setwd("C:/Users/nateb/OneDrive/Documents/ACCPlayerCase")

```r
#Chi Squared test
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data = read.csv("C:/Users/nateb/OneDrive/Documents/ACCPlayerCase/ACCPlayers.csv")

# filter the data frame to only include the school state and the player's hometown state
data_filtered = data %>%
  select(`School.State`, State)

# tabulate data for chi sqaured
chi_table = table(data_filtered$School.State, data_filtered$State)

# chi squared test for relationship between Hometown State and School State
chi_squared_test = chisq.test(chi_table)
```

```
## Warning in chisq.test(chi_table): Chi-squared approximation may be incorrect
```

```r
chi_squared_test
```

```
##
##  Pearson's Chi-squared test
##
## data:  chi_table
## X-squared = 3305.7, df = 621, p-value < 2.2e-16
```

```r
library(ggplot2)
library(pROC)
```

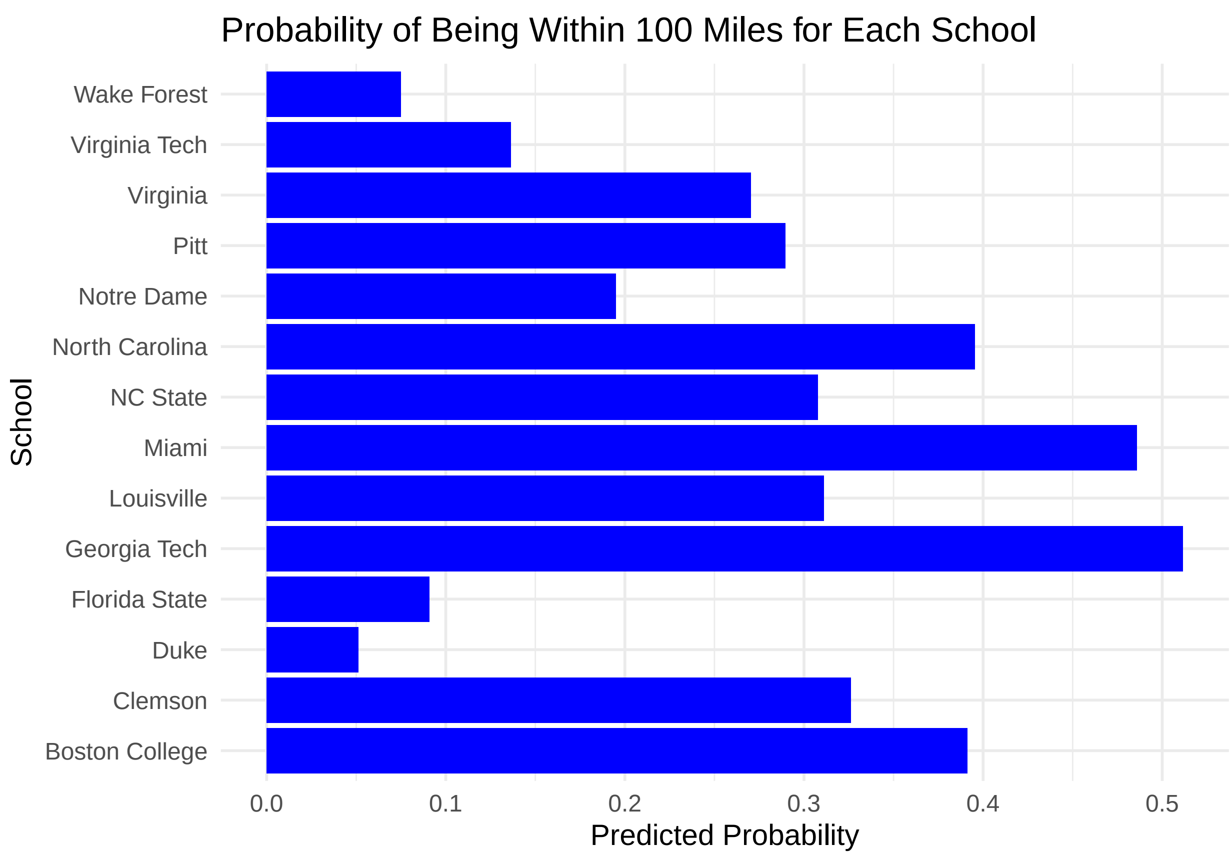
```
## Warning: package 'pROC' was built under R version 4.3.3

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(dplyr)
library(tibble)

data <- read.csv("C:/Users/nateb/OneDrive/Documents/ACCPlayerCase/Player_School_Distances.csv")

# set the radius distance parameter (100 miles as baseline) and create a binary column for Within_Radiu
radius_threshold <- 100
data <- data %>%
  mutate(Within_Radius = ifelse(Distance_mi <= radius_threshold, 1, 0),
         School = factor(School))  # Factorize the School variable for releveling

# set Duke as the reference school (they serve as a good baseline given their low in-state concentratio
data$School <- relevel(data$School, ref = "Duke")

# logistic regression on distance to school
model <- glm(Within_Radius ~ School, data = data, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = Within_Radius ~ School, family = binomial, data = data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -2.9178     0.7259  -4.019 5.84e-05 ***
## SchoolBoston College    2.4759     0.7863   3.149 0.001639 **
## SchoolClemson           2.1918     0.7911   2.770 0.005598 **
## SchoolFlorida State     0.6152     0.8955   0.687 0.492116
## SchoolGeorgia Tech      2.9643     0.7874   3.764 0.000167 ***
## SchoolLouisville        2.1228     0.7941   2.673 0.007515 **
## SchoolMiami             2.8606     0.8009   3.572 0.000354 ***
## SchoolNC State          2.1068     0.8046   2.619 0.008830 **
## SchoolNorth Carolina    2.4929     0.7901   3.155 0.001604 **
## SchoolNotre Dame        1.5007     0.8260   1.817 0.069245 .
## SchoolPitt              2.0198     0.8093   2.496 0.012566 *
## SchoolVirginia          1.9245     0.8149   2.362 0.018190 *
## SchoolVirginia Tech     1.0719     0.8485   1.263 0.206471
## SchoolWake Forest       0.4055     0.9420   0.430 0.666879
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 681.30  on 579  degrees of freedom
## Residual deviance: 617.73  on 566  degrees of freedom
## AIC: 645.73
##
## Number of Fisher Scoring iterations: 5
```

```r
# probabilities for each school and create prediction data for plotting
prediction_data <- data.frame(School = levels(data$School),
                              Predicted_Probability = predict(model, newdata = data.frame(School = level
```

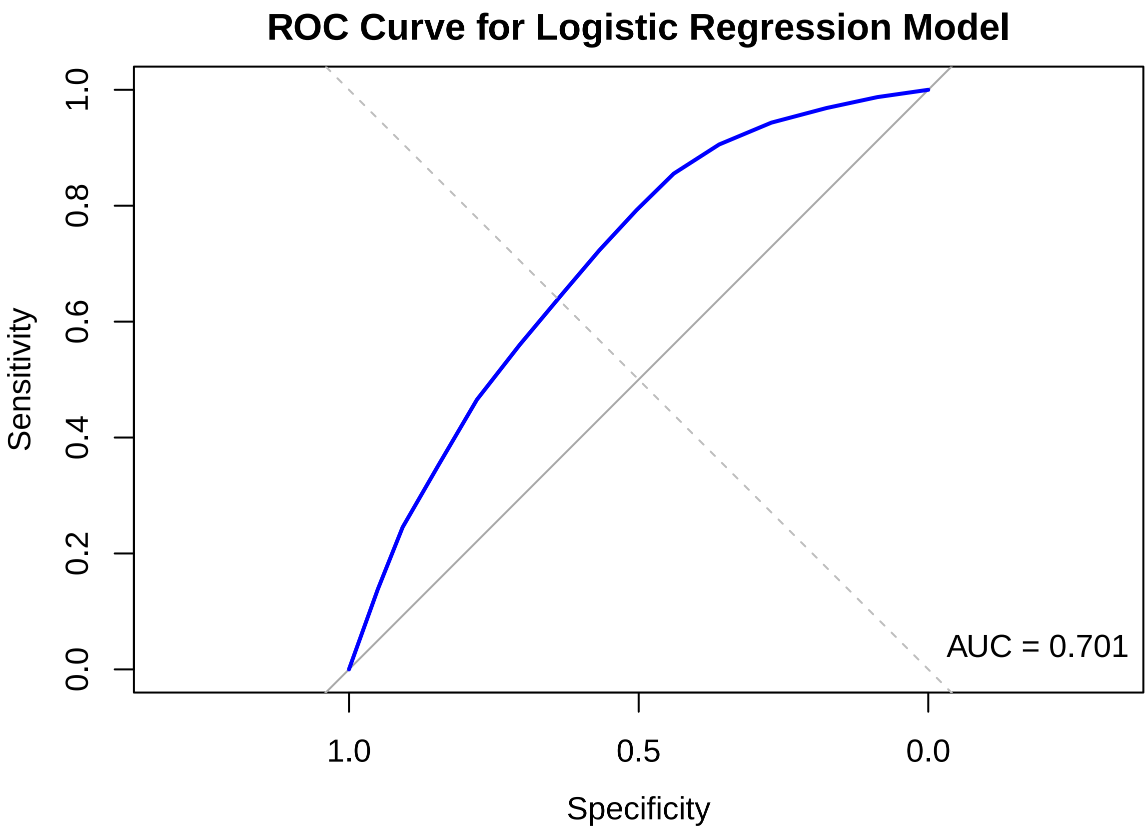
```r
# plot predicted probabilities using ggplot2
ggplot(prediction_data, aes(x = School, y = Predicted_Probability)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Probability of Being Within 100 Miles for Each School",
       x = "School",
       y = "Predicted Probability") +
  theme_minimal() +
  coord_flip()  # Horizontal bar plot
```




```r
# predict the probabilities for the full dataset
data$Predicted_Probability <- predict(model, type = "response")

# calculate the ROC curve and AUC
roc_curve <- roc(data$Within_Radius, data$Predicted_Probability)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
auc_value <- auc(roc_curve)

# plot the ROC curve with AUC
plot(roc_curve, col = "blue", main = "ROC Curve for Logistic Regression Model")
abline(a = 0, b = 1, lty = 2, col = "gray")  # Diagonal reference line for random guessing
legend("bottomright", legend = paste("AUC =", round(auc_value, 3)), bty = "n", col = "blue")
```




```r
# create confusion matrix
predicted_classes <- ifelse(data$Predicted_Probability > 0.5, 1, 0)  # Threshold at 0.5
confusion_matrix <- table(Predicted = predicted_classes, Actual = data$Within_Radius)

# convert the confusion matrix to a tibble for easier viewing and manipulation
confusion_tibble <- as_tibble(as.data.frame(confusion_matrix))
print(confusion_tibble)  # Print the confusion matrix as a tibble
```

```
## # A tibble: 4 x 3
##   Predicted Actual  Freq
##   <fct>     <fct>  <int>
## 1 0         0        400
## 2 1         0         21
## 3 0         1        137
## 4 1         1         22
```

```r
# calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy:", round(accuracy, 3)))
```

```
## [1] "Accuracy: 0.728"
```