * 100, 2)
```

K Means

```
k <- 4  # number of clusters
kmeans.fit <- kmeans(x_data_scaled, centers = k, nstart = 20)

fviz_cluster(kmeans.fit, data = x_data_scaled,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#792536"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
             )
```

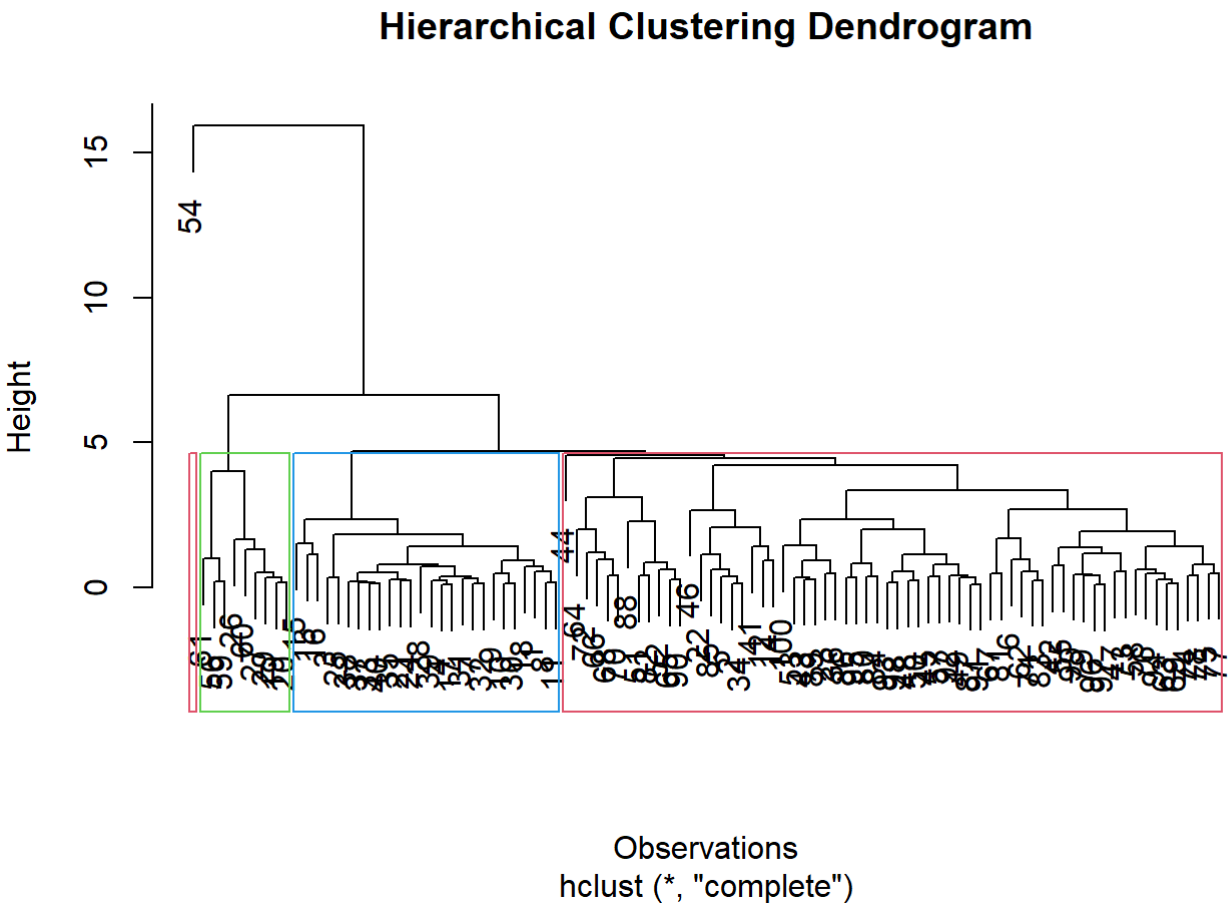



```
# Plotting vs actual labels
pca_result <- prcomp(x_data_scaled)
plot(pca_result$x[, 1], pca_result$x[, 2], col = kmeans.fit$cluster, pch = (1:4), main = "K-M
eans Clustering with Actual labels", xlab=paste('Dim1 [',variance_explained[1],'%]',sep =
''), ylab=paste('Dim2 [',variance_explained[2],'%]',sep = ''), xlim=c(-3,4), ylim=c(-3, 3))
points(pca_result$x[, 1], pca_result$x[, 2], col = y_data[[1]]+10, pch = (15:18))
legend("topleft", legend = c("Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4", "Asthama",
"COPD", "HC", "Infected"), col = c(1, 1), pch = c(1,2,3,4, 15,16,17,18), title = "Groups")
```



Hirarchaial

```
# Perform hierarchical clustering
hclust_result <- hclust(dist(x_data_scaled))

# Plot the dendrogram
plot(hclust_result, main = "Hierarchical Clustering Dendrogram", xlab = "Observations", sub =
NULL)

# Cut the tree to create clusters
num_clusters <- 4
clusters <- cutree(hclust_result, num_clusters)
rect.hclust(hclust_result, k = num_clusters, border = 2:num_clusters)
```

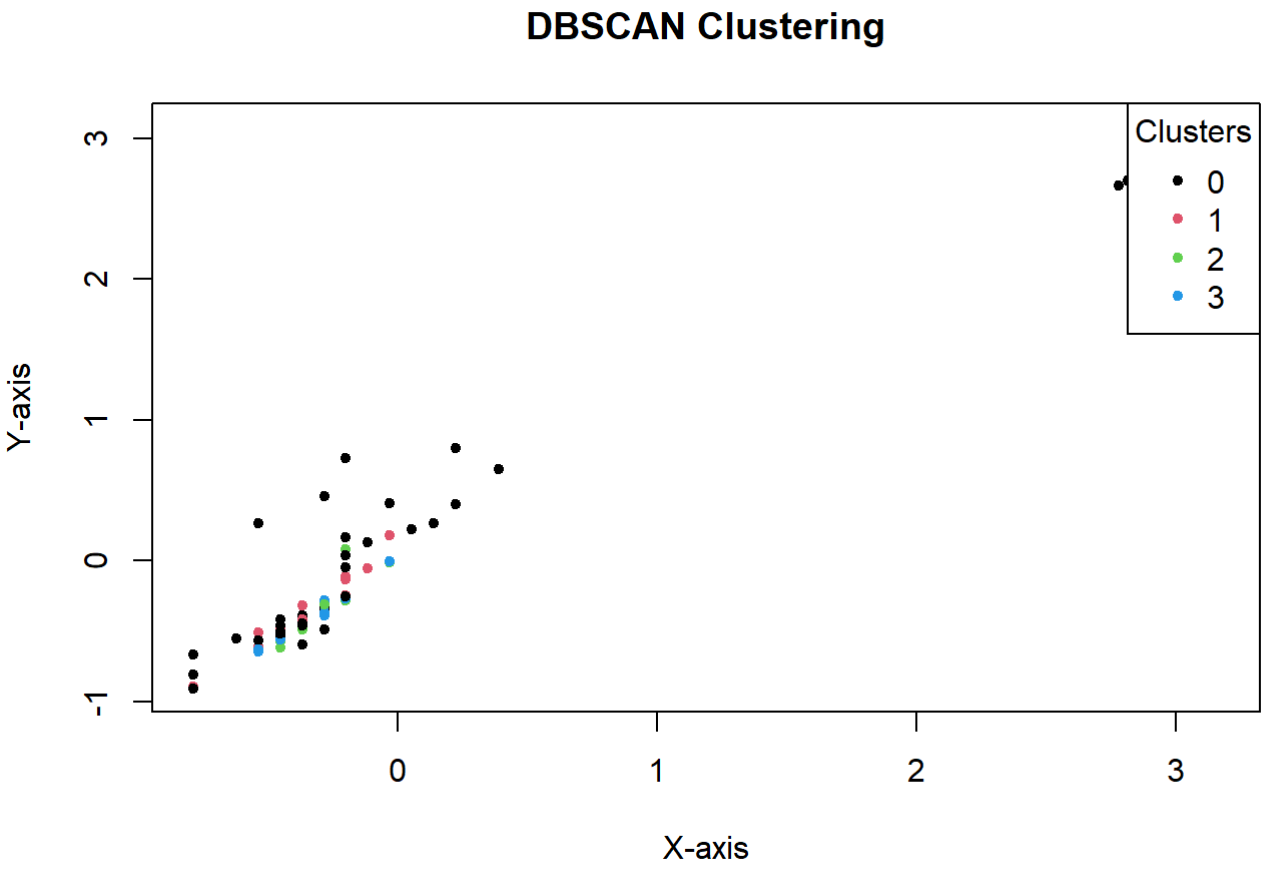


DBSCAN

```
# Set the parameters for DBSCAN
eps <- 0.5  # epsilon, the radius of the neighborhood
minPts <- 5  # minimum number of points to form a dense region (including the point itself)
dbscan_result <- dbscan(x_data_scaled, eps = eps, MinPts = minPts)
```

```
## Warning in dbscan(x_data_scaled, eps = eps, MinPts = minPts): converting
## argument MinPts (fpc) to minPts (dbscan)!
```

```
# Visualize the clusters
plot(x_data_scaled, col = dbscan_result$cluster + 1, pch = 20, main = "DBSCAN Clustering", xl
ab = "X-axis", ylab = "Y-axis")
legend("topright", legend = unique(dbscan_result$cluster), col = unique(dbscan_result$cluste
r) + 1, pch = 20, title = "Clusters")
```



Supervised Modelling

Importing and Preprocessing Data

```
# Read the data
setwd("..")
data <- read_csv('Data/cleaned_data.csv')
```

```
## Rows: 100 Columns: 8
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (8): Imaginary Part: Min, Imaginary Part: Avg, Real Part: Min, Real Part...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data <- data %>% setNames(c('ImaginaryPartMin', 'ImaginaryPartAvg', 'RealPartMin', 'RealPartA
vg', 'Gender', 'Age', 'Smoking', 'Diagnosis'))

# Standardize the data
scaled_data <- scale(data[, -ncol(data)])

# Train-test split
set.seed(123)  # Set seed for reproducibility
split <- sample.split(data$Diagnosis, SplitRatio = 0.7)
train_data <- subset(data, split == TRUE)
test_data <- subset(data, split == FALSE)
```

Support Vector Classifier (SVC)

```
svc_model <- svm(Diagnosis ~ ., data = train_data, kernel = 'radial')
svc_pred_class <- predict(svc_model, newdata = test_data)
```

Calculate and print metrics for Support Vector Classifier (SVC)

```
calculate_metrics(test_data$Diagnosis, svc_pred_class, "Support Vector Classifier (SVC)")
```

```
## ============================================
## Performance Metrics for Support Vector Classifier (SVC) :
## ============================================
## Confusion Matrix:
##             Actual Positive Actual Negative
## Predicted Positive       0               0
## Predicted Negative       1               1
##
## Accuracy:  0.0333
## Precision:  0.0000  0.0833  0.0000  0.0000
## Recall:  0.0000  0.0833  0.0000  0.0000
## F1-Score:  NaN  0.0833  NaN  NaN
## Specificity:  0.0000
```

Bagging Decision Tree

```
bagging_model <- randomForest(factor(Diagnosis) ~ ., data = train_data)
bagging_pred_class <- predict(bagging_model, newdata = test_data)
```

Calculate and print metrics for Bagging Decision Tree

```
calculate_metrics(test_data$Diagnosis, bagging_pred_class, "Bagging Decision Tree")
```

```
## ==========================================
## Performance Metrics for Bagging Decision Tree :
## ==========================================
## Confusion Matrix:
##             Actual Positive Actual Negative
## Predicted Positive       1               0
## Predicted Negative       0               12
##
## Accuracy:  0.8667
## Precision:  0.3333  1.0000  0.8333  1.0000
## Recall:  0.3333  1.0000  0.8333  1.0000
## F1-Score:  0.3333  1.0000  0.8333  1.0000
## Specificity:  0.3333
```

Random Forest

```
rf_model <- randomForest(factor(Diagnosis) ~ ., data = train_data)
rf_pred_class <- predict(rf_model, newdata = test_data)
```

Calculate and print metrics for Random Forest

```
calculate_metrics(test_data$Diagnosis, rf_pred_class, "Random Forest")
```

```
## ==========================================
## Performance Metrics for Random Forest :
## ==========================================
## Confusion Matrix:
##             Actual Positive Actual Negative
## Predicted Positive       1               0
## Predicted Negative       0               12
##
## Accuracy:  0.8667
## Precision:  0.3333  1.0000  0.8333  1.0000
## Recall:  0.3333  1.0000  0.8333  1.0000
## F1-Score:  0.3333  1.0000  0.8333  1.0000
## Specificity:  0.3333
```

Ridge Classifier

```
ridge_model <- lda(Diagnosis ~ ., data = train_data)
ridge_pred_class <- predict(ridge_model, newdata = test_data)$class
```

Calculate and print metrics for Ridge Classifier

```
calculate_metrics(test_data$Diagnosis, ridge_pred_class, "Ridge Classifier")
```

```
## ==========================================
## Performance Metrics for Ridge Classifier :
## ==========================================
## Confusion Matrix:
##             Actual Positive Actual Negative
## Predicted Positive       0                0
## Predicted Negative       1               11
##
## Accuracy:  0.6667
## Precision:  0.0000  0.9167  0.6667  0.3333
## Recall:  0.0000  0.9167  0.6667  0.3333
## F1-Score:  NaN  0.9167  0.6667  0.3333
## Specificity:  0.0000
```