ACS Analysis

2024-03-05

ACS Analysis

Data Loading

first load the dataset and libraries

```
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
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(tidyr)
library(readr)
```

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

```
library(RColorBrewer)
library(corrplot)
```

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

```
## corrplot 0.92 loaded
```

```
library(confintr)
```

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

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.3.3
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(clustMixType)
```

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

```
library(Rtsne)
```

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

```
# Specify the path to the CSV file
file_path = "F:/pc docs/Project datasets/American Community Survey/acs.csv"

# Read the CSV file into a data frame
data = read.csv(file_path)

# View the first few rows of the data frame
head(data)
```

```
##   income         employment hrs_work  race age gender citizen time_to_work
## 1  60000 not in labor force       40 white  68 female     yes           NA
## 2      0 not in labor force       NA white  88   male     yes           NA
## 3     NA               <NA>       NA white  12 female     yes           NA
## 4      0 not in labor force       NA white  17   male     yes           NA
## 5      0 not in labor force       NA white  77 female     yes           NA
## 6   1700           employed       40 other  35 female     yes           15
##      lang married         edu disability   birth_qrtr
## 1 english      no     college         no jul thru sep
## 2 english      no hs or lower        yes jan thru mar
## 3 english      no hs or lower         no oct thru dec
## 4   other      no hs or lower         no oct thru dec
## 5   other      no hs or lower        yes jul thru sep
## 6   other     yes hs or lower        yes jul thru sep
```

income: Annual income. employment: Employment status. hrs_work: Hours worked per week. race: Race. age: Age in years. gender: Gender. citizen: U.S. citizenship status. time_to_work: Travel time to work in minutes. lang: Language spoken at home. married: Marital status. edu: Education level. disability: Disability status. birth_qrtr: Quarter of the year the person was born (e.g., Jan thru Mar).

Let's look at the dimensions of the dataset.

```
# Dimensions of the dataframe
dim(data)
```

```
## [1] 2000   13
```

Let's look at some summary statistics.

```
# Summary including some basic statistics
summary(data)
```

```
##      income        employment           hrs_work         race
##  Min.   :     0   Length:2000        Min.   : 1.00   Length:2000
##  1st Qu.:     0   Class :character   1st Qu.:32.00   Class :character
##  Median :  3000   Mode  :character   Median :40.00   Mode  :character
##  Mean   : 23600                      Mean   :37.98
##  3rd Qu.: 33700                      3rd Qu.:40.00
##  Max.   :450000                      Max.   :99.00
##  NA's   :377                         NA's   :1041
##       age           gender            citizen           time_to_work
##  Min.   : 0.00   Length:2000        Length:2000        Min.   :  1
##  1st Qu.:19.75   Class :character   Class :character   1st Qu.: 10
##  Median :40.00   Mode  :character   Mode  :character   Median : 20
##  Mean   :40.22                                         Mean   : 26
##  3rd Qu.:59.00                                         3rd Qu.: 30
##  Max.   :94.00                                         Max.   :163
##                                                        NA's   :1217
##      lang             married              edu             disability
##  Length:2000        Length:2000        Length:2000        Length:2000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   birth_qrtr
##  Length:2000
##  Class :character
##  Mode  :character
##
##
##
##
```

Let's look at the unique values for the categorical variables.

```
# Initialize an empty list to store the unique values
unique_values_list = list()

# Iterate through each column of the dataframe
for(col_name in names(data)) {
  # Check if the column contains character data
  if(is.character(data[[col_name]])) {
    # Store unique values for the column in the list
    unique_values_list[[col_name]] = unique(data[[col_name]])
  }
}

print(unique_values_list)
```

```
## $employment
## [1] "not in labor force" NA                  "employed"
## [4] "unemployed"
##
## $race
## [1] "white" "other" "asian" "black"
##
## $gender
## [1] "female" "male"
##
## $citizen
## [1] "yes" "no"
##
## $lang
## [1] "english" "other"   NA
##
## $married
## [1] "no"  "yes"
##
## $edu
## [1] "college"     "hs or lower" "grad"        NA
##
## $disability
## [1] "no"  "yes"
##
## $birth_qrtr
## [1] "jul thru sep" "jan thru mar" "oct thru dec" "apr thru jun"
```

Data Cleaning

Let's check for missing values.

```
# Count the missing values in each column
sapply(data, function(x) sum(is.na(x)))
```

```
##       income   employment     hrs_work         race          age       gender
##          377          395         1041            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0         1217          105            0           58            0
##   birth_qrtr
##            0
```

We must figure out a meaningful way to handle missing data.

Simply filling in the missing values with mode for categorical variables and median for numerical variables will change the outcome of the analyses in a way the skews the meaning of the data.

for income, missing data for ‘unemployed’ or ‘not in labor force’ will be filled with 0, and ‘employed’ will be filled with median.

for employment, all missing data will be filled with ‘not in labor force’.

for hours worked, all missing data where employment is ‘not in labor force’ or ‘unemployed’ will be filled with 0, and all missing data where employment is ‘employed’ will be filled with 40.

time to work will be filled with the median if ‘employed’, otherwise it will be 0.

for language, all missing data will be filled as ‘english’.

for education, all missing data will be filled as ‘hs or lower’.

```
# Calculate the median income for employed individuals ahead of time
median_income_employed <- median(data$income[data$employment == 'employed'], na.rm = TRUE)

data <- data %>%
  # Adjust employment status and income together when both are missing
  mutate(employment = case_when(
    is.na(employment) & !is.na(income) & income != 0 ~ 'employed',
    is.na(employment) & (is.na(income) | income == 0) ~ 'not in labor force',
    TRUE ~ as.character(employment)
  ),
  income = if_else(is.na(employment) & is.na(income), 0, income)) %>%

  # Then, adjust income imputation based on updated employment status
  mutate(income = case_when(
    is.na(income) & (employment %in% c('unemployed', 'not in labor force')) ~ 0,
    is.na(income) & employment == 'employed' ~ median_income_employed,
    TRUE ~ income
  )) %>%

  # Fill missing hrs_work based on employment status
  mutate(hrs_work = case_when(
    is.na(hrs_work) & (employment %in% c('not in labor force', 'unemployed')) ~ 0,
    is.na(hrs_work) & employment == 'employed' ~ 40,
    TRUE ~ hrs_work
  )) %>%

  # Adjust time_to_work imputation based on employment status
  mutate(time_to_work = case_when(
    employment == 'employed' & is.na(time_to_work) ~ median(data$time_to_work[data$employment ==
'employed'], na.rm = TRUE),
    is.na(time_to_work) ~ 0,
    TRUE ~ time_to_work
  )) %>%

  # Fill missing lang with 'english'
  mutate(lang = if_else(is.na(lang), 'english', lang)) %>%

  # Fill missing edu with 'hs or lower'
  mutate(edu = if_else(is.na(edu), 'hs or lower', edu))

# Check the structure and summary to confirm changes
summary(data)
```

```
##      income        employment           hrs_work        race
##  Min.   :     0   Length:2000        Min.   : 0.00   Length:2000
##  1st Qu.:     0   Class :character   1st Qu.: 0.00   Class :character
##  Median :     0   Mode  :character   Median : 0.00   Mode  :character
##  Mean   : 19151                      Mean   :18.21
##  3rd Qu.: 24425                      3rd Qu.:40.00
##  Max.   :450000                      Max.   :99.00
##       age           gender            citizen           time_to_work
##  Min.   : 0.00   Length:2000        Length:2000        Min.   :  0.00
##  1st Qu.:19.75   Class :character   Class :character   1st Qu.:  0.00
##  Median :40.00   Mode  :character   Mode  :character   Median :  0.00
##  Mean   :40.22                                         Mean   : 10.78
##  3rd Qu.:59.00                                         3rd Qu.: 20.00
##  Max.   :94.00                                         Max.   :163.00
##      lang             married              edu             disability
##  Length:2000        Length:2000        Length:2000        Length:2000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##   birth_qrtr
##  Length:2000
##  Class :character
##  Mode  :character
##
##
##
```

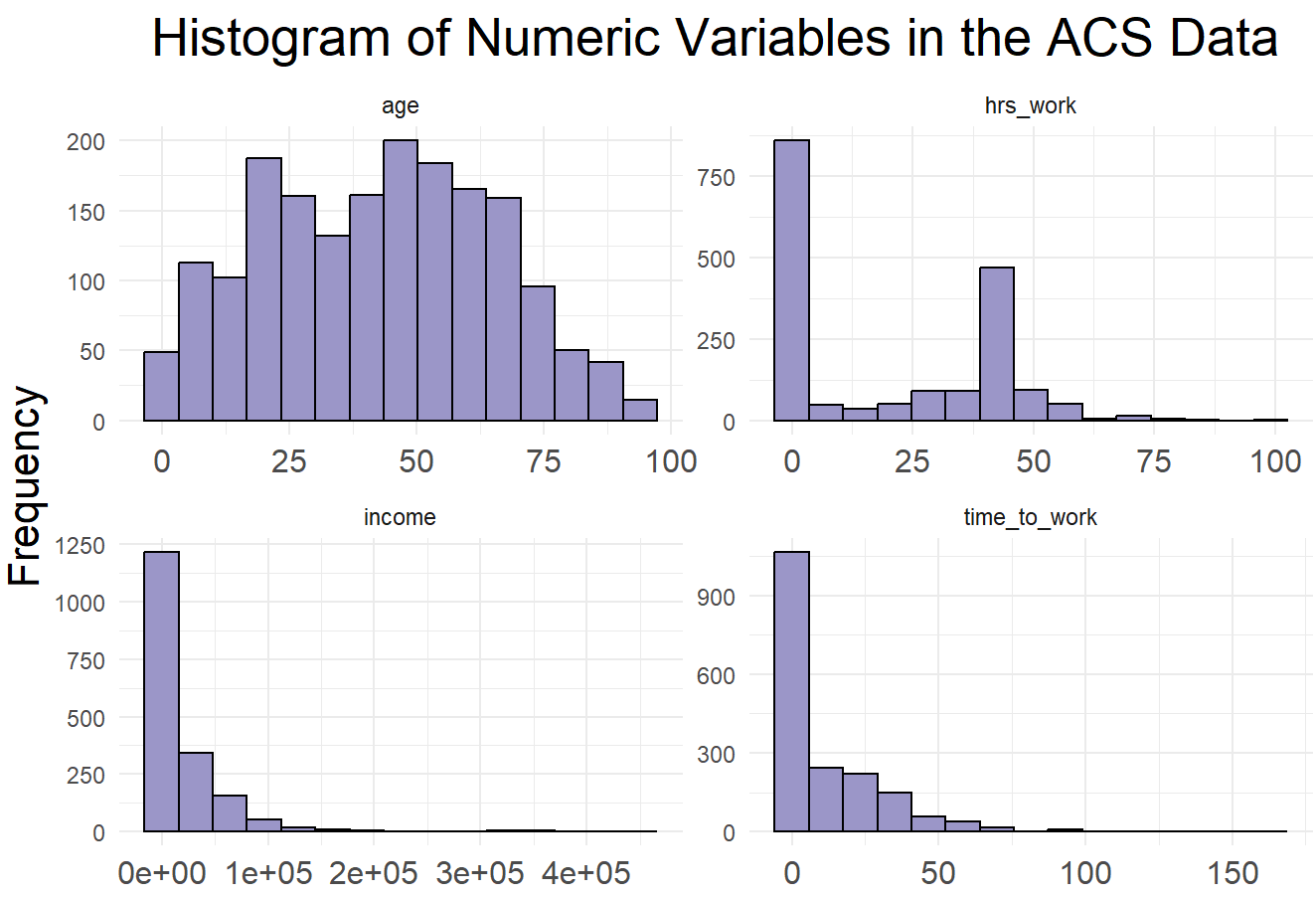
Let’s remove duplicate entries.

```
data = unique(data)
```

Data Visualization and Exploratory Data Analysis (EDA)

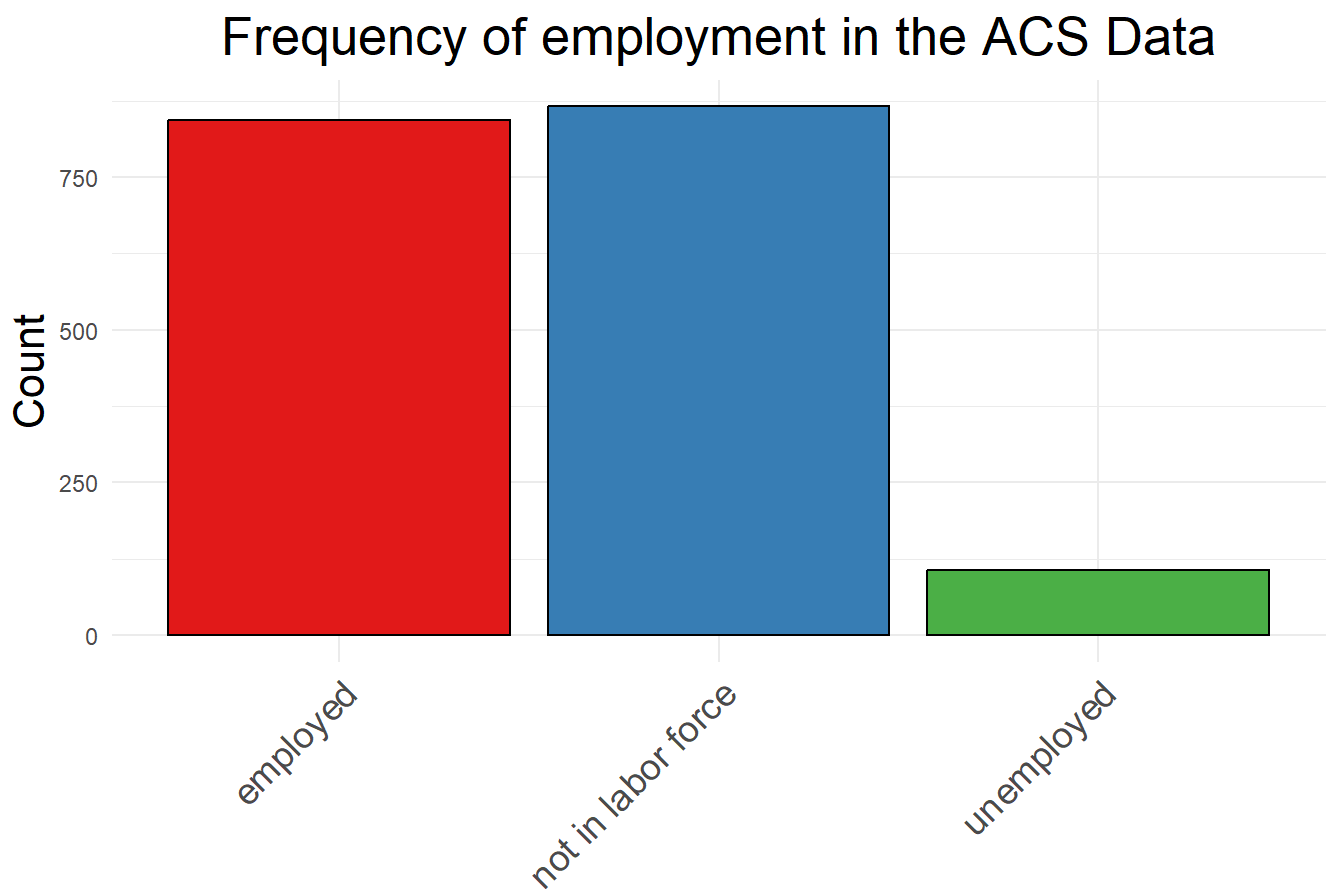
Let’s plot the frequency histograms for the numeric variables.

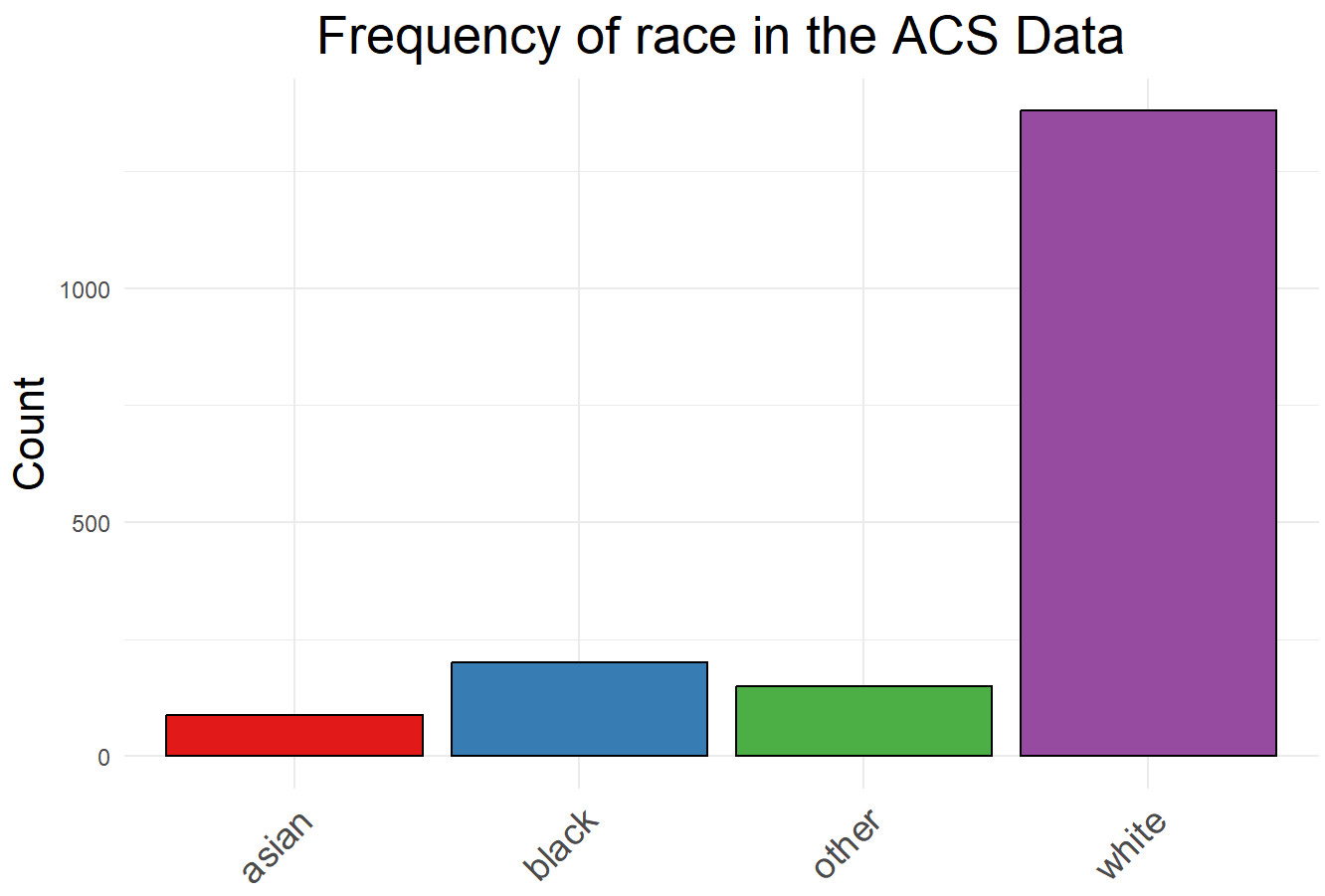
```
# Select only the numeric columns identified: income, hrs_work, age, and time_to_work
numeric_data = data %>%
  select(income, hrs_work, age, time_to_work)

numeric_data_long = numeric_data %>%
  pivot_longer(cols = everything(), names_to = "Variable", values_to = "Value")

p = ggplot(numeric_data_long, aes(x = Value)) +
  geom_histogram(bins = 15, fill = "#9e9ac8", color = "black") +
  facet_wrap(~Variable, scales = "free") +
  theme_minimal() +
  labs(title = "Histogram of Numeric Variables in the ACS Data", x = "", y = "Frequency") +
  theme(axis.text.x = element_text(size = 12),  # Increase x-axis tick labels size
        axis.title.x = element_text(size = 18),  # Increase x-axis title size
        axis.title.y = element_text(size = 16),  # Increase y-axis title size
        plot.title = element_text(size = 20, hjust = 0.5))  # Increase plot title size and cente
r it

print(p)
```

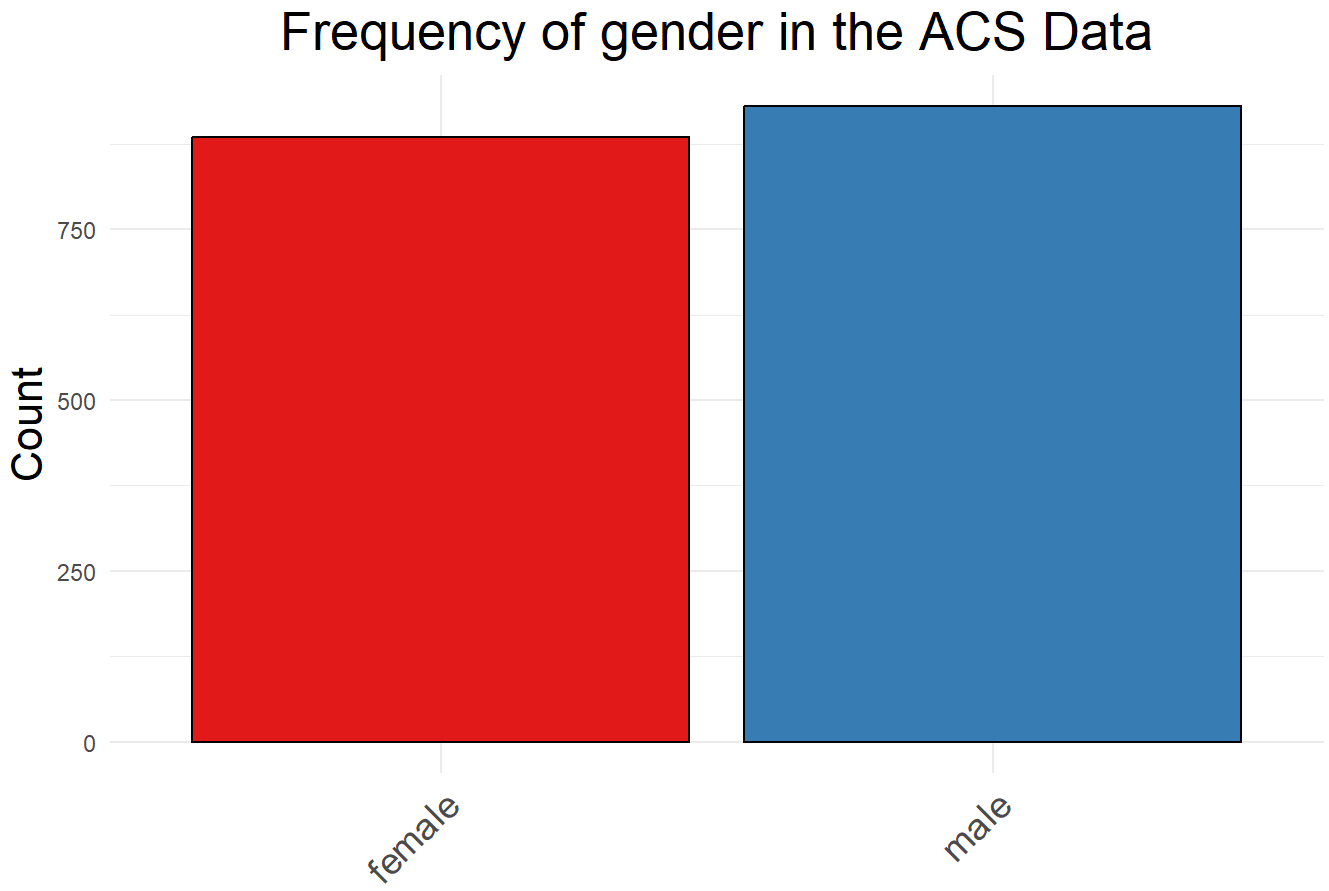


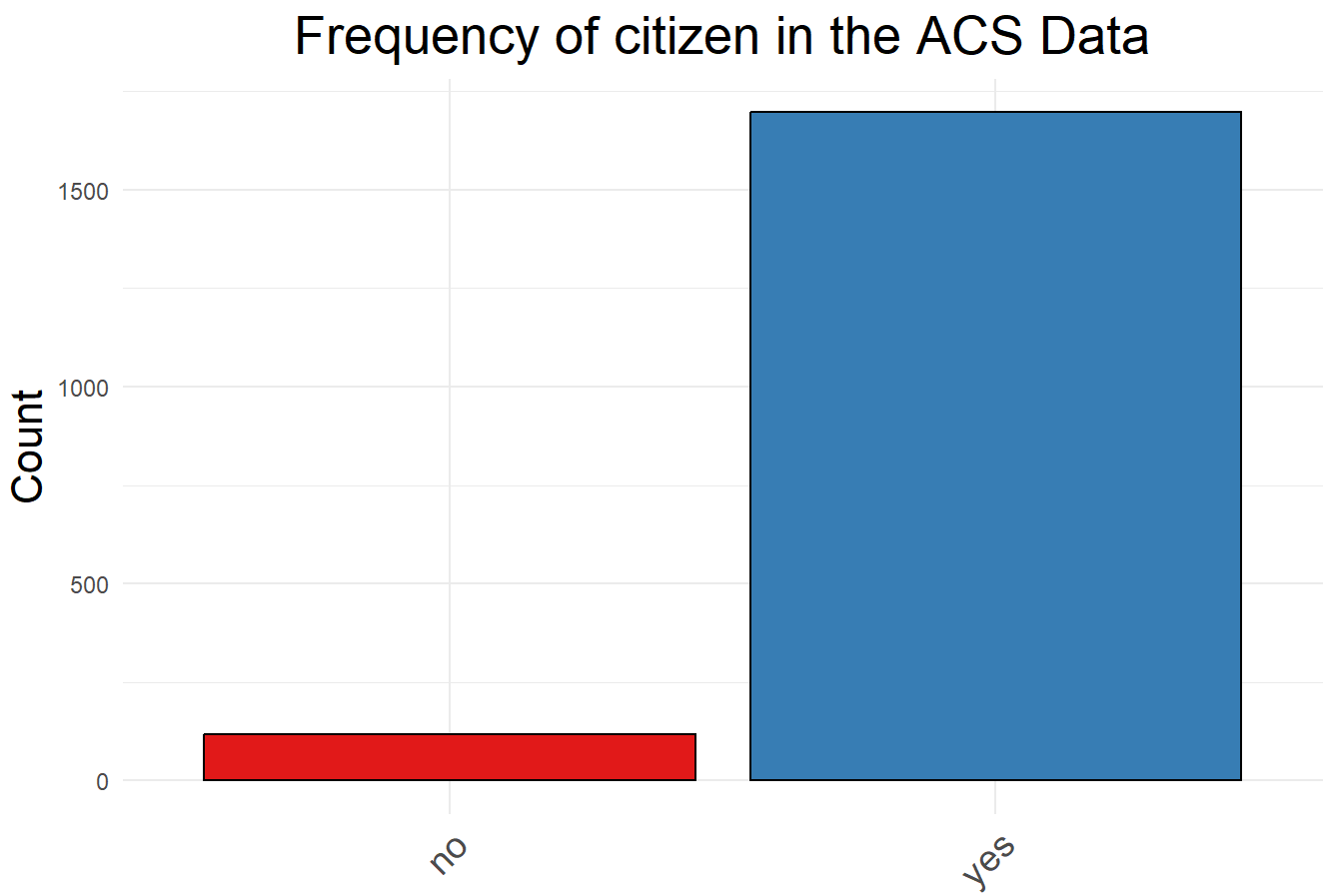
Let's plot the bar graphs for the categorical variables.

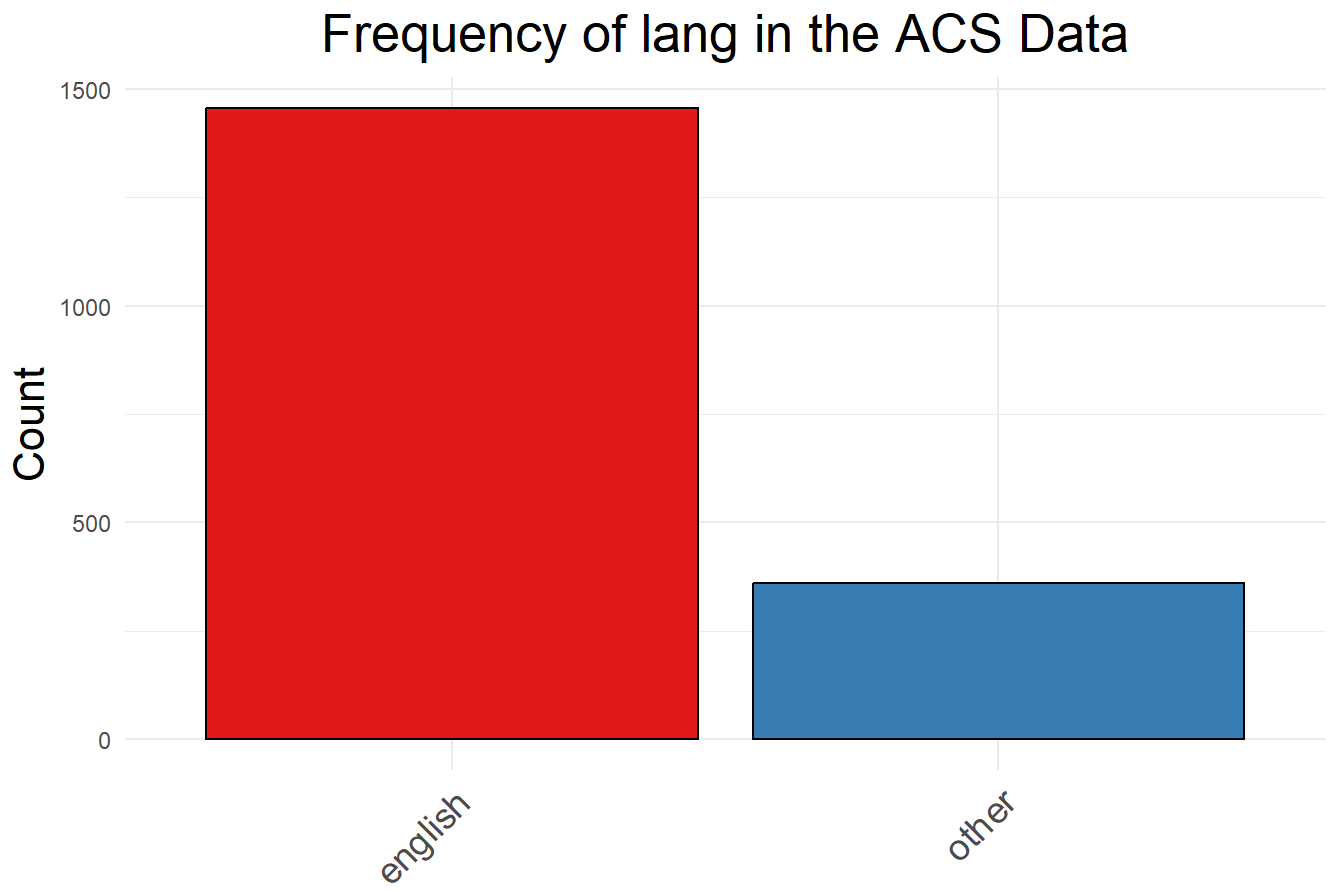
```
# Identify categorical variables based on their data type
categorical_variables = names(select_if(data, is.character))

# Create a function to plot a single categorical variable using a color-blind-friendly palette
plot_categorical_variable = function(data, variable_name) {
  # Convert the variable to a factor for better control over the fill aesthetic
  data[[variable_name]] = as.factor(data[[variable_name]])

  plot = ggplot(data, aes(x = !!sym(variable_name), fill = !!sym(variable_name))) +
    geom_bar(color = "black") +  # Outline color
    scale_fill_brewer(palette = "Set1") +  # Color-blind-friendly palette
    theme_minimal() +
    labs(title = paste("Frequency of", variable_name, "in the ACS Data"), x = "", y = "Count") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 14), # Increase axis text siz
e
          axis.title.x = element_text(size = 18),  # Increase x-axis title size
          axis.title.y = element_text(size = 16),  # Increase y-axis title size
          plot.title = element_text(size = 20, hjust = 0.5), # Increase plot title size and cent
er it
          legend.position = "none")  # Hide legend since it's redundant

  # Print the plot
  print(plot)
}

# Loop through each categorical variable and plot it
for(variable_name in categorical_variables) {
  plot_categorical_variable(data, variable_name)
}
```

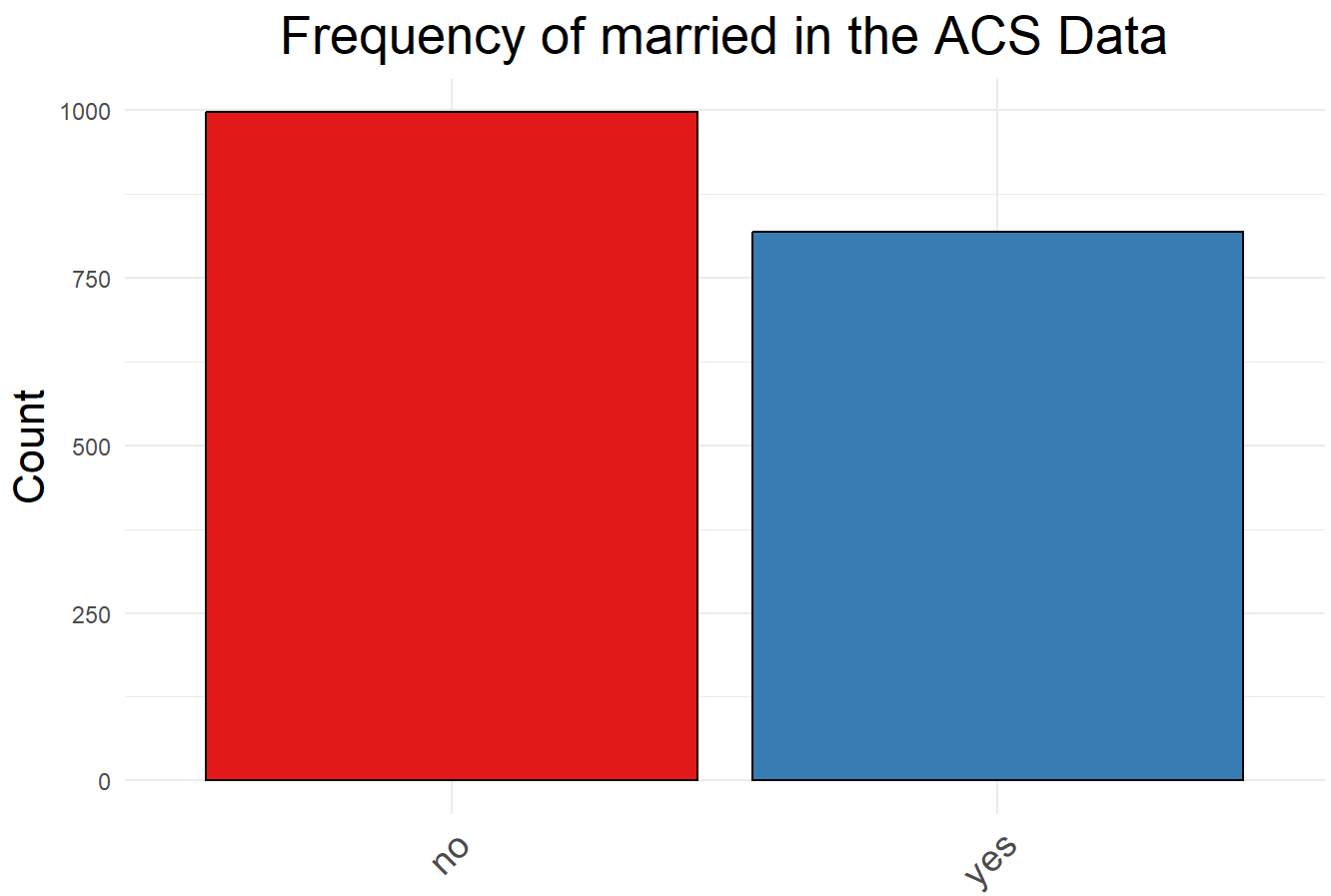

Frequency of employment in the ACS Data
Count
750
500
250
0
employed
not in labor force
unemployed

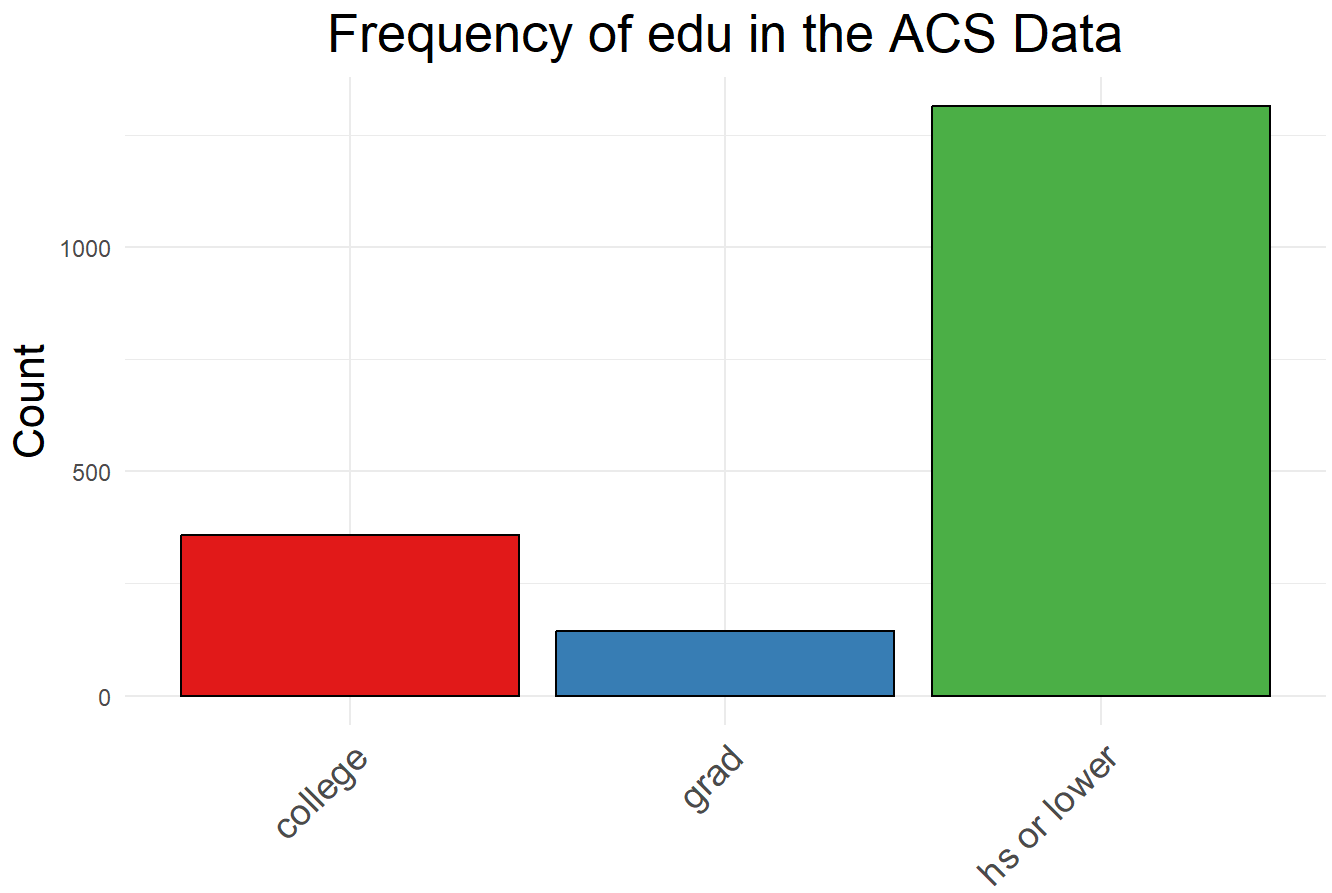

Frequency of race in the ACS Data
Count
1000
500
0
asian
black
other
white

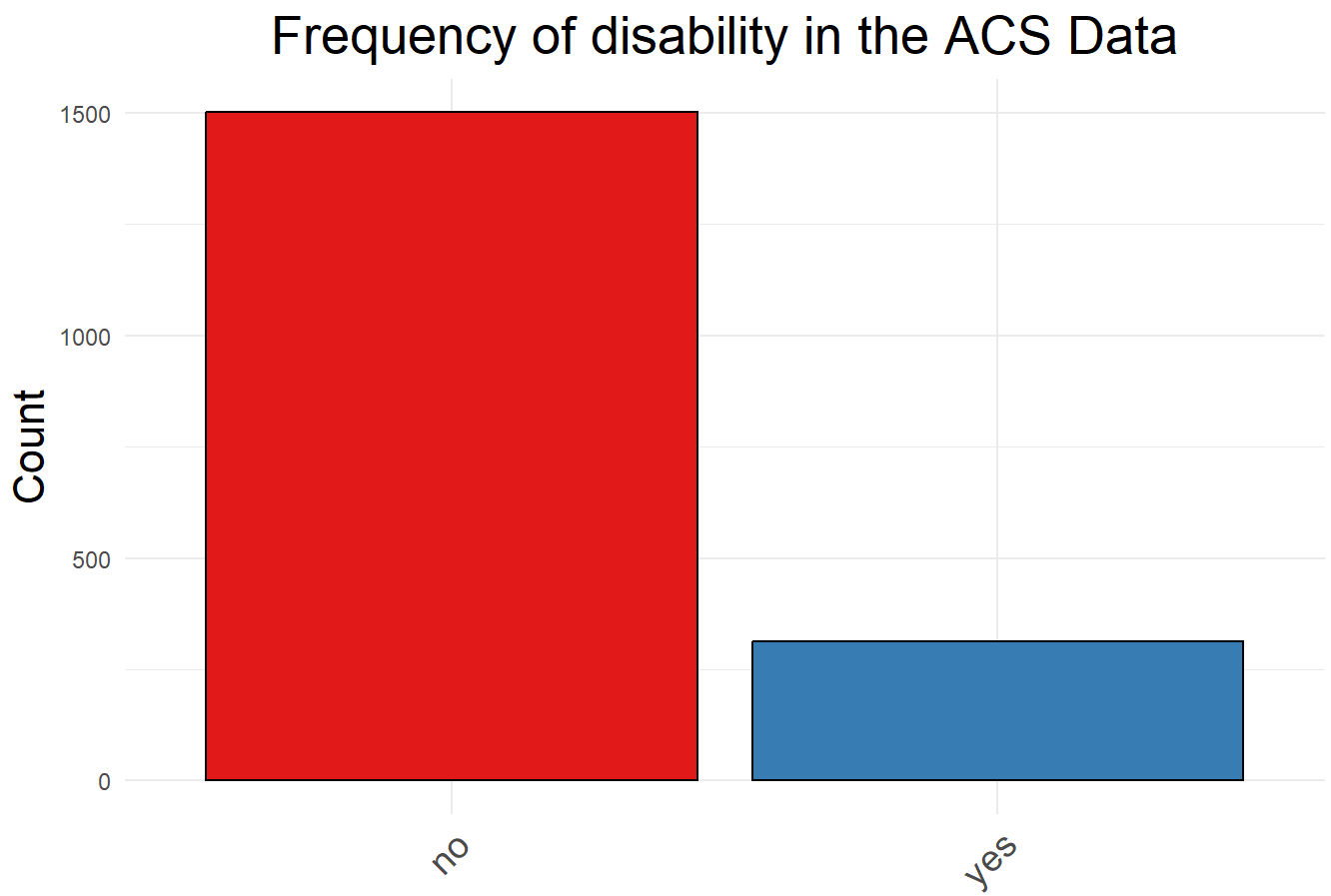

Frequency of gender in the ACS Data
Count
750
500
250
0
female
male

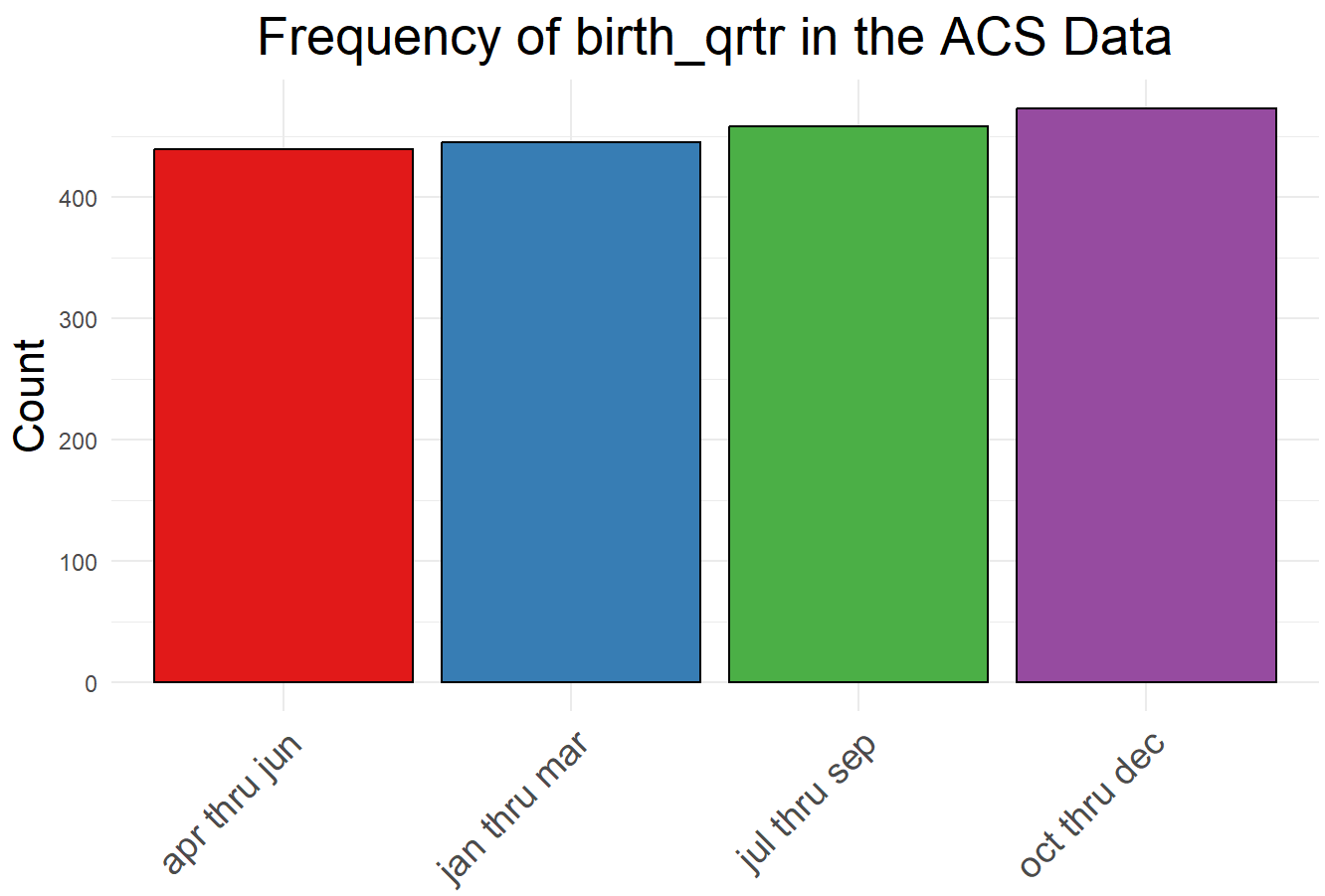

Frequency of citizen in the ACS Data
Count
1500
1000
500
0
no
yes


Frequency of lang in the ACS Data
Count
1500
1000
500
0
english
other


Frequency of married in the ACS Data
Count
1000
750
500
250
0
no
yes

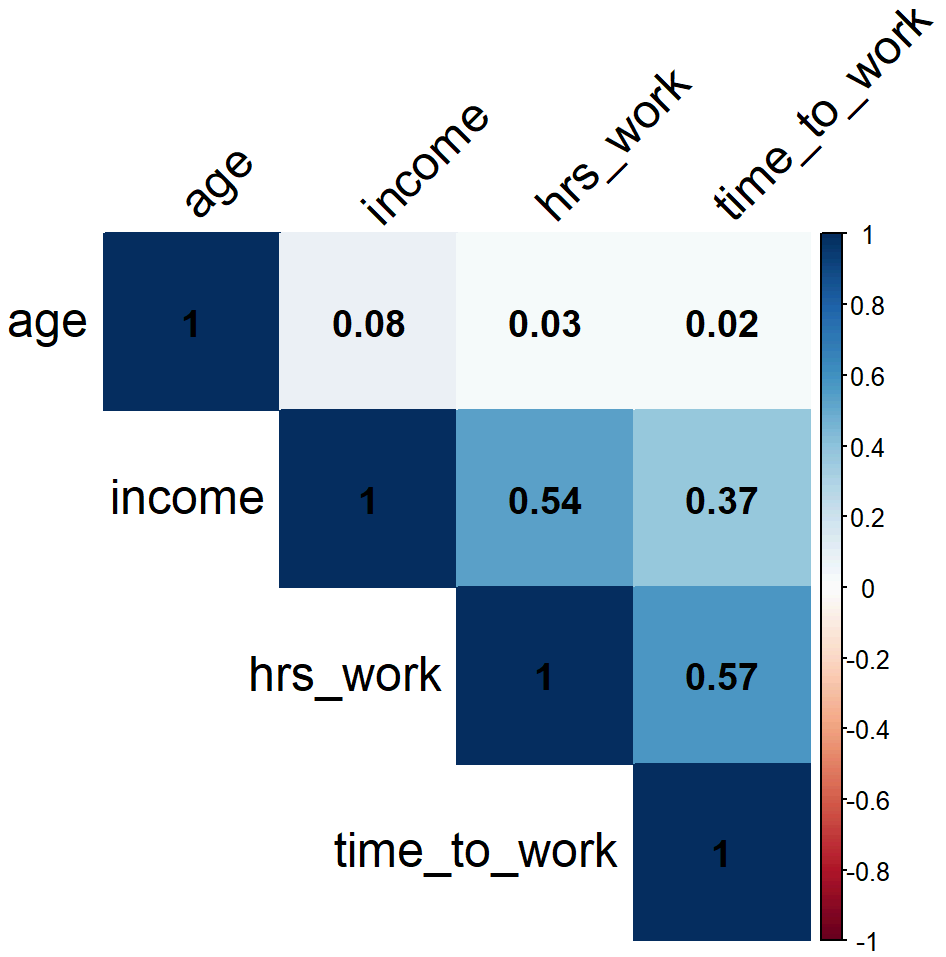

Frequency of edu in the ACS Data
Count
1000
500
0
college
grad
hs or lower


Frequency of disability in the ACS Data
Count
1500
1000
500
0
no
yes



Let's create a correlation matrix for the numeric variables.

```
# Select only numeric variables for the correlation matrix
numeric_data = select_if(data, is.numeric)

# Calculate the correlation matrix, handling missing values by excluding them
cor_matrix = cor(numeric_data)

# Plot the correlation heatmap with increased text sizes
corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45, # Text label color and rotation
         addCoef.col = "black", # Add correlation coefficients to the plot
         tl.cex = 1.5, # Increase size of text labels (axis labels)
         number.cex = 1.2) # Increase size of correlation coefficients
```



We will convert the categorical variables to factors.

Education is an ordered factor where ‘hs or lower’ < ‘college’ < ‘grad’.

```
# Convert categorical variables to factors
# and specify levels for 'edu' to treat it as an ordinal factor
data = data %>%
  mutate(across(where(is.character), as.factor), # Convert all character columns to factors
         edu = factor(edu, levels = c('hs or lower', 'college', 'grad'), ordered = TRUE)) # Make
'edu' an ordered factor
```

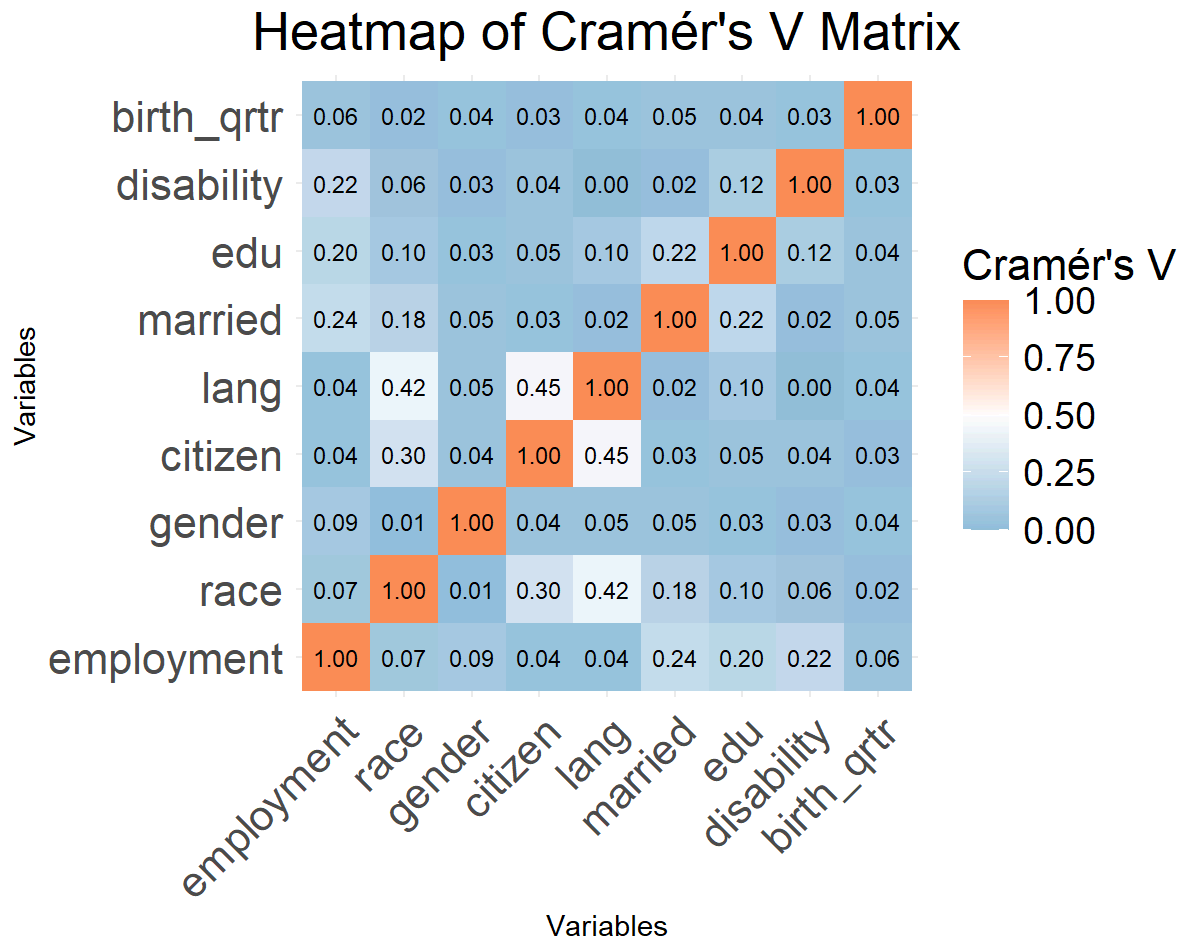
We will apply Cramer’s V to the categorical variables to see correlations as measured on a scale of 0 to 1.

```
# Re-identify categorical variables now that they are explicitly factors
categorical_variables = select_if(data, is.factor)

calculate_cramers_v = function(data) {
  var_names = names(data)
  results = matrix(NA, nrow = length(var_names), ncol = length(var_names), dimnames = list(var_n
ames, var_names))

  for(i in seq_along(var_names)) {
    for(j in seq_along(var_names)) {
      if(i == j) {
        results[i, j] = 1
      } else if (i < j) {
        # Using CramersV from the confintr package
        results[i, j] = results[j, i] = cramersv(table(data[[var_names[i]]], data[[var_names
[j]]]))
      }
    }
  }

  return(results)
}

# Reapply the function on the identified categorical variables
cramers_v_matrix = calculate_cramers_v(categorical_variables)

# View the results
print(cramers_v_matrix)
```

```
##            employment       race     gender    citizen         lang    married
## employment 1.00000000 0.06929768 0.09431158 0.04000171 0.0391331383 0.23855324
## race       0.06929768 1.00000000 0.01279150 0.30146751 0.4237397141 0.18088597
## gender     0.09431158 0.01279150 1.00000000 0.04230598 0.0524389869 0.05285678
## citizen    0.04000171 0.30146751 0.04230598 1.00000000 0.4525070055 0.03062486
## lang       0.03913314 0.42373971 0.05243899 0.45250701 1.0000000000 0.02167533
## married    0.23855324 0.18088597 0.05285678 0.03062486 0.0216753280 1.00000000
## edu        0.19626968 0.09660121 0.03378446 0.04972176 0.0979278138 0.21890012
## disability 0.22321230 0.06339455 0.02515100 0.04347446 0.0003288157 0.02071057
## birth_qrtr 0.06154632 0.02416906 0.03789071 0.02511527 0.0408473776 0.05218724
##                   edu   disability birth_qrtr
## employment 0.19626968 0.2232122996 0.06154632
## race       0.09660121 0.0633945500 0.02416906
## gender     0.03378446 0.0251510038 0.03789071
## citizen    0.04972176 0.0434744616 0.02511527
## lang       0.09792781 0.0003288157 0.04084738
## married    0.21890012 0.0207105744 0.05218724
## edu        1.00000000 0.1195682319 0.04378658
## disability 0.11956823 1.0000000000 0.03203349
## birth_qrtr 0.04378658 0.0320334885 1.00000000
```

Let’s create a heatmap of the correlations from the Cramer’s V matrix.

```
# Convert the matrix to a data frame for plotting with ggplot2
melted_cramers_v_matrix = melt(cramers_v_matrix)

# Plotting the heatmap with Cramér's V values
ggplot(melted_cramers_v_matrix, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() + # Use geom_tile() for heatmap representation
  geom_text(aes(label = sprintf("%.2f", value)), color = "black", size = 3) + # Increase Cramé
r's V values label size
  scale_fill_gradient2(low = "#91bfdb", high = "#fc8d59", mid = "white", midpoint = 0.5,
                       limits = c(0, 1), space = "Lab", name="Cramér's V") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1, size = 16), # Increase x-ax
is text size
        axis.text.y = element_text(size = 16), # Increase y-axis text size
        plot.title = element_text(size = 20, hjust = 0.5), # Increase chart title size
        legend.title = element_text(size = 16), # Increase legend title size
        legend.text = element_text(size = 14)) + # Increase legend text size
  labs(title = "Heatmap of Cramér's V Matrix", x = "Variables", y = "Variables") +
  coord_fixed()
```



Data Standardization

Now we will standardize the variables to prepare for modeling using unsupervised machine learning, specifically k-prototype clustering (a version of k-means that handles both numerical and categorical variables).

```
# Standardize only the numeric columns
data_standardized = data %>%
  mutate(across(where(is.numeric), scale)) %>%
  # Ensure factor variables remain untouched
  mutate(across(where(is.factor), as.factor))
```

Clustering Parameter Evaluation

We will create an elbow plot to visualize the number of clusters vs the total within sum of squares (TWSS).

Look for the “elbow” in the plot where the rate of decrease in TWSS sharply changes. This point suggests adding more clusters doesn’t significantly improve the fit.

```
set.seed(1) # For reproducibility

# Calculate TWSS for a range of cluster numbers
twss = numeric(20)
for (k in 1:20) {
  set.seed(1)
  model = kproto(x = data_standardized, k = k)
  twss[k] = model$tot.withinss
}
```

```
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
```

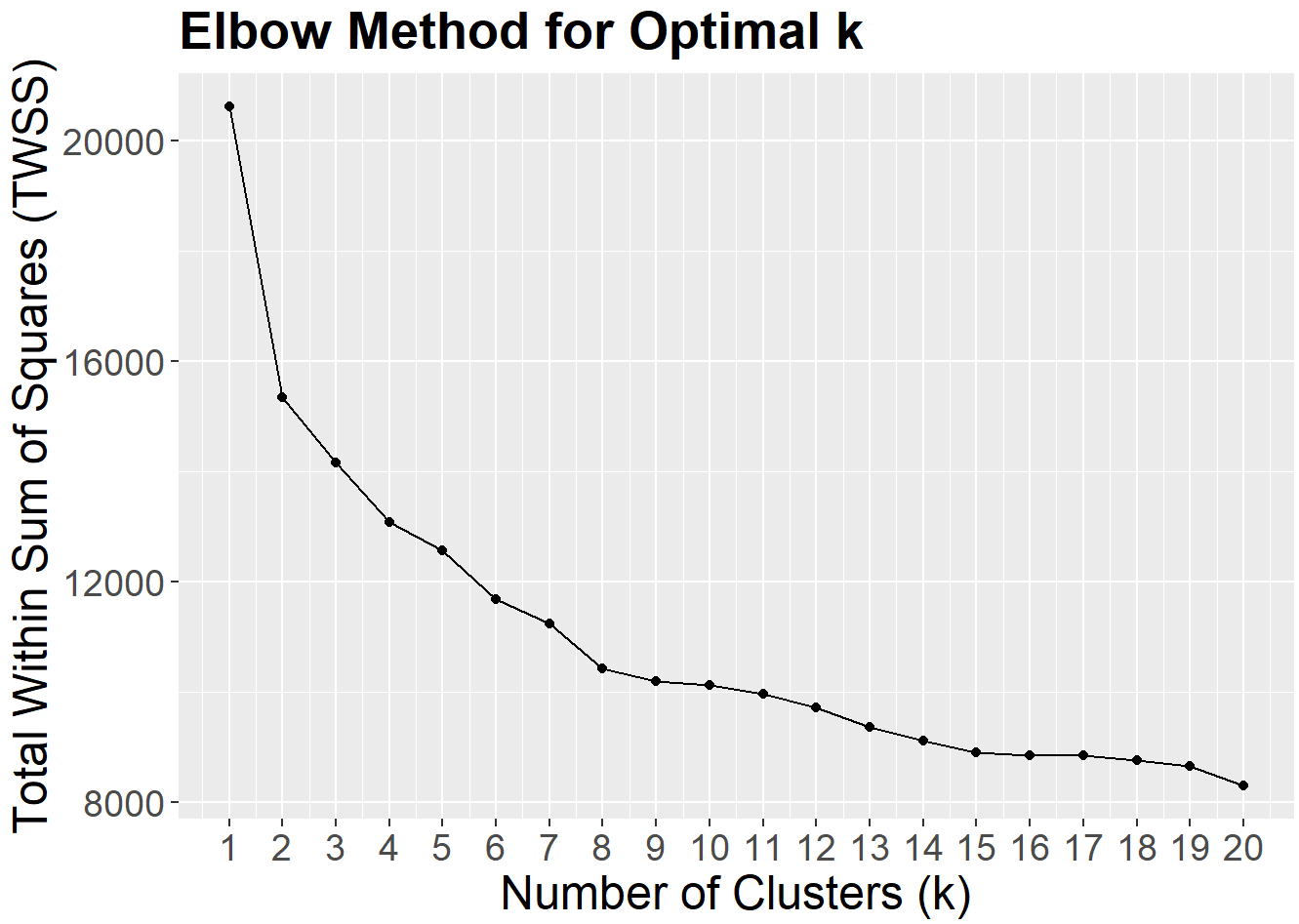
```
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
```

```
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
```

```
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
```

```
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
##
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
```

```
# Plot the TWSS (elbow method) with larger text
k_values = 1:20
ggplot(data.frame(k = k_values, TWSS = twss), aes(x = k, y = TWSS)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = k_values) +
  labs(title = "Elbow Method for Optimal k", x = "Number of Clusters (k)", y = "Total Within Sum
of Squares (TWSS)") +
  theme(text = element_text(size = 16), # Increase general text size
        plot.title = element_text(size = 20, face = "bold"), # Increase and bold plot title
        axis.title = element_text(size = 18), # Increase axis title text size
        axis.text = element_text(size = 14)) # Increase axis text size
```



Final Clustering Algorithm

We will use the optimal number of clusters from the plot above to apply to the final clustering algorithm.

```
k_optimal = 8  # Seemingly observed optimal number of clusters

# Fit the k-prototypes model with the optimal number of clusters
set.seed(1)
final_model = kproto(x = data_standardized, k = k_optimal)
```

```
## # NAs in variables:
##       income   employment     hrs_work         race          age       gender
##            0            0            0            0            0            0
##      citizen time_to_work         lang      married          edu   disability
##            0            0            0            0            0            0
##   birth_qrtr
##            0
## 0 observation(s) with NAs.
##
## Estimated lambda: 2.334942
##
## 0 observation(s) with NAs.
```

```
# Print the clustering result
print(final_model)
```

```
## Distance type: standard
##
## Numeric predictors: 4
## Categorical predictors: 9
## Lambda: 2.334942
##
## Number of Clusters: 8
## Cluster sizes: 307 211 341 23 70 286 183 394
## Within cluster error: 1607.54 1181.598 1706.737 183.6964 639.1009 1663.44 1111.977 2334.635
##
## Cluster prototypes:
##        income         employment   hrs_work  race        age gender citizen
## 1  0.39398909           employed  1.0577156 white  0.1427647   male     yes
## 2 -0.46180223 not in labor force -0.8470811 white -1.0186970 female     yes
## 3 -0.45674722 not in labor force -0.8042993 white -0.7179534   male     yes
## 4  6.61073673           employed  1.3531888 white  0.3234510   male     yes
## 5  0.66067973           employed  1.0808843 white  0.2278803   male     yes
## 6 -0.03836948           employed  0.6685307 white -0.2664448 female     yes
## 7  0.69930972           employed  0.9579922 white  0.1676145 female     yes
## 8 -0.46461450 not in labor force -0.8756754 white  1.1118710 female     yes
##   time_to_work    lang married         edu disability   birth_qrtr
## 1    0.3554818 english     yes hs or lower         no apr thru jun
## 2   -0.6018171   other      no hs or lower         no apr thru jun
## 3   -0.5758442 english      no hs or lower         no oct thru dec
## 4    0.7085056 english     yes        grad         no jan thru mar
## 5    3.3641199 english     yes hs or lower         no jul thru sep
## 6    0.3286065 english      no hs or lower         no jul thru sep
## 7    0.5701318 english     yes     college         no jan thru mar
## 8   -0.5986960 english     yes hs or lower         no jan thru mar
```

Let's add the cluster labels to a new dataframe.

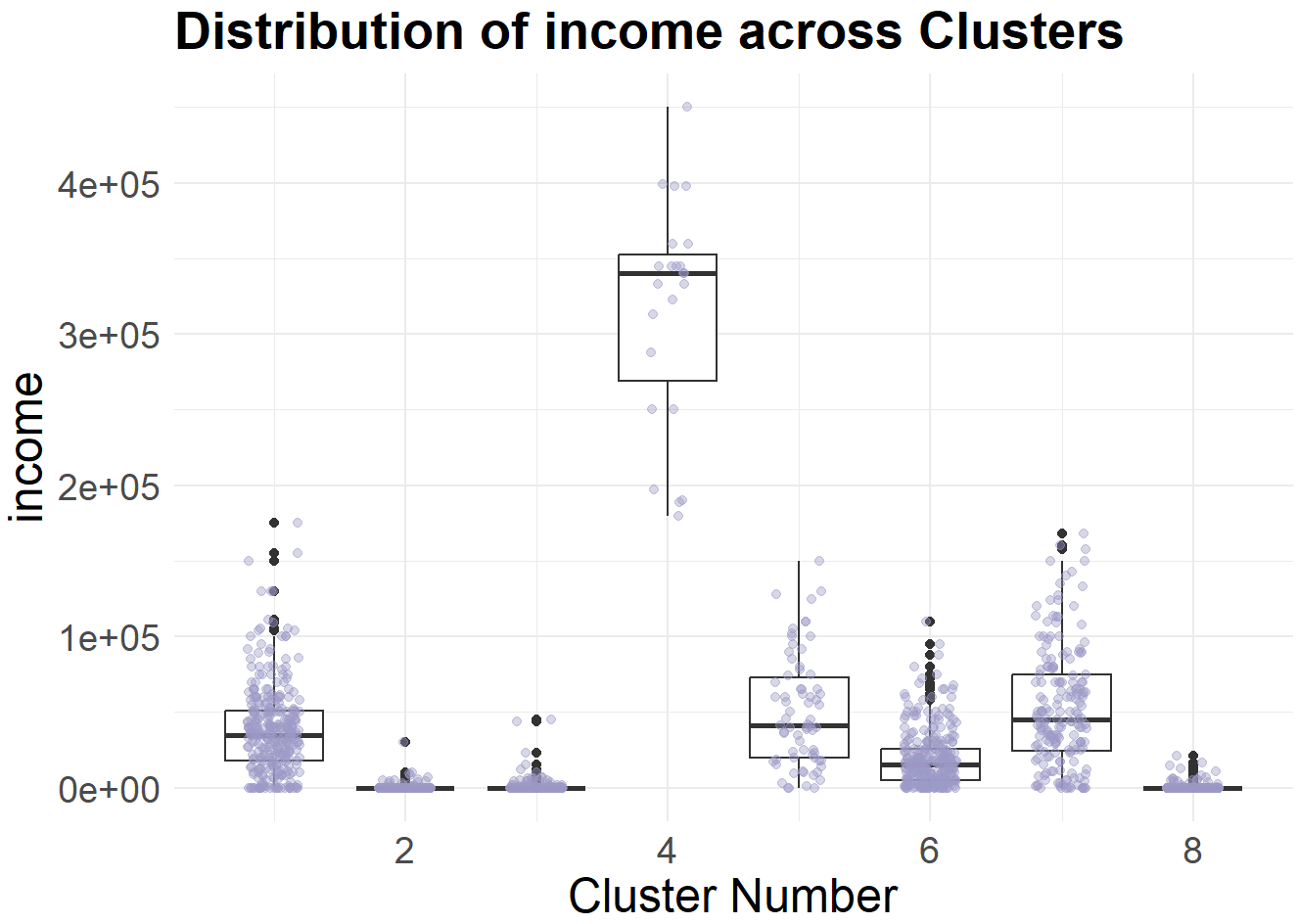
```
# Create a new dataframe
data_clust = data

# Add the clusters from the final model to them
data_clust$cluster = final_model$cluster

# Create separate cluster variable
clusters = final_model$cluster
```

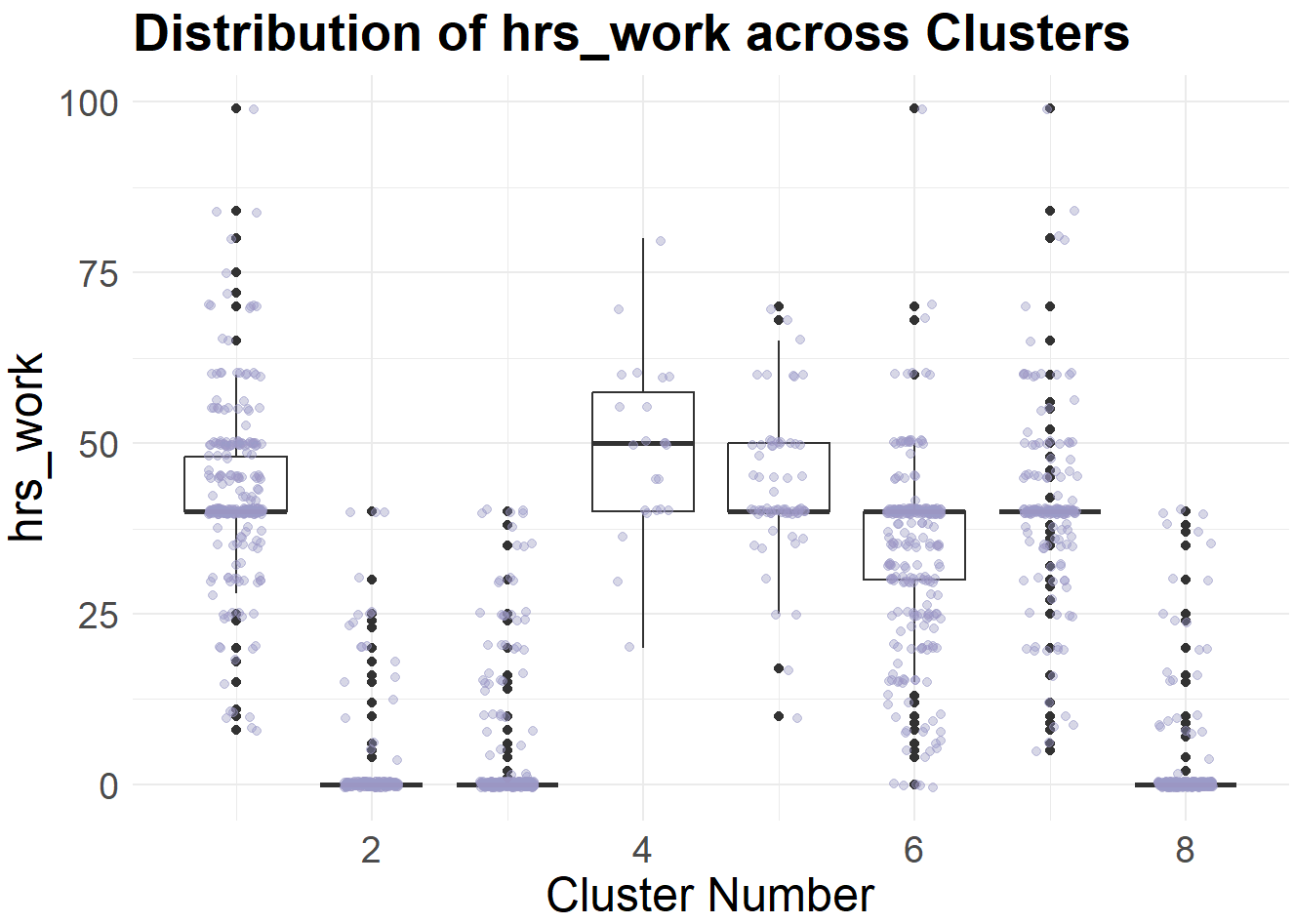
Data Visualization

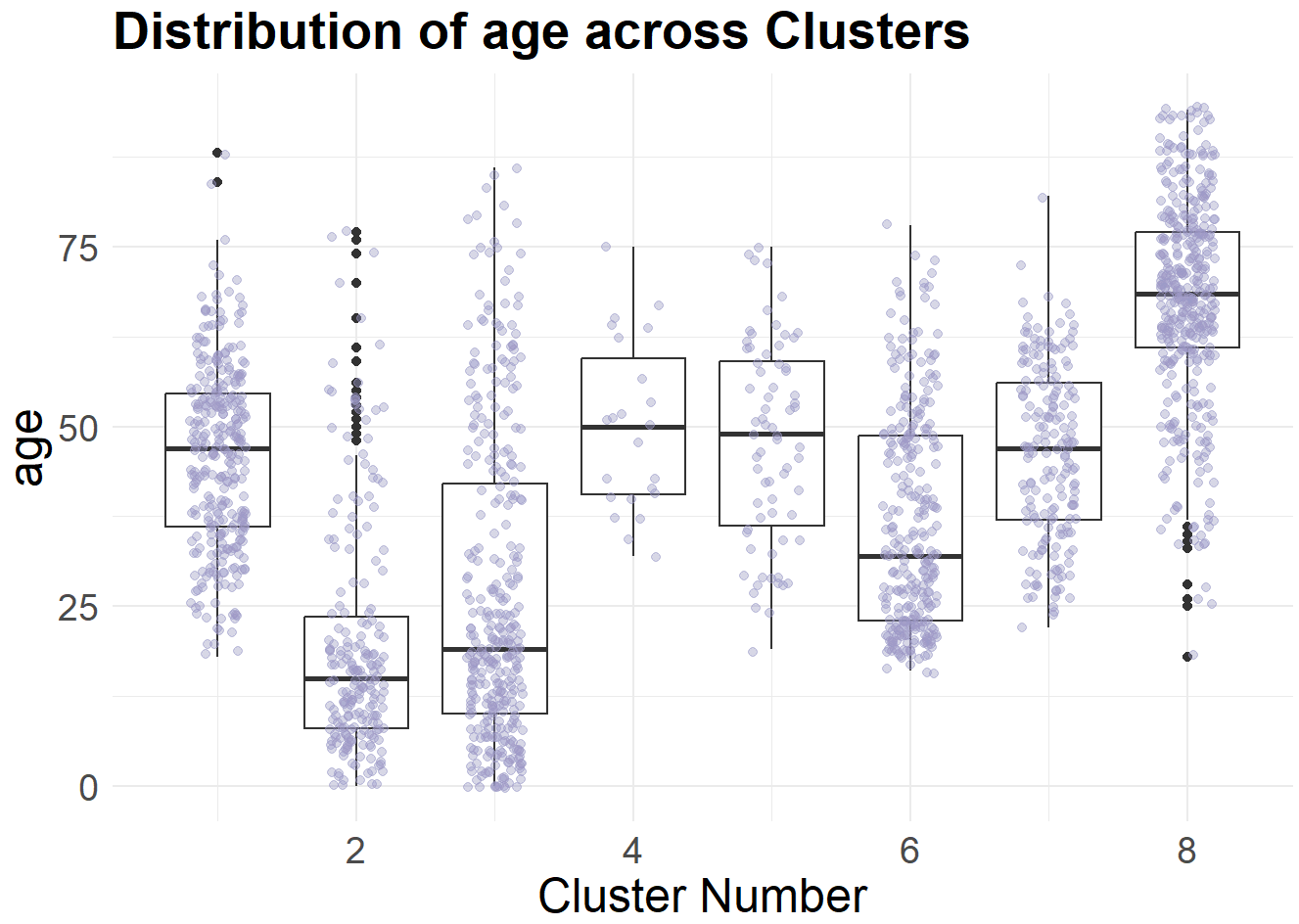
Let's visualize clusters for the numeric variables.

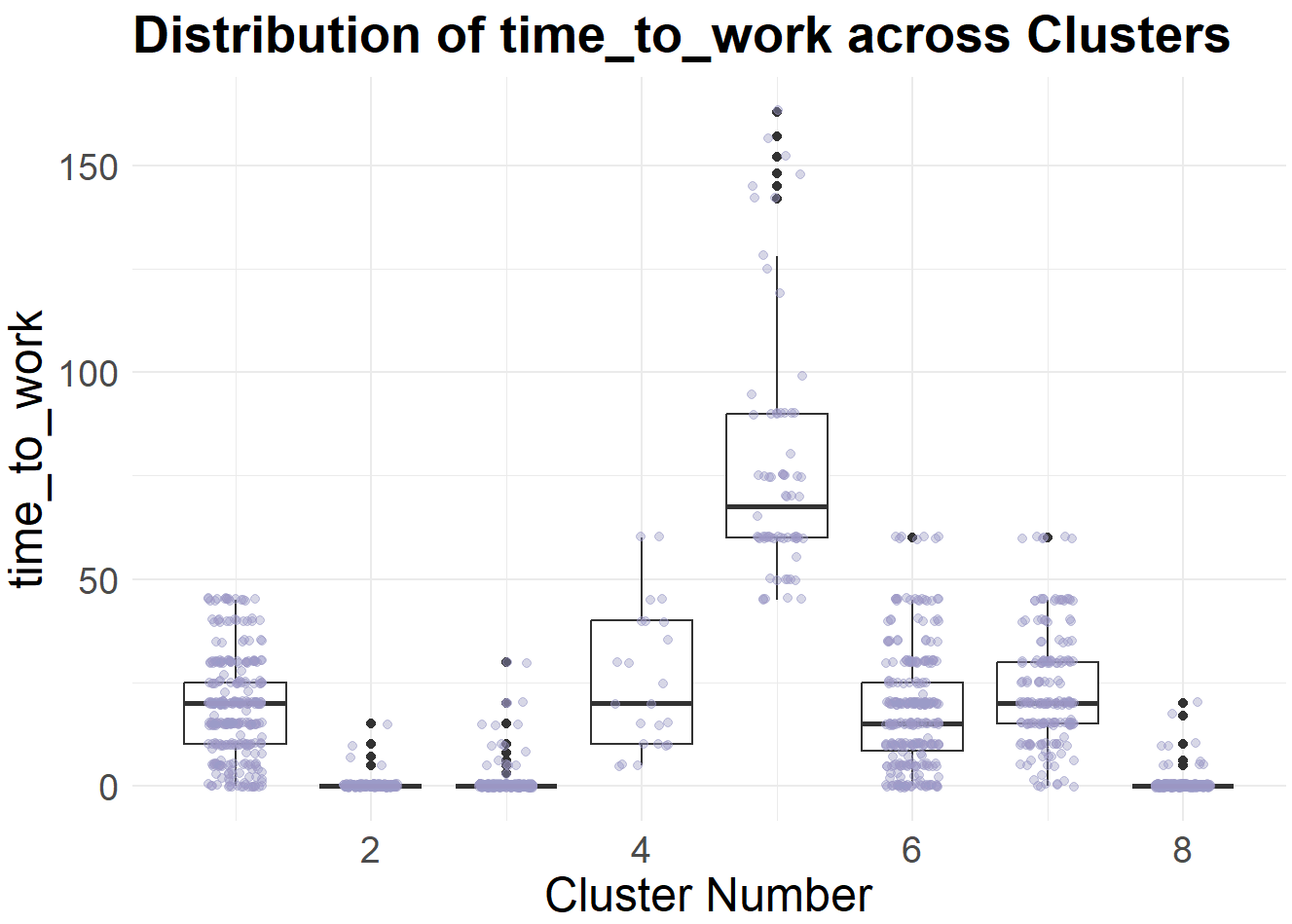
```
# Identify numeric variables excluding the cluster column
numeric_vars = names(data_clust)[sapply(data_clust, is.numeric) & names(data_clust) != 'cluste
r']

# Loop through each numeric variable to create a plot against cluster number
for (var in numeric_vars) {
  # Create the plot
  p = ggplot(data_clust, aes_string(x = 'cluster', y = var, group = 'cluster')) +
    geom_boxplot() +  # Boxplot to visualize distribution
    geom_jitter(width = 0.2, alpha = 0.4, color = "#9e9ac8") +  # Jitter to show individual data
points with dusty purple color
    labs(title = paste("Distribution of", var, "across Clusters"),
         x = "Cluster Number",
         y = var) +
    theme_minimal() +
    theme(text = element_text(size = 16),  # Increase general text size
          plot.title = element_text(size = 20, face = "bold"),  # Increase and bold plot title
          axis.title = element_text(size = 18),  # Increase axis title text size
          axis.text = element_text(size = 14))  # Increase axis text size

  # Print the plot
  print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Distribution of income across Clusters
income
4e+05
3e+05
2e+05
1e+05
0e+00
2
4
6
8
Cluster Number

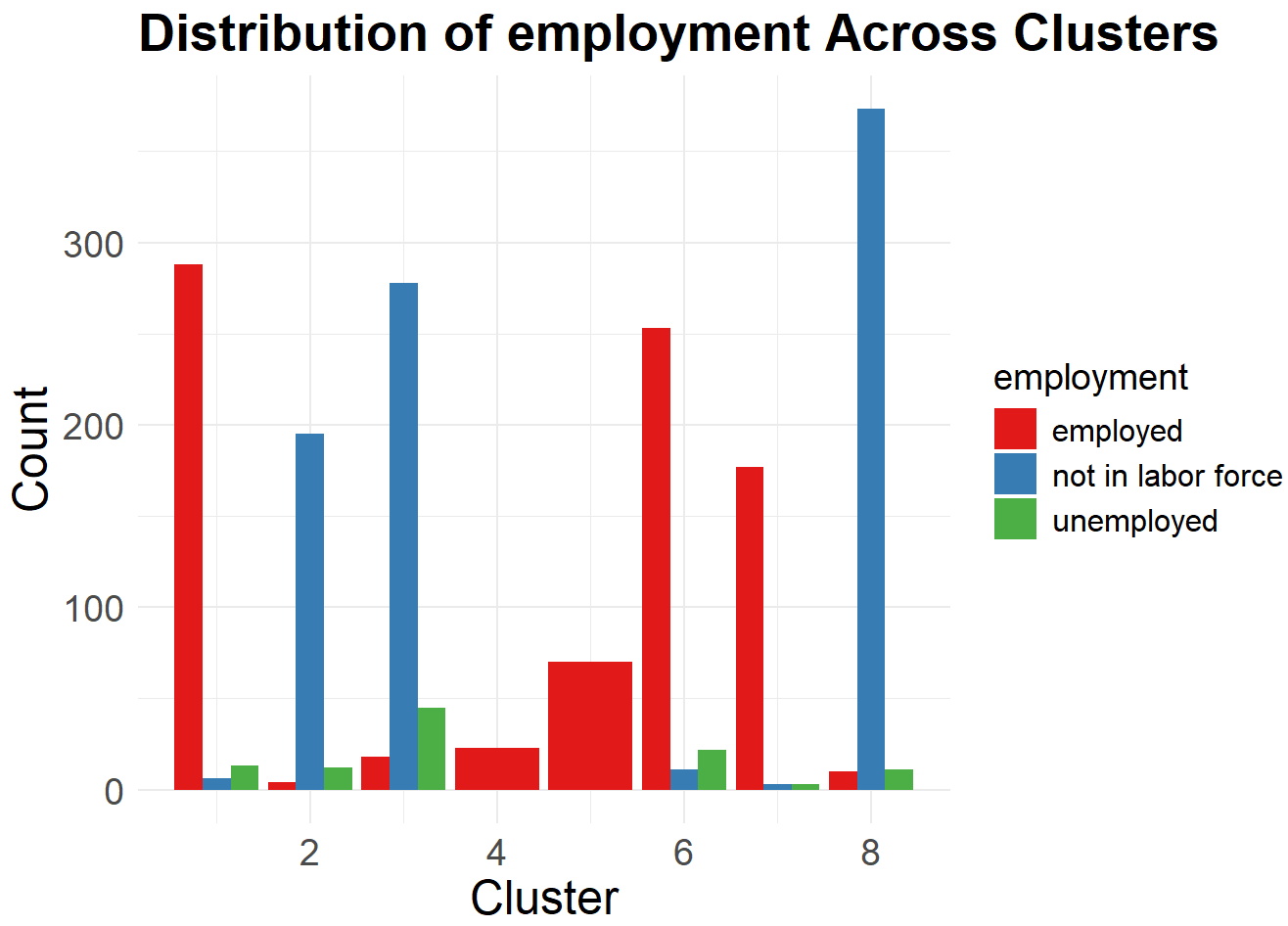

Distribution of hrs_work across Clusters
hrs_work
100
75
50
25
0
2
4
6
8
Cluster Number


Distribution of age across Clusters
age
75
50
25
0
2
4
6
8
Cluster Number

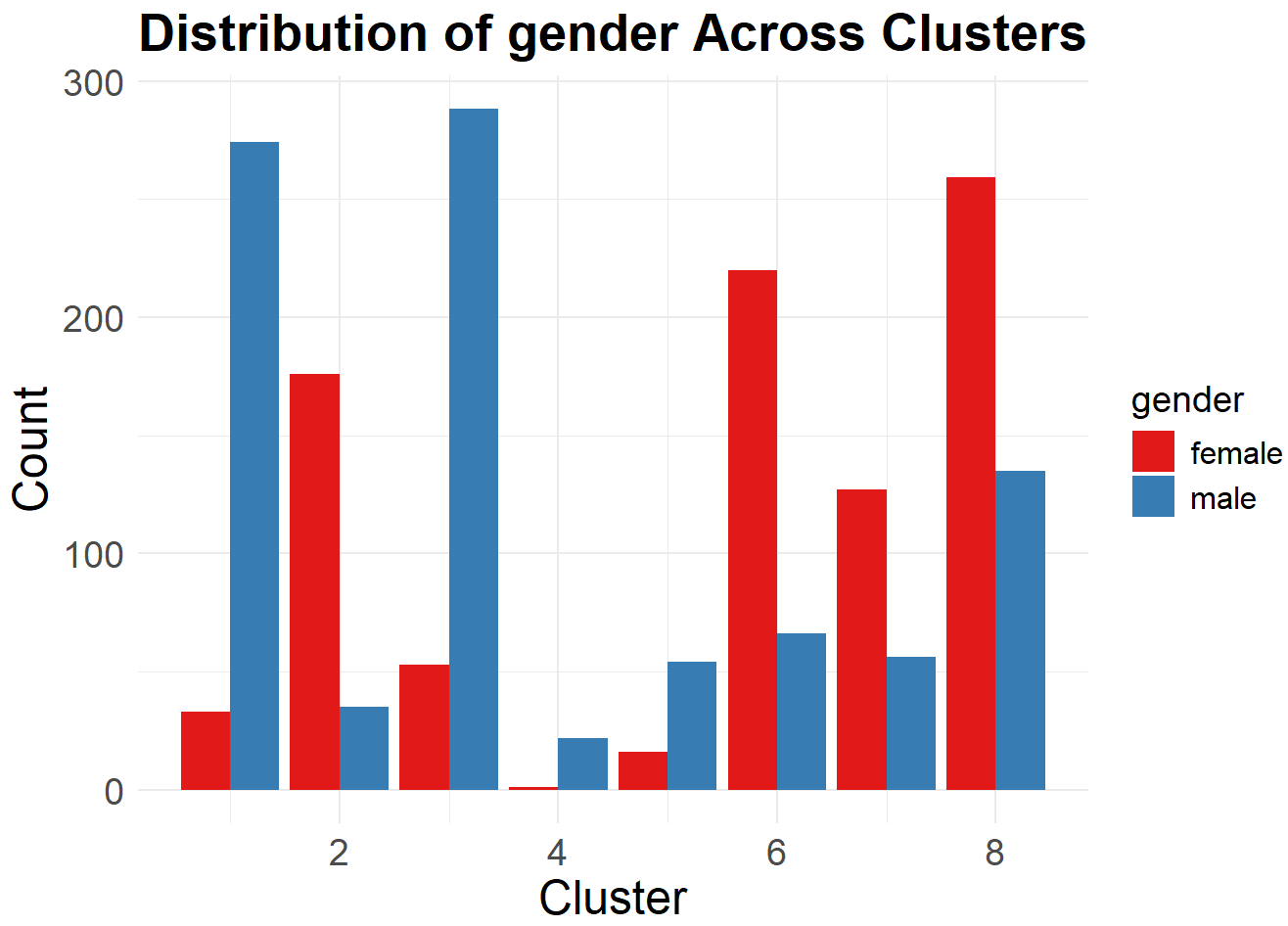

Distribution of time_to_work across Clusters
time_to_work
150
100
50
0
2
4
6
8
Cluster Number

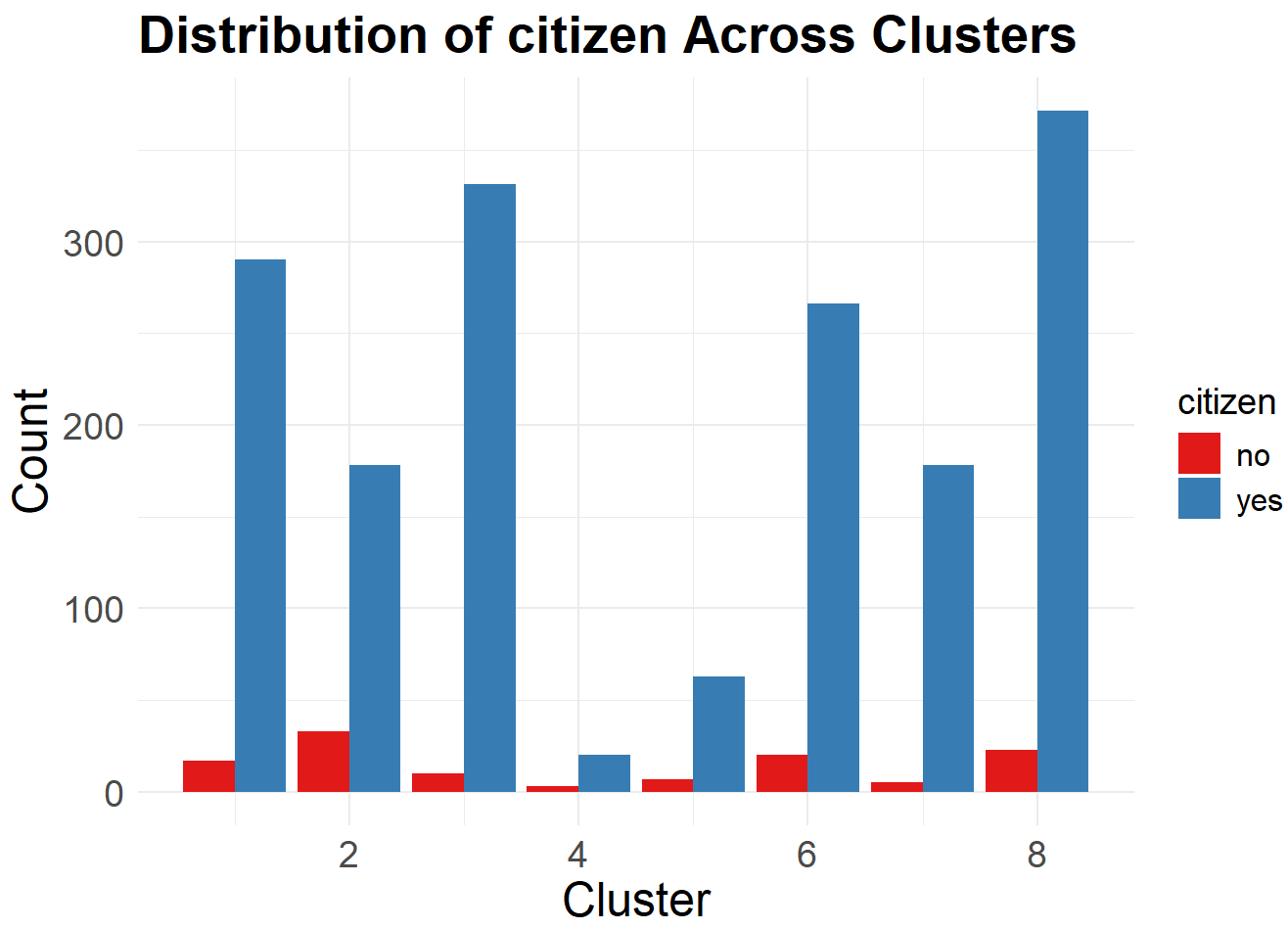
Let’s look at some bar graphs for the categorical variables.

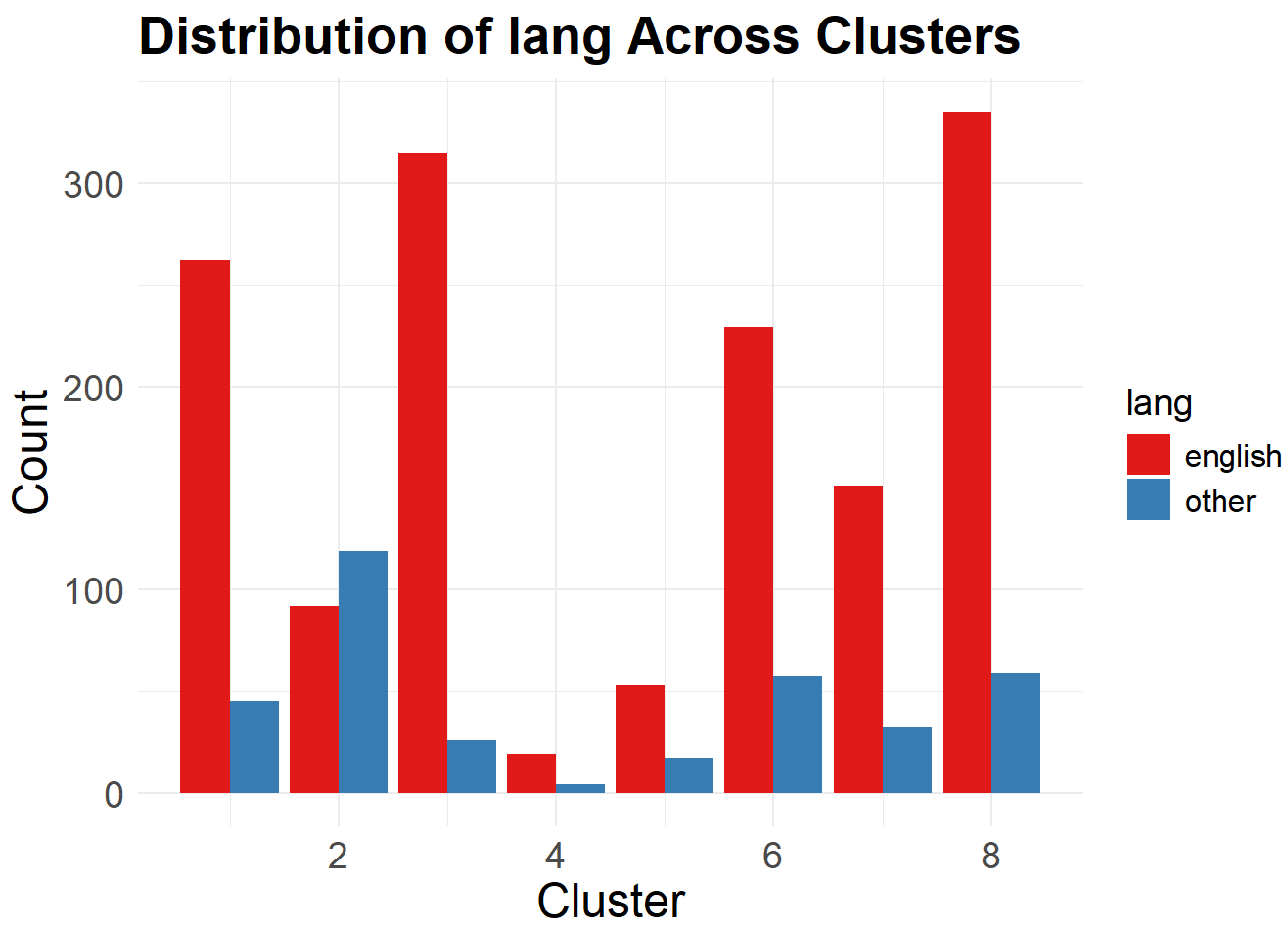
```
# Add the clusters to the categorical variables
categorical_variables$cluster = final_model$cluster

# Loop through each column, except 'cluster'
for(col_name in names(categorical_variables)[-which(names(categorical_variables) == "cluster")])
{

  # Generate the plot for the current column
  p = ggplot(categorical_variables, aes_string(x = "cluster", fill = col_name)) +
    geom_bar(position = "dodge") +
    labs(title = paste("Distribution of", col_name, "Across Clusters"), x = "Cluster", y = "Coun
t") +
    scale_fill_brewer(palette = "Set1") +  # Use ColorBrewer's Set3 color scheme
    theme_minimal() +
    theme(text = element_text(size = 16),  # Increase general text size
          plot.title = element_text(size = 20, face = "bold"),  # Increase and bold plot title
          axis.title = element_text(size = 18),  # Increase axis title text size
          axis.text = element_text(size = 14),  # Increase axis text size
          legend.title = element_text(size = 14),  # Increase legend title size
          legend.text = element_text(size = 12))  # Increase legend text size

  # Print the plot
  print(p)
}
```

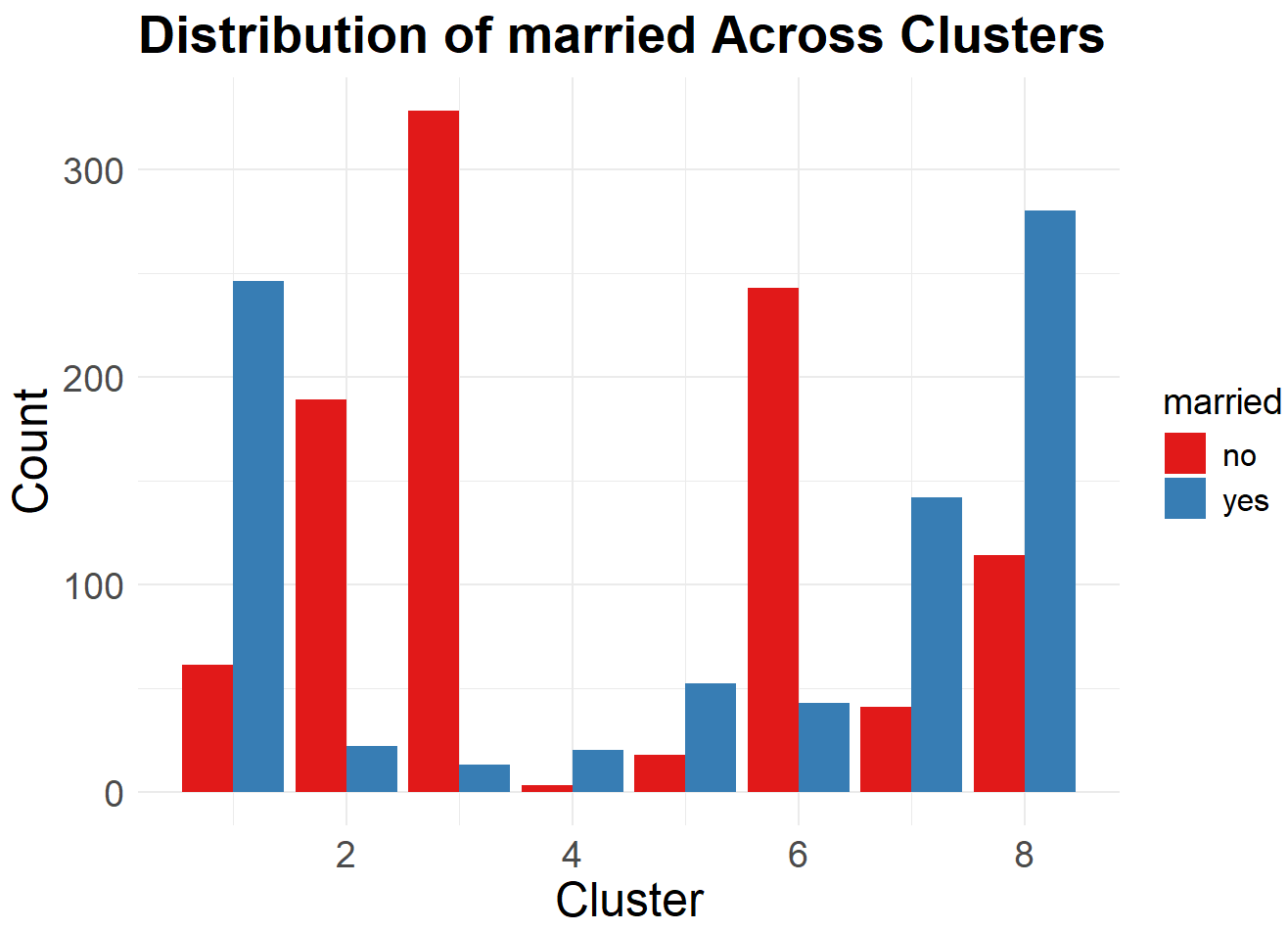

Distribution of employment Across Clusters
Count
300
200
100
0
2
4
6
8
Cluster
employment
employed
not in labor force
unemployed

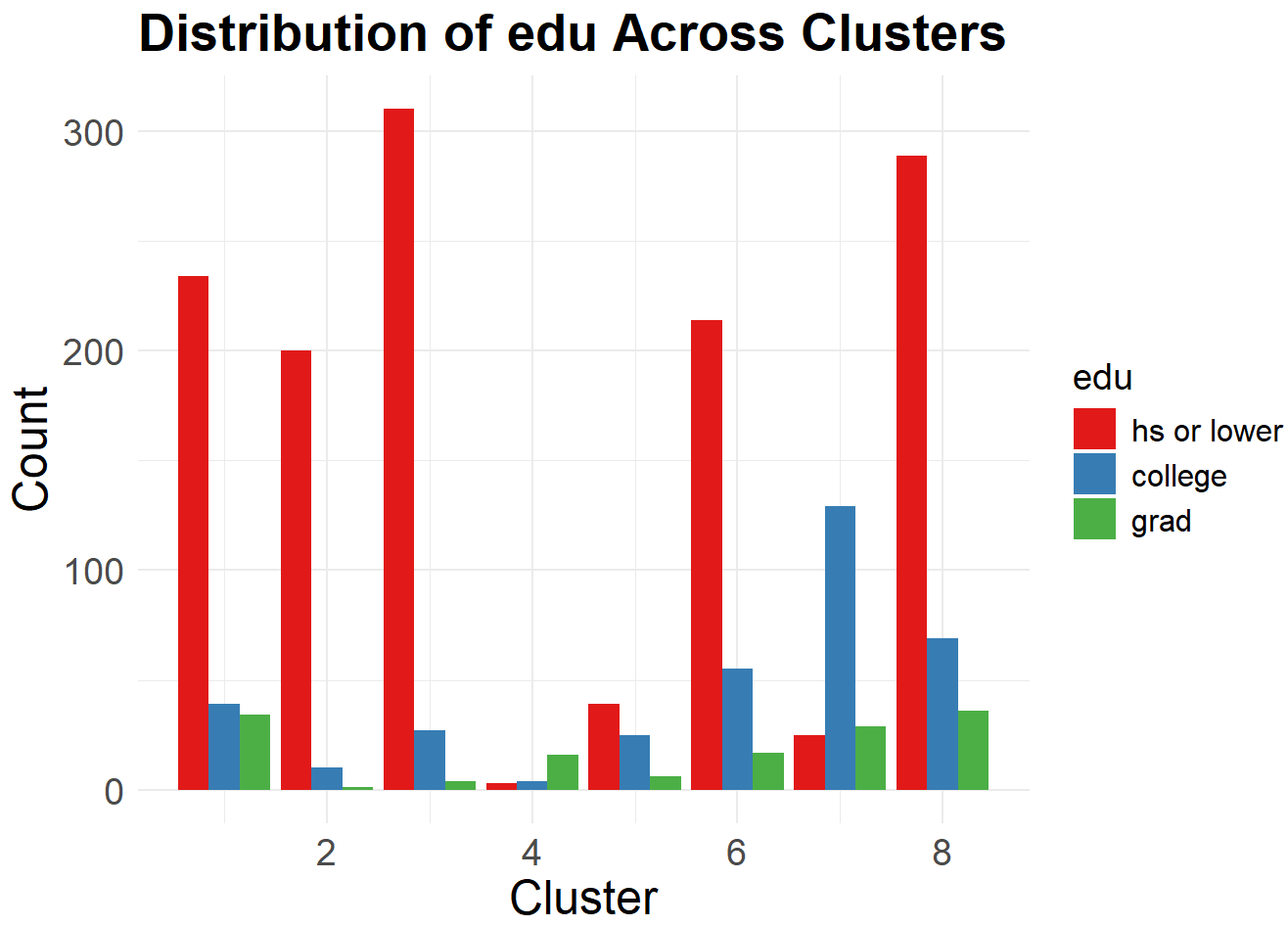

Distribution of race Across Clusters
Count
300
200
100
0
2
4
6
8
Cluster
race
asian
black
other
white

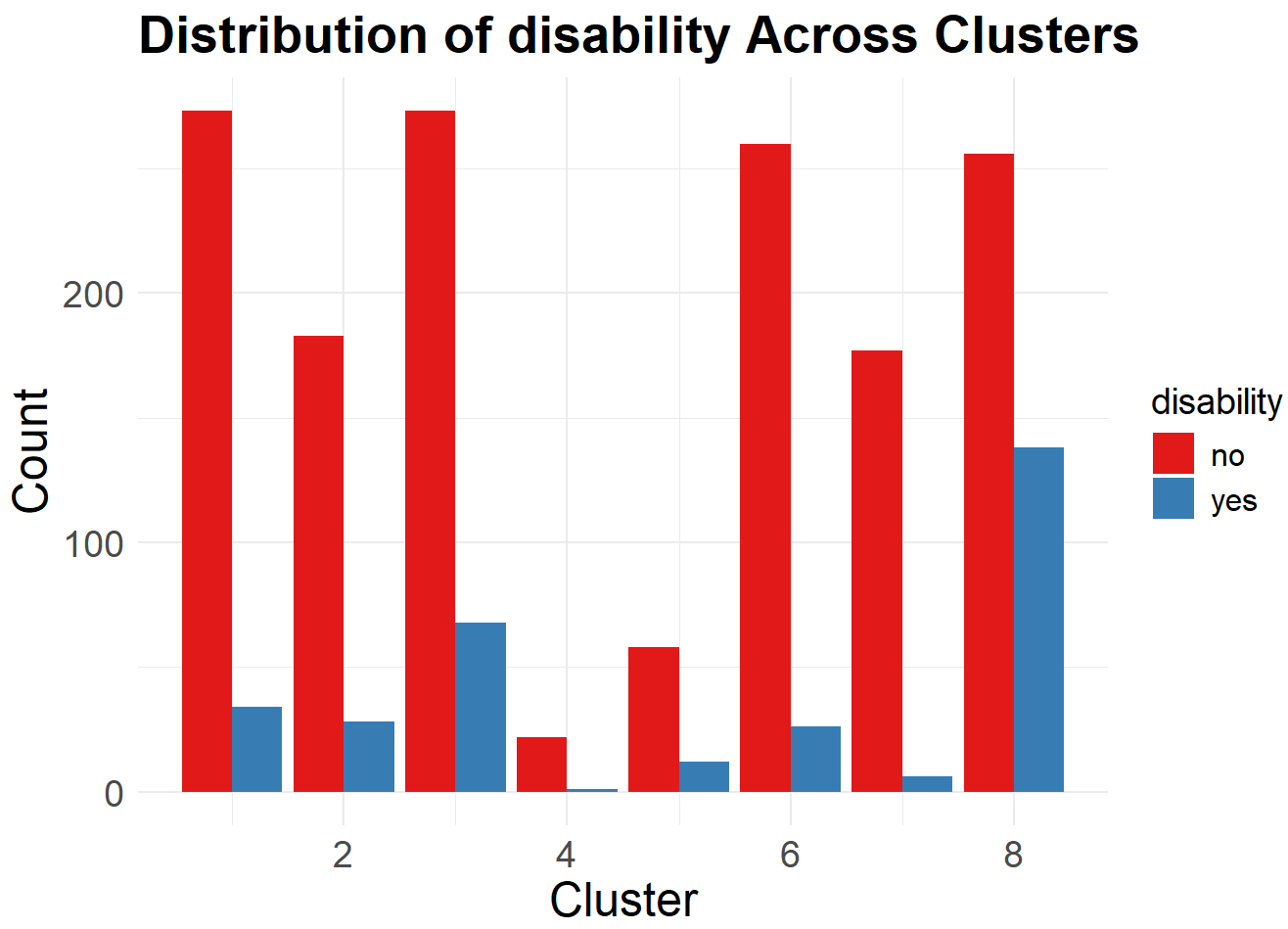

Distribution of gender Across Clusters
Count
300
200
100
0
2
4
6
8
Cluster
gender
female
male

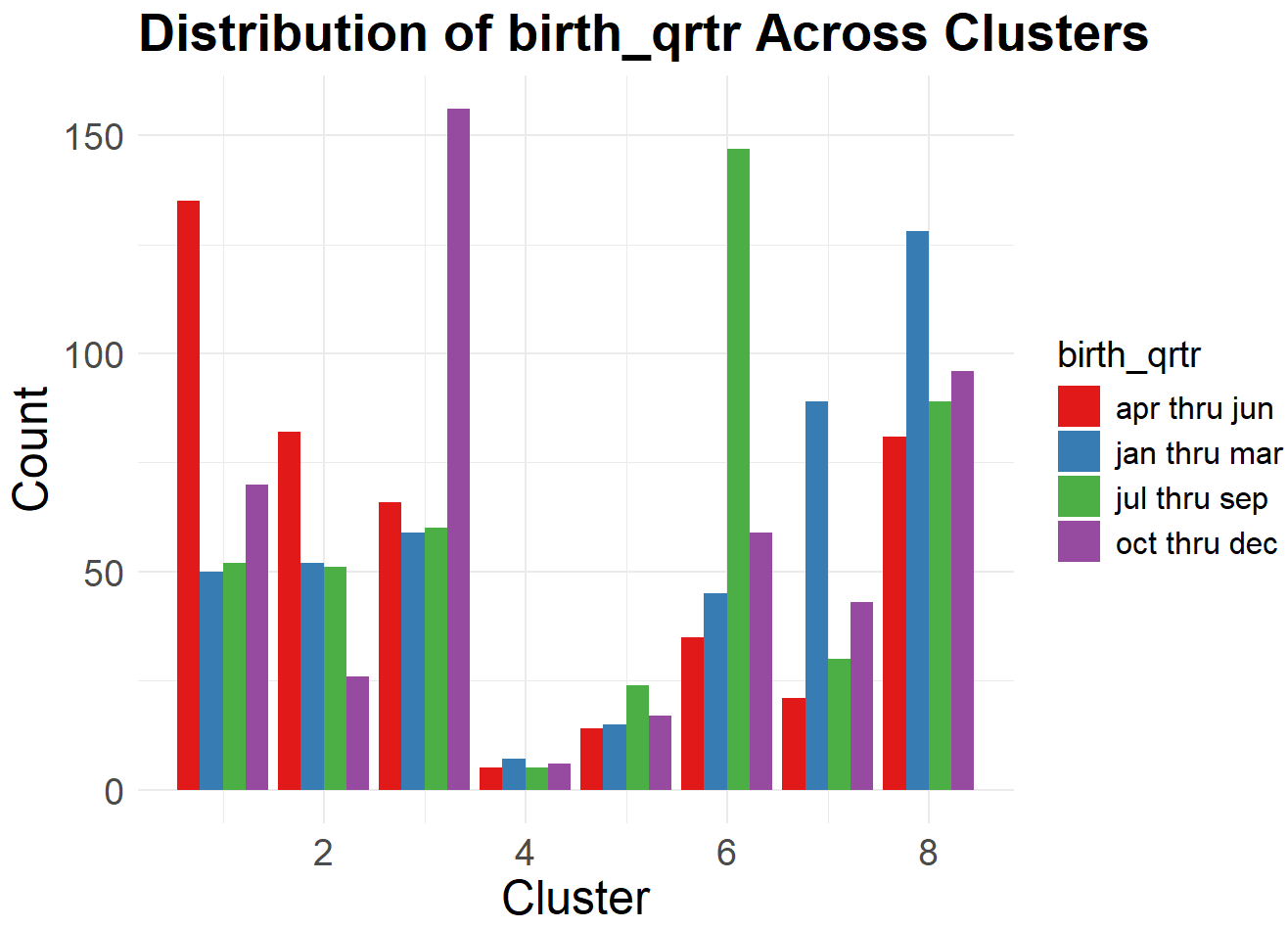

Distribution of citizen Across Clusters
Count
300
200
100
0
2
4
6
8
Cluster
citizen
no
yes


Distribution of lang Across Clusters
Count
300
200
100
0
2
4
6
8
Cluster
lang
english
other


Distribution of married Across Clusters
Count
300
200
100
0
2
4
6
8
Cluster
married
no
yes


Distribution of edu Across Clusters
Count
300
200
100
0
2
4
6
8
Cluster
edu
hs or lower
college
grad


Distribution of disability Across Clusters
Count
200
100
0
2
4
6
8
Cluster
disability
no
yes



t-SNE Dimensionality Reduction

Let's prepare data for t-SNE dimensionality reduction.

We need to remove any duplicates of numeric data from dataset or else an error will be returned.

```
# Create the numeric variable with the standardized data
data_numeric = data_standardized[, sapply(data_standardized, is.numeric)]

# Add the clusters
data_numeric$cluster = final_model$cluster

# Remove any duplicate entries so t-SNE will work properly
data_numeric_unique = unique(data_numeric)
```

Now we can run t-SNE on the numerical data.

```
# Run t-SNE on the numeric data
set.seed(1) # For reproducibility
tsne_results = Rtsne(data_numeric_unique, dims = 2, perplexity = 30, verbose = TRUE)
```

```
## Performing PCA
## Read the 1155 x 5 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.10 seconds (sparsity = 0.102395)!
## Learning embedding...
## Iteration 50: error is 63.796532 (50 iterations in 0.13 seconds)
## Iteration 100: error is 58.428615 (50 iterations in 0.12 seconds)
## Iteration 150: error is 57.730771 (50 iterations in 0.12 seconds)
## Iteration 200: error is 57.462641 (50 iterations in 0.12 seconds)
## Iteration 250: error is 57.321047 (50 iterations in 0.14 seconds)
## Iteration 300: error is 0.864571 (50 iterations in 0.11 seconds)
## Iteration 350: error is 0.663995 (50 iterations in 0.11 seconds)
## Iteration 400: error is 0.611477 (50 iterations in 0.11 seconds)
## Iteration 450: error is 0.590756 (50 iterations in 0.11 seconds)
## Iteration 500: error is 0.576957 (50 iterations in 0.11 seconds)
## Iteration 550: error is 0.567415 (50 iterations in 0.11 seconds)
## Iteration 600: error is 0.560938 (50 iterations in 0.11 seconds)
## Iteration 650: error is 0.555136 (50 iterations in 0.11 seconds)
## Iteration 700: error is 0.551774 (50 iterations in 0.11 seconds)
## Iteration 750: error is 0.549416 (50 iterations in 0.11 seconds)
## Iteration 800: error is 0.547228 (50 iterations in 0.11 seconds)
## Iteration 850: error is 0.545557 (50 iterations in 0.11 seconds)
## Iteration 900: error is 0.544048 (50 iterations in 0.11 seconds)
## Iteration 950: error is 0.542494 (50 iterations in 0.11 seconds)
## Iteration 1000: error is 0.541124 (50 iterations in 0.11 seconds)
## Fitting performed in 2.28 seconds.
```

```
# Combine the t-SNE dimensions with the cluster assignments
tsne_data = data.frame(X = tsne_results$Y[,1], Y = tsne_results$Y[,2], Cluster = data_numeric_un
ique$cluster)
```

Let’s visualize the clusters on a t-SNE plot with applied dimensionality reduction.

```
# Convert cluster to a factor so that the color scale works properly
tsne_data$Cluster = factor(tsne_data$Cluster)

# Now, use ggplot with scale_color_manual
ggplot(tsne_data, aes(x = X, y = Y, color = Cluster)) +
  geom_point(alpha = 0.7) +
  scale_color_manual(values = rainbow(length(levels(tsne_data$Cluster)))) +
  labs(title = "t-SNE Visualization with Adjusted Clusters",
       x = "t-SNE Dimension 1", y = "t-SNE Dimension 2", color = "Cluster") +
  theme_minimal() +
  theme(text = element_text(size = 16),  # General text size increase
        plot.title = element_text(size = 20, face = "bold"),  # Increase plot title size and mak
e it bold
        axis.title = element_text(size = 18),  # Increase axis title text size
        axis.text = element_text(size = 14),  # Increase axis text size
        legend.title = element_text(size = 14),  # Increase legend title size
        legend.text = element_text(size = 12))  # Increase legend text size
```