DS 6306 Final Project 2

Oluwadamilola Owolabi

2024-04-22

#Helpful Links

Please try out my Rshinyapp in order to experience my model's prediction rate : https://oluwadamilolaowolabi.shinyapps.io/DS_6372_Project_2/ (https://oluwadamilolaowolabi.shinyapps.io/DS_6372_Project_2/)

Heres a Youtube Link to my presentation: https://youtu.be/hwyr3ZPVgIQ (https://youtu.be/hwyr3ZPVgIQ)

INTRODUCTION

Good afternoon Mr. Steven Williams – CEO and Mrs. Christy Jacoby - CFO. I am here today to present my findings on my analysis on the factors affecting the rate of attrition among employees within a company. I am an employee of DDSAnalytivs, and i am here to present how my analysis can help you retain talented employees within Frito Lays.

Our data set was extracted from the company's AWS bucket. Where we interviewed 870 random employees, question related to their possibility of attrition.We got 36 answers from each employee. 34 of those factors (answers) were used for our dependent variables. hle 2 of them were used for our predictions: The monthly income for our Regression prediction, and the Attrition for our Classification prediction.

My aim is to provide valuable insight on what factors affects attrition rate and Monthly Income

LOADING LIBRARIES

```
library(ggplot2) #For data visualization
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(dplyr)  # For data manipulation
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
##
## 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)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## Warning: package 'stringr' was built under R version 4.3.2
```

```
## ── 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     1.3.0
## ✓ readr     2.1.4
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
#install.packages('aws.s3')
library(aws.s3) #to access the aws s3 package
```

```
## Warning: package 'aws.s3' was built under R version 4.3.3
```

```
library(caret) # Load the caret package
```

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

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(pROC) # for the ROC curve
```

```
## Warning: package 'pROC' was built under R version 4.3.2
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(class) #calling the knn function
```

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

```
library(e1071)  # for naiveBayes function
```

```
## Warning: package 'e1071' was built under R version 4.3.2
```

GETTING THE CSV FILE FROM AWS USING AWS.S3 PACKAGES

```
Sys.setenv("AWS_ACCESS_KEY_ID" = "AKIASXWFQWBV2AEG7EEN",
           "AWS_SECRET_ACCESS_KEY" = "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",
           "AWS_DEFAULT_REGION" = "us-east-2")


# Using aws.s3
aws.s3::bucketlist()
```

```
##        Bucket             CreationDate
## 1 ds6306fls13 2024-04-03T00:47:17.000Z
```

```
aws.s3::get_bucket("msdsds6306")
```

```
## Bucket: msdsds6306
##
## $Contents
## Key:            Beers.csv
## LastModified:   2024-04-03T00:47:50.000Z
## ETag:           "3df3c62066b3102d3072f0435f5ecbff"
## Size (B):       136885
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            Case2PredictionsClassifyEXAMPLE.csv
## LastModified:   2023-11-25T15:25:00.000Z
## ETag:           "bd1de75effe9449f7d49a4de5116205a"
## Size (B):       3012
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            Case2PredictionsRegressEXAMPLE.csv
## LastModified:   2023-11-25T15:24:59.000Z
## ETag:           "a0f1f01c30e2cd00488822ad3c9aa6fe"
## Size (B):       3187
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            CaseStudy2-data.csv
## LastModified:   2023-11-25T15:25:01.000Z
## ETag:           "d68dd080517407fb3a4f05d91fed27d7"
## Size (B):       138428
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            CaseStudy2CompSet No Attrition.csv
## LastModified:   2023-11-25T15:25:03.000Z
## ETag:           "6c9d92b8a6fc5fd805ff0a5d4dfddde0"
## Size (B):       47686
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            CaseStudy2CompSet No Salary.csv
## LastModified:   2023-11-25T15:24:58.000Z
## ETag:           "30f4c83e1ebb33b19fe6b44c377a1fbd"
## Size (B):       46614
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            CaseStudy2CompSet No Salary.xlsx
```

```
## LastModified:   2023-11-25T15:25:02.000Z
## ETag:           "bdcb211847739638a631f828a3278339"
## Size (B):       56381
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            Creativity.csv
## LastModified:   2023-11-15T00:20:38.000Z
## ETag:           "08bdc7b11b2b9ed721e2d502aa4da1ec"
## Size (B):       752
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            iris.csv
## LastModified:   2023-11-15T00:11:09.000Z
## ETag:           "885a41d063a7923ae8d0e8404d9291a3"
## Size (B):       4821
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
##
## $Contents
## Key:            iris.json
## LastModified:   2023-11-15T00:16:53.000Z
## ETag:           "c2b11954af90bd715cfc71a757f27eb9"
## Size (B):       14460
## Owner:          15a554773c0d84536ce02c70d7513a1db00372e75f2bcebece99b2bd3cb7105b
## Storage class:  STANDARD
```

```
# read and write from ojbect

#Read in the train data
Talent_Train = s3read_using(FUN = read.csv,
                    bucket = "msdsds6306",
                    object = "CaseStudy2-data.csv")

#Reading in  the Test Data for Attrition
Talent_Test_Attrition = s3read_using(FUN = read.csv,
                    bucket = "msdsds6306",
                    object = "CaseStudy2CompSet No Attrition.csv")

#Reading in  the Test Data for monthly income
Talent_Test_Salary = s3read_using(FUN = read.csv,
                    bucket = "msdsds6306",
                    object = "CaseStudy2CompSet No Salary.csv")

#Reading in  an example of prediction Regeression
Example = s3read_using(FUN = read.csv,
                    bucket = "msdsds6306",
                    object = "Case2PredictionsRegressEXAMPLE.csv")

#Reading in  an example of prediction classification
Example2 = s3read_using(FUN = read.csv,
                    bucket = "msdsds6306",
                    object = "Case2PredictionsClassifyEXAMPLE.csv")


#Talent_Train; Talent_Test_Attrition; Talent_Test_Salary

names(Talent_Train) # looking at the variables in the dataset
```

```
##  [1] "ID"                       "Age"
##  [3] "Attrition"                "BusinessTravel"
##  [5] "DailyRate"                "Department"
##  [7] "DistanceFromHome"         "Education"
##  [9] "EducationField"           "EmployeeCount"
## [11] "EmployeeNumber"           "EnvironmentSatisfaction"
## [13] "Gender"                   "HourlyRate"
## [15] "JobInvolvement"           "JobLevel"
## [17] "JobRole"                  "JobSatisfaction"
## [19] "MaritalStatus"            "MonthlyIncome"
## [21] "MonthlyRate"              "NumCompaniesWorked"
## [23] "Over18"                   "OverTime"
## [25] "PercentSalaryHike"        "PerformanceRating"
## [27] "RelationshipSatisfaction" "StandardHours"
## [29] "StockOptionLevel"         "TotalWorkingYears"
## [31] "TrainingTimesLastYear"    "WorkLifeBalance"
## [33] "YearsAtCompany"           "YearsInCurrentRole"
## [35] "YearsSinceLastPromotion"  "YearsWithCurrManager"
```

```
#How many variables in the dataset ? 36

#Our response variable will be based on the attrition variable

head(Example) #Looking at our example regression
```

```
##    ID MonthlyIncome
## 1 871          1000
## 2 872          1000
## 3 873          1000
## 4 874          1000
## 5 875          1000
## 6 876          1000
```

```
head(Example2) #Looking at our example regression
```

```
##     ID Attrition
## 1 1171       Yes
## 2 1172       Yes
## 3 1173       Yes
## 4 1174       Yes
## 5 1175       Yes
## 6 1176       Yes
```

From the Train talent data, there are 870 random employees (rows), all ordered by IDD, and 36 variables (columns).

LOOKING AT THE DATASET

```
set.seed(1234)

# Set levels for Talent_Train
Talent_Train$Attrition <-factor(Talent_Train$Attrition, levels=c('Yes','No')) #factoring the res
ponse variable
Talent_Train <- Talent_Train %>% mutate(Over18_binary = ifelse(Over18 == "Y", 1, 0)) # breaking
the Over18 variable into 2, because it has only one level, and keeps throwing an error.

Talent_Train <- subset(Talent_Train, select = -c(Over18)) #removing the over18 column

# Set levels for Talent_Test_Salary
Talent_Test_Salary$Attrition <-factor(Talent_Test_Salary$Attrition, levels=c('No','Yes')) #facto
ring the response variable
Talent_Test_Salary <- Talent_Test_Salary %>% mutate(Over18_binary = ifelse(Over18 == "Y", 1, 0))
# breaking the Over18 variable into 2, because it has only one level.
Talent_Test_Salary <- subset(Talent_Test_Salary , select = -c(Over18)) #removing the over18 colu
mn

yes_data <- Talent_Train[Talent_Train$Attrition == 'Yes',] # 4250
no_data <- Talent_Train[Talent_Train$Attrition == 'No',] # 31918
```

The rate of Yes Attrition to no Attrition is 140: 730. Which presents the issue of an unbalanced data. This might be costly to sensitivity metric. Thankfully, i come equipped with knowledge from my Data Science professor (Prof. Bivin Sadler) on how to resolve this.

ADDRESSING MISSING VALUES

It is important to look at missing values earlier on, as it might be affect our models

```
NaSum <- sum(is.na(Talent_Train)) #check for missing values in the Talent Dataset

# Print the total count of missing values
print(paste("Total missing values:", NaSum))
```

```
## [1] "Total missing values: 0"
```

From the results above, there appears to be no missing values in the Dataset (thank God!!)

SALARY

Looking at the Summary Statistics

```
# Using a for loop to get the summary statistics

# Initialize an empty list
numerical_count <- 0 # There are 27
categorical_count <- 0 # There are 9
summary_stats_numerical <- list() # Storing summary statistics for the numerical variables
summary_stats_categorical <- list() # Storing summary statistics for the categorical variables
summary_stats2_categorical <- list() # Storing summary statistics for the categorical variables
categorical_variables <- list() #Storing categorical variables
numerical_variables <- list() #Storing numerical variables

# Iterate over each column in the data frame
for (Variable in names(Talent_Train)) {
  # Calculate summary statistics for numeric variables
  if (is.numeric(Talent_Train[[Variable]])) {
    summary_stats_numerical[[Variable]] <- summary(Talent_Train[[Variable]])
    numerical_variables[[Variable]] <- Variable
    numerical_count <- numerical_count + 1
  } else {
    # For non-numeric variables, calculate frequency table
    summary_stats_categorical[[Variable]] <- table(Talent_Train[[Variable]])
    summary_stats2_categorical[[Variable]] <- summary(Talent_Train[[Variable]])
    categorical_variables[[Variable]] <- Variable
    categorical_count <- categorical_count + 1
  }
}

cat("There are  ", numerical_count, " numerical variables \n")
```

```
## There are   28  numerical variables
```

```
cat("The numerical variables are: ")
```

```
## The numerical variables are:
```

```
#Printing the numerical variables
for (variable in names(numerical_variables)){
  print(numerical_variables[[variable]])
}
```

```
## [1] "ID"
## [1] "Age"
## [1] "DailyRate"
## [1] "DistanceFromHome"
## [1] "Education"
## [1] "EmployeeCount"
## [1] "EmployeeNumber"
## [1] "EnvironmentSatisfaction"
## [1] "HourlyRate"
## [1] "JobInvolvement"
## [1] "JobLevel"
## [1] "JobSatisfaction"
## [1] "MonthlyIncome"
## [1] "MonthlyRate"
## [1] "NumCompaniesWorked"
## [1] "PercentSalaryHike"
## [1] "PerformanceRating"
## [1] "RelationshipSatisfaction"
## [1] "StandardHours"
## [1] "StockOptionLevel"
## [1] "TotalWorkingYears"
## [1] "TrainingTimesLastYear"
## [1] "WorkLifeBalance"
## [1] "YearsAtCompany"
## [1] "YearsInCurrentRole"
## [1] "YearsSinceLastPromotion"
## [1] "YearsWithCurrManager"
## [1] "Over18_binary"
```

```
cat("\n")
```

```
cat("There are  ", categorical_count, " categorical variables \n")
```

```
## There are   8  categorical variables
```

```
cat("The categorical variables are: ")
```

```
## The categorical variables are:
```

```
#Printing the categorical variables
for (variable in names(categorical_variables)){
  print(categorical_variables[[variable]])
}
```

```
## [1] "Attrition"
## [1] "BusinessTravel"
## [1] "Department"
## [1] "EducationField"
## [1] "Gender"
## [1] "JobRole"
## [1] "MaritalStatus"
## [1] "OverTime"
```

```
cat("\n")
```

```
# Print summary statistics for each numerical variable
for (col_name in names(summary_stats_numerical)) {
  cat("Summary statistics for", col_name, ":\n")
  print(summary_stats_numerical[[col_name]])
  cat("\n")
}
```

```
## Summary statistics for ID :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   218.2   435.5   435.5   652.8   870.0
##
## Summary statistics for Age :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   30.00   35.00   36.83   43.00   60.00
##
## Summary statistics for DailyRate :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   103.0   472.5   817.5   815.2  1165.8  1499.0
##
## Summary statistics for DistanceFromHome :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   7.000   9.339  14.000  29.000
##
## Summary statistics for Education :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.901   4.000   5.000
##
## Summary statistics for EmployeeCount :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
##
## Summary statistics for EmployeeNumber :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   477.2  1039.0  1029.8  1561.5  2064.0
##
## Summary statistics for EnvironmentSatisfaction :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.701   4.000   4.000
##
## Summary statistics for HourlyRate :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   30.00   48.00   66.00   65.61   83.00  100.00
##
## Summary statistics for JobInvolvement :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.723   3.000   4.000
##
## Summary statistics for JobLevel :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   2.000   2.039   3.000   5.000
##
## Summary statistics for JobSatisfaction :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.709   4.000   4.000
##
## Summary statistics for MonthlyIncome :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1081    2840    4946    6390    8182   19999
##
```

```
## Summary statistics for MonthlyRate :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2094    8092   14074   14326   20456   26997
##
## Summary statistics for NumCompaniesWorked :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   2.000   2.728   4.000   9.000
##
## Summary statistics for PercentSalaryHike :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    11.0    12.0    14.0    15.2    18.0    25.0
##
## Summary statistics for PerformanceRating :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   3.000   3.152   3.000   4.000
##
## Summary statistics for RelationshipSatisfaction :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.707   4.000   4.000
##
## Summary statistics for StandardHours :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      80      80      80      80      80      80
##
## Summary statistics for StockOptionLevel :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.7839  1.0000  3.0000
##
## Summary statistics for TotalWorkingYears :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    6.00   10.00   11.05   15.00   40.00
##
## Summary statistics for TrainingTimesLastYear :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   3.000   2.832   3.000   6.000
##
## Summary statistics for WorkLifeBalance :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.782   3.000   4.000
##
## Summary statistics for YearsAtCompany :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   5.000   6.962  10.000  40.000
##
## Summary statistics for YearsInCurrentRole :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   3.000   4.205   7.000  18.000
##
## Summary statistics for YearsSinceLastPromotion :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   1.000   2.169   3.000  15.000
##
```

```
## Summary statistics for YearsWithCurrManager :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    2.00    3.00    4.14    7.00   17.00
##
## Summary statistics for Over18_binary :
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

```
# Print summary statistics for each categorical variable
for (col_name in names(summary_stats_categorical)) {
  cat("Summary statistics for", col_name, ":\n")
  print(summary_stats_categorical[[col_name]])
  #print(summary_stats2_categorical[[col_name]])
  cat("\n")
}
```

```
## Summary statistics for Attrition :
##
## Yes  No
## 140 730
##
## Summary statistics for BusinessTravel :
##
##        Non-Travel Travel_Frequently     Travel_Rarely
##                94               158               618
##
## Summary statistics for Department :
##
##        Human Resources Research & Development                  Sales
##                     35                    562                    273
##
## Summary statistics for EducationField :
##
##  Human Resources    Life Sciences        Marketing          Medical
##               15              358              100              270
##            Other Technical Degree
##               52               75
##
## Summary statistics for Gender :
##
## Female   Male
##    354    516
##
## Summary statistics for JobRole :
##
## Healthcare Representative           Human Resources     Laboratory Technician
##                        76                        27                       153
##                   Manager    Manufacturing Director         Research Director
##                        51                        87                        51
##        Research Scientist           Sales Executive      Sales Representative
##                       172                       200                        53
##
## Summary statistics for MaritalStatus :
##
## Divorced  Married   Single
##      191      410      269
##
## Summary statistics for OverTime :
##
##  No Yes
## 618 252
```

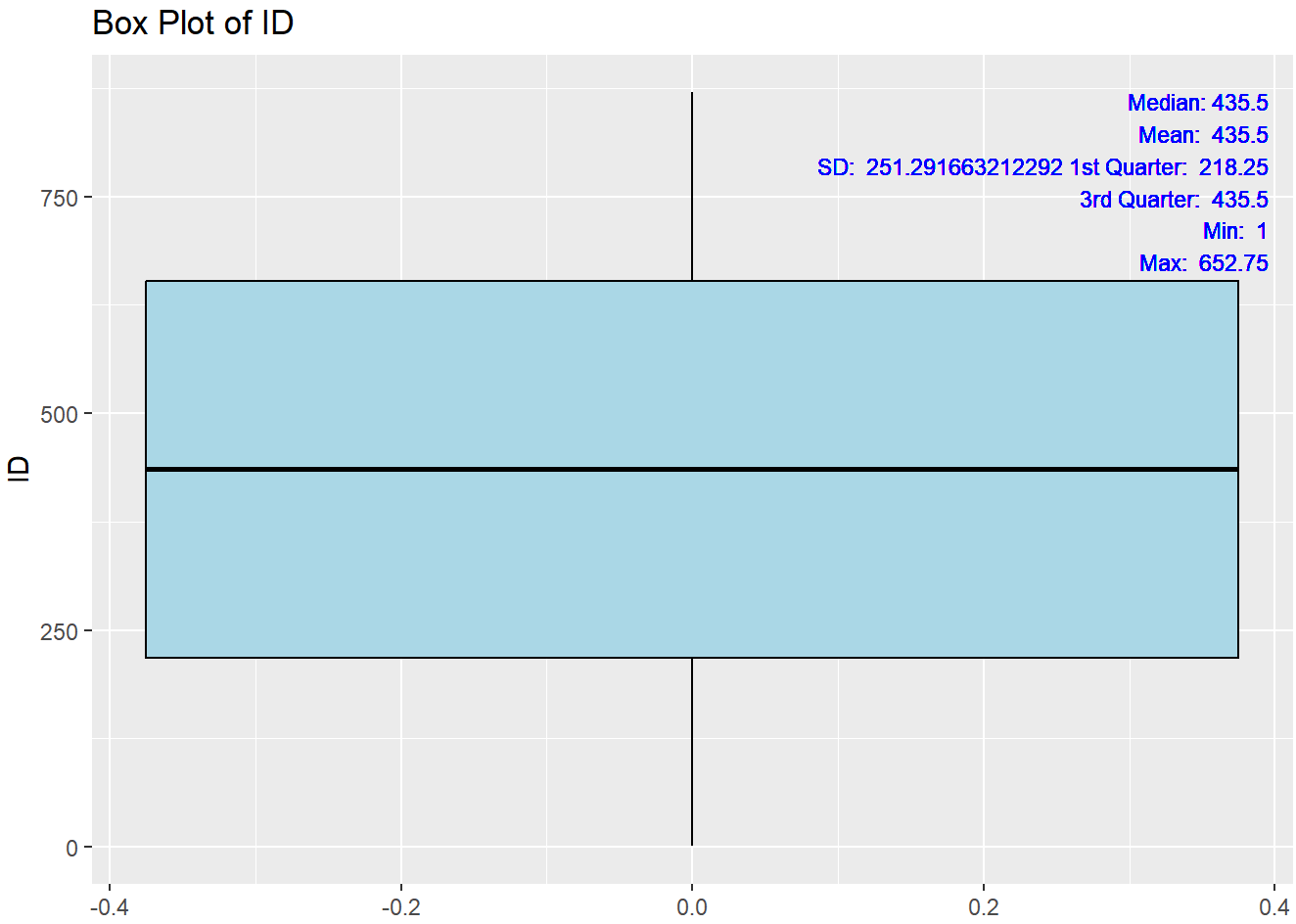
From our summary Statistics, there are 28 numerical variables and 8 categorical variables. All adding to 36 variables. Some variables that swtood out were the EmployeeCount, which have a constant 1 values, and the Job Involvement, which values ranges between 1 and 4, hence providing a really low variance,.

VISUALIZING THE NUMERICAL VARIABLE'S SUMMARY STATISTICS

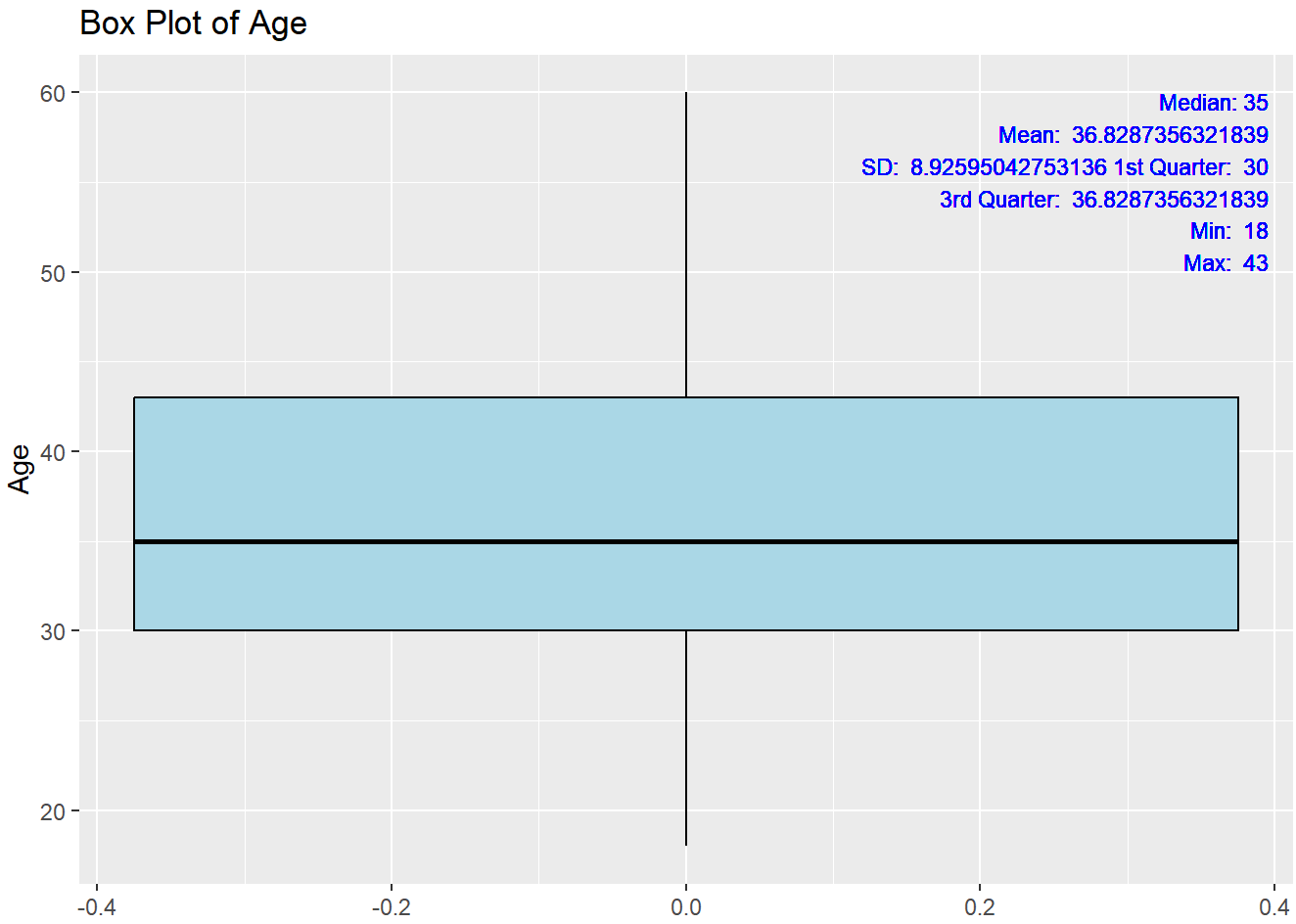
```
# Since the variables are a lot, and i plan on saving time, I plan using a for loop to iterate t
hrough the variables and plot their box plots to visualize their summary statistics



for (Variable in names(Talent_Train)){
  # Check if the column is numeric
  if (is.numeric(Talent_Train[[Variable]])) {
    # Create a box plot for numeric variables
   plot <-  ggplot(Talent_Train, aes_string(x = , y = Variable)) +
      geom_boxplot(color = "black", fill = "lightblue") +
      labs(title = paste("Box Plot of", Variable)) +
      geom_text(aes(label = paste("Median:", median(Talent_Train[[Variable]]), "\n",
                                  "Mean: ", mean(Talent_Train[[Variable]]), "\n",
                                  "SD: ", sd(Talent_Train[[Variable]]),
                                  "1st Quarter: ", summary(Talent_Train[[Variable]])[2], "\n",
                                  "3rd Quarter: ", summary(Talent_Train[[Variable]])[4], "\n",
                                  "Min: ", summary(Talent_Train[[Variable]])[1], "\n",
                                  "Max: ", summary(Talent_Train[[Variable]])[5], "\n")),
                x = 0.4, y = max(Talent_Train[[Variable]]), hjust = 1, vjust = 1, size = 3, colo
r = "blue")

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

```
## 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.
```

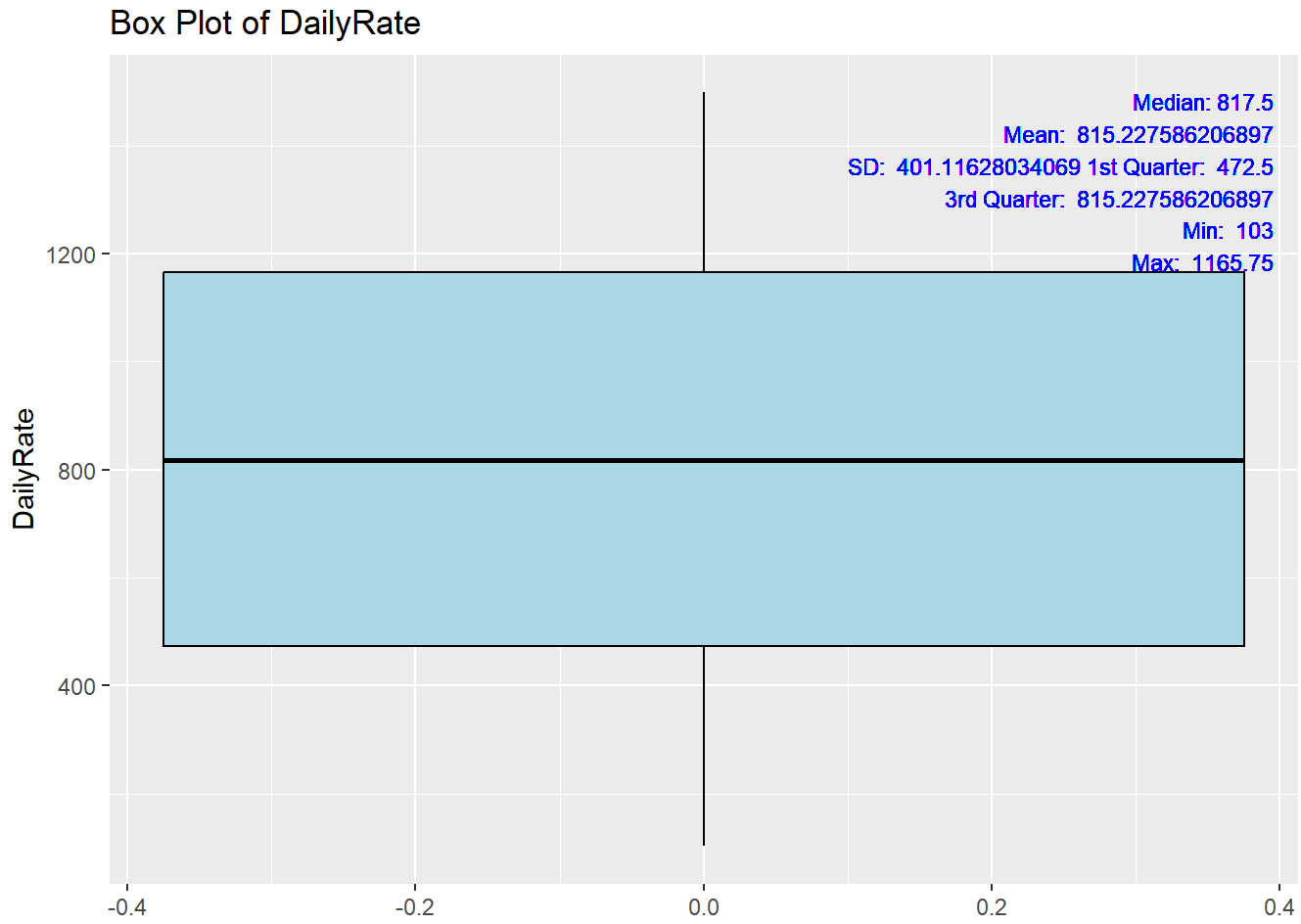
```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## i Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## i Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## i Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## i Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## i Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## i Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## i Use `.data[[Variable]]` instead.
```



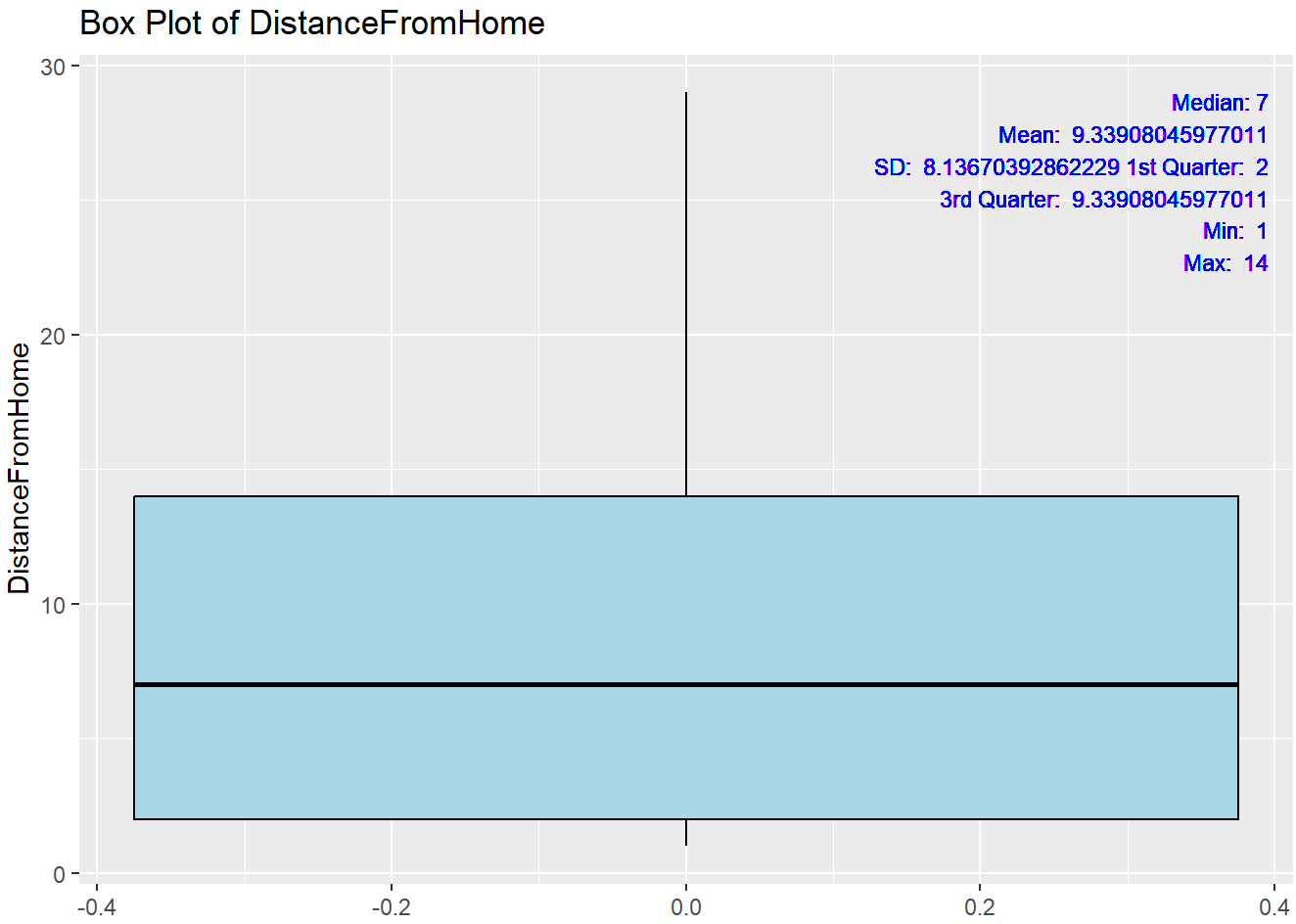

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



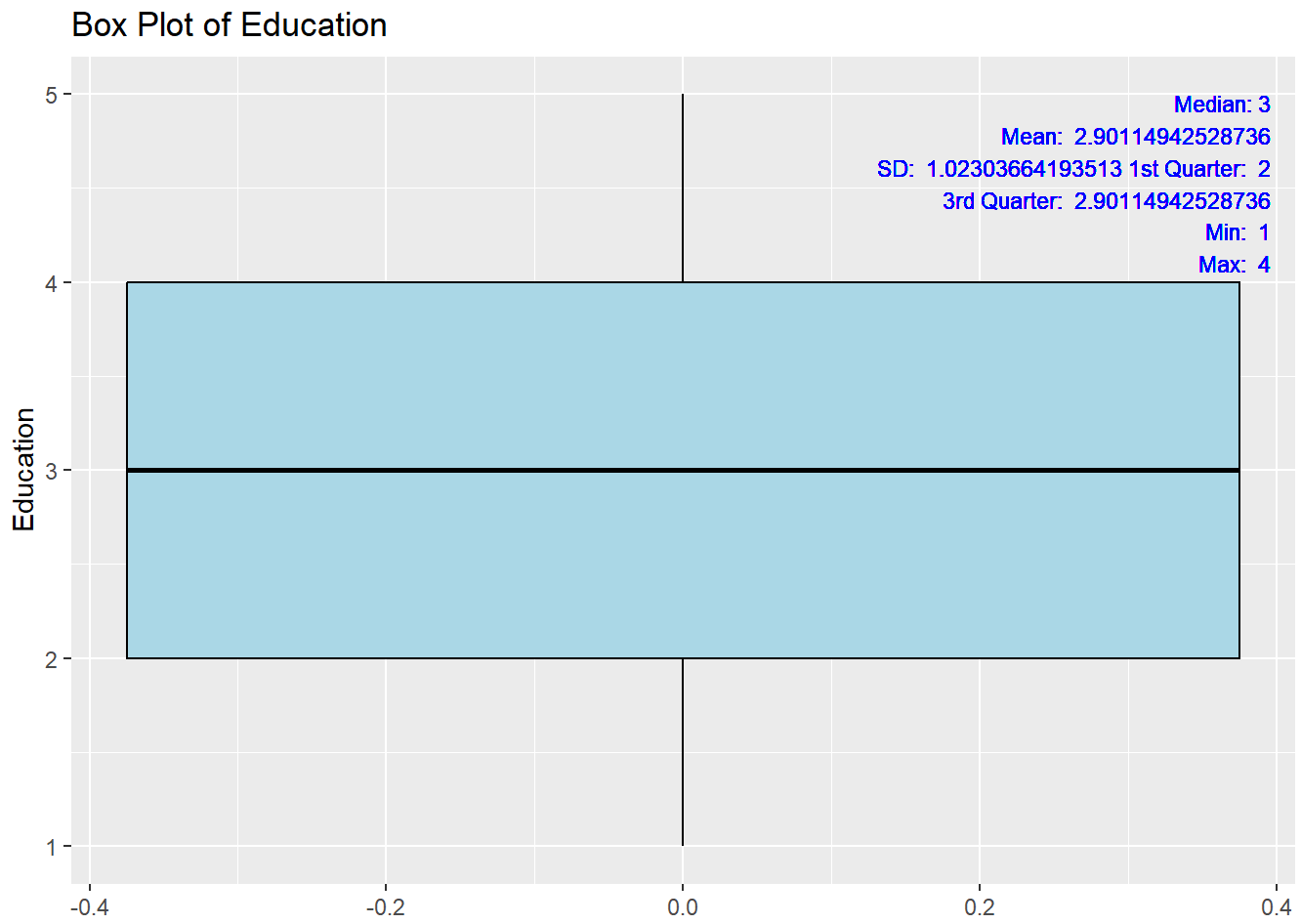

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



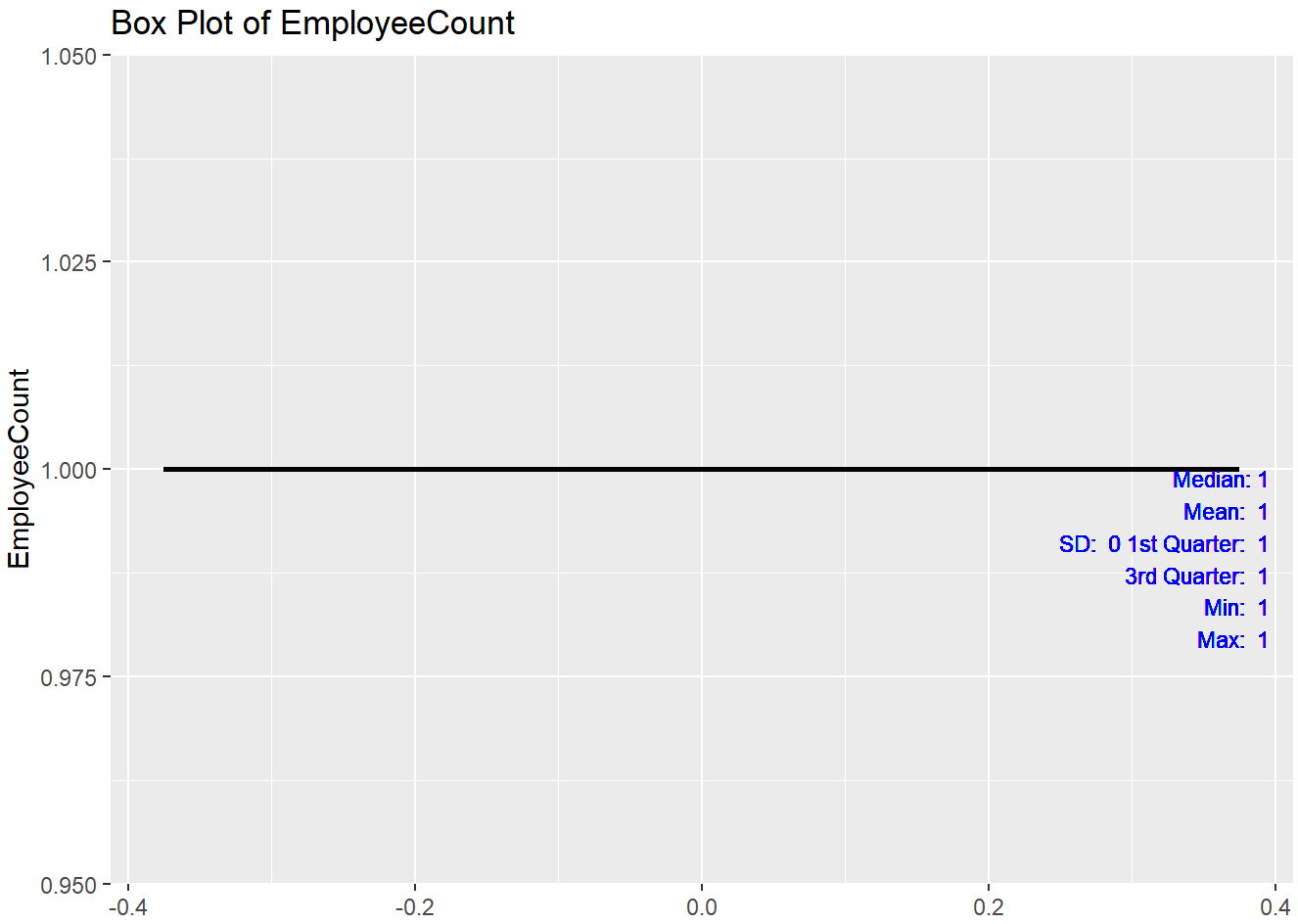

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



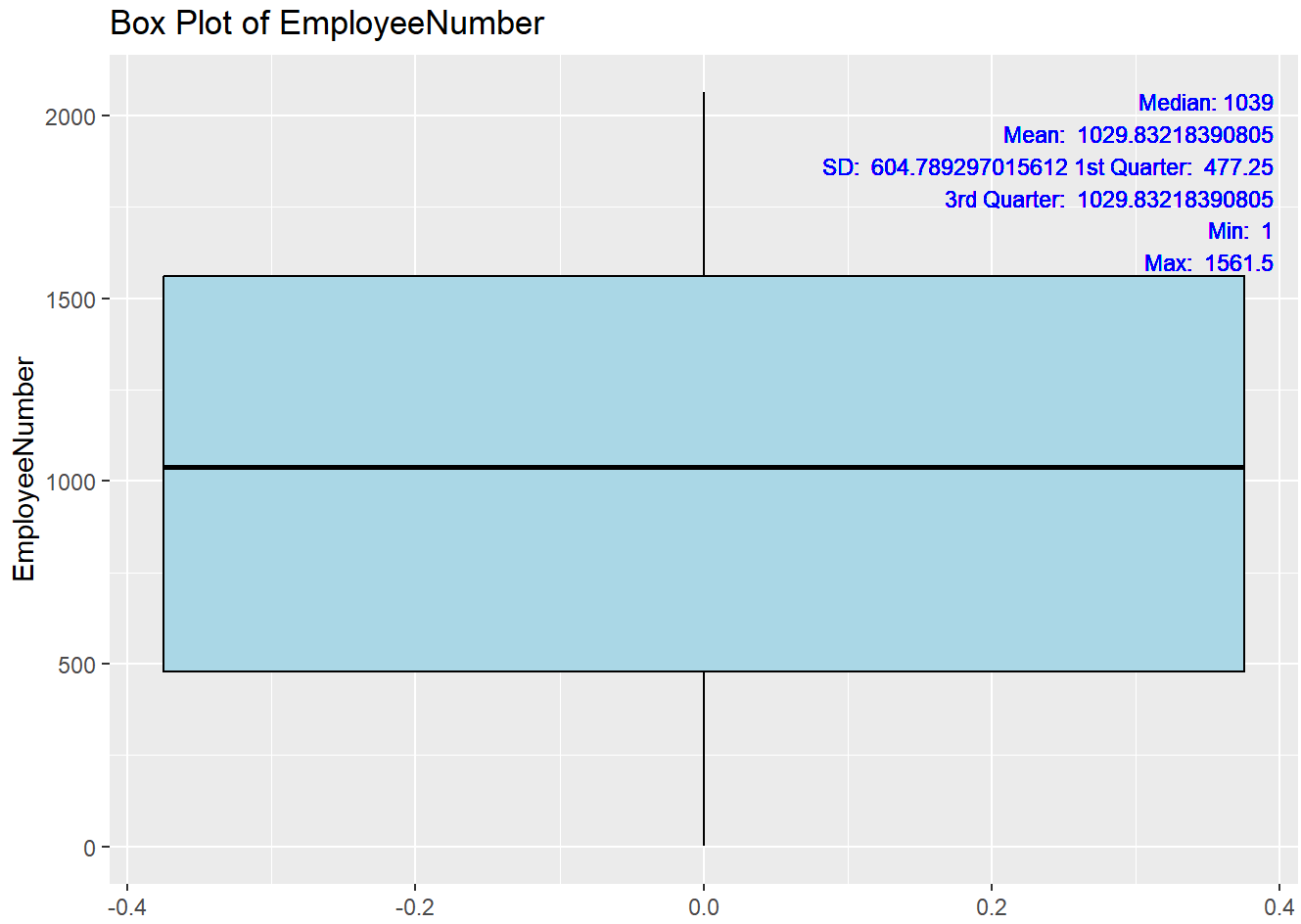

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



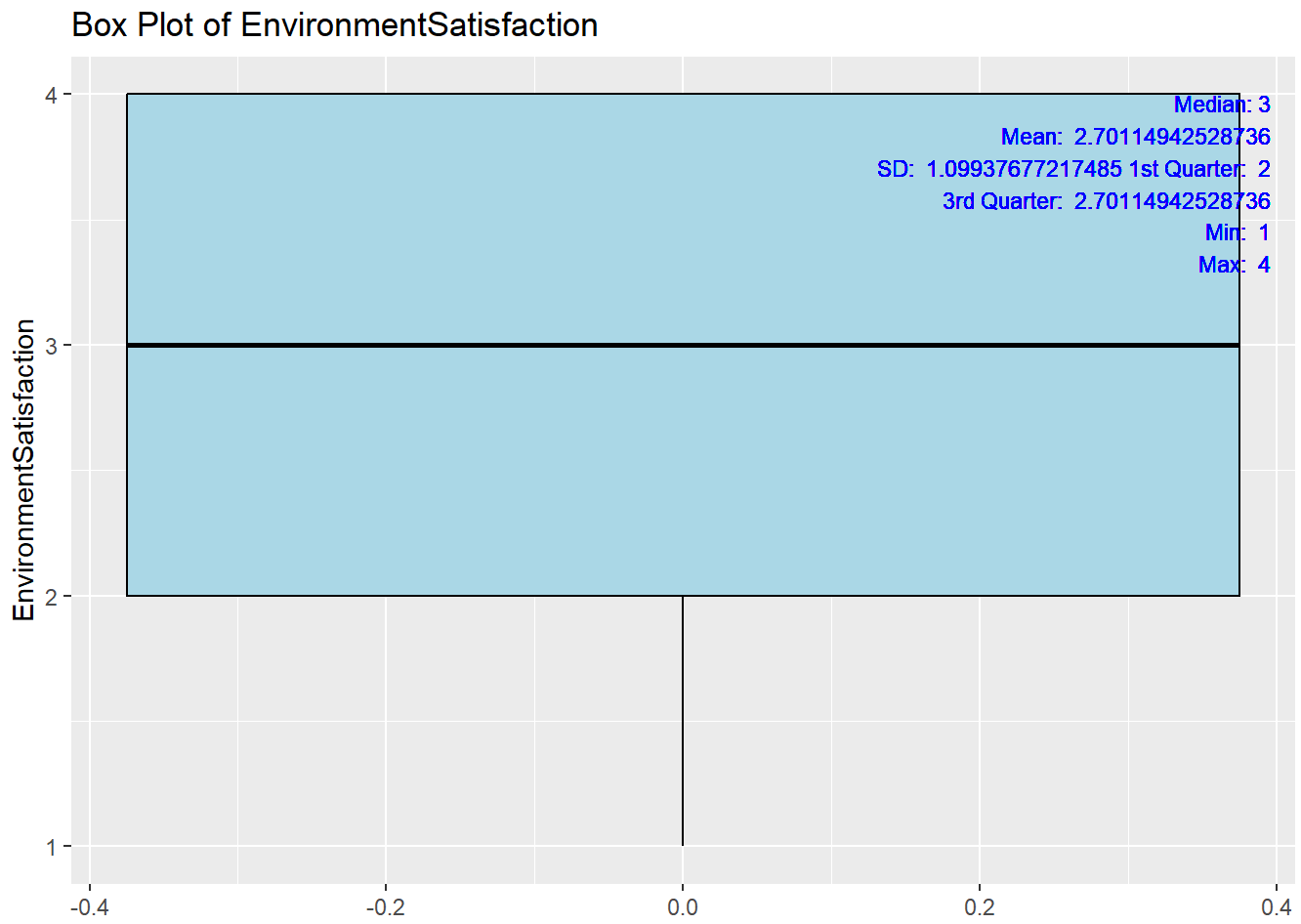

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



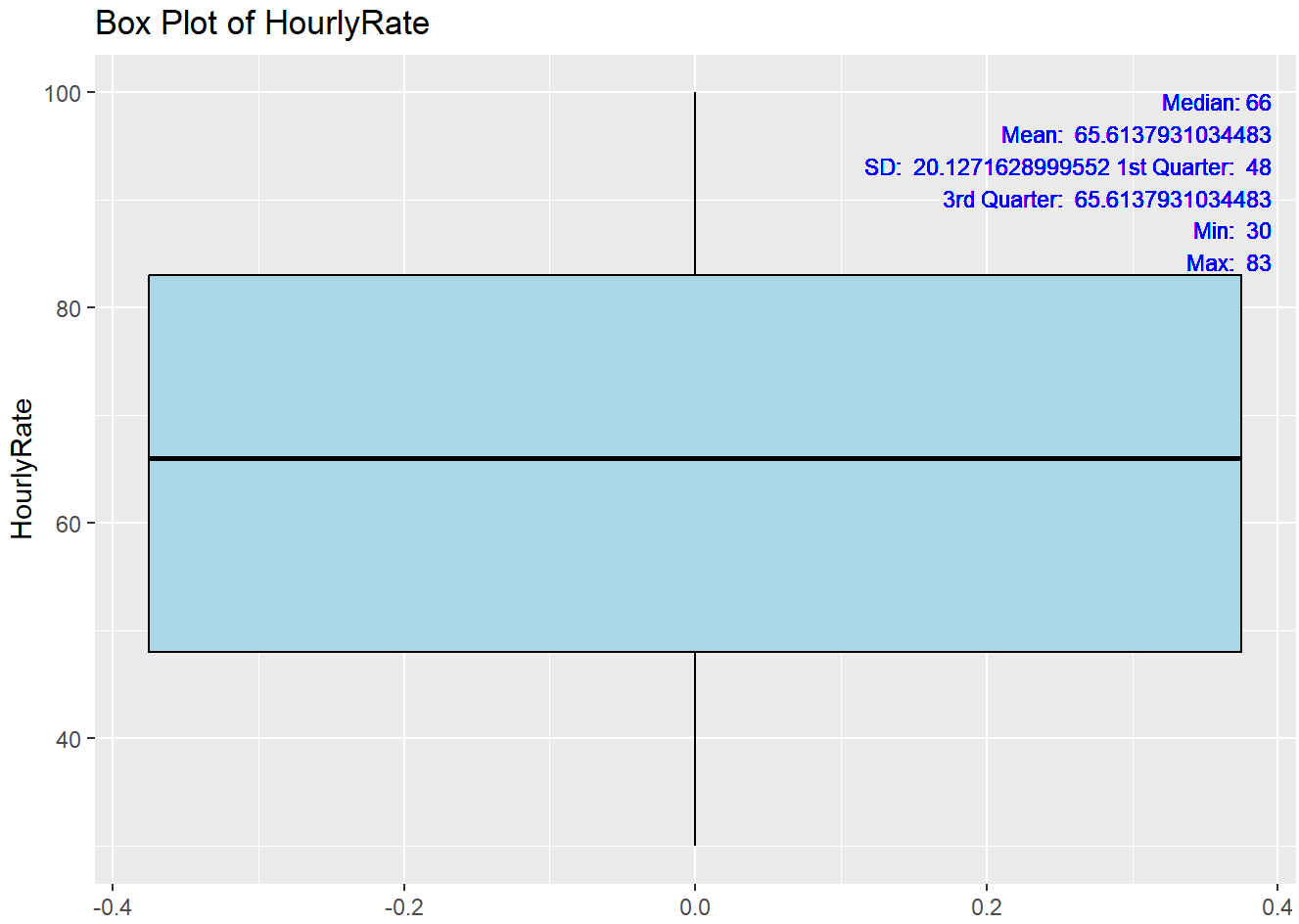

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



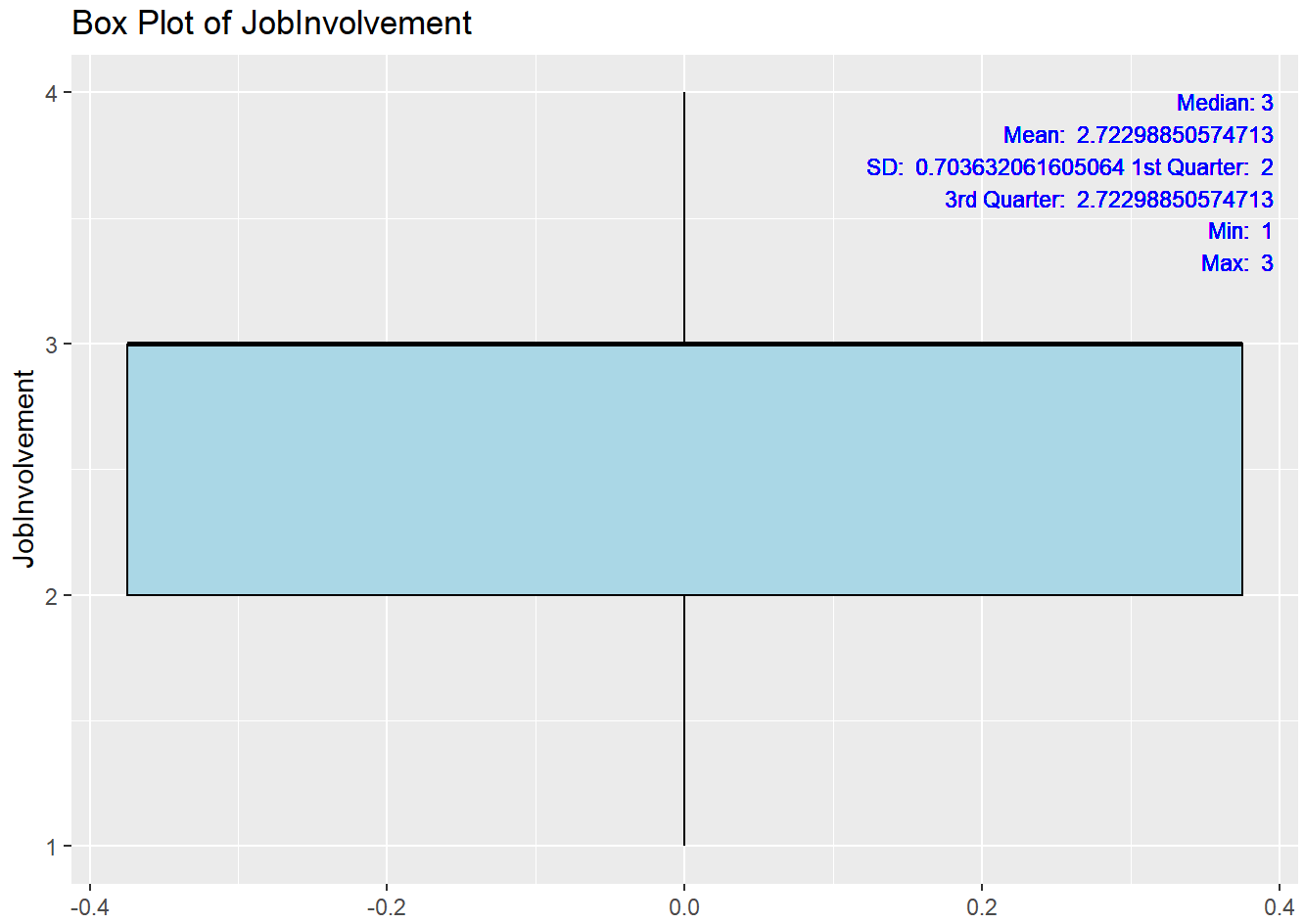

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



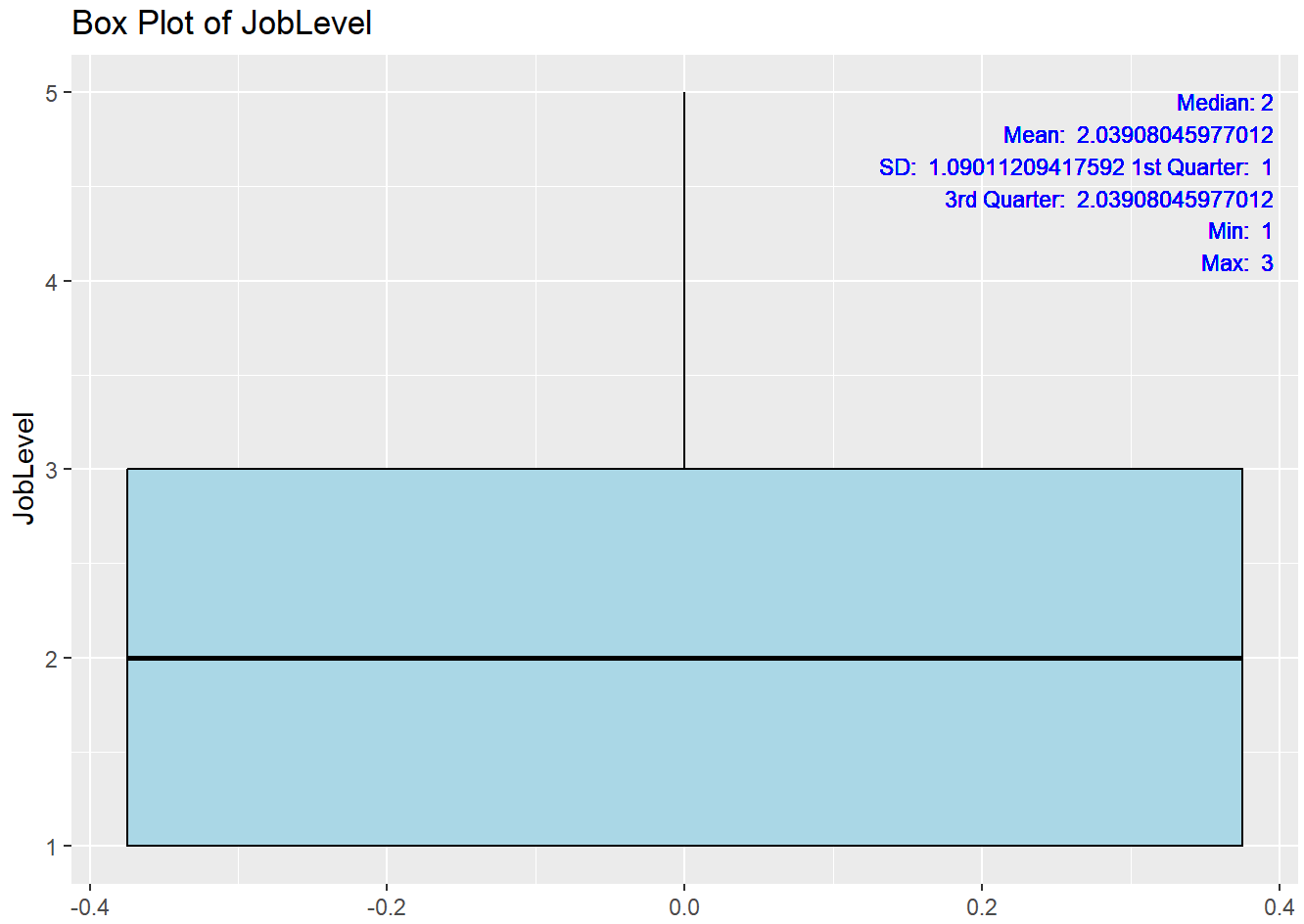

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



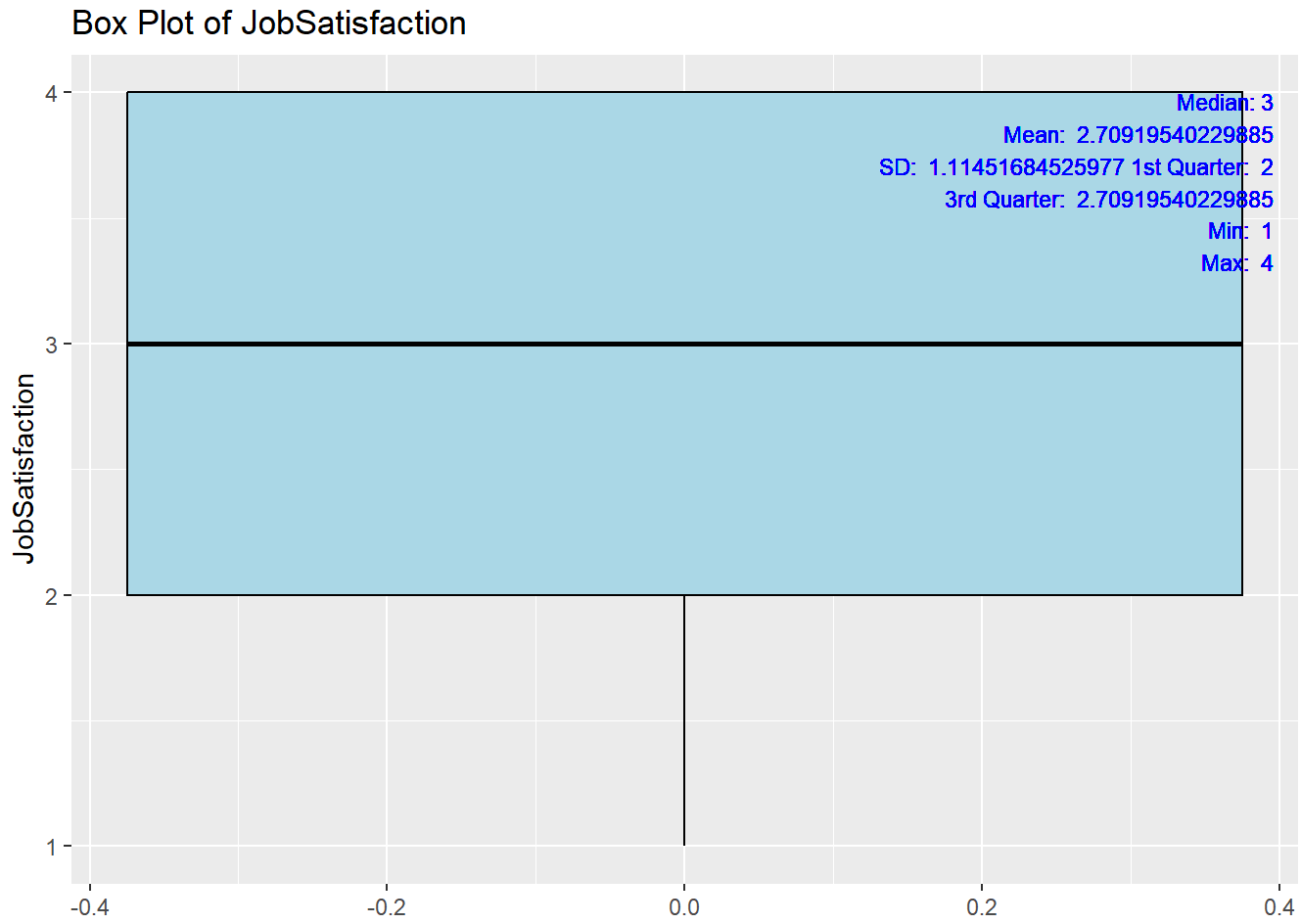

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



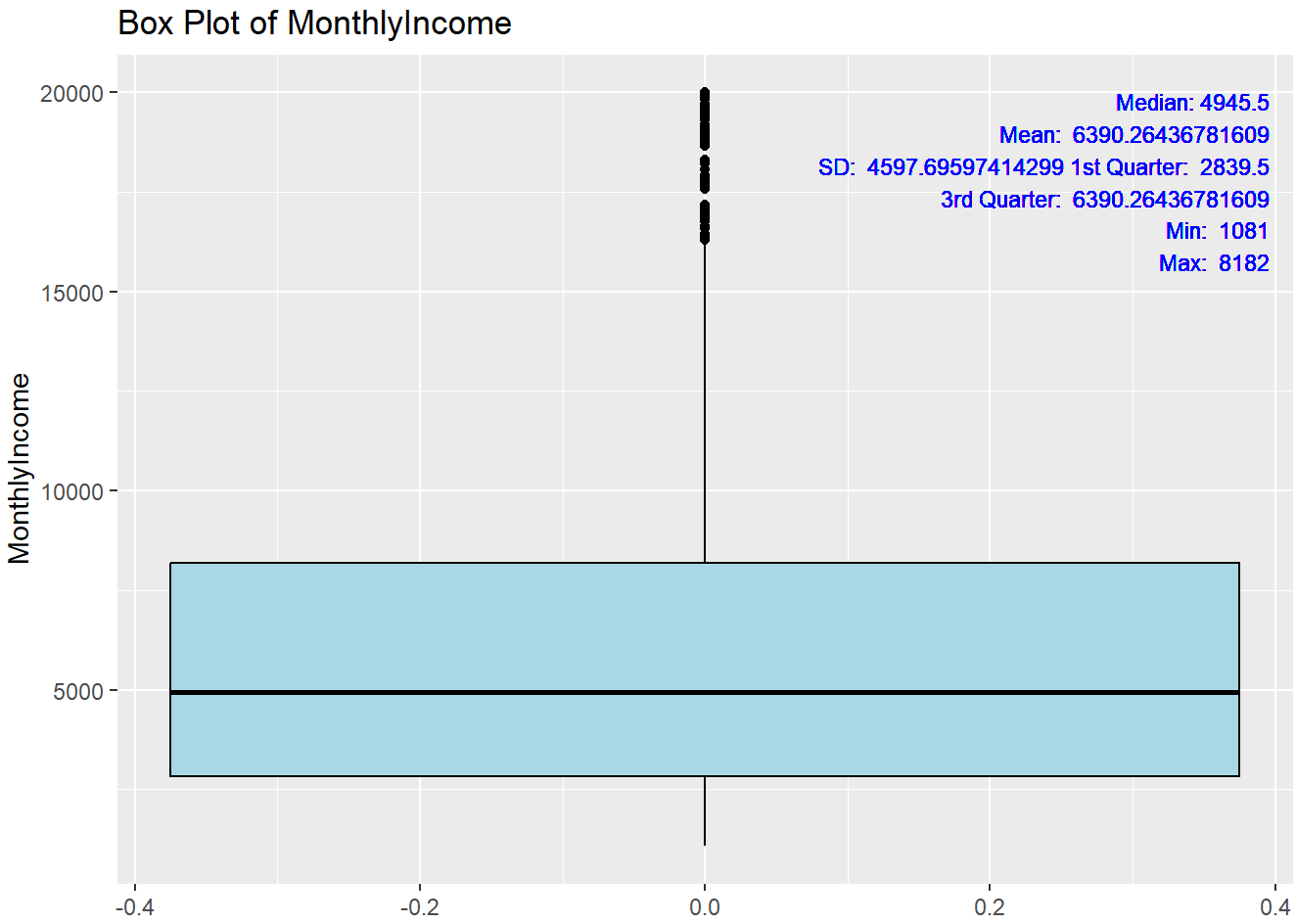

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



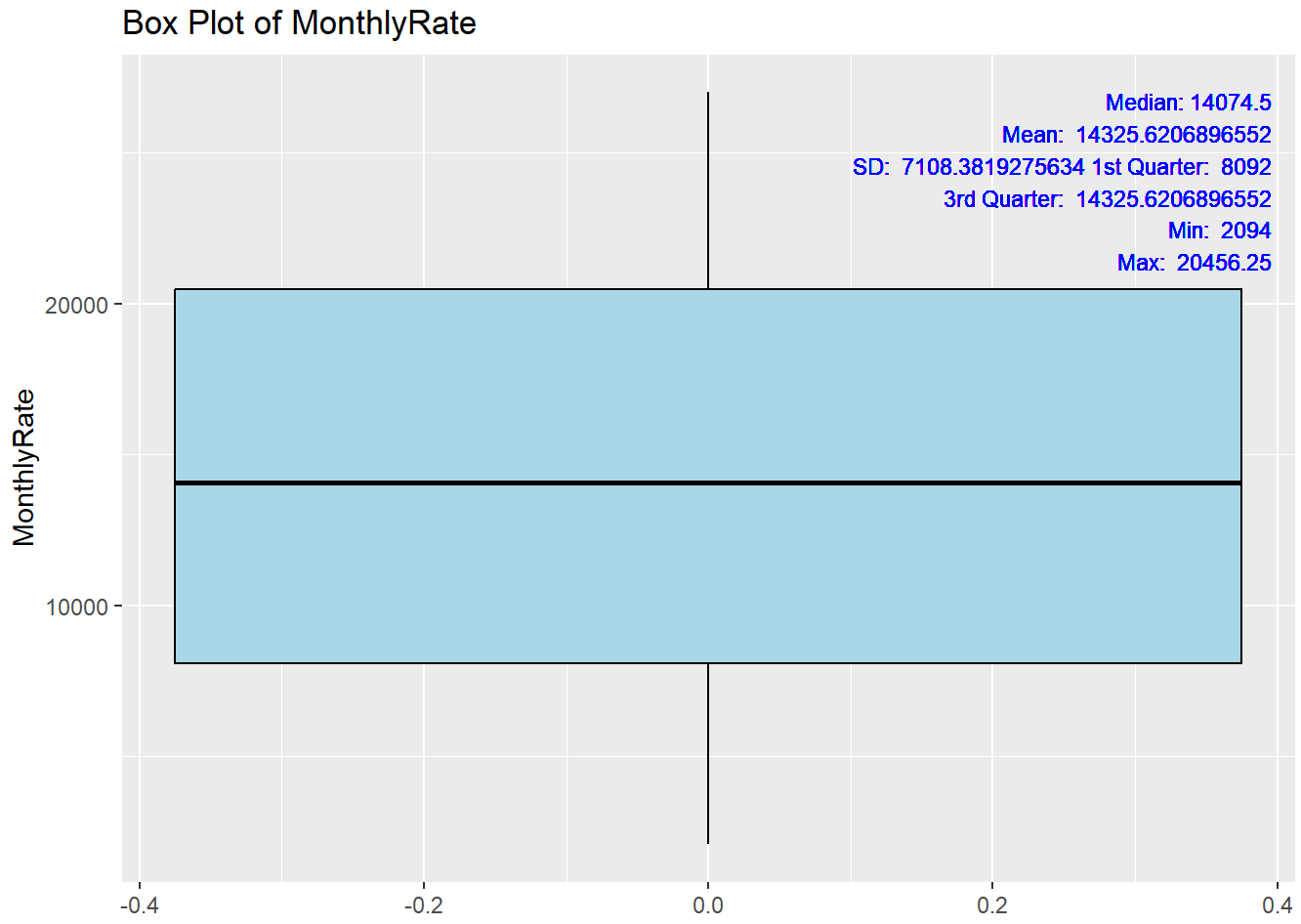

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



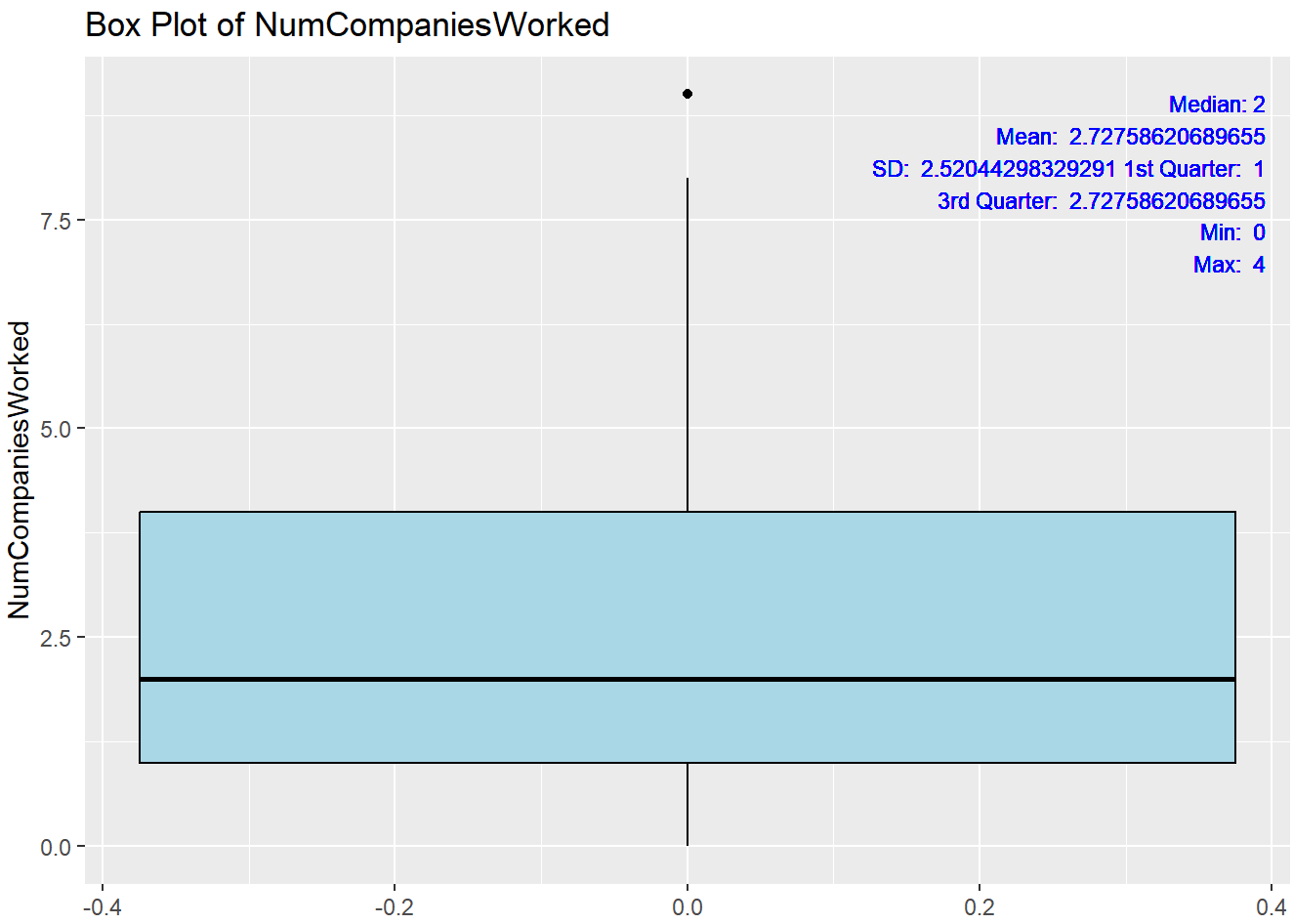

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



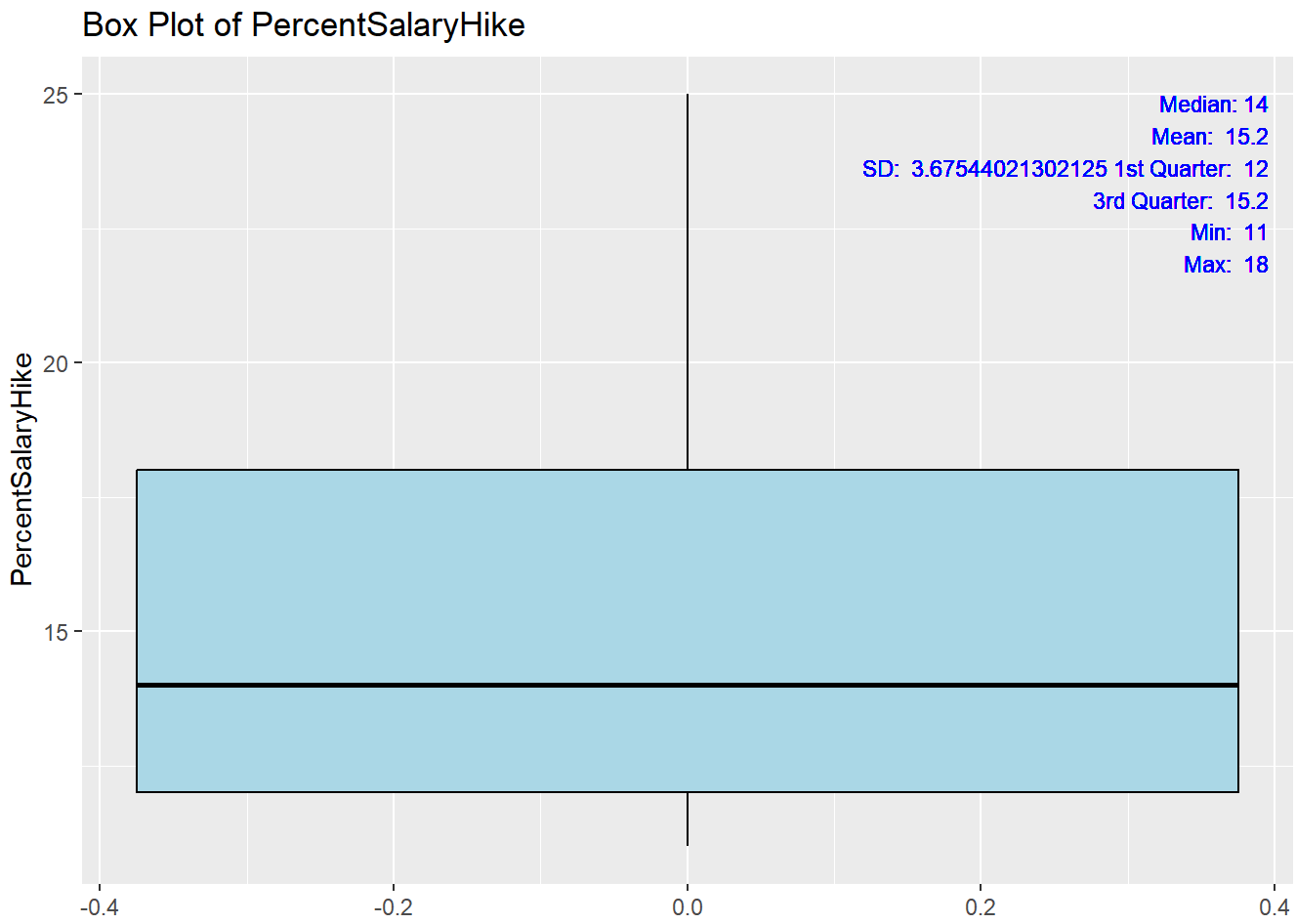

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



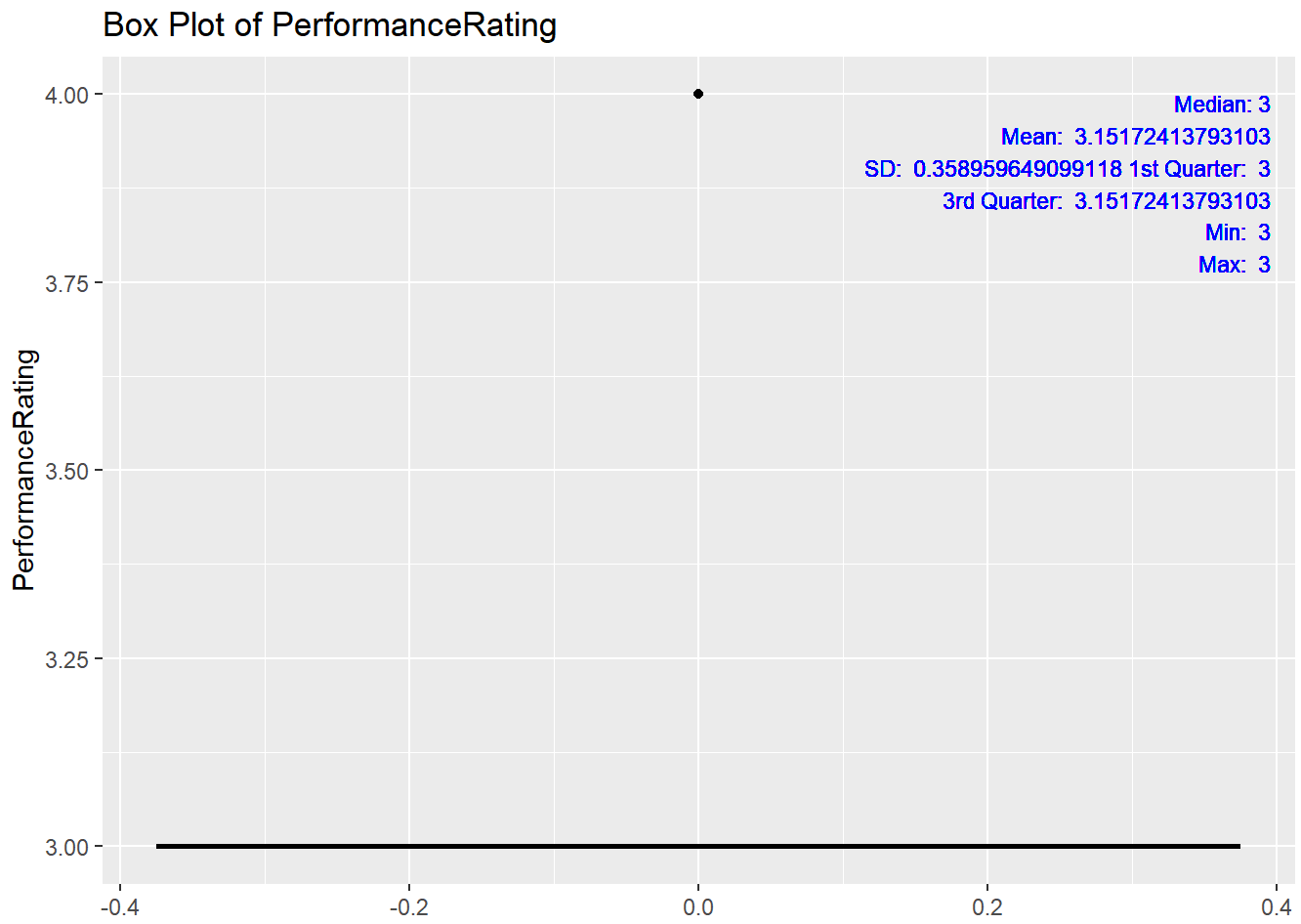

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



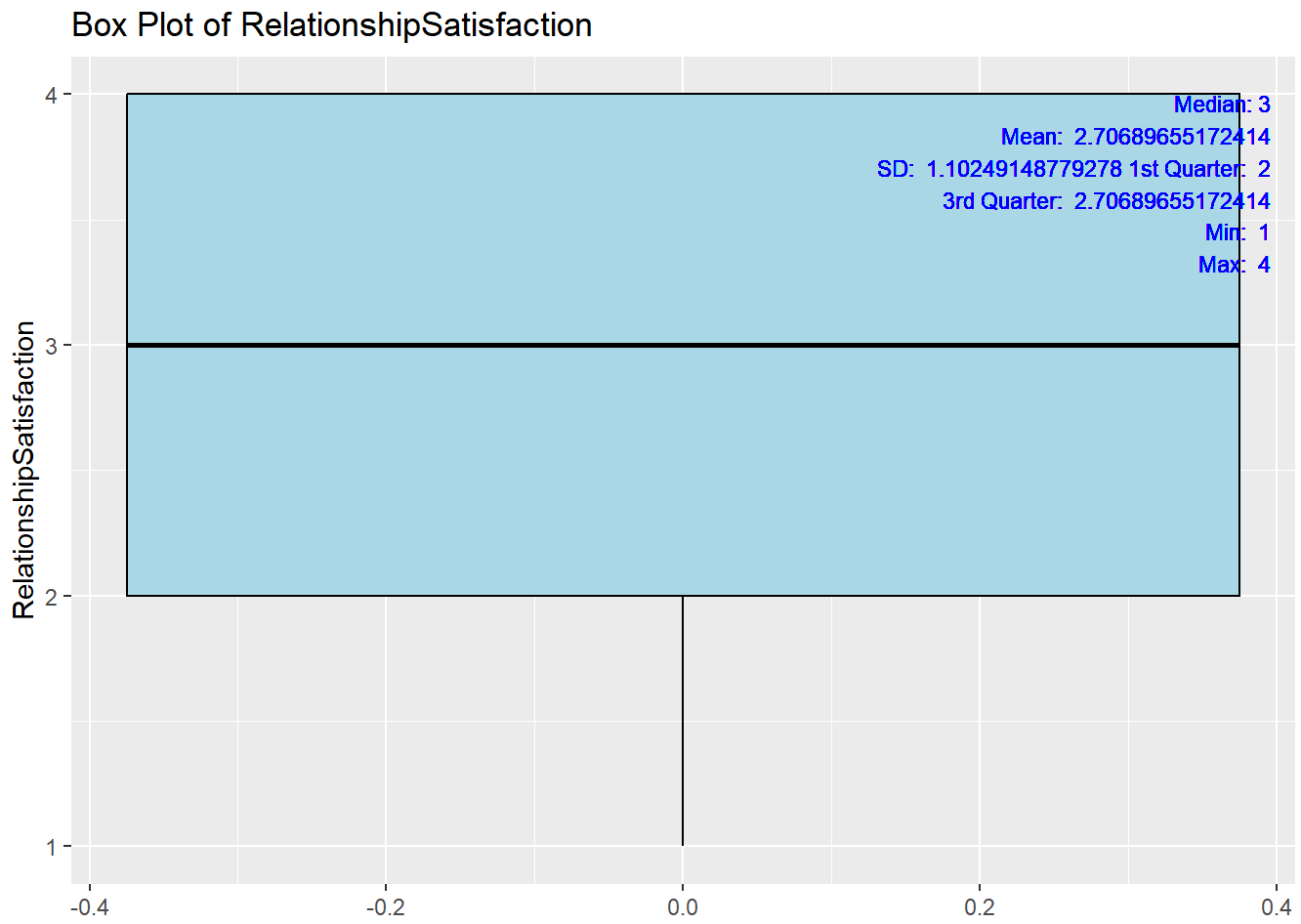

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



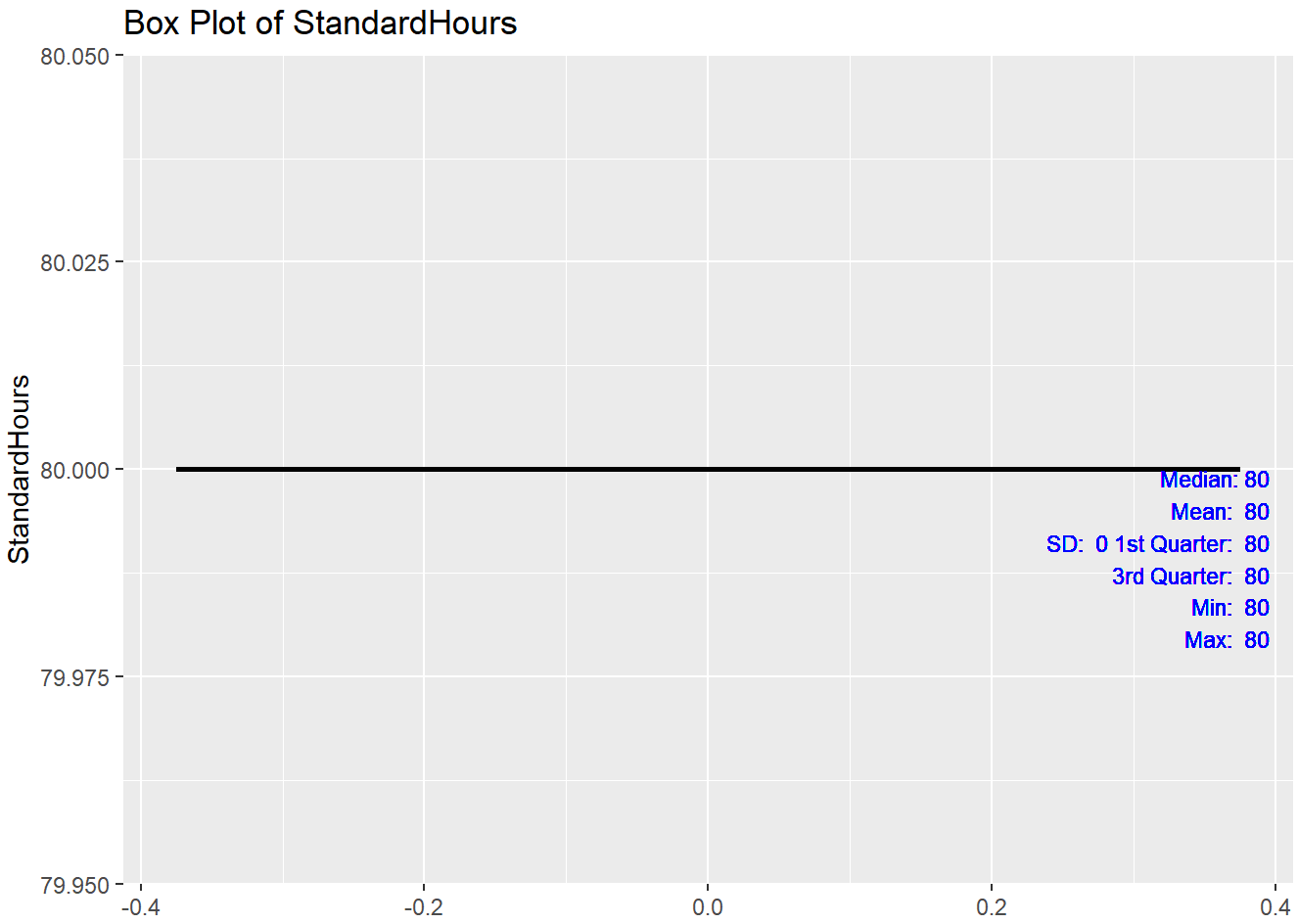

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



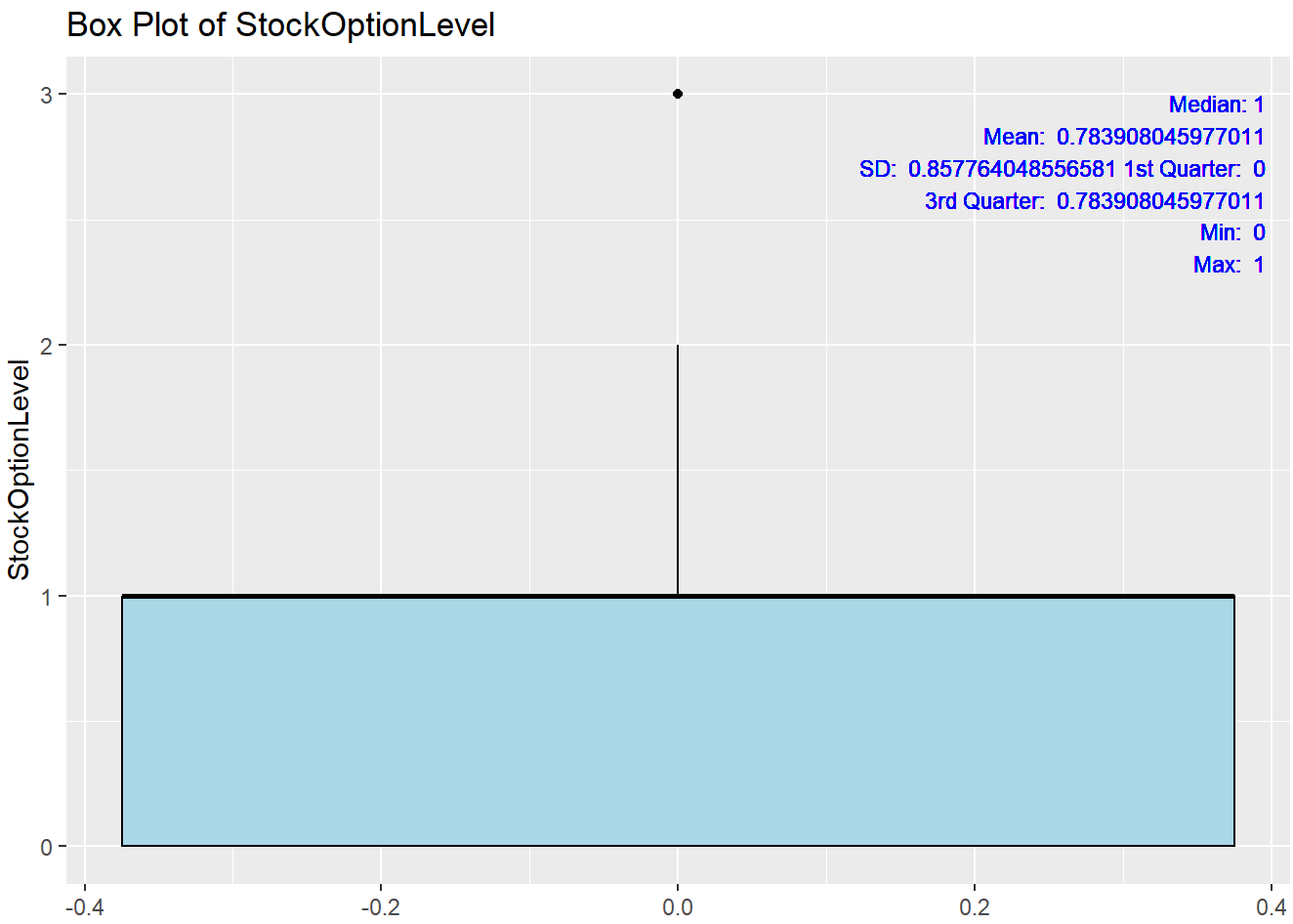

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



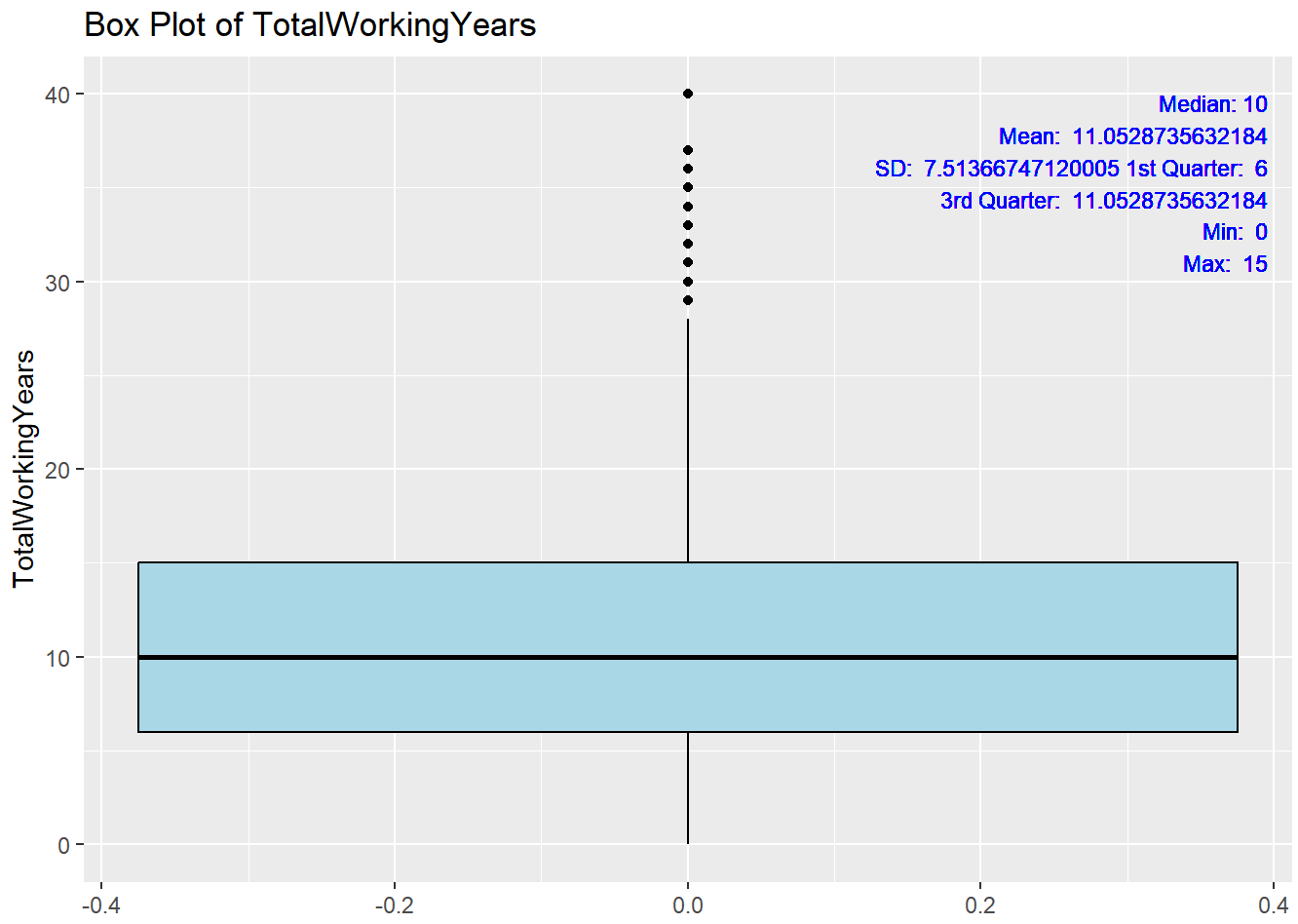

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



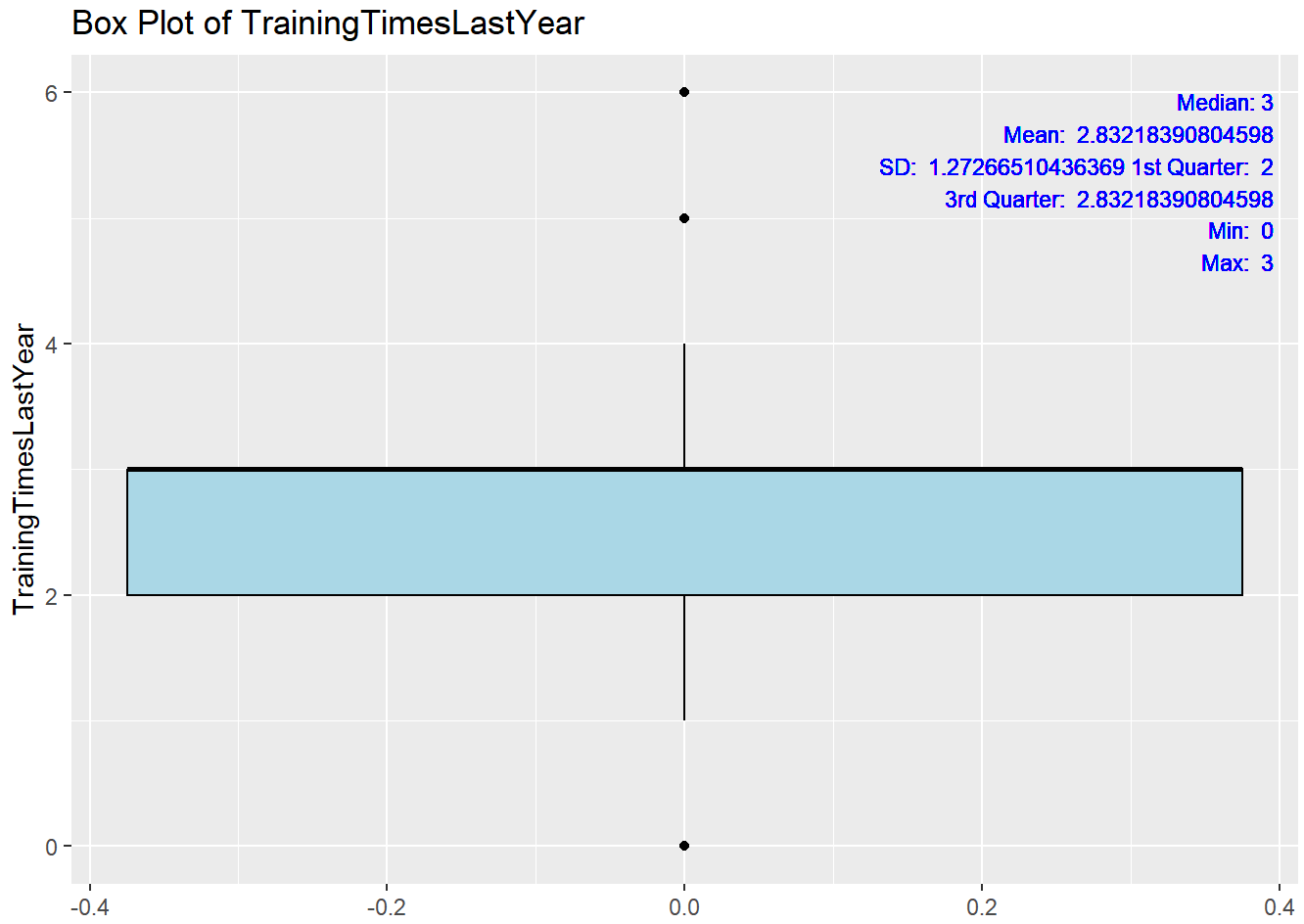

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



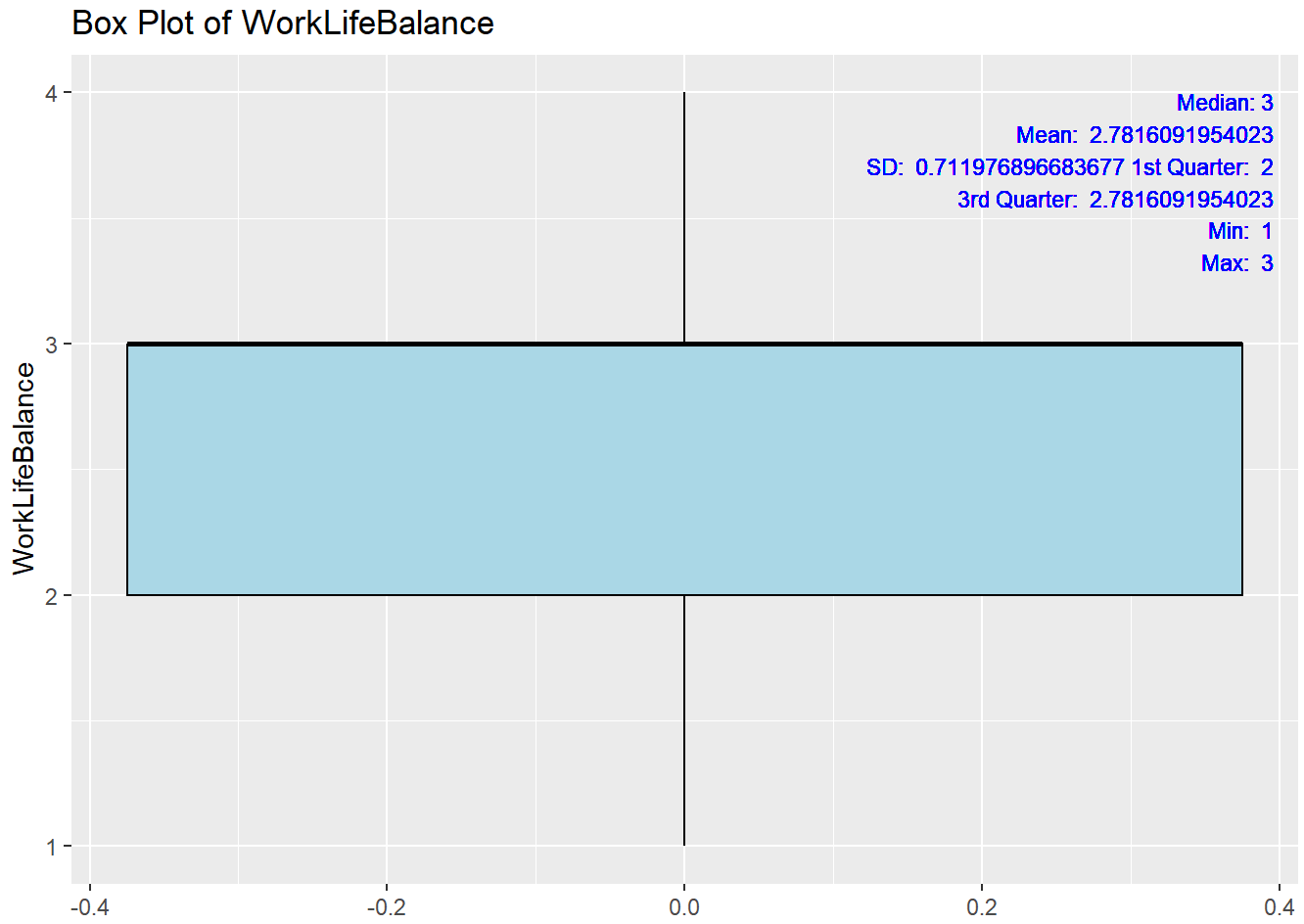

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



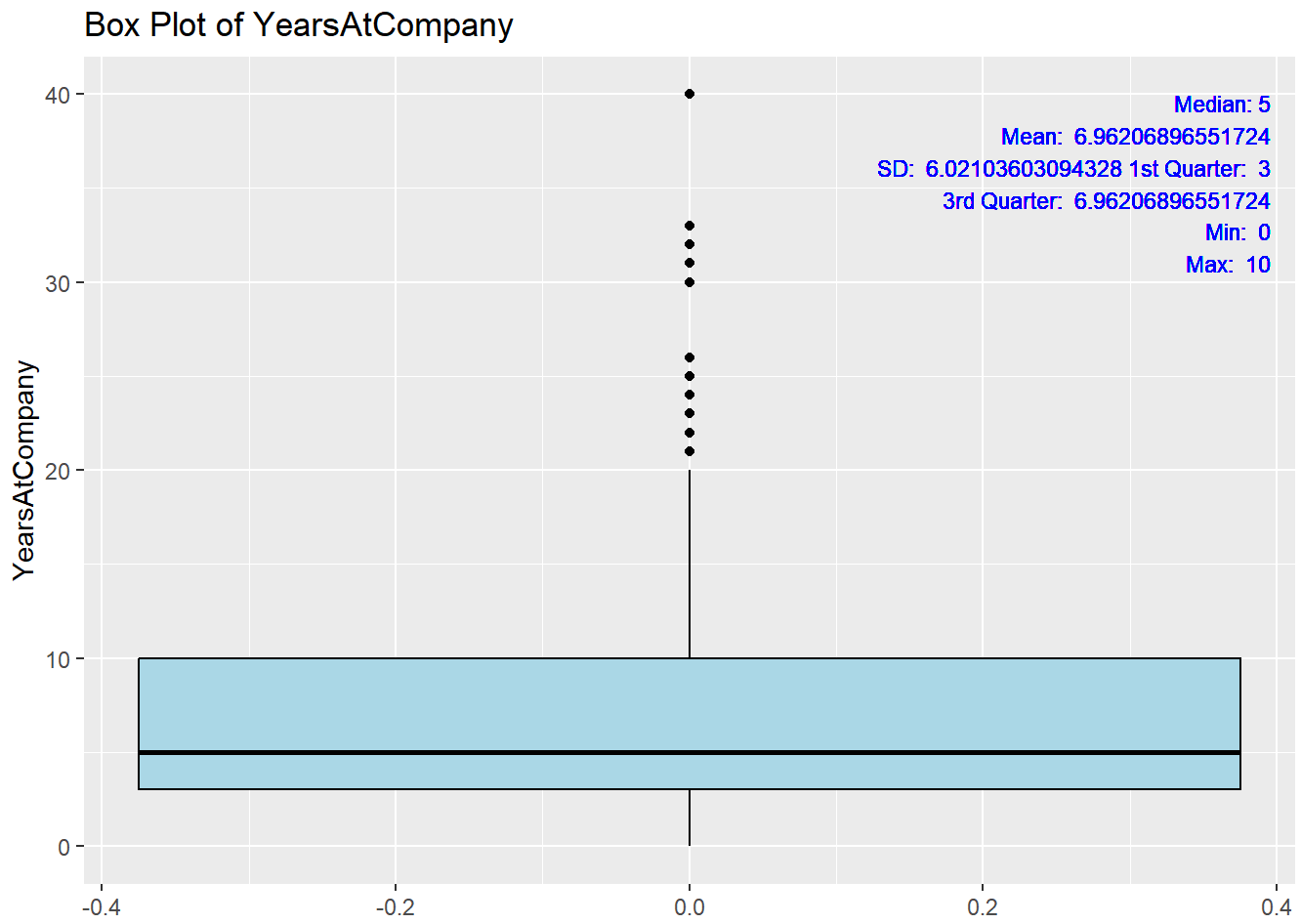

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



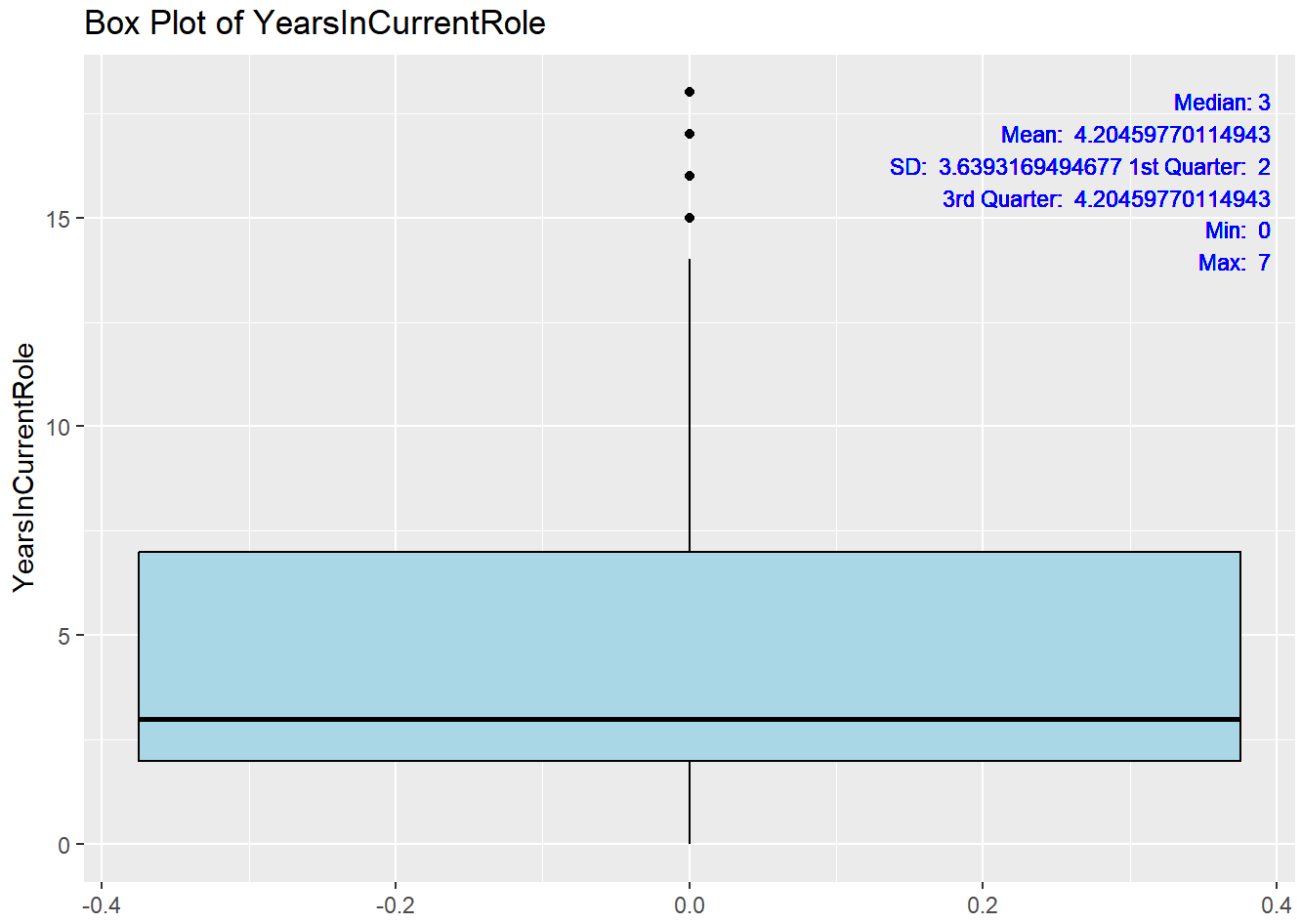

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



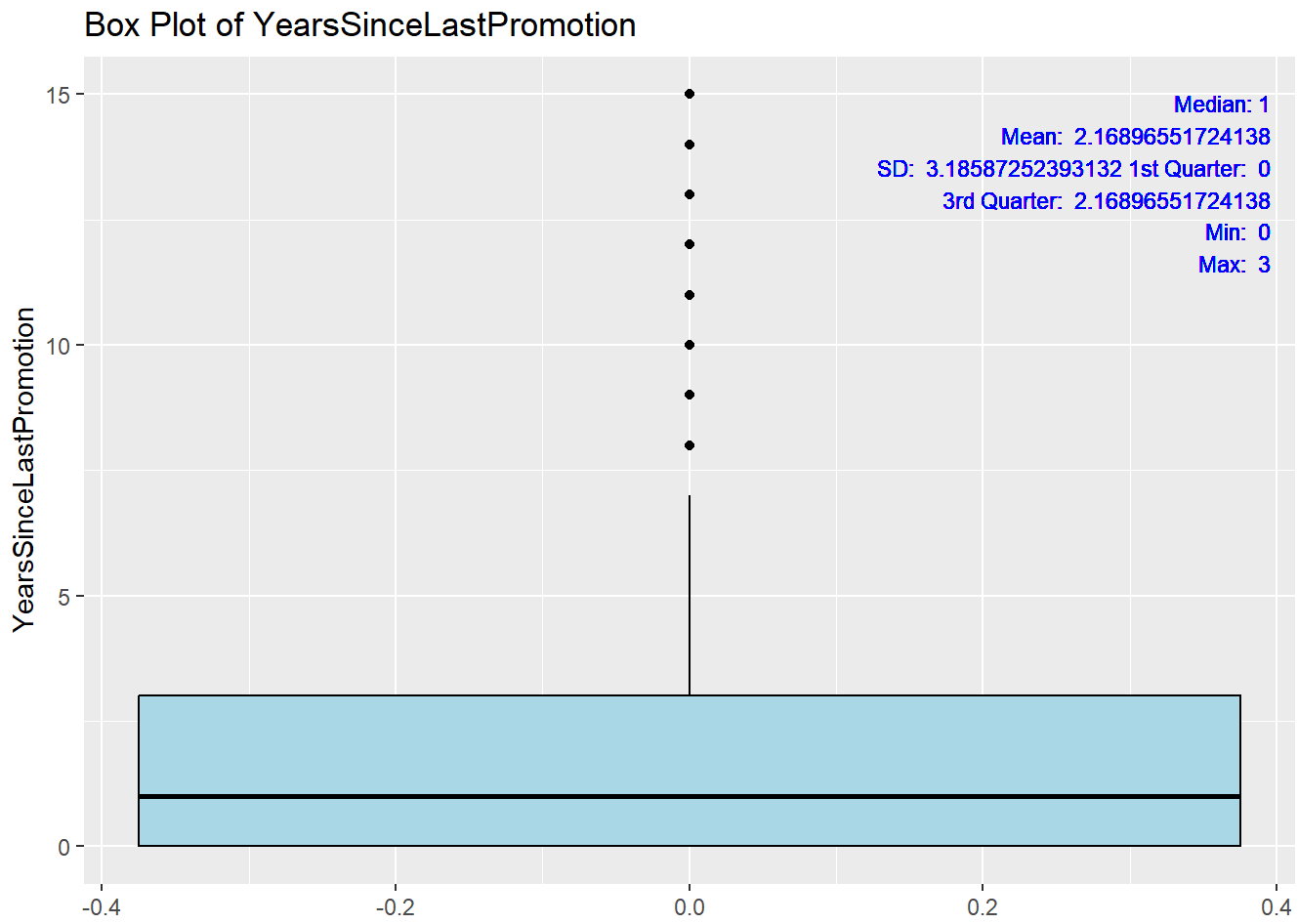

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



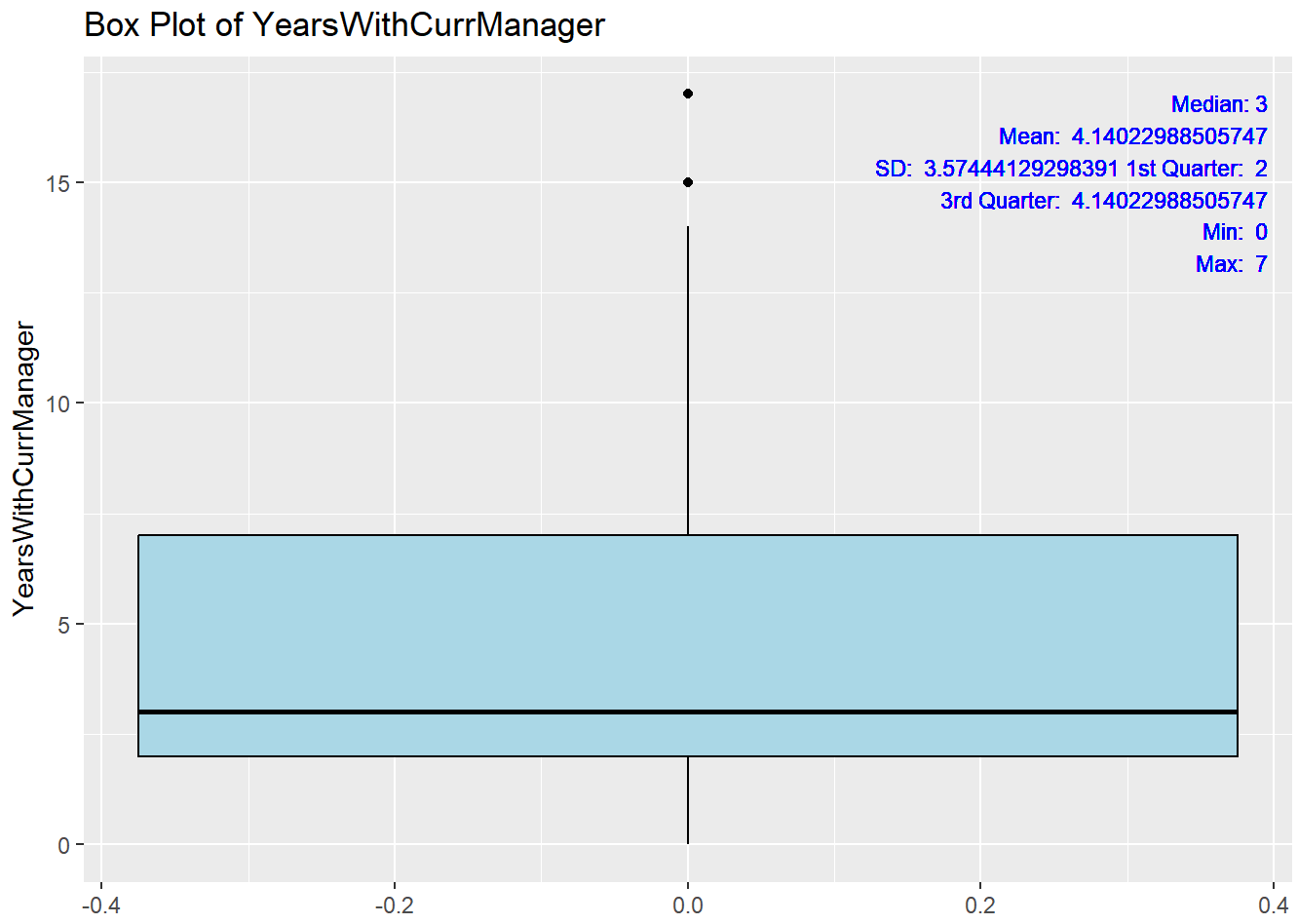

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```



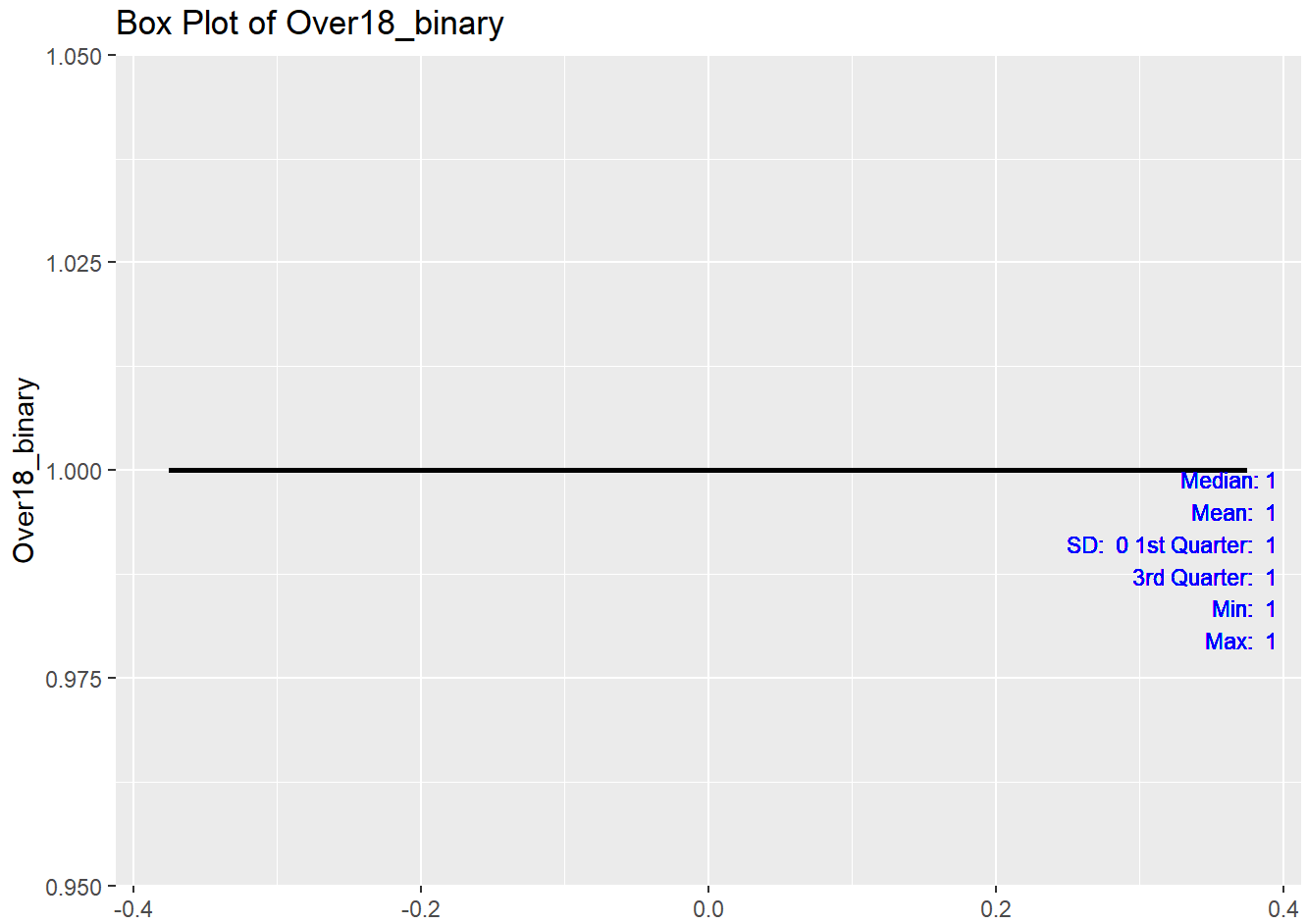

```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```




```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```




```
## Warning: Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
## Use of `Talent_Train[[Variable]]` is discouraged.
## ℹ Use `.data[[Variable]]` instead.
```

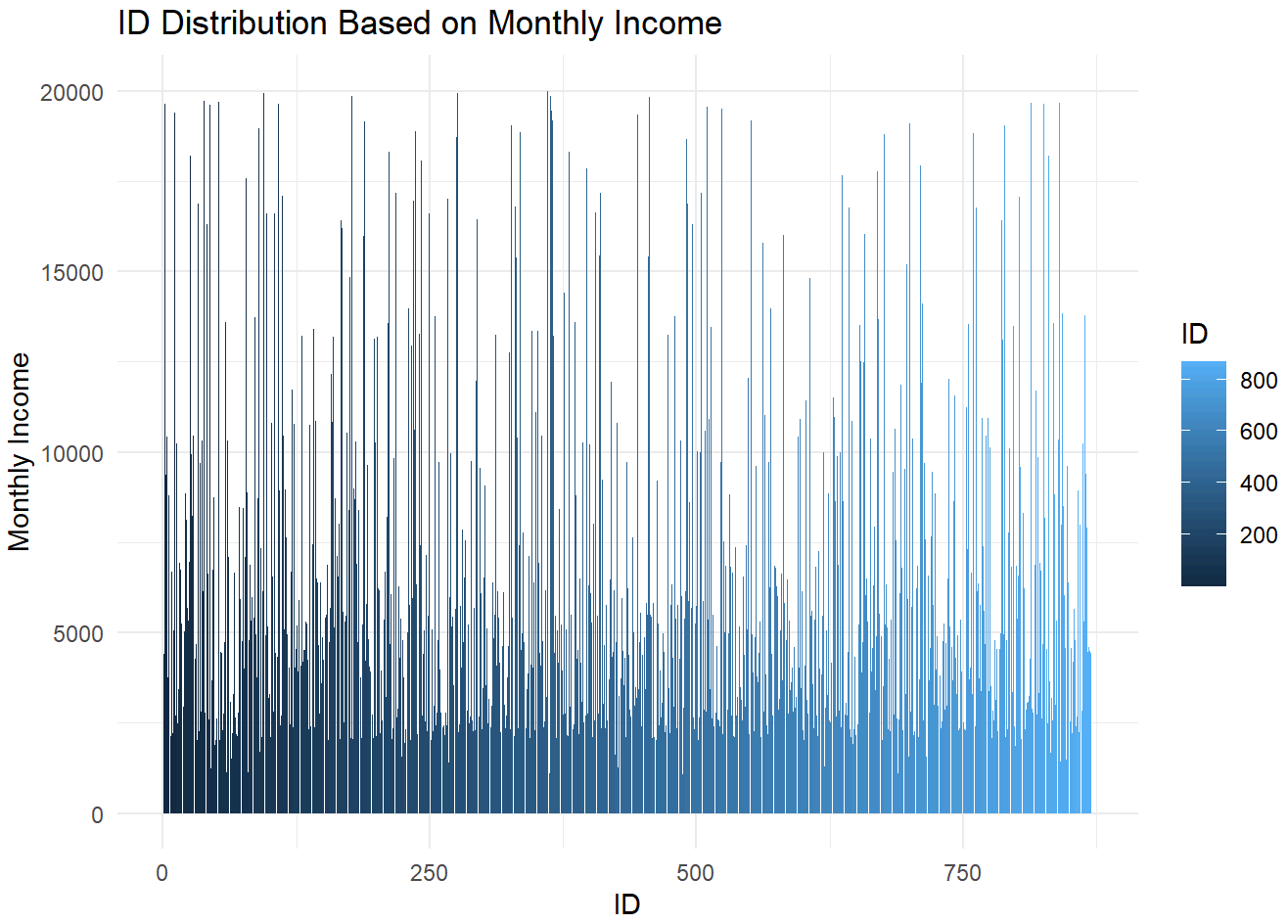


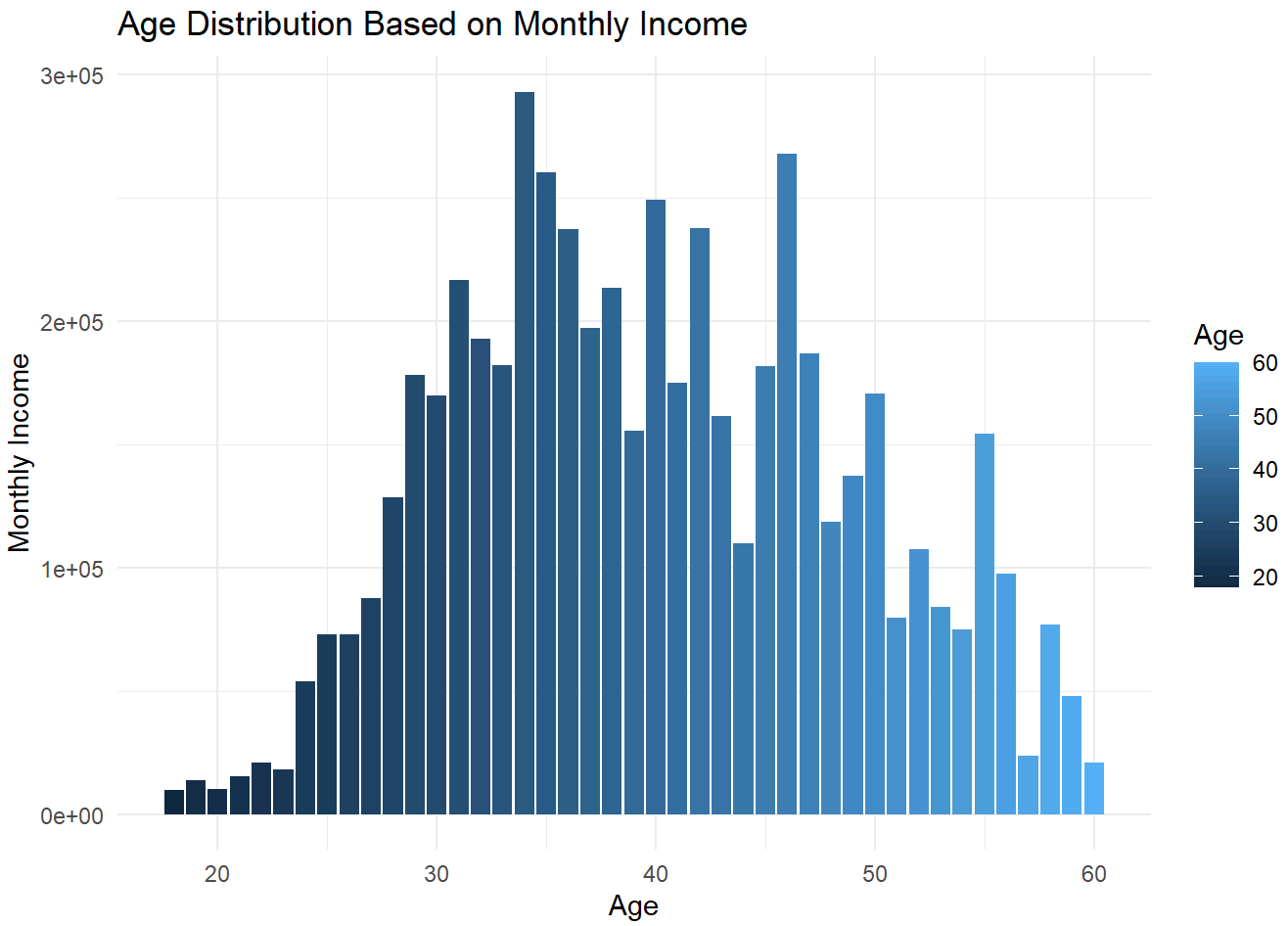
From the box plots, 4 variables have really bad spread of data. They are : EmployeeCount, Over_18_Binary, PerformanceRating, and StandardHours

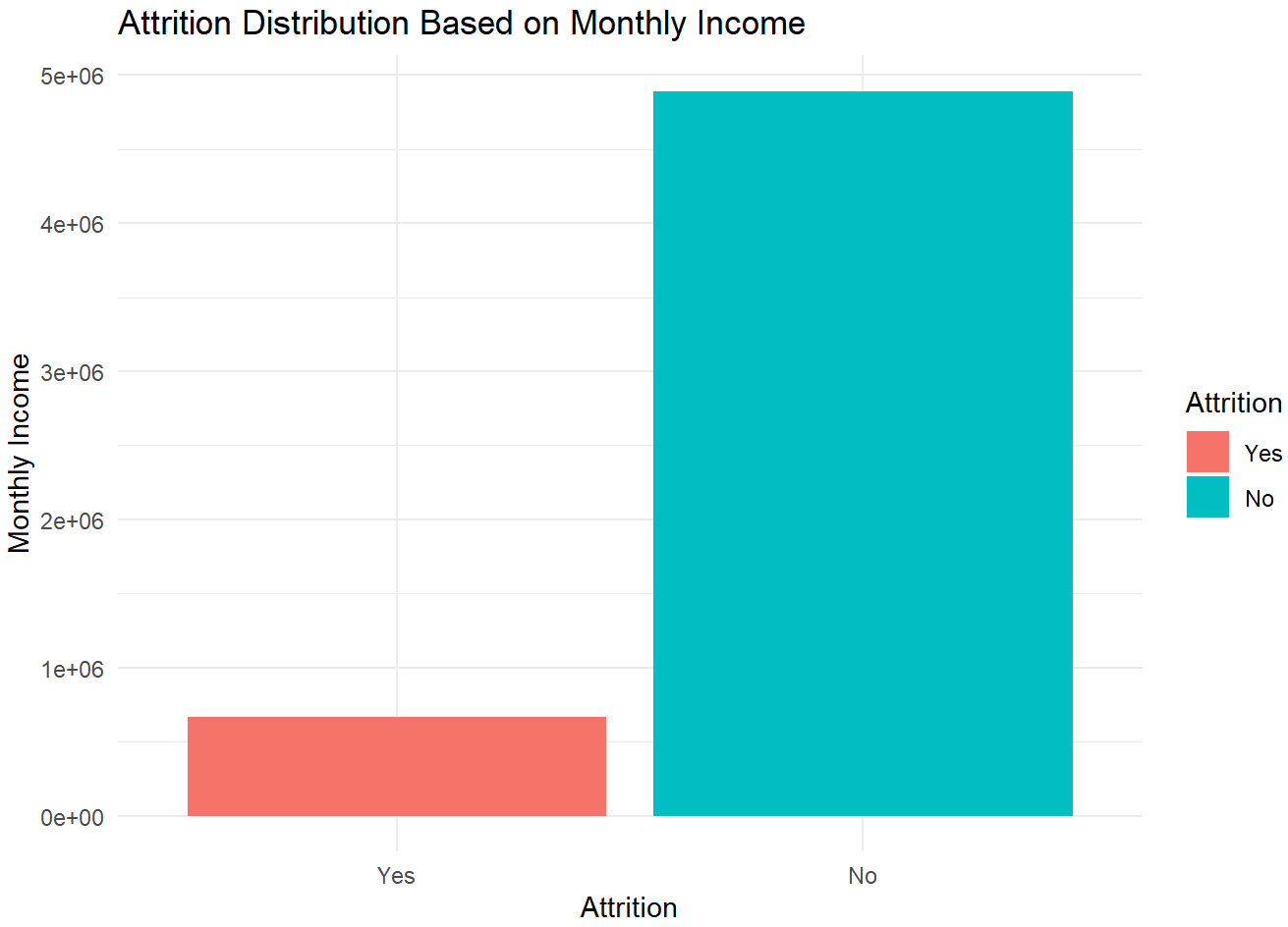
VISUALIZING THE VARIABLEs INTERACTIONS WITH THE RESPONSE VARIABLE (MONTHLY

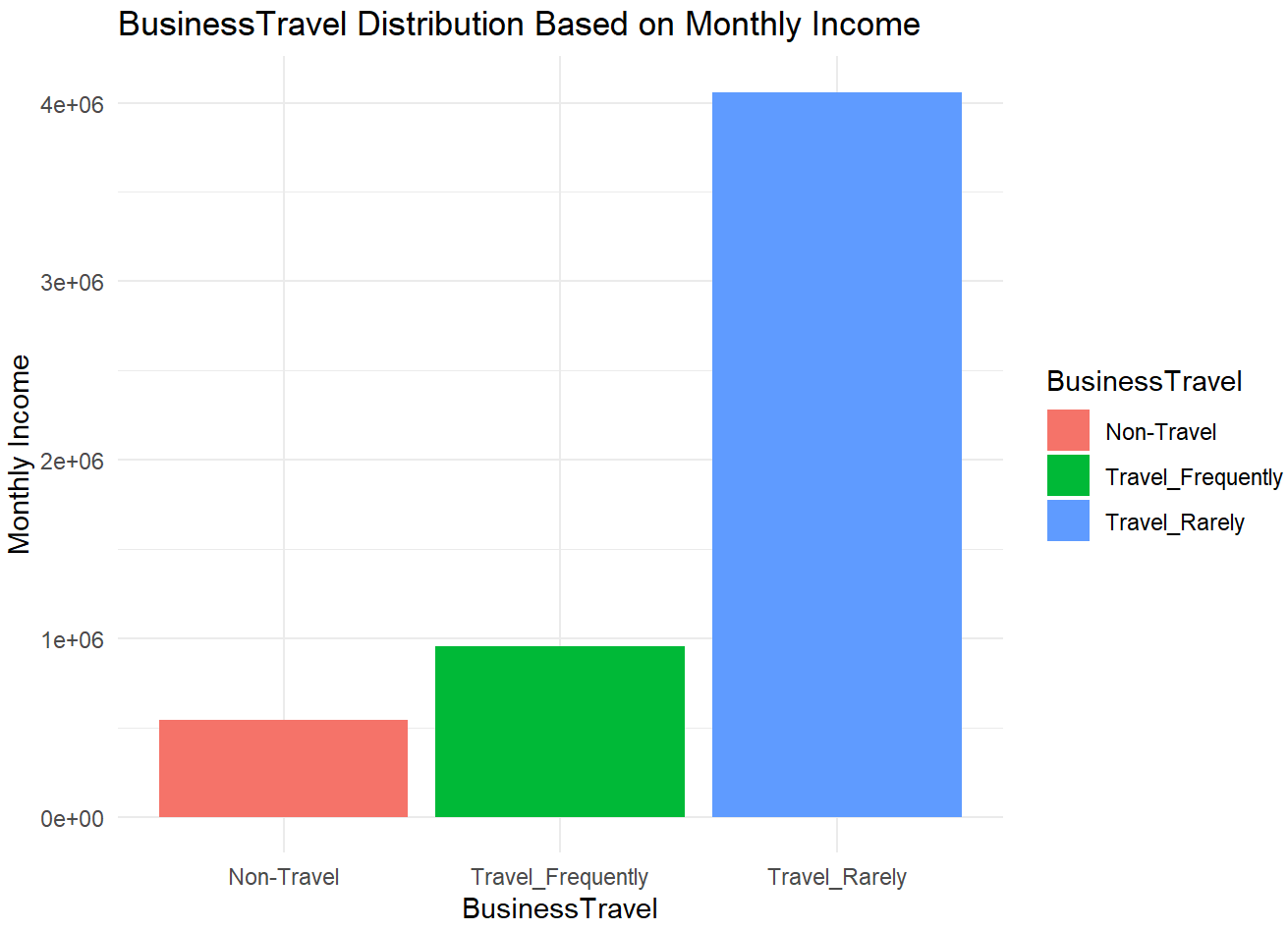
INCOME)

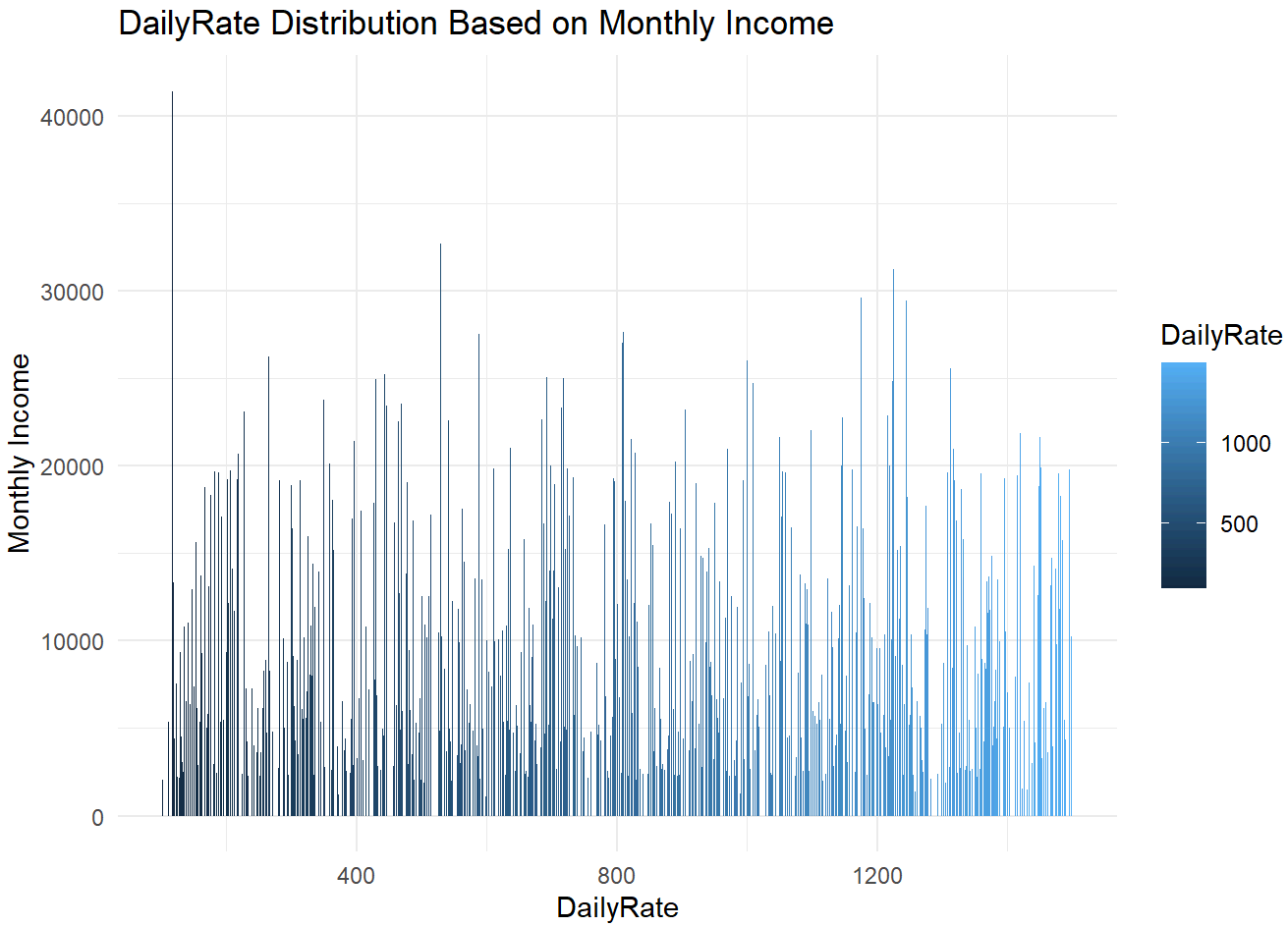
```
# Since the variables are a lot, and i plan on saving time, I plan using a for loop to iterate t
hrough the variables and plot their bar plots to visualize their EDA

# Iterate through the dataset and execute the plot command
for (Variable in names(Talent_Train)){
  # Check if the column is numeric
  if (!(Variable == "MonthlyIncome")) {

    # Construct the bunch of commands with the current variable

  command <- paste0(
    "Talent_Train %>% ggplot(aes(x=", Variable, ",y= MonthlyIncome, fill = ", Variable, ")) + ge
om_bar(stat='identity')  + \n",
    "ylab('Monthly Income') + xlab('", Variable, "') + ggtitle('", Variable, " Distribution Base
d on Monthly Income') + theme_minimal() \n"
  )

    # Execute the command
    plot <- eval(parse(text = command))

    print(plot)
  }
}
```

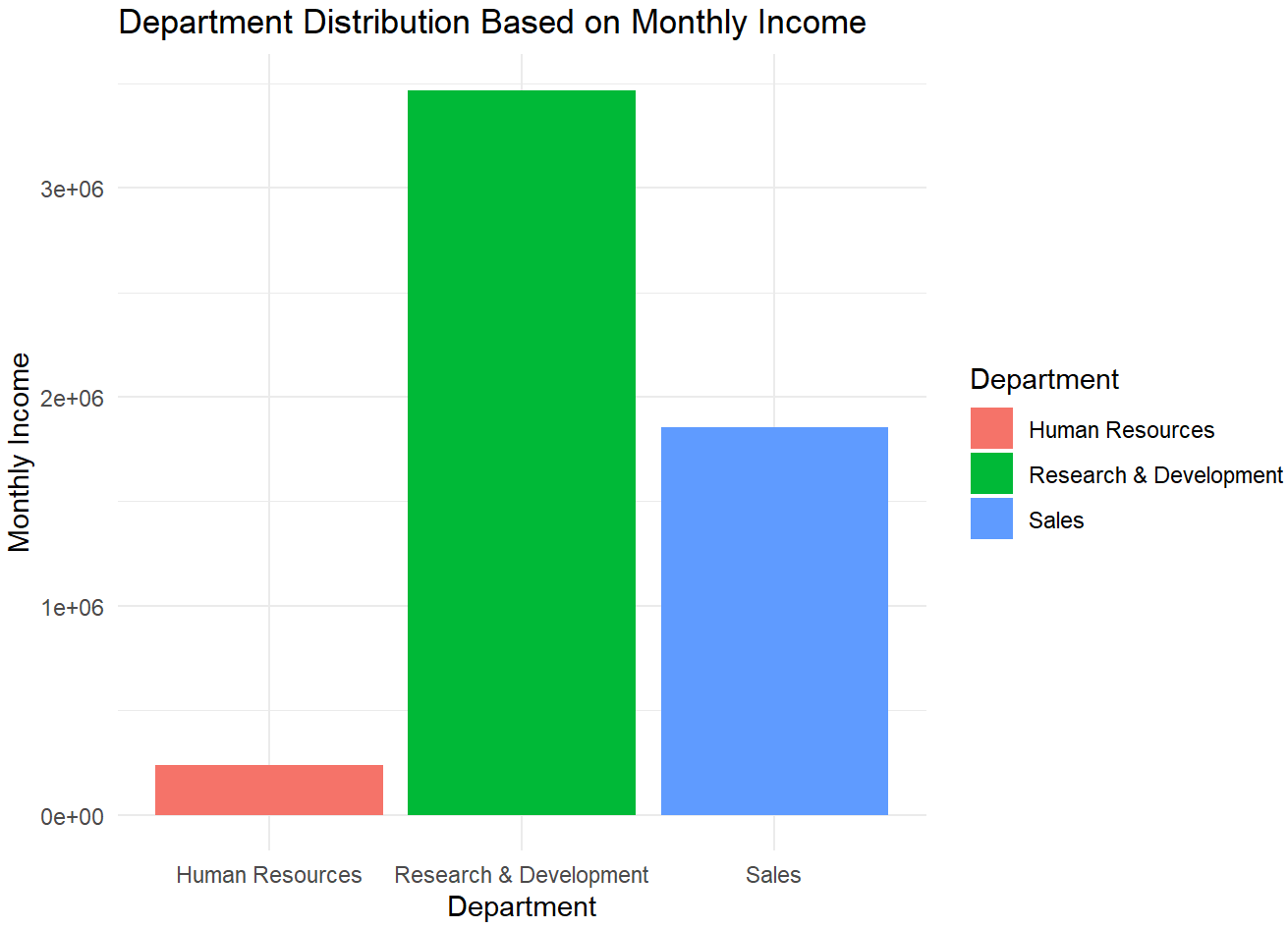

ID Distribution Based on Monthly Income
Monthly Income
20000
15000
10000
5000
0
0
250
500
750
ID
ID
800
600
400
200

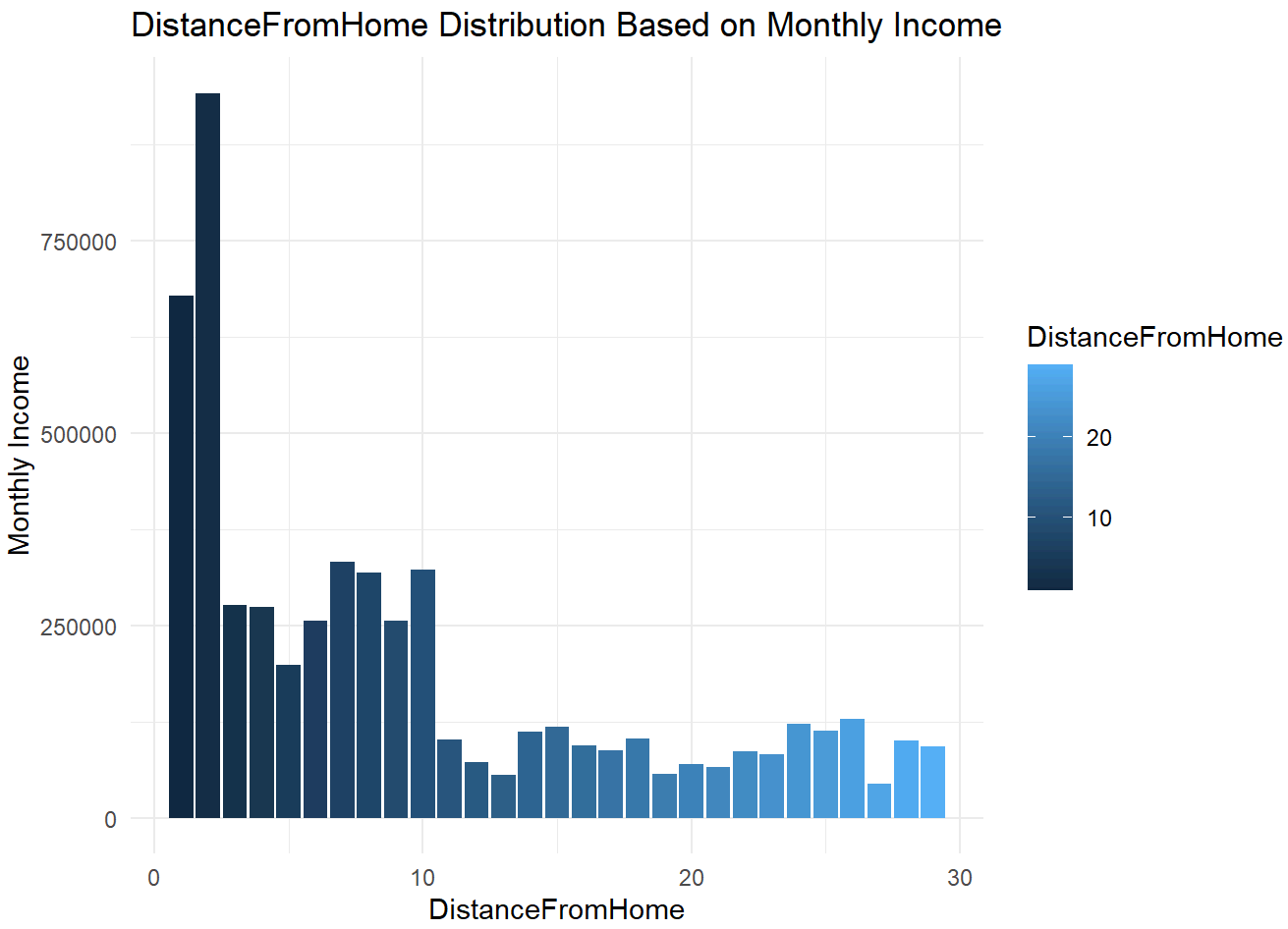

Age Distribution Based on Monthly Income
Monthly Income
3e+05
2e+05
1e+05
0e+00
20
30
40
50
60
Age
Age
60
50
40
30
20

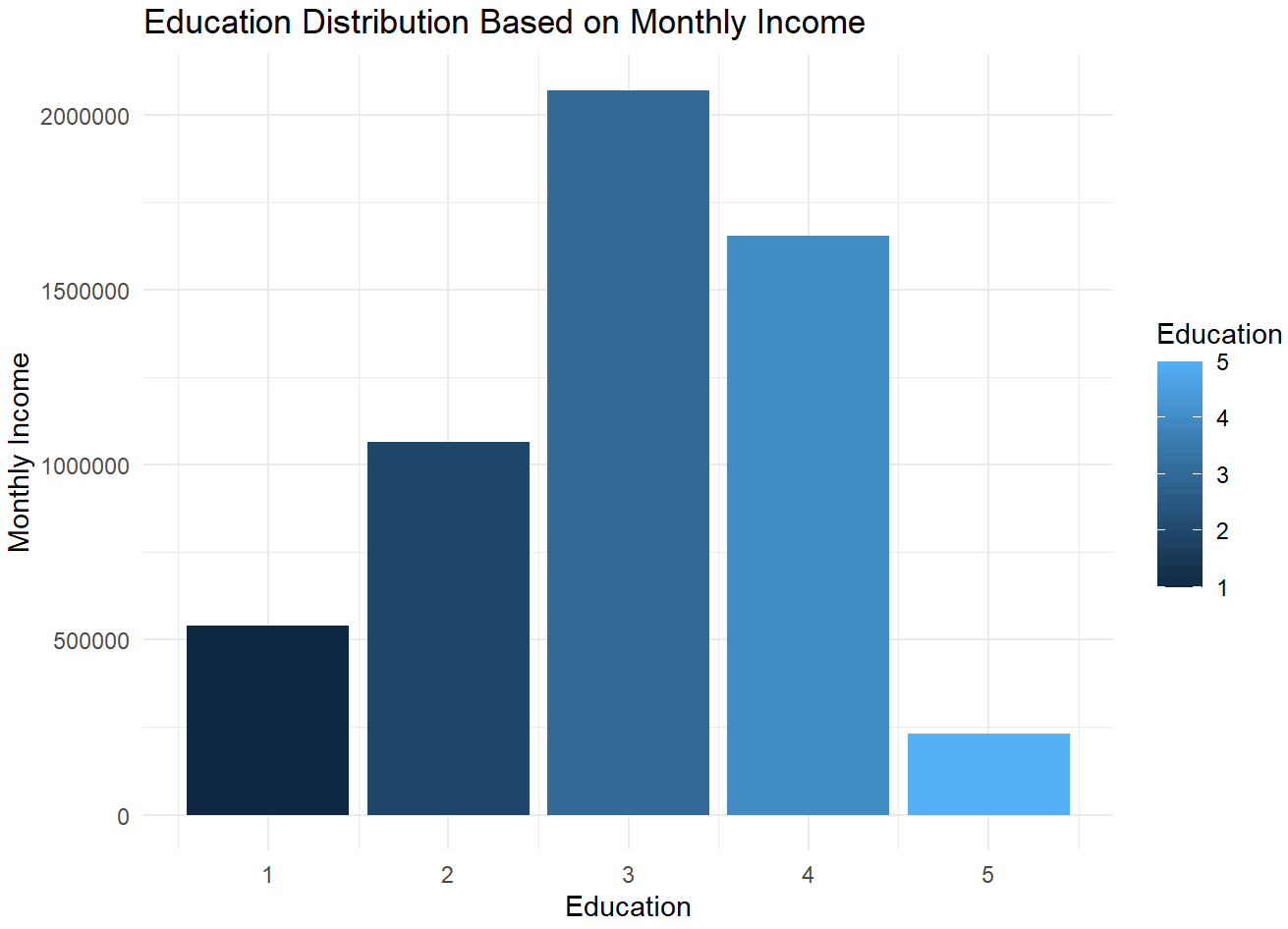

Attrition Distribution Based on Monthly Income
Monthly Income
5e+06
4e+06
3e+06
2e+06
1e+06
0e+00
Yes
No
Attrition
Attrition
Yes
No

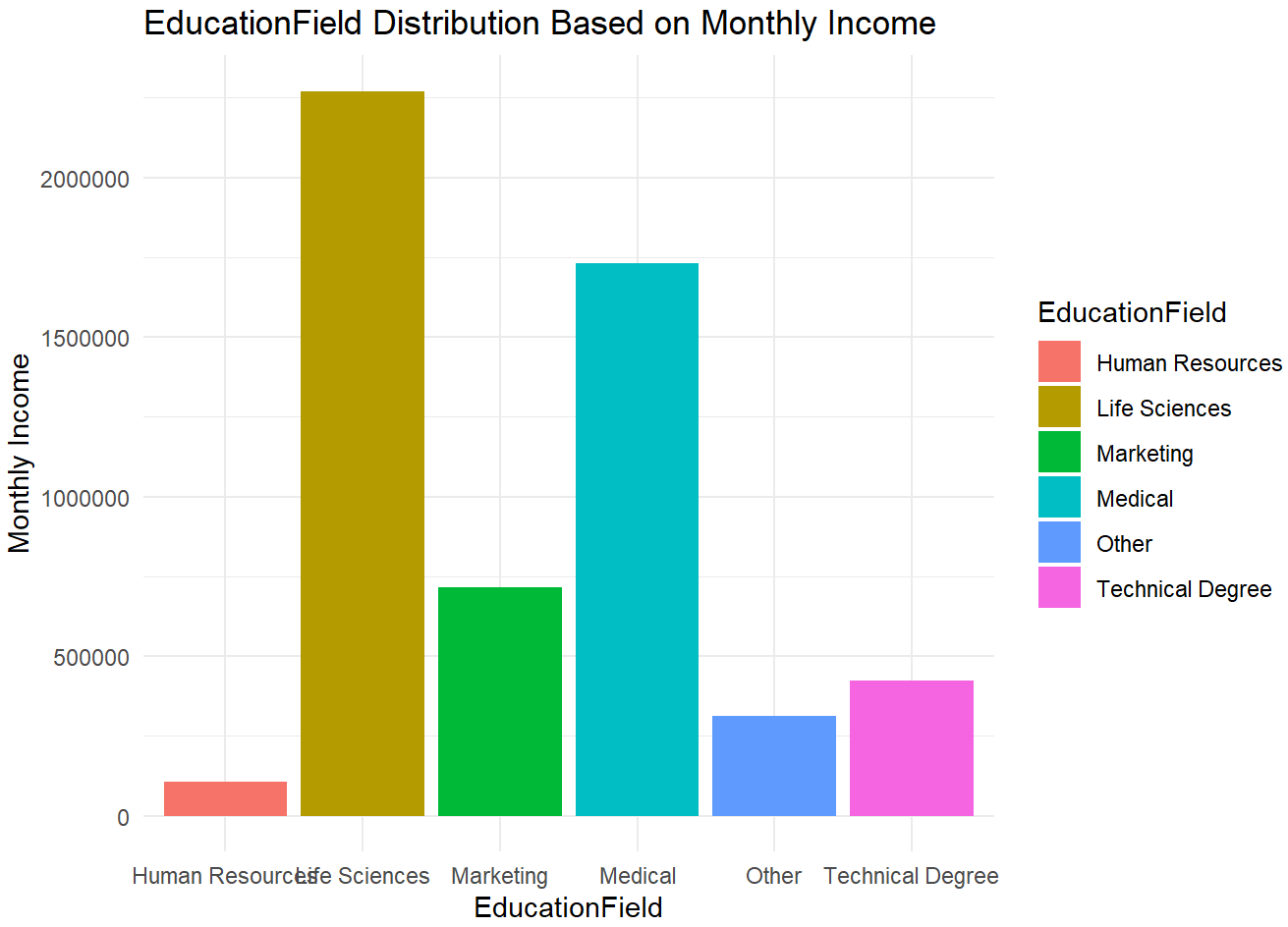

BusinessTravel Distribution Based on Monthly Income
Monthly Income
4e+06
3e+06
2e+06
1e+06
0e+00
Non-Travel
Travel_Frequently
Travel_Rarely
BusinessTravel
BusinessTravel
Non-Travel
Travel_Frequently
Travel_Rarely

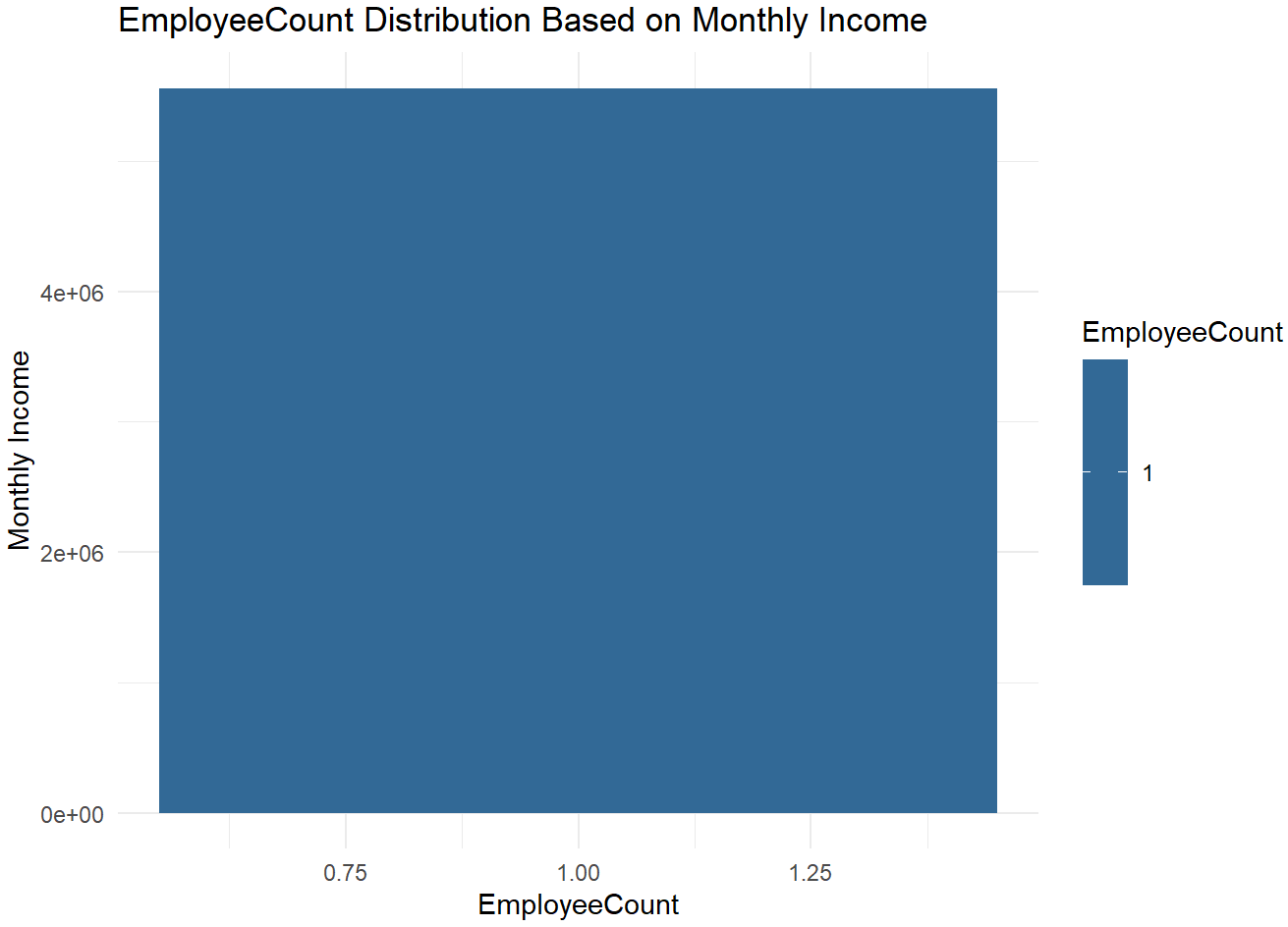

DailyRate Distribution Based on Monthly Income
Monthly Income
40000
30000
20000
10000
0
400
800
1200
DailyRate
DailyRate
1000
500

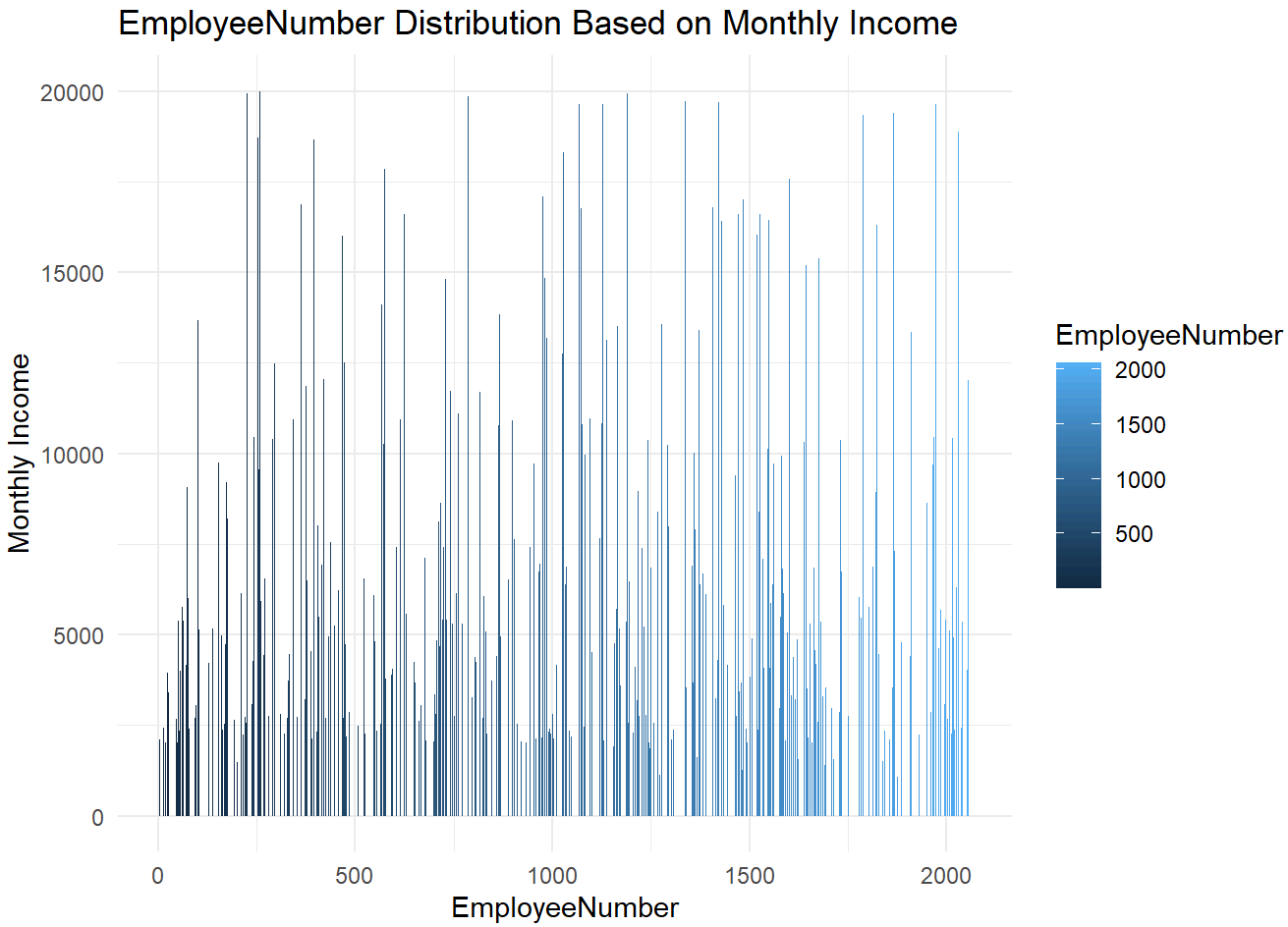

Department Distribution Based on Monthly Income
Monthly Income
3e+06
2e+06
1e+06
0e+00
Human Resources
Research & Development
Sales
Department
Department
Human Resources
Research & Development
Sales

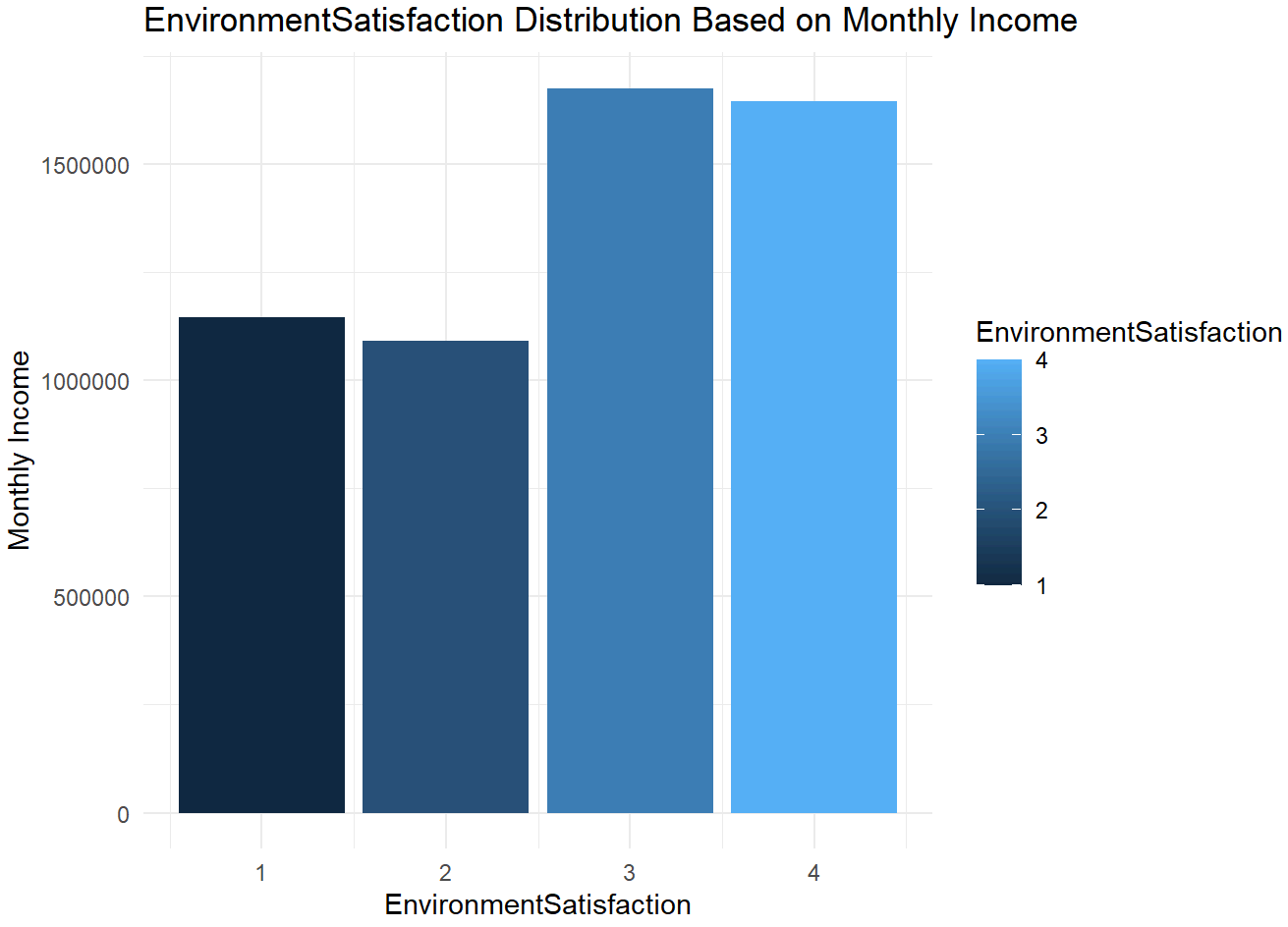

DistanceFromHome Distribution Based on Monthly Income
Monthly Income
750000
500000
250000
0
0
10
20
30
DistanceFromHome
DistanceFromHome
20
10

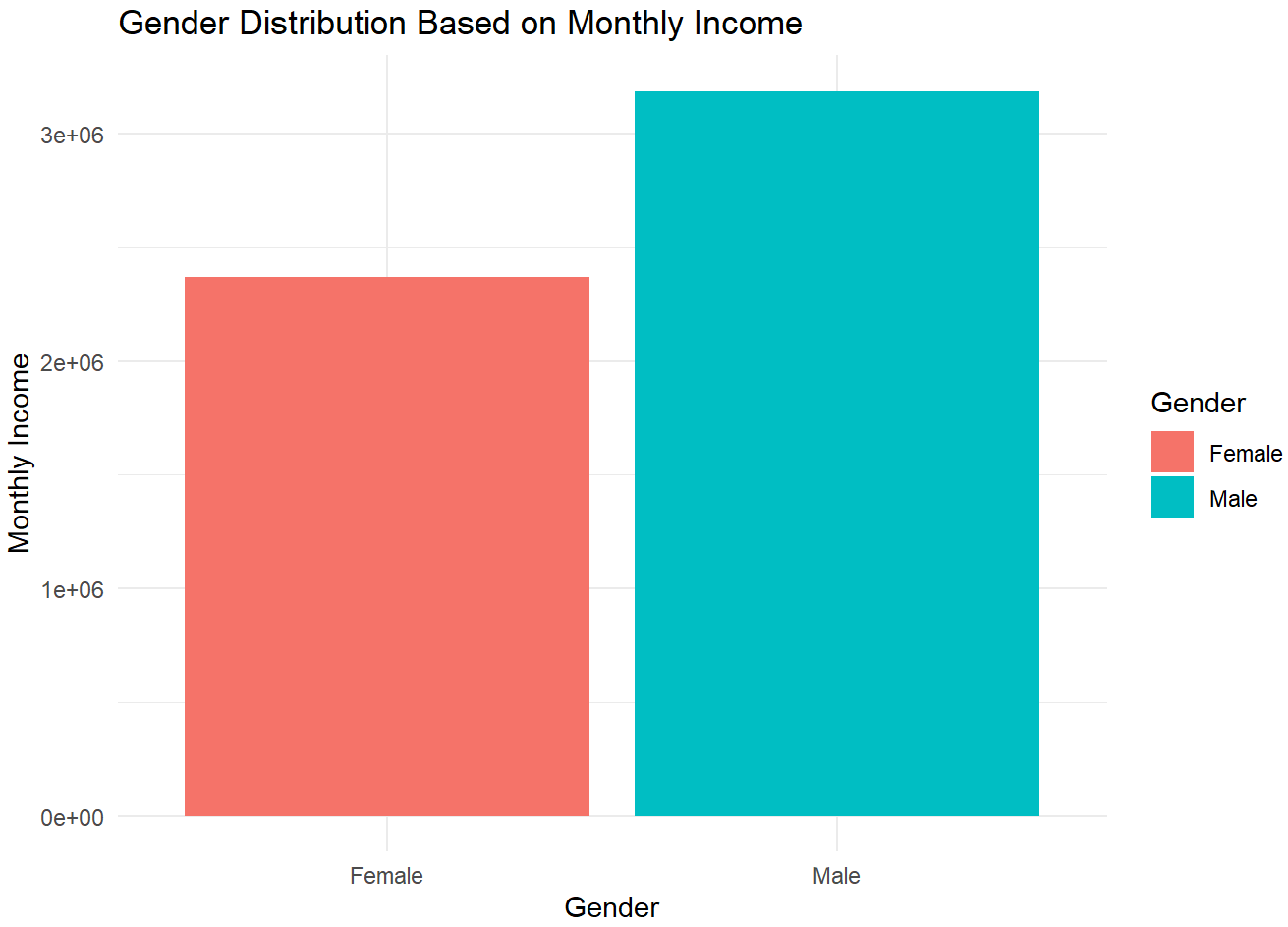

Education Distribution Based on Monthly Income
Monthly Income
2000000
1500000
1000000
500000
0
1
2
3
4
5
Education
Education
5
4
3
2
1

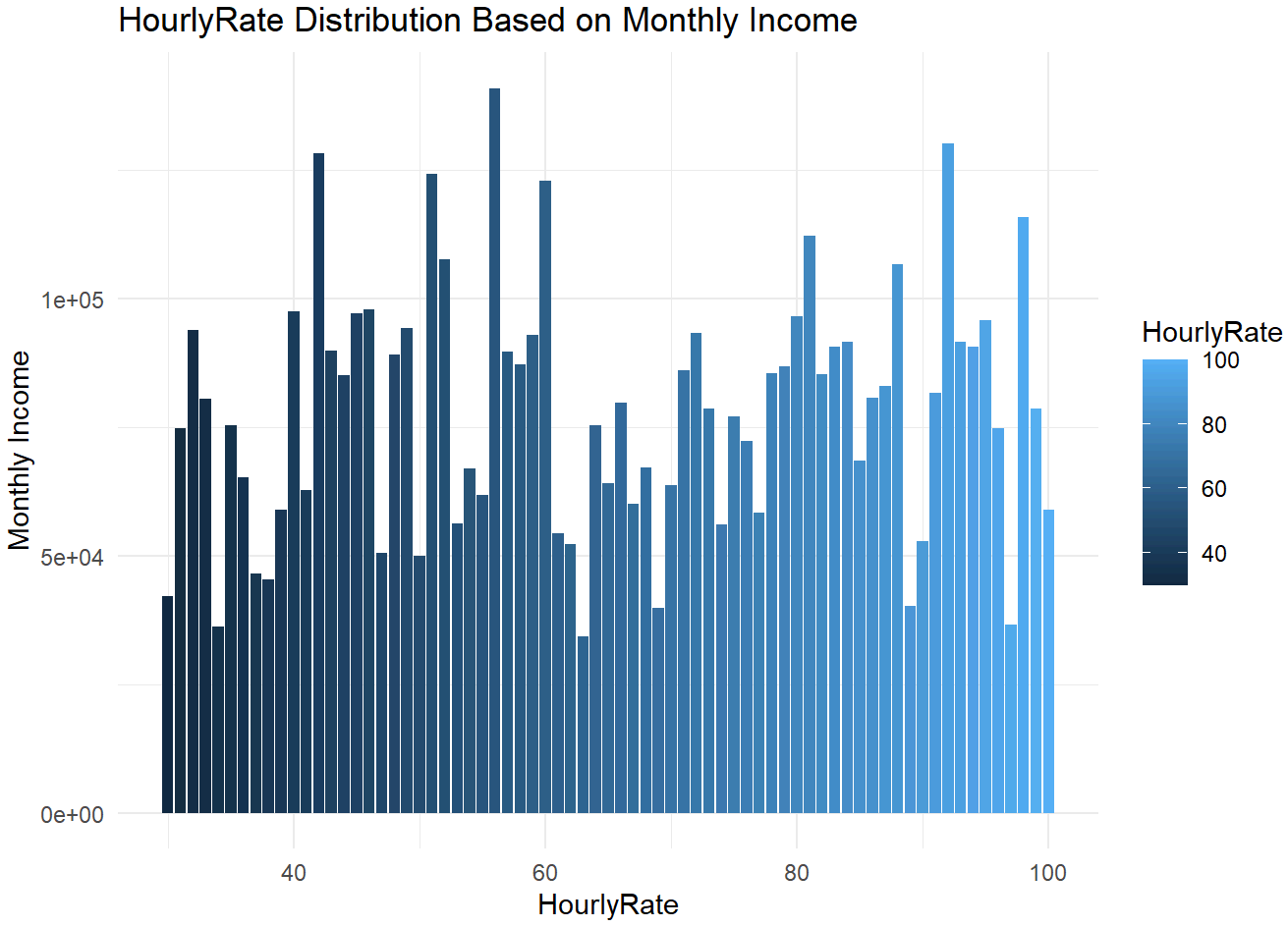

EducationField Distribution Based on Monthly Income
Monthly Income
2000000
1500000
1000000
500000
0
Human Resources
Life Sciences
Marketing
Medical
Other
Technical Degree
EducationField
EducationField
Human Resources
Life Sciences
Marketing
Medical
Other
Technical Degree

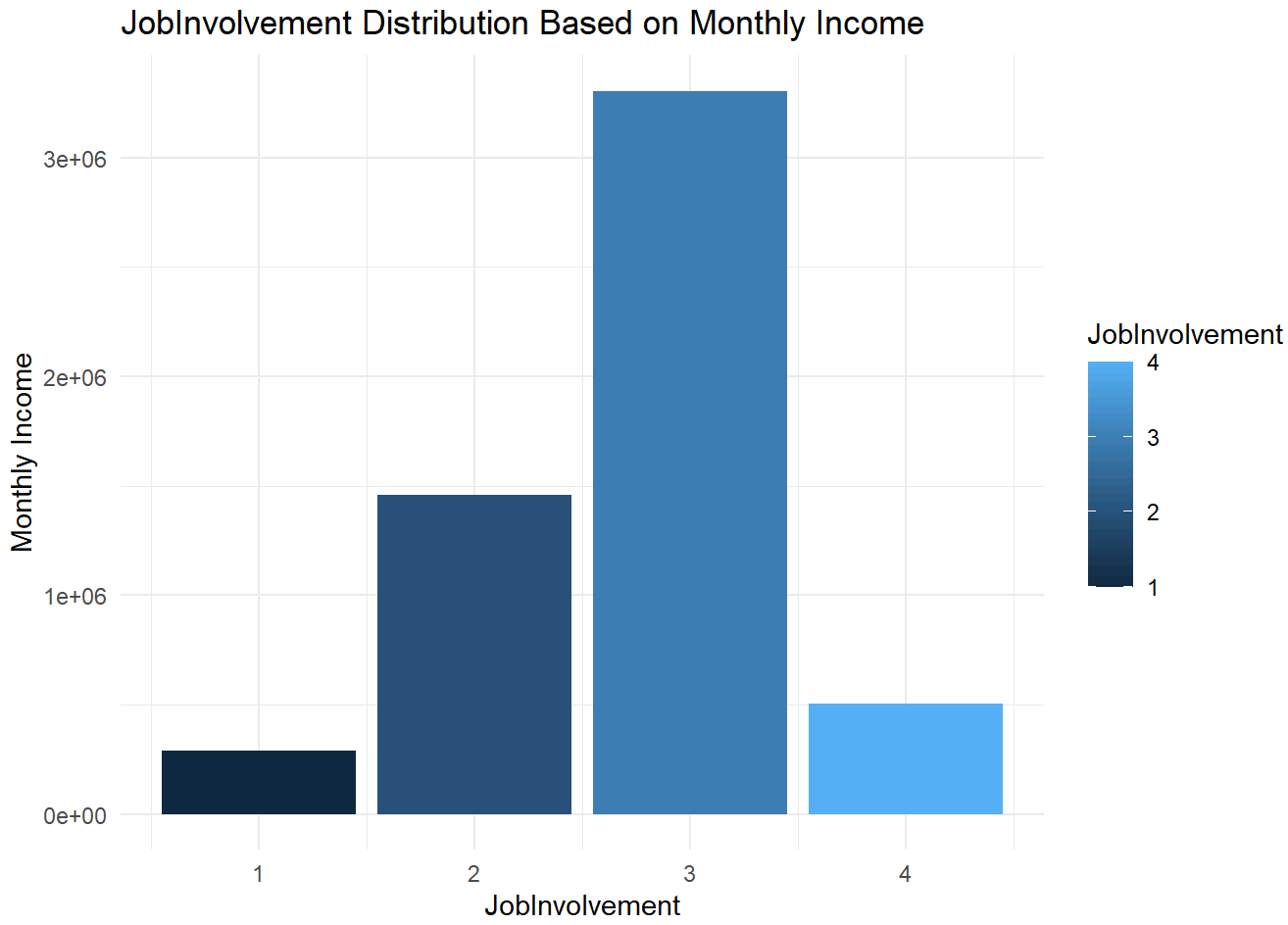

EmployeeCount Distribution Based on Monthly Income
Monthly Income
4e+06
2e+06
0e+00
0.75
1.00
1.25
EmployeeCount
EmployeeCount
1

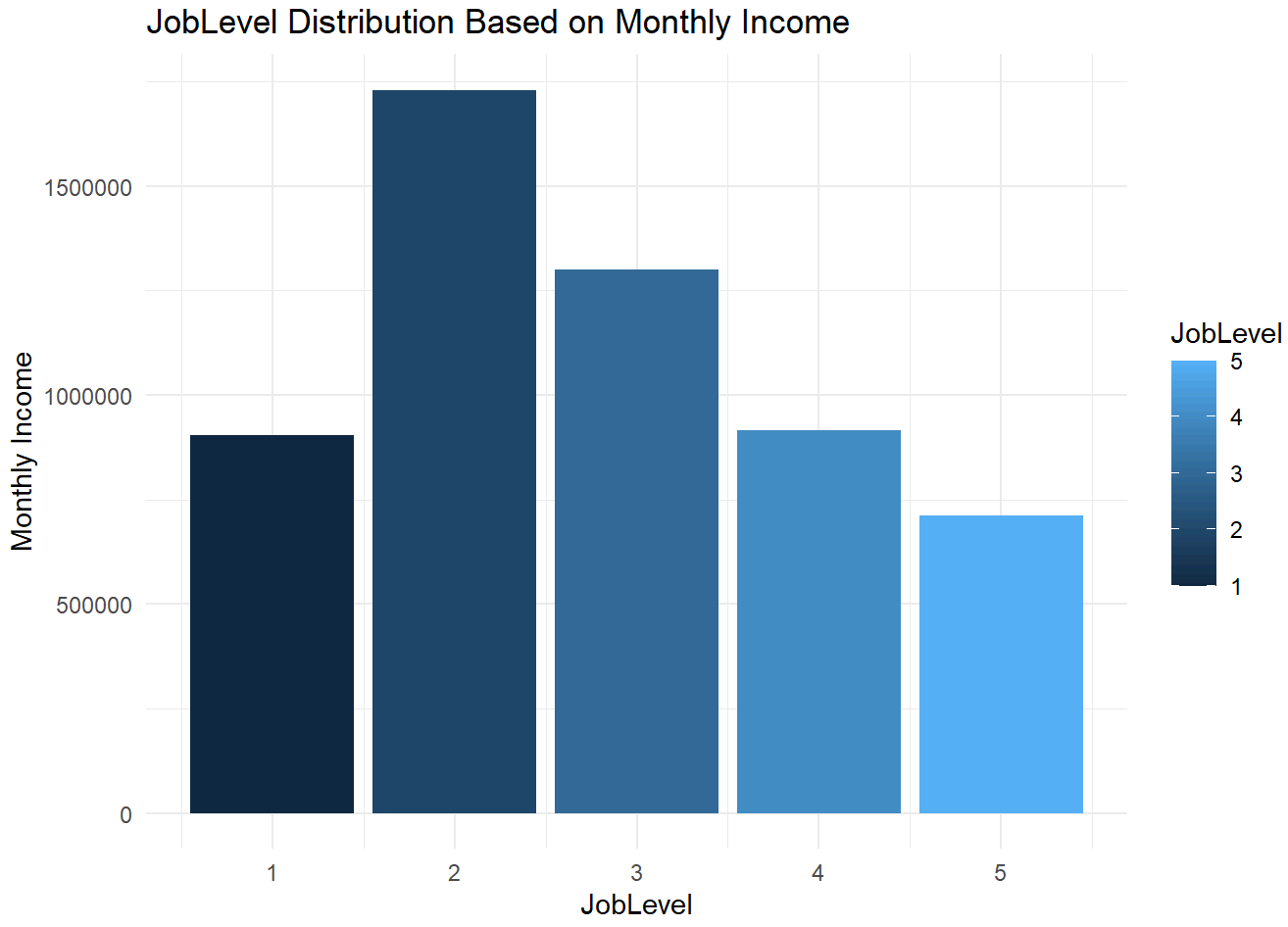

EmployeeNumber Distribution Based on Monthly Income
Monthly Income
20000
15000
10000
5000
0
0
500
1000
1500
2000
EmployeeNumber
EmployeeNumber
2000
1500
1000
500

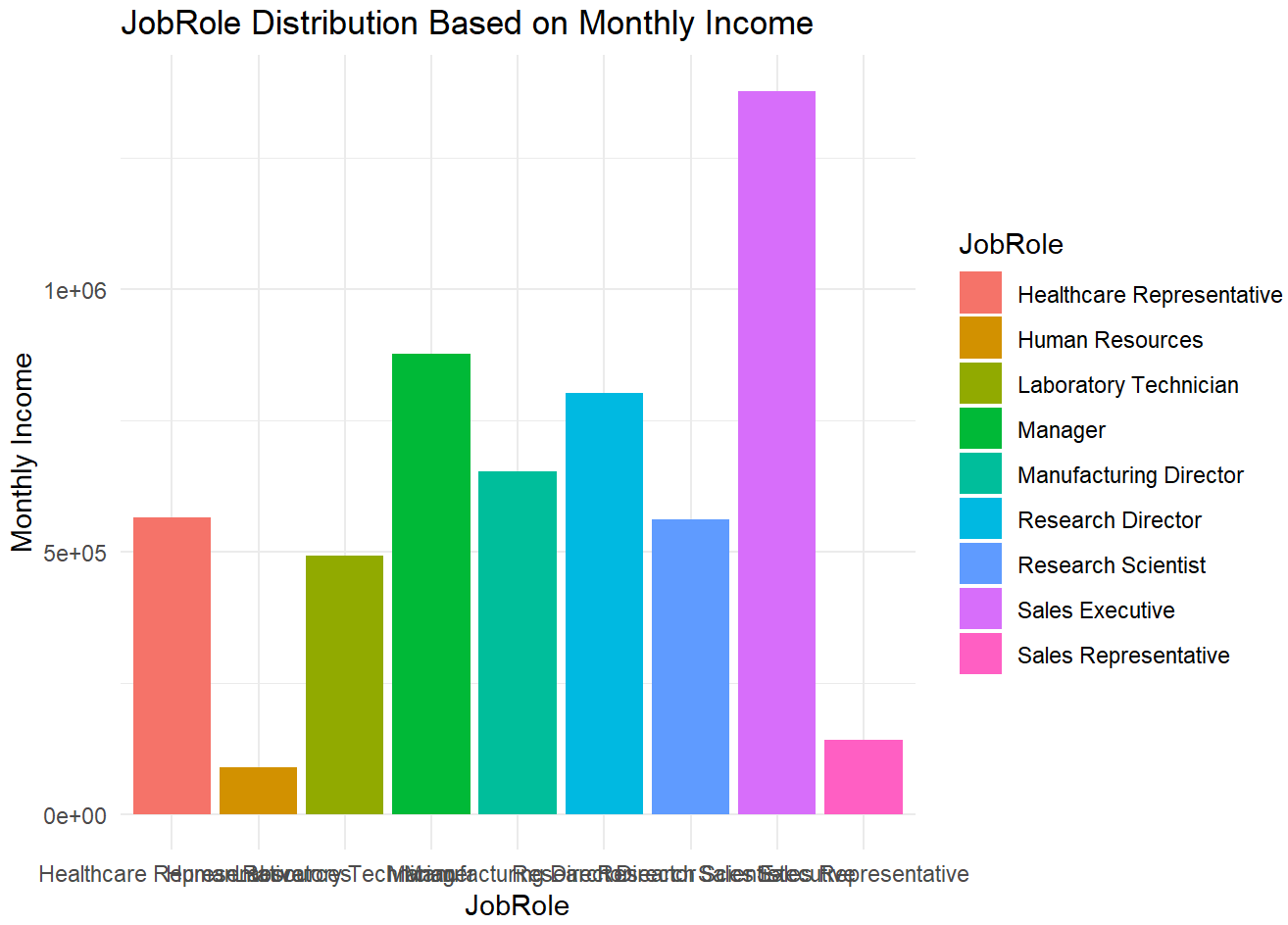

EnvironmentSatisfaction Distribution Based on Monthly Income
Monthly Income
1500000
1000000
500000
0
1
2
3
4
EnvironmentSatisfaction
EnvironmentSatisfaction
4
3
2
1

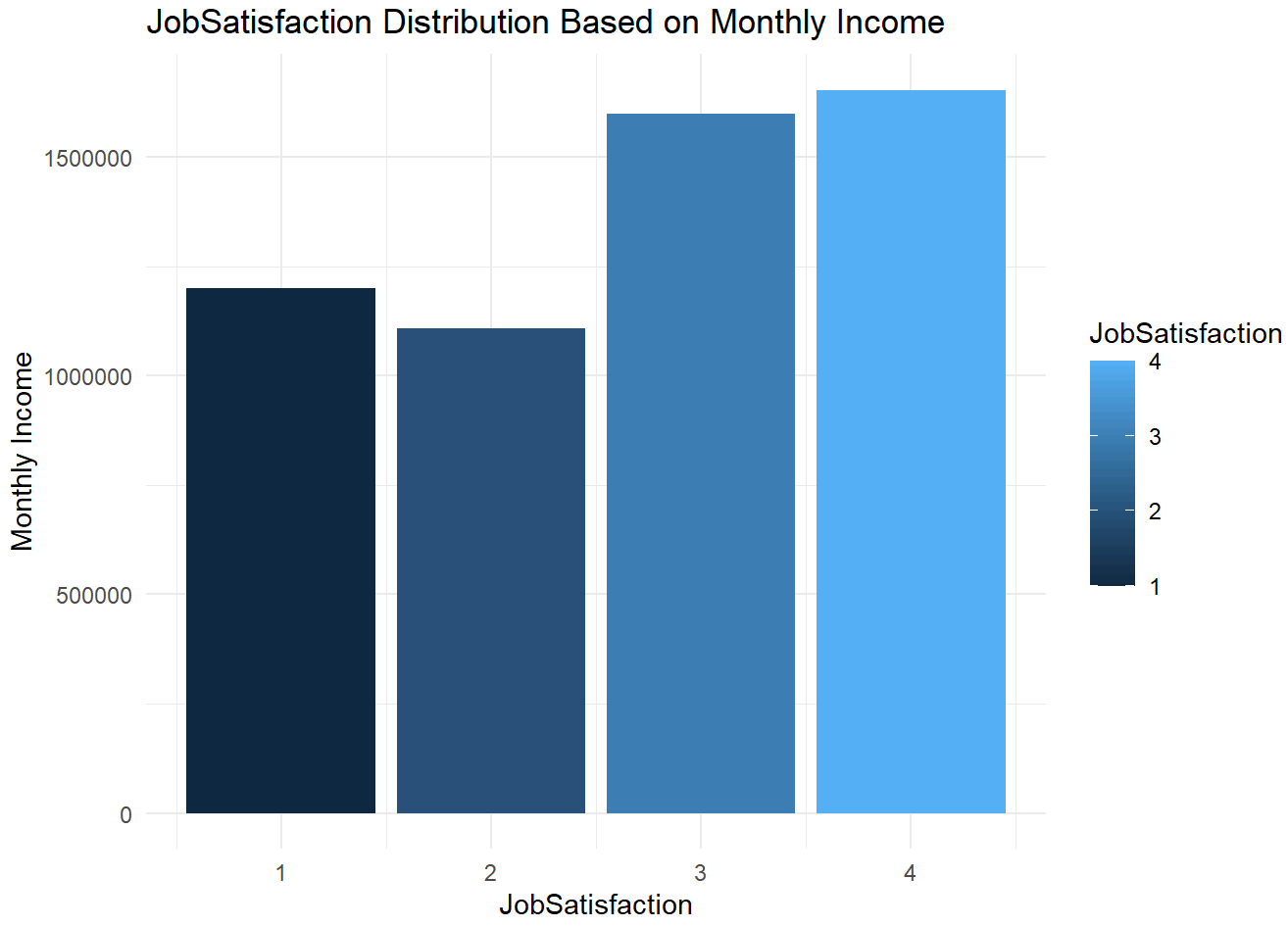

Gender Distribution Based on Monthly Income
Monthly Income
3e+06
2e+06
1e+06
0e+00
Female
Male
Gender
Gender
Female
Male

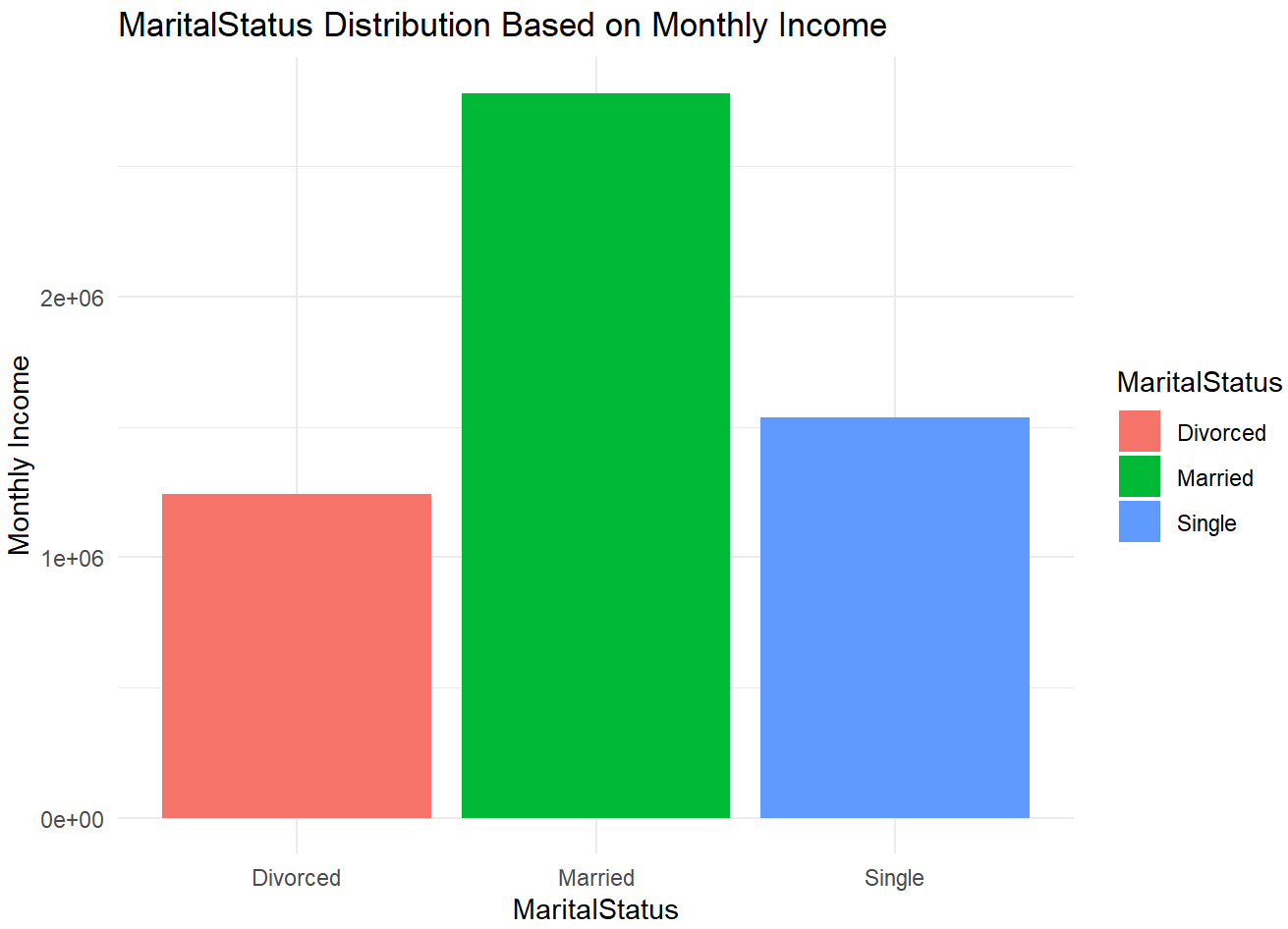

HourlyRate Distribution Based on Monthly Income
Monthly Income
1e+05
5e+04
0e+00
40
60
80
100
HourlyRate
HourlyRate
100
80
60
40

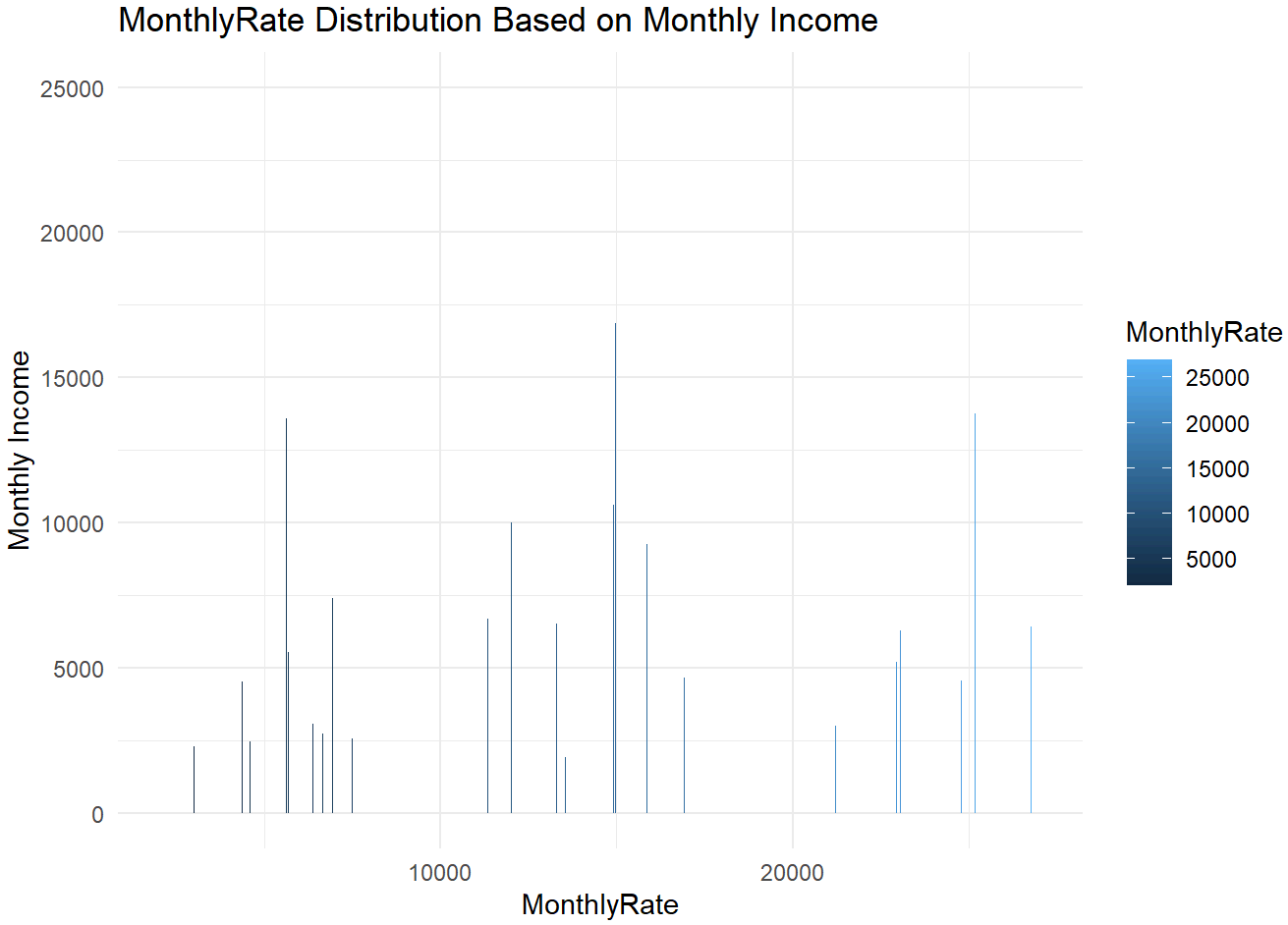

JobInvolvement Distribution Based on Monthly Income
Monthly Income
3e+06
2e+06
1e+06
0e+00
1
2
3
4
JobInvolvement
JobInvolvement
4
3
2
1

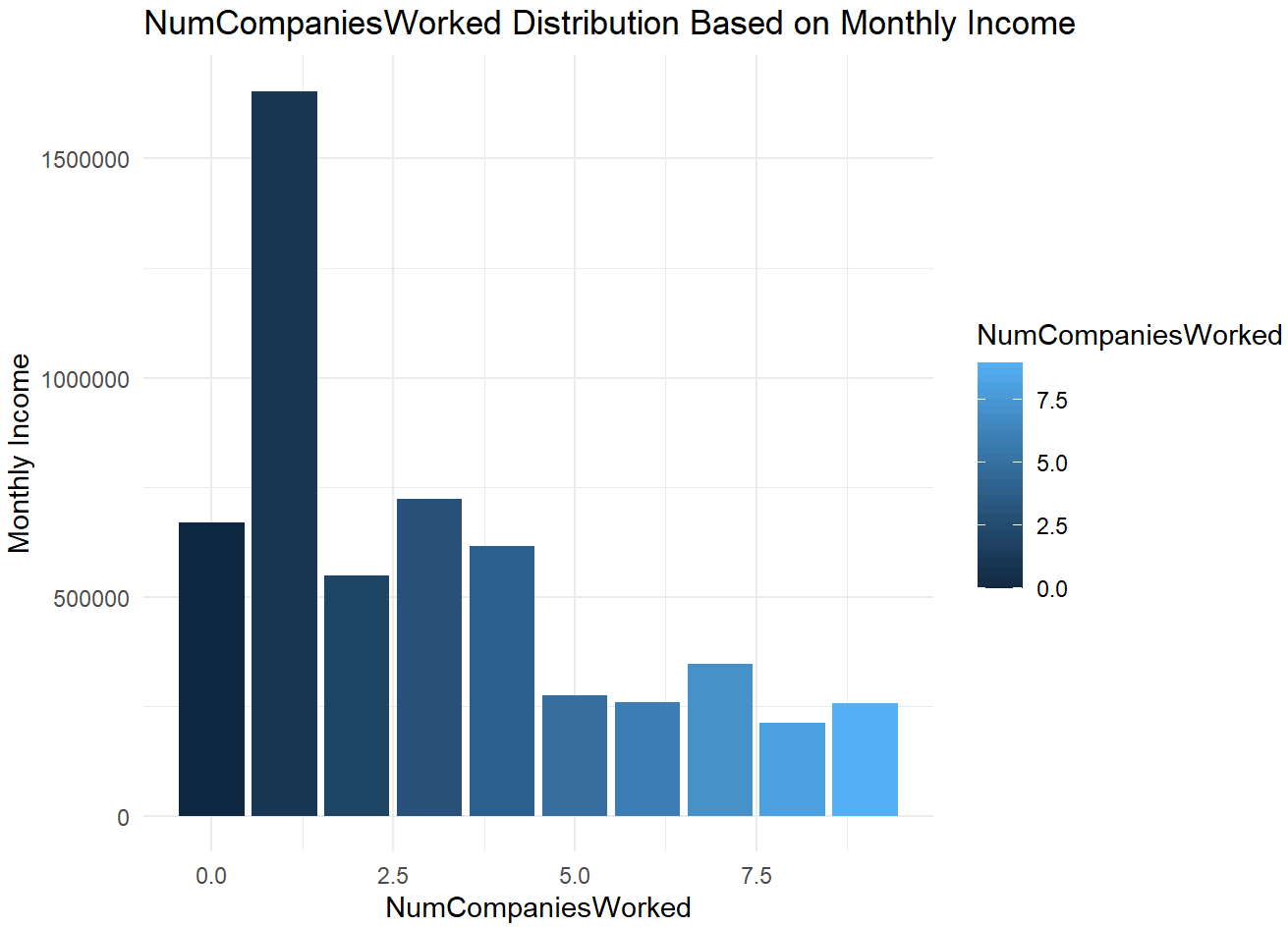

JobLevel Distribution Based on Monthly Income
Monthly Income
1500000
1000000
500000
0
1
2
3
4
5
JobLevel
JobLevel
5
4
3
2
1


JobRole Distribution Based on Monthly Income
Monthly Income
1e+06
5e+05
0e+00
JobRole
JobRole
Healthcare Representative
Human Resources
Laboratory Technician
Manager
Manufacturing Director
Research Director
Research Scientist
Sales Executive
Sales Representative

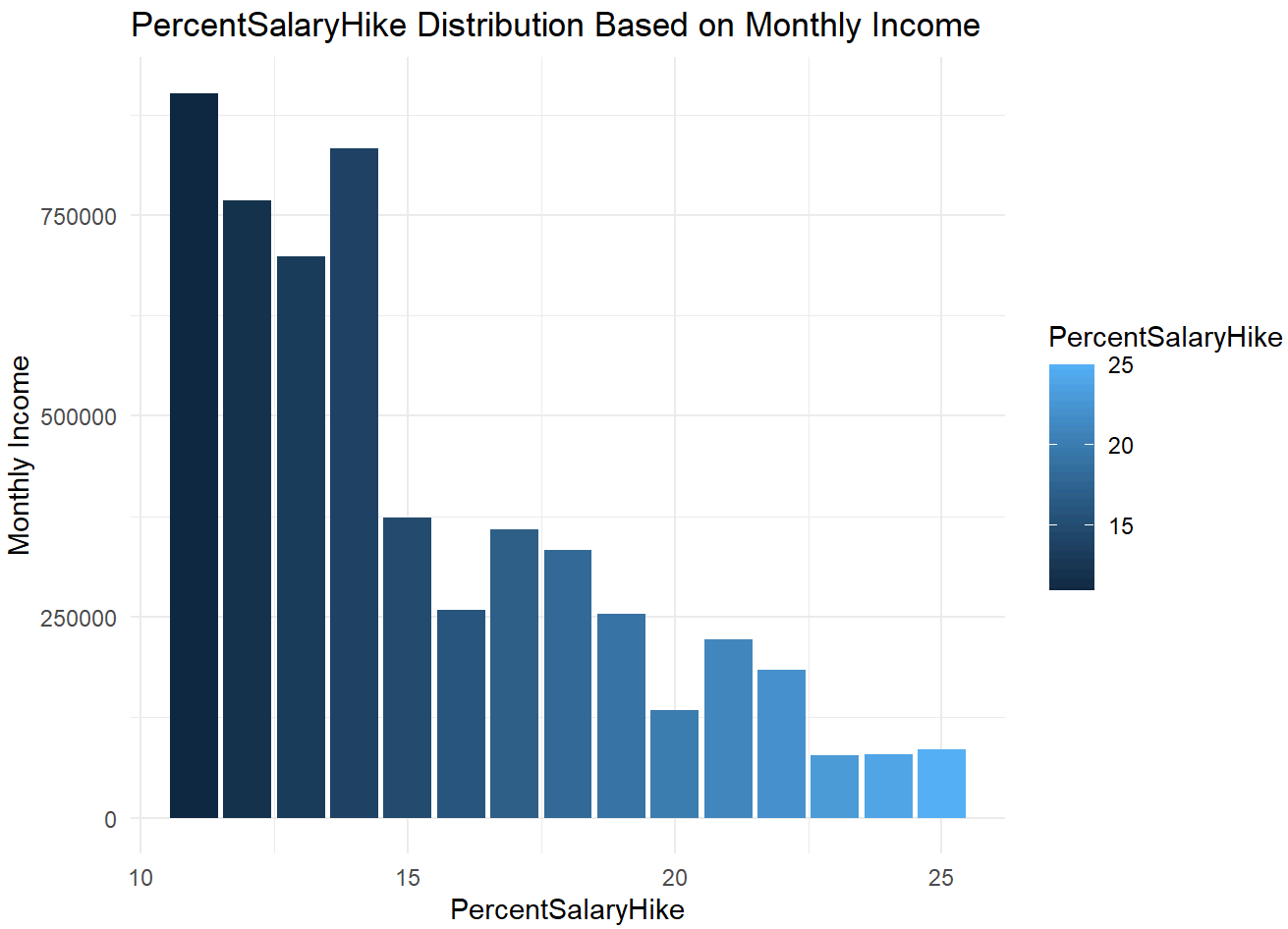

JobSatisfaction Distribution Based on Monthly Income
Monthly Income
1500000
1000000
500000
0
1
2
3
4
JobSatisfaction
JobSatisfaction
4
3
2
1

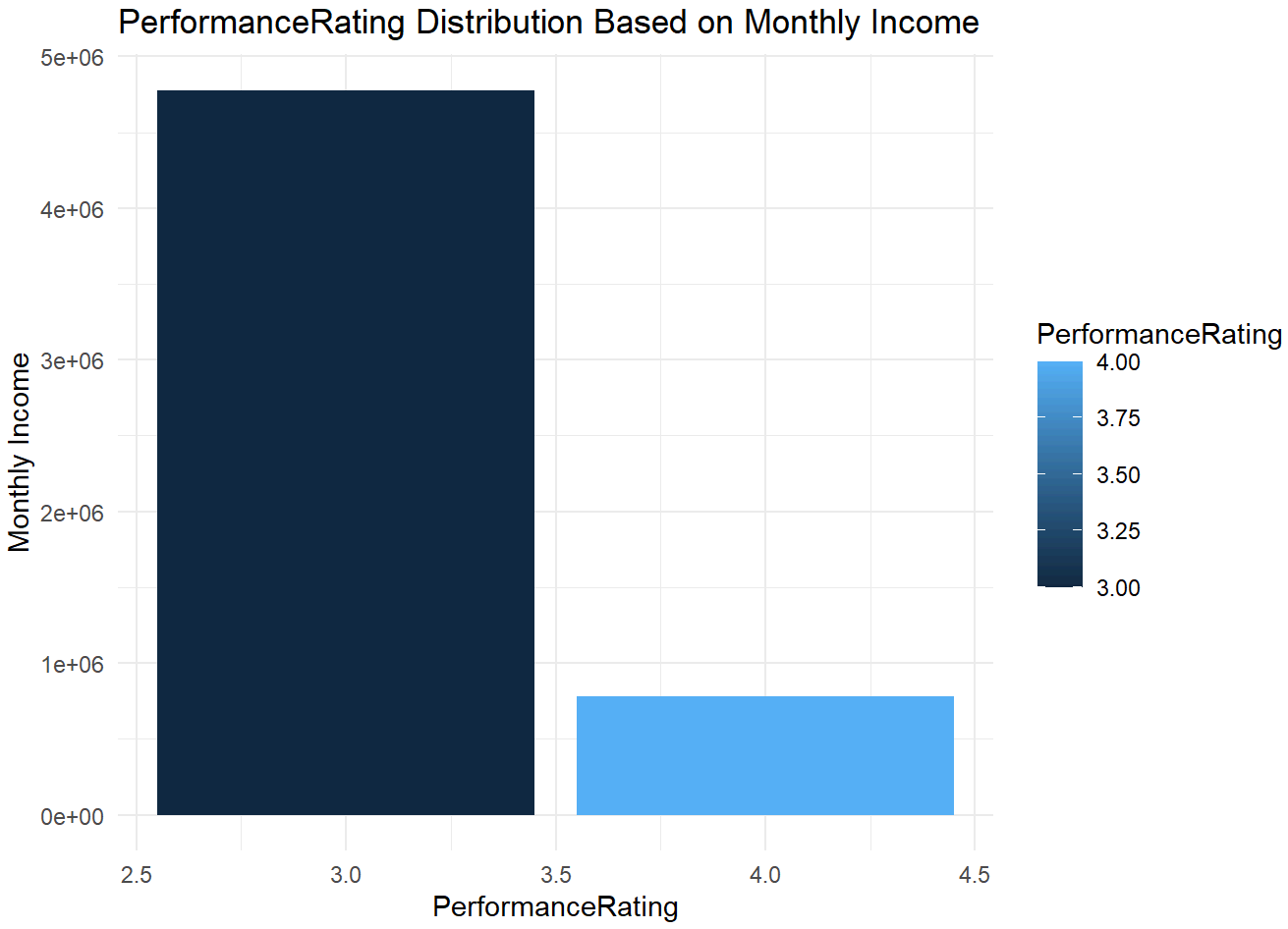

MaritalStatus Distribution Based on Monthly Income
Monthly Income
2e+06
1e+06
0e+00
Divorced
Married
Single
MaritalStatus
MaritalStatus
Divorced
Married
Single

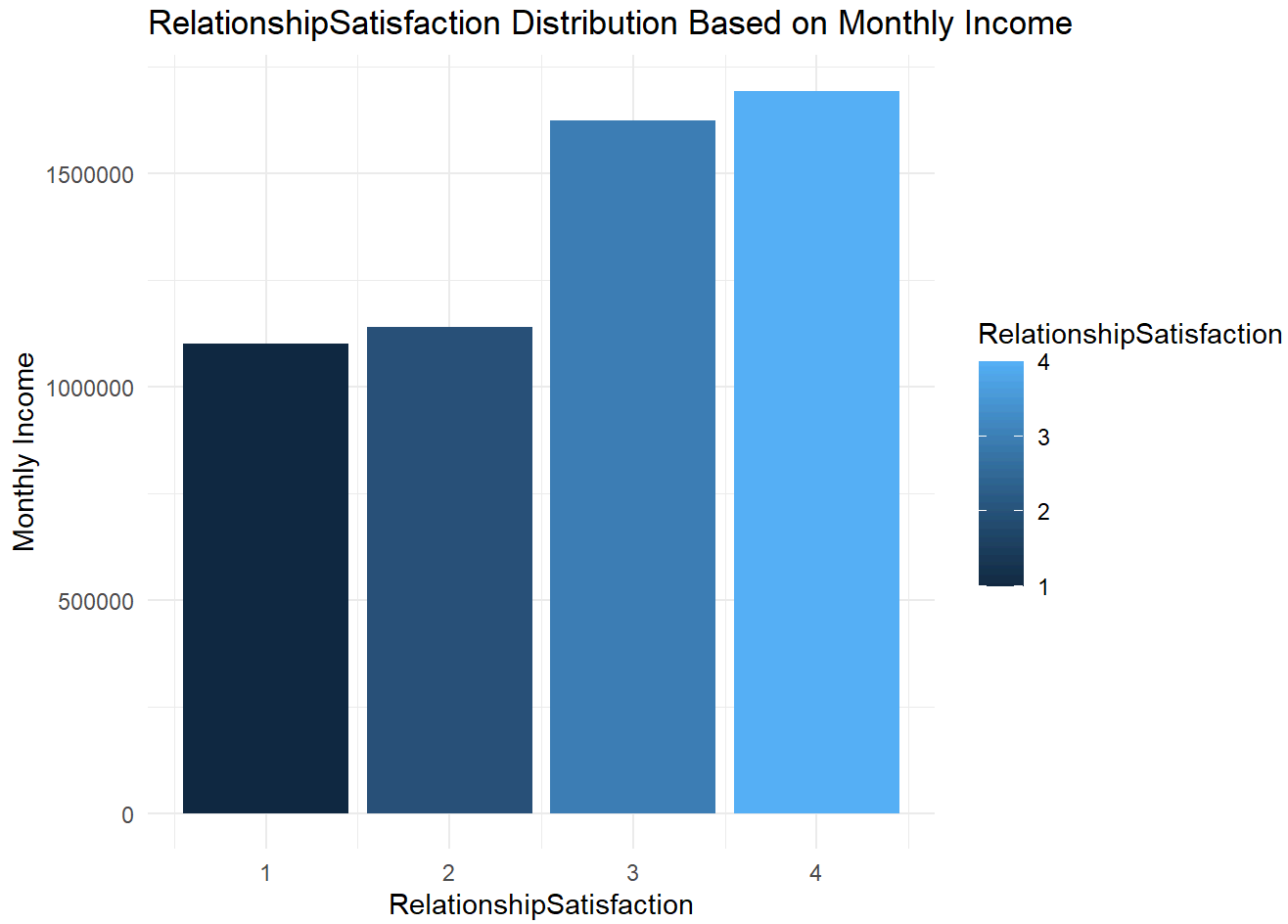

MonthlyRate Distribution Based on Monthly Income
Monthly Income
25000
20000
15000
10000
5000
0
10000
20000
MonthlyRate
MonthlyRate
25000
20000
15000
10000
5000

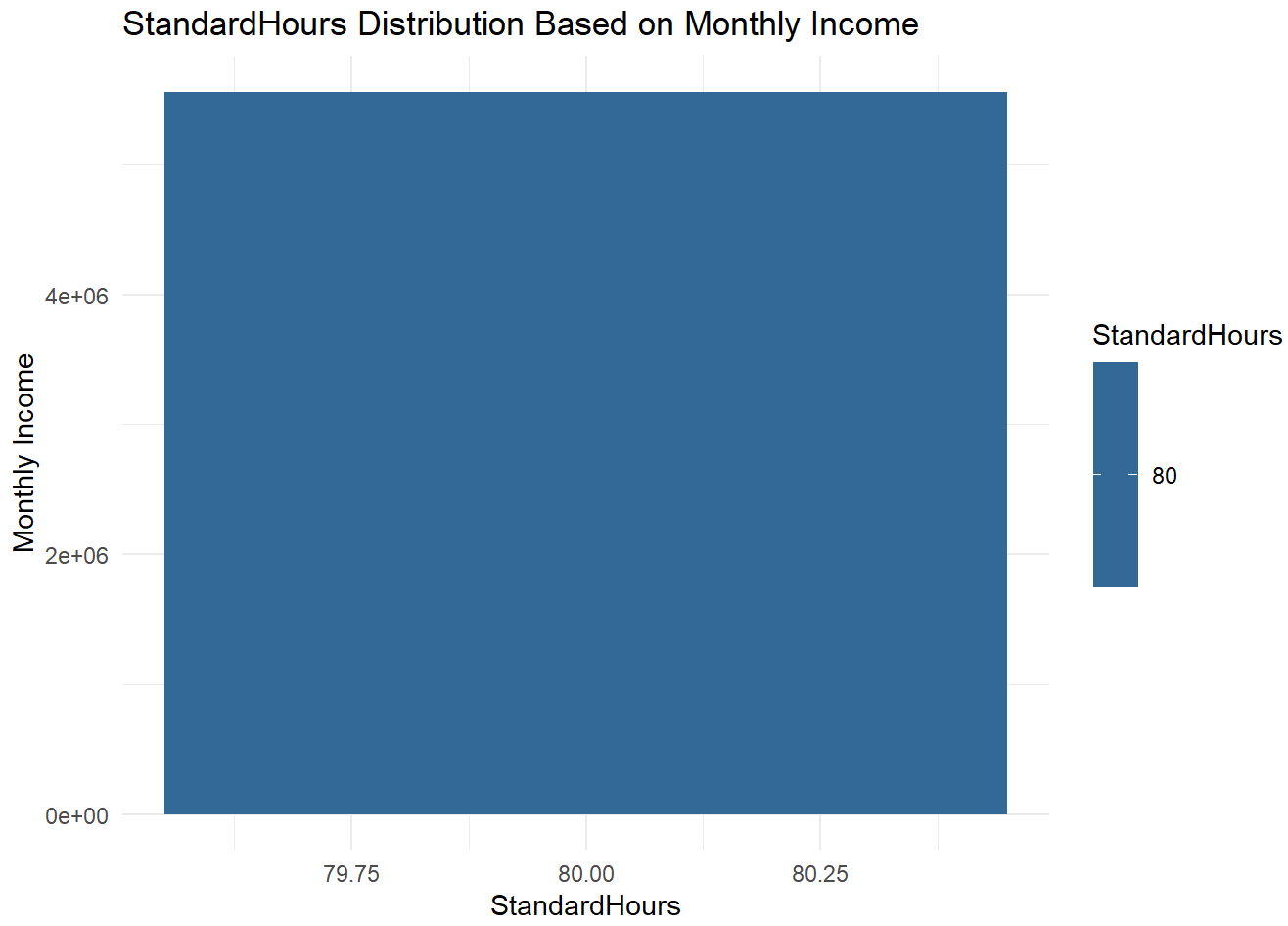

NumCompaniesWorked Distribution Based on Monthly Income
Monthly Income
1500000
1000000
500000
0
0.0
2.5
5.0
7.5
NumCompaniesWorked
NumCompaniesWorked
7.5
5.0
2.5
0.0

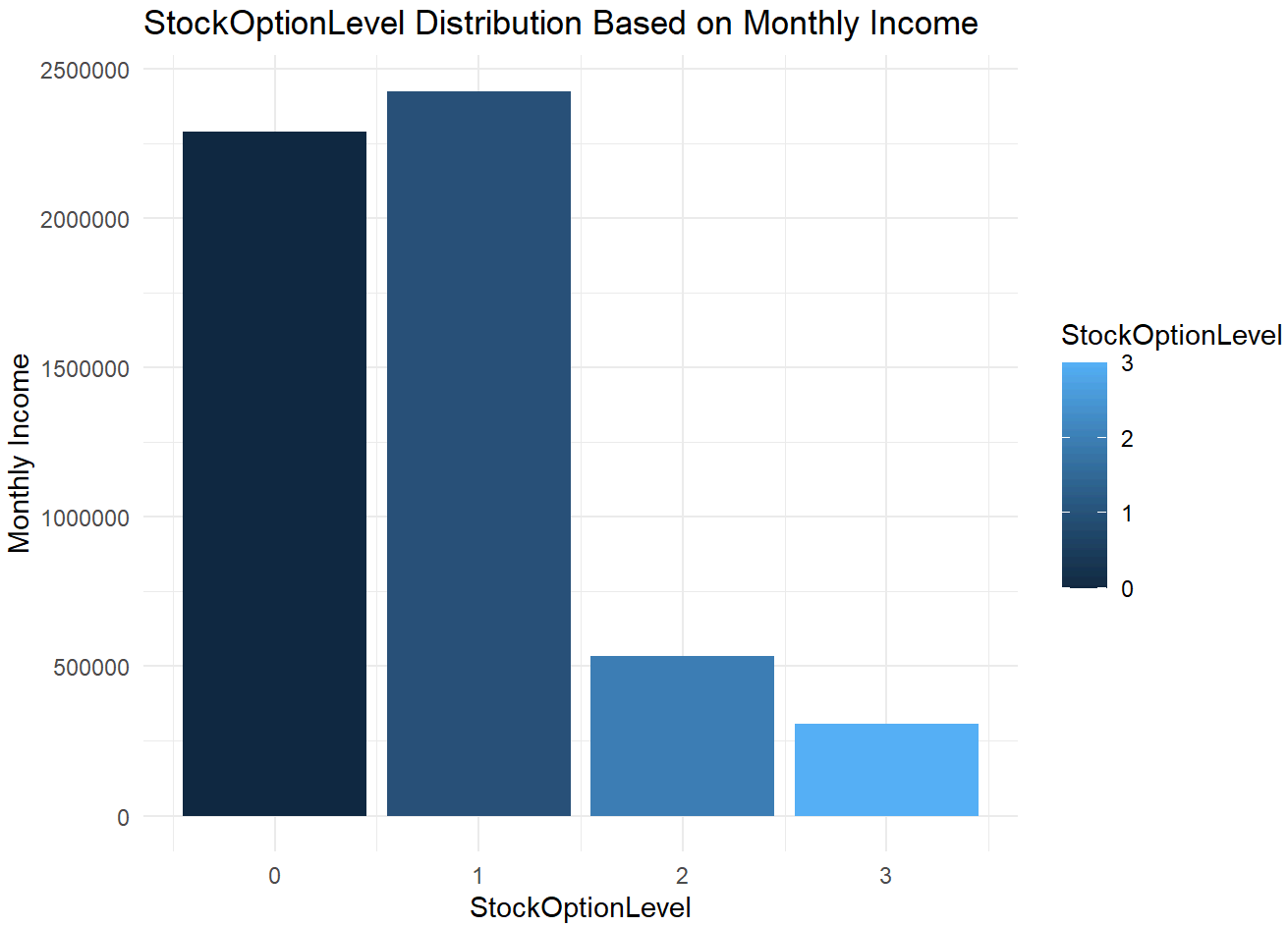

OverTime Distribution Based on Monthly Income
Monthly Income
4e+06
3e+06
2e+06
1e+06
0e+00
No
Yes
OverTime
OverTime
No
Yes

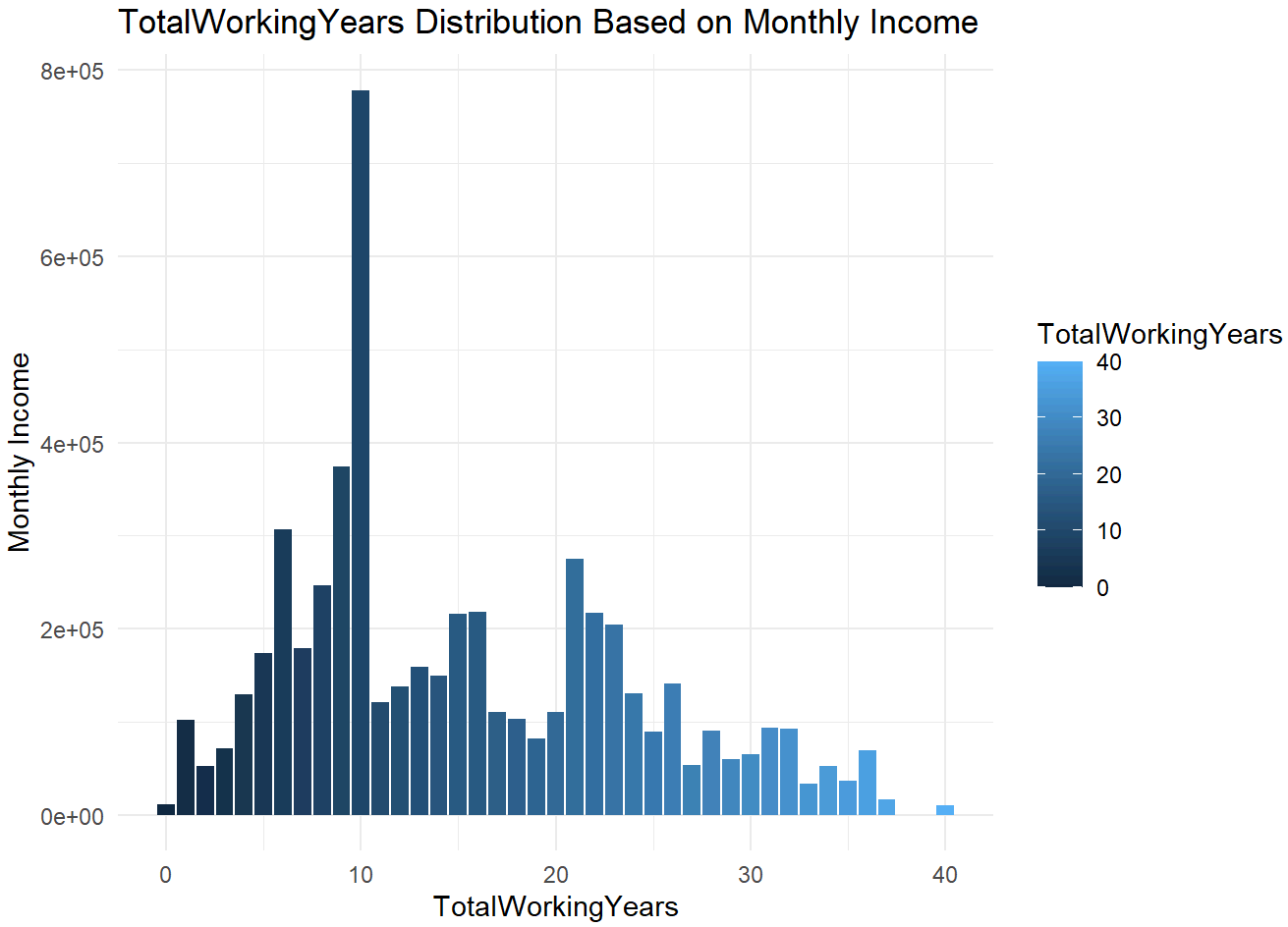

PercentSalaryHike Distribution Based on Monthly Income
Monthly Income
750000
500000
250000
0
10
15
20
25
PercentSalaryHike
PercentSalaryHike
25
20
15


PerformanceRating Distribution Based on Monthly Income
Monthly Income
5e+06
4e+06
3e+06
2e+06
1e+06
0e+00
2.5
3.0
3.5
4.0
4.5
PerformanceRating
PerformanceRating
4.00
3.75
3.50
3.25
3.00

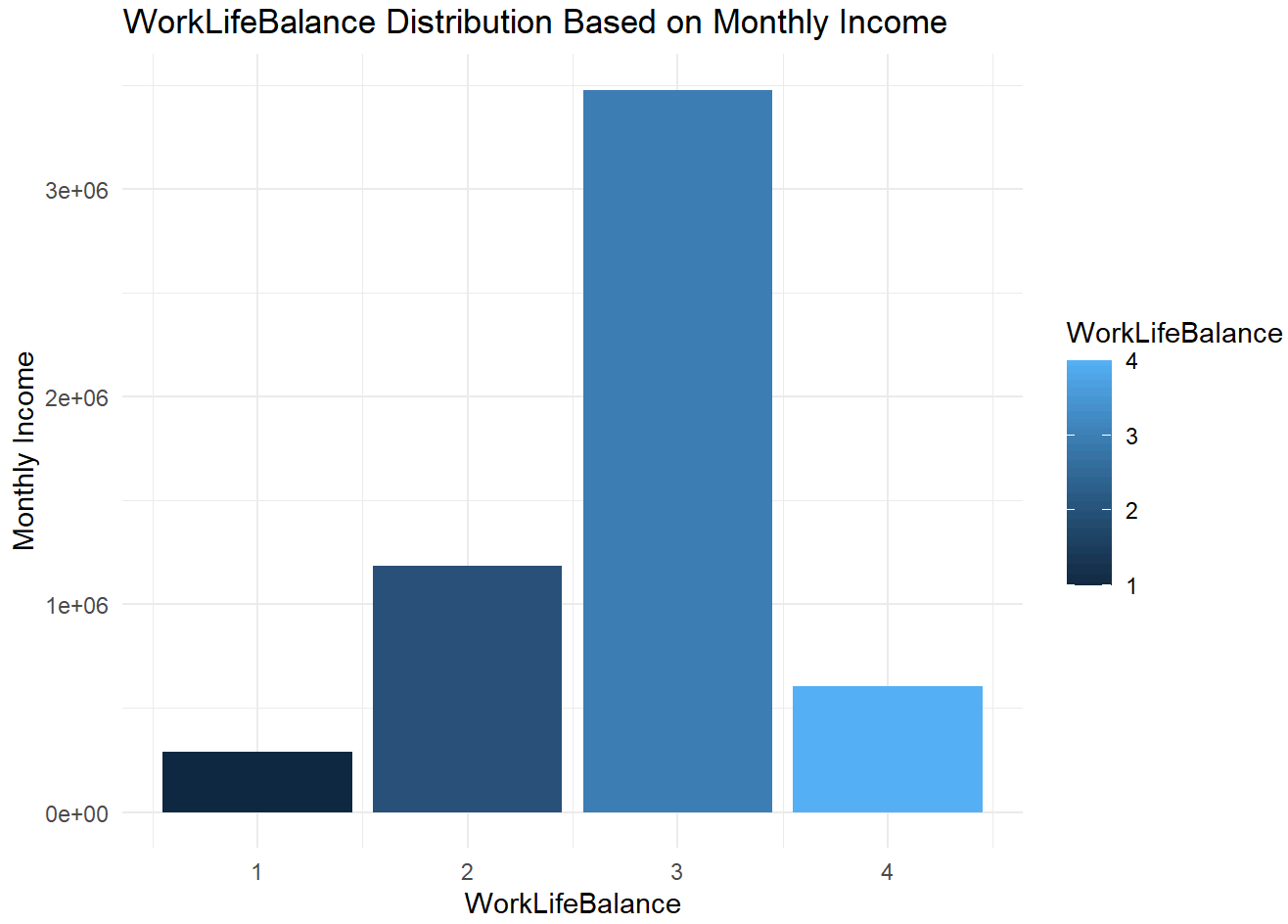

RelationshipSatisfaction Distribution Based on Monthly Income
Monthly Income
1500000
1000000
500000
0
1
2
3
4
RelationshipSatisfaction
RelationshipSatisfaction
4
3
2
1

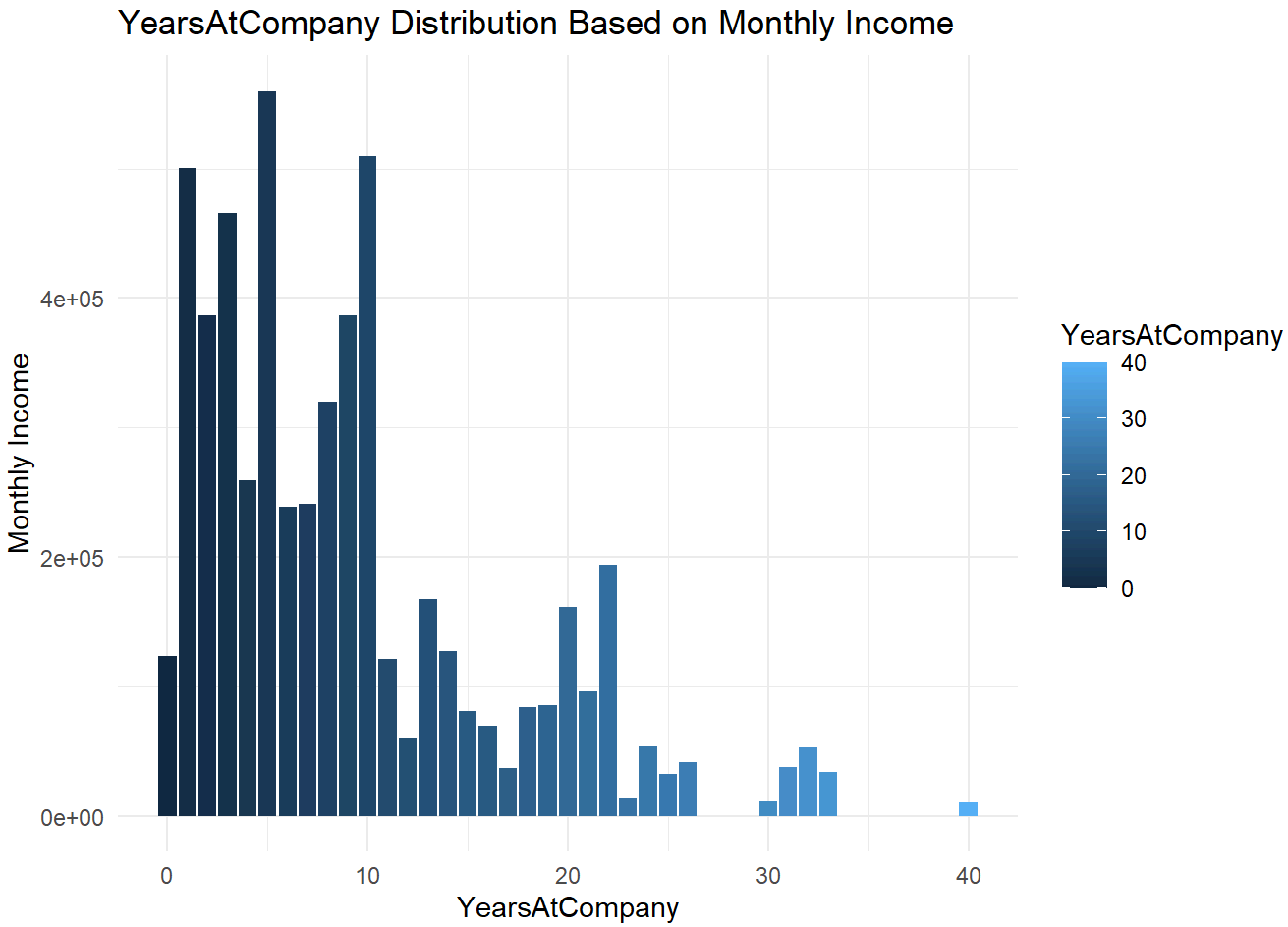

StandardHours Distribution Based on Monthly Income
Monthly Income
4e+06
2e+06
0e+00
79.75
80.00
80.25
StandardHours
StandardHours
80

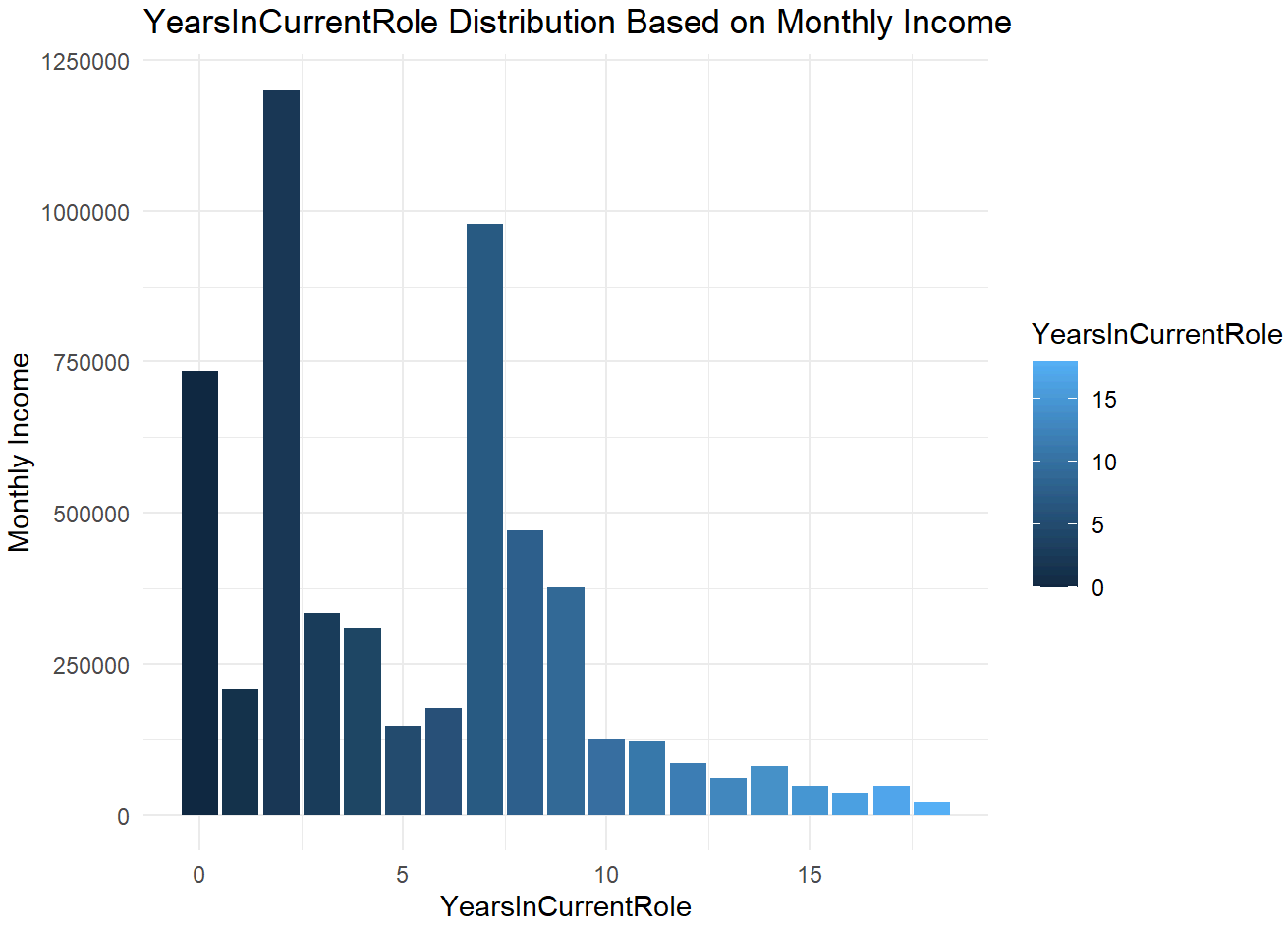

StockOptionLevel Distribution Based on Monthly Income
Monthly Income
2500000
2000000
1500000
1000000
500000
0
0
1
2
3
StockOptionLevel
StockOptionLevel
3
2
1
0

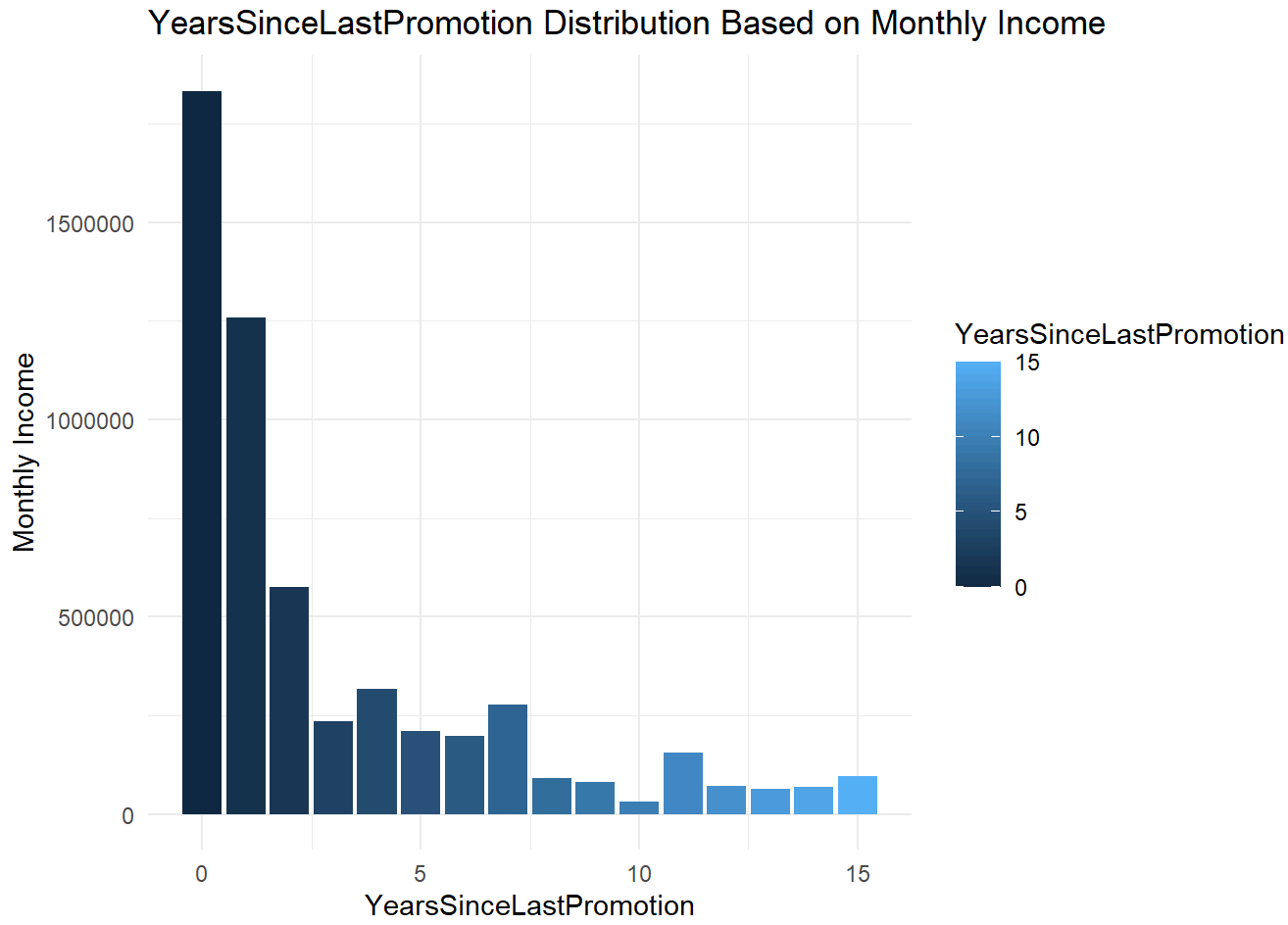

TotalWorkingYears Distribution Based on Monthly Income
Monthly Income
8e+05
6e+05
4e+05
2e+05
0e+00
0
10
20
30
40
TotalWorkingYears
TotalWorkingYears
40
30
20
10
0

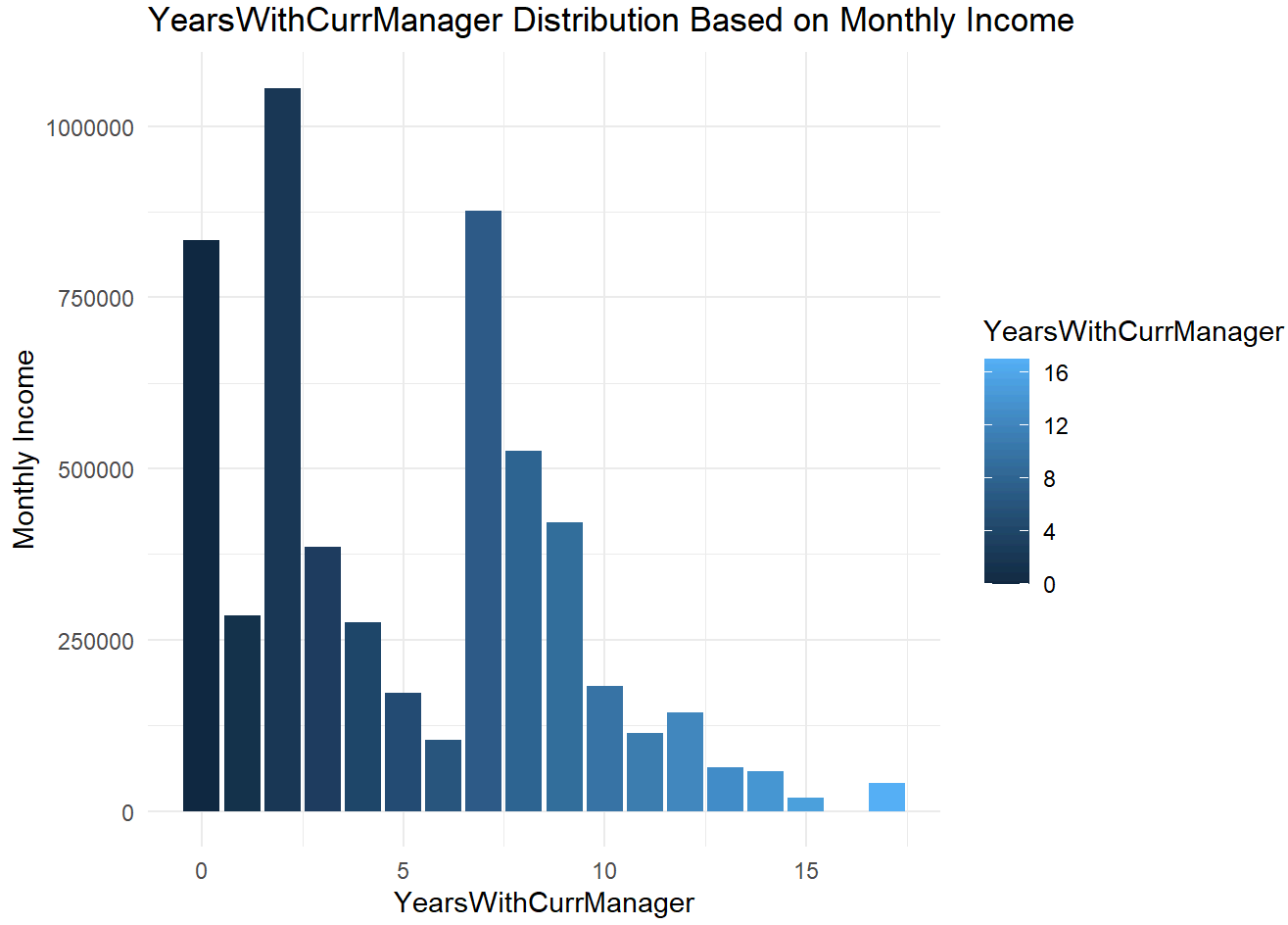

TrainingTimesLastYear Distribution Based on Monthly Income
Monthly Income
2000000
1500000
1000000
500000
0
0
2
4
6
TrainingTimesLastYear
TrainingTimesLastYear
6
5
4
3
2
1
0

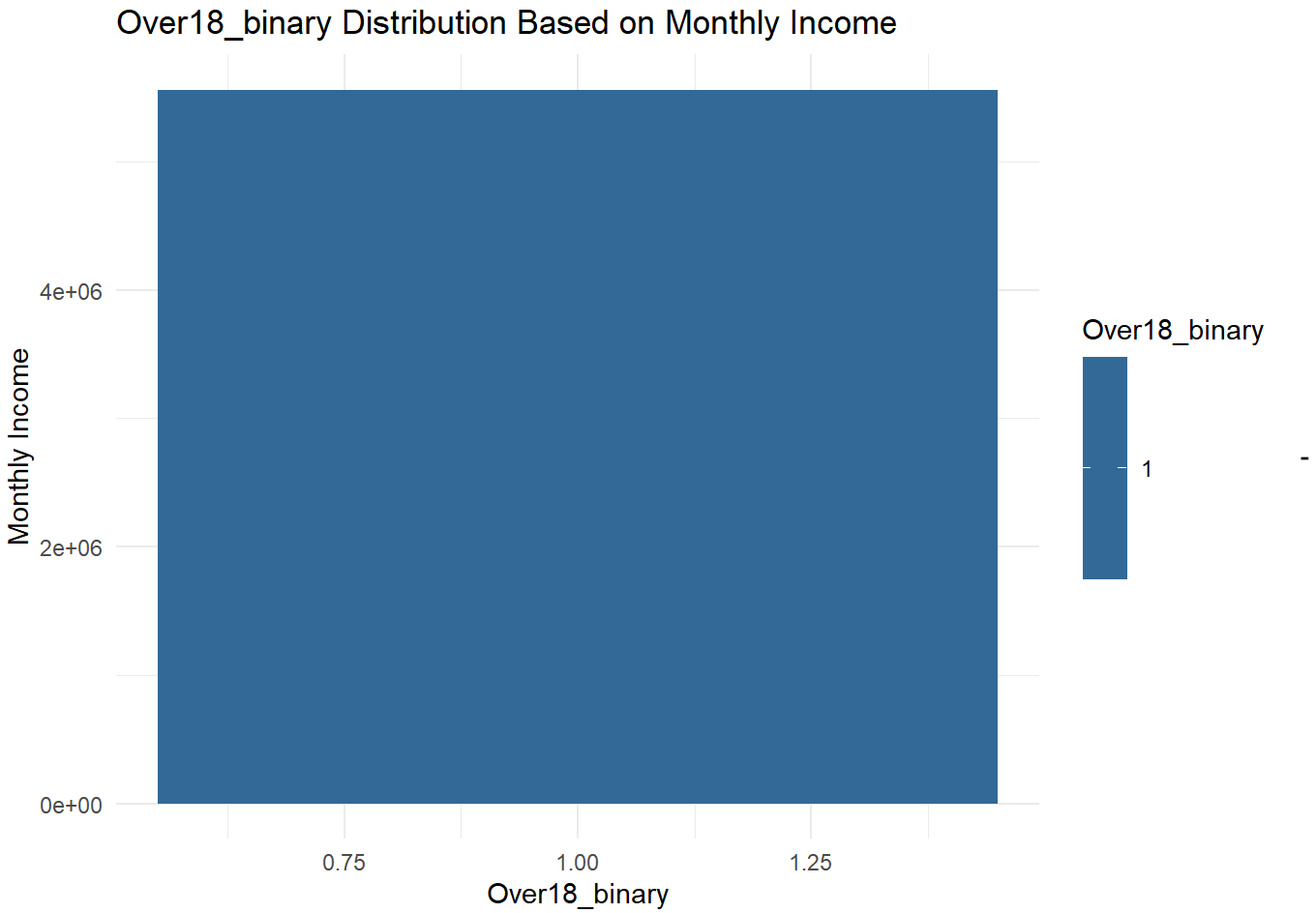

WorkLifeBalance Distribution Based on Monthly Income
Monthly Income
3e+06
2e+06
1e+06
0e+00
1
2
3
4
WorkLifeBalance
WorkLifeBalance
4
3
2
1


YearsAtCompany Distribution Based on Monthly Income
Monthly Income
4e+05
2e+05
0e+00
0
10
20
30
40
YearsAtCompany
YearsAtCompany
40
30
20
10
0


YearsInCurrentRole Distribution Based on Monthly Income
Monthly Income
1250000
1000000
750000
500000
250000
0
0
5
10
15
YearsInCurrentRole
YearsInCurrentRole
15
10
5
0


YearsSinceLastPromotion Distribution Based on Monthly Income
Monthly Income
1500000
1000000
500000
0
0
5
10
15
YearsSinceLastPromotion
YearsSinceLastPromotion
15
10
5
0


YearsWithCurrManager Distribution Based on Monthly Income
Monthly Income
1000000
750000
500000
250000
0
0
5
10
15
YearsWithCurrManager
YearsWithCurrManager
16
12
8
4
0



Employees in Job Level 2 seems to be paid the best - Sales Executive seems to be the highest paid in the company. - The employees with around 10 years seem to be paid the highest. - Employees with Job involvement rate of 3 are the highest paid in the company. - The employees with Stock option level 1 seem to have the highest monthly salary. - Human Resources Employees seem to be the lowest paid

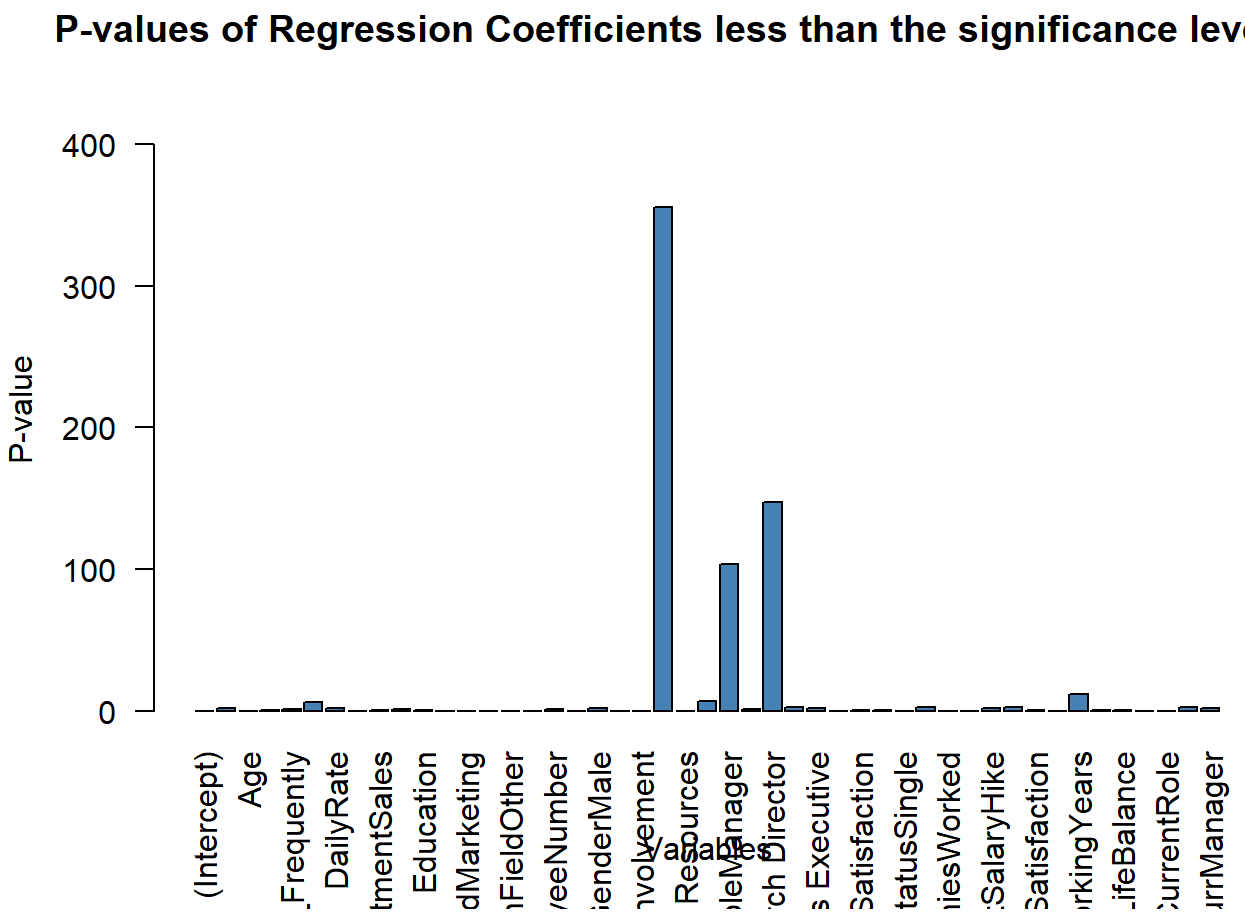
LOOKING AT THE PVALUE DISTRIBUTIONS FOR THE MONTHLY INCOME

Looking at how each variable in the model, significantly impacts our response variable (MonthlyIncome)

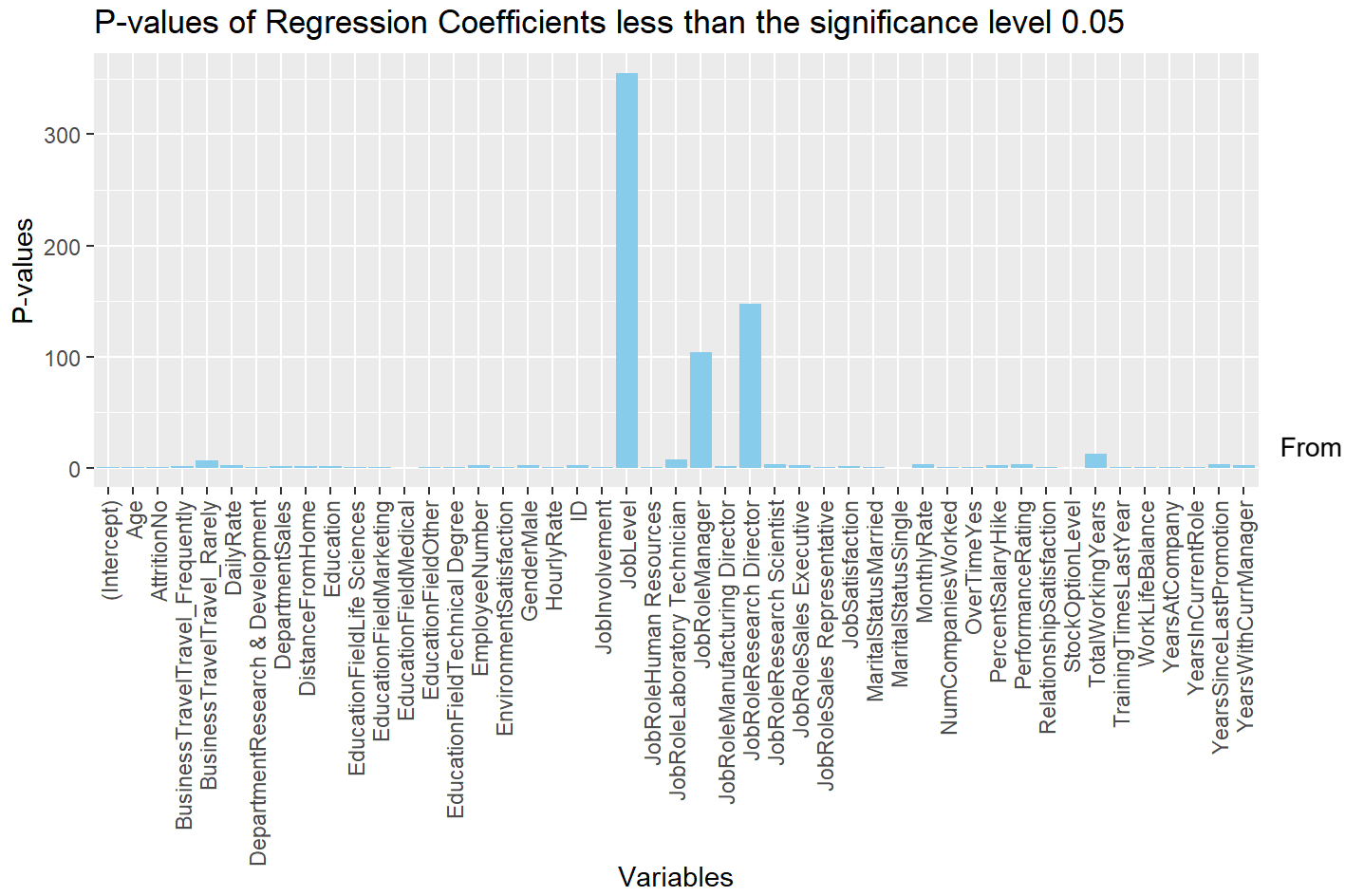
```
model <-lm(MonthlyIncome ~ ., data = Talent_Train)

# Extract variable names
variable_names <- rownames(summary(model)$coefficients)

# getting the  p-values from the model3
p_values <- summary(model)$coefficients[, 4]  # Assuming p-values are in the 4th column of the s
ummary table
p_values <- data.frame(p_values)$p_values

df <- data.frame(variable_names, p_values) #combining the pvalues and variable names into a data
frame

df <- df[!df$p_value == 0 , ] #removing varaiables with pvalue = 0
df$p_values <- log(df$p_values)  * -1

# Rank p-values in the dataset from max to min
df$rank <- rank(-df$p_value)

sorted_df <- df[order(df$rank), ]

sorted_df
```

```
##                      variable_names     p_values rank
## 22                         JobLevel 355.50839712    1
## 27          JobRoleResearch Director 147.27554033    2
## 25                   JobRoleManager 103.71500561    3
## 41                TotalWorkingYears  12.35695622    4
## 24     JobRoleLaboratory Technician   7.38850243    5
## 6       BusinessTravelTravel_Rarely   6.21230807    6
## 28        JobRoleResearch Scientist   3.19931378    7
## 46          YearsSinceLastPromotion   3.11440399    8
## 38                PerformanceRating   3.09295539    9
## 34                      MonthlyRate   2.73419587   10
## 37                PercentSalaryHike   2.27934762   11
## 47             YearsWithCurrManager   2.23296225   12
## 7                        DailyRate   2.18870241   13
## 2                               ID   2.18290967   14
## 19                       GenderMale   1.96815169   15
## 29            JobRoleSales Executive   1.95581659   16
## 17                    EmployeeNumber   1.86259693   17
## 10                 DistanceFromHome   1.75205868   18
## 5   BusinessTravelTravel_Frequently   1.70132764   19
## 26    JobRoleManufacturing Director   1.25002804   20
## 11                         Education   1.15802948   21
## 31                  JobSatisfaction   1.11694526   22
## 9                   DepartmentSales   1.05964131   23
## 42            TrainingTimesLastYear   0.90854367   24
## 4                       AttritionNo   0.81932519   25
## 43                 WorkLifeBalance   0.69058238   26
## 32             MaritalStatusMarried   0.68617702   27
## 39         RelationshipSatisfaction   0.47260772   28
## 23           JobRoleHuman Resources   0.37010298   29
## 12      EducationFieldLife Sciences   0.33652451   30
## 13          EducationFieldMarketing   0.32151404   31
## 35               NumCompaniesWorked   0.28952101   32
## 21                   JobInvolvement   0.28409054   33
## 8  DepartmentResearch & Development   0.22832410   34
## 3                              Age   0.22300668   35
## 16   EducationFieldTechnical Degree   0.22167042   36
## 15              EducationFieldOther   0.20252776   37
## 30      JobRoleSales Representative   0.18910119   38
## 44                  YearsAtCompany   0.17325279   39
## 45               YearsInCurrentRole   0.17128149   40
## 20                      HourlyRate   0.16959936   41
## 1                       (Intercept)   0.16581060   42
## 18          EnvironmentSatisfaction   0.15987311   43
## 36                      OverTimeYes   0.14065683   44
## 14            EducationFieldMedical   0.07431878   45
## 33              MaritalStatusSingle   0.06830232   46
## 40                StockOptionLevel   0.03905786   47
```

```
barplot(df$p_values,
        main = "P-values of Regression Coefficients less than the significance level 0.05",
        xlab = "Variables",
        ylab = "P-value",
        names.arg = df$variable_names,
        las = 2,  # Rotate x-axis labels vertically for better readability
        col = "steelblue",  # Set color of bars
        ylim = c(exp(0.05) * -1, max(sorted_df$p_values) * 1.2)  # Set ylim from the significanc
e level to the max p-values

)
```




```
library(ggplot2)
ggplot(sorted_df,aes(variable_names,p_values, fill = ifelse(p_values > (exp(0.05) * -1), "Positi
ve", "Negative"))) + #filtering just the highly significant p-values
  geom_bar(stat="identity", fill = "skyblue") +
  #geom_text(aes(label = variable_names), vjust = -0.5) +  # Add text labels on top of bars
  scale_fill_manual(values = c("Positive" = "skyblue", "Negative" = "salmon")) +
  labs(x = "Variables", y = "P-values", title = "P-values of Regression Coefficients less than t
he significance level 0.05") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))  # Rotate x-axis labels fo
r better readability
```



From

the plot above, we can see that the top 5 highly significant values with respect to the response variable y in terms of its relative most significant to least significant variables are JobLevel, JobRole, TotalWorkingYears, BusinessTravel, YearsSinceLastPromotion

USING FORWARD SELECTION TO GET THE BEST REGRESSION MODEL WITH THE LOWEST RMSE FOR THE MONTHLY INCOME

My plan is to iterate through finding the logistic regression with the min rmse value when interacting our response variable with various predictors. Then ill pick the variable with the min rmse. Ill add my min rmse variable to the model, then use the next iteration to get the min rmse when add another variable with our prior variable, then interact them with our response variable. Then i repeat the steps again to get my optimum model.

GETTING THE FIRST VARIABLE

Im looking for the perfect (MonhlyIncome ~ dependent variable) combination to get the minimum RMSE

```
# FORWARD SELECTION # 1
set.seed(21)
vars <- names(Talent_Train)
vars <- vars[vars != "MonthlyIncome"] #iterating thru variables that are not "MonthlyIncome"
vars <- vars[vars != "Attrition"] #iterating thru variables that are not "Attrition"
num_vars <- length(vars)
var_rmse <- data.frame("vars" = vars)
num_folds <- 10
for (j in 1:num_vars) {
  var <- vars[j]
  #print(var)
  folds <- createFolds(Talent_Train$MonthlyIncome, k = num_folds)
  rmse_scores <- numeric(num_folds)
  for (i in 1:num_folds) {
    #using the train data
    train_indices <- unlist(folds[-i])
    test_indices <- unlist(folds[i])
    train <- Talent_Train[train_indices, ]
    test <- Talent_Train[test_indices, ]
    form <- as.formula(paste("MonthlyIncome ~ ",var,sep=""))
    model <-lm(form, data = train) #fit a linear regression model
    predictions <- predict(model, newdata = test) #Make Predictiions
    residuals <- predictions - test$MonthlyIncome # Calculate residuals
    rmse <- sqrt(mean(residuals^2)) # Compute RMSE
    rmse_scores[i] <- rmse
  }
  var_rmse$rmse[var_rmse$var == var] <- mean(rmse_scores)
}

min_rmse = min(var_rmse$rmse) #finding the max_rmse
min_rmse_variable <- var_rmse$vars[which.min(var_rmse$rmse)]

#printing out rresult
cat("optimum Variable is ", min_rmse_variable, " with a minimum rmse of ", min_rmse)
```

```
## optimum Variable is  JobLevel  with a minimum rmse of  1409.276
```

optimum Variable is JobLevel with a minimum rmse of 1409.276

GETTING THE SECOND VARIABLE

Next ill look for the optimumm variable to add to our regression model (Monthly ~ JobLevel) in order to provide the minimum rmse

```
# FORWARD SELECTION # 2
set.seed(21)
vars <- names(Talent_Train)
vars <- vars[vars != "MonthlyIncome"] #iterating thru variables that are not "MonthlyIncome"
vars <- vars[vars != "Attrition"] #iterating thru variables that are not "Attrition"
vars <- vars[vars != "JobLevel"] #iterating thru variables that are not "JobeLevel"
num_vars <- length(vars)
var_rmse <- data.frame("vars" = vars)
num_folds <- 10
for (j in 1:num_vars) {
  var <- vars[j]
  #print(var)
  folds <- createFolds(Talent_Train$MonthlyIncome, k = num_folds)
  rmse_scores <- numeric(num_folds)
  for (i in 1:num_folds) {
    #using the train data
    train_indices <- unlist(folds[-i])
    test_indices <- unlist(folds[i])
    train <- Talent_Train[train_indices, ]
    test <- Talent_Train[test_indices, ]
    form <- as.formula(paste("MonthlyIncome ~ JobLevel +  ",var,sep=""))
    model <-lm(form, data = train) #fit a linear regression model
    predictions <- predict(model, newdata = test) #Make Predictiions
    residuals <- predictions - test$MonthlyIncome # Calculate residuals
    rmse <- sqrt(mean(residuals^2)) # Compute RMSE
    rmse_scores[i] <- rmse
  }
  var_rmse$rmse[var_rmse$var == var] <- mean(rmse_scores)
}

min_rmse = min(var_rmse$rmse) #finding the max_rmse
min_rmse_variable <- var_rmse$vars[which.min(var_rmse$rmse)]

cat("optimum Variable is ", min_rmse_variable, " with a minimum rmse of ", min_rmse)
```

```
## optimum Variable is  JobRole  with a minimum rmse of  1085.182
```

optimum Variable is JobRole with a minimum rmse of 1085.182

GETTING THE THIRD VARIABLE

Next ill look for the optimumm variable to add to our regression model (Monthly ~ JobLevel + JobRole) in order to provide the minimum rmse

```
# FORWARD SELECTION # 3
set.seed(21)
vars <- names(Talent_Train)
vars <- vars[vars != "MonthlyIncome"] #iterating thru variables that are not "MonthlyIncome"
vars <- vars[vars != "Attrition"] #iterating thru variables that are not "Attrition"
vars <- vars[vars != "JobLevel"] #iterating thru variables that are not "JobeLevel"
vars <- vars[vars != "JobRole"] #iterating thru variables that are not "JobRole"
num_vars <- length(vars)
var_rmse <- data.frame("vars" = vars)
num_folds <- 10
for (j in 1:num_vars) {
  var <- vars[j]
  #print(var)
  folds <- createFolds(Talent_Train$MonthlyIncome, k = num_folds)
  rmse_scores <- numeric(num_folds)
  for (i in 1:num_folds) {
    #using the train data
    train_indices <- unlist(folds[-i])
    test_indices <- unlist(folds[i])
    train <- Talent_Train[train_indices, ]
    test <- Talent_Train[test_indices, ]
    form <- as.formula(paste("MonthlyIncome ~ JobLevel +  JobRole + ",var,sep=""))
    model <-lm(form, data = train) #fit a linear regression model
    predictions <- predict(model, newdata = test) #Make Predictiions
    residuals <- predictions - test$MonthlyIncome # Calculate residuals
    rmse <- sqrt(mean(residuals^2)) # Compute RMSE
    rmse_scores[i] <- rmse
  }
  var_rmse$rmse[var_rmse$var == var] <- mean(rmse_scores)
}

min_rmse = min(var_rmse$rmse) #finding the max_rmse
min_rmse_variable <- var_rmse$vars[which.min(var_rmse$rmse)]

cat("optimum Variable is ", min_rmse_variable, " with a minimum rmse of ", min_rmse)
```

```
## optimum Variable is  TotalWorkingYears  with a minimum rmse of  1061.89
```

optimum Variable is TotalWorkingYears with a minimum rmse of 1061.89

TESTING THE RMSE WITH THE TEST DATASET

Next, ill test our optimum model, wit out dataset, splitting the dataset as test and train with a 70 - 30 split respectively.

```
#splitting 70% - 30%
trainIndices = sample(seq(1:length(Talent_Train$Age)),round(.7*length(Talent_Train$MonthlyIncom
e))) #split is 70% and 30%
train = Talent_Train[trainIndices,] #train datset
test = Talent_Train[-trainIndices,] #test dataset

# Fit the linear regression model
model <- lm(MonthlyIncome ~ JobLevel + JobRole + TotalWorkingYears, data = train)

# Make predictions
predictions <- predict(model, newdata = test)

# Calculate residuals
residuals <- predictions - test$MonthlyIncome

# Compute RMSE
rmse <- sqrt(mean(residuals^2))

# Print RMSE
print(rmse)
```

```
## [1] 975.667
```

```
test$predictions <- predictions

# Generation our regression for the test dataset
new_data <- test %>% select(predictions, ID)

new_data <- new_data[order(new_data$ID), ]

# Specify the file path where you want to save the file
file_path <- "Case2PredictionsOwolabi Salary.csv"

# Write the dataset to a CSV file
write.csv(new_data, file = file_path, row.names = FALSE)

# Check if the file is created
file.exists(file_path)
```

```
## [1] TRUE
```

the RMSE is 1091.37, which is far lower than our expected RMSE of 3000. Hence proving this is an optimum dataset.

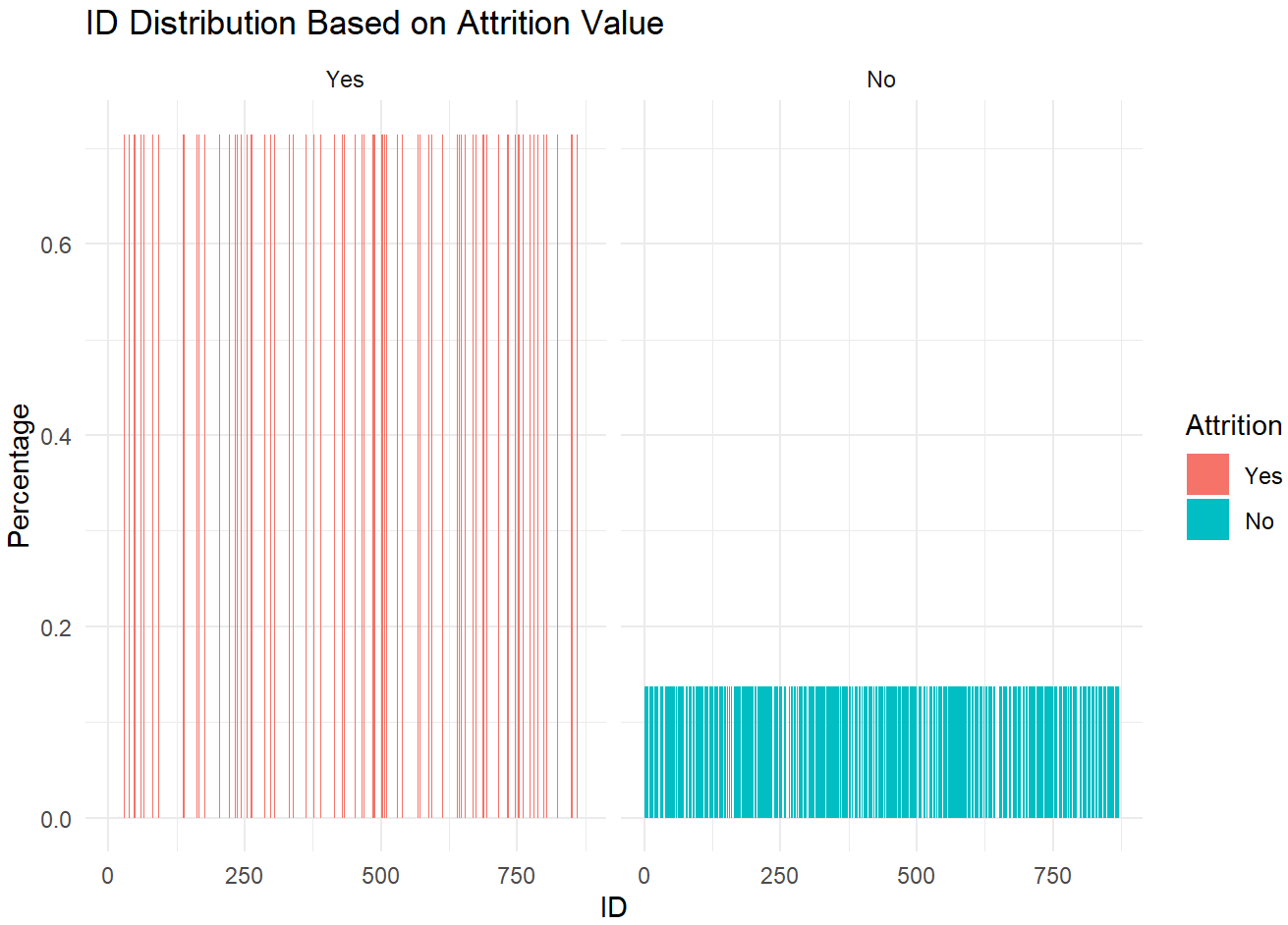
PREDICTION MODEL WITH ATTRITION

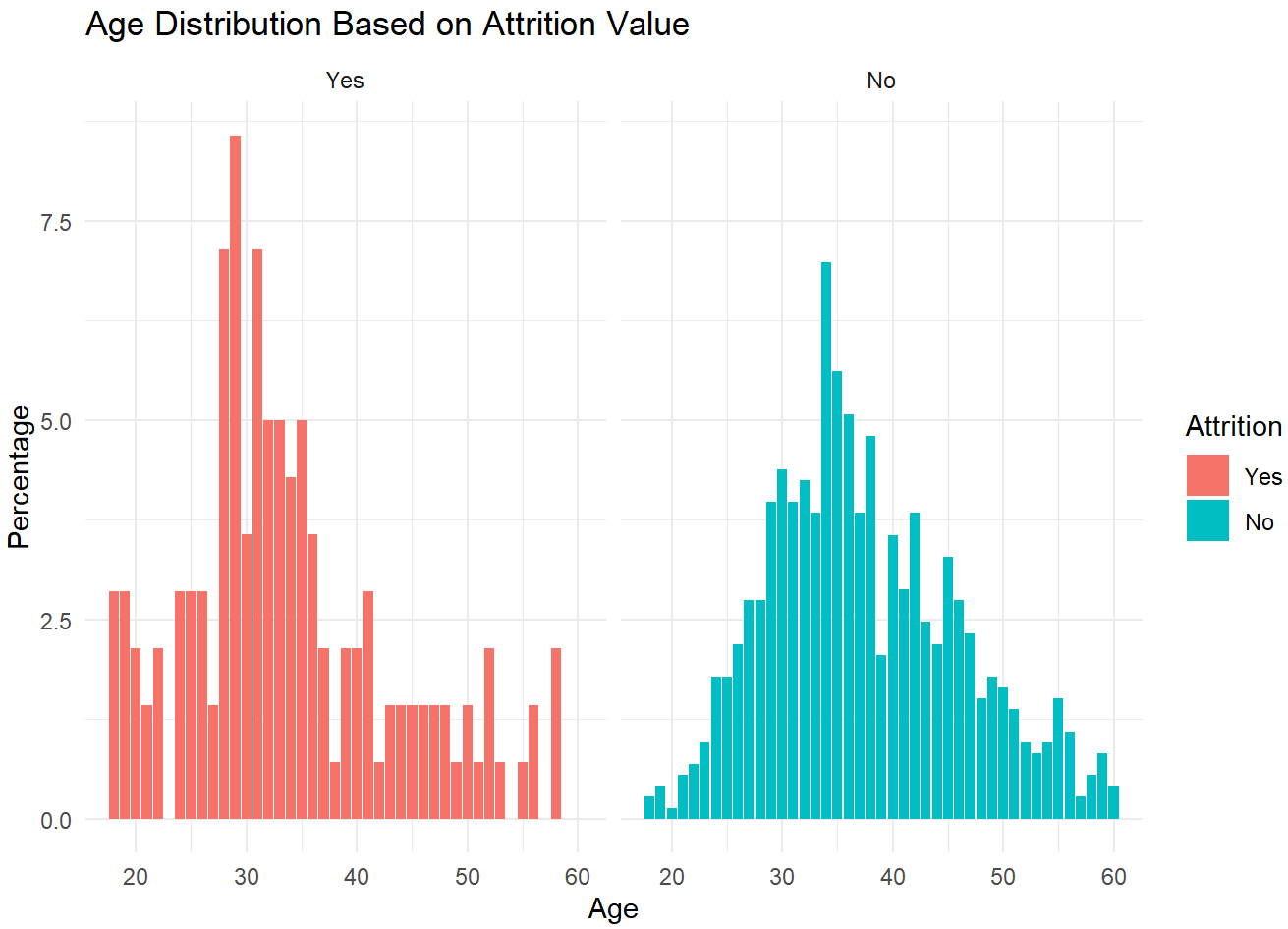
Next, i will try finding the best variables for our predictive classification model with Attrition as ou response variable

VISUALIZING THE CATEGORICAI VARIABLEs INTERACTIONS WITH THE RESPONSE VARIABLE (ATTRITION)

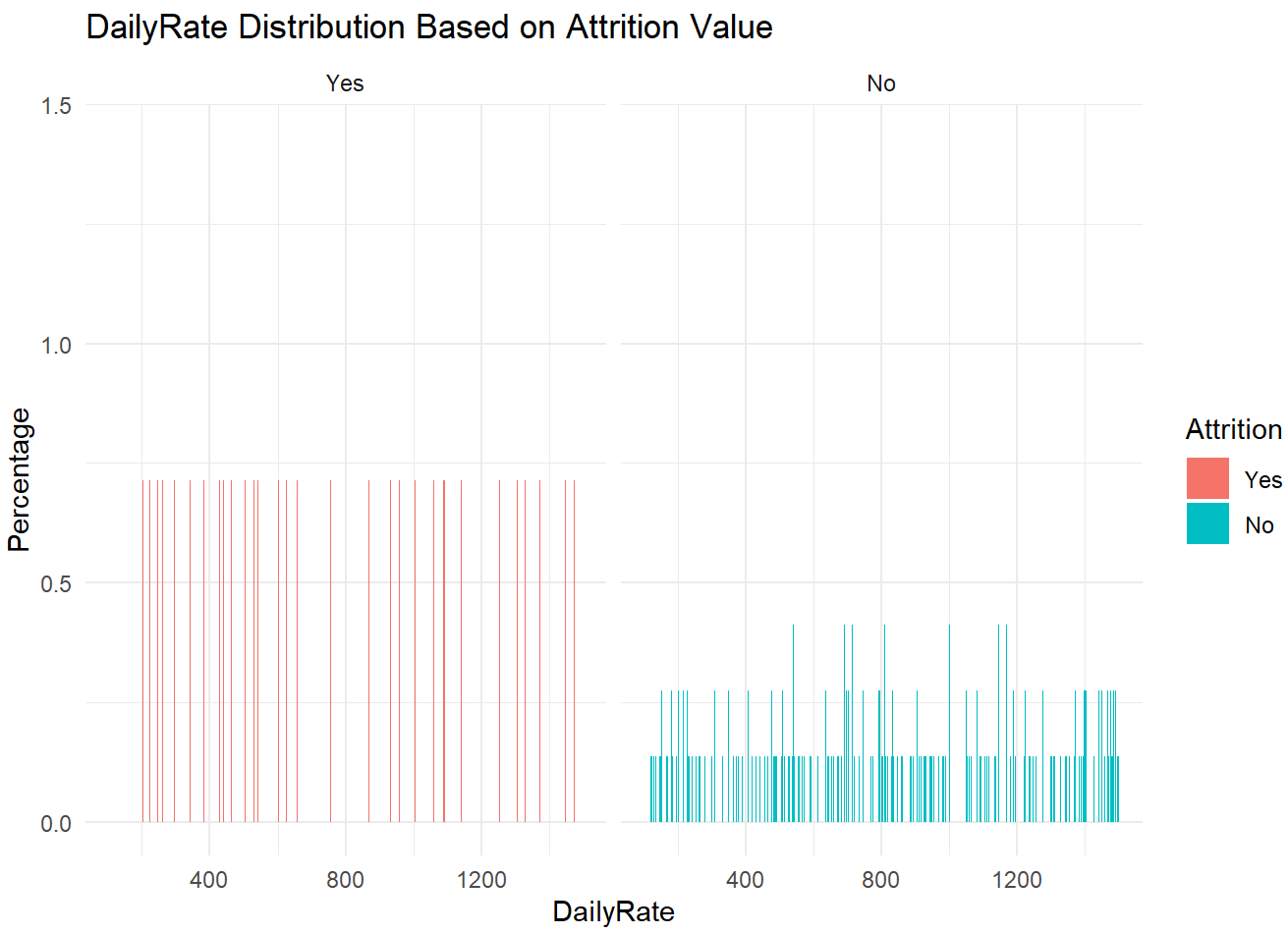
Looking at the Exploratory data analysis to visualize the relation ship between the attrition rate and other dependent variables

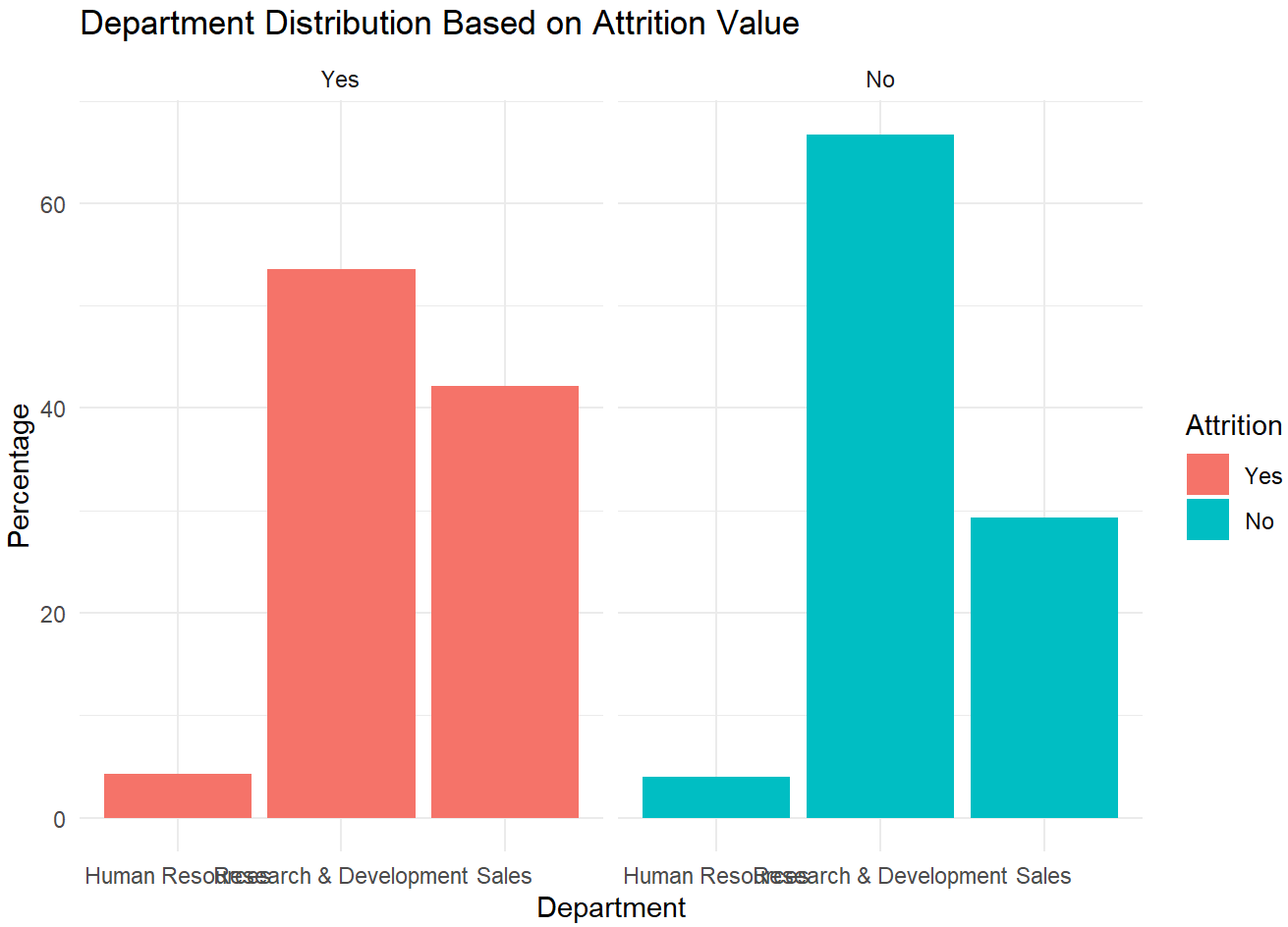
```
# Since the variables are a lot, and i plan on saving time, I plan using a for loop to iterate t
hrough the variables and plot their bar plots to visualize their EDA

# Iterate through the dataset and execute the plot command
for (Variable in names(Talent_Train)){
  # Check if the column is numeric
  if  (!(Variable == "Attrition")) { #(!is.numeric(Talent_Train[[Variable]]) &&

    # Construct the bunch of commands with the current variable

  command <- paste0(
    "summary <- Talent_Train %>% \n",
    "group_by(", Variable, ", Attrition) %>% \n",
    "summarize(count=n(), .groups = 'drop') \n",
    "summary$perc <- 0 \n",
    "summary$perc[summary$Attrition == 'No'] <- summary$count[summary$Attrition == 'No'] / nrow
(Talent_Train[Talent_Train$Attrition == 'No',]) * 100 \n",
    "summary$perc[summary$Attrition == 'Yes'] <- summary$count[summary$Attrition == 'Yes'] / nro
w(Talent_Train[Talent_Train$Attrition == 'Yes',]) * 100 \n",
    "summary %>% ggplot(aes(x=", Variable, ",y=perc,fill=Attrition)) + geom_bar(stat='identity')
+ facet_wrap(~Attrition) + \n",
    "ylab('Percentage') + xlab('", Variable, "') + ggtitle('", Variable, " Distribution Based on
Attrition Value') + theme_minimal() \n"
  )

    # Execute the command
    plot <- eval(parse(text = command))

    print(plot)
  }
}
```

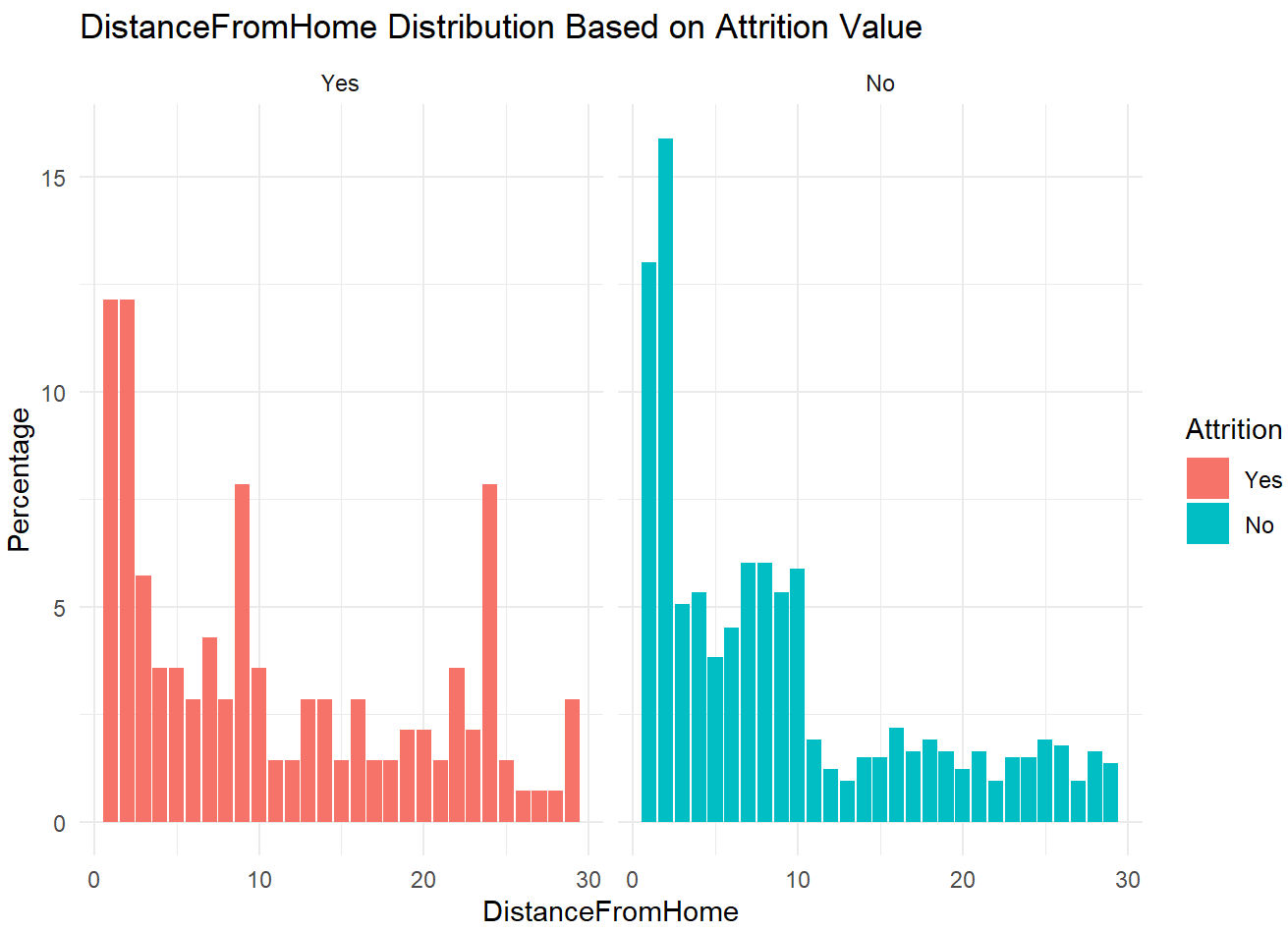

ID Distribution Based on Attrition Value
Yes
No
Percentage
0.0
0.2
0.4
0.6
0
250
500
750
ID
Attrition
Yes
No

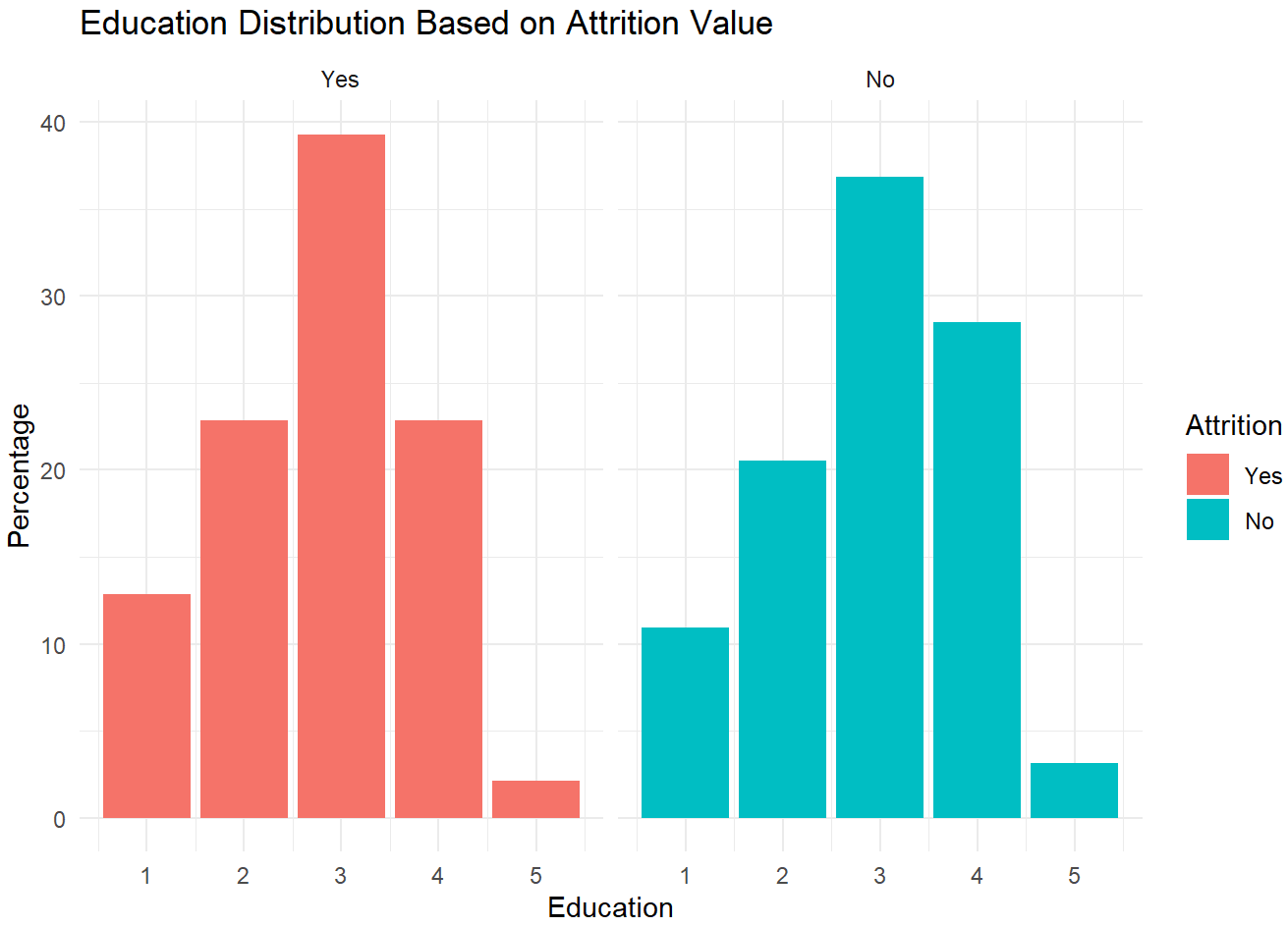

Age Distribution Based on Attrition Value
Yes
No
Percentage
0.0
2.5
5.0
7.5
20
30
40
50
60
Age
Attrition
Yes
No

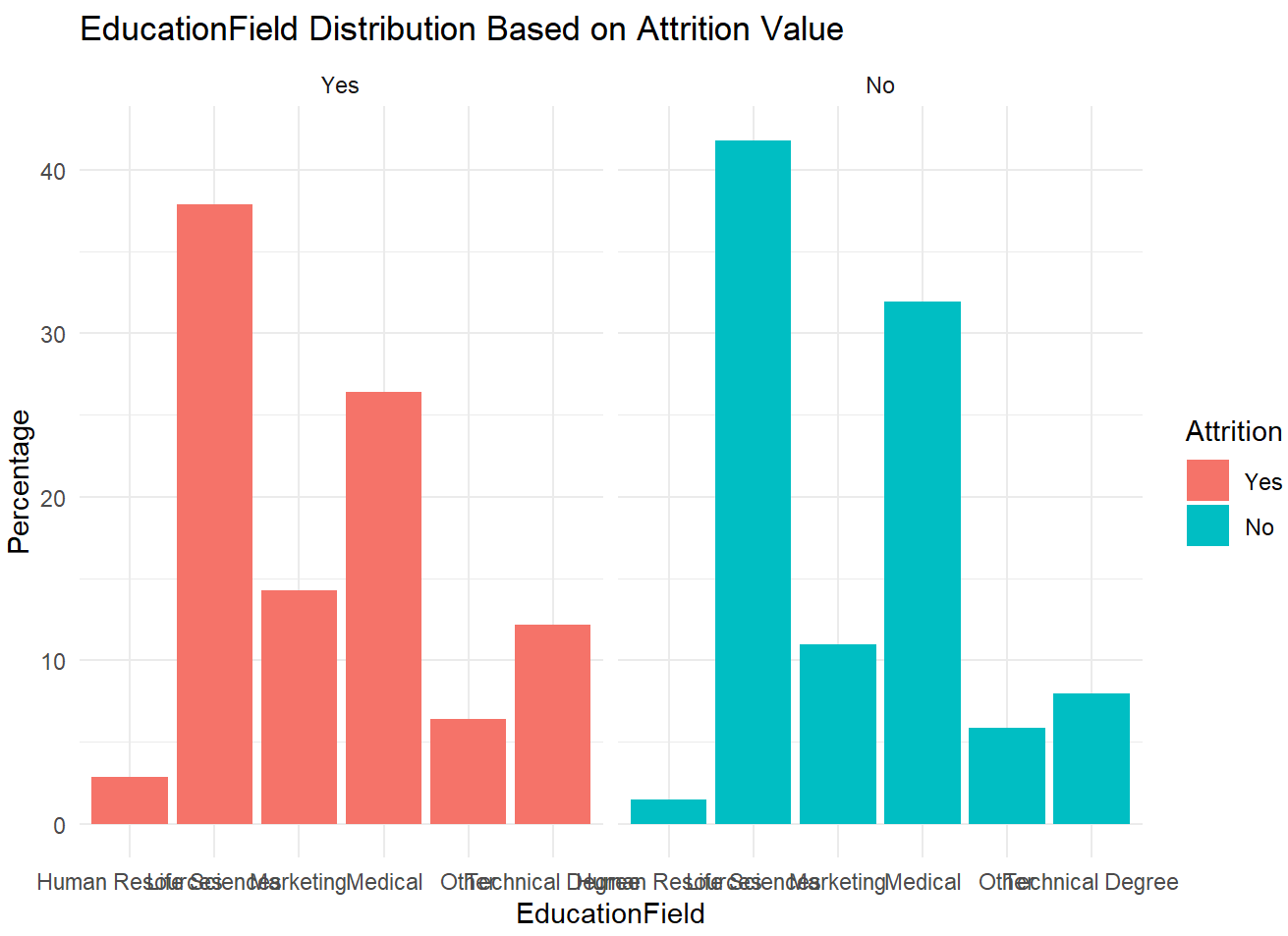

BusinessTravel Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
Non-Travel
Travel_Frequently
Travel_Rarely
BusinessTravel
Attrition
Yes
No

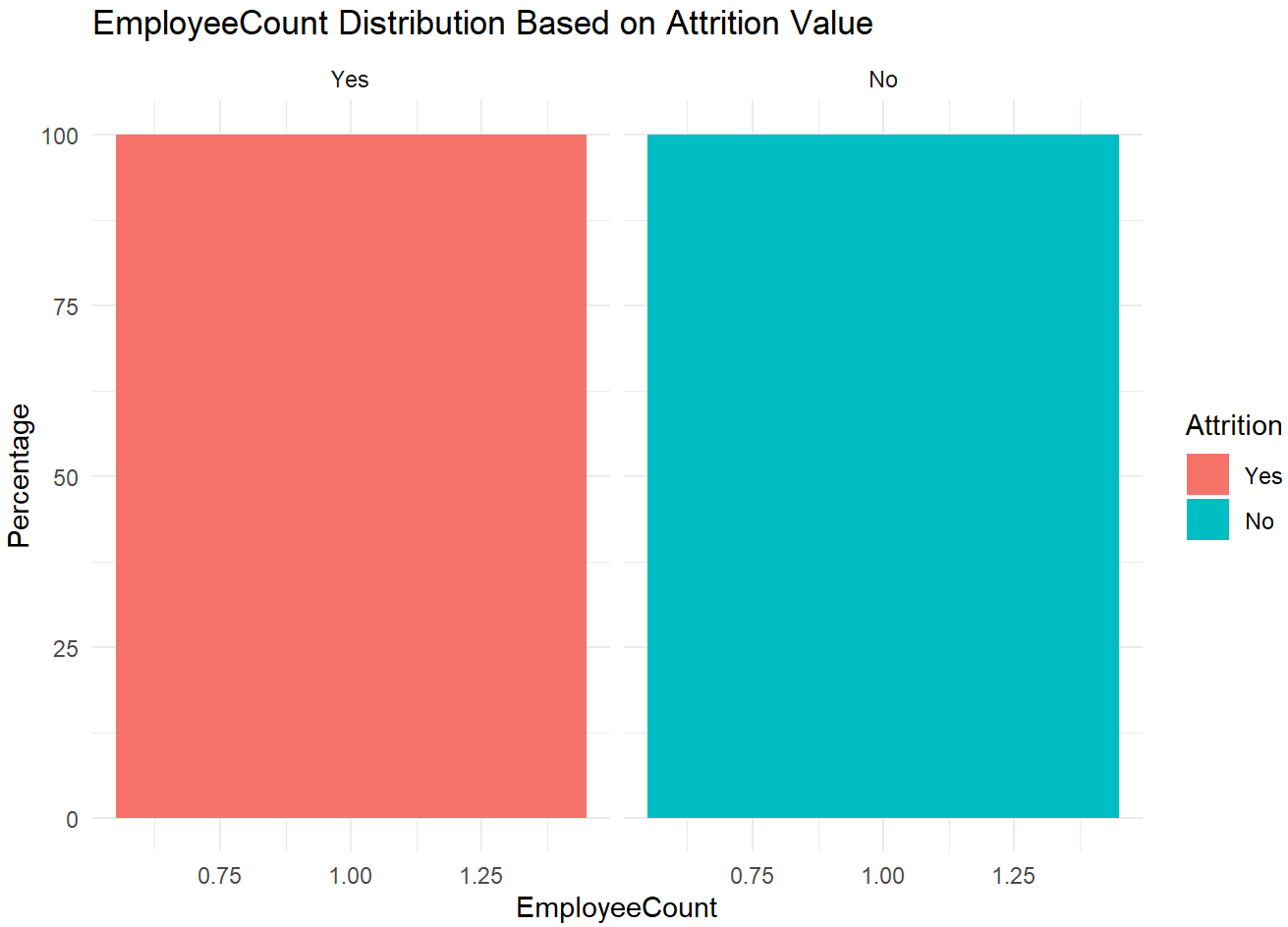

DailyRate Distribution Based on Attrition Value
Yes
No
Percentage
0.0
0.5
1.0
1.5
400
800
1200
DailyRate
Attrition
Yes
No

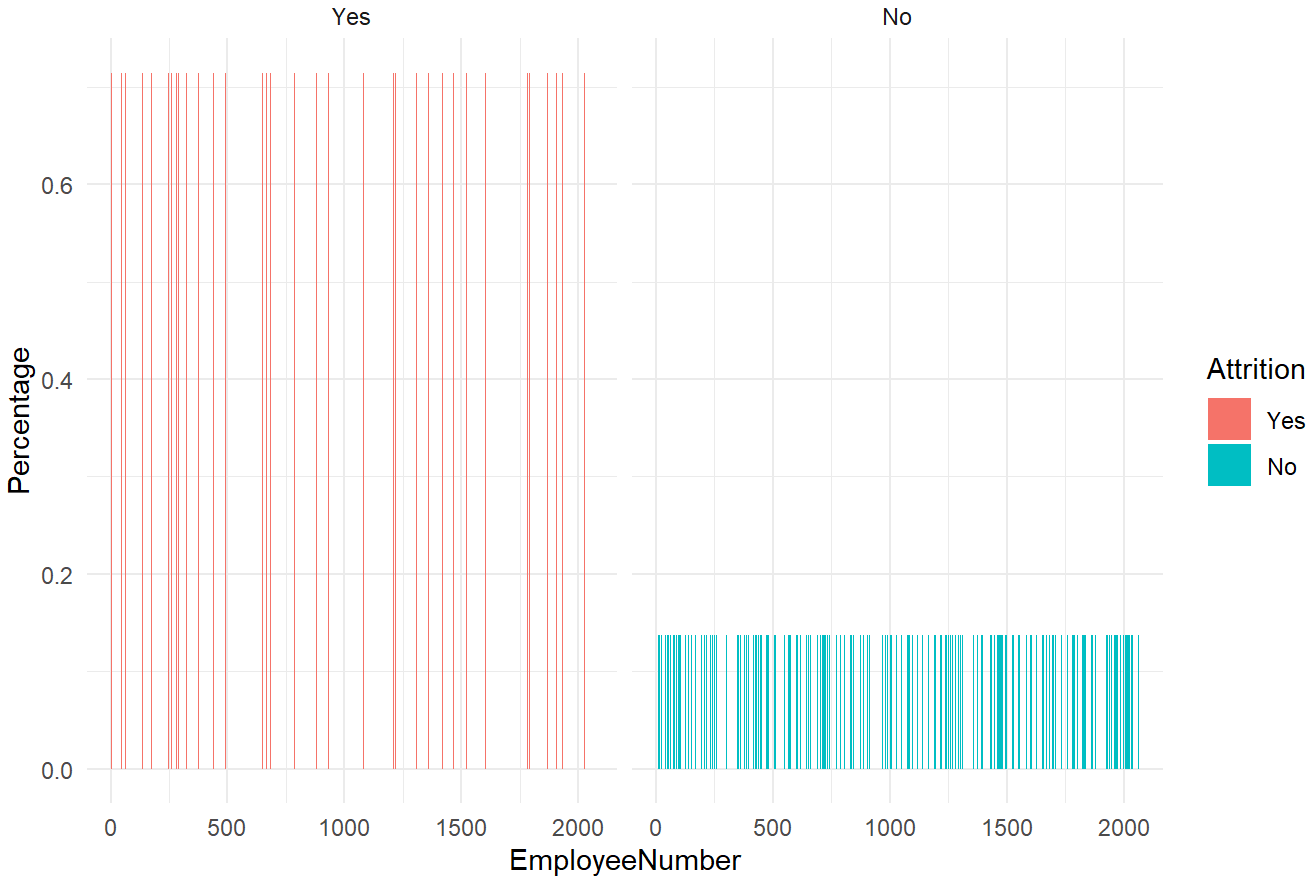

Department Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
Human Resources
Research & Development
Sales
Department
Attrition
Yes
No


DistanceFromHome Distribution Based on Attrition Value
Yes
No
Percentage
0
5
10
15
0
10
20
30
DistanceFromHome
Attrition
Yes
No

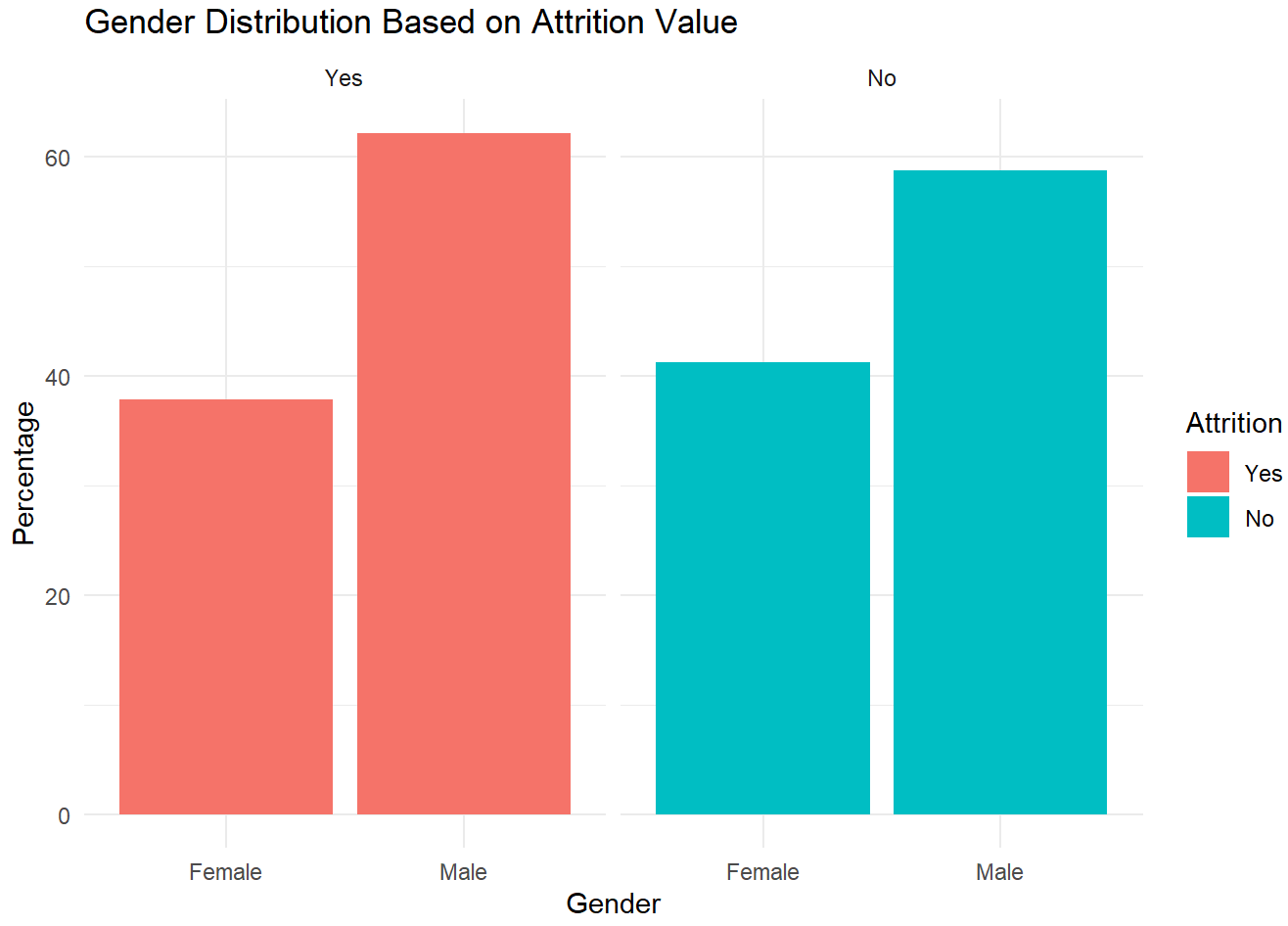

Education Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
40
1
2
3
4
5
Education
Attrition
Yes
No

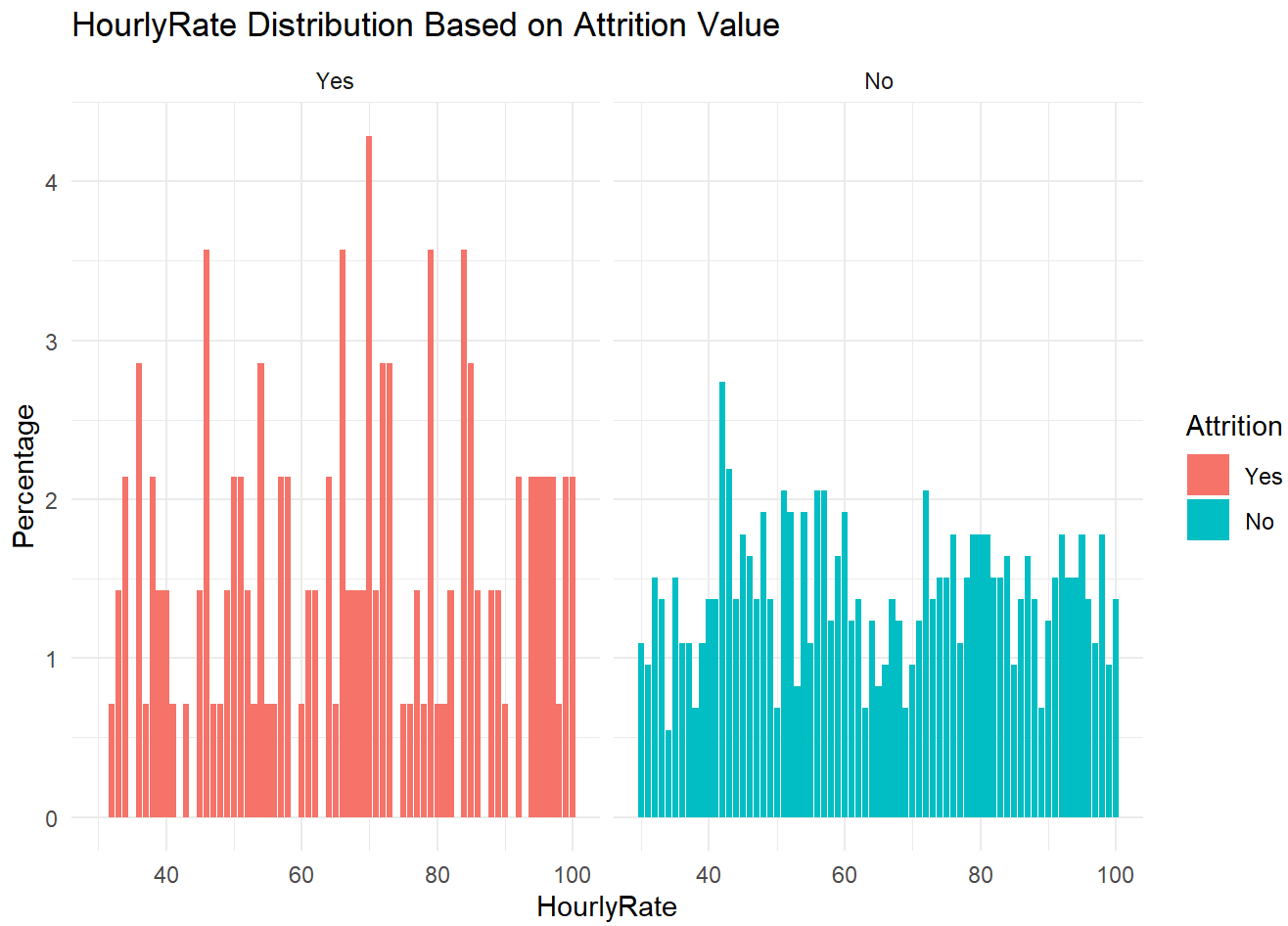

EducationField Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
40
Human Resources
Life Sciences
Marketing
Medical
Other
Technical Degree
EducationField
Attrition
Yes
No


EmployeeCount Distribution Based on Attrition Value
Yes
No
100
75
50
25
0
Percentage
0.75
1.00
1.25
EmployeeCount
Attrition
Yes
No

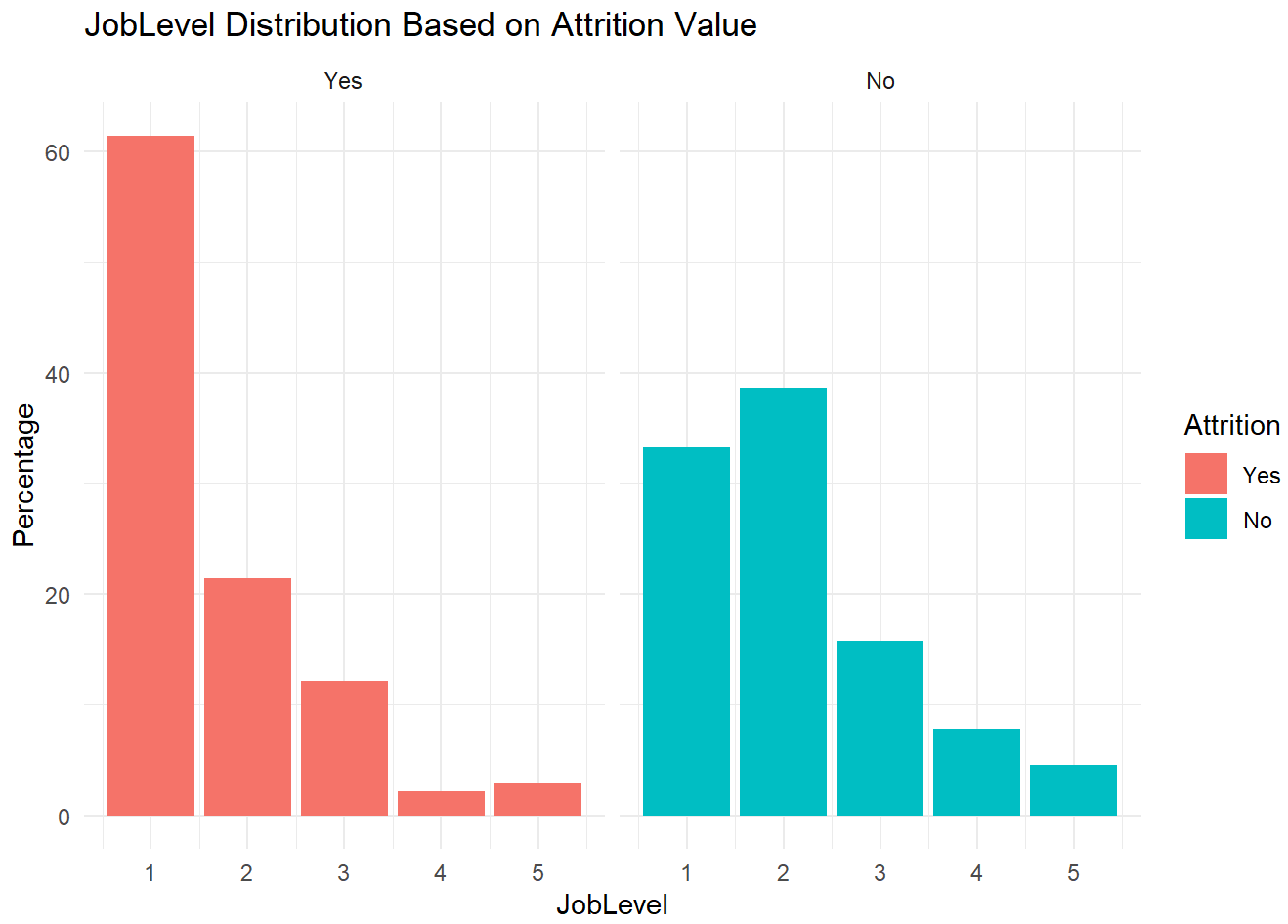

EmployeeNumber Distribution Based on Attrition Value
Yes
No
0.6
0.4
0.2
0.0
Percentage
0
500
1000
1500
2000
EmployeeNumber
Attrition
Yes
No

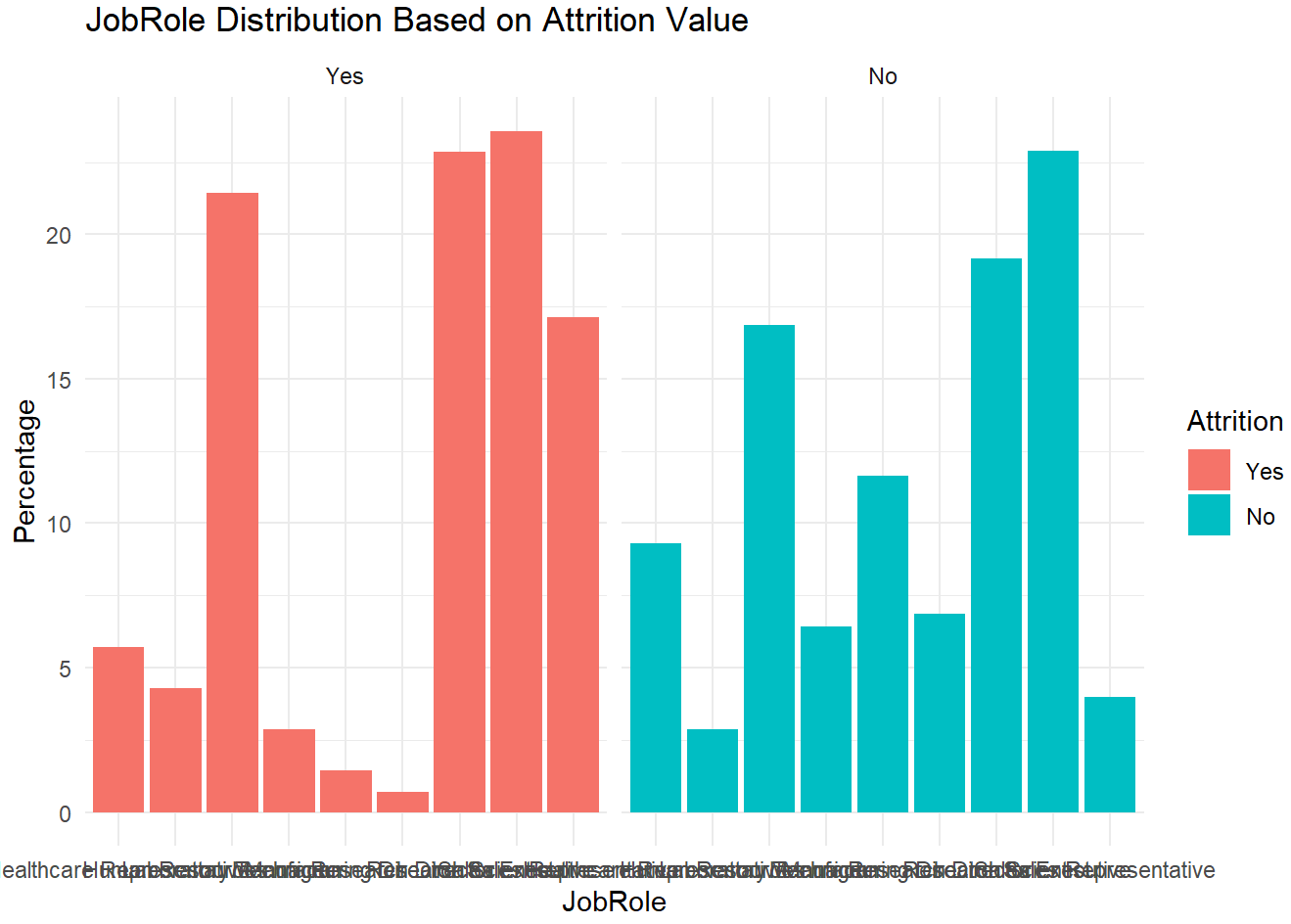

EnvironmentSatisfaction Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
1
2
3
4
EnvironmentSatisfaction
Attrition
Yes
No

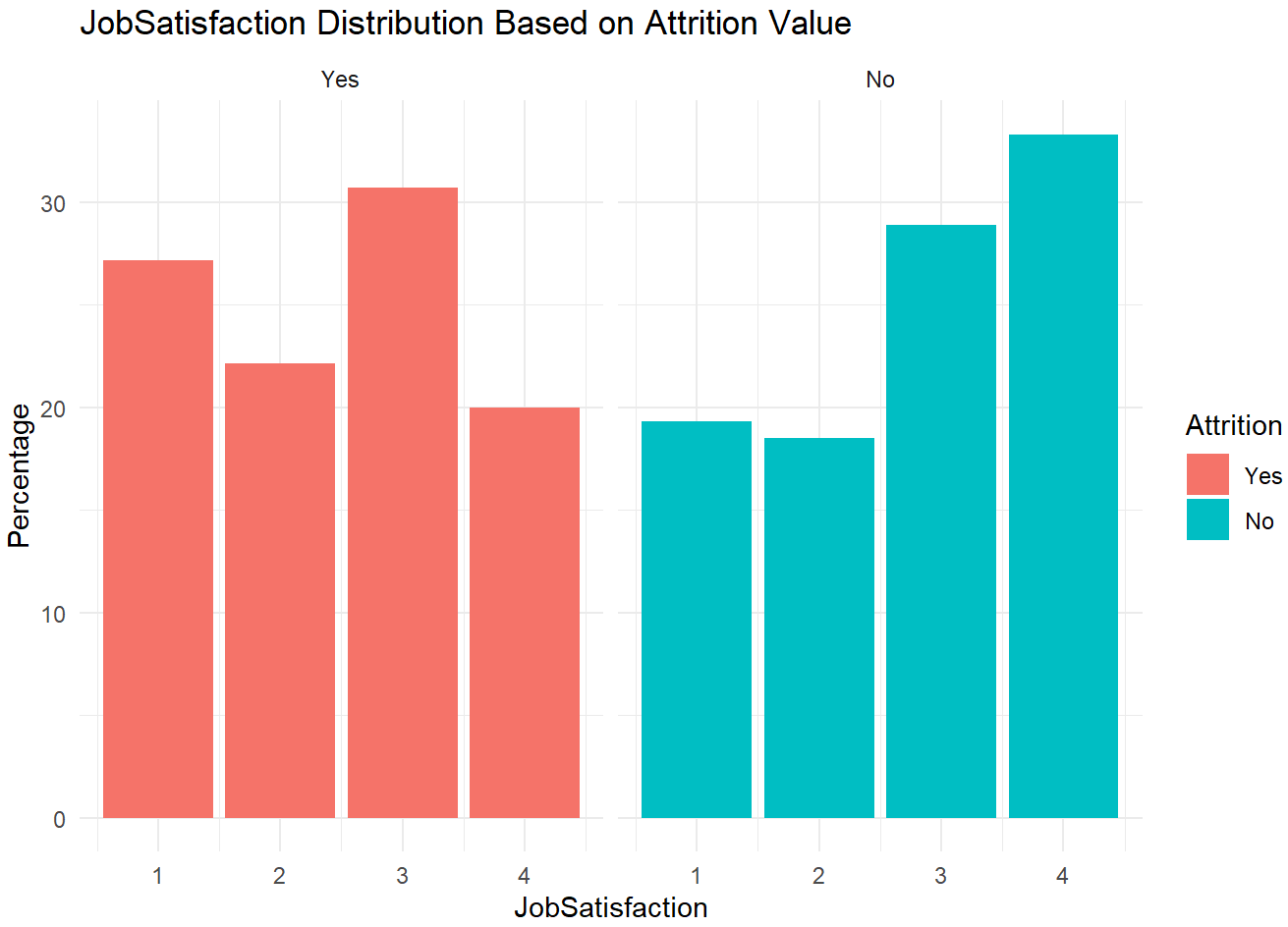

Gender Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
Female
Male
Gender
Attrition
Yes
No

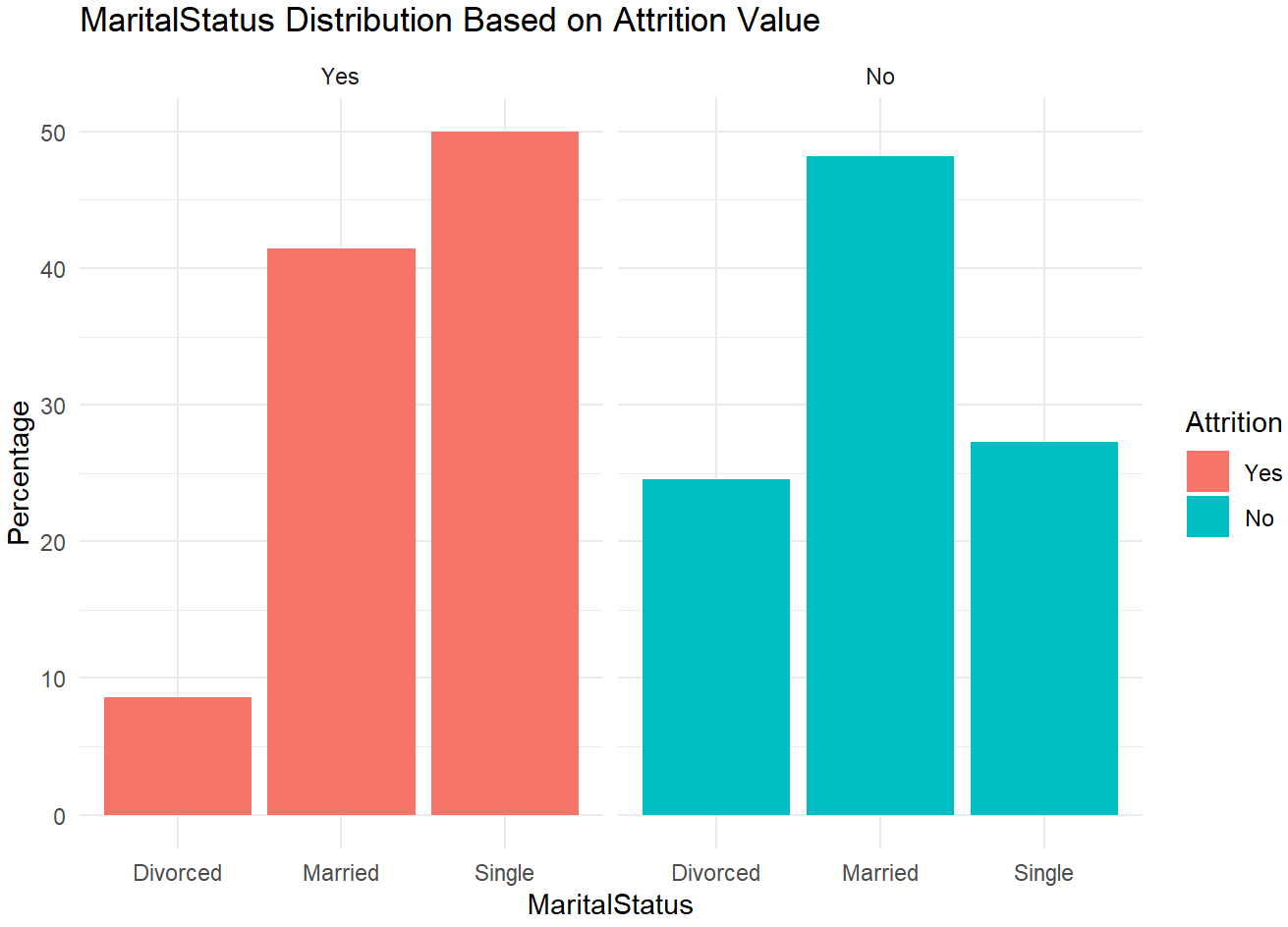

HourlyRate Distribution Based on Attrition Value
Yes
No
Percentage
0
1
2
3
4
40
60
80
100
HourlyRate
Attrition
Yes
No

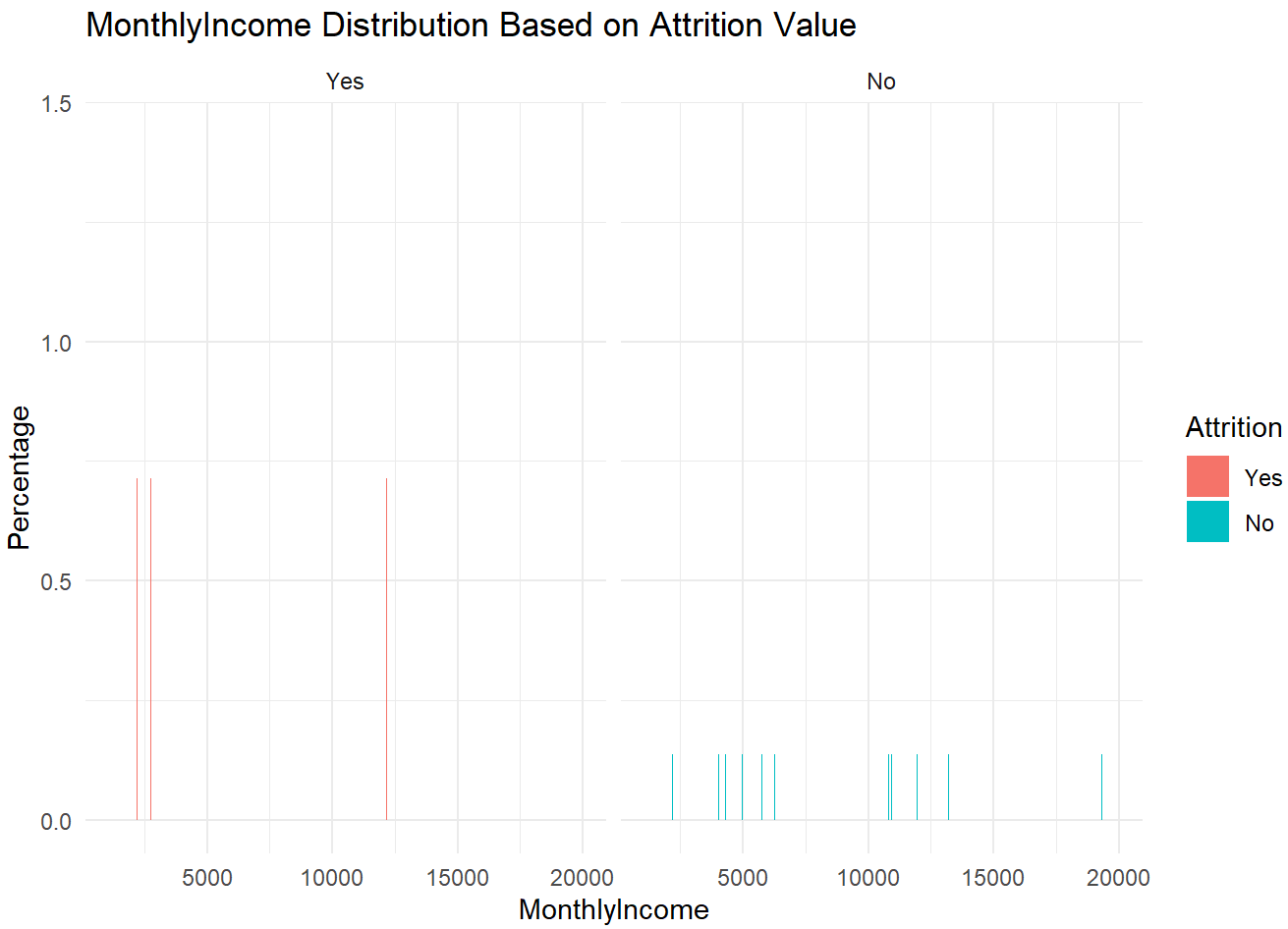

JobInvolvement Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
1
2
3
4
JobInvolvement
Attrition
Yes
No

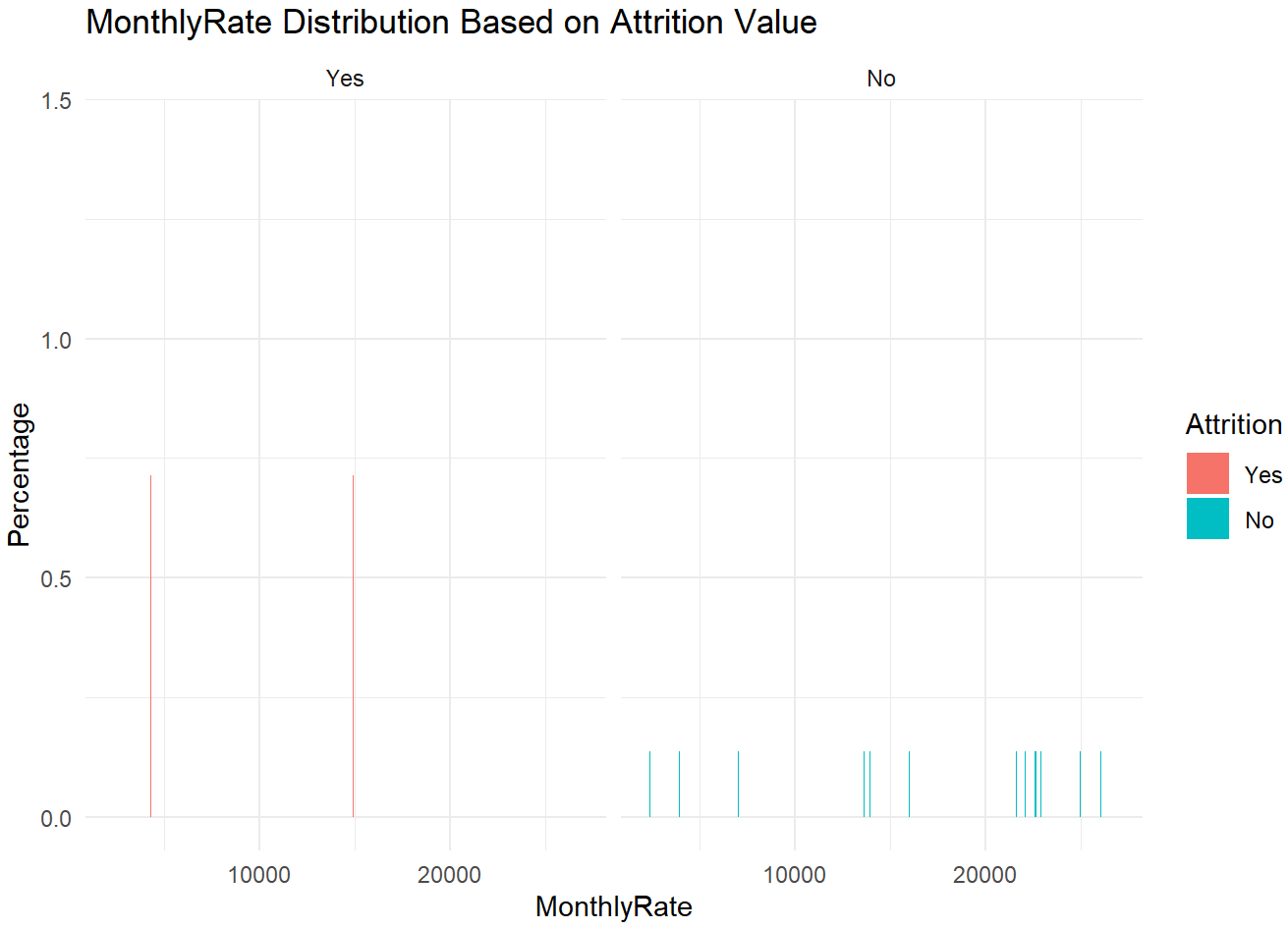

JobLevel Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
1
2
3
4
5
JobLevel
Attrition
Yes
No

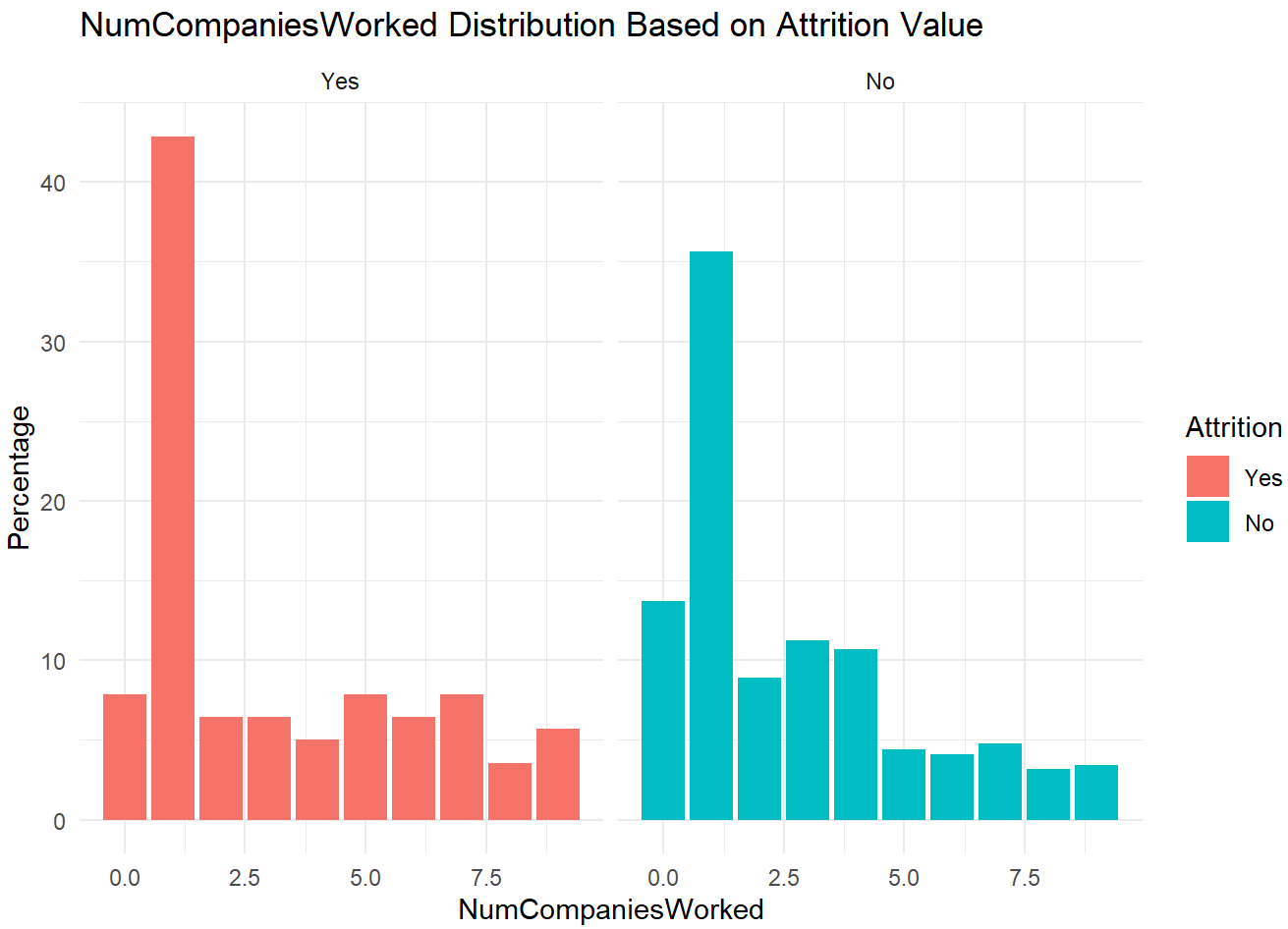

JobRole Distribution Based on Attrition Value
Yes
No
Percentage
0
5
10
15
20
JobRole
Attrition
Yes
No

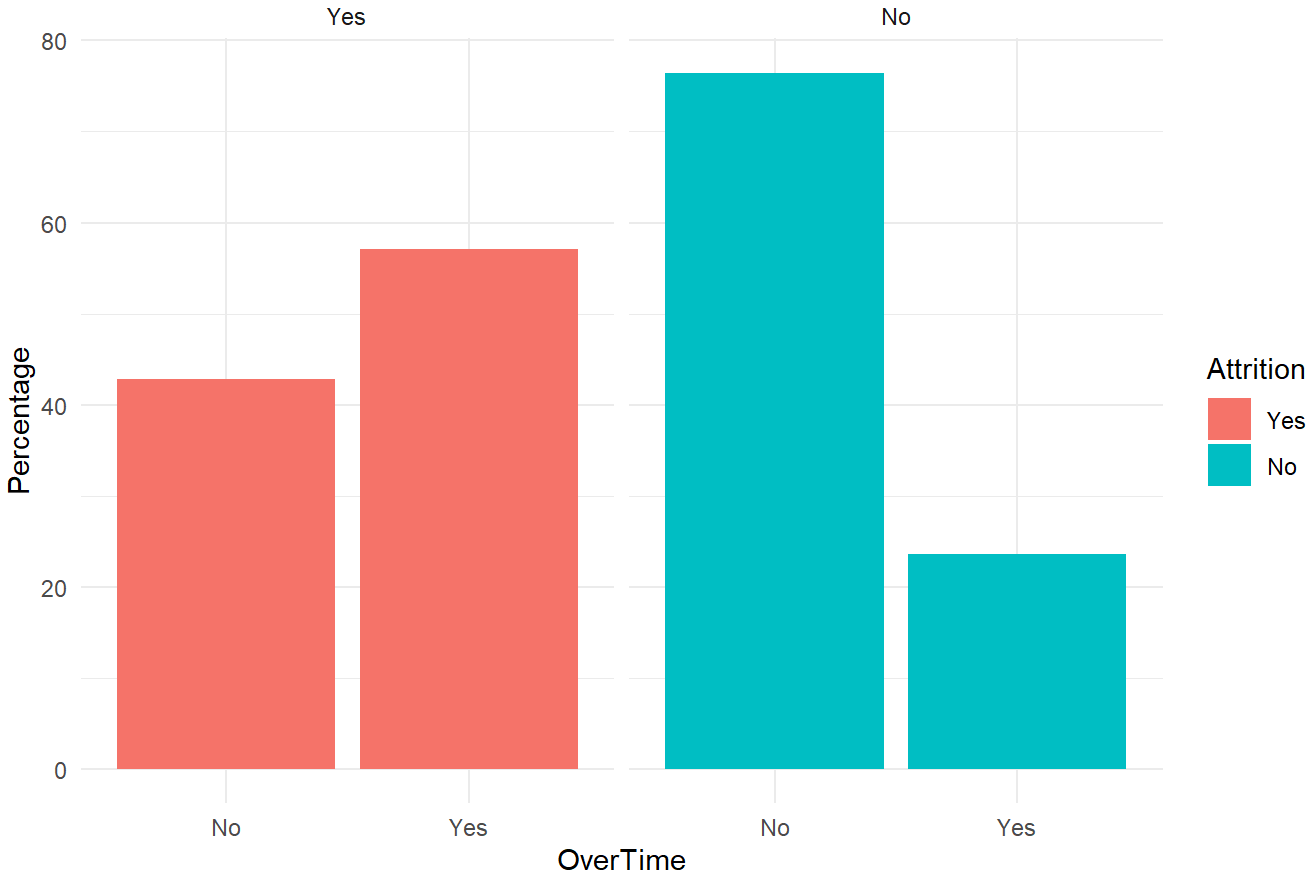

JobSatisfaction Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
1
2
3
4
JobSatisfaction
Attrition
Yes
No

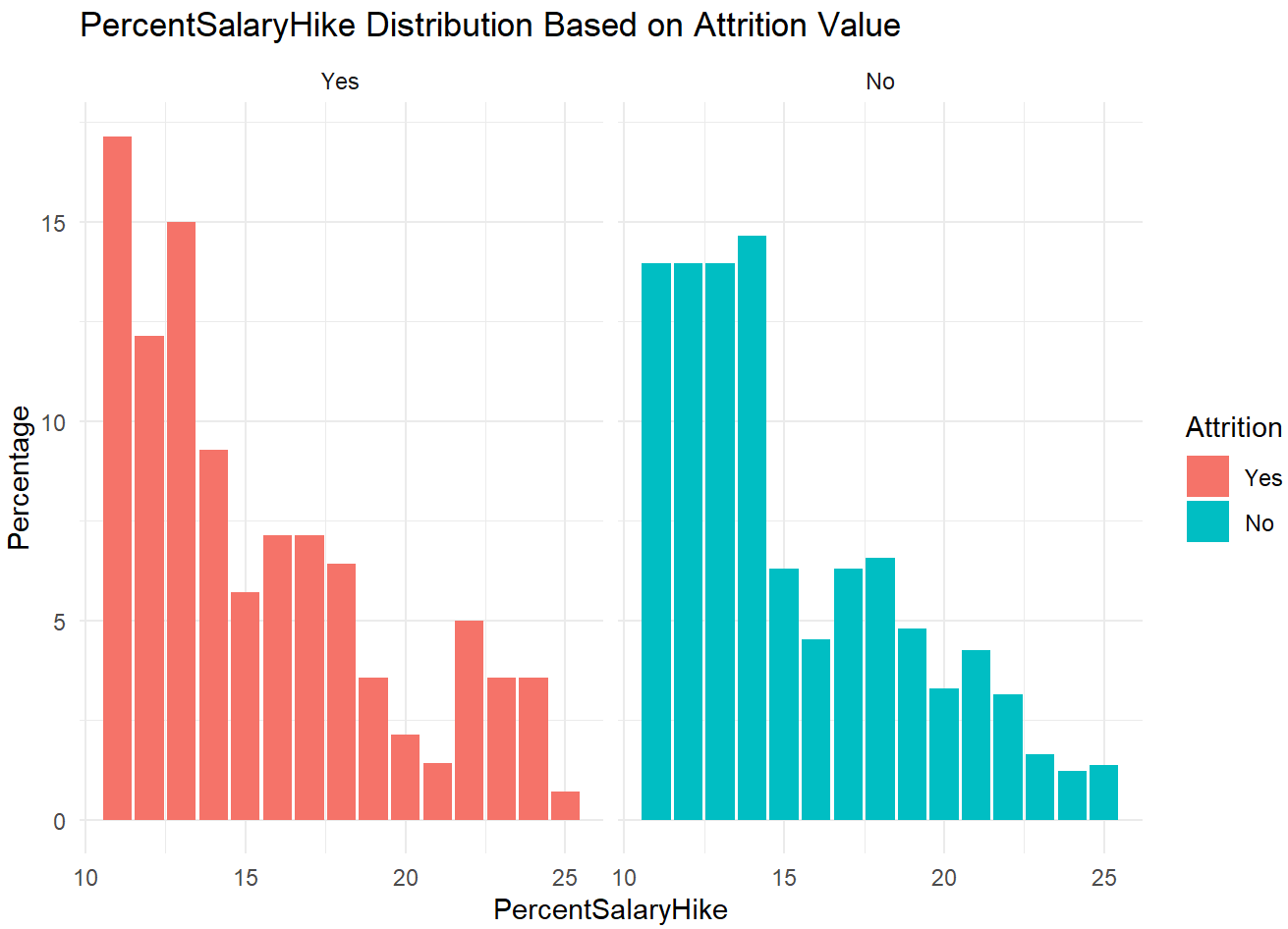

MaritalStatus Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
40
50
Divorced
Married
Single
MaritalStatus
Attrition
Yes
No

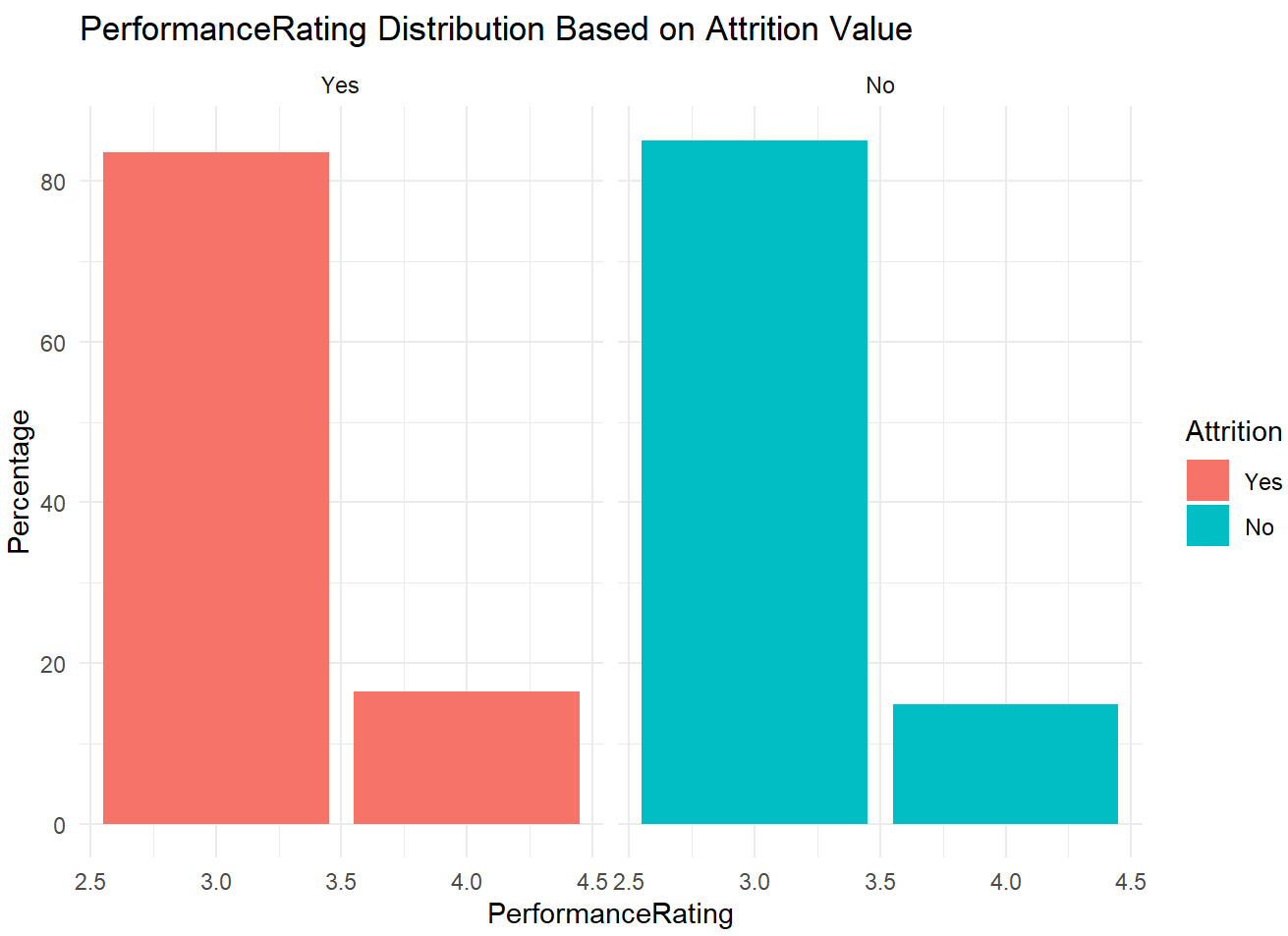

MonthlyIncome Distribution Based on Attrition Value
Yes
No
1.5
1.0
0.5
0.0
Percentage
5000
10000
15000
20000
MonthlyIncome
Attrition
Yes
No


MonthlyRate Distribution Based on Attrition Value
Yes
No
1.5
1.0
0.5
0.0
Percentage
10000
20000
MonthlyRate
Attrition
Yes
No

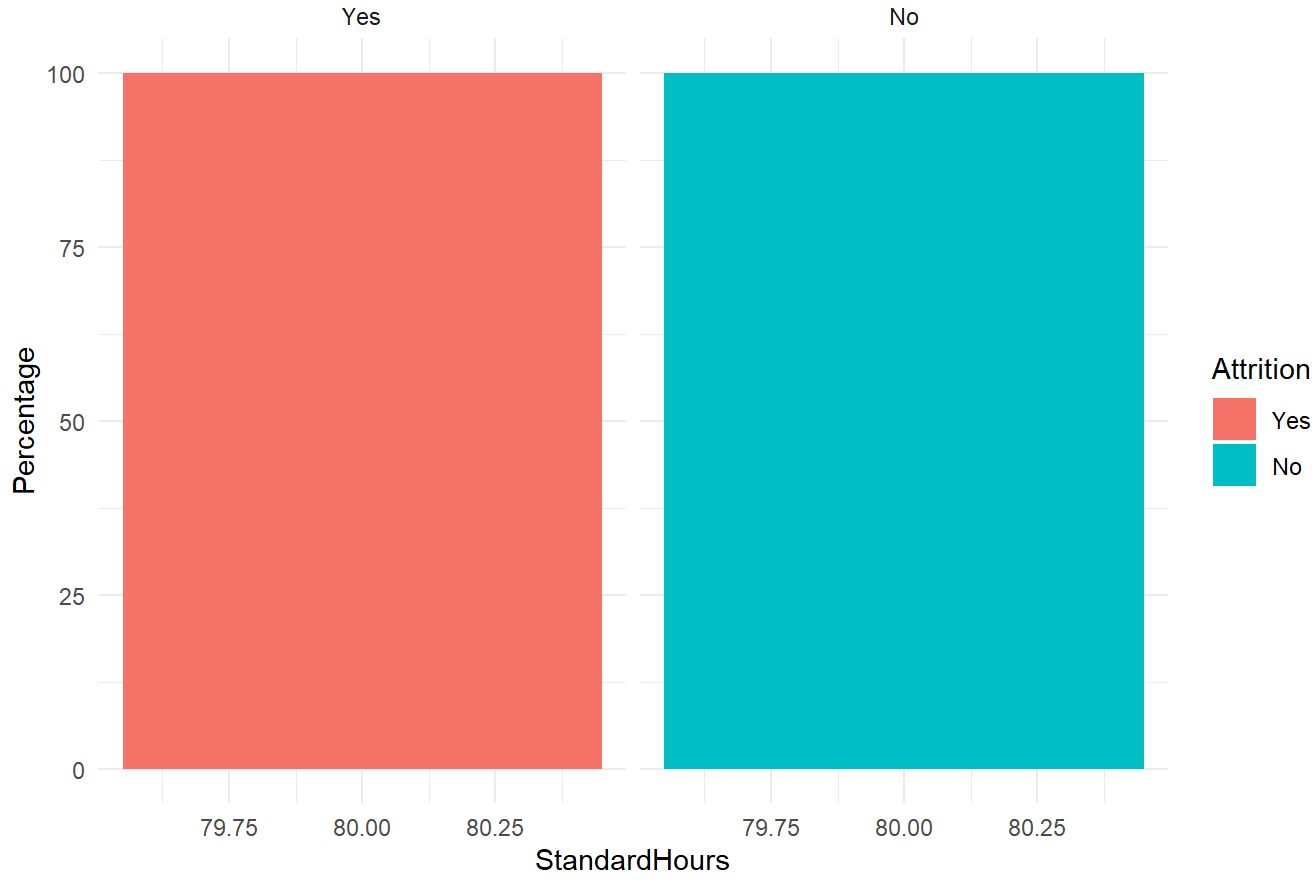

NumCompaniesWorked Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
40
0.0
2.5
5.0
7.5
NumCompaniesWorked
Attrition
Yes
No

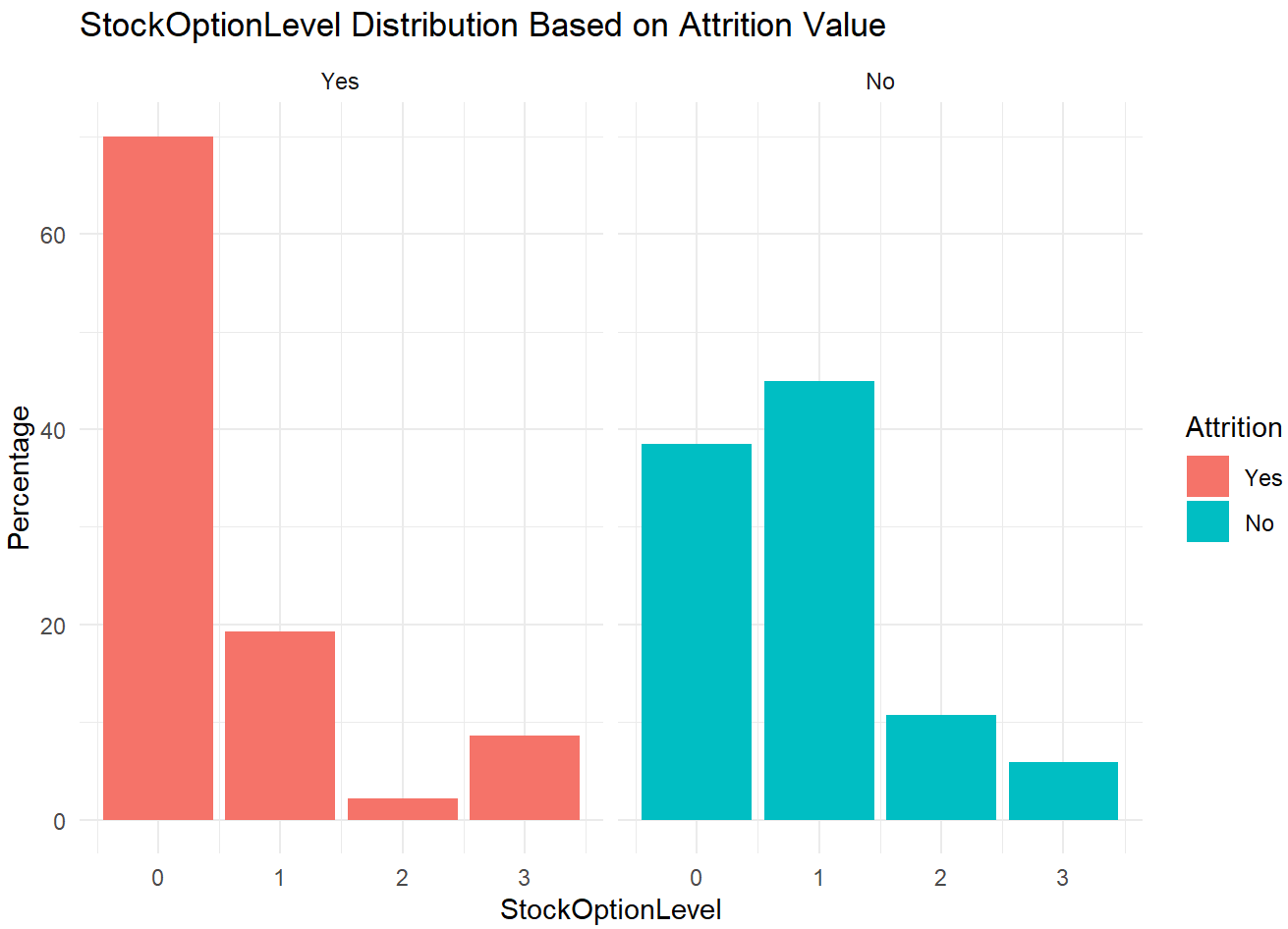

OverTime Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
80
No
Yes
OverTime
Attrition
Yes
No

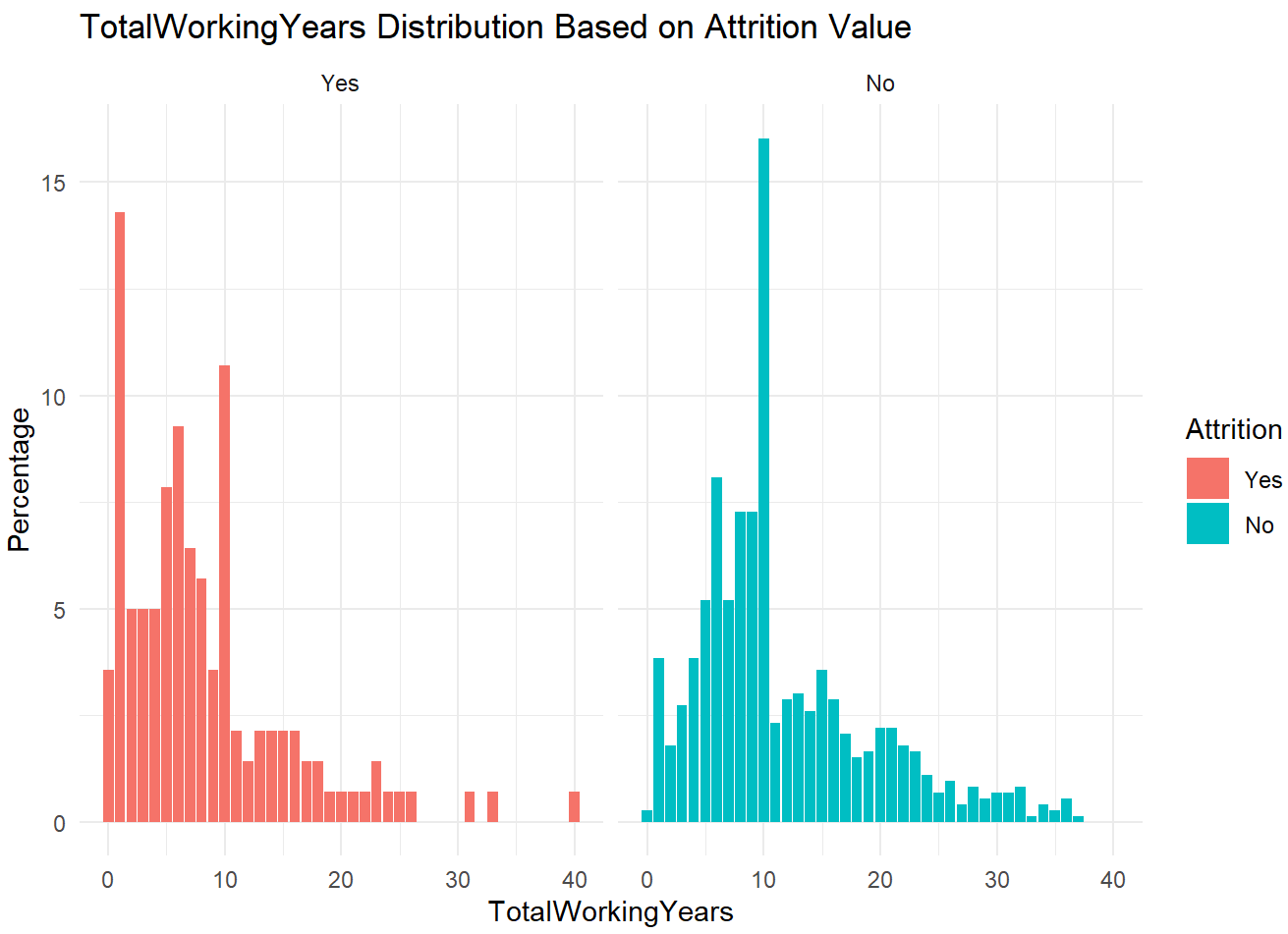

PercentSalaryHike Distribution Based on Attrition Value
Yes
No
Percentage
0
5
10
15
10
15
20
25
PercentSalaryHike
Attrition
Yes
No


PerformanceRating Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
80
2.5
3.0
3.5
4.0
4.5
PerformanceRating
Attrition
Yes
No

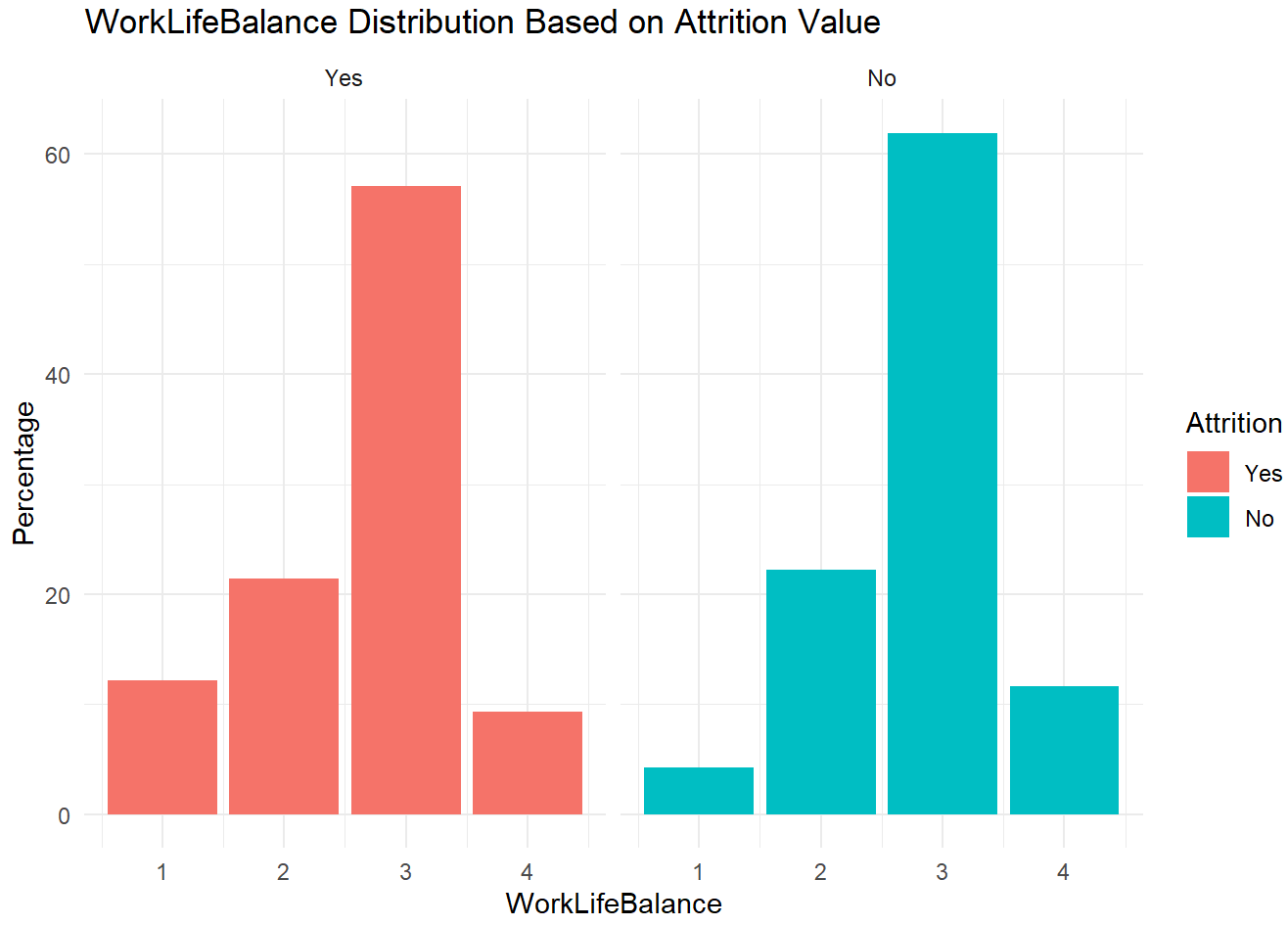

RelationshipSatisfaction Distribution Based on Attrition Value
Yes
No
Percentage
30
20
10
0
1
2
3
4
RelationshipSatisfaction
Attrition
Yes
No

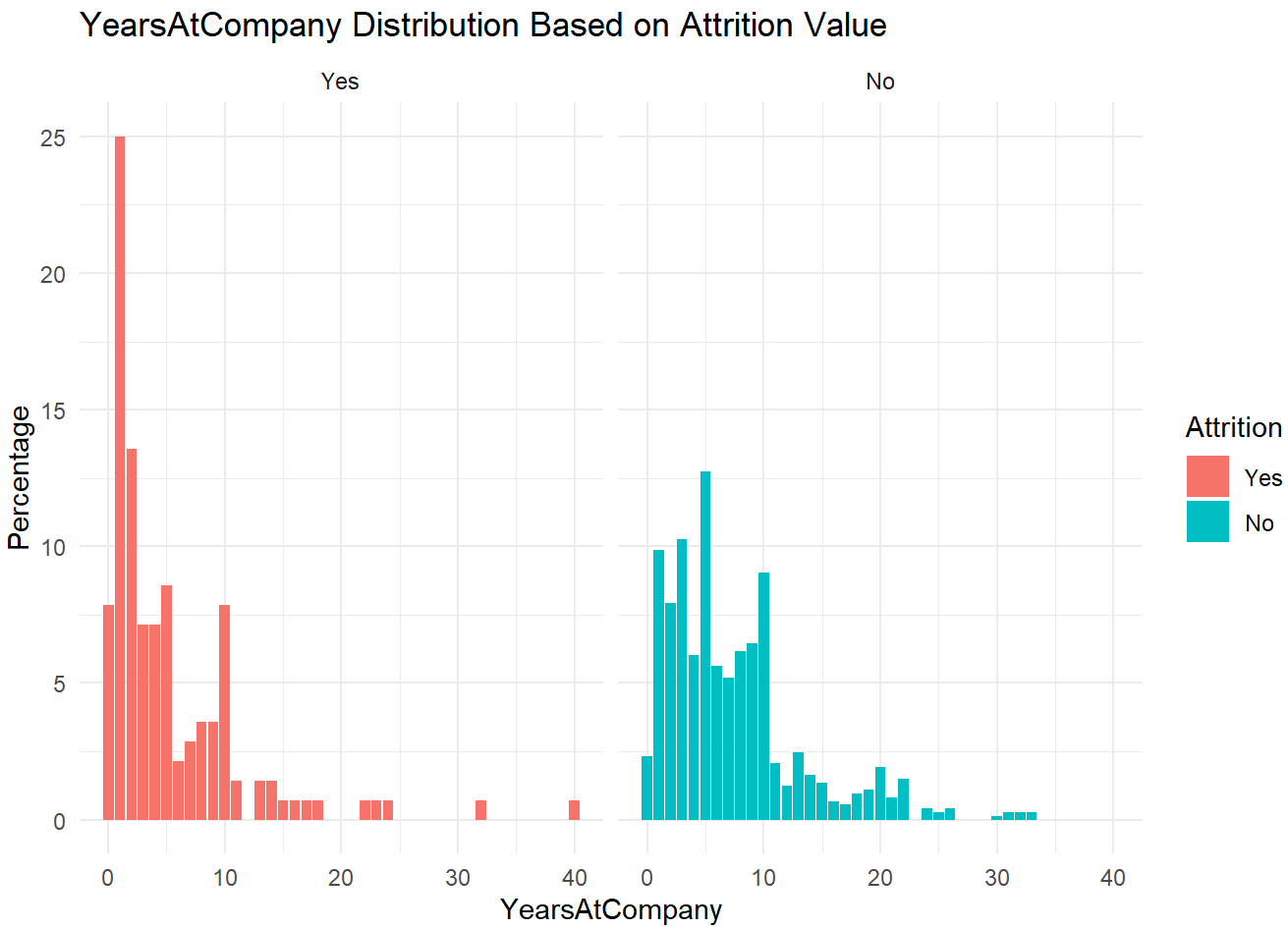

StandardHours Distribution Based on Attrition Value
Yes
No
Percentage
100
75
50
25
0
79.75
80.00
80.25
StandardHours
Attrition
Yes
No

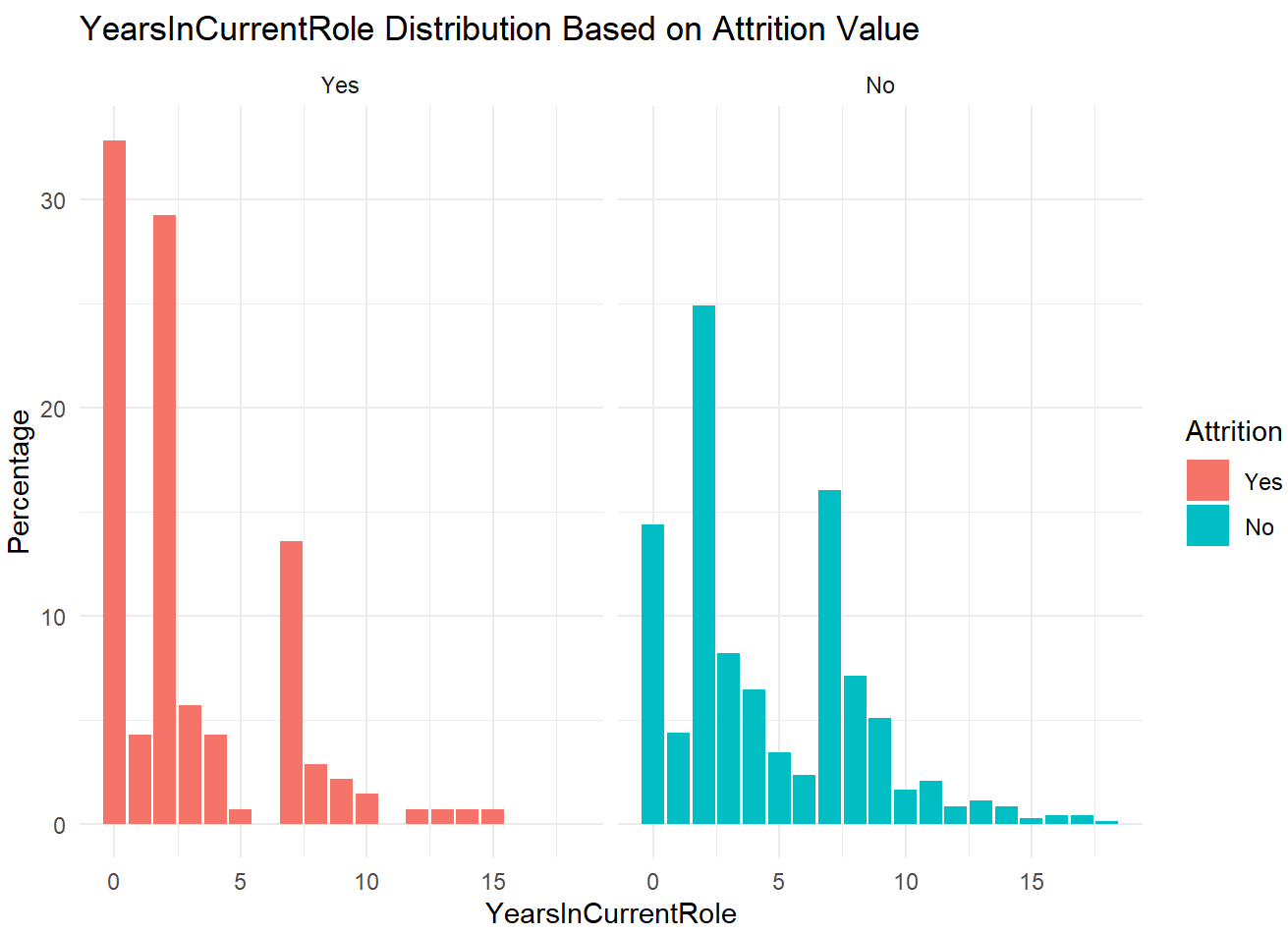

StockOptionLevel Distribution Based on Attrition Value
Yes
No
Percentage
60
40
20
0
0
1
2
3
StockOptionLevel
Attrition
Yes
No

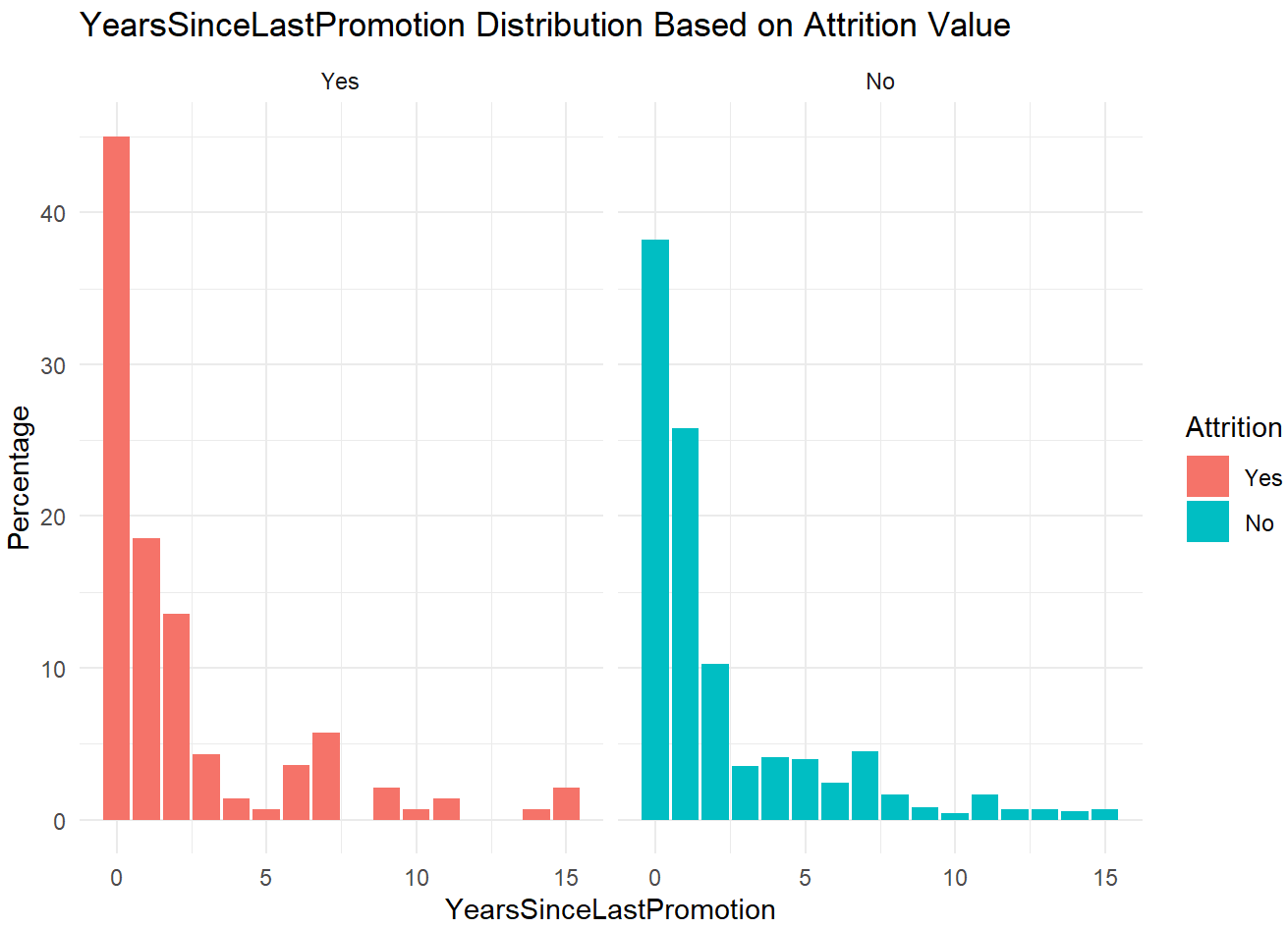

TotalWorkingYears Distribution Based on Attrition Value
Yes
No
Percentage
15
10
5
0
0
10
20
30
40
TotalWorkingYears
Attrition
Yes
No

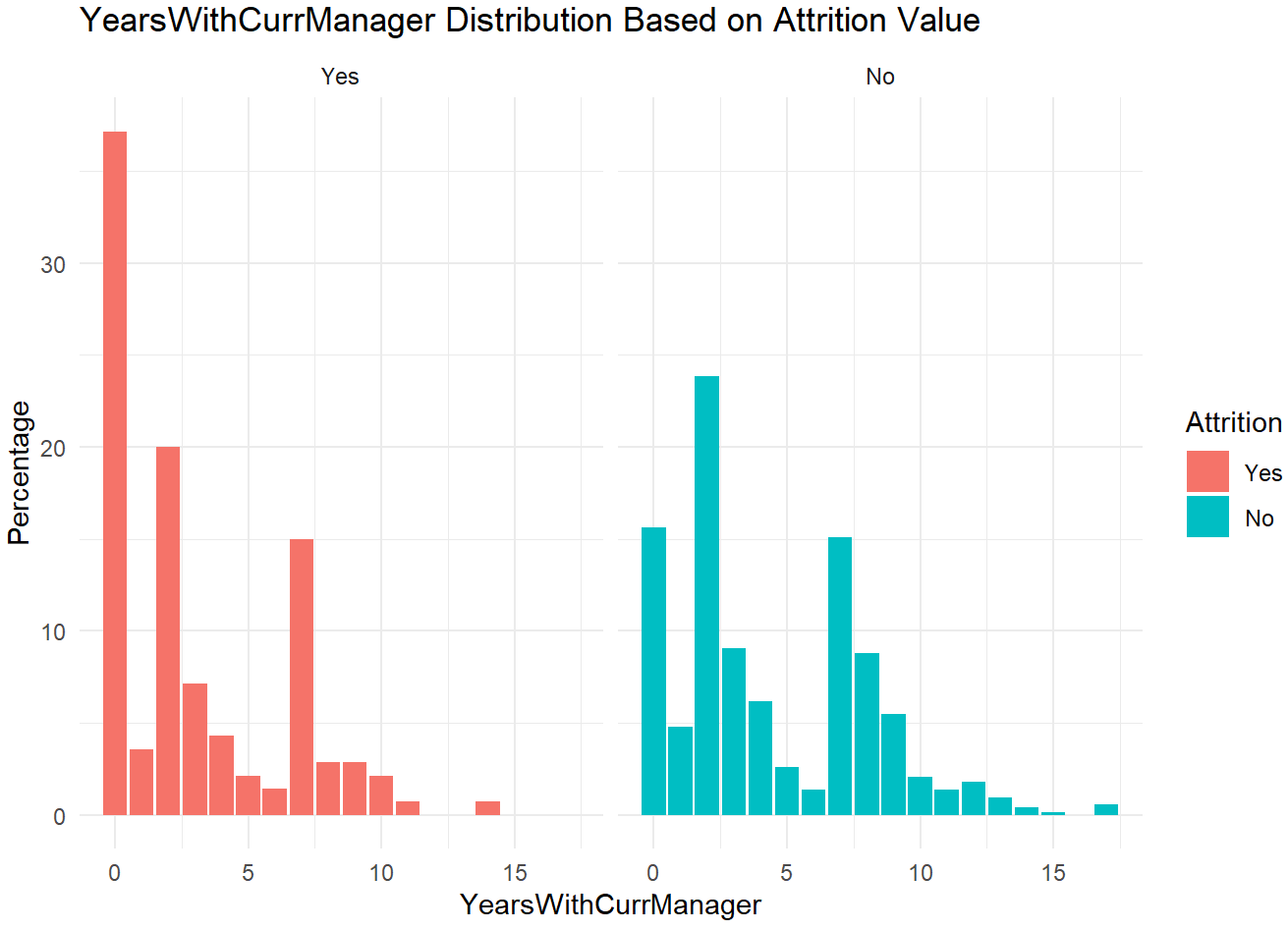

TrainingTimesLastYear Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
40
0
2
4
6
TrainingTimesLastYear
Attrition
Yes
No

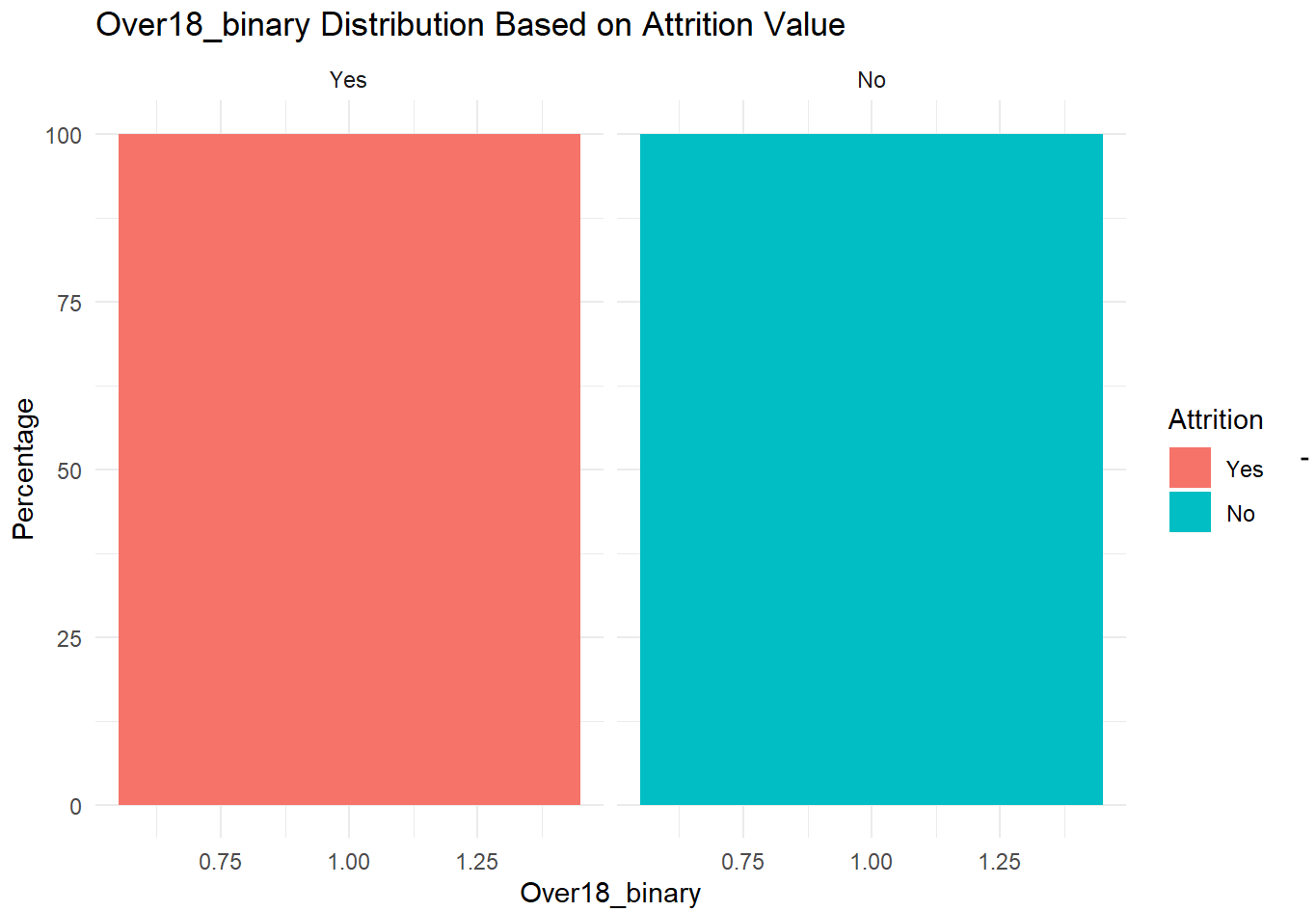

WorkLifeBalance Distribution Based on Attrition Value
Yes
No
Percentage
0
20
40
60
1
2
3
4
WorkLifeBalance
Attrition
Yes
No


YearsAtCompany Distribution Based on Attrition Value
Yes
No
Percentage
YearsAtCompany
Attrition
Yes
No


YearsInCurrentRole Distribution Based on Attrition Value
Yes
No
Percentage
YearsInCurrentRole
Attrition
Yes
No


YearsSinceLastPromotion Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
40
0
5
10
15
YearsSinceLastPromotion
Attrition
Yes
No


YearsWithCurrManager Distribution Based on Attrition Value
Yes
No
Percentage
0
10
20
30
0
5
10
15
YearsWithCurrManager
Attrition
Yes
No



There seems to be a high turnover rate among employees with zero stock options level. - There is a higher level of Employee turnover Rate among the employees in Job Level - There is a similar distribution overall between the monthly salary and employees that didn't leave in terms of JobLevel. - There are similarities among the yes and no distributions. - There seems to be high turnover rate among employees with 10 years of in the company. - There seems to be a low turn over rate among employees with 1-3 years in the company. - There are similar distribution among the monthly income and the no attrition rates.

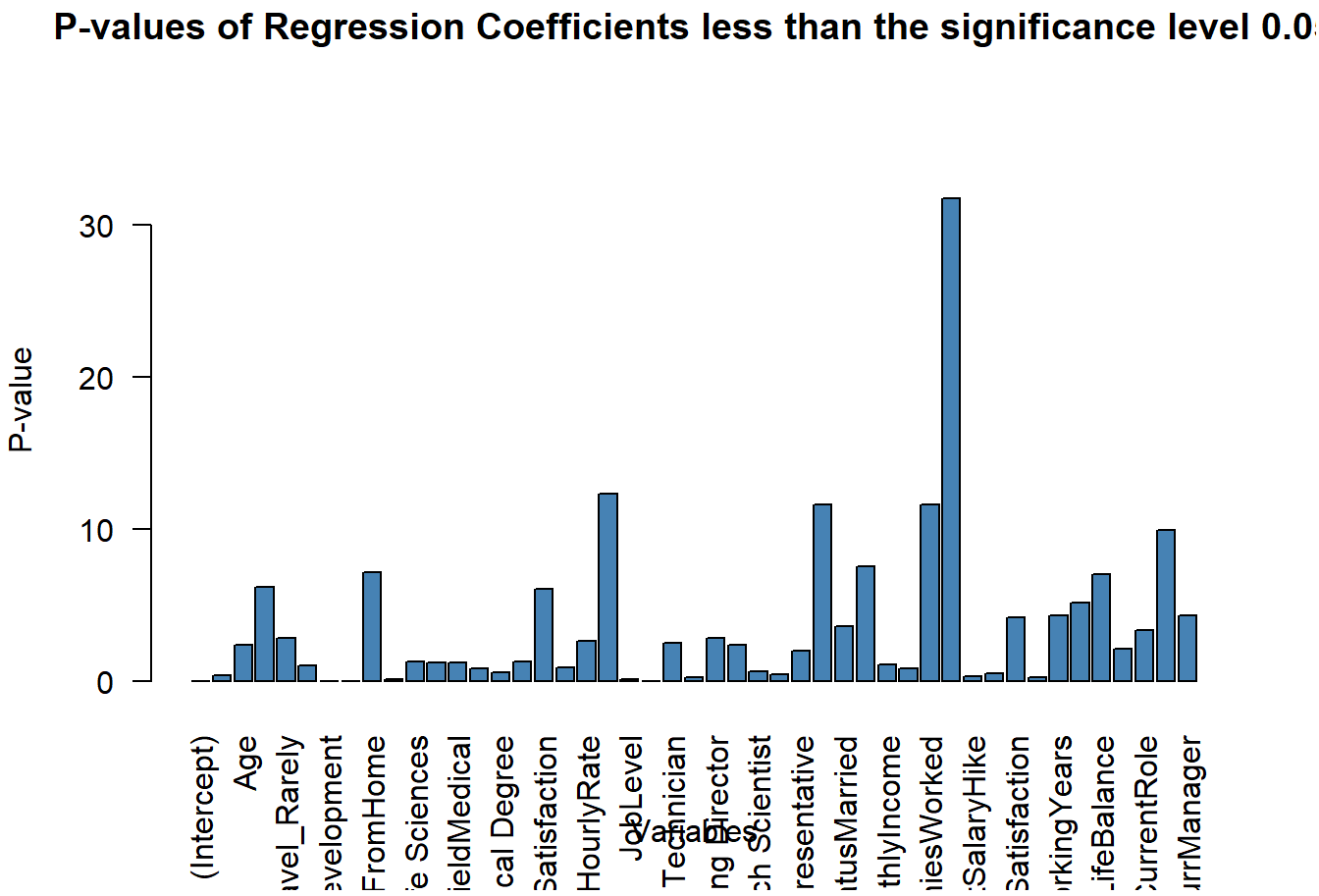
LOOKING AT THE PVALUE DISTRIBUTIONS FOR THE ATTRITION

Looking at how each variable in the model, significantly impacts our response variable (MonthlyIncome)

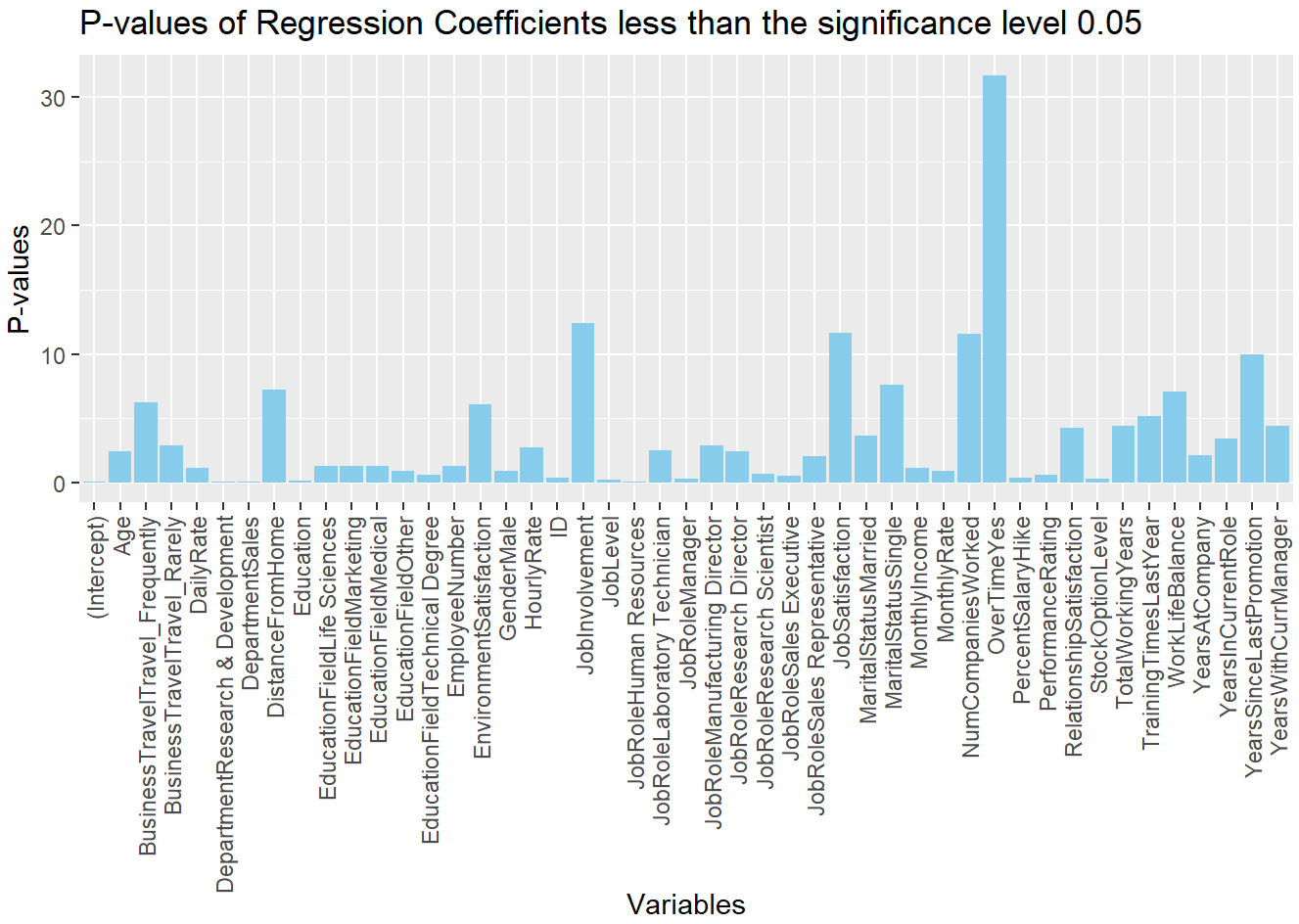
```
model <-glm(Attrition ~ ., data = Talent_Train, family="binomial")

# Extract variable names
variable_names <- rownames(summary(model)$coefficients)

# getting the  p-values from the model3
p_values <- summary(model)$coefficients[, 4]  # Assuming p-values are in the 4th column of the s
ummary table
p_values <- data.frame(p_values)$p_values

df <- data.frame(variable_names, p_values) #combining the pvalues and variable names into a data
frame

df <- df[!df$p_value == 0 , ] #removing varaiables with pvalue = 0
df$p_values <- log(df$p_values)  * -1

# Rank p-values in the dataset from max to min
df$rank <- rank(-df$p_value)

sorted_df <- df[order(df$rank), ]

sorted_df
```

```
##                      variable_names    p_values rank
## 36                      OverTimeYes 31.74224176    1
## 20                   JobInvolvement 12.37046792    2
## 30                  JobSatisfaction 11.63504785    3
## 35               NumCompaniesWorked 11.60462375    4
## 46          YearsSinceLastPromotion  9.97531527    5
## 32              MaritalStatusSingle  7.58076069    6
## 9                  DistanceFromHome  7.21448298    7
## 43                  WorkLifeBalance  7.06700978    8
## 4   BusinessTravelTravel_Frequently  6.20191210    9
## 17          EnvironmentSatisfaction  6.06532036   10
## 42            TrainingTimesLastYear  5.17313554   11
## 47             YearsWithCurrManager  4.37140068   12
## 41                TotalWorkingYears  4.35937156   13
## 39         RelationshipSatisfaction  4.24141087   14
## 31              MaritalStatusMarried  3.63311280   15
## 45               YearsInCurrentRole  3.38610894   16
## 5       BusinessTravelTravel_Rarely  2.86198583   17
## 25    JobRoleManufacturing Director  2.85797405   18
## 19                       HourlyRate  2.69974091   19
## 23     JobRoleLaboratory Technician  2.51599837   20
## 26         JobRoleResearch Director  2.43238637   21
## 3                               Age  2.42424876   22
## 44                   YearsAtCompany  2.12687585   23
## 29      JobRoleSales Representative  2.03535340   24
## 16                   EmployeeNumber  1.29547658   25
## 11      EducationFieldLife Sciences  1.28483457   26
## 13            EducationFieldMedical  1.25422723   27
## 12          EducationFieldMarketing  1.23724515   28
## 33                     MonthlyIncome  1.12209532   29
## 6                          DailyRate  1.07822391   30
## 18                       GenderMale  0.91844351   31
## 34                     MonthlyRate  0.87965822   32
## 14              EducationFieldOther  0.85418284   33
## 27        JobRoleResearch Scientist  0.66623328   34
## 15   EducationFieldTechnical Degree  0.58716066   35
## 38                PerformanceRating  0.54856660   36
## 28           JobRoleSales Executive  0.49217697   37
## 2                                ID  0.38600491   38
## 37                PercentSalaryHike  0.32506224   39
## 40                StockOptionLevel  0.28262683   40
## 24                   JobRoleManager  0.25649774   41
## 21                         JobLevel  0.17266157   42
## 10                       Education  0.15417643   43
## 22           JobRoleHuman Resources  0.01652359   44
## 7  DepartmentResearch & Development  0.01526688   45
## 8                    DepartmentSales  0.01521545   46
## 1                        (Intercept)  0.01189552   47
```

```
barplot(df$p_values,
        main = "P-values of Regression Coefficients less than the significance level 0.05",
        xlab = "Variables",
        ylab = "P-value",
        names.arg = df$variable_names,
        las = 2,  # Rotate x-axis labels vertically for better readability
        col = "steelblue",  # Set color of bars
        ylim = c(exp(0.05) * -1, max(sorted_df$p_values) * 1.2)  # Set ylim from the significanc
e level to the max p-values

)
```




```
library(ggplot2)
ggplot(sorted_df,aes(variable_names,p_values, fill = ifelse(p_values > (exp(0.05) * -1), "Positi
ve", "Negative"))) + #filtering just the highly significant p-values
  geom_bar(stat="identity", fill = "skyblue") +
  #geom_text(aes(label = variable_names), vjust = -0.5) +  # Add text labels on top of bars
  scale_fill_manual(values = c("Positive" = "skyblue", "Negative" = "salmon")) +
  labs(x = "Variables", y = "P-values", title = "P-values of Regression Coefficients less than t
he significance level 0.05") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))  # Rotate x-axis labels fo
r better readability
```



From the plot above, we can see that the top 5 highly significant values with respect to the response variable Attrition in terms of its relative most significant to least significant variables are OverTime, JobInvolvement, JobSatisfaction, NumCompaniesWorked, YearsSinceLastPromotion

USING FORWARD SELECTION TO GET THE VARIABLE WITH THE BEST AUC METRIC FOR THE PREDICTION MODEL ON THE ATTRITION VARIABLE I pLAN TO USE ONLY NUMERIC VARIABLES FOR KNN

I plan on using the same strategy as for our regression model. This time i shall look at the Area under the curve (AUROC), which represents of the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) as the classification threshold is varied. Ill be looking for the variables that provides the maximum auroc using a general logistiic model

GETTING THE 1ST VARIABLE:

Im looking for the perfect (Attrition ~ dependent variable) combination to get the maximum auroc score

```
Talent_Clean <- Talent_Train %>% select(Age, DailyRate, DistanceFromHome, Education, EmployeeCou
nt, EmployeeNumber, EnvironmentSatisfaction, HourlyRate, JobInvolvement, JobLevel, JobSatisfacti
on, MonthlyIncome, MonthlyRate, NumCompaniesWorked, PercentSalaryHike, PerformanceRating, Relati
onshipSatisfaction, StandardHours, StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, W
orkLifeBalance, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManage
r, Over18_binary, Attrition) #selecting only numeric variables

# Forward Selection # 1
set.seed(21)
vars <- names(Talent_Clean)
vars <- vars[vars!="Attrition"] #iterating thru variables that are not "Attrition"
vars <- vars[vars != "MonthlyIncome"] #iterating thru variables that are not "MonthlyIncome"
num_vars <- length(vars)
var_aucs <- data.frame("vars" = vars)
num_folds <- 10
for (j in 1:num_vars) {
  var <- vars[j]
  #print(var)
  folds <- createFolds(Talent_Clean$Attrition, k = num_folds)
  auc_scores <- numeric(num_folds)
  for (i in 1:num_folds) {
    train_indices <- unlist(folds[-i])
    test_indices <- unlist(folds[i])
    train <- Talent_Clean[train_indices, ]
    test <- Talent_Clean[test_indices, ]
    form <- as.formula(paste("Attrition ~ ",var,sep=""))
    model <- glm(form, data = train, family = "binomial")
    predictions <- predict(model, newdata = test, type = "response")
    roc <- roc(response=test$Attrition,predictor=predictions,levels=c("No", "Yes"),direction =
">")
    auc_scores[i] <- auc(roc) #calculating the area under the curve
  }
  var_aucs$auc[var_aucs$var == var] <- mean(auc_scores)
}

max_auc = max(var_aucs$auc) #finding the max_auc
max_auc_variable <- var_aucs$vars[which.max(var_aucs$auc)]

cat("optimum Variable is ", max_auc_variable, " with a maximum auc of ", max_auc)
```

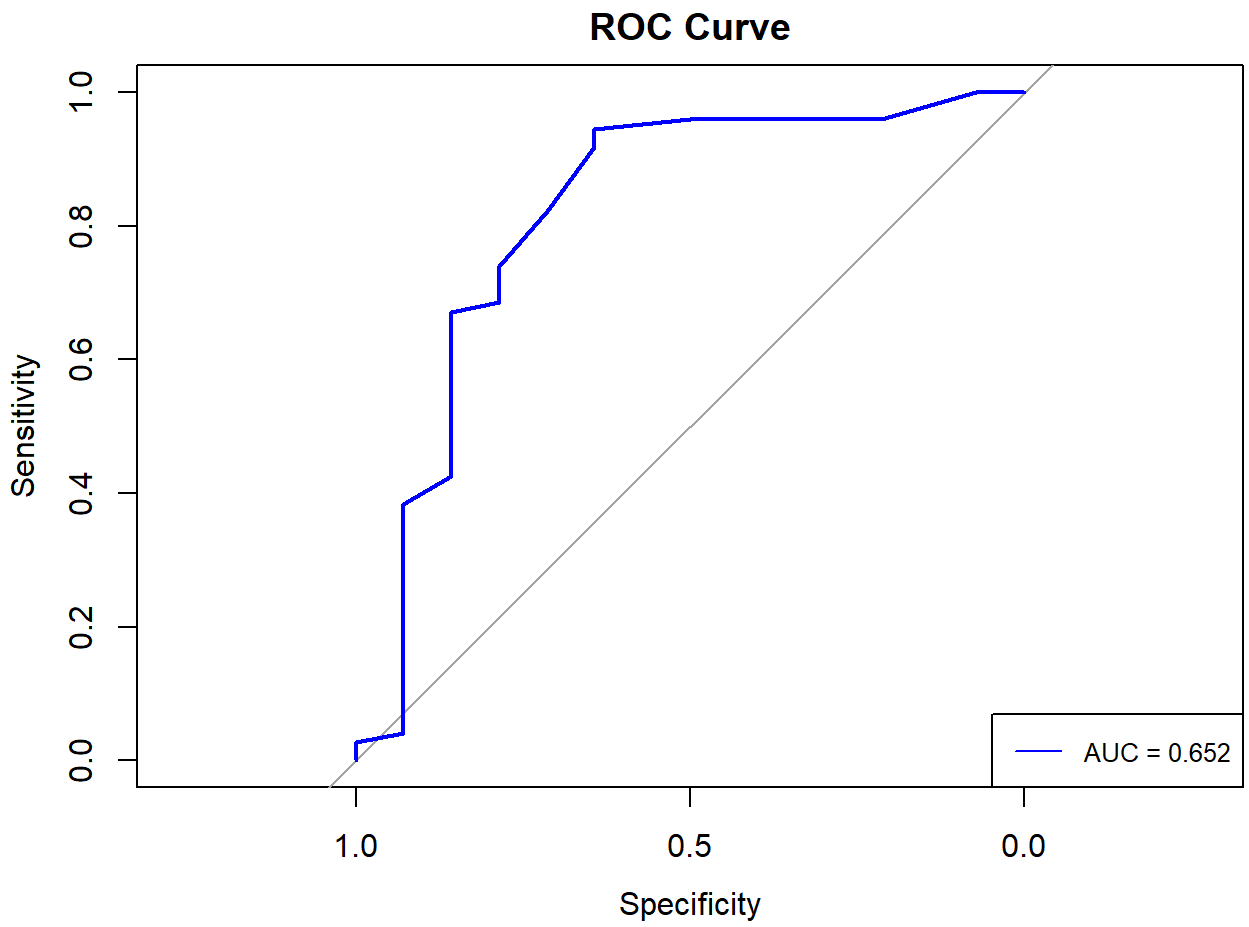
```
## optimum Variable is  TotalWorkingYears  with a maximum auc of  0.6523973
```

```
#Plotting an auc curve
form <- Attrition ~ TotalWorkingYears
model <- glm(form, data = train, family = "binomial")
predictions <- predict(model, newdata = test, type = "response")
roc <- roc(test$Attrition, predictions)
```

```
## Setting levels: control = Yes, case = No
```

```
## Setting direction: controls < cases
```

```
plot(roc, main = "ROC Curve", col = "blue", lwd = 2)
# Add AUC to the plot
legend("bottomright", legend = paste("AUC =", round(max_auc, 3)), col = "blue", lty = 1, cex =
0.8)
```



optimum Variable is TotalWorkingYears with a maximum auc of 0.6523973

GETTIONG THE SECOND VARIABLE

Next ill look for the optimumm variable to add to our classification model (Attrition ~ TotalWorkingYears) in order to provide the maximum auc score

```
# Forward Selection # 1
set.seed(21)
vars <- names(Talent_Clean)
vars <- vars[vars!="Attrition"] #iterating thru variables that are not "Attrition"
vars <- vars[vars != "MonthlyIncome"] #iterating thru variables that are not "MonthlyIncome"
vars <- vars[vars != "TotalWorkingYears "] #iterating thru variables that are not "TotalWorkingY
ears "
num_vars <- length(vars)
var_aucs <- data.frame("vars" = vars)
num_folds <- 10
numks = 60
for (j in 1:num_vars) {
  var <- vars[j]
  #print(var)
  folds <- createFolds(Talent_Clean$Attrition, k = num_folds)
  auc_scores <- numeric(num_folds)
  for (i in 1:num_folds) {
    train_indices <- unlist(folds[-i])
    test_indices <- unlist(folds[i])
    train <- Talent_Clean[train_indices, ]
    test <- Talent_Clean[test_indices, ]
    form <- as.formula(paste("Attrition ~ TotalWorkingYears  + ",var,sep=""))
    model <- glm(form, data = train, family = "binomial")
    predictions <- predict(model, newdata = test, type = "response")
    roc <- roc(response=test$Attrition,predictor=predictions,levels=c("No", "Yes"),direction =
">")
    auc_scores[i] <- auc(roc) #calculating the area under the curve
  }
  var_aucs$auc[var_aucs$var == var] <- mean(auc_scores)
}

max_auc = max(var_aucs$auc) #finding the max_auc
max_auc_variable <- var_aucs$vars[which.max(var_aucs$auc)]

cat("optimum Variable is ", max_auc_variable, " with a maximum auc of ", max_auc)
```

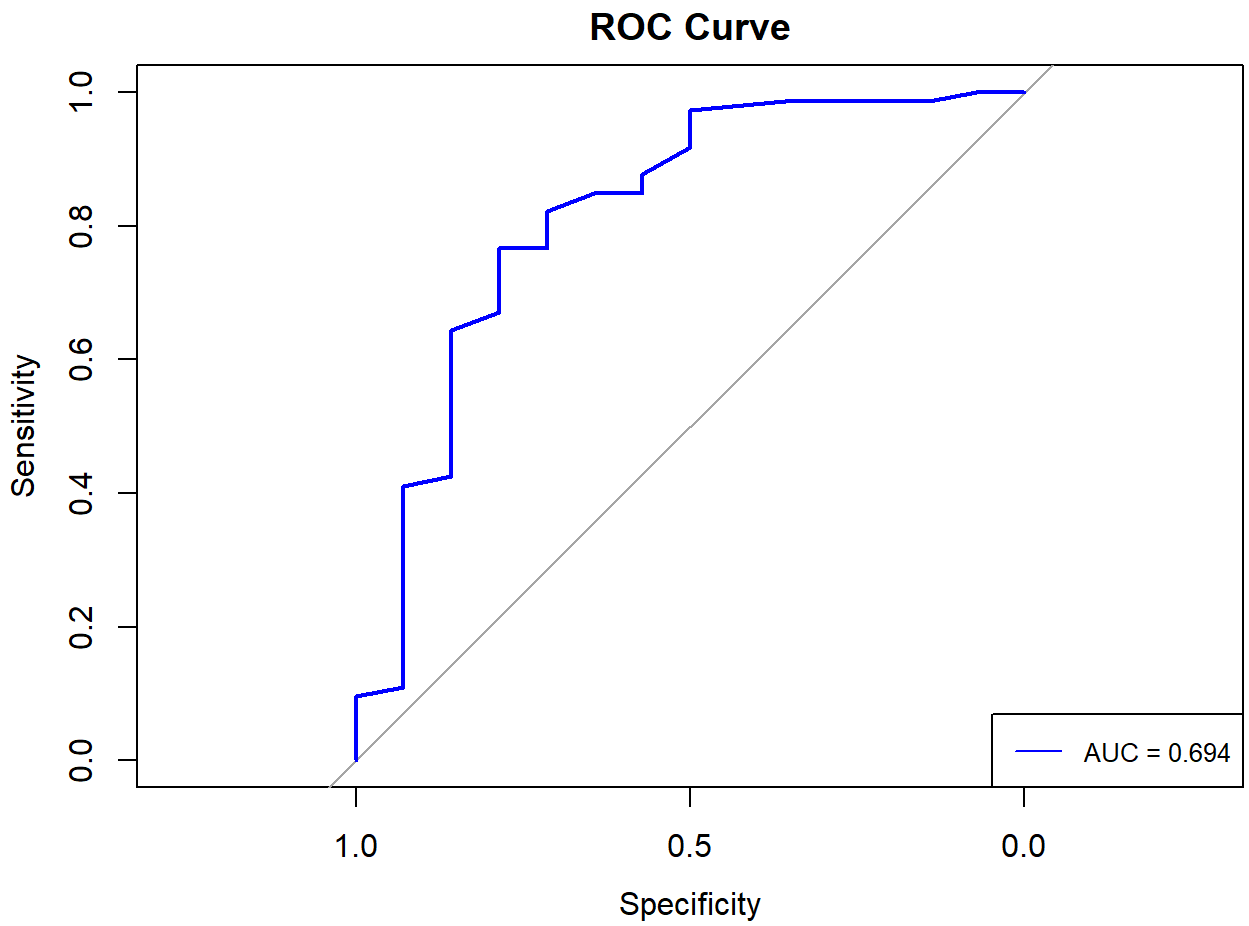
```
## optimum Variable is  StockOptionLevel  with a maximum auc of  0.6940313
```

```
#Plotting an auc curve
form <- Attrition ~ TotalWorkingYears + StockOptionLevel
model <- glm(form, data = train, family = "binomial")
predictions <- predict(model, newdata = test, type = "response")
roc <- roc(test$Attrition, predictions)
```

```
## Setting levels: control = Yes, case = No
```

```
## Setting direction: controls < cases
```

```
plot(roc, main = "ROC Curve", col = "blue", lwd = 2)
# Add AUC to the plot
legend("bottomright", legend = paste("AUC =", round(max_auc, 3)), col = "blue", lty = 1, cex =
0.8)
```



optimum Variable is StockOptionLevel with a maximum auc of 0.6940313

GETTIONG THE THIRD VARIABLE

Next ill look for the optimumm variable to add to our classification model (Attrition ~ TotalWorkingYears + StockOptionLevel) in order to provide the maximum auc score

```
# Forward Selection # 1
set.seed(21)
vars <- names(Talent_Clean)
vars <- vars[vars!="Attrition"] #iterating thru variables that are not "Attrition"
vars <- vars[vars != "MonthlyIncome"] #iterating thru variables that are not "MonthlyIncome"
vars <- vars[vars != "TotalWorkingYears "] #iterating thru variables that are not "TotalWorkingY
ears "
vars <- vars[vars != "StockOptionLevel"] #iterating thru variables that are not "JobRole"
num_vars <- length(vars)
var_aucs <- data.frame("vars" = vars)
num_folds <- 10
numks = 60
for (j in 1:num_vars) {
  var <- vars[j]
  #print(var)
  folds <- createFolds(Talent_Clean$Attrition, k = num_folds)
  auc_scores <- numeric(num_folds)
  for (i in 1:num_folds) {
    train_indices <- unlist(folds[-i])
    test_indices <- unlist(folds[i])
    train <- Talent_Clean[train_indices, ]
    test <- Talent_Clean[test_indices, ]
    form <- as.formula(paste("Attrition ~ TotalWorkingYears  + StockOptionLevel + ",var,sep=""))
    model <- glm(form, data = train, family = "binomial")
    predictions <- predict(model, newdata = test, type = "response")
    roc <- roc(response=test$Attrition,predictor=predictions,levels=c("No", "Yes"),direction =
">")
    auc_scores[i] <- auc(roc) #calculating the area under the curve
  }
  var_aucs$auc[var_aucs$var == var] <- mean(auc_scores)
}

max_auc = max(var_aucs$auc) #finding the max_auc
max_auc_variable <- var_aucs$vars[which.max(var_aucs$auc)]

cat("optimum Variable is ", max_auc_variable, " with a maximum auc of ", max_auc)
```

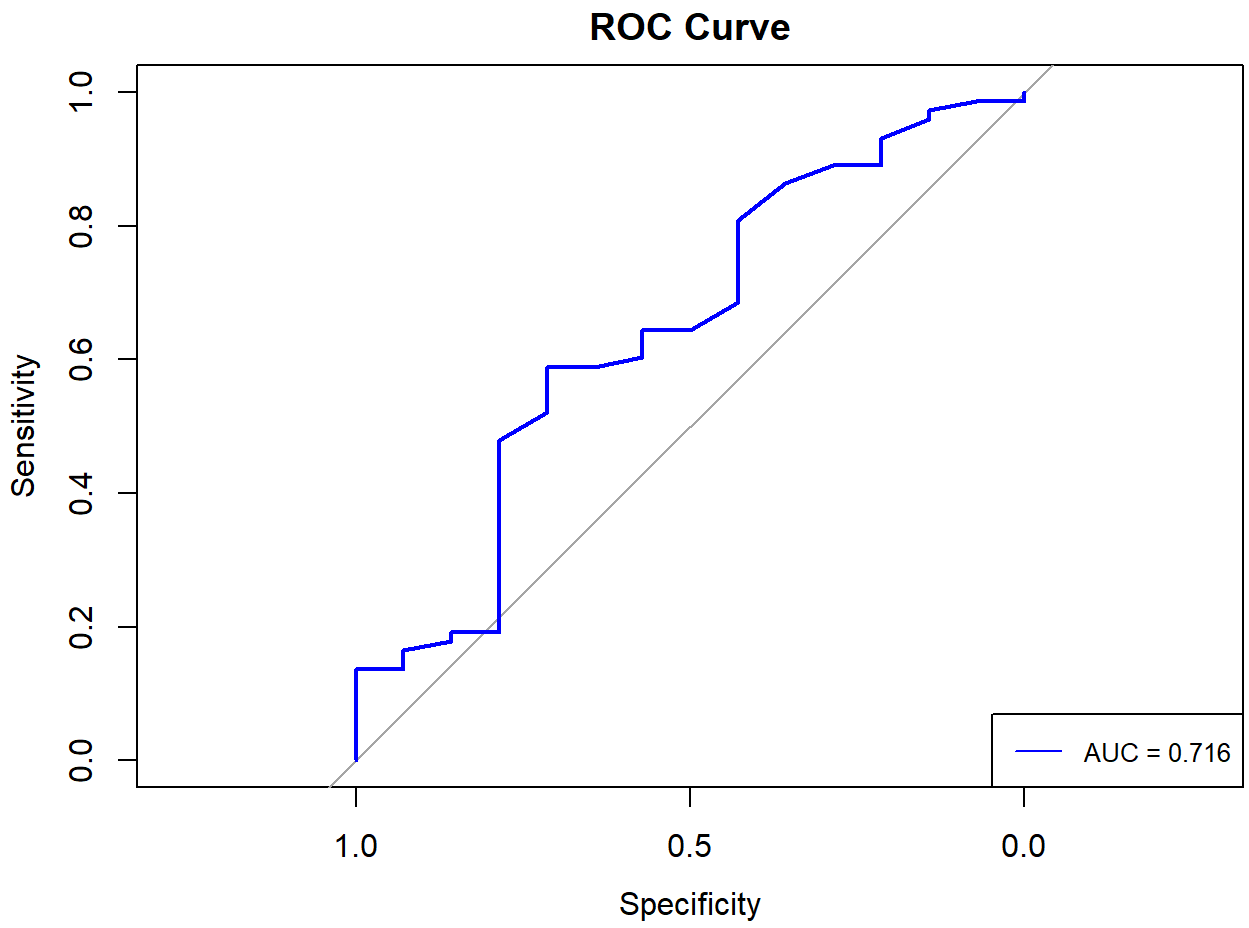
```
## optimum Variable is  JobInvolvement  with a maximum auc of  0.7157045
```

```
#Plotting an auc curve
form <- Attrition ~ TotalWorkingYears + StockOptionLevel + JobInvolvement
model <- glm(form, data = train, family = "binomial")
predictions <- predict(model, newdata = test, type = "response")
roc <- roc(test$Attrition, predictions)
```

```
## Setting levels: control = Yes, case = No
```

```
## Setting direction: controls < cases
```

```
plot(roc, main = "ROC Curve", col = "blue", lwd = 2)
# Add AUC to the plot
legend("bottomright", legend = paste("AUC =", round(max_auc, 3)), col = "blue", lty = 1, cex =
0.8)
```



optimum Variable is JobInvolvement with a maximum auc of 0.7157045

Looking at the Sensitivity & Specificity Metric FOR NB

Next ill be looking at the sensitivity and specificity metric of our classification model using k- Nearest Neigbors (KNN) and Naive Bayes

NAIVE BAYES

FINDING THE THRESHOLD

finding the best threshold for our Naive Bayes Model, in order to provide the best metric. The threshold will be gotten from the maximum F1 score, which is a metric used to evaluate the performance of a binary classification model. It combines both precision and recall into a single metric and is particularly useful when the classes are imbalanced.

```
set.seed(123)
Talent_Clean <- Talent_Train %>% select(Attrition, TotalWorkingYears, StockOptionLevel, JobInvol
vement) #selecting our variables
trainIndices <- sample(seq(1:length(Talent_Clean$Attrition)), round(0.7 * length(Talent_Clean$At
trition)))
train <- Talent_Clean[trainIndices, ]
test <- Talent_Clean[-trainIndices, ]

train$Attrition <-factor(train$Attrition, levels=c('Yes','No')) #factoring the response variable
test$Attrition <-factor(test$Attrition, levels=c('Yes','No')) #factoring the response variable

naive_bayes_model <- naiveBayes(Attrition ~ TotalWorkingYears  + StockOptionLevel + JobInvolveme
nt, data = train)

# Make predictions on the test set
predictions <- predict(naive_bayes_model, test)

# Evaluate model performance
conf_matrix <- confusionMatrix(predictions, test$Attrition)

# Print the confusion matrix
#print(conf_matrix)

# Adjust label levels for better interpretation
test$Attrition <- relevel(test$Attrition, ref = "No")

# Make predictions with probabilities
predictions_with_probs <- predict(naive_bayes_model, test, type = "raw")

# Get probabilities of the positive class ("Yes")
probs <- predictions_with_probs[, "Yes"]

# Define a new threshold
new_threshold <- 0.5

# Create new labels based on the new threshold
new_labels <- ifelse(probs > new_threshold, "Yes", "No")

# Convert the predicted labels to a factor with the same levels as the actual labels
new_labels_factor <- factor(new_labels, levels = levels(test$Attrition))

# Evaluate model performance with the new threshold
new_conf_matrix <- confusionMatrix(new_labels_factor, test$Attrition)

# Print the confusion matrix with the new threshold
#print(new_conf_matrix)

# Calculate the macro F1 score with the new threshold
macro_f1_new_threshold <- mean(c(new_conf_matrix[4]$byClass["F1"], conf_matrix[4]$byClass["F
1"]))
```

```
# Print the macro F1 score with the new threshold
print(macro_f1_new_threshold)
```

```
## [1] 0.5725983
```

Our optimum threshold is 0.5725983

Looking at the Metrics

```
test$Attrition = relevel(test$Attrition, ref = 'Yes')

# Train a Naive Bayes model
naive_bayes_model <- naiveBayes(Attrition ~ TotalWorkingYears  + StockOptionLevel + JobInvolveme
nt, data = train)

# Get predicted probabilities for the positive class (assuming 'Yes' is the positive class)
predicted_probabilities <- predict(naive_bayes_model, test, type = "raw")[, "Yes"]

# Define threshold
threshold <- macro_f1_new_threshold  # Adjust as needed

# Adjust predictions based on threshold
adjusted_predictions <- ifelse(predicted_probabilities > threshold, "Yes", "No")

# Evaluate model performance
adjusted_predictions <- factor(adjusted_predictions, levels = levels(test$Attrition))

conf_matrix <- confusionMatrix(table(adjusted_predictions, test$Attrition))
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##
## adjusted_predictions Yes  No
##                  Yes   6   3
##                  No   46 206
##
##                Accuracy : 0.8123
##                  95% CI : (0.7595, 0.8578)
##     No Information Rate : 0.8008
##     P-Value [Acc > NIR] : 0.3542
##
##                   Kappa : 0.1465
##
##  Mcnemar's Test P-Value : 1.973e-09
##
##             Sensitivity : 0.11538
##             Specificity : 0.98565
##          Pos Pred Value : 0.66667
##          Neg Pred Value : 0.81746
##              Prevalence : 0.19923
##          Detection Rate : 0.02299
##    Detection Prevalence : 0.03448
##       Balanced Accuracy : 0.55052
##
##        'Positive' Class : Yes
##
```

```
accuracy_nb <- conf_matrix$overall["Accuracy"]
sensitivity_nb <- conf_matrix$byClass["Sensitivity"]
specificity_nb <- conf_matrix$byClass["Specificity"]
```

The sensitivity and specificity were pretty low, past our target of > 0.6 for both sensitivity and specificity. We got sensitivity to be : 0.1153846 (This is due to the low amount of Nos.) , and specificity as : 0.9856459. Accuracy is 0.5505 and Pvalue is 1.94e-09 < sig level, which is Highly significant

#KNN MODEL #### Looking at the Sensitivity & Specificity Metric FOR KNN using undersampling I plan on using undersampling to get our metrics, due to the imbalance between Yes and No attrition rates. I am basically reducing the No dataset, in order to match the Yes dataset, to improve sensitivity.

```
set.seed(123)
Talent_Clean <- Talent_Train %>% select(Attrition, TotalWorkingYears, StockOptionLevel, JobInvol
vement, ID) #selecting our variables
trainIndices <- sample(seq(1:length(Talent_Clean$Attrition)), round(0.7 * length(Talent_Clean$At
trition)))
train <- Talent_Clean[trainIndices, ]
test <- Talent_Clean[-trainIndices, ]

# Sample only a subset of 'No' instances to balance the classes
OnlyNoAttrition <- train %>% filter(Attrition == "No")
numYesAttrition <- sum(train$Attrition == "Yes")
sampled_NoAttrition <- OnlyNoAttrition[sample(seq(1, nrow(OnlyNoAttrition), 1), numYesAttritio
n),]

# Combine sampled 'No' instances with 'Yes' instances to create a balanced dataset
balanced_data <- rbind(train %>% filter(Attrition == "Yes"), sampled_NoAttrition)

dim(balanced_data)
```

```
## [1] 176   5
```

```
classifications = knn(balanced_data[,2:4],train[,2:4], balanced_data[,1], prob = TRUE, k = 5) #
using the F (original dataset) as the test set

table(classifications,train[,1])
```

```
##
## classifications Yes  No
##             Yes  58 137
##             No   30 384
```

```
CM = confusionMatrix(table(classifications,train[,1]), mode = "everything")
CM
```

```
## Confusion Matrix and Statistics
##
##
## classifications Yes  No
##             Yes  58 137
##             No   30 384
##
##                Accuracy : 0.7258
##                  95% CI : (0.6885, 0.7609)
##     No Information Rate : 0.8555
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2632
##
##  Mcnemar's Test P-Value : 2.354e-16
##
##             Sensitivity : 0.65909
##             Specificity : 0.73704
##          Pos Pred Value : 0.29744
##          Neg Pred Value : 0.92754
##               Precision : 0.29744
##                  Recall : 0.65909
##                      F1 : 0.40989
##              Prevalence : 0.14450
##          Detection Rate : 0.09524
##    Detection Prevalence : 0.32020
##       Balanced Accuracy : 0.69807
##
##        'Positive' Class : Yes
##
```

```
#Get Macro F1
train$Attrition = relevel(train$Attrition, ref = 'Yes')
classifications = knn(train[,2:4],train[2:4],train[,1], prob = TRUE, k = 5)
CM_Yes = confusionMatrix(table(classifications,train[,1]), mode = "everything") # Note F1

CM_Yes
```

```
## Confusion Matrix and Statistics
##
##
## classifications Yes  No
##             Yes  13   4
##             No   75 517
##
##                Accuracy : 0.8703
##                  95% CI : (0.841, 0.8959)
##     No Information Rate : 0.8555
##     P-Value [Acc > NIR] : 0.1637
##
##                   Kappa : 0.2107
##
##  Mcnemar's Test P-Value : 3.391e-15
##
##             Sensitivity : 0.14773
##             Specificity : 0.99232
##          Pos Pred Value : 0.76471
##          Neg Pred Value : 0.87331
##               Precision : 0.76471
##                  Recall : 0.14773
##                      F1 : 0.24762
##              Prevalence : 0.14450
##          Detection Rate : 0.02135
##    Detection Prevalence : 0.02791
##       Balanced Accuracy : 0.57002
##
##        'Positive' Class : Yes
##
```

```
train$Attrition = relevel(train$Attrition, ref = 'No')
classifications = knn(train[,2:4],train[2:4],train[,1], prob = TRUE, k = 5)
CM_No = confusionMatrix(table(classifications,train[,1]), mode = "everything") # Note F1

CM_No
```

```
## Confusion Matrix and Statistics
##
##
## classifications  No Yes
##             No  518  74
##             Yes   3  14
##
##                Accuracy : 0.8736
##                  95% CI : (0.8445, 0.8989)
##     No Information Rate : 0.8555
##     P-Value [Acc > NIR] : 0.1118
##
##                   Kappa : 0.2307
##
##  Mcnemar's Test P-Value : 1.496e-15
##
##             Sensitivity : 0.9942
##             Specificity : 0.1591
##          Pos Pred Value : 0.8750
##          Neg Pred Value : 0.8235
##               Precision : 0.8750
##                  Recall : 0.9942
##                      F1 : 0.9308
##              Prevalence : 0.8555
##          Detection Rate : 0.8506
##    Detection Prevalence : 0.9721
##       Balanced Accuracy : 0.5767
##
##        'Positive' Class : No
##
```

```
Macro_F1_Under = mean(c(CM_Yes[4]$byClass["F1"],CM_No[4]$byClass["F1"]))
Macro_F1_Under
```

```
## [1] 0.5892183
```

TESTING THE METRIC

Testing for sensitivity and specificity

```
test$Attrition = relevel(test$Attrition, ref = 'Yes')

knn_model <- knn(balanced_data[,2:4], test[, c(2,3,4)], balanced_data[,1], prob = TRUE,  k = 5)

CM = confusionMatrix(table(knn_model,test[,1]), mode = "everything")
CM
```

```
## Confusion Matrix and Statistics
##
##
## knn_model Yes  No
##       Yes  25  60
##       No   27 149
##
##                Accuracy : 0.6667
##                  95% CI : (0.6059, 0.7236)
##     No Information Rate : 0.8008
##     P-Value [Acc > NIR] : 0.9999999
##
##                   Kappa : 0.1564
##
##  Mcnemar's Test P-Value : 0.0006019
##
##             Sensitivity : 0.48077
##             Specificity : 0.71292
##          Pos Pred Value : 0.29412
##          Neg Pred Value : 0.84659
##               Precision : 0.29412
##                  Recall : 0.48077
##                      F1 : 0.36496
##              Prevalence : 0.19923
##          Detection Rate : 0.09579
##    Detection Prevalence : 0.32567
##       Balanced Accuracy : 0.59684
##
##        'Positive' Class : Yes
##
```

```
# Evaluate model performance

accuracy_knn <- CM$overall["Accuracy"]
sensitivity_knn <- CM$byClass["Sensitivity"]
specificity_knn <- CM$byClass["Specificity"]

test$predictions <- knn_model

# Generation our regression for the test dataset
new_data <- test %>% select(predictions, ID)

new_data <- new_data[order(new_data$ID), ]

# Specify the file path where you want to save the file
file_path <- "Case2PredictionsOwolabi Attrition.csv"

# Write the dataset to a CSV file
write.csv(new_data, file = file_path, row.names = FALSE)
```

Specificity is 0.7081., Sensitivity is 0.4808, Accuracy is 0.5944, Pvalue is 4.4e-03 < sig level which is Highly significant. Sadly i was unable to get sensitivity and specificity metric to be both greater than 60%. Despite spending hours,and mental strain on this project. I had to stop at this point, due to almost approaching the deadline. I learnt the important lesson of knowing when to give up and fight my battles later.

If I have more time, i plan to find the perfect classification model, based of finding the variables that maximisies the F1 metric, I believe that might improve specificity and sensitivity metric. Also I plan on looking at the best k value for our kNN model. And experimenrt further with oversampling and undersampling.

MORE EDAS

METRICS COMPARISON

comparing accuracy, sensitivity and specificity metric for both KNN and naive bayes

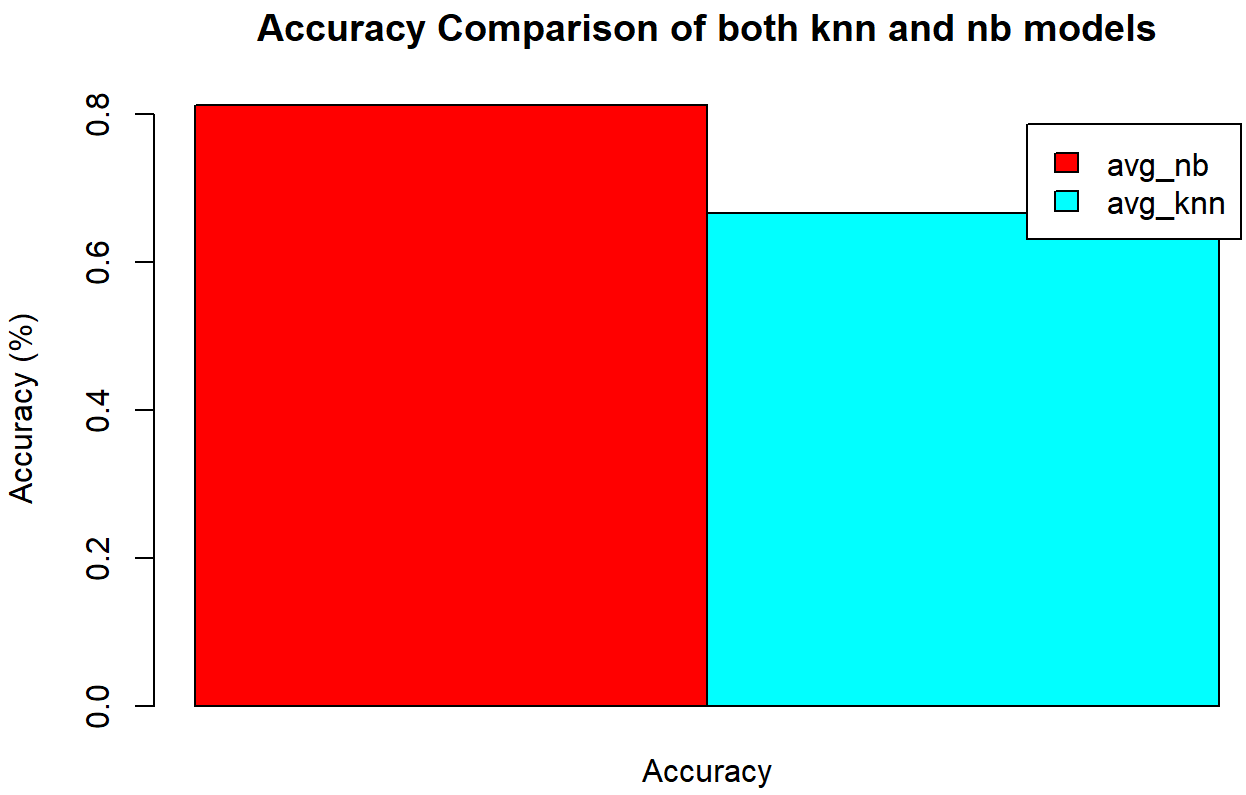
```
results_nb <- data.frame(Accuracy = accuracy_nb, Sensitivity = sensitivity_nb, Specificity =  sp
ecificity_nb, total = sensitivity_nb + specificity_nb)
avg_nb = colMeans(results_nb[, c("Accuracy", "Sensitivity", "Specificity", "total")])
avg_nb
```

```
##    Accuracy Sensitivity Specificity       total
##   0.8122605   0.1153846   0.9856459   1.1010305
```

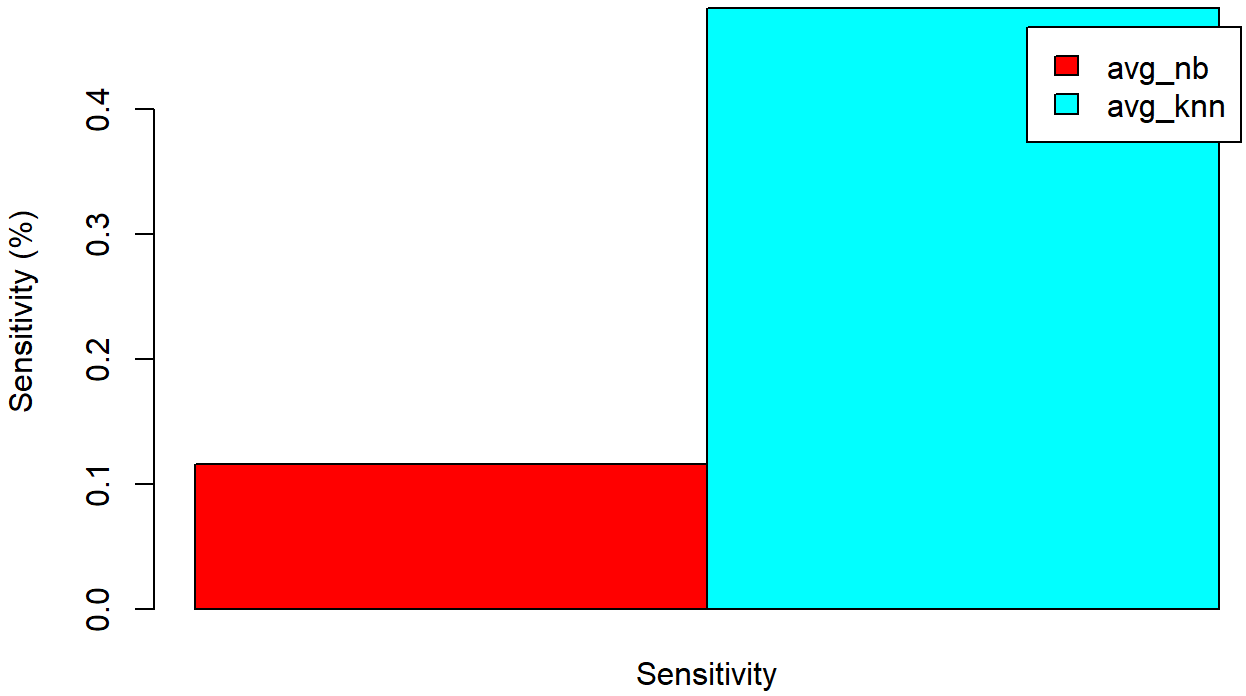
```
results_knn <- data.frame(Accuracy = accuracy_knn, Sensitivity = sensitivity_knn, Specificity =
specificity_knn, total = sensitivity_knn + specificity_knn)
avg_knn = colMeans(results_knn[, c("Accuracy", "Sensitivity", "Specificity", "total")])
avg_knn
```

```
##    Accuracy Sensitivity Specificity       total
##   0.6666667   0.4807692   0.7129187   1.1936879
```

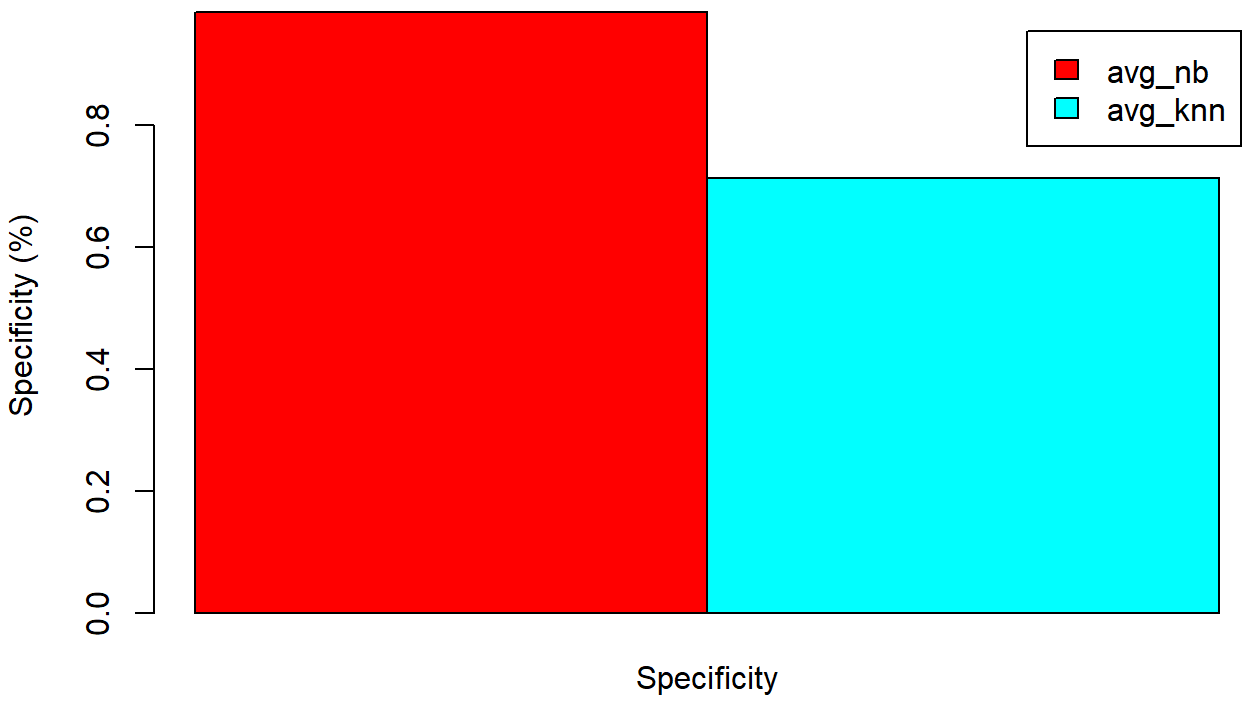
```
#Getting the barchart plots for average analysis
combined_results <- cbind(data.frame(avg_nb), data.frame(avg_knn))


barplot(t(combined_results[1, ]), beside = TRUE, col = rainbow(2),
        main = "Accuracy Comparison of both knn and nb models",
        ylab = "Accuracy (%)", legend.text = names(combined_results[1, ]))
```




```
barplot(t(combined_results[2, ]), beside = TRUE, col = rainbow(2),
        main = "Sensitivity Comparison of both knn and nb models",
        ylab = "Sensitivity (%)", legend.text = names(combined_results[2, ]))
```




```
barplot(t(combined_results[3, ]), beside = TRUE, col = rainbow(2),
        main = "Specificity Comparison of both knn and nb models",
        ylab = "Specificity (%)", legend.text = names(combined_results[3, ]))
```




```
barplot(t(combined_results[4, ]), beside = TRUE, col = rainbow(2),
        main = "Specificity Comparison of both knn and nb models",
        ylab = "Specificity + Sensitivity (%)", legend.text = names(combined_results[4, ]))
```

- Knn

has a higher sensitivity and specificity + sensitivity. - Naïve Bayes has higher specificity and accuracy.

IN CONCLUSION

TIPS FOR FRITO LAY

- Try providing more incentives to employees to improve retention rate
- Let the stock options match the amount of years the employees worked.

FUTURE WORK

- Try to improve sensitivity values for the classification model.
- Explore other variables that might affect the classification and prediction model.
- Try using other classification techniques to predict my model likeSupport Vector Machine (SVM) or RandomForest