P&S-2024: Final Project

Vladyslav Sydorak, Mykhailo Ponomarenko, Mykhailo Ivasiuk

```r
library(tidyr)
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
library(ggplot2)
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.4.2
```

```r
data <- read.csv("socioeconomic_voting.csv")
colnames(data) <- c("State", "County", "FIPS", "Unemployment", "MedianIncome", "UrbanInfluenceCod
e", "IncomePercentile", "BachelorsPercentage", "WinningParty", "PercentageForDominantParty")
data
```

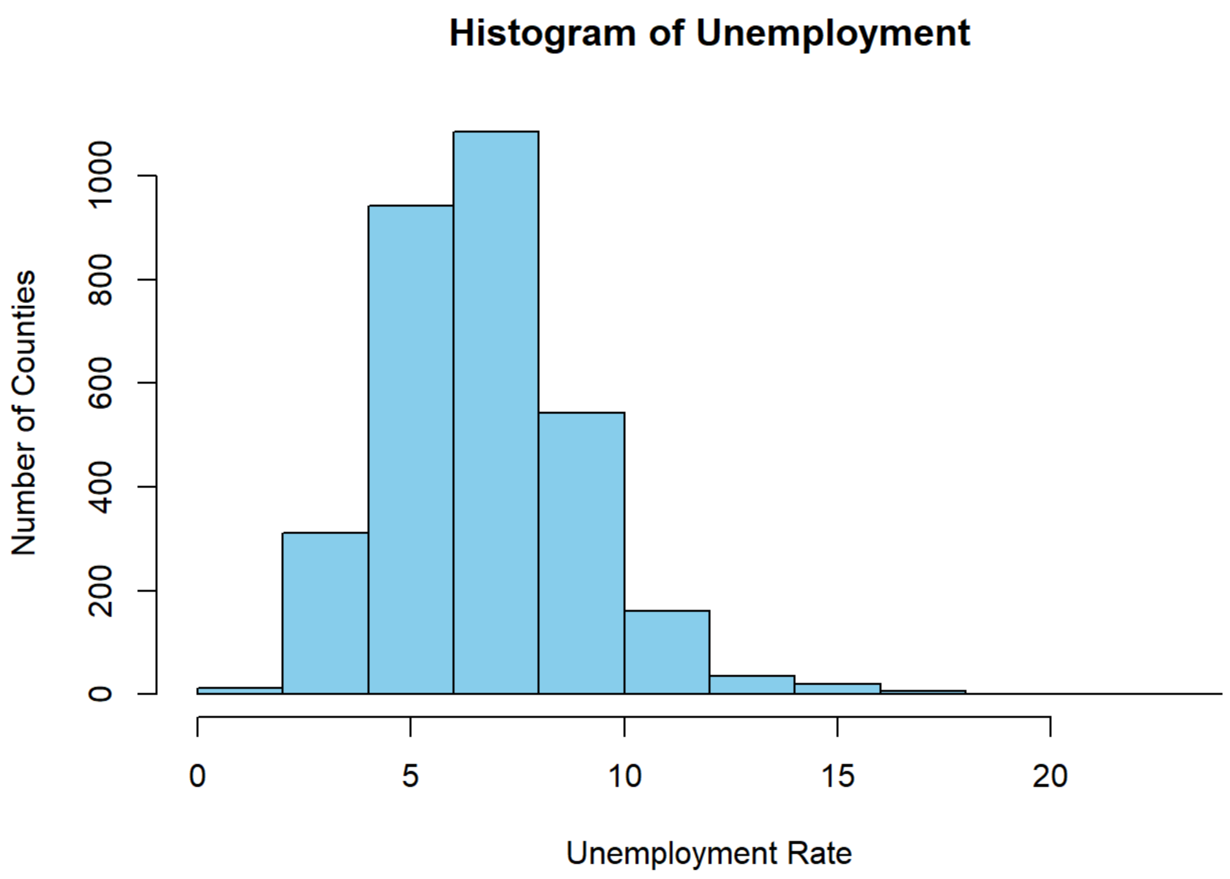
State <chr>	County <chr>	FI… <int>	Unemployment <dbl>	MedianIncome <dbl>	UrbanInfluenceCode <dbl>	IncomePercentile <dbl>
ALABAMA	AUTAUGA	1001	5.3	66444	2	0.955223881
ALABAMA	BALDWIN	1003	6.1	65658	2	0.940298507
ALABAMA	BARBOUR	1005	7.7	38649	6	0.164179104
ALABAMA	BIBB	1007	7.3	48454	1	0.611940299
ALABAMA	BLOUNT	1009	4.5	56894	1	0.895522388
ALABAMA	BULLOCK	1011	6.0	32027	6	0.074626866
ALABAMA	BUTLER	1013	9.5	39442	6	0.194029851
ALABAMA	CALHOUN	1015	7.7	48166	2	0.597014925
ALABAMA	CHAMBERS	1017	7.4	45447	5	0.410447761
ALABAMA	CHEROKEE	1019	5.1	46365	6	0.507462687

1-10 of 3,115 rows | 1-7 of 10 columns Previous **1** 2 3 4 5 6 ... 312 Next

Processing math: 100%

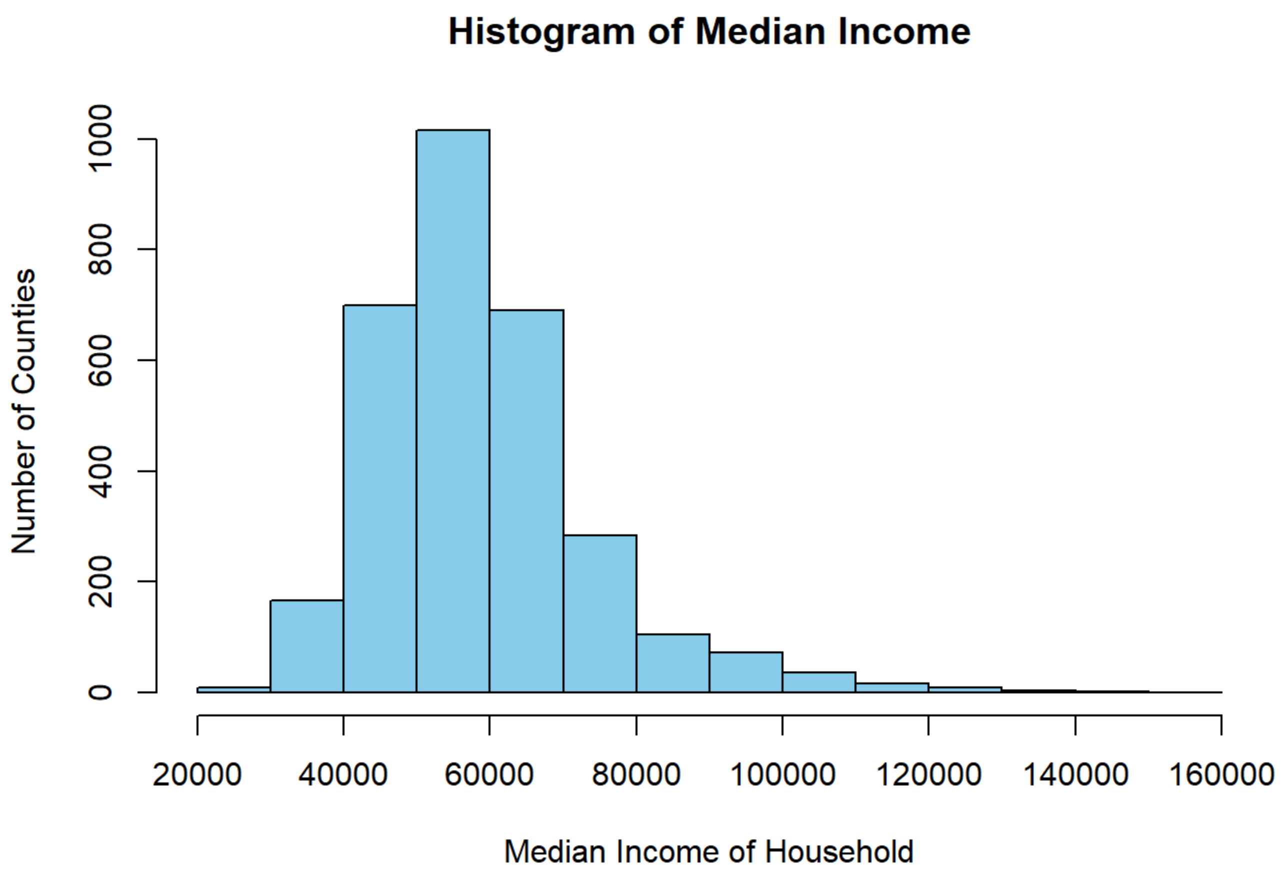
So, in our dataset we have 10 columns, now we have to conduct our initial descriptive analysis of this data. First three columns contain pretty obvious data-information about the place which our dataset describes. Name of the state, of a county within it and FIPS-a unique code for each county. For all of the following columns we will draw a histogram to approximate their distribution and find a relationship between them.

```
hist(data$Unemployment, main = "Histogram of Unemployment",
     xlab = "Unemployment Rate", ylab = "Number of Counties",  col = "skyblue", border = "black")
```

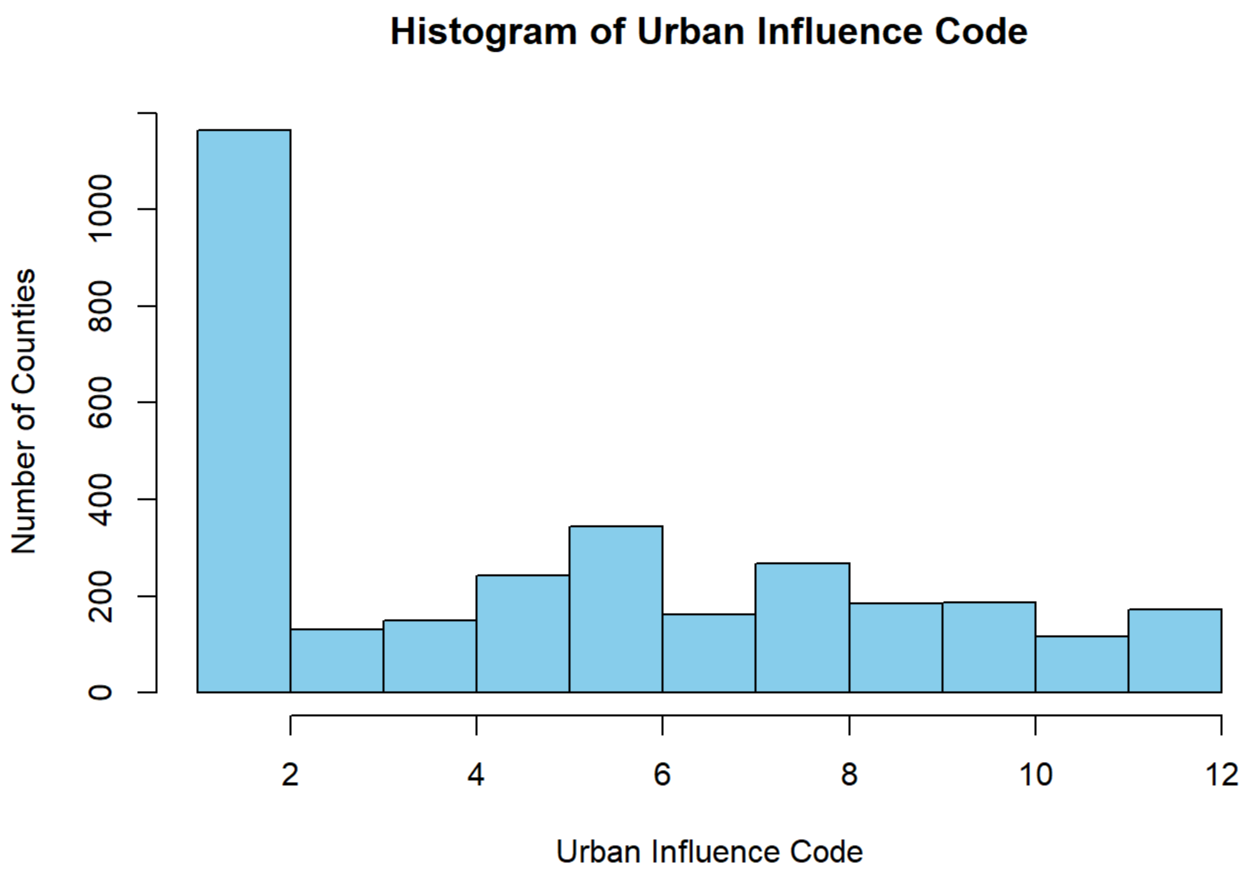


```
hist(data$MedianIncome, main = "Histogram of Median Income",
     xlab = "Median Income of Household", ylab = "Number of Counties", col = "skyblue", border =
"black")
```

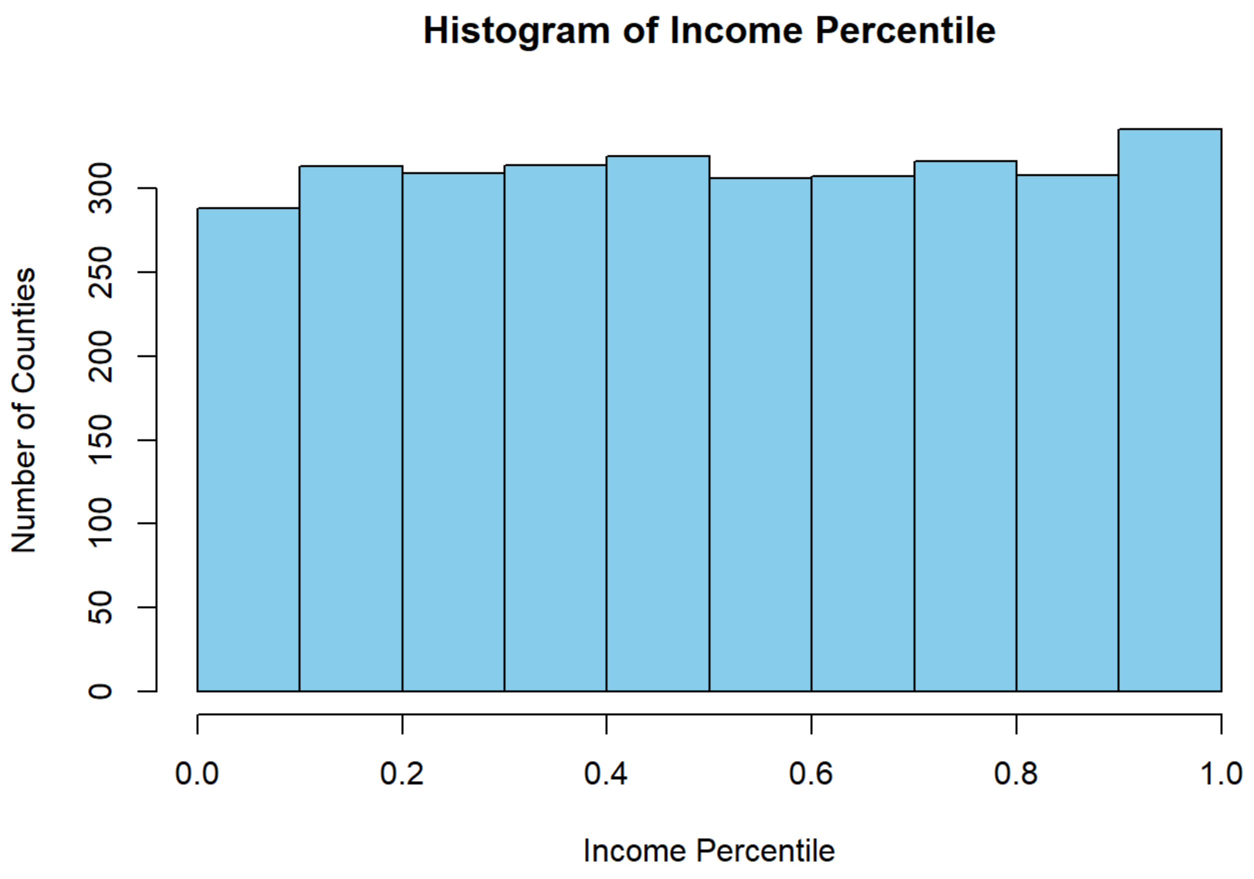
Processing math: 100%



```
hist(data$UrbanInfluenceCode, main = "Histogram of Urban Influence Code",
     xlab = "Urban Influence Code", ylab = "Number of Counties", col = "skyblue", border = "blac
k")
```

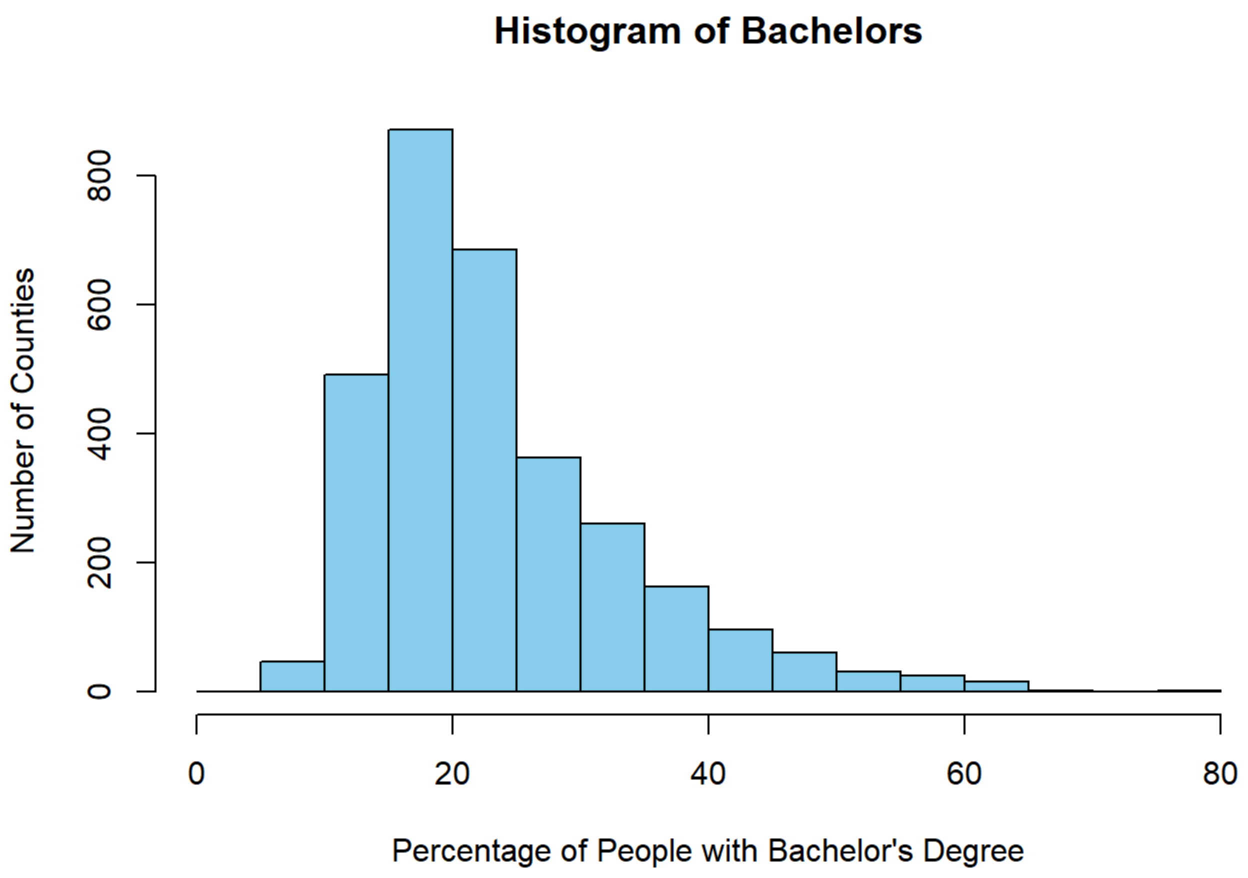


```
hist(data$IncomePercentile, main = "Histogram of Income Percentile",
     xlab = "Income Percentile", ylab = "Number of Counties", col = "skyblue", border = "black")
```

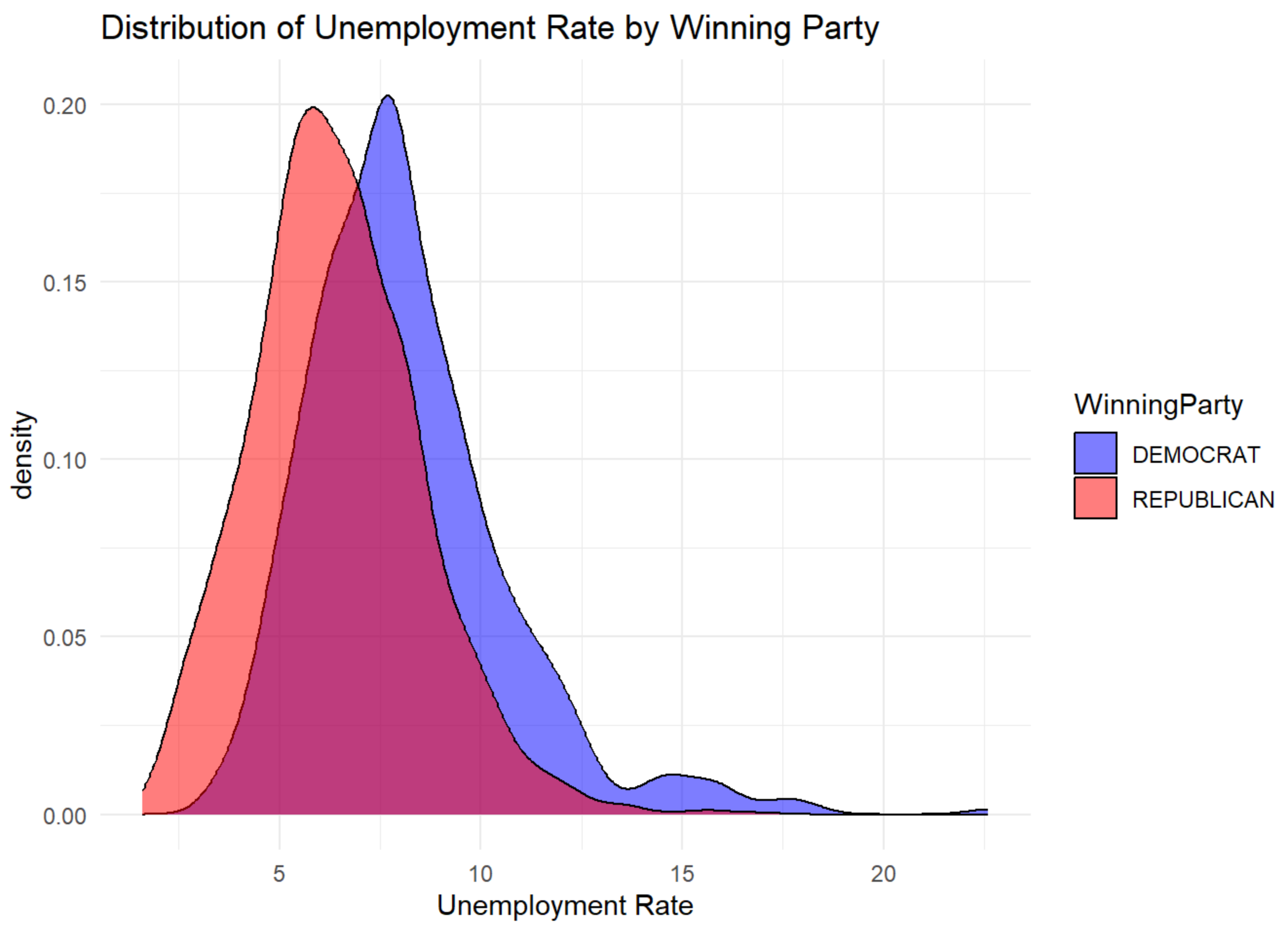


Processing math: 100%

```
hist(data$BachelorsPercentage, main = "Histogram of Bachelors",
     xlab = "Percentage of People with Bachelor's Degree", ylab = "Number of Counties", col = "sky
blue", border = "black")
```



```
ggplot(data, aes(x = Unemployment, fill = WinningParty)) +
  geom_density(alpha = 0.5) +
  xlab("Unemployment Rate") +
  ggtitle("Distribution of Unemployment Rate by Winning Party") +
  scale_fill_manual(values = c("REPUBLICAN" = "red", "DEMOCRAT" = "blue")) +
  theme_minimal()
```



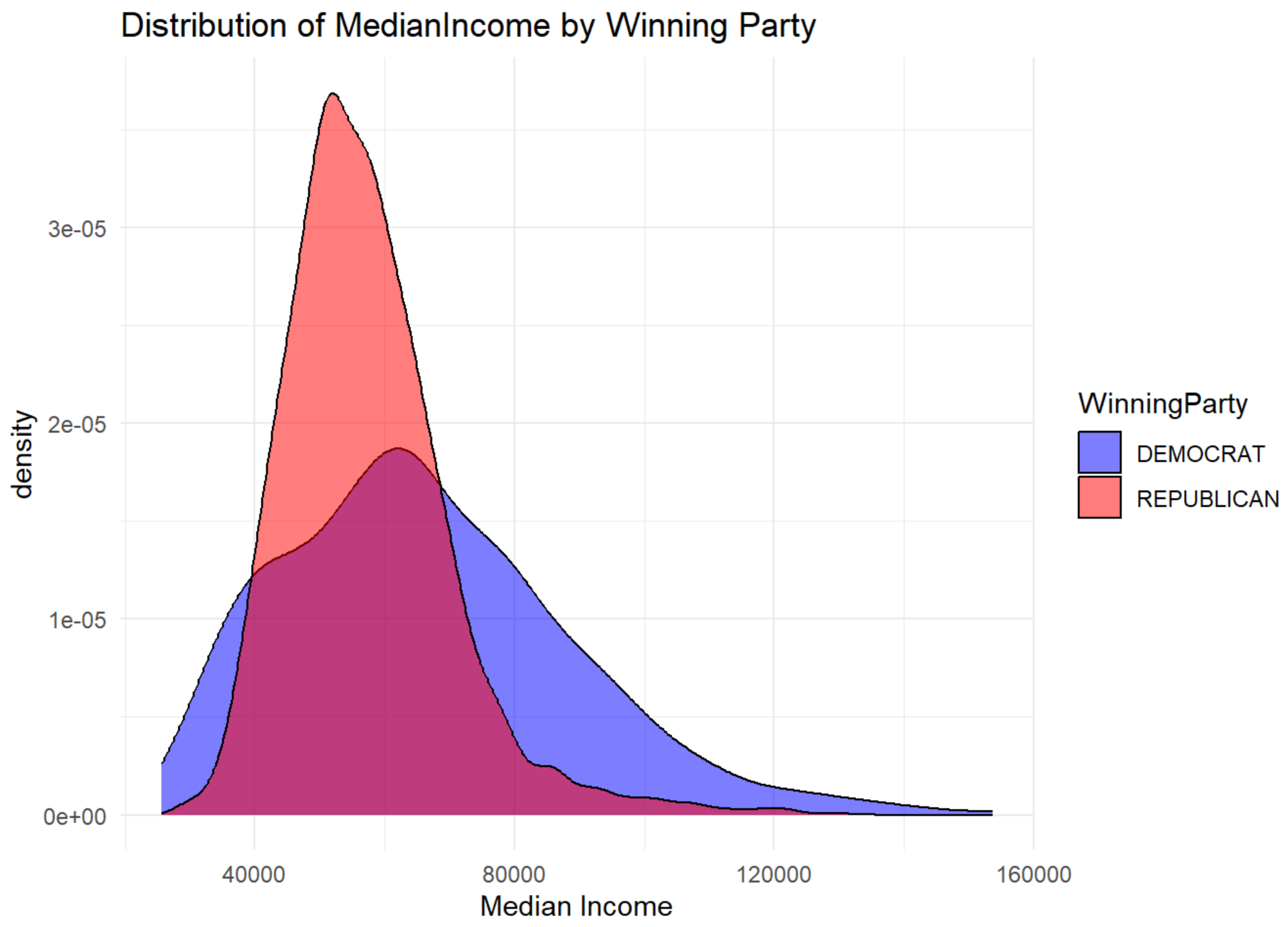
Republican voters have a slightly lower unemployment rate, as the blue curve peaks closer to the left in contrast to democratics.

There is significant overlap between the two curves, indicating that unemployment rate alone does not perfectly distinguish Democrat vs Republican counties.

The Republican and Democrat curves have a sharper, narrower peak, suggesting that most of them distribute around lower unemployment rates. (~ 5 to 8)

```
ggplot(data, aes(x = MedianIncome, fill = WinningParty)) +
  geom_density(alpha = 0.5) +
  xlab("Median Income") +
  ggtitle("Distribution of MedianIncome by Winning Party") +
  scale_fill_manual(values = c("REPUBLICAN" = "red", "DEMOCRAT" = "blue")) +
  theme_minimal()
```

Processing math: 100%



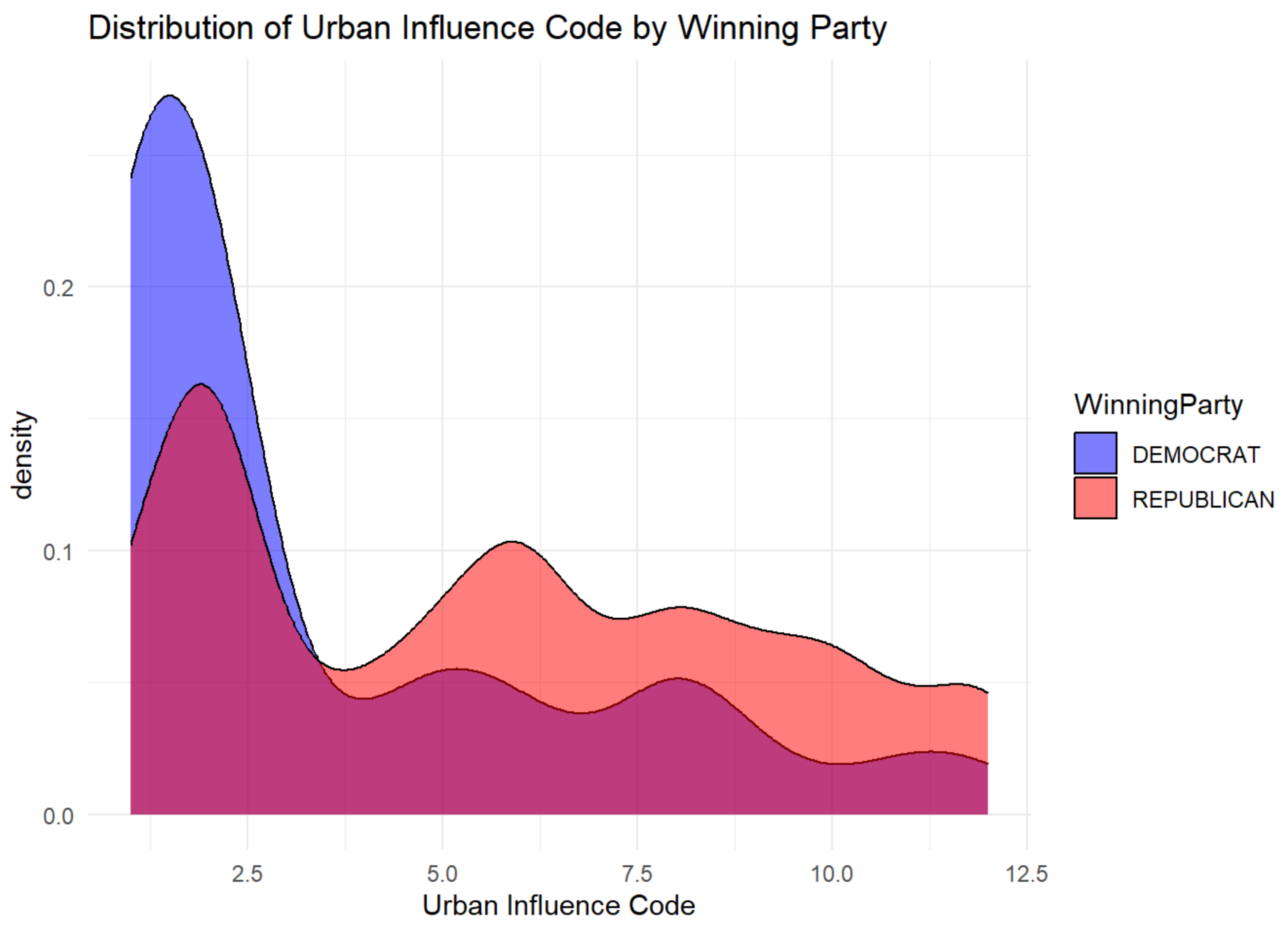
The Republican curve peaks at a lower median income (around \$40,000–\$50,000). The Democratic curve peaks at a slightly higher median income (around \$50,000–\$65,000).

Also note: Republican counties have a narrower distribution, with most counties concentrated in the \$40,000–\$60,000 income range. There is very little representation above \$80,000.

Democrat counties have a wider distribution. While many are in the \$50,000–\$60,000 range, there is a significant number of counties with median incomes well above \$80,000

```
ggplot(data, aes(x = UrbanInfluenceCode, fill = WinningParty)) +
  geom_density(alpha = 0.5) +
  xlab("Urban Influence Code") +
  ggtitle("Distribution of Urban Influence Code by Winning Party") +
  scale_fill_manual(values = c("REPUBLICAN" = "red", "DEMOCRAT" = "blue")) +
  theme_minimal()
```

Processing math: 100%



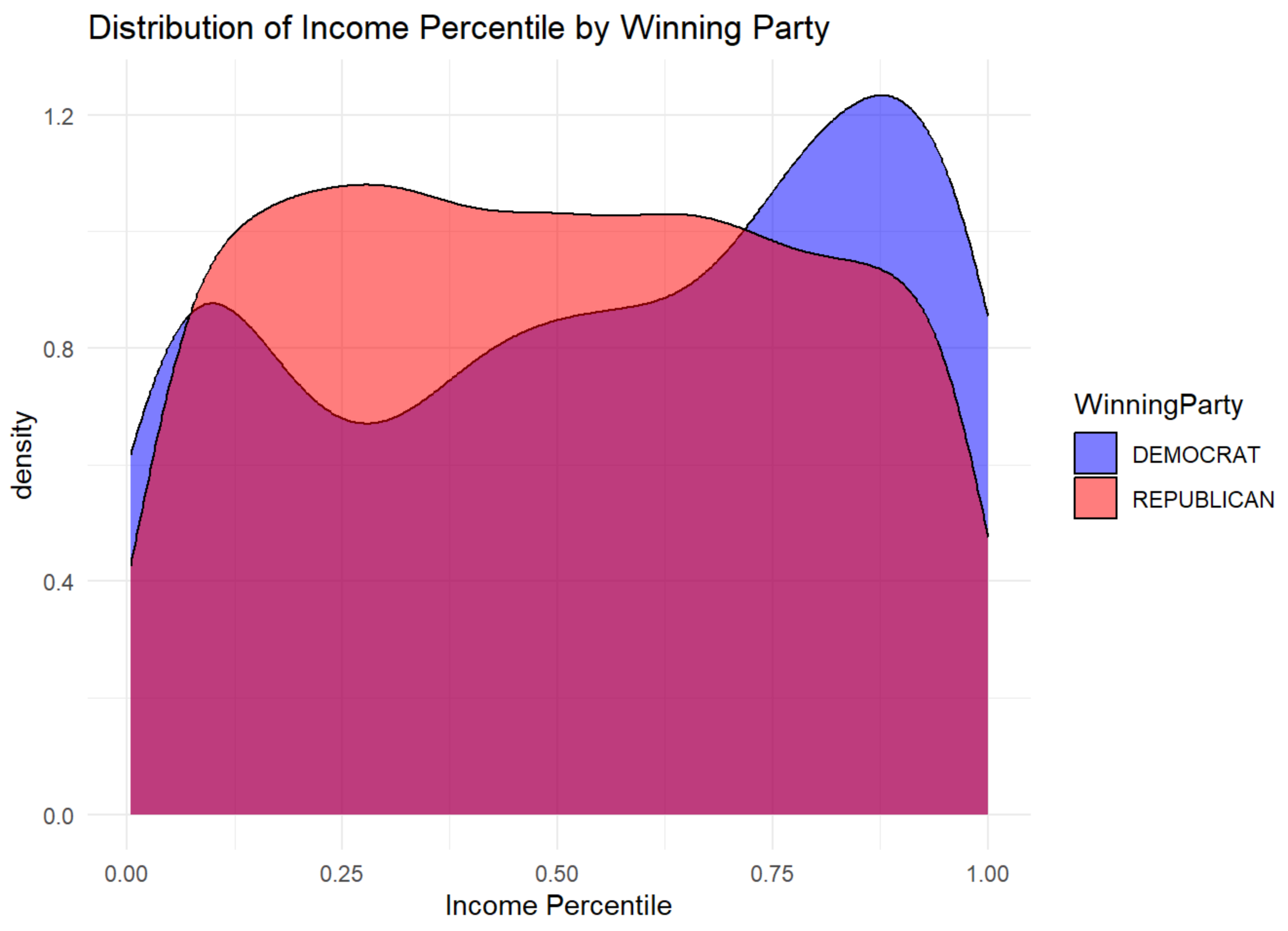
The Democratic curve is highest for lower values of Urban Influence Code (around 1-2) indicating highly urban areas.

The Republican curve shows a broader distribution with peaks around the middle values (3-5) and a steady decline afterward, more evenly distributed across mid-to-high values, indicating less concentration in the most urban areas.

For very high Urban Influence Code, the distributions are similar, but Republicans retain a slight density.

```
ggplot(data, aes(x = IncomePercentile, fill = WinningParty)) +
  geom_density(alpha = 0.5) +
  xlab("Income Percentile") +
  ggtitle("Distribution of Income Percentile by Winning Party") +
  scale_fill_manual(values = c("REPUBLICAN" = "red", "DEMOCRAT" = "blue")) +
  theme_minimal()
```

Processing math: 100%

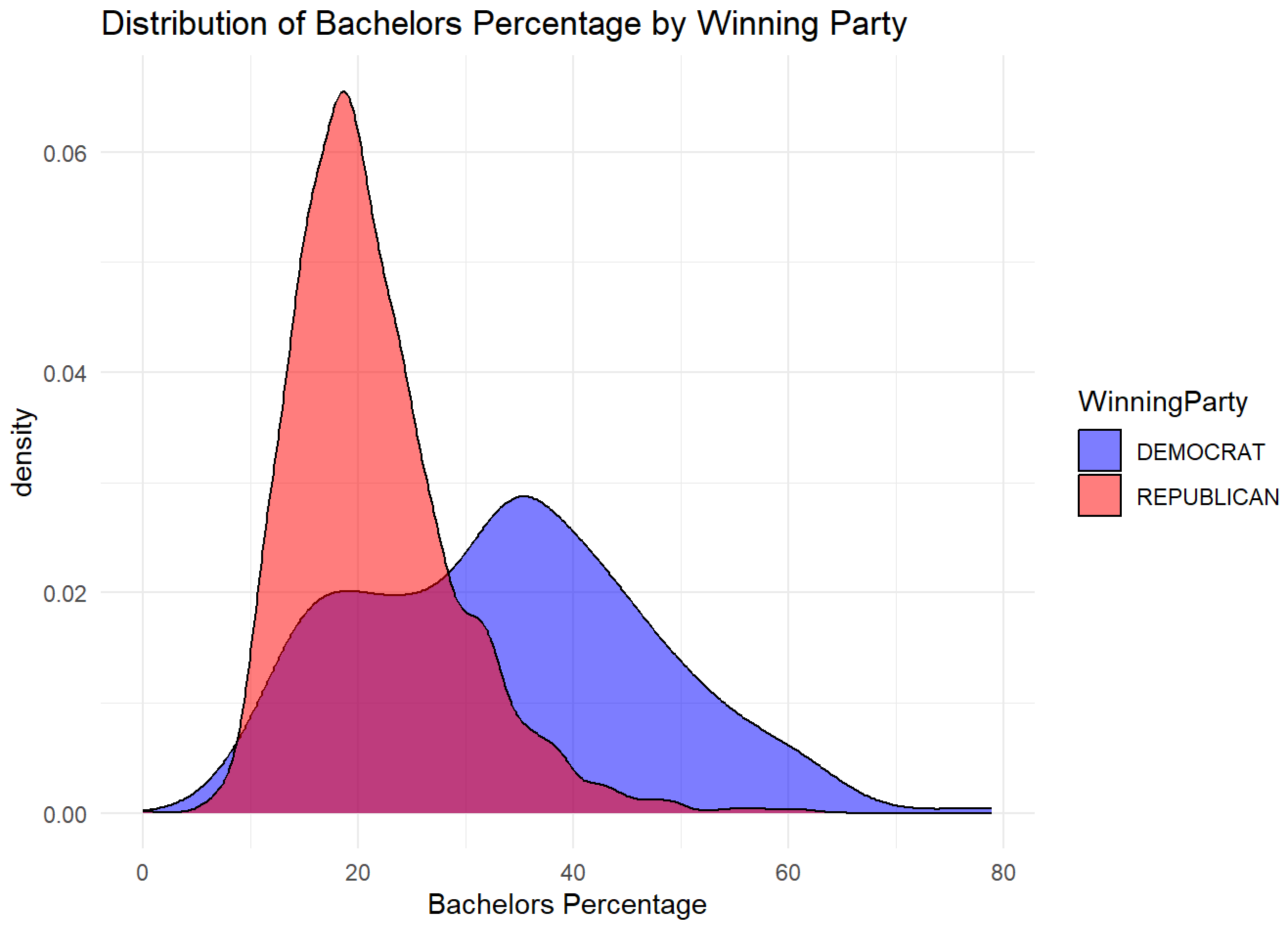


The Democratic curve has two peaks, indicating a small distribution around lowest-income and a noticeable peak in the higher income percentiles. Concentration in higher-income groups.

The Republican curve has a very strong density at lower to mid-income percentiles. Significant levels of middle-income.

```
ggplot(data, aes(x = BachelorsPercentage, fill = WinningParty)) +
  geom_density(alpha = 0.5) +
  xlab("Bachelors Percentage") +
  ggtitle("Distribution of Bachelors Percentage by Winning Party") +
  scale_fill_manual(values = c("REPUBLICAN" = "red", "DEMOCRAT" = "blue")) +
  theme_minimal()
```

Processing math: 100%



The Republican curve shows the distribution peaks at lower levels of bachelor's degree percentages, around 20%. The density decreases significantly beyond 20%, indicating small levels of high education.

The Democratic curve has a peak at higher levels of education (around 30-40%), also in contrast to Republicans, Democrats maintain density at bachelor's percentages around 50-60%.

Overlap in the low-mid range, indicates moderate low-mid level of education for both party's.

Hypotheses Testing

The Chi-Squared test is used to determine if there is a significant difference between observed and expected frequencies. The test statistic is given by:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

where:

- O_i = observed frequency for category i,
- E_i = expected frequency for category i,
- k = number of categories.

This statistic follows a Chi-Squared distribution with $k - 1$ degrees of freedom.

Interpreting the Result

To determine whether the result is statistically significant, we compare the test statistic (χ^2) to the critical value of the Chi-Squared distribution at the chosen significance level (α):

- Find the critical value, χ^2_{critical}, from a Chi-Squared distribution table with $k - 1$ degrees of freedom.

Processing math: 100%

- If $\chi^2 > \chi^2_{\text{critical}}$, reject the null hypothesis H_0.

Alternatively, calculate the p-value:

- The p-value is the probability of observing a test statistic at least as extreme as the calculated χ^2, assuming H_0 is true.
- Use statistical software or a Chi-Squared distribution table to find the p-value.
- If the p-value is less than α, reject the null hypothesis H_0.

Conclusion

Based on the comparison:

- If $\chi^2 > \chi^2_{\text{critical}}$ or $\text{p-value} < \alpha$, conclude that there is a significant difference.
- Otherwise, fail to reject H_0, indicating no significant difference.

Testing

Now let's check how different factors affect the outcome of the election.

```
data$WinningParty <- ifelse(data$WinningParty == "DEMOCRAT", 1, 0)
```

```
neyman_pearson <- function(list) {
   democratic <- c()
   republican <- c()
   values <- unique(list)
   for (i in 1:length(values)) {
     democratic[i] = 0
     republican[i] = 0
   }
   for (i in 1:nrow(data)) {
     num <- which(values == list[i])
     if (data$WinningParty[i] == 1) {
       democratic[num] <- democratic[num] + 1
     }
     else {
       republican[num] <- republican[num] + 1
     }

   }
   df <- data.frame(democratic, republican)
   return(df)
}
print("Unemployment")
```

```
## [1] "Unemployment"
```

```
chisq.test(neyman_pearson(data$Unemployment))
```

```
## Warning in chisq.test(neyman_pearson(data$Unemployment)): Chi-squared
## approximation may be incorrect
```

Processing math: 100%

```
##
##  Pearson's Chi-squared test
##
## data:  neyman_pearson(data$Unemployment)
## X-squared = 425.56, df = 140, p-value < 2.2e-16
```

```
print("Median Income")
```

```
## [1] "Median Income"
```

```
chisq.test(neyman_pearson(data$MedianIncome))
```

```
## Warning in chisq.test(neyman_pearson(data$MedianIncome)): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  neyman_pearson(data$MedianIncome)
## X-squared = 3066.6, df = 3013, p-value = 0.2436
```

```
print("Urban Influence Code")
```

```
## [1] "Urban Influence Code"
```

```
chisq.test(neyman_pearson(data$UrbanInfluenceCode))
```

```
##
##  Pearson's Chi-squared test
##
## data:  neyman_pearson(data$UrbanInfluenceCode)
## X-squared = 284.55, df = 11, p-value < 2.2e-16
```

```
print("Bachelors Percentage")
```

```
## [1] "Bachelors Percentage"
```

```
chisq.test(neyman_pearson(data$BachelorsPercentage))
```

```
## Warning in chisq.test(neyman_pearson(data$BachelorsPercentage)): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  neyman_pearson(data$BachelorsPercentage)
## X-squared = 1323.3, df = 460, p-value < 2.2e-16
```

As we can see after using the Chi-squared test, Bachelors Percentage, Unemployment rate and Urban Influence Code have the largest influence on the outcome of the election^ median income also affects it, but not so heavily.

Naive Bayes

A Naive Bayes Classifier is a simple algorithm based on Bayes' theorem, commonly used for solving classification problems. In this context, a classification problem involves determining whether a given observation of parameters (e.g., UnemploymentRate, MedianIncome, etc.) should be classified as Democratic or Republican.

It is a **probabilistic classifier**, which means it predicts based on the probability of an observation belonging to each class. To compute it, this algorithm uses **Bayes' formula,**:

$$\mathrm{P}(\text{class} \mid \text{observation}) = \frac{\mathrm{P}(\text{observation} \mid \text{class})\mathrm{P}(\text{class})}{\mathrm{P}(\text{observation})}$$

In our case we have two classes: Democratic and Republican

Under the strong **independence** assumption, one can calculate $\mathrm{P}(\text{observation} \mid \text{class})$ as

$$\mathrm{P}(\text{observation}) = \prod_{i=1}^{n} \mathrm{P}(\text{feature}_i), \qquad \mathrm{P}(\text{observation} \mid \text{class}) = \prod_{i=1}^{n} \mathrm{P}(\text{feature}_i \mid \text{class}),$$

where n is the total number of features describing a given observation

Thus, $\mathrm{P}(\text{class}|\text{observation})$ now can be calculated as

$$\mathrm{P}(\text{class} \mid \text{observation}) = \mathrm{P}(\text{class}) \times \prod_{i=1}^{n} \frac{\mathrm{P}(\text{feature}_i \mid \text{class})}{\mathrm{P}(\text{feature}_i)}$$

Now let's make a Naive Bayes estimator which given Unemployment rate, Median income, Urban Influence code, Income percentile and percentage of people with Bachelor's degree will determine the dominant party for a given county

```
split <- sample.split(data$WinningParty, SplitRatio = 0.7)
train_data <- subset(data, split == TRUE)
test_data <- subset(data, split == FALSE)
```

```
naive_bayes <- function(unemployment, median, uic, percentile, bachelors, train_data) {
  total_count <- nrow(train_data)
  dem_count <- sum(train_data$WinningParty == 1)
  rep_count <- sum(train_data$WinningParty == 0)

  dem_prob <- dem_count / total_count
  rep_prob <- rep_count / total_count



  dem_prob <- dem_prob * mean(train_data$Unemployment[train_data$WinningParty == 1] == unemploymen
t)
  rep_prob <- rep_prob * mean(train_data$Unemployment[train_data$WinningParty == 0] == unemploymen
t)

  dem_prob <- dem_prob * mean(train_data$MedianIncome[train_data$WinningParty == 1] == median)
  rep_prob <- rep_prob * mean(train_data$MedianIncome[train_data$WinningParty == 0] == median)

  dem_prob <- dem_prob * mean(train_data$UrbanInfluenceCode[train_data$WinningParty == 1] == uic)
  rep_prob <- rep_prob * mean(train_data$UrbanInfluenceCode[train_data$WinningParty == 0] == uic)

  dem_prob <- dem_prob * mean(train_data$IncomePercentile[train_data$WinningParty == 1] == percent
ile)
  rep_prob <- rep_prob * mean(train_data$IncomePercentile[train_data$WinningParty == 0] == percent
ile)

  dem_prob <- dem_prob * mean(train_data$BachelorsPercentage[train_data$WinningParty == 1] == bach
elors)
  rep_prob <- rep_prob * mean(train_data$BachelorsPercentage[train_data$WinningParty == 0] == bach
elors)

  if (dem_prob > rep_prob) {
    return(1)
  } else {
    return(0)
  }
}
```

```
counter <- 0
for (i in 1:nrow(test_data)) {
  predicted_party <- naive_bayes(test_data$Unemployment[i],
                                 test_data$MedianIncome[i],
                                 test_data$UrbanInfluenceCode[i],
                                 test_data$IncomePercentile[i],
                                 test_data$BachelorsPercentage[i],
                                 train_data)
  if (predicted_party == test_data$WinningParty[i]) {
    counter <- counter + 1
  }
}

accuracy <- counter / nrow(test_data)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.8245989
```

Processing math: 100%

Logistic Regression

The linear probability model (LPM) is a simple approach to modeling binary outcomes, but it has notable disadvantages:

- Fitted probabilities can fall outside the range $[0, 1]$.
- The partial effect of any explanatory variable is constant.

To overcome these limitations, we use more sophisticated binary response models such as the **logit** model. This model ensures that probabilities are always between 0 and 1, providing a better theoretical foundation.

Response Probability

In a binary response model, the primary interest lies in the response probability:

$$P(y = 1 | x) = P(y = 1 | x_1, x_2, \dots, x_k),$$

where y is the dependent variable (in our case $WinningParty$) and $x_1, x_2, \dots, x_k$ represent explanatory variables (e.g., $MedianIncome$, $UnemploymentRate$, etc.).

Specifying the Logit Model

Logistic regression models the relationship between a binary dependent variable Y and a set of predictors $X_1, X_2, \dots, X_k$. The dependent variable Y takes values:

$$Y = \begin{cases} 1 & \text{(Event occurs, e.g., Democrat wins)} \\ 0 & \text{(Event does not occur, e.g., Republican wins)} \end{cases}$$

The logit model assumes that the log-odds (odds compare the probability of an event happening (P) to the probability of it not happening $\frac{P}{1-P}$) of the probability $P(Y = 1 | X)$ follows a specific functional form to ensure values are strictly between 0 and 1:

$$P(y = 1 | x) = G(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k),$$

where:

$$G(z) = \frac{\exp(z)}{1 + \exp(z)}.$$

This logistic function $G(z)$ is the **cumulative distribution function (CDF)** of a standard logistic random variable. It has the following properties:

- $0 < G(z) < 1$ for all real z,
- $G(z) \to 0$ as $z \to -\infty$,
- $G(z) \to 1$ as $z \to +\infty$.

The logit model is widely used because it provides a flexible and interpretable way to model probabilities.

Estimation Using Maximum Likelihood

The logit model is estimated using **Maximum Likelihood Estimation (MLE)**. Assume a random sample of size n. The likelihood function for n observations is:

$$L(\beta) = \prod_{i=1}^{n} \left[G(x_i\beta)^{y_i} \cdot \left(1 - G(x_i\beta)\right)^{1-y_i} \right],$$

Processing math: 100%

Taking the natural logarithm of the likelihood gives the **log-likelihood function**:

$$\ell(\beta) = \sum_{i=1}^{n} \left[y_i \log G(x_i \beta) + (1 - y_i) \log(1 - G(x_i \beta)) \right].$$

The MLE of β, denoted $\hat{\beta}$, maximizes the log-likelihood $\ell(\beta)$.

Advantages of the Logit Model

- Probabilities are bounded between 0 and 1.
- The relationship between predictors and the probability of the outcome is nonlinear, allowing for more flexibility.
- Coefficients can be interpreted as the effect on the log-odds (odds compare the probability of an event happening (P) to the probability of it not happening $\frac{P}{1-P}$) of the outcome.

```
logit_model <- glm(WinningParty ~ Unemployment + MedianIncome +
                   UrbanInfluenceCode + IncomePercentile + BachelorsPercentage,
                   data = train_data, family = binomial)

summary(logit_model)
```

```
##
## Call:
## glm(formula = WinningParty ~ Unemployment + MedianIncome + UrbanInfluenceCode +
##     IncomePercentile + BachelorsPercentage, family = binomial,
##     data = train_data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -6.635e+00  5.397e-01 -12.292  < 2e-16 ***
## Unemployment         4.641e-01  3.805e-02  12.199  < 2e-16 ***
## MedianIncome        -3.078e-05  8.219e-06  -3.745  0.00018 ***
## UrbanInfluenceCode  -1.564e-01  2.814e-02  -5.558 2.74e-08 ***
## IncomePercentile    -2.190e+00  4.227e-01  -5.181 2.21e-07 ***
## BachelorsPercentage  2.029e-01  1.225e-02  16.563  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2023.4  on 2179  degrees of freedom
## Residual deviance: 1213.3  on 2174  degrees of freedom
## AIC: 1225.3
##
## Number of Fisher Scoring iterations: 6
```

```
test_data$predicted_prob <- predict(logit_model, newdata = test_data, type = "response")

test_data$predicted_party <- ifelse(test_data$predicted_prob > 0.5, 1, 0)

confusion_matrix <- table(test_data$WinningParty, test_data$predicted_party)
print(confusion_matrix)
```

Processing math: 100%

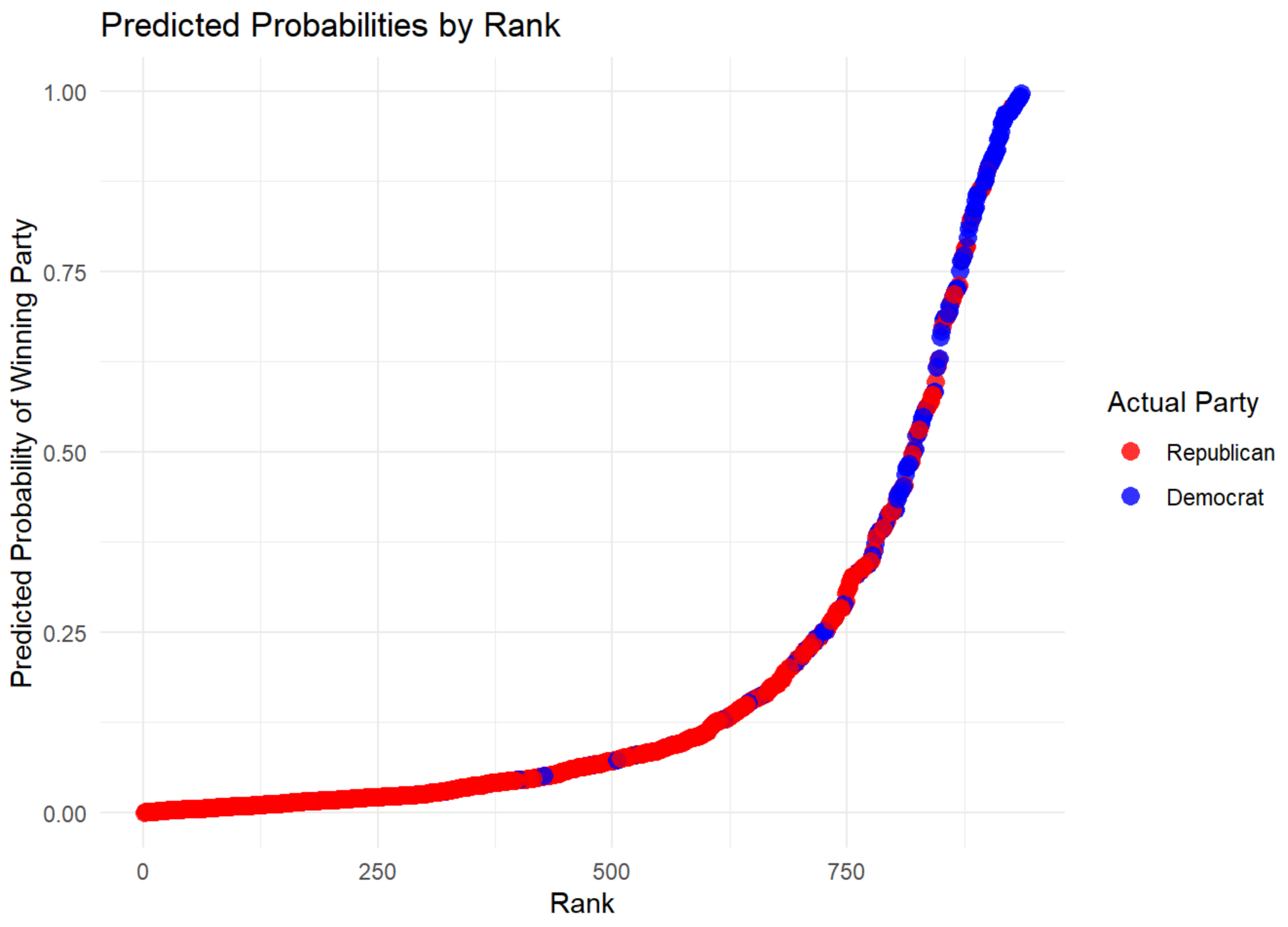
```
##
##       0   1
##   0 739  32
##   1  81  83
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.8791444
```

```
test_data$rank <- rank(test_data$predicted_prob)

ggplot(test_data, aes(x = rank, y = predicted_prob)) +
  geom_point(aes(color = as.factor(WinningParty)), alpha = .8, shape = 16, size = 3) +
  xlab("Rank") +
  ylab("Predicted Probability of Winning Party") +
  ggtitle("Predicted Probabilities by Rank") +
  theme_minimal() +
  scale_color_manual(values = c("red", "blue"),
                     name = "Actual Party",
                     labels = c("Republican", "Democrat"))
```



The graph shows the predicted probabilities of the winning party. Most of the counties painted Blue are close two 1, which means that the model is quite accurate, same with the counties painted Red.

Conclusion

Processing math: 100%

The logit model is a powerful tool for modeling binary outcomes, overcoming the limitations of the LPM while providing interpretable results.

Comparison of Two Models

The Naive Bayes estimator achieves an accuracy of 0.82570.8257, while the Logit model performs better with 0.89950.8995. This difference is due to Naive Bayes assuming independence between predictors, which often does not hold in real-world datasets. Violations of this assumption can limit its ability to model complex relationships accurately. On the other hand, the Logit model estimates the probability of outcomes without assuming predictor independence. By modeling the log-odds of the outcome as a linear function of predictors, it effectively captures dependencies and interactions, resulting in improved accuracy.