The University of Texas at Dallas

Naveen Jindal School of Management

Fall 2024

H1-B Visa Program Analysis

Business Analytics with R – BUAN 6356.006

Under the Guidance of: Prof. Zhe Zhang

Group – 5

Group Members: **Yash Thakkar, Abin Roy, Priya Medankar, Vidhi Agarwal**

Project Motivation/Background

The H1-B visa program is critical for enabling skilled foreign workers to contribute to the U.S. economy. However, it is often scrutinized for its perceived lack of transparency and fairness. This study aims to address key concerns:

- Identifying trends in visa applications and approvals.
- Investigating potential biases in approval processes.
- Enhancing the efficiency of the program through data-driven insights.

As F1 international students, the H1-B visa is a significant milestone in our professional journey. Recognizing its importance and the challenges it entails, we decided to explore this topic for our project. Our goal was to gain a deeper understanding of the H1-B visa application process and leverage our findings to provide meaningful insights. Given the direct relevance of the subject to our experiences and aspirations, we felt this project was both timely and impactful.

By combining analytics techniques with class teachings, we aim to deliver actionable recommendations that not only highlight systemic issues but also empower applicants, employers, and policymakers to enhance the program's fairness and efficiency.

Data Description

The dataset comprises H1-B visa applications submitted by U.S. employers. It includes approximately 33,000 to 56,000 records with 30 attributes, such as:

- LCA_CASE_NUMBER: Unique identifier for applications.
- STATUS: Certification status (certified/rejected).
- LCA_CASE_JOB_TITLE: Applicant's job title.
- LCA_CASE_WAGE_RATE_FROM/TO: Wage range for the position.
- FULL_TIME_POS: Full-time position indicator (Y/N).
- TOTAL_WORKERS: Number of workers requested.

Post data preprocessing, the class imbalance (90% certified, 10% rejected) was mitigated to a 60:40 ratio for better model training and evaluation.

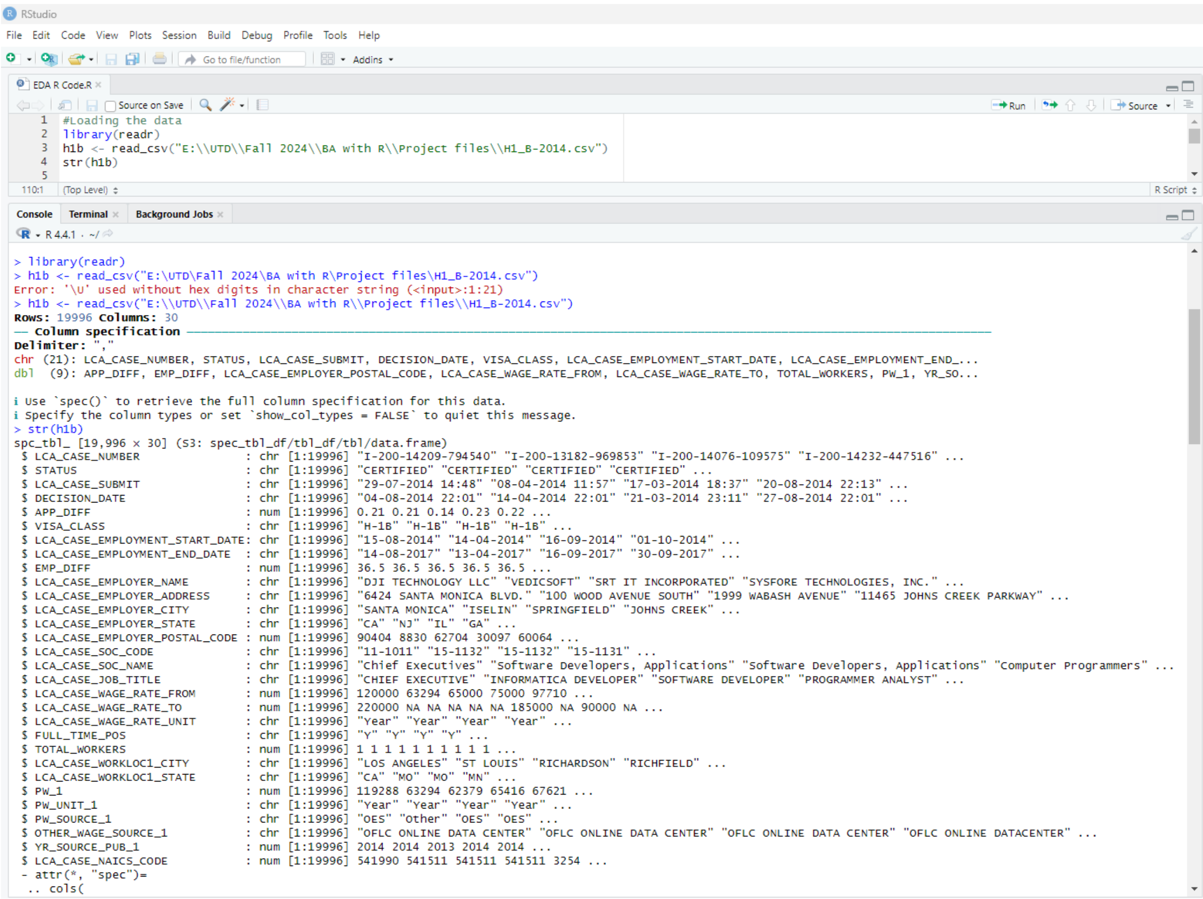
Business Intelligence (BI) Model and Process

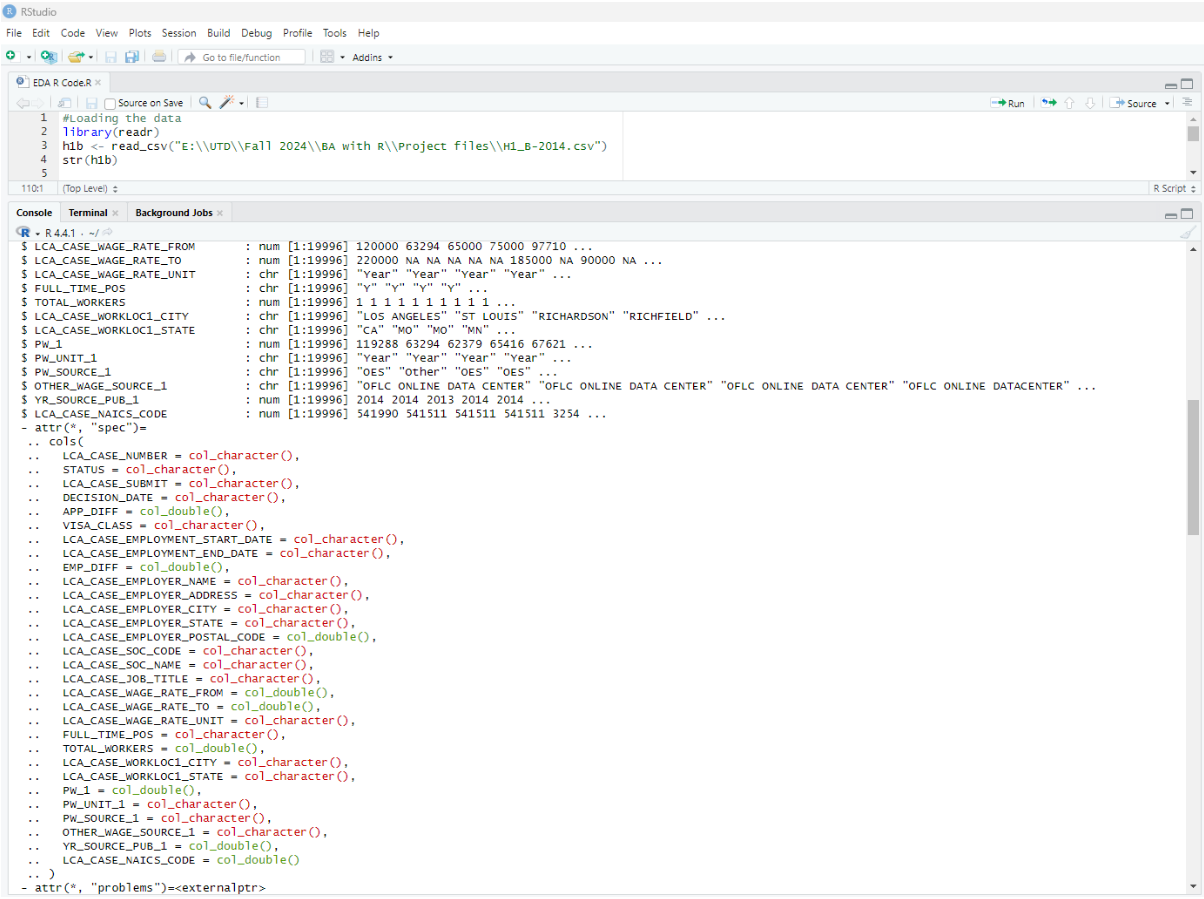
Steps of Analysis

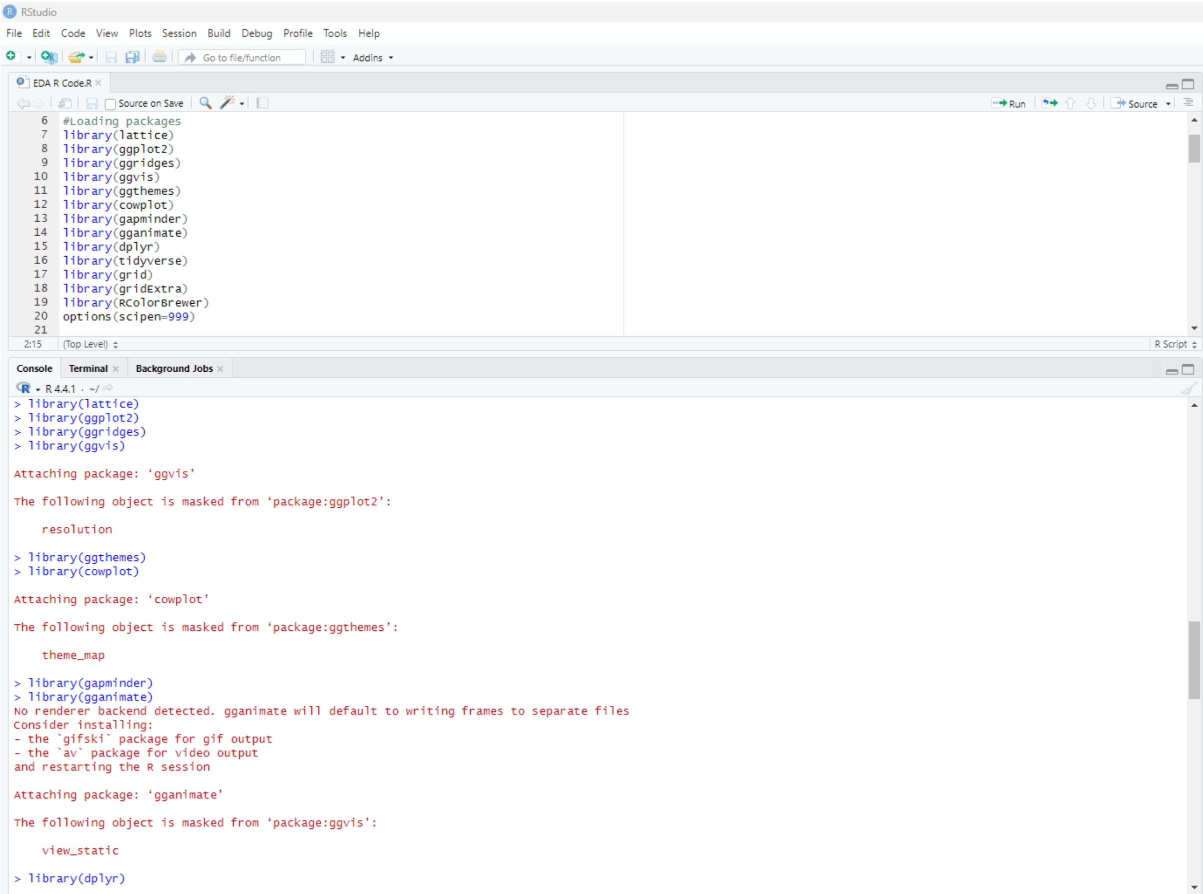
1. Data Preprocessing:
   - Handling missing values.
   - Normalizing numeric attributes.
   - Converting categorical data into numerical format.
2. Exploratory Data Analysis (EDA):
   - Bar charts, scatterplots, histograms.
   - Analysed approval rates and wage distributions.

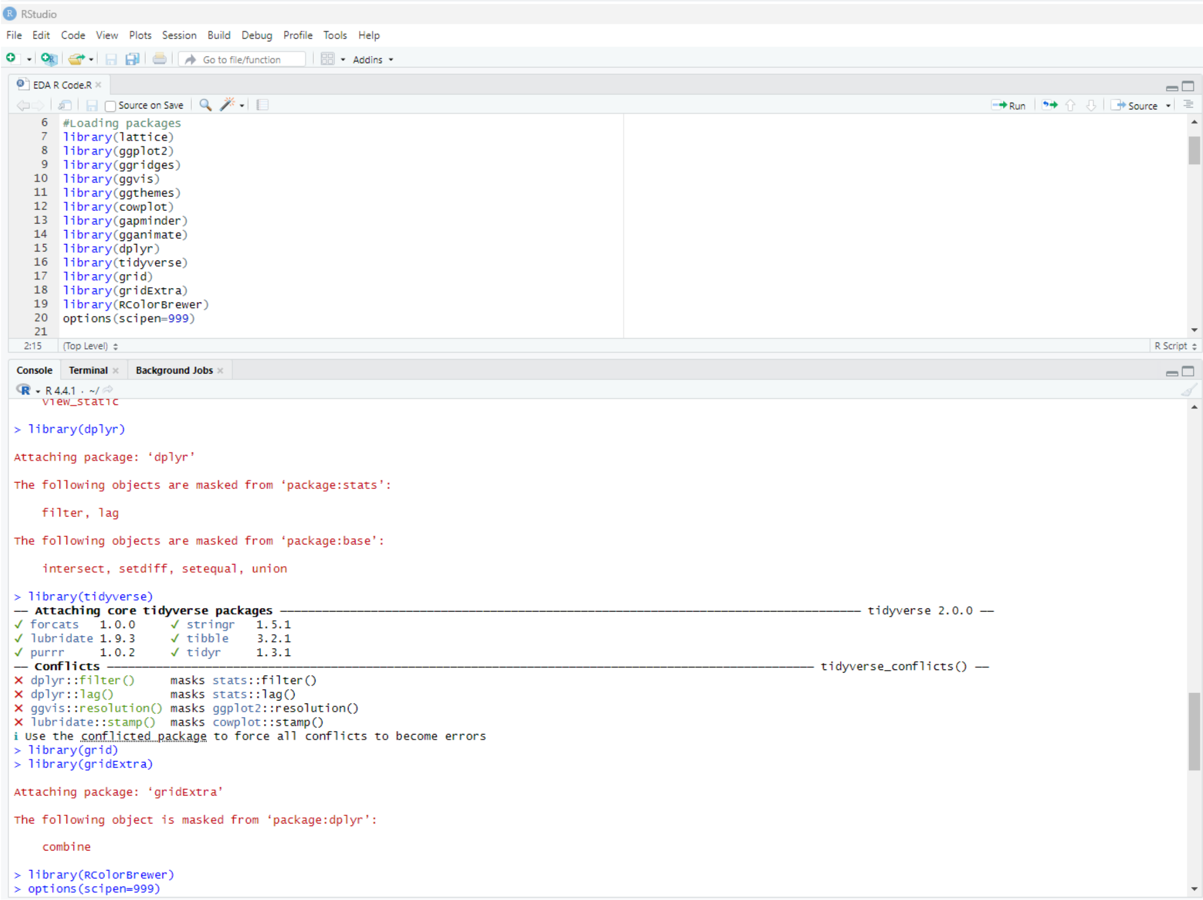
Exploratory Data Analysis (EDA)

- Loading the data
- Loading packages

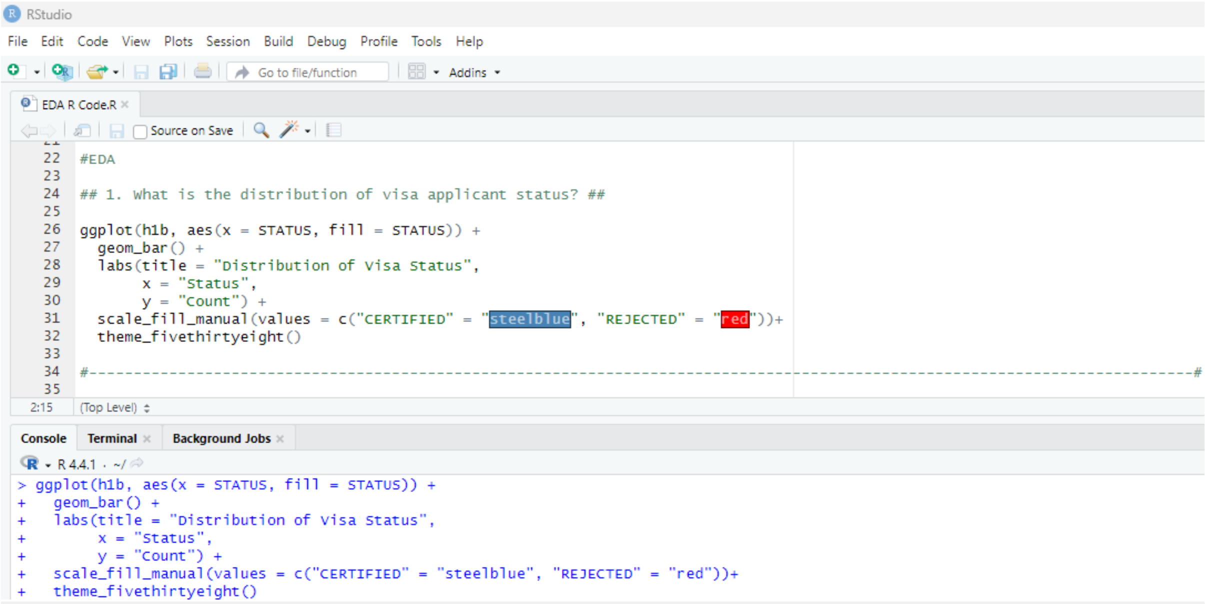


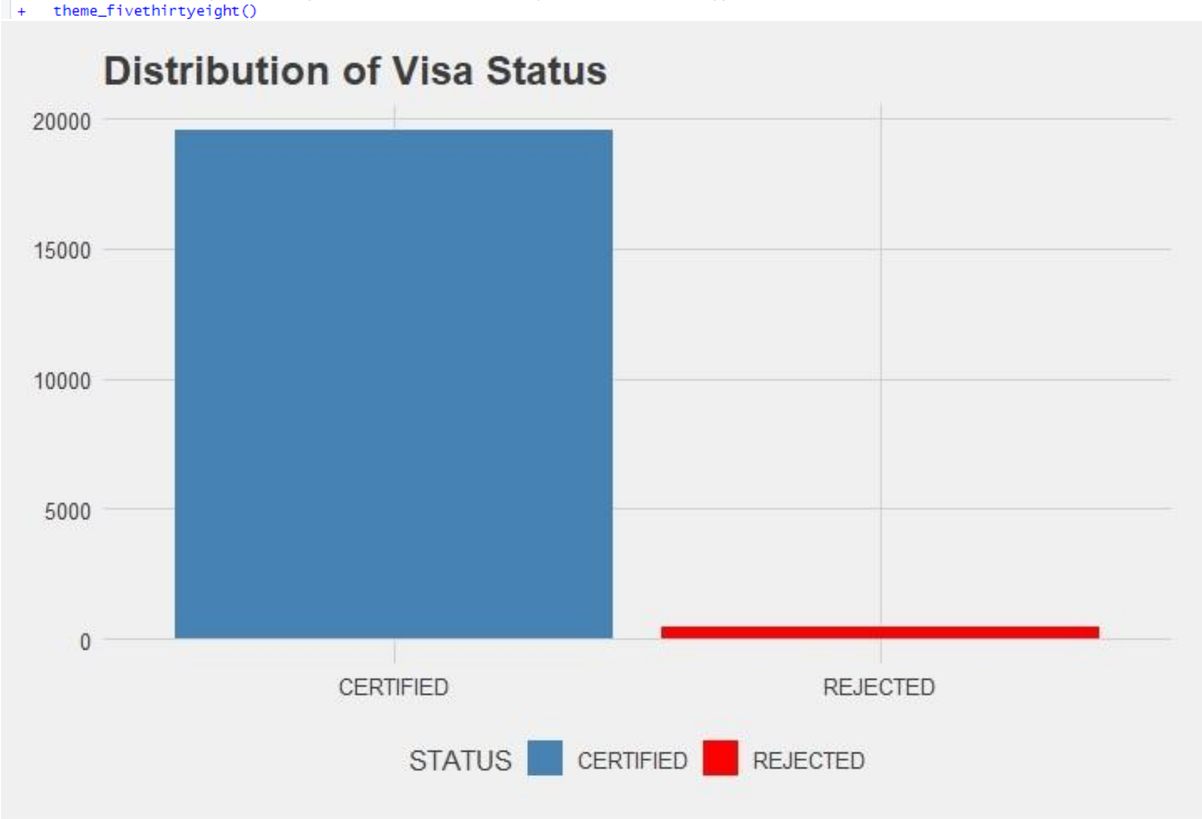

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
EDA R Code.R
Source on Save
Run
Source
#Loading the data
library(readr)
h1b <- read_csv("E:\\UTD\\Fall 2024\\BA with R\\Project files\\H1_B-2014.csv")
str(h1b)
110:1
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.4.1 · ~/
$ LCA_CASE_WAGE_RATE_FROM : num [1:19996] 120000 63294 65000 75000 97710 ...
$ LCA_CASE_WAGE_RATE_TO : num [1:19996] 220000 NA NA NA NA NA 185000 NA 90000 NA ...
$ LCA_CASE_WAGE_RATE_UNIT : chr [1:19996] "Year" "Year" "Year" "Year" ...
$ FULL_TIME_POS : chr [1:19996] "Y" "Y" "Y" "Y" ...
$ TOTAL_WORKERS : num [1:19996] 1 1 1 1 1 1 1 1 1 1 ...
$ LCA_CASE_WORKLOC1_CITY : chr [1:19996] "LOS ANGELES" "ST LOUIS" "RICHARDSON" "RICHFIELD" ...
$ LCA_CASE_WORKLOC1_STATE : chr [1:19996] "CA" "MO" "MO" "MN" ...
$ PW_1 : num [1:19996] 119288 63294 62379 65416 67621 ...
$ PW_UNIT_1 : chr [1:19996] "Year" "Year" "Year" "Year" ...
$ PW_SOURCE_1 : chr [1:19996] "OES" "Other" "OES" "OES" ...
$ OTHER_WAGE_SOURCE_1 : chr [1:19996] "OFLC ONLINE DATA CENTER" "OFLC ONLINE DATA CENTER" "OFLC ONLINE DATA CENTER" "OFLC ONLINE DATACENTER" ...
$ YR_SOURCE_PUB_1 : num [1:19996] 2014 2014 2013 2014 2014 ...
$ LCA_CASE_NAICS_CODE : num [1:19996] 541990 541511 541511 541511 3254 ...
- attr(*, "spec")=
.. cols(
.. LCA_CASE_NUMBER = col_character(),
.. STATUS = col_character(),
.. LCA_CASE_SUBMIT = col_character(),
.. DECISION_DATE = col_character(),
.. APP_DIFF = col_double(),
.. VISA_CLASS = col_character(),
.. LCA_CASE_EMPLOYMENT_START_DATE = col_character(),
.. LCA_CASE_EMPLOYMENT_END_DATE = col_character(),
.. EMP_DIFF = col_double(),
.. LCA_CASE_EMPLOYER_NAME = col_character(),
.. LCA_CASE_EMPLOYER_ADDRESS = col_character(),
.. LCA_CASE_EMPLOYER_CITY = col_character(),
.. LCA_CASE_EMPLOYER_STATE = col_character(),
.. LCA_CASE_EMPLOYER_POSTAL_CODE = col_double(),
.. LCA_CASE_SOC_CODE = col_character(),
.. LCA_CASE_SOC_NAME = col_character(),
.. LCA_CASE_JOB_TITLE = col_character(),
.. LCA_CASE_WAGE_RATE_FROM = col_double(),
.. LCA_CASE_WAGE_RATE_TO = col_double(),
.. LCA_CASE_WAGE_RATE_UNIT = col_character(),
.. FULL_TIME_POS = col_character(),
.. TOTAL_WORKERS = col_double(),
.. LCA_CASE_WORKLOC1_CITY = col_character(),
.. LCA_CASE_WORKLOC1_STATE = col_character(),
.. PW_1 = col_double(),
.. PW_UNIT_1 = col_character(),
.. PW_SOURCE_1 = col_character(),
.. OTHER_WAGE_SOURCE_1 = col_character(),
.. YR_SOURCE_PUB_1 = col_double(),
.. LCA_CASE_NAICS_CODE = col_double()
..)
- attr(*, "problems")=<externalptr>


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
EDA R Code.R
Source on Save
Run
Source
6 #Loading packages
7 library(lattice)
8 library(ggplot2)
9 library(ggridges)
10 library(ggvis)
11 library(ggthemes)
12 library(cowplot)
13 library(gapminder)
14 library(gganimate)
15 library(dplyr)
16 library(tidyverse)
17 library(grid)
18 library(gridExtra)
19 library(RColorBrewer)
20 options(scipen=999)
21
2:15 (Top Level)
R Script
Console Terminal Background Jobs
R 4.4.1 · ~/
> library(lattice)
> library(ggplot2)
> library(ggridges)
> library(ggvis)
Attaching package: 'ggvis'
The following object is masked from 'package:ggplot2':
resolution
> library(ggthemes)
> library(cowplot)
Attaching package: 'cowplot'
The following object is masked from 'package:ggthemes':
theme_map
> library(gapminder)
> library(gganimate)
No renderer backend detected. gganimate will default to writing frames to separate files
Consider installing:
- the `gifski` package for gif output
- the `av` package for video output
and restarting the R session
Attaching package: 'gganimate'
The following object is masked from 'package:ggvis':
view_static
> library(dplyr)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
EDA R Code.R
Source on Save
Run
Source
6 #Loading packages
7 library(lattice)
8 library(ggplot2)
9 library(ggridges)
10 library(ggvis)
11 library(ggthemes)
12 library(cowplot)
13 library(gapminder)
14 library(gganimate)
15 library(dplyr)
16 library(tidyverse)
17 library(grid)
18 library(gridExtra)
19 library(RColorBrewer)
20 options(scipen=999)
21
2:15 (Top Level)
R Script
Console Terminal Background Jobs
R 4.4.1 · ~/
> 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(tidyverse)
— Attaching core tidyverse packages ———— tidyverse 2.0.0 —
✓ forcats 1.0.0 ✓ stringr 1.5.1
✓ lubridate 1.9.3 ✓ tibble 3.2.1
✓ purrr 1.0.2 ✓ tidyr 1.3.1
— Conflicts ———— tidyverse_conflicts() —
✕ dplyr::filter() masks stats::filter()
✕ dplyr::lag() masks stats::lag()
✕ ggvis::resolution() masks ggplot2::resolution()
✕ lubridate::stamp() masks cowplot::stamp()
ℹ Use the conflicted package to force all conflicts to become errors
> library(grid)
> library(gridExtra)
Attaching package: 'gridExtra'
The following object is masked from 'package:dplyr':
combine
> library(RColorBrewer)
> options(scipen=999)

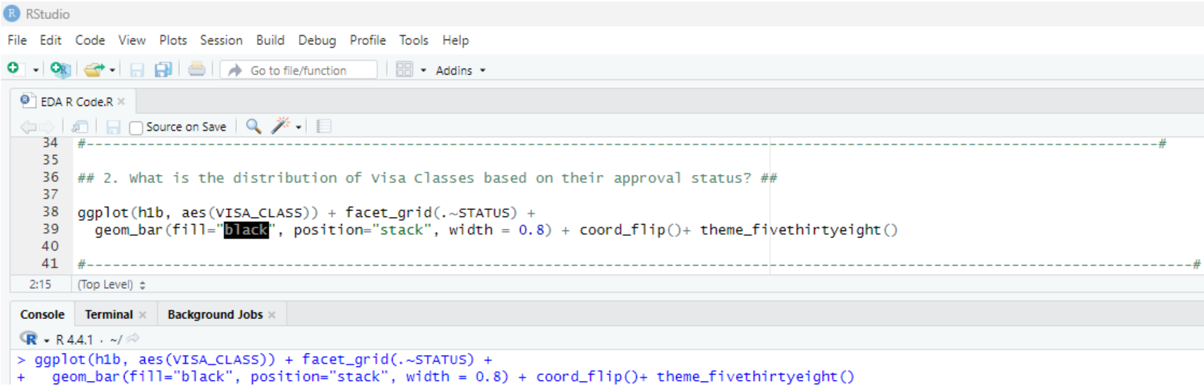
1) What is the distribution of visa applicant status?

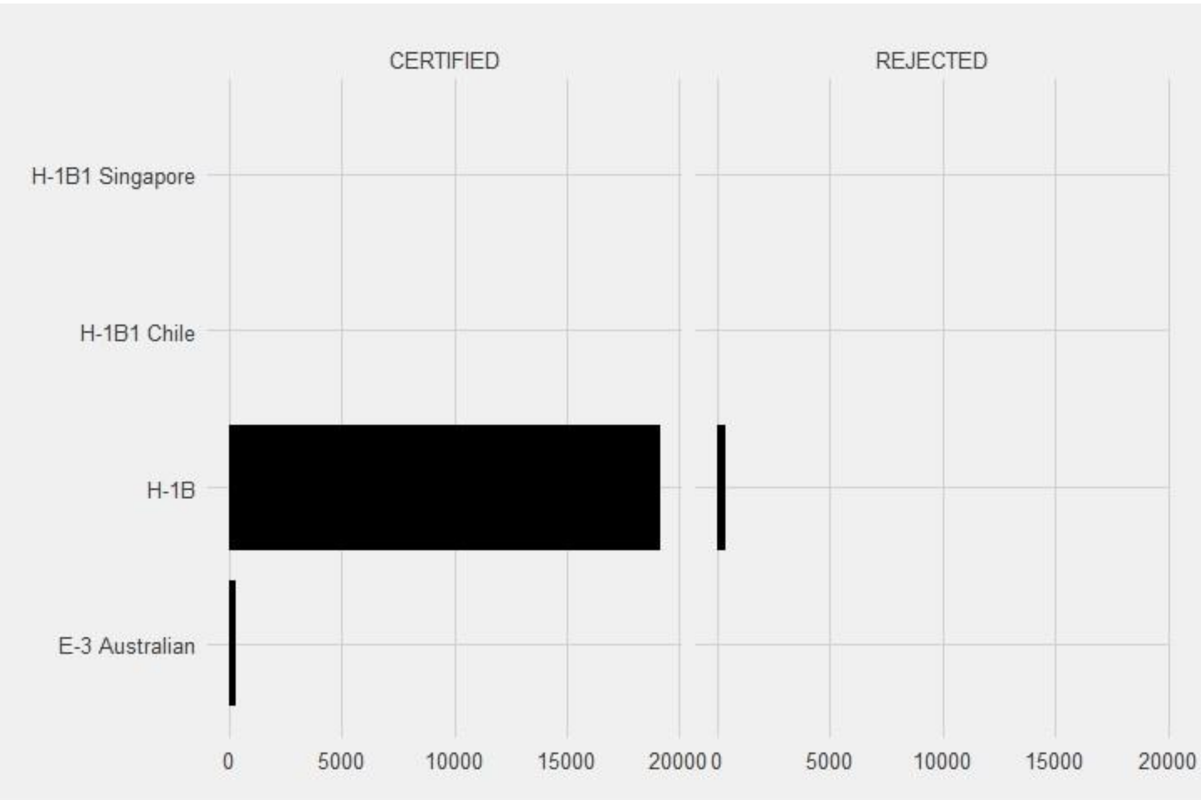




Inference: It can be observed that majority of the applicants have been approved of their visa application status.

2) What is the distribution of Visa Classes based on their approval status?





Inference:

- The analysis suggests that most applicants opted for H1B, with E-3 Australian being the next popular choice.
- Notably, H1B stands out as having the highest number of certifications, indicating a predominantly positive status.

3) Try to understand the Wage rate for case applicants and dig further into the segregation based on Visa applied for.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

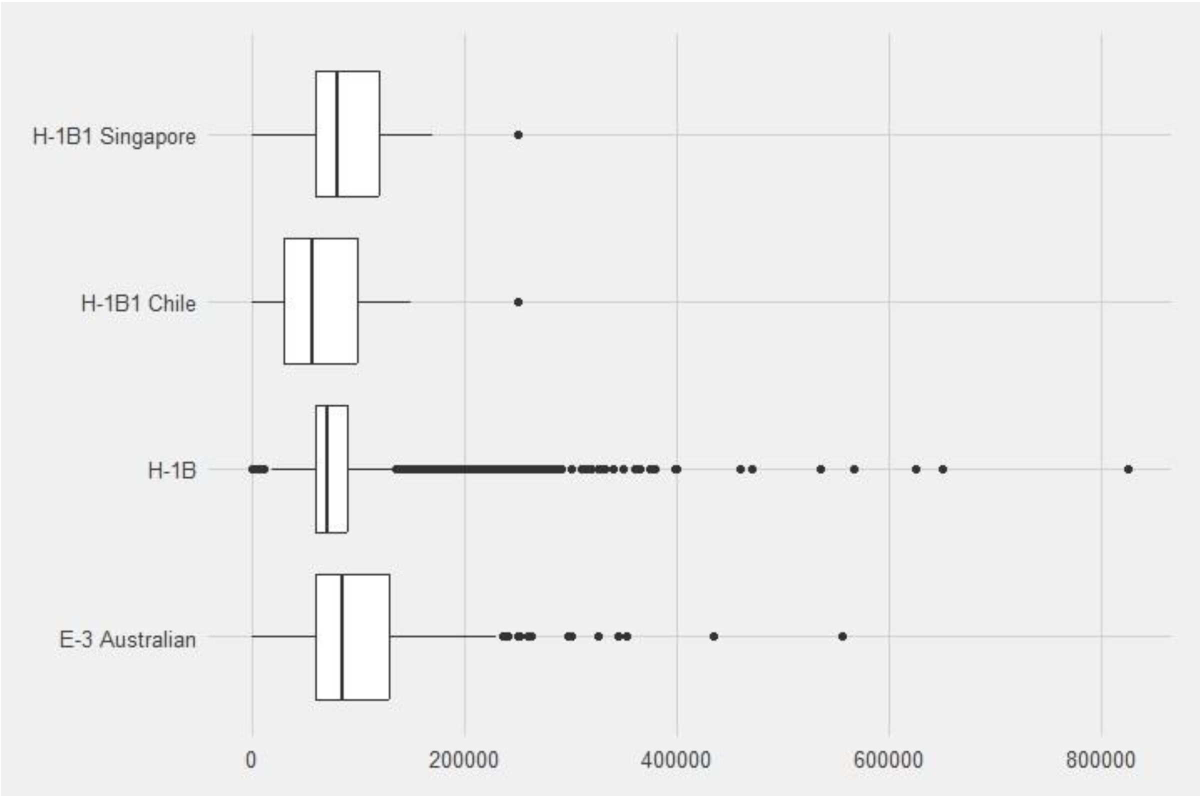
EDA R Code.R

```r
## 3. Try to understand the wage rate for case applicants, and dig further into the segregation based on Visa applied for ##

summary(h1b$LCA_CASE_WAGE_RATE_FROM)

ggplot(h1b, aes(x = VISA_CLASS, y = LCA_CASE_WAGE_RATE_FROM)) +
  geom_boxplot() + coord_flip()+
  theme_fivethirtyeight()

#-----------------------------------------------------------------------------------------------------------#
```

Console Terminal Background Jobs

R 4.4.1 · ~/

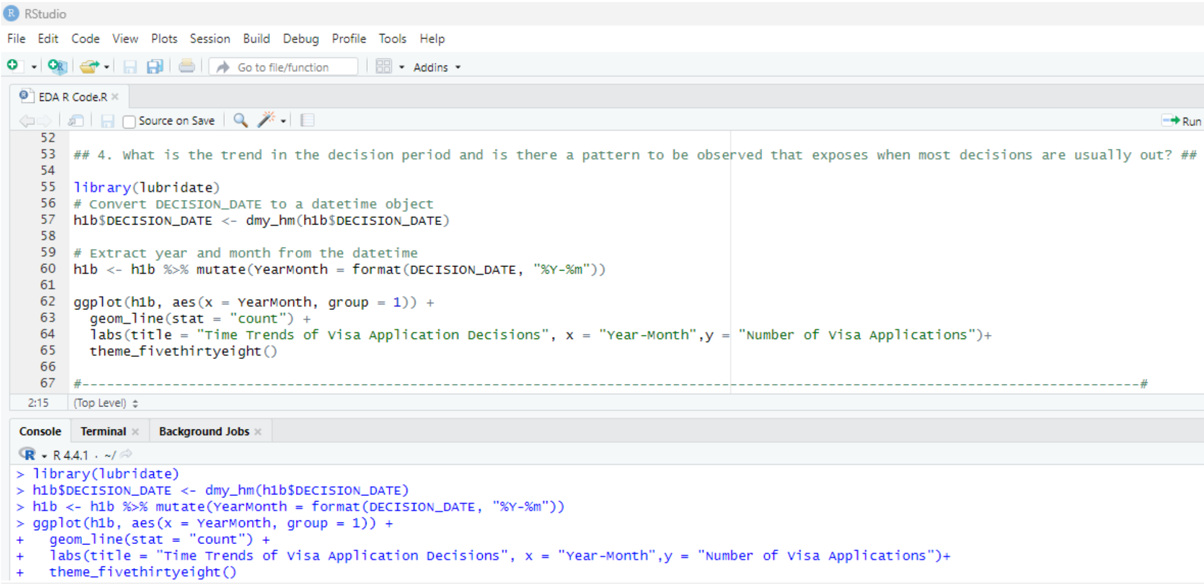
```
> summary(h1b$LCA_CASE_WAGE_RATE_FROM)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
     8.5  60000.0  70000.0  75439.8  90000.0 825000.0 
> ggplot(h1b, aes(x = VISA_CLASS, y = LCA_CASE_WAGE_RATE_FROM)) +
+   geom_boxplot() + coord_flip()+
+   theme_fivethirtyeight()
```

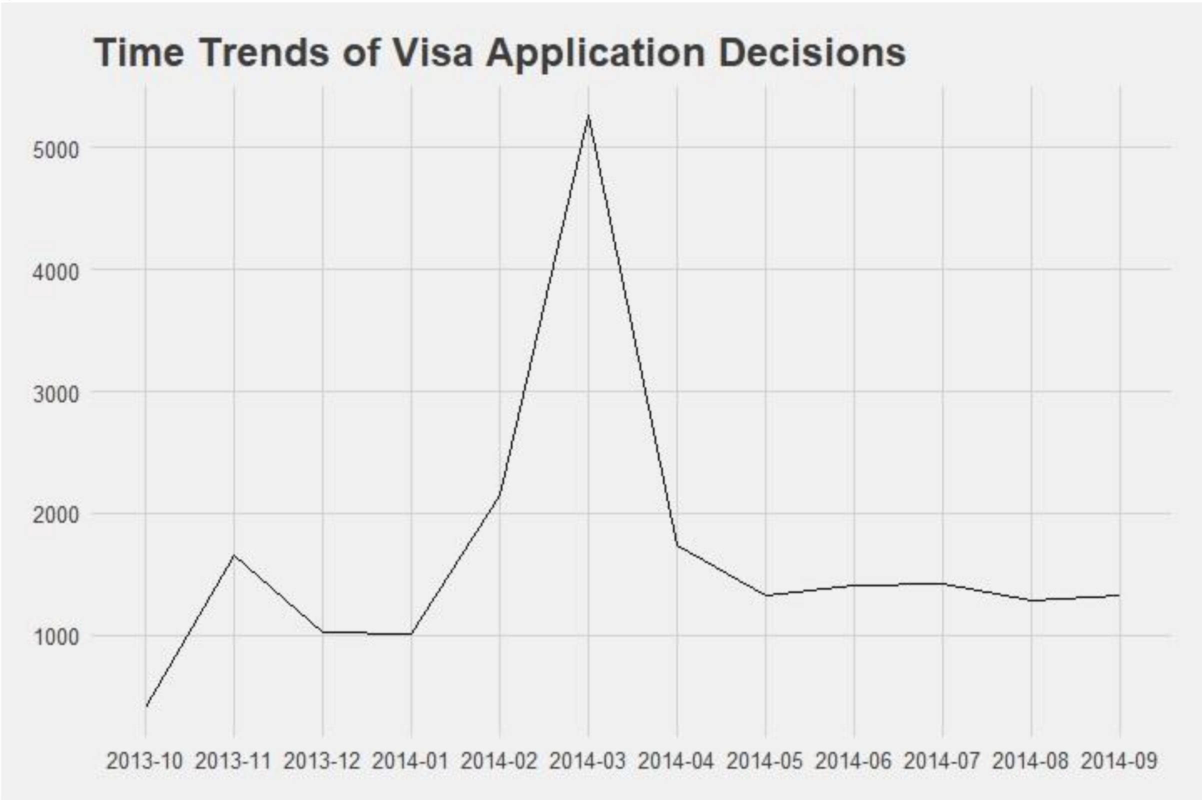


Inference:

- The evidence demonstrates that the median amounts for most visa types are approximately within a similar range.
- However, an outlier is noticeable in the H-1B visa type, where the maximum value exceeds 800,000.

4) What is the trend in the decision period and is there a pattern to be observed that exposes when most decisions are usually out?

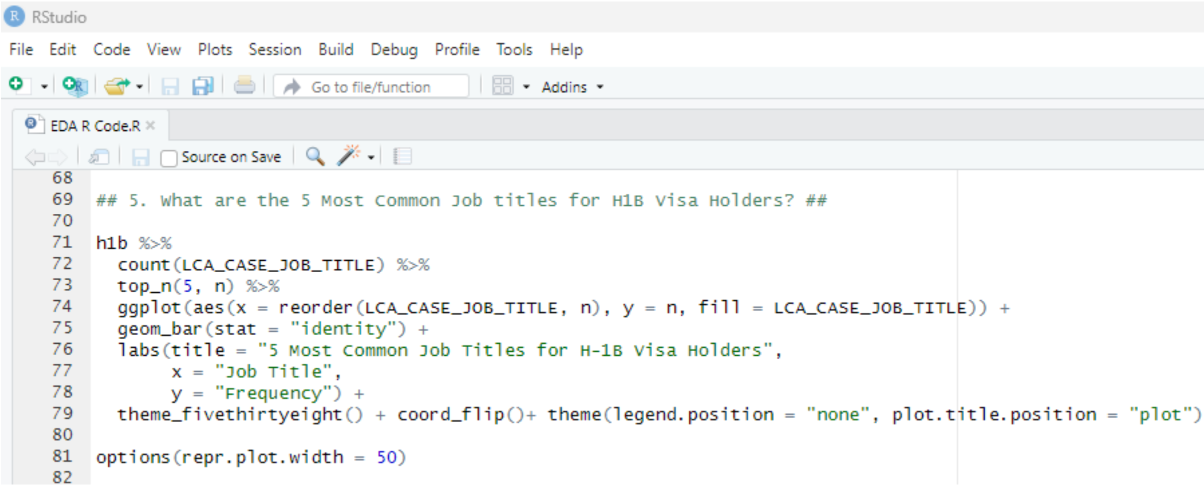


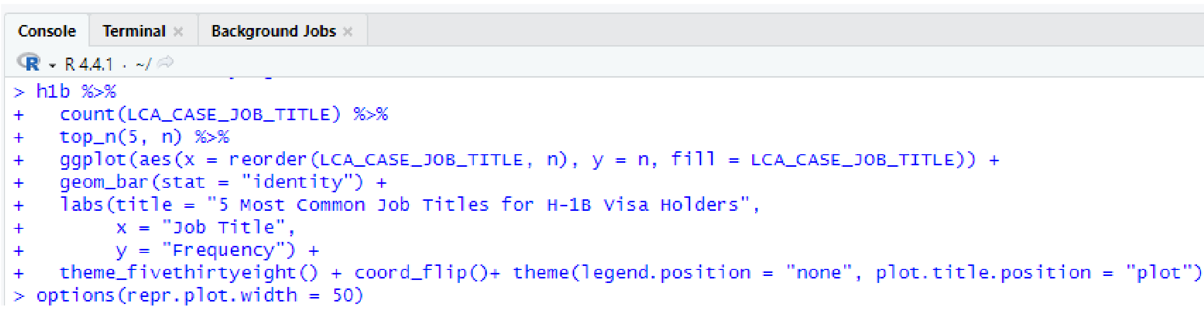


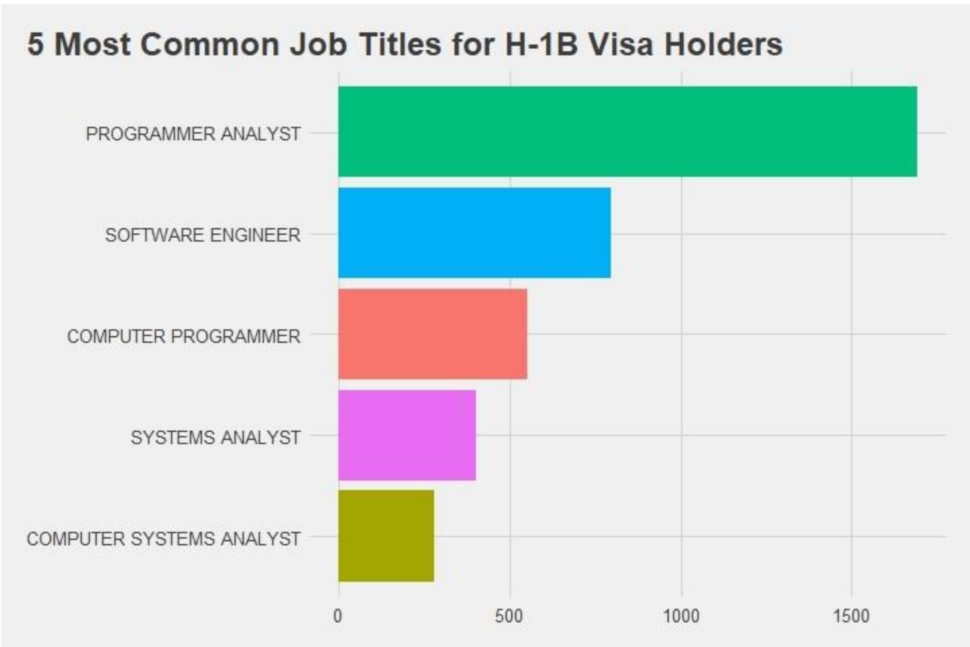
Inference:

- It is noticeable that most results were announced in March for the year 2014.
- Following March, there was a significant decrease, averaging around 1,500, which was also observed before March.

5) What are the 5 Most Common Job titles for H1B Visa Holders?







Inference:

- The visual features the top 5 job roles of H1B Visa Holders.
- **Programmer Analyst is a clear winner in this!**

6) Comparison between Wage Rates & Prevailing Wage

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

EDA R Code.R

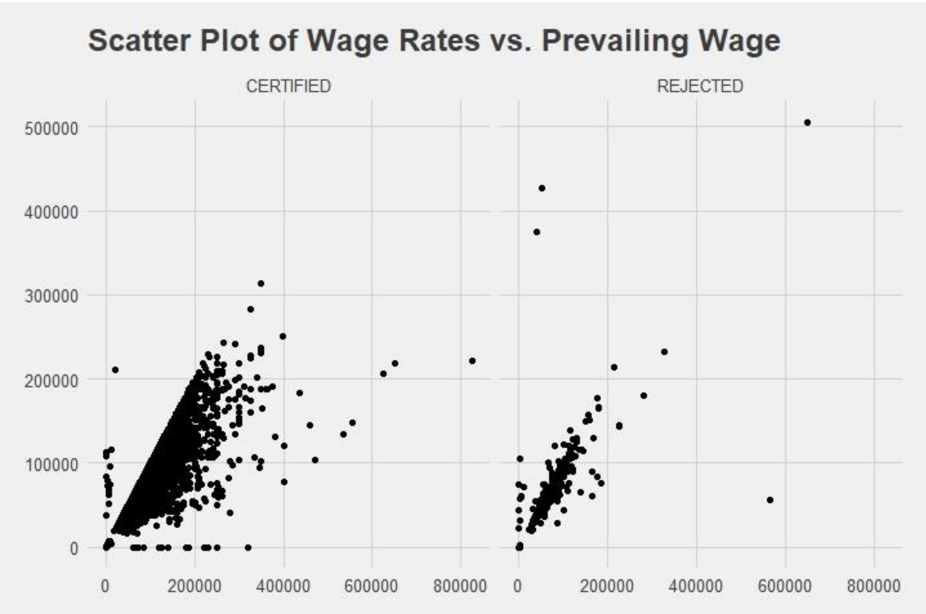
Source on Save

```r
#------------------------------------------------------------------

## 6. Comparison between Wage Rates & Prevailing Wage ##

ggplot(h1b, aes(x = LCA_CASE_WAGE_RATE_FROM, y = PW_1)) +
  facet_wrap(.~STATUS)+
  geom_point() +
  labs(title = "Scatter Plot of Wage Rates vs. Prevailing Wage",
       x = "Wage Rate (From)",
       y = "Prevailing Wage") +
  theme_fivethirtyeight()

#------------------------------------------------------------------
```

Console Terminal Background Jobs

R 4.4.1 · ~/

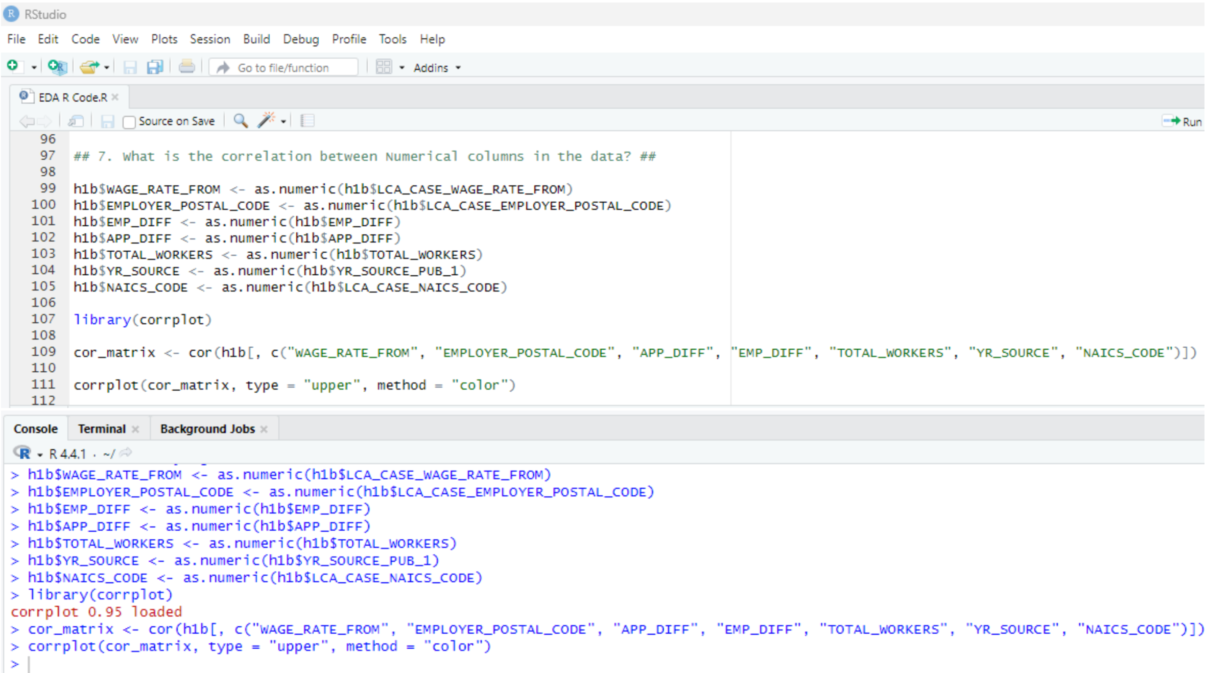
```r
> ggplot(h1b, aes(x = LCA_CASE_WAGE_RATE_FROM, y = PW_1)) +
+   facet_wrap(.~STATUS)+
+   geom_point() +
+   labs(title = "Scatter Plot of Wage Rates vs. Prevailing Wage",
+        x = "Wage Rate (From)",
+        y = "Prevailing Wage") +
+   theme_fivethirtyeight()
```

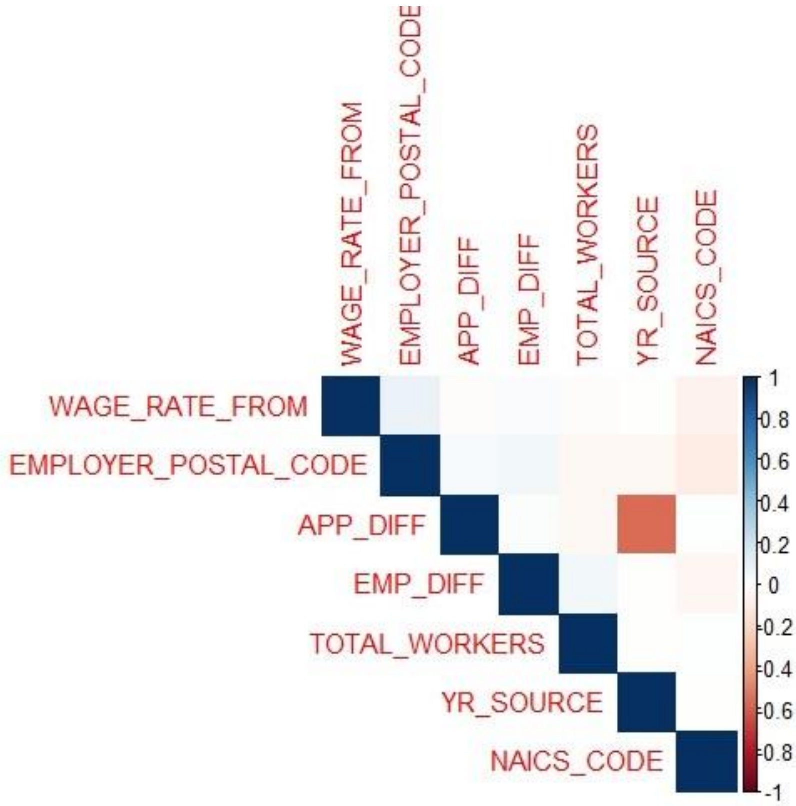


Inference:

- The examination of the scatter plot indicates a robust positive correlation due to the concentrated grouping of most points along a rising trend.
- Some outliers are observed, particularly in Rejected cases.
- A linear relationship is evident between the two variables.

7) What is the correlation between Numerical columns in the data?





Inference:

- Most of these columns exhibit no correlations, with a few displaying subtle positive associations.
- Examples of columns with positive correlations include: Wage rate and employer postal code; Employer Postal Code and the applicant's tenure in the company; Likewise, the applicant's tenure in the company and the total number of workers in that specific company.
- There is a negative correlation between the year of introduction of the prevailing wage source (YR_SOURCE) and the time difference between the applicant's case date and decision date (APP_DIFF).

3. Dimensionality Reduction:
    - Principal Component Analysis (PCA): Reduced dataset to five components, explaining 79.41% variance.

- Loading the data
- Loading packages

```r
# Load necessary libraries
library(corrplot)
library(psych)
library(factoextra)

# Read the dataset
h1b <- read_csv("H1_B-2014.csv")  # Load the dataset. Ensure the file is in your working directory.

# Inspect the structure of the dataset
str(h1b)  # Displays the structure of the data, including column types and a preview of the data.
```

Output

```
12:1    (Top Level) ÷                                                                         R Script
Console   Terminal ×   Background Jobs ×
R 4.4.1 · C:/Users/ABIN/OneDrive/Desktop/BA_with_R/BAR/
chr  (21): LCA_CASE_NUMBER, STATUS, LCA_CASE_SUBMIT, DECISION_DATE, VISA_CLASS, LCA_CASE_EMPLOYMENT_START_D...
dbl  (9): APP_DIFF, EMP_DIFF, LCA_CASE_EMPLOYER_POSTAL_CODE, LCA_CASE_WAGE_RATE_FROM, LCA_CASE_WAGE_RATE_T...

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.
> # Inspect the structure of the dataset
> str(h1b)  # Displays the structure of the data, including column types and a preview of the data.
spc_tbl_ [1,138 × 30] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ LCA_CASE_NUMBER              : chr [1:1138] "I-200-14209-794540" "I-200-13182-969853" "I-200-14076-109575" "I
-200-14232-447516" ...
 $ STATUS                       : chr [1:1138] "CERTIFIED" "CERTIFIED" "CERTIFIED" "CERTIFIED" ...
 $ LCA_CASE_SUBMIT              : chr [1:1138] "29-07-2014 14:48" "08-04-2014 11:57" "17-03-2014 18:37" "20-08-2
014 22:13" ...
 $ DECISION_DATE                : chr [1:1138] "04-08-2014 22:01" "14-04-2014 22:01" "21-03-2014 23:11" "27-08-2
014 22:01" ...
 $ APP_DIFF                     : num [1:1138] 0.21 0.21 0.14 0.23 0.22 ...
 $ VISA_CLASS                   : chr [1:1138] "H-1B" "H-1B" "H-1B" "H-1B" ...
 $ LCA_CASE_EMPLOYMENT_START_DATE: chr [1:1138] "15-08-2014" "14-04-2014" "16-09-2014" "01-10-2014" ...
 $ LCA_CASE_EMPLOYMENT_END_DATE : chr [1:1138] "14-08-2017" "13-04-2017" "16-09-2017" "30-09-2017" ...
 $ EMP_DIFF                     : num [1:1138] 36.5 36.5 36.5 36.5 36.5 ...
 $ LCA_CASE_EMPLOYER_NAME       : chr [1:1138] "DJI TECHNOLOGY LLC" "VEDICSOFT" "SRT IT INCORPORATED" "SYSFORE T
ECHNOLOGIES, INC." ...
 $ LCA_CASE_EMPLOYER_ADDRESS    : chr [1:1138] "6424 SANTA MONICA BLVD." "100 WOOD AVENUE SOUTH" "1999 WABASH AV
ENUE" "11465 JOHNS CREEK PARKWAY" ...
 $ LCA_CASE_EMPLOYER_CITY       : chr [1:1138] "SANTA MONICA" "ISELIN" "SPRINGFIELD" "JOHNS CREEK" ...
 $ LCA_CASE_EMPLOYER_STATE      : chr [1:1138] "CA" "NJ" "IL" "GA" ...
 $ LCA_CASE_EMPLOYER_POSTAL_CODE : num [1:1138] 90404 8830 62704 30097 60064 ...
 $ LCA_CASE_SOC_CODE            : chr [1:1138] "11-1011" "15-1132" "15-1132" "15-1131" ...
 $ LCA_CASE_SOC_NAME            : chr [1:1138] "Chief Executives" "Software Developers, Applications" "Software
Developers, Applications" "Computer Programmers" ...
```

1. What is the correlation matrix for selected variables:

```
# Compute the correlation matrix for selected variables
cor_matrix <- cor(h1b[, c("LCA_CASE_WAGE_RATE_FROM", "LCA_CASE_EMPLOYER_POSTAL_CODE",
                          "APP_DIFF", "EMP_DIFF", "TOTAL_WORKERS", "YR_SOURCE_PUB_1",
                          "LCA_CASE_NAICS_CODE")])
cor_matrix  # Displays the correlation matrix of the selected numerical columns.
```

Output :

```
> # Compute the correlation matrix for selected variables
> cor_matrix <- cor(h1b[, c("LCA_CASE_WAGE_RATE_FROM", "LCA_CASE_EMPLOYER_POSTAL_CODE",
+                          "APP_DIFF", "EMP_DIFF", "TOTAL_WORKERS", "YR_SOURCE_PUB_1",
+                          "LCA_CASE_NAICS_CODE")])
> cor_matrix  # Displays the correlation matrix of the selected numerical columns.
                              LCA_CASE_WAGE_RATE_FROM LCA_CASE_EMPLOYER_POSTAL_CODE    APP_DIFF   EMP_DIFF
LCA_CASE_WAGE_RATE_FROM                    1.00000000                    0.02297687 -0.01729757 0.05316424
LCA_CASE_EMPLOYER_POSTAL_CODE              0.02297687                    1.00000000  0.02711751 0.08207685
APP_DIFF                                  -0.01729757                    0.02711751  1.00000000 0.01603995
EMP_DIFF                                   0.05316424                    0.08207685  0.01603995 1.00000000
TOTAL_WORKERS                             -0.01268854                   -0.04827070 -0.03059037 0.05875255
YR_SOURCE_PUB_1                            0.00705286                    0.05404782 -0.38644500 0.03995793
LCA_CASE_NAICS_CODE                        0.03870463                   -0.07454110  0.04928138 0.04667738
                              TOTAL_WORKERS YR_SOURCE_PUB_1 LCA_CASE_NAICS_CODE
LCA_CASE_WAGE_RATE_FROM        -0.012688539     0.007052860         0.038704627
LCA_CASE_EMPLOYER_POSTAL_CODE  -0.048270699     0.054047816        -0.074541097
APP_DIFF                       -0.030590368    -0.386445001         0.049281385
EMP_DIFF                        0.058752550     0.039957930         0.046677385
TOTAL_WORKERS                   1.000000000     0.033764902         0.007638878
YR_SOURCE_PUB_1                 0.033764902     1.000000000        -0.008487363
LCA_CASE_NAICS_CODE             0.007638878    -0.008487363         1.000000000
```

2) Perform Principal Component Analysis (PCA) with scaling:

```
# Perform Principal Component Analysis (PCA) with scaling
h1b_pca <- prcomp(h1b[, c("LCA_CASE_WAGE_RATE_FROM", "LCA_CASE_EMPLOYER_POSTAL_CODE",
                          "APP_DIFF", "EMP_DIFF", "TOTAL_WORKERS", "YR_SOURCE_PUB_1",
                          "LCA_CASE_NAICS_CODE")],
                  scale = TRUE)  # PCA normalizes variables to have unit variance.
h1b_pca  # Displays the PCA results, including the rotation matrix.

# Summary of PCA
summary(h1b_pca)  # Provides explained variance and cumulative variance for each principal component.
```

Output :

```
+                          "APP_DIFF", "EMP_DIFF", "TOTAL_WORKERS", "YR_SOURCE_PUB_1",
+                          "LCA_CASE_NAICS_CODE")],
+                  scale = TRUE)  # PCA normalizes variables to have unit variance.
> h1b_pca  # Displays the PCA results, including the rotation matrix.
Standard deviations (1, .., p=7):
[1] 1.1827490 1.0557476 1.0449319 1.0071075 0.9691433 0.9175385 0.7741047

Rotation (n x k) = (7 x 7):
                                      PC1         PC2         PC3         PC4         PC5         PC6
LCA_CASE_WAGE_RATE_FROM        0.04206434 -0.45967161  0.11247372 -0.54030739 -0.69088100 -0.05638995
LCA_CASE_EMPLOYER_POSTAL_CODE  0.06892545 -0.46491833 -0.62414703  0.10287340  0.16885038 -0.57264689
APP_DIFF                      -0.69202579 -0.14509440 -0.05093141  0.10356989  0.01401352 -0.06914707
EMP_DIFF                       0.06659708 -0.70626160  0.10519845  0.27153786  0.22626070  0.59772724
TOTAL_WORKERS                  0.11043650 -0.09558526  0.45169501  0.69946402 -0.37266594 -0.38258642
YR_SOURCE_PUB_1                0.69864001 -0.01729196  0.01049945 -0.06071024  0.15146282 -0.06133974
LCA_CASE_NAICS_CODE           -0.09925188 -0.20798938  0.61643394 -0.34653977  0.53005832 -0.39585554
                                      PC7
LCA_CASE_WAGE_RATE_FROM        0.04305591
LCA_CASE_EMPLOYER_POSTAL_CODE -0.15007623
APP_DIFF                       0.69408177
EMP_DIFF                      -0.05906014
TOTAL_WORKERS                 -0.01169069
YR_SOURCE_PUB_1                0.69361633
LCA_CASE_NAICS_CODE           -0.09563175
> # Summary of PCA
> summary(h1b_pca)  # Provides explained variance and cumulative variance for each principal component.
Importance of components:
                          PC1    PC2    PC3    PC4    PC5    PC6     PC7
Standard deviation     1.1827 1.0557 1.0449 1.0071 0.9691 0.9175 0.77410
Proportion of Variance 0.1998 0.1592 0.1560 0.1449 0.1342 0.1203 0.08561
Cumulative Proportion  0.1998 0.3591 0.5151 0.6600 0.7941 0.9144 1.00000
```

3. All variances

```
# Eigenvalues (variance explained by each principal component)
(eigen_h1b <- h1b_pca$sdev^2)  # Calculate eigenvalues by squaring the standard deviations of PCs.
names(eigen_h1b) <- paste("PC", 1:7, sep = "")  # Assign names to eigenvalues.
eigen_h1b  # Display eigenvalues.

# Total variance explained
sumlambdas <- sum(eigen_h1b)  # Sum of eigenvalues should equal the number of variables.
sumlambdas

# Proportion of variance explained by each component
propvar <- eigen_h1b / sumlambdas  # Calculate the proportion of variance for each component.
propvar

# Cumulative proportion of variance
cumvar_h1b <- cumsum(propvar)  # Cumulative variance explained.
cumvar_h1b
```

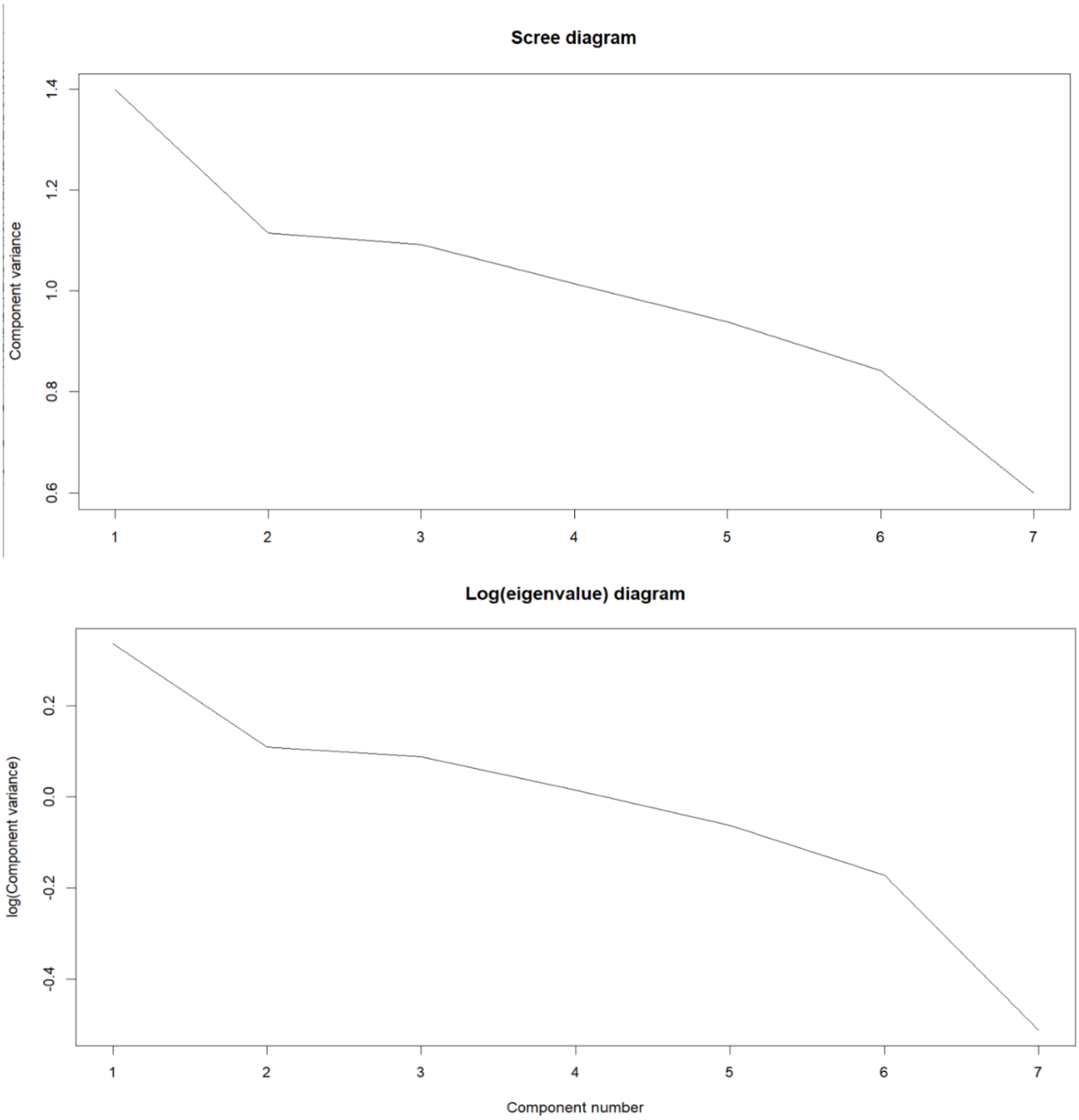
Output:

```
> (eigen_h1b <- h1b_pca$sdev^2)  # Calculate eigenvalues by squaring the standard deviations of PCs.
[1] 1.3988953 1.1146029 1.0918827 1.0142655 0.9392388 0.8418768 0.5992380
> names(eigen_h1b) <- paste("PC", 1:7, sep = "")  # Assign names to eigenvalues.
> eigen_h1b  # Display eigenvalues.
      PC1       PC2       PC3       PC4       PC5       PC6       PC7 
1.3988953 1.1146029 1.0918827 1.0142655 0.9392388 0.8418768 0.5992380 
> 
> # Total variance explained
> sumlambdas <- sum(eigen_h1b)  # Sum of eigenvalues should equal the number of variables.
> sumlambdas
[1] 7
> (eigen_h1b <- h1b_pca$sdev^2)  # Calculate eigenvalues by squaring the standard deviations of PCs.
[1] 1.3988953 1.1146029 1.0918827 1.0142655 0.9392388 0.8418768 0.5992380
> names(eigen_h1b) <- paste("PC", 1:7, sep = "")  # Assign names to eigenvalues.
> eigen_h1b  # Display eigenvalues.
      PC1       PC2       PC3       PC4       PC5       PC6       PC7 
1.3988953 1.1146029 1.0918827 1.0142655 0.9392388 0.8418768 0.5992380 
> # Total variance explained
> sumlambdas <- sum(eigen_h1b)  # Sum of eigenvalues should equal the number of variables.
> sumlambdas
[1] 7
> # Proportion of variance explained by each component
> propvar <- eigen_h1b / sumlambdas  # Calculate the proportion of variance for each component.
> propvar
       PC1        PC2        PC3        PC4        PC5        PC6        PC7 
0.19984218 0.15922899 0.15598324 0.14489507 0.13417697 0.12026812 0.08560543 
> # Cumulative proportion of variance
> cumvar_h1b <- cumsum(propvar)  # Cumulative variance explained.
> cumvar_h1b
      PC1       PC2       PC3       PC4       PC5       PC6       PC7 
0.1998422 0.3590712 0.5150544 0.6599495 0.7941265 0.9143946 1.0000000 
```

4. Scree plot: visualize component variance

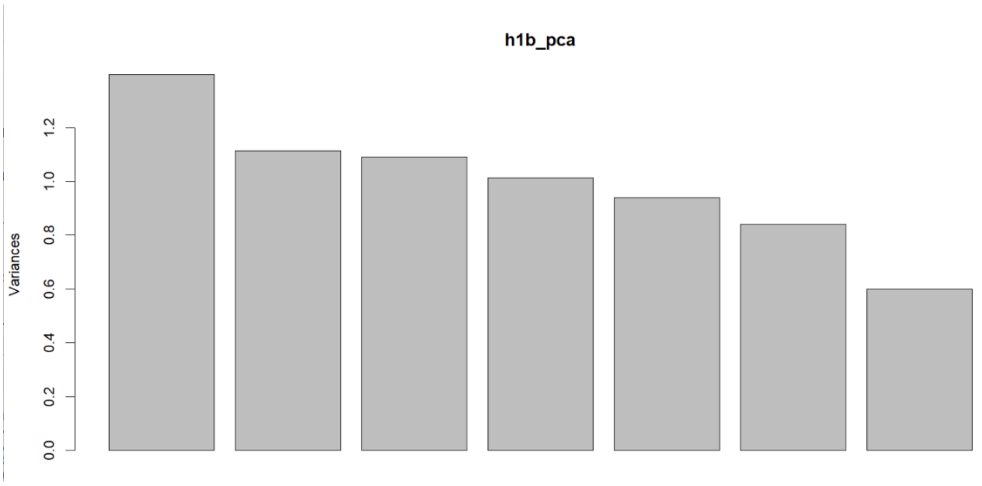
```
# Scree plot: visualize component variance
plot(eigen_h1b,
     xlab = "Component number",
     ylab = "Component variance",
     type = "l",
     main = "Scree diagram")  # Helps identify significant components (elbow point).

# Logarithmic scree plot
plot(log(eigen_h1b),
     xlab = "Component number",
     ylab = "log(Component variance)",
     type = "l",
     main = "Log(eigenvalue) diagram")  # Visualizes eigenvalues on a log scale.
```

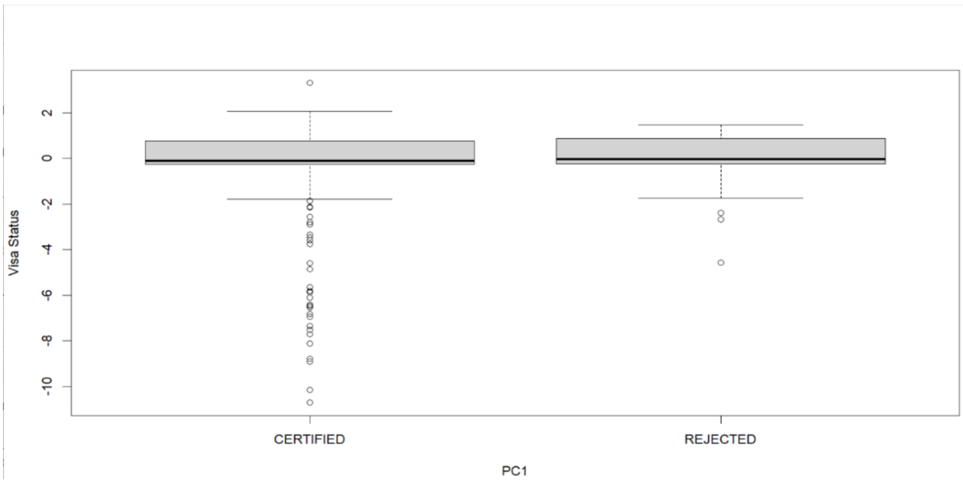
Output

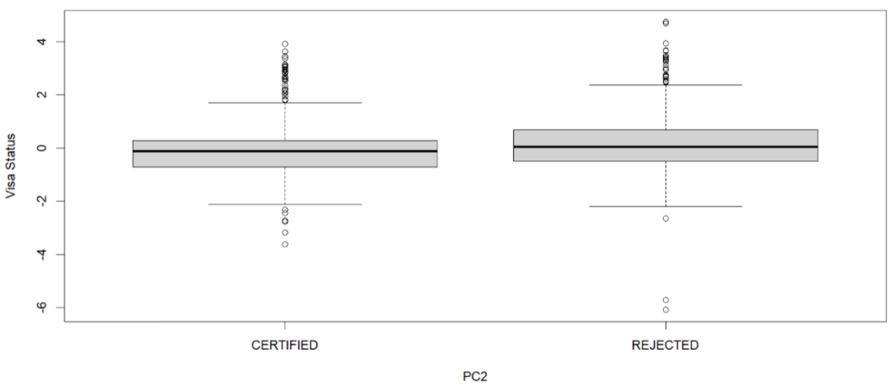


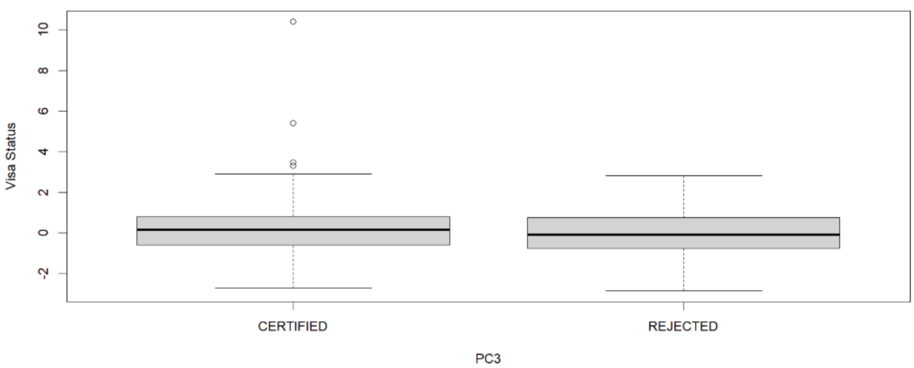
5. Plotting of PCA components

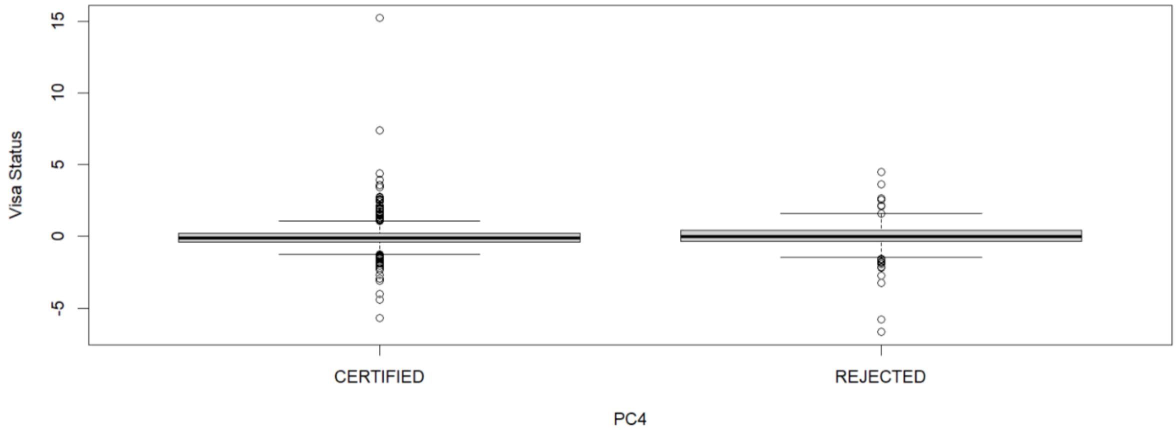
```
# Basic plot of PCA results
plot(h1b_pca)  # Default PCA plot showing variance explained by each principal com
# Convert STATUS column to factor
h1b$STATUS <- as.factor(h1b$STATUS)  # Converts the STATUS column to a categorica
# Box plots of Visa Status against each principal component
out <- sapply(1:7, function(i) {
  plot(h1b$STATUS, h1b_pca$x[, i],
       xlab = paste("PC", i, sep = ""),
       ylab = "Visa Status")  # Visualize how the visa status relates to PCs.
})
```

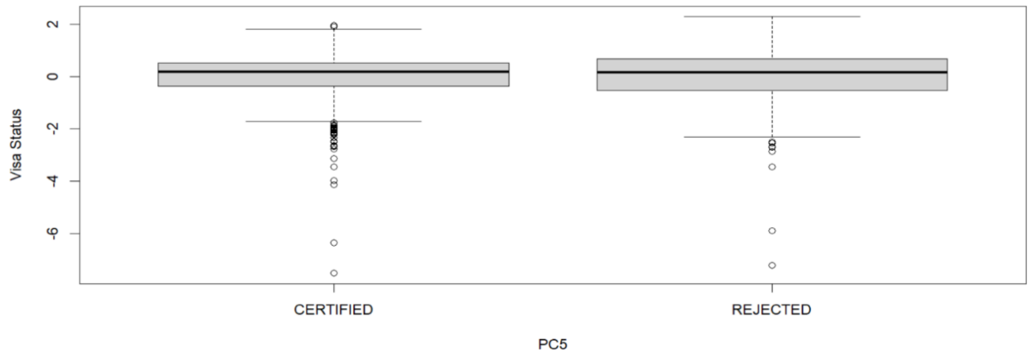

h1b_pca
Variances
0.0
0.2
0.4
0.6
0.8
1.0
1.2

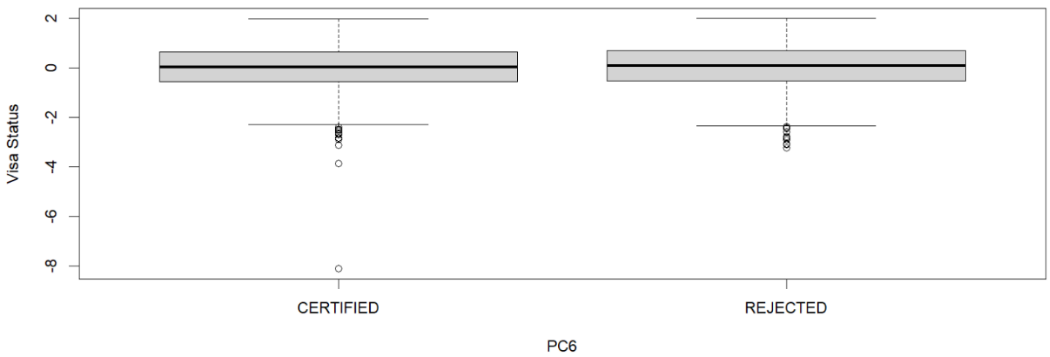

Visa Status
2
0
-2
-4
-6
-8
-10
CERTIFIED
REJECTED
PC1

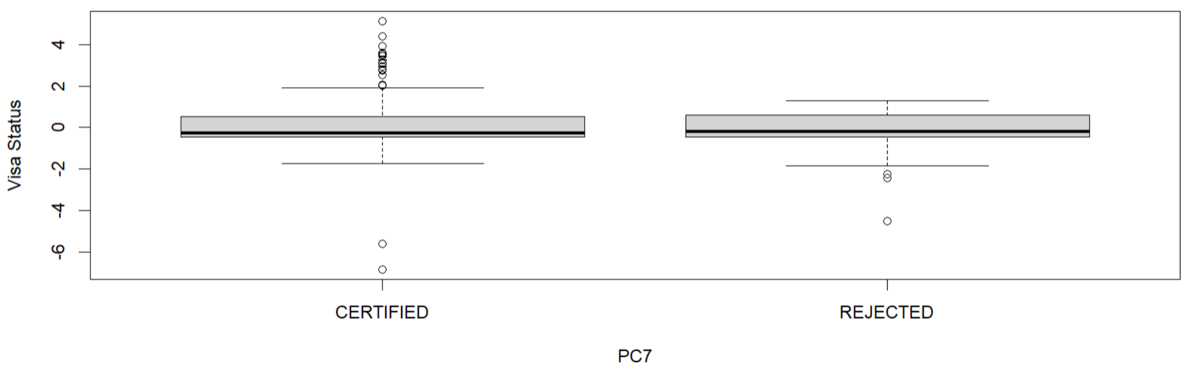

Visa Status
4
2
0
-2
-4
-6
CERTIFIED
REJECTED
PC2


Visa Status
10
8
6
4
2
0
-2
CERTIFIED
REJECTED
PC3


Visa Status
15
10
5
0
-5
CERTIFIED
REJECTED
PC4


Visa Status
2
0
-2
-4
-6
CERTIFIED
REJECTED
PC5

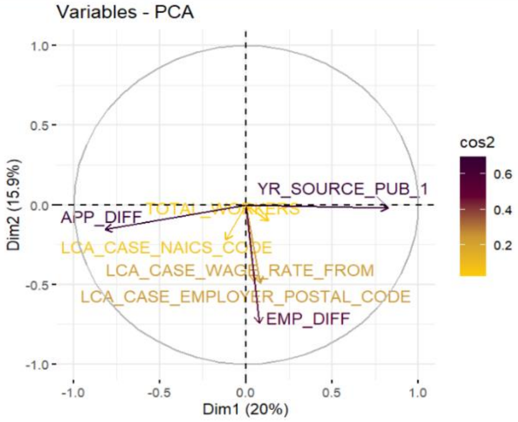

Visa Status
2
0
-2
-4
-6
-8
CERTIFIED
REJECTED
PC6

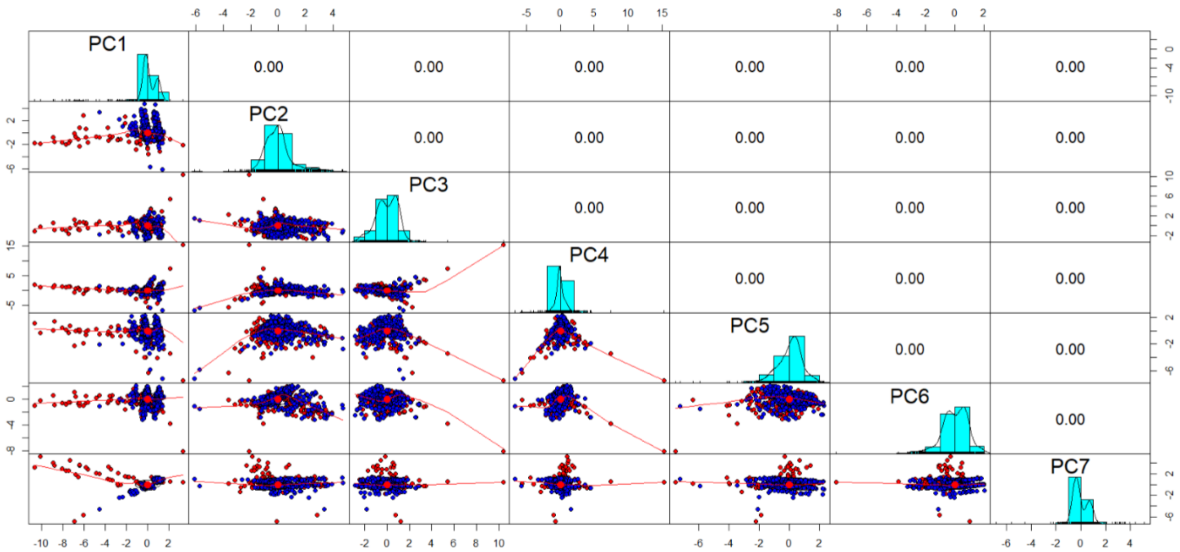

Visa Status
4
2
0
-2
-4
-6
CERTIFIED
REJECTED
PC7

6. Visualise the PCA variables -

```
# Visualize PCA variables
fviz_pca_var(h1b_pca,
             col.var = "cos2",  # Color by the quality of representation (cos2)
             gradient.cols = c("#FFCC00", "#CC9933", "#660033", "#330033"),  #
             repel = TRUE)  # Avoid overlapping labels for better readability.

# Pairwise scatter plots of principal components
pairs.panels(h1b_pca$x,
             gap = 0,
             bg = c("red", "blue")[h1b$STATUS],  # Use color to distinguish vis
             pch = 21)  # Pair plot showing relationships between principal com
```

Output:





Inference

- **Purpose of PCA**: PCA reduces the 30-dimensional dataset into 5 uncorrelated components, capturing 79.41% of the total variance, simplifying analysis while retaining essential information.
- **Insights Gained**: It highlights variable correlations (e.g., wage rate and total workers) and uncovers patterns, helping interpret key trends like separation by visa status.
- **Visualization Benefits**: PCA enables clear visualization of data clusters and relationships, making complex data more accessible for analysis and decision-making.

Modelling Techniques

1) **Loading required libraries and reading pruned dataset.**

```
# Required Libraries
library(ggplot2)
library(caret)
library(pROC)
library(nnet)
library(rpart)

# Load Data
h1b  <- read.csv("C:/Users/yasht/Downloads/H1_B-2014 (3).csv", header=T)
```

2) **Data Preprocessing**: Converts relevant columns to numeric or factor types, subsets features, and normalizes continuous variables for modelling.

```
# Data Preprocessing
h1b$WAGE_RATE_FROM <- as.numeric(h1b$LCA_CASE_WAGE_RATE_FROM)
h1b$EMPLOYER_POSTAL_CODE <- as.numeric(h1b$LCA_CASE_EMPLOYER_POSTAL_CODE)
h1b$EMP_DIFF <- as.numeric(h1b$EMP_DIFF)
h1b$APP_DIFF <- as.numeric(h1b$APP_DIFF)
h1b$TOTAL_WORKERS <- as.numeric(h1b$TOTAL_WORKERS)
h1b$YR_SOURCE <- as.numeric(h1b$YR_SOURCE_PUB_1)
h1b$NAICS_CODE <- as.numeric(h1b$LCA_CASE_NAICS_CODE)

h1b$STATUS <- as.factor(h1b$STATUS)

# Subset the relevant features
h1b_df <- h1b[, c("STATUS", "WAGE_RATE_FROM", "EMPLOYER_POSTAL_CODE", "APP_DIFF", "EMP_DIFF", "TOTAL_WORKERS", "YR_SOURCE", "NAICS_CODE")]

# Convert STATUS to numeric for neural network
h1b_df$STATUS_NUM <- as.numeric(h1b_df$STATUS) - 1  # CERTIFIED = 0, REJECTED = 1

# Normalize continuous variables for neural network
normalize <- function(x) (x - min(x)) / (max(x) - min(x))
h1b_nn <- h1b_df
h1b_nn[, c("WAGE_RATE_FROM", "EMPLOYER_POSTAL_CODE", "APP_DIFF", "EMP_DIFF", "TOTAL_WORKERS", "YR_SOURCE", "NAICS_CODE")] <-
  lapply(h1b_nn[, c("WAGE_RATE_FROM", "EMPLOYER_POSTAL_CODE", "APP_DIFF", "EMP_DIFF", "TOTAL_WORKERS", "YR_SOURCE", "NAICS_CODE")], normalize)
```

1. Linear Regression: Fits a linear regression model, evaluates performance using AUC, accuracy, and confusion matrix, and computes predictions.

```
# ---- 1. Linear Regression Model ----
linear_reg <- lm(STATUS_NUM ~ ., data=h1b_df[, -1])  # Exclude the factor STATUS
summary(linear_reg)

# Predictions and ROC for Linear Regression
linear_probs <- predict(linear_reg, newdata=h1b_df)
roc_linear <- roc(h1b_df$STATUS_NUM, linear_probs)
auc_linear <- auc(roc_linear)
print(paste("Linear Regression AUC:", auc_linear))

# Binary predictions for accuracy (threshold = 0.5)
linear_pred <- as.factor(ifelse(linear_probs > 0.5, "REJECTED", "CERTIFIED"))
conf_linear <- confusionMatrix(linear_pred, h1b_df$STATUS)
accuracy_linear <- conf_linear$overall['Accuracy']
print(paste("Linear Regression Accuracy:", accuracy_linear))

# Print confusion matrix for Linear Regression
print("Linear Regression Confusion Matrix:")
print(conf_linear)
```

```
> linear_reg <- lm(STATUS_NUM ~ ., data=h1b_df[, -1])  # Exclude the factor STATUS
> summary(linear_reg)

Call:
lm(formula = STATUS_NUM ~ ., data = h1b_df[, -1])

Residuals:
     Min       1Q   Median       3Q      Max 
-0.60872 -0.37595 -0.32276  0.57089  1.21552 

Coefficients:
                       Estimate  Std. Error t value  Pr(>|t|)    
(Intercept)          -7.3731e+01  4.9098e+01 -1.5017 0.1334491    
WAGE_RATE_FROM       -9.9693e-07  2.8611e-07 -3.4844 0.0005122 ***
EMPLOYER_POSTAL_CODE -2.7411e-07  4.2011e-07 -0.6525 0.5142258    
APP_DIFF             -1.5104e-02  4.7222e-03 -3.1986 0.0014192 ** 
EMP_DIFF             -5.4317e-03  2.1512e-03 -2.5250 0.0117070 *  
TOTAL_WORKERS        -5.4651e-03  3.1483e-03 -1.7359 0.0828608 .  
YR_SOURCE             3.7006e-02  2.4389e-02  1.5173 0.1294636    
NAICS_CODE           -2.2773e-07  7.3107e-08 -3.1150 0.0018860 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.47484 on 1130 degrees of freedom
Multiple R-squared:  0.045838,	Adjusted R-squared:  0.039927 
F-statistic:  7.755 on 7 and 1130 DF,  p-value: 3.4818e-09
```

```
> auc_linear <- auc(roc_linear)
> print(paste("Linear Regression AUC:", auc_linear))
[1] "Linear Regression AUC: 0.630034882190338"
>
> # Binary predictions for accuracy (threshold = 0.5)
> linear_pred <- as.factor(ifelse(linear_probs > 0.5, "REJECTED", "CERTIFIED"))
> conf_linear <- confusionMatrix(linear_pred, h1b_df$STATUS)
> accuracy_linear <- conf_linear$overall['Accuracy']
> print(paste("Linear Regression Accuracy:", accuracy_linear))
[1] "Linear Regression Accuracy: 0.642355008787346"
>
> # Print confusion matrix for Linear Regression
> print("Linear Regression Confusion Matrix:")
[1] "Linear Regression Confusion Matrix:"
> print(conf_linear)
Confusion Matrix and Statistics

           Reference
Prediction  CERTIFIED REJECTED
  CERTIFIED       670      367
  REJECTED         40       61

               Accuracy : 0.64236
                 95% CI : (0.61373, 0.67024)
    No Information Rate : 0.6239
    P-Value [Acc > NIR] : 0.10455

                  Kappa : 0.1016

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.94366
            Specificity : 0.14252
         Pos Pred Value : 0.64609
         Neg Pred Value : 0.60396
             Prevalence : 0.62390
```

- Accuracy: 64.24%, AUC: 0.63.
- Struggled to differentiate between certified and rejected cases.

2. Neural Network: Trains a neural network with class weights for imbalance, evaluates performance using AUC, accuracy, and confusion matrix, and computes predictions.

```
# ---- 2. Neural Network Model ----
# Train neural network with more hidden units and class weights
nn_model <- nnet(STATUS_NUM ~ ., data=h1b_nn[, -1], size=10, linout=FALSE, maxit=500,
                 class.weights = c(0.1, 0.9))  # Adjust class weights to handle class imbalance

# Predictions and ROC for Neural Network
nn_probs <- predict(nn_model, newdata=h1b_nn)
roc_nn <- roc(h1b_nn$STATUS_NUM, nn_probs)
auc_nn <- auc(roc_nn)
print(paste("Neural Network AUC:", auc_nn))

# Binary predictions for accuracy (threshold = 0.5)
nn_pred <- as.factor(ifelse(nn_probs > 0.5, "REJECTED", "CERTIFIED"))
conf_nn <- confusionMatrix(nn_pred, h1b_df$STATUS)
accuracy_nn <- conf_nn$overall['Accuracy']
print(paste("Neural Network Accuracy:", accuracy_nn))

# Print confusion matrix for Neural Network
print("Neural Network Confusion Matrix:")
print(conf_nn)
```

```
> auc_nn <- auc(roc_nn)
> print(paste("Neural Network AUC:", auc_nn))
[1] "Neural Network AUC: 0.78229564301698"
>
> # Binary predictions for accuracy (threshold = 0.5)
> nn_pred <- as.factor(ifelse(nn_probs > 0.5, "REJECTED", "CERTIFIED"))
> conf_nn <- confusionMatrix(nn_pred, h1b_df$STATUS)
> accuracy_nn <- conf_nn$overall['Accuracy']
> print(paste("Neural Network Accuracy:", accuracy_nn))
[1] "Neural Network Accuracy: 0.746924428822496"
>
> # Print confusion matrix for Neural Network
> print("Neural Network Confusion Matrix:")
[1] "Neural Network Confusion Matrix:"
> print(conf_nn)
Confusion Matrix and Statistics

           Reference
Prediction  CERTIFIED REJECTED
  CERTIFIED       638      216
  REJECTED         72      212

               Accuracy : 0.74692
                 95% CI : (0.72061, 0.77196)
    No Information Rate : 0.6239
    P-Value [Acc > NIR] : < 2.22e-16

                  Kappa : 0.42212

 Mcnemar's Test P-Value : < 2.22e-16

               Accuracy : 0.74692
                 95% CI : (0.72061, 0.77196)
    No Information Rate : 0.6239
    P-Value [Acc > NIR] : < 2.22e-16

                  Kappa : 0.42212

 Mcnemar's Test P-Value : < 2.22e-16

            Sensitivity : 0.89859
            Specificity : 0.49533
         Pos Pred Value : 0.74707
         Neg Pred Value : 0.74648
             Prevalence : 0.62390
         Detection Rate : 0.56063
   Detection Prevalence : 0.75044
      Balanced Accuracy : 0.69696

       'Positive' Class : CERTIFIED
```

- Accuracy: 74.17%, AUC: 0.76.
- Captured non-linear relationships better than linear regression.

3. Decision Tree (Best Model): Builds a decision tree, evaluates performance using AUC, accuracy, and confusion matrix, and computes predictions.

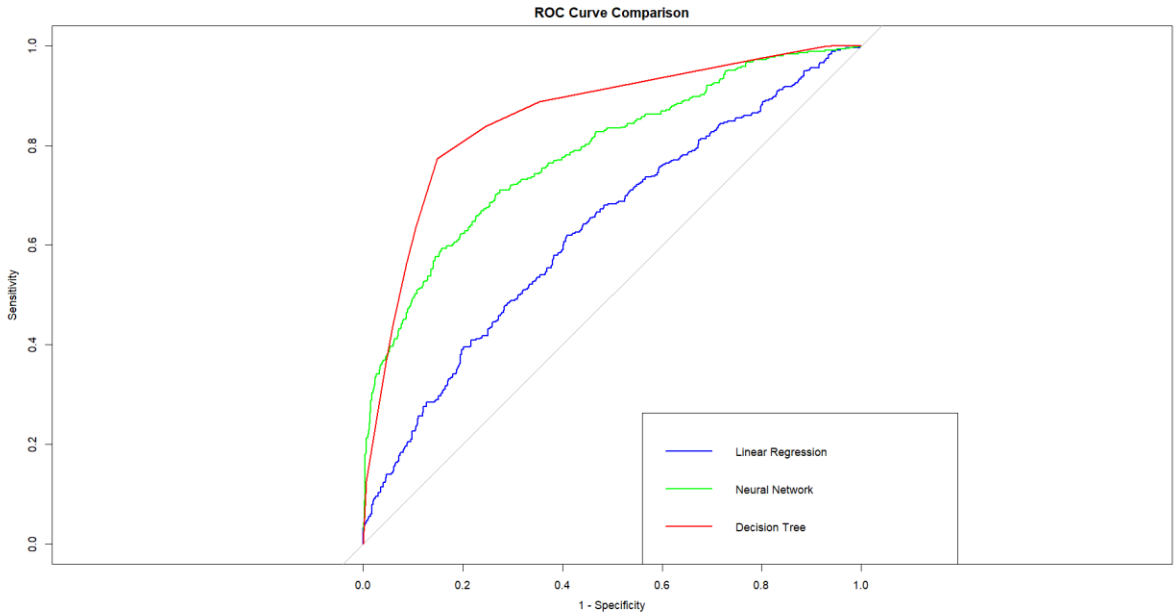
```r
# ---- 3. Decision Tree Model ----
dt_model <- rpart(STATUS_NUM ~ ., data=h1b_df[, -1], method="class")

# Predictions and ROC for Decision Tree
dt_probs <- predict(dt_model, newdata=h1b_df, type="prob")[,2]  # Probability for "REJECTED"
roc_dt <- roc(h1b_df$STATUS_NUM, dt_probs)
auc_dt <- auc(roc_dt)
print(paste("Decision Tree AUC:", auc_dt))

# Binary predictions for accuracy (threshold = 0.5)
dt_pred <- as.factor(ifelse(dt_probs > 0.5, "REJECTED", "CERTIFIED"))
conf_dt <- confusionMatrix(dt_pred, h1b_df$STATUS)
accuracy_dt <- conf_dt$overall['Accuracy']
print(paste("Decision Tree Accuracy:", accuracy_dt))

# Print confusion matrix for Decision Tree
print("Decision Tree Confusion Matrix:")
print(conf_dt)
```

```
> print(paste("Decision Tree AUC:", auc_dt))
[1] "Decision Tree AUC: 0.856124127945242"
>
> # Binary predictions for accuracy (threshold = 0.5)
> dt_pred <- as.factor(ifelse(dt_probs > 0.5, "REJECTED", "CERTIFIED"))
> conf_dt <- confusionMatrix(dt_pred, h1b_df$STATUS)
> accuracy_dt <- conf_dt$overall['Accuracy']
> print(paste("Decision Tree Accuracy:", accuracy_dt))
[1] "Decision Tree Accuracy: 0.821616871704745"
>
> # Print confusion matrix for Decision Tree
> print("Decision Tree Confusion Matrix:")
[1] "Decision Tree Confusion Matrix:"
> print(conf_dt)
Confusion Matrix and Statistics

           Reference
Prediction  CERTIFIED REJECTED
  CERTIFIED       604       97
  REJECTED        106      331

               Accuracy : 0.82162
                 95% CI : (0.79811, 0.84345)
    No Information Rate : 0.6239
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.62147

 Mcnemar's Test P-Value : 0.57446

            Sensitivity : 0.85070
            Specificity : 0.77336
         Pos Pred Value : 0.86163
         Neg Pred Value : 0.75744
             Prevalence : 0.62390
         Detection Rate : 0.53076
   Detection Prevalence : 0.61599
      Balanced Accuracy : 0.81203
```

- Accuracy: 82.16%, AUC: 0.86.
- Provided interpretable insights into decision-making.

4. **ROC Curve Comparison**: Plots ROC curves for all models to visually compare predictive performance.

```
# ---- 4. ROC Curve Comparison ----
plot(roc_linear, col="blue", main="ROC Curve Comparison", legacy.axes=TRUE)
plot(roc_nn, col="green", add=TRUE)
plot(roc_dt, col="red", add=TRUE)
legend("bottomright", legend=c("Linear Regression", "Neural Network", "Decision Tree"),
       col=c("blue", "green", "red"), lwd=2)
```



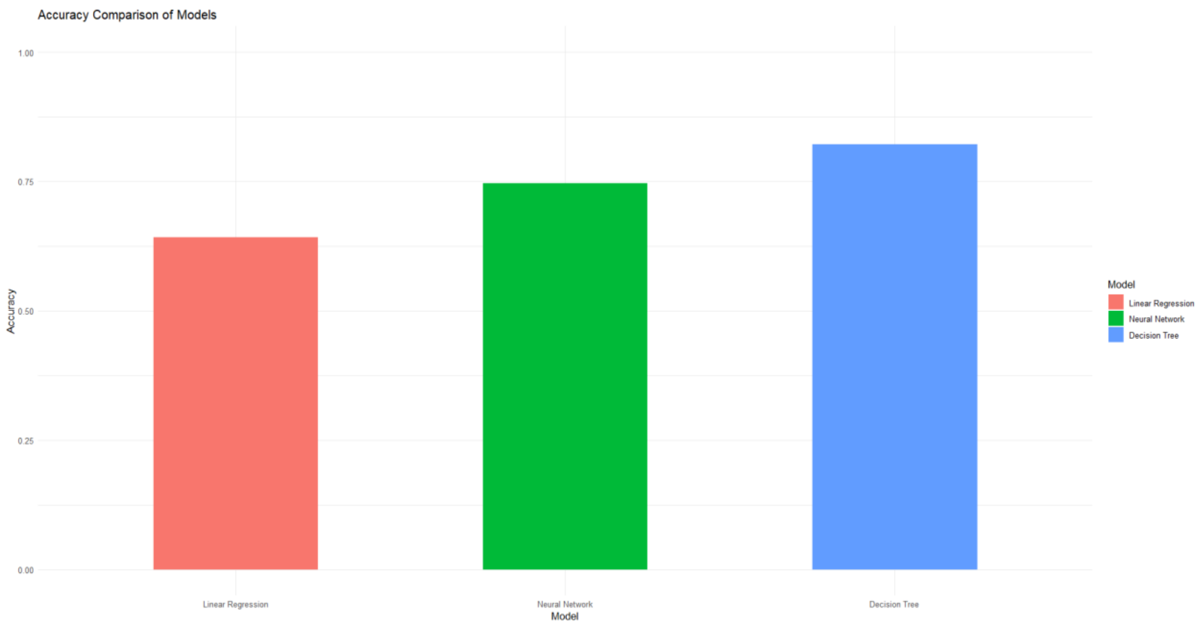
5. **Performance Comparison Table**: Creates a summary table of accuracy and AUC for each model.

```
# ---- 5. Performance Comparison Table ----
comparison_table <- data.frame(
  Model = c("Linear Regression", "Neural Network", "Decision Tree"),
  Accuracy = c(accuracy_linear, accuracy_nn, accuracy_dt),
  AUC = c(auc_linear, auc_nn, auc_dt)
)

print(comparison_table)
```

```
> # ---- 5. Performance Comparison Table ----
> comparison_table <- data.frame(
+   Model = c("Linear Regression", "Neural Network", "Decision Tree"),
+   Accuracy = c(accuracy_linear, accuracy_nn, accuracy_dt),
+   AUC = c(auc_linear, auc_nn, auc_dt)
+ )
>
> print(comparison_table)
              Model   Accuracy        AUC
1 Linear Regression 0.64235501 0.63003488
2    Neural Network 0.74692443 0.78229564
3     Decision Tree 0.82161687 0.85612413
>
```

6. **Bar Chart for Accuracy Comparison**: Generates a sorted bar chart to compare model accuracy visually.

```r
# ---- 6. Final Bar Chart for Accuracy Comparison (Sorted from Lowest to Highest) ----
accuracy_df <- data.frame(
  Model = c("Linear Regression", "Neural Network", "Decision Tree"),
  Accuracy = c(accuracy_linear, accuracy_nn, accuracy_dt)
)

# Sort the data frame by Accuracy from lowest to highest
accuracy_df <- accuracy_df[order(accuracy_df$Accuracy), ]

# Re-level the Model factor to ensure correct order in the plot
accuracy_df$Model <- factor(accuracy_df$Model, levels = accuracy_df$Model)

# Plot the bar chart with sorted accuracy from lowest to highest
ggplot(data=accuracy_df, aes(x=Model, y=Accuracy, fill=Model)) +
  geom_bar(stat="identity", width=0.5) +
  ylim(0, 1) +
  xlab("Model") +
  ylab("Accuracy") +
  ggtitle("Accuracy Comparison of Models") +
  theme_minimal()
```



Findings and Managerial Implications

Key Findings

- Approval Trends: Programmer Analysts had the highest approval rates.
- Wages and Approval: Higher wage rates positively correlated with application success.
- Geographical Insights: Certain locations showed higher rejection rates, suggesting regional disparities.

Managerial Implications

1. For Employers:
    - Provide wage benchmarks to enhance approval chances.
    - Emphasize specific job titles with higher success rates.
2. For Applicants:
    - Recommend skill enhancements aligned with high-demand occupations.
    - Highlight critical factors affecting rejection.
3. For Policymakers:
    - Use predictive models to detect systemic biases.
    - Streamline application processes for greater transparency.

Recommendations

- Employer Tool Development: Assist organizations in optimizing applications with predictive feedback.
- Policy Adjustments: Advocate for measures addressing regional disparities and systemic biases.
- Future Research: Incorporate ensemble techniques like Random Forests to enhance predictive accuracy.

References

- H1-B Visa Dataset, U.S. Department of Labor.
- Lecture notes and materials from Prof. Zhe Zhang’s course.
- R documentation and statistical analysis resources.