Predicting Insurance Company Customer Behavior

Daniel Shang

```r
# The client is an insurance company that provides health insurance. It plans to
# launch a new vehicle insurance service and wants a model to predict whether the
# policy holders (customers) from past year will also be interested in vehicle
# insurance provided by the company. With a model, the company can plan its
# communication and marketing strategy to reach out to those customers,
# optimizing its business model and revenue.
```

```r
# Loaded the necessary packages and read the data from a CSV file
library(ggplot2)
library(dplyr)
```

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

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

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

```r
library(ggthemes)
library(ggcorrplot)
library(mefa4)
```

```
## Loading required package: Matrix

## mefa4 0.3-7   2020-02-28
```

```r
library(e1071)
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```r
library(caret)
```

```
## Loading required package: lattice
```

```r
data = read.csv('C:/Users/34527/Desktop/dataset.csv')
data_raw = read.csv('C:/Users/34527/Desktop/dataset.csv')
```

———————————————————- Data cleaning ———————————————————-

```r
# Check if there is any missing data in the data set

for (i in 1:ncol(data)) {
  print(paste(colnames(data)[i], ':', sum(is.null(data[, i]))))
  print(paste(colnames(data)[i], ':', sum(data[, i] == 'NA')))
  print(paste(colnames(data)[i], ':', sum(data[, i] == 'N/A')))
  print(paste(colnames(data)[i], ':', sum(data[, i] == '')))
}
```

```
## [1] "id : 0"
## [1] "id : 0"
## [1] "id : 0"
## [1] "id : 0"
## [1] "Gender : 0"
## [1] "Gender : 0"
## [1] "Gender : 0"
## [1] "Gender : 0"
## [1] "Age : 0"
## [1] "Age : 0"
```

```
## [1] "Age : 0"
## [1] "Age : 0"
## [1] "Driving_License : 0"
## [1] "Driving_License : 0"
## [1] "Driving_License : 0"
## [1] "Driving_License : 0"
## [1] "Region_Code : 0"
## [1] "Region_Code : 0"
## [1] "Region_Code : 0"
## [1] "Region_Code : 0"
## [1] "Previously_Insured : 0"
## [1] "Previously_Insured : 0"
## [1] "Previously_Insured : 0"
## [1] "Previously_Insured : 0"
## [1] "Vehicle_Age : 0"
## [1] "Vehicle_Age : 0"
## [1] "Vehicle_Age : 0"
## [1] "Vehicle_Age : 0"
## [1] "Vehicle_Damage : 0"
## [1] "Vehicle_Damage : 0"
## [1] "Vehicle_Damage : 0"
## [1] "Vehicle_Damage : 0"
## [1] "Annual_Premium : 0"
## [1] "Annual_Premium : 0"
## [1] "Annual_Premium : 0"
## [1] "Annual_Premium : 0"
## [1] "Policy_Sales_Channel : 0"
## [1] "Policy_Sales_Channel : 0"
## [1] "Policy_Sales_Channel : 0"
## [1] "Policy_Sales_Channel : 0"
## [1] "Vintage : 0"
## [1] "Vintage : 0"
## [1] "Vintage : 0"
## [1] "Vintage : 0"
## [1] "Response : 0"
## [1] "Response : 0"
## [1] "Response : 0"
## [1] "Response : 0"
```

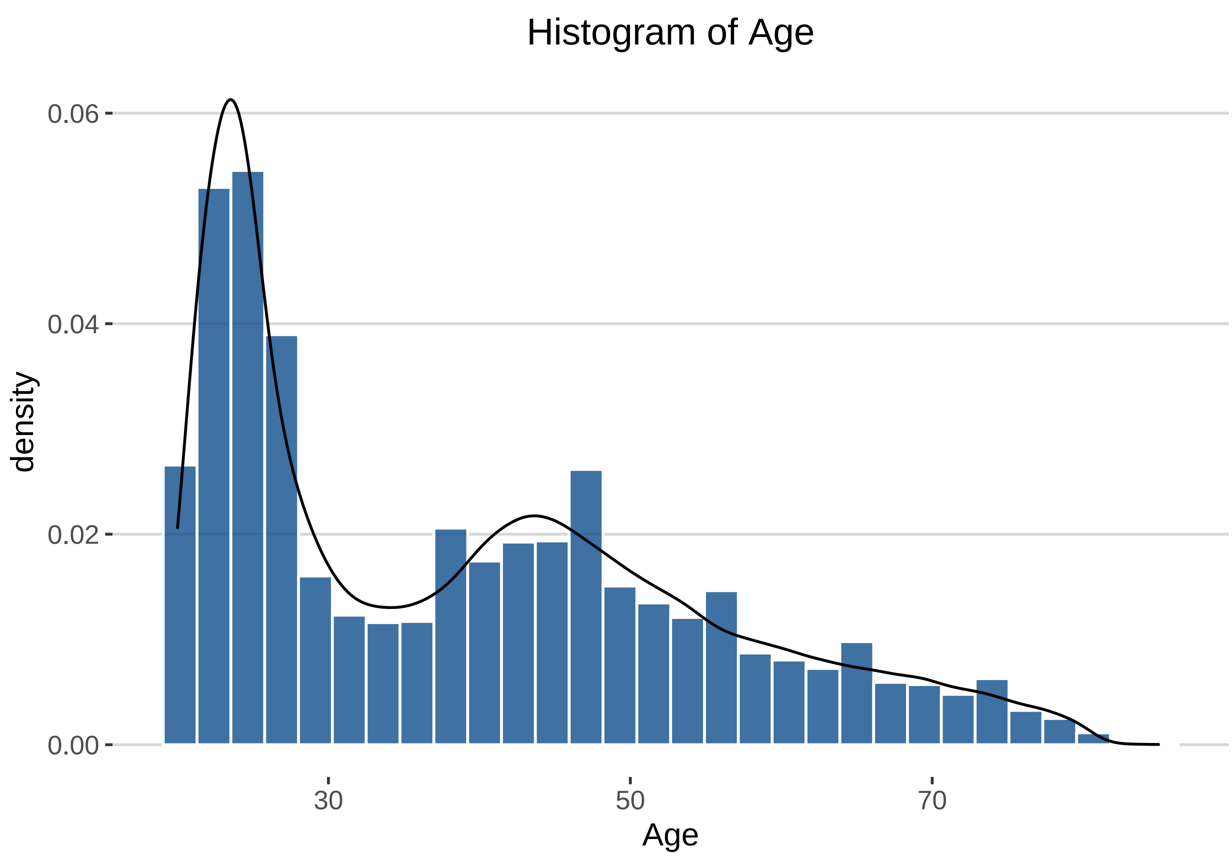
```
# Transform categorical variables to dummy variables for easier analysis and
# better model performance

data[data[, "Vehicle_Damage"] == "Yes", "Vehicle_Damage"] = 1
data[data[, "Vehicle_Damage"] == "No", "Vehicle_Damage"] = 0
data[data[, "Vehicle_Age"] == "< 1 Year", "Vehicle_Age"] = 0
data[data[, "Vehicle_Age"] == "1-2 Year", "Vehicle_Age"] = 1
data[data[, "Vehicle_Age"] == "> 2 Years", "Vehicle_Age"] = 2
data[data[, "Gender"] == "Female", "Gender"] = 0
data[data[, "Gender"] == "Male", "Gender"] = 1
```

```
# Manually remove the data points that is overly unrepresentative. The existence
# of these unrepresentative data points would abort the train/test split and
# other analysis
```

```r
to_be_removed = data %>%
    count(Policy_Sales_Channel) %>%
    group_by(Policy_Sales_Channel) %>%
    filter(n < 5)
data = data[data$Policy_Sales_Channel %notin% to_be_removed$Policy_Sales_Channel,
    ]
data = data[, colnames(data) != "id"]
```

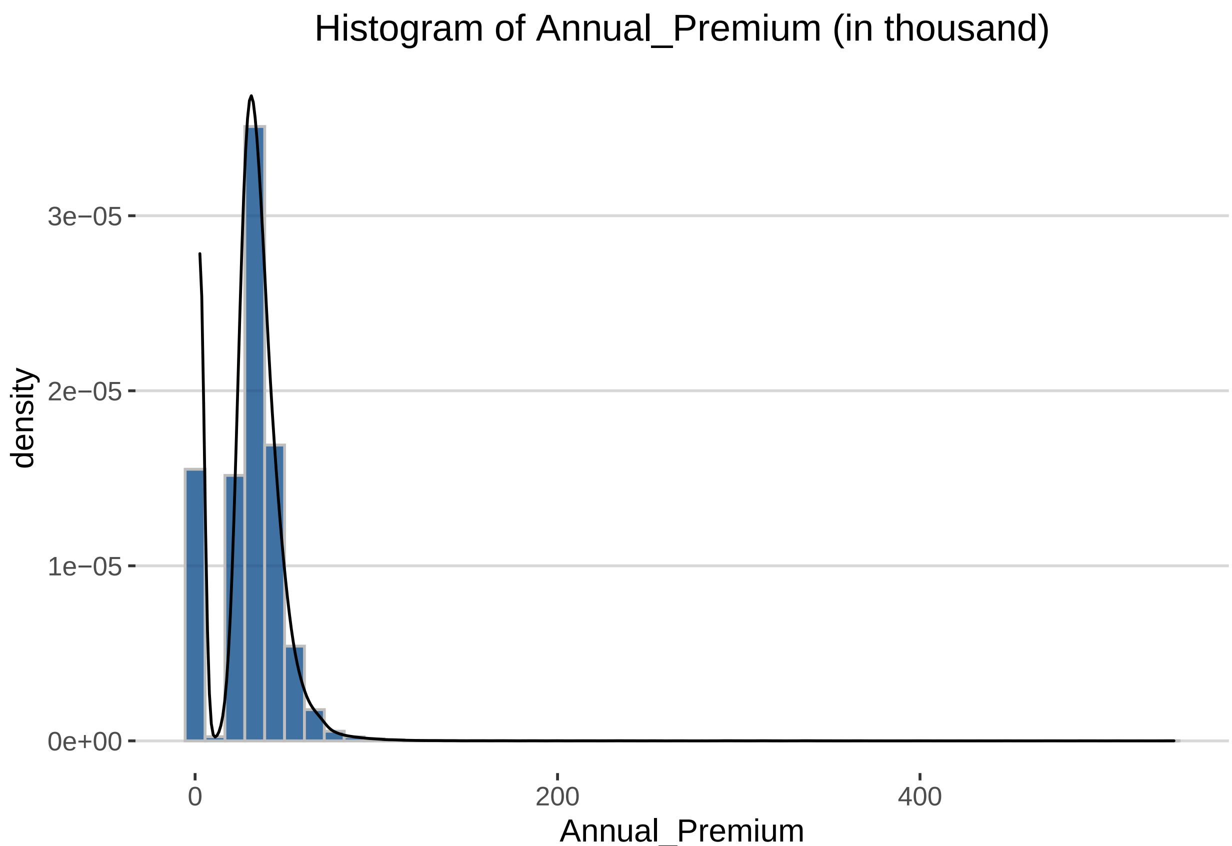
Exploratory Data Analysis

```r
# A histogram showing the distribution of customer age. We can see that the
# customers of the company are relatively young. This could mean that the
# customer's average life-time value is high.

ggplot(data = data, aes(Age)) + geom_histogram(aes(y = ..density..), fill = "dodgerblue4",
    color = "white", alpha = 0.8, bins = 30) + geom_density() + labs(title = "Histogram of Age") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```




```r
# A histogram showing the distribution of annual premium customers currently pay.
# Although most customers' premium falls within a narrow range, there are some
# outliers.

ggplot(data = data, aes(Annual_Premium)) + geom_histogram(aes(y = ..density..), fill = "dodgerblue4",
    color = "gray", alpha = 0.8, bins = 50) + geom_density(adjust = 3) + labs(title = "Histogram of Annu
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc() + scale_x_continuous(labels = function(x)
```

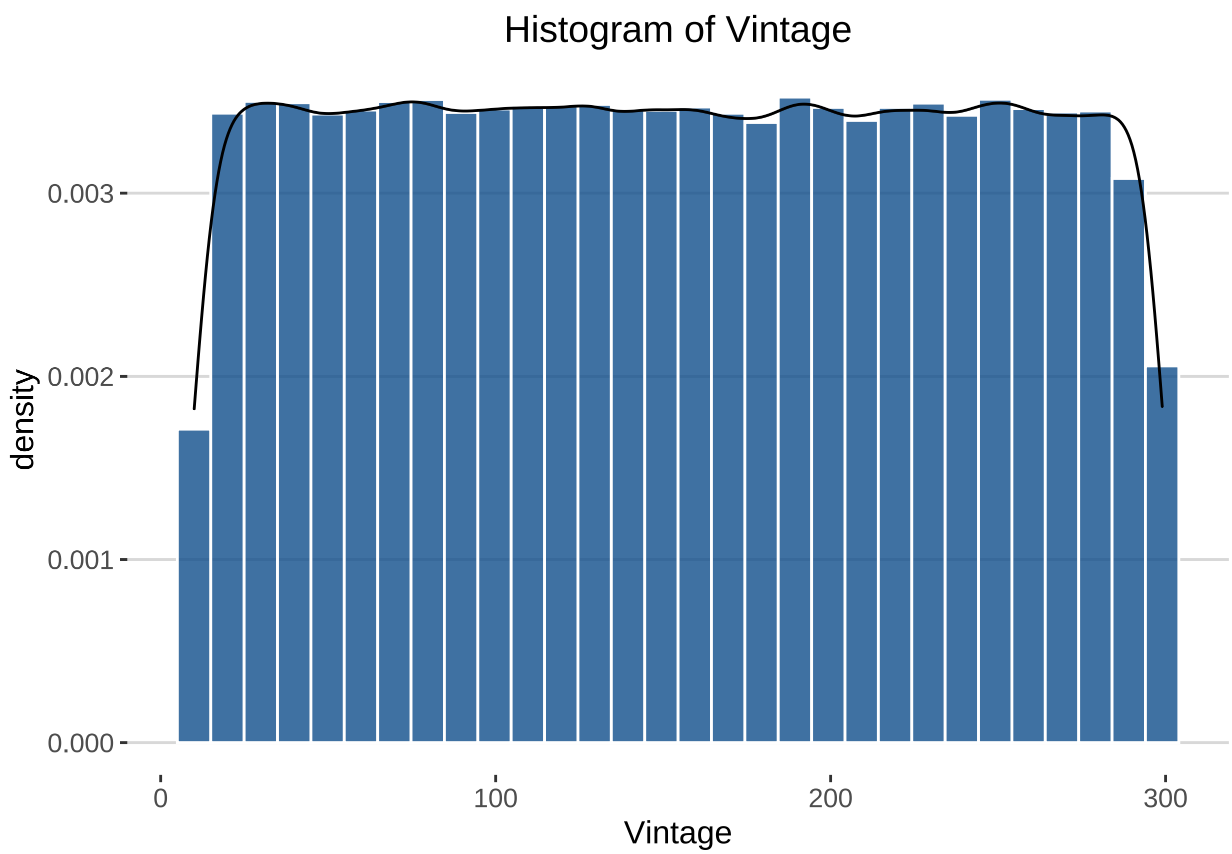
```r
    x/10^3
})
```



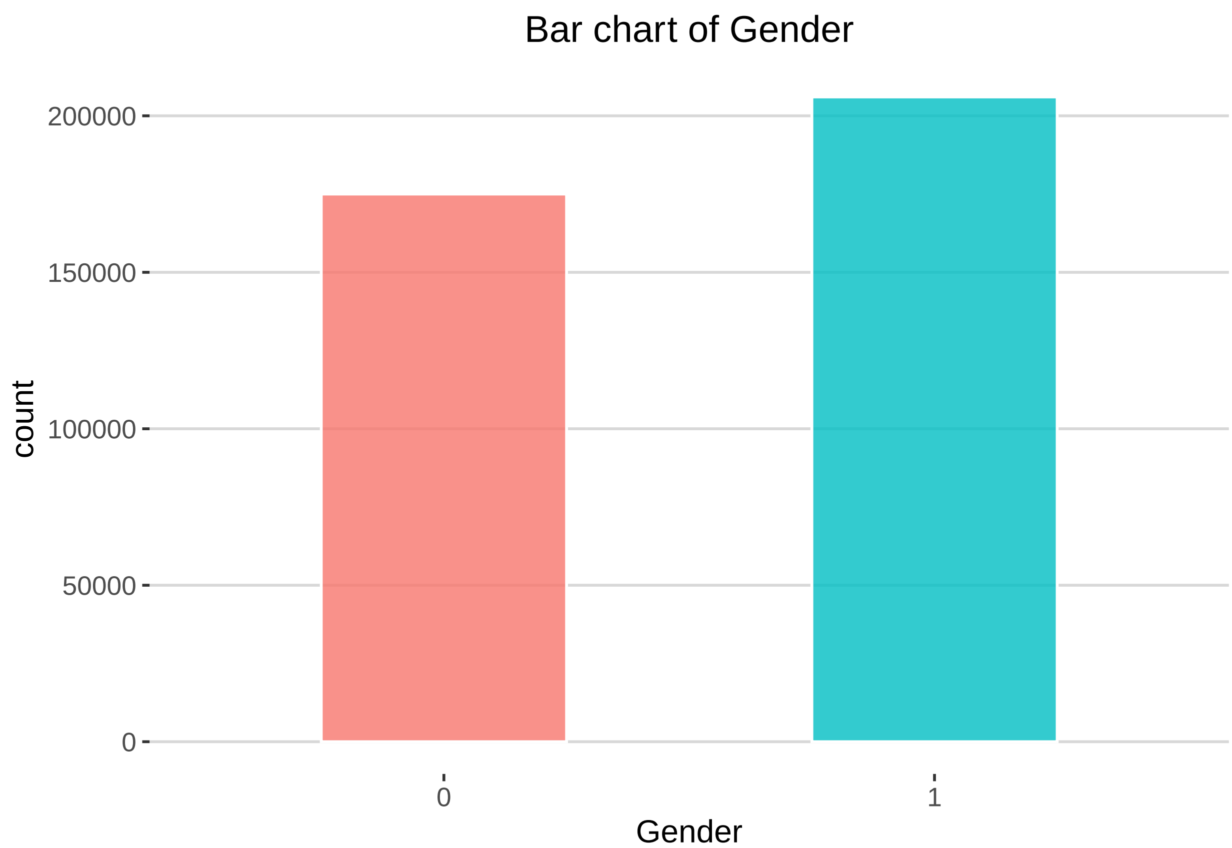

```r
# Summarize the annual premium data to better understand the distribution. From
# the 1st and 3rd quantile, we can tell that most customers pay a premium that
# falls within that range. But some customers are paying a premium as low as
# $2,630 or as high as $540,165

summary(data$Annual_Premium)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2630   24405   31668   30564   39400  540165
```

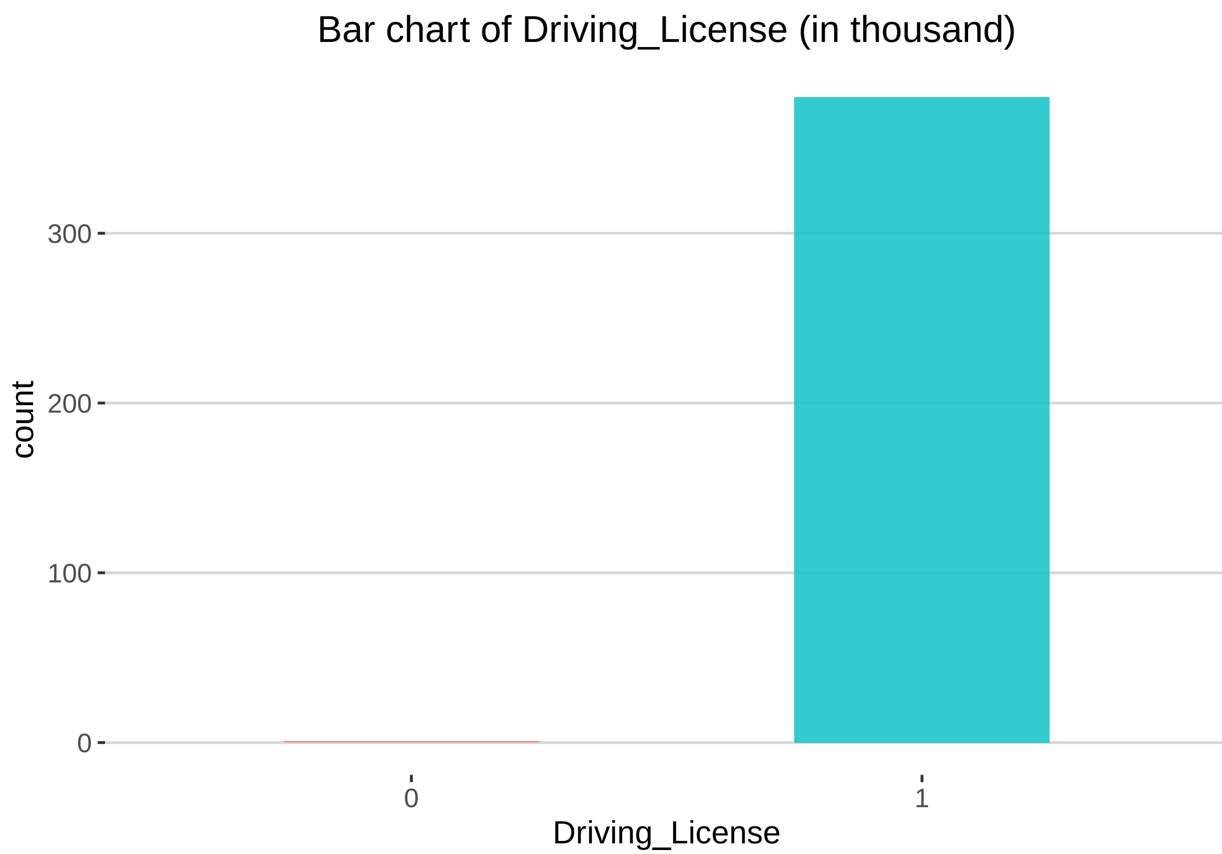
```r
# A histogram showing the distribution of customer vintage (the number of days
# they have been associated with the company). We can see that the customers are
# almost evenly distributed in terms of the vintage.

ggplot(data = data, aes(Vintage)) + geom_histogram(aes(y = ..density..), fill = "dodgerblue4",
    color = "white", alpha = 0.8, bins = 30) + geom_density() + labs(title = "Histogram of Vintage") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



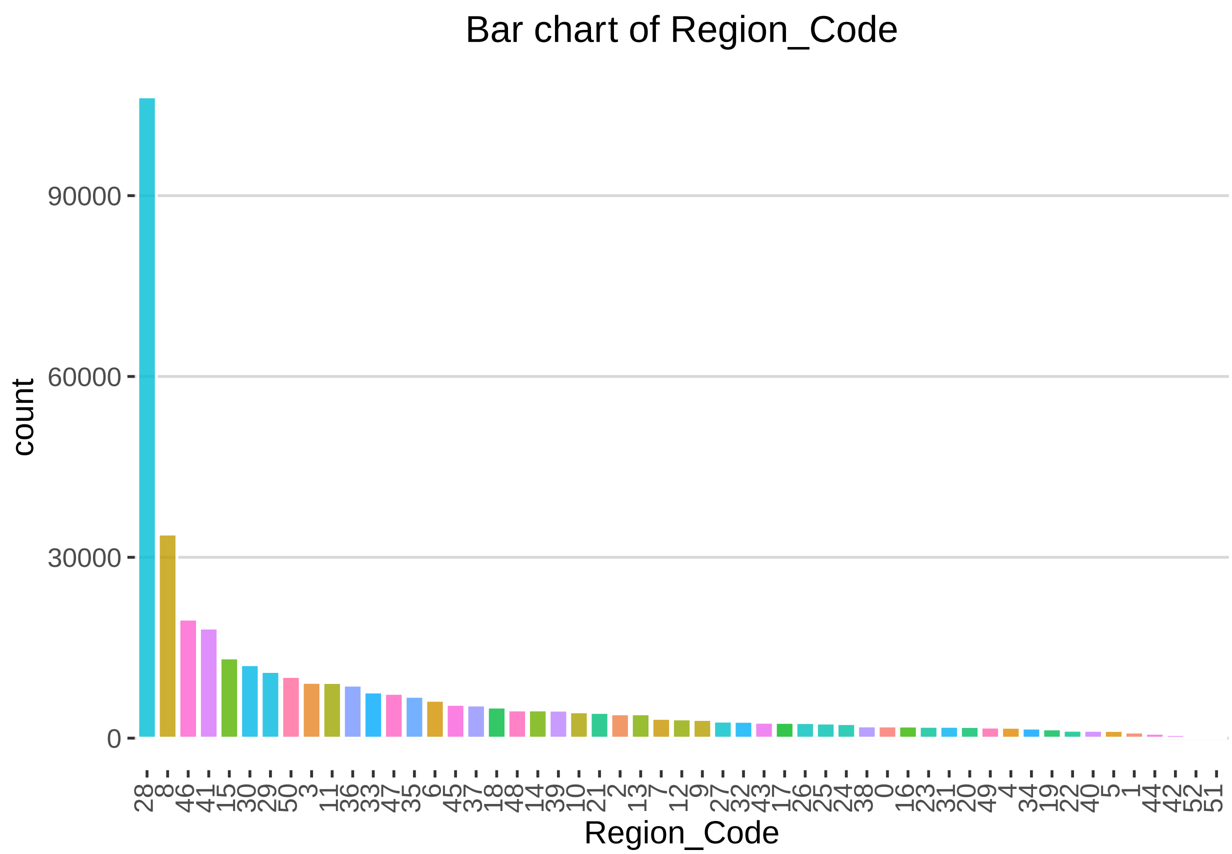

```r
# A bar chart showing the number of customers in each gender group. This shows
# that the customers are relatively evenly distributed in terms of their gender

ggplot(data = data, aes(Gender, fill = Gender)) + geom_bar(width = 0.5, alpha = 0.8,
    color = "white", show.legend = FALSE) + labs(title = "Bar chart of Gender") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



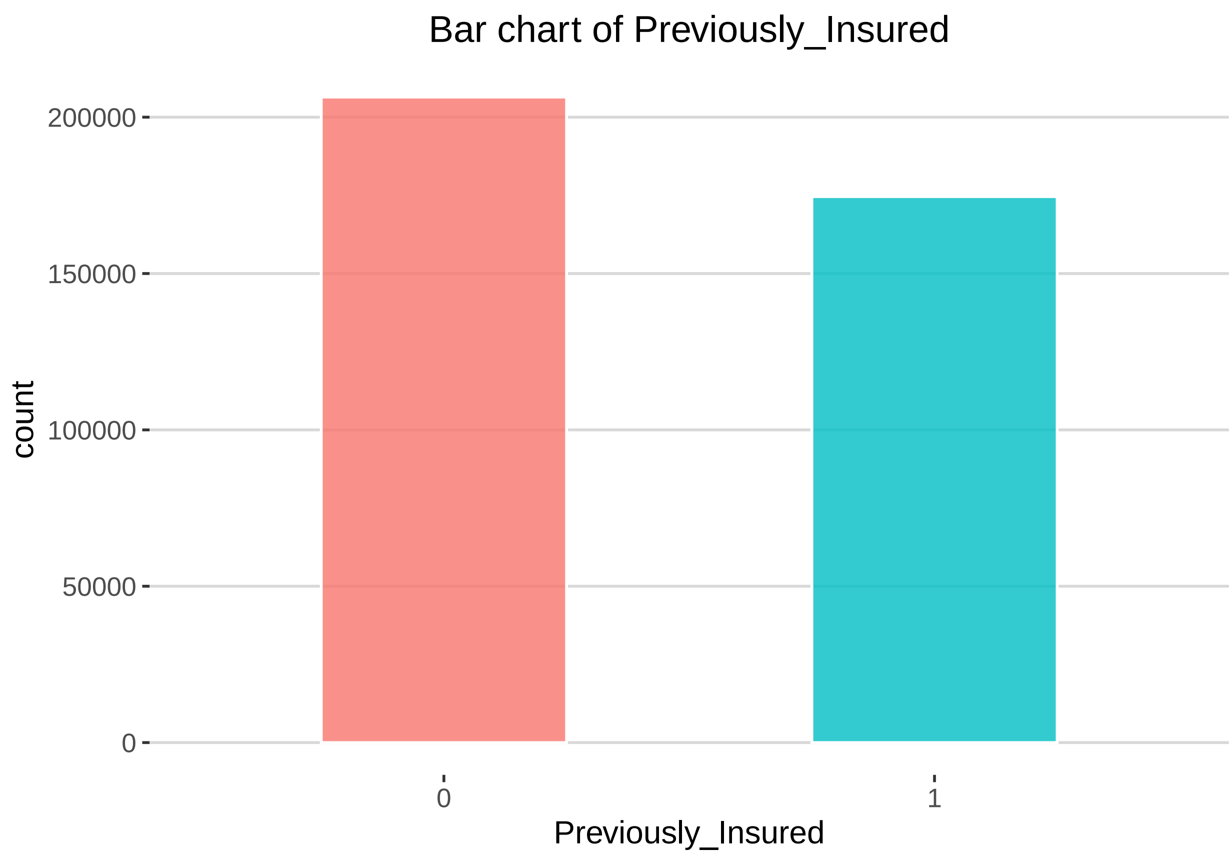

```r
# A bar chart showing the number of customers with and without a driver's
# license. We can see that only a tiny portion of customers do not have a
# driver's license. This indicates a great great start, because those who have a
# driver's license are likely to have a car, thereby likely to need vehicle
# insurance.

ggplot(data = data, aes(factor(Driving_License), fill = factor(Driving_License))) +
    geom_bar(width = 0.5, alpha = 0.8, show.legend = FALSE) + labs(title = "Bar chart of Driving_Licens
    x = "Driving_License") + theme(plot.title = element_text(hjust = 0.5), legend.position = "none") +
    scale_y_continuous(labels = function(y) {
        y/10^3
    }) + theme_hc()
```



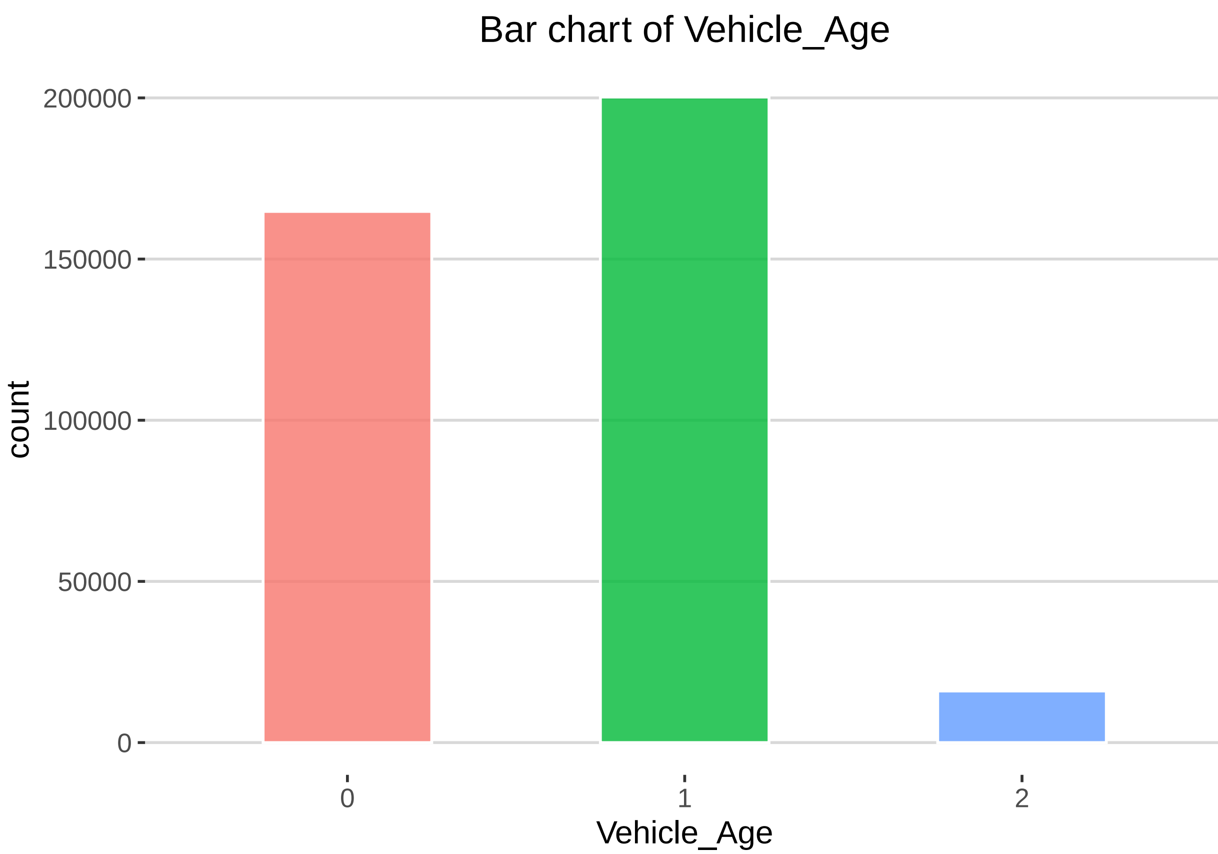

```r
# A bar chart showing the number of customers within each region code. We can see
# that most customers of the company are in region with a code of 28 and 8.
# Therefore, the company may want to allocate more resources to these regions.

ggplot(data = data, aes(x = reorder(Region_Code, Region_Code, function(x) -length(x)),
    fill = factor(Region_Code))) + geom_bar(alpha = 0.8, color = "white", show.legend = FALSE) +
    labs(title = "Bar chart of Region_Code", x = "Region_Code") + theme(plot.title = element_text(hjust
    axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.3)) + theme_hc()
```



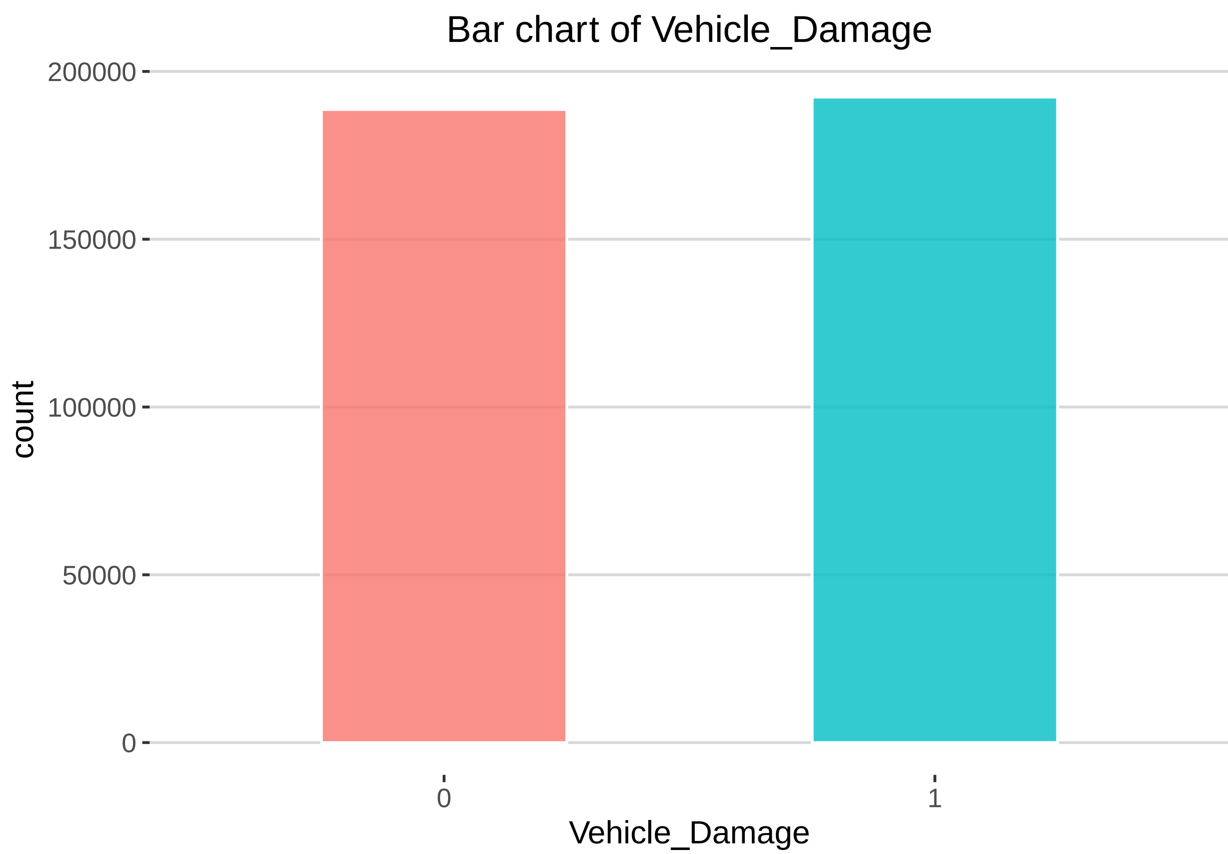

```r
# A bar chart showing the number of customers who are and are not previously
# insured. We can see that more customers are not previously insured than those
# who are previously insured. This may be a good thing, because less previously
# insured customers could convert to more current need for the new vehicle
# insurance.

ggplot(data = data, aes(factor(Previously_Insured), fill = factor(Previously_Insured))) +
    geom_bar(width = 0.5, alpha = 0.8, color = "white", show.legend = FALSE) + labs(title = "Bar chart o
    x = "Previously_Insured") + theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



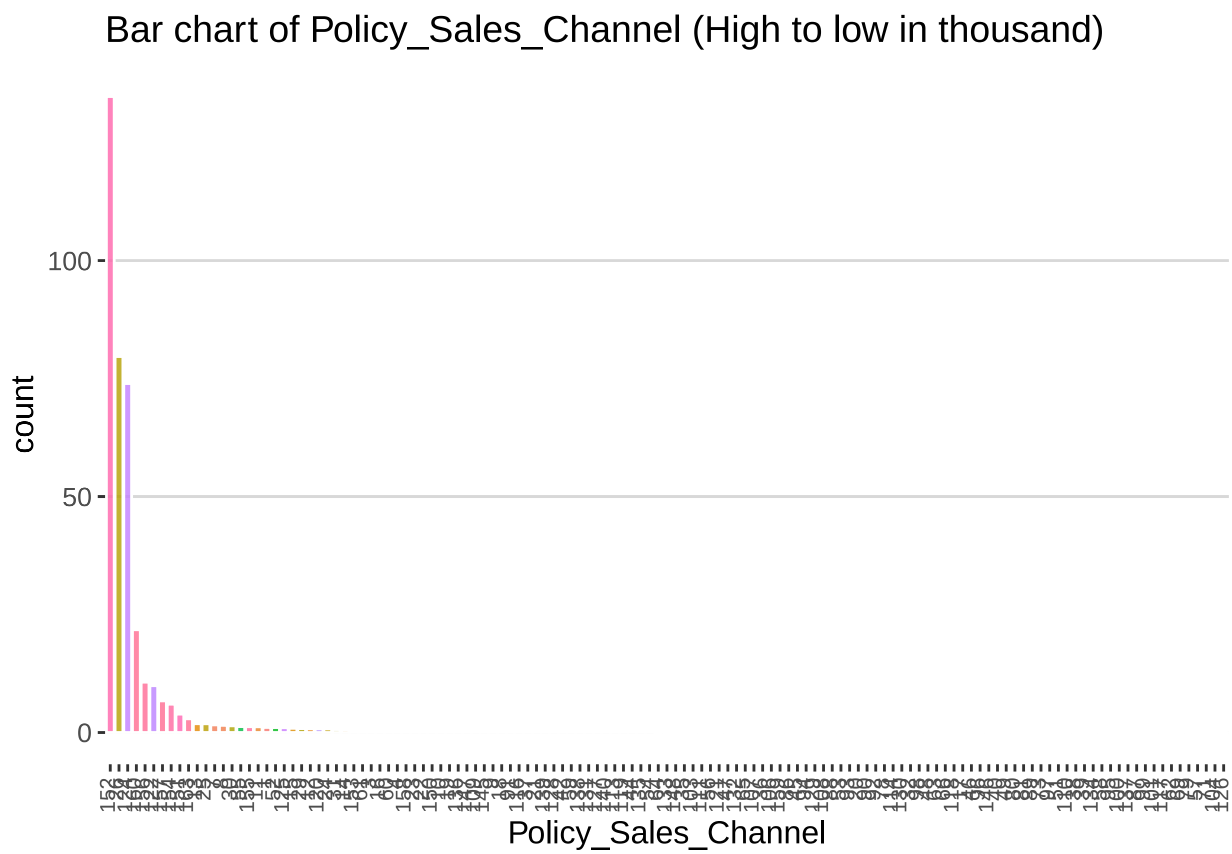

```r
# A bar chart showing the count of customers with a vehicle that falls into each
# of the three vehicle age groups. We can see that most customers have a vehicle
# aged 2 or less years.

ggplot(data = data, aes(factor(Vehicle_Age, ordered = TRUE, levels = c("0", "1",
    "2")), fill = factor(Vehicle_Age))) + geom_bar(width = 0.5, alpha = 0.8, color = "white",
    show.legend = FALSE) + labs(title = "Bar chart of Vehicle_Age", x = "Vehicle_Age") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



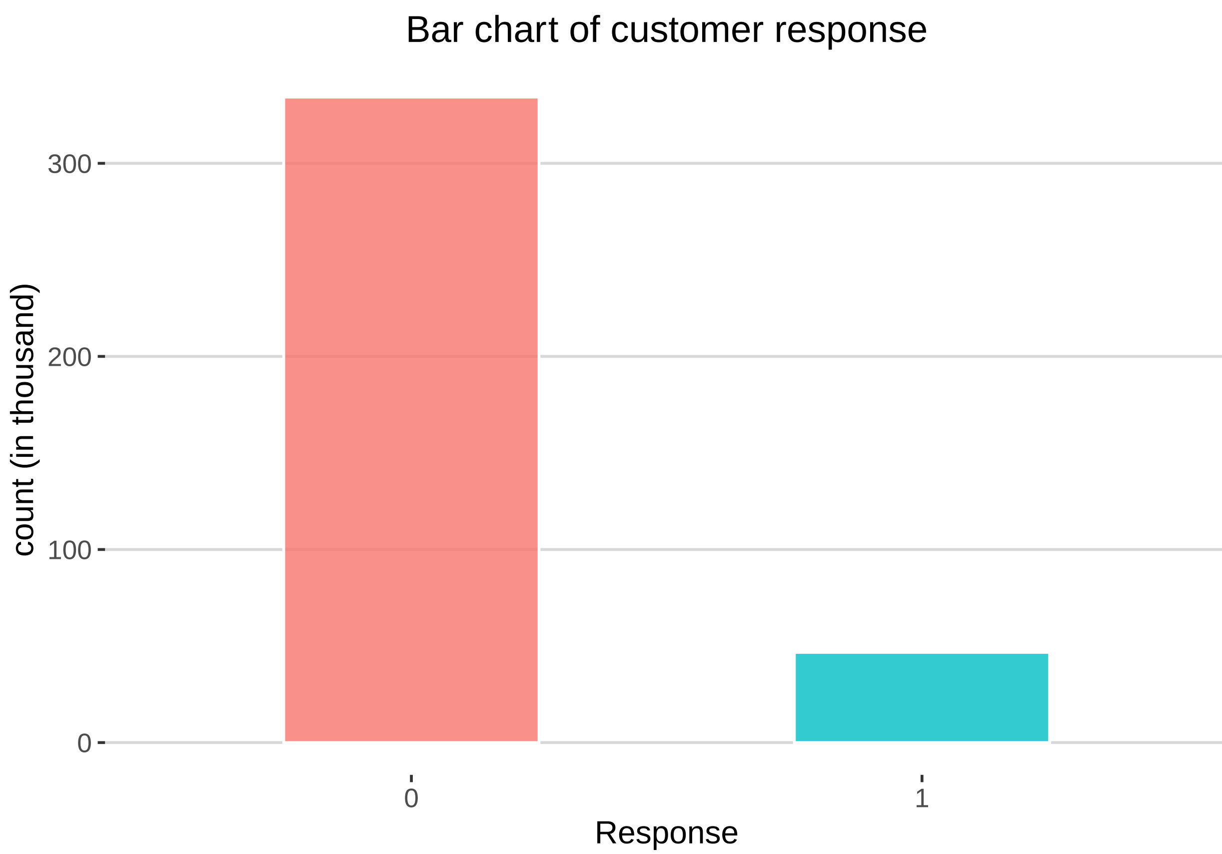

```r
# A bar chart showing the number of customers with and without their vehicle been
# damaged. We can see that slighly more customers have their vehicle damaged
# before. This indicates that the company should be careful about pricing.

ggplot(data = data, aes(Vehicle_Damage, fill = Vehicle_Damage)) + geom_bar(width = 0.5,
    alpha = 0.8, color = "white", show.legend = FALSE) + labs(title = "Bar chart of Vehicle_Damage") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



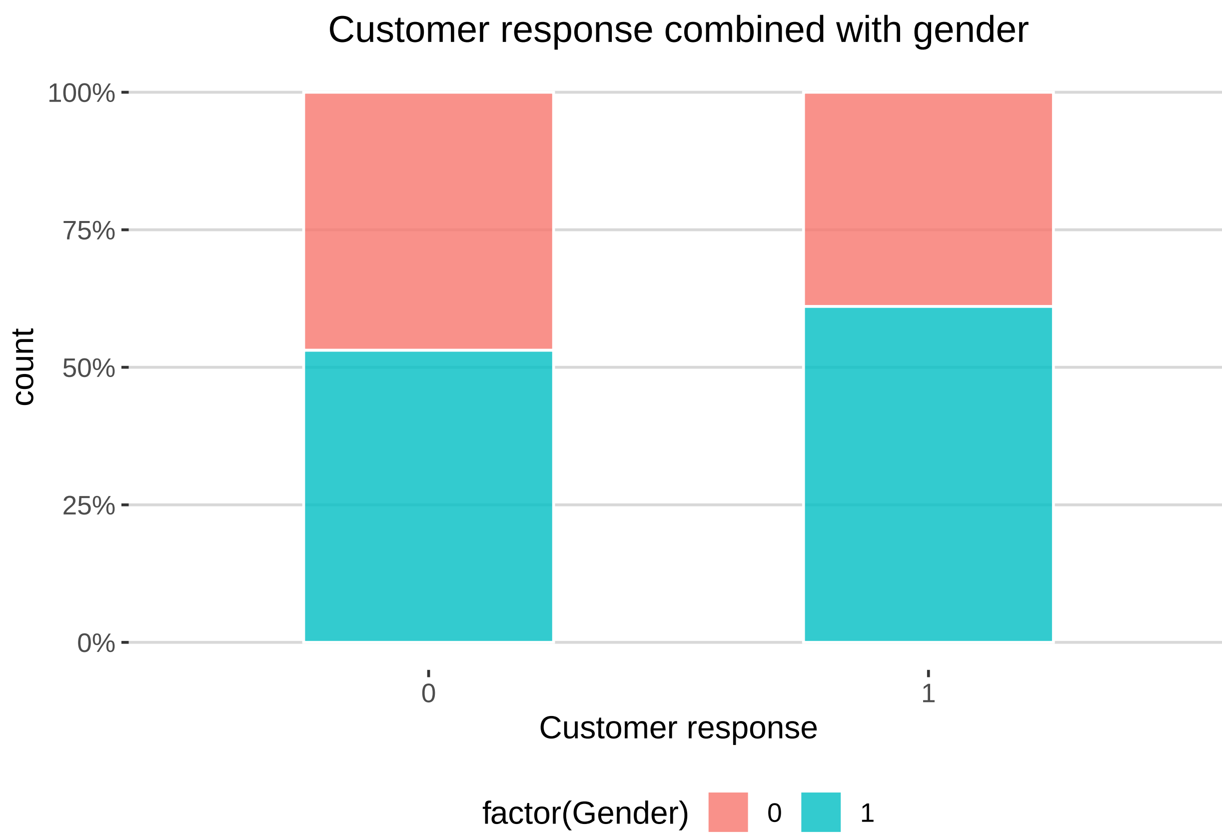

```
# A bar chart showing the number of customers within each policy sales channel.
# Similar to the distribution of region code, most customers have a policy sales
# channel of 152, 26, and 124. Based on this information, the company may plan
# its resources accordingly.

data %>%
    ggplot(aes(x = reorder(factor(Policy_Sales_Channel), Policy_Sales_Channel, function(x) -length(x)),
        fill = factor(Policy_Sales_Channel))) + geom_bar(alpha = 0.8, color = "white",
    show.legend = FALSE) + labs(title = "Bar chart of Policy_Sales_Channel (High to low in thousand)",
    x = "Policy_Sales_Channel") + theme(axis.text.x = element_text(hjust = 1, angle = 90,
    vjust = 0.3, size = 7.5)) + scale_y_continuous(labels = function(y) {
    y/10^3
}) + theme_hc()
```



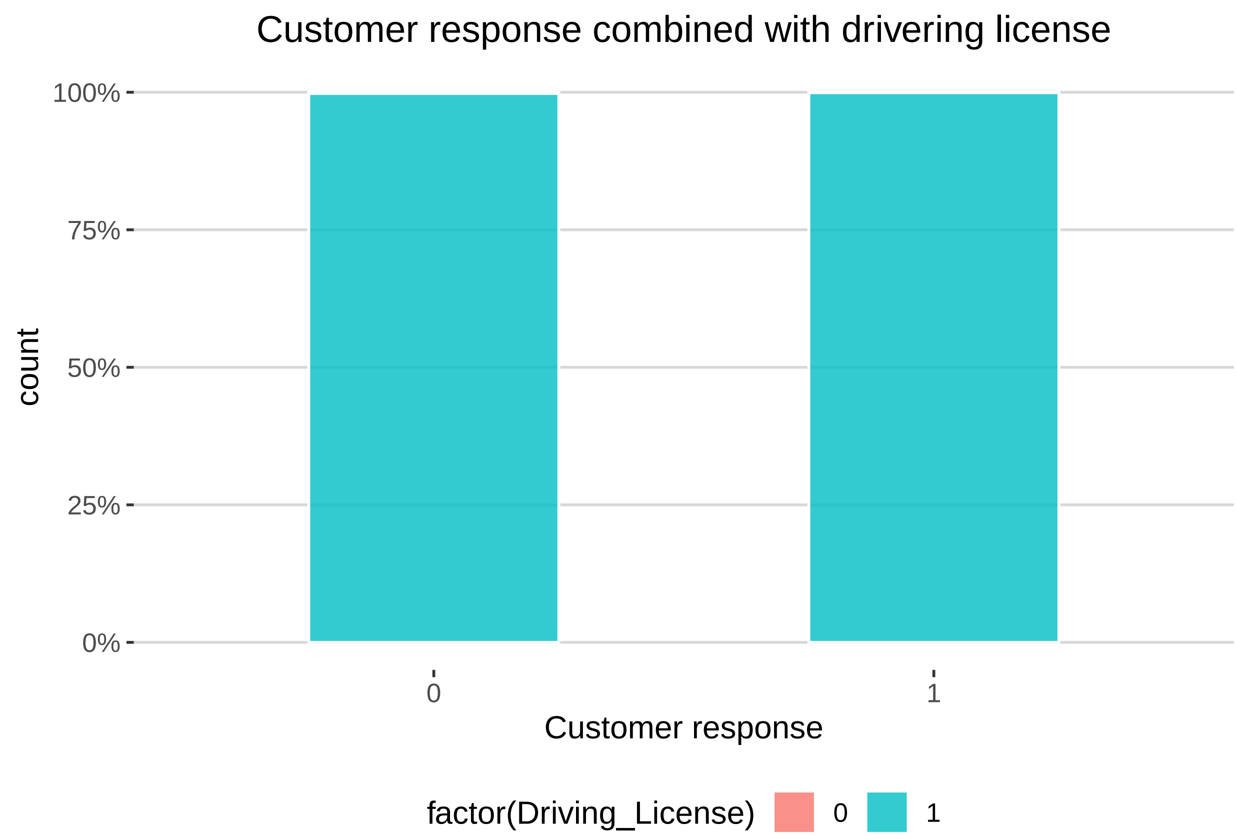

```
# A bar chart showing the count of customers' response. We can see that most
# customers are not interested in the vehicle insurance. The plot distribution
# indicates a huge data imbalance. We will handle this later because, if we train
# the model using the original data, the resulting model will focus much on
# predicting the customer response with a 0 value.

ggplot(data = data, aes(factor(Response), fill = factor(Response))) + geom_bar(stat = "count",
    width = 0.5, alpha = 0.8, color = "white", show.legend = FALSE) + labs(title = "Bar chart of custome
    x = "Response", y = "count (in thousand)") + scale_y_continuous(labels = function(y) {
    y/1000
}) + theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



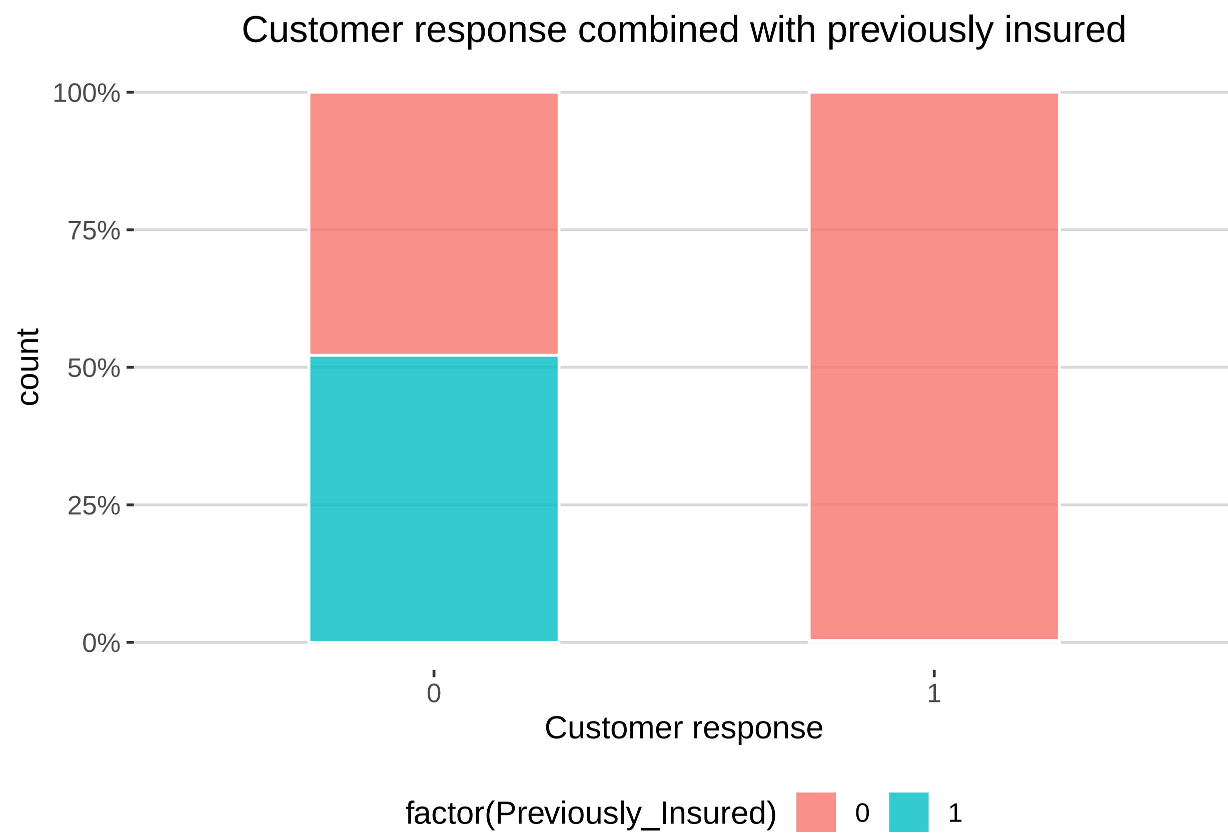

```r
# A stacked bar chart showing, within each customer response group, what
# percentage of customers is male and female. Although we can see that male
# customers are more likely to respond to the new insurance, the percentage
# difference does not seem to be statistically significant

ggplot(data, aes(factor(Response), fill = factor(Gender))) + geom_bar(width = 0.5,
    alpha = 0.8, position = "fill", color = "white") + scale_y_continuous(labels = scales::percent) +
    labs(x = "Customer response", title = "Customer response combined with gender") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



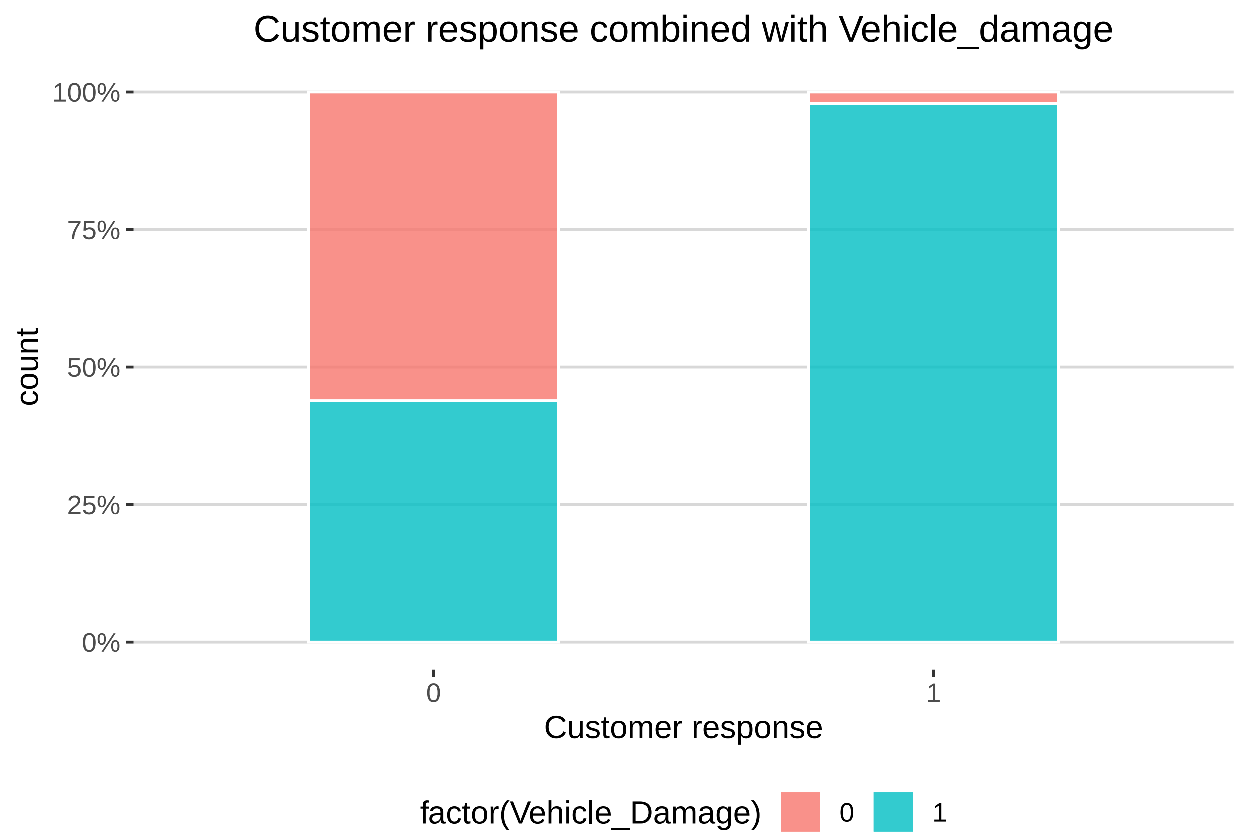

```r
# A stacked bar chart showing, within each customer response group, what
# percentage of customers have or do not have a driver's license. Since most
# customers have a driver's license, the pink part is invisible

ggplot(data, aes(x = factor(Response), fill = factor(Driving_License))) + geom_bar(width = 0.5,
    alpha = 0.8, position = "fill", color = "white") + scale_y_continuous(labels = scales::percent) +
    labs(x = "Customer response", title = "Customer response combined with drivering license") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



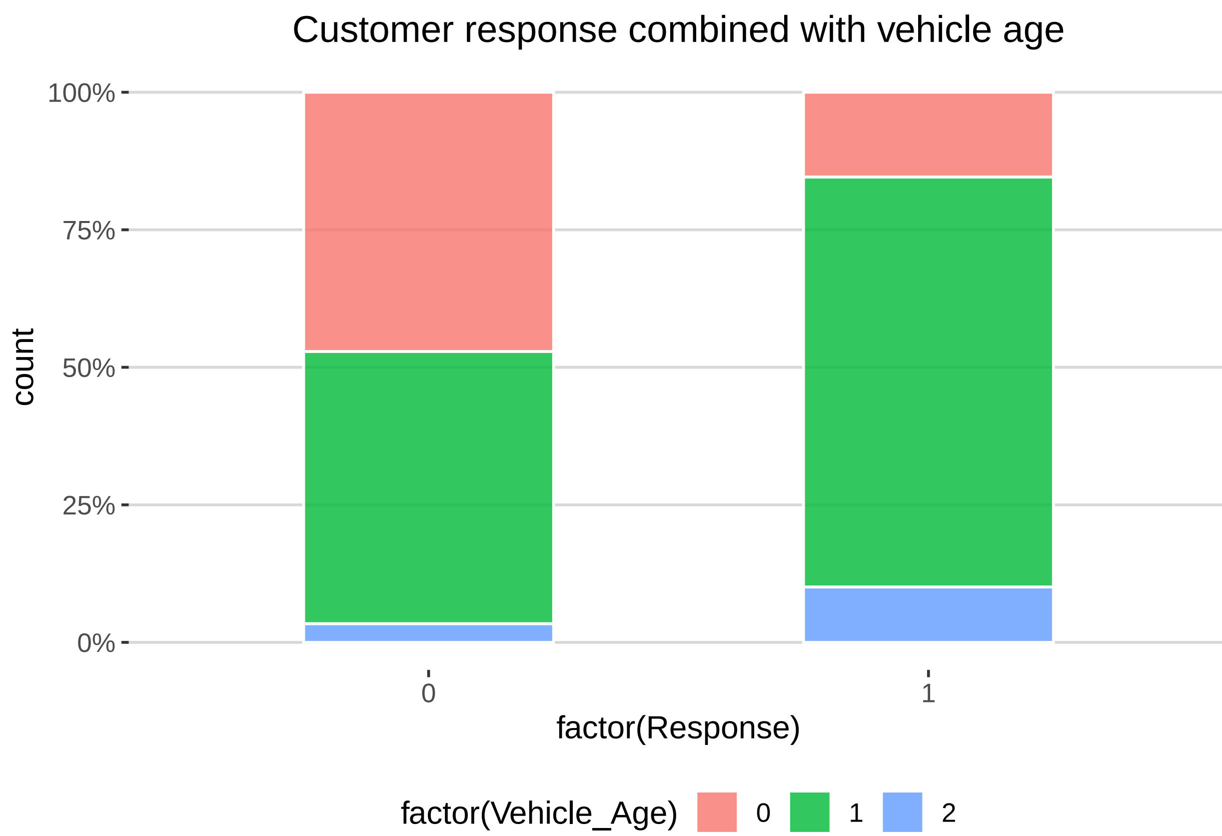

```r
# A stacked bar chart showing, within each customer response group, what
# percentage of customers are previously insured. We can see that more previously
# uninsured customers responded to the new vehicle insurance. However, those
# previously insured show no interest in the vehicle insurance. This already
# indicates that we should focus more on those customers who are not insured
# previously.

ggplot(data = data, aes(factor(Response), fill = factor(Previously_Insured))) + geom_bar(width = 0.5,
    alpha = 0.8, position = "fill", color = "white") + scale_y_continuous(labels = scales::percent) +
    labs(x = "Customer response", title = "Customer response combined with previously insured") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



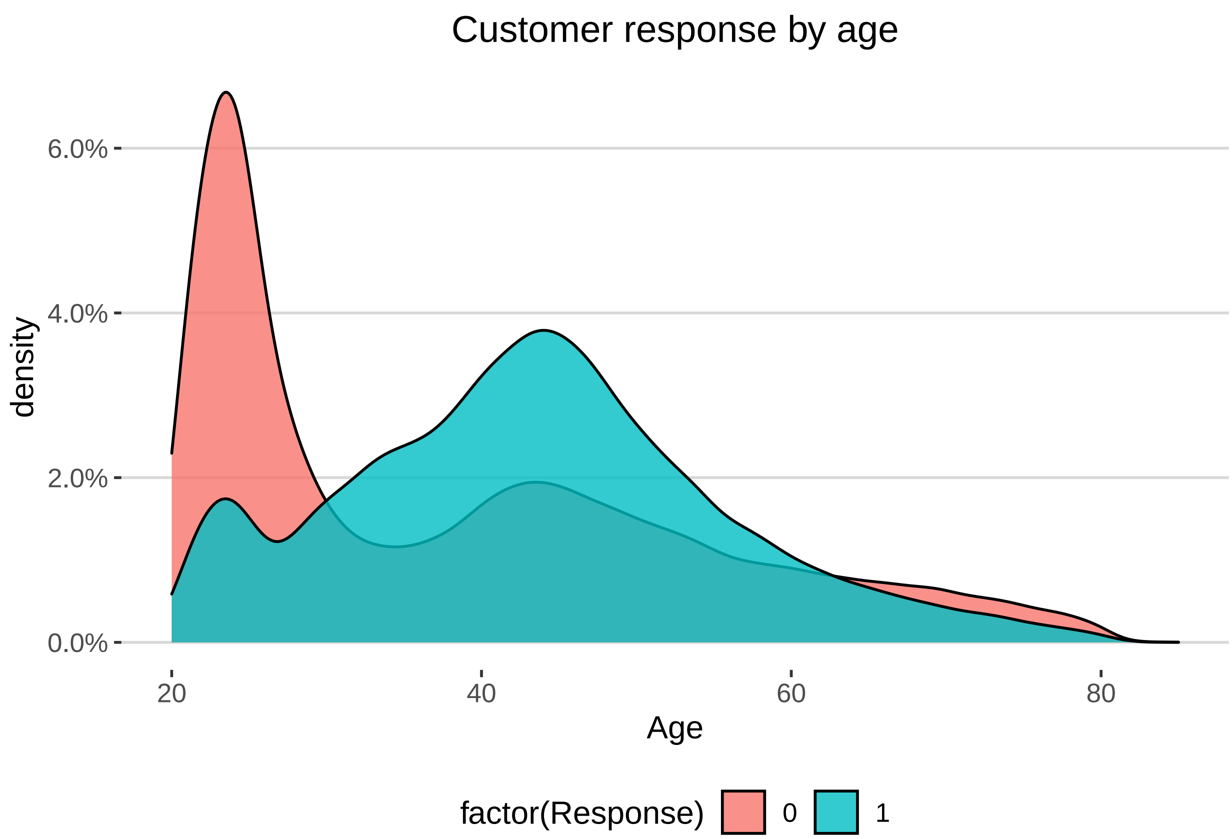

```
# A stacked bar chart showing, within each customer response group, what
# percentage of customers have their vehicle damaged before. We can see those
# most customers who are interested in the vehicle insurance have their vehicles
# been damaged before. This indicates that the company should be careful and
# conservative about pricing, because the new customers of the vehicle insurance
# are probably going to get into accidents again.

ggplot(data, aes(x = factor(Response), fill = factor(Vehicle_Damage))) + geom_bar(width = 0.5,
    alpha = 0.8, position = "fill", color = "white") + scale_y_continuous(labels = scales::percent) +
    labs(x = "Customer response", title = "Customer response combined with Vehicle_damage") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```




```r
# A stacked bar chart showing, within each customer response group, what
# percentage of customers have a vehicle with an age below one year, from one to
# two years, and over two years respectively

ggplot(data, aes(x = factor(Response), fill = factor(Vehicle_Age))) + geom_bar(width = 0.5,
    alpha = 0.8, position = "fill", color = "white") + scale_y_continuous(labels = scales::percent) +
    labs(title = "Customer response combined with vehicle age") + theme(plot.title = element_text(hjust
    theme_hc()
```



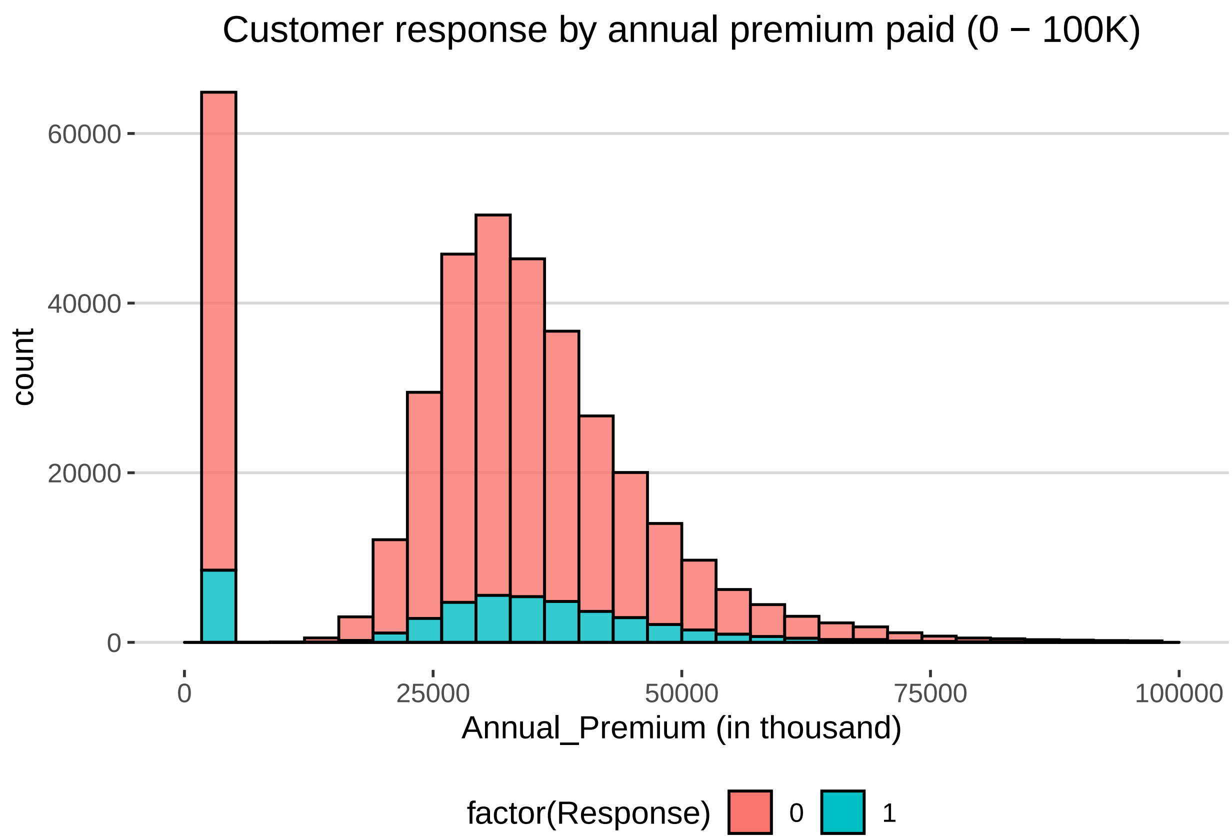

```
# A density plot showing how customers' ages are distributed within each response
# group. The plot indicates that older customers are more likely to be interested
# in the vehicle customers.

ggplot(data, aes(x = Age, fill = factor(Response))) + geom_density(alpha = 0.8) +
    labs(title = "Customer response by age") + scale_y_continuous(labels = scales::percent) +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



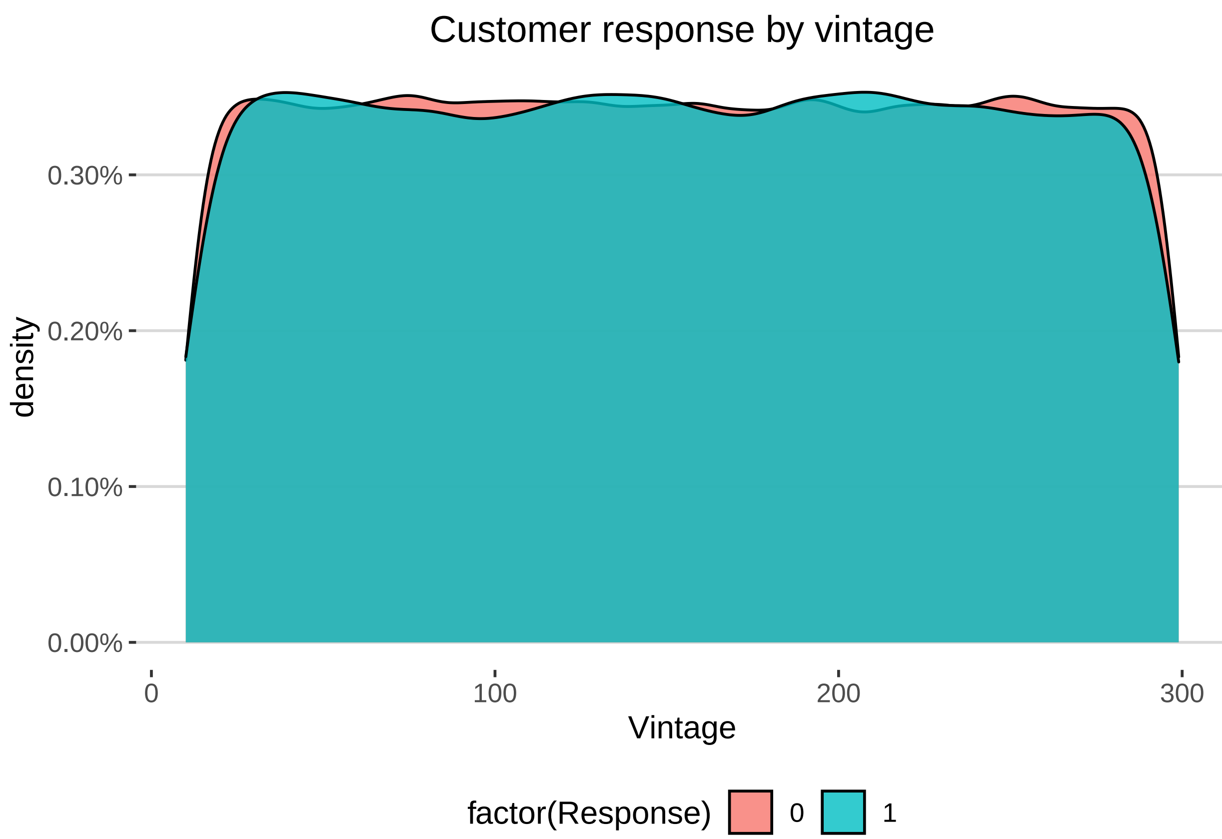

```
# A histogram showing how customers' annual premiums are distributed within each
# response group. The plot shows somewhat similar distributions of annual premium
# paid for the two groups of customers.

ggplot(data, aes(x = Annual_Premium, fill = factor(Response))) + geom_histogram(alpha = 0.8,
    color = "black") + geom_density() + labs(title = "Customer response by annual premium paid (0 - 100k
    x = "Annual_Premium (in thousand)") + scale_x_continuous(labels = function(x) {
    x/1000
}) + theme(plot.title = element_text(hjust = 0.5)) + xlim(0, 1e+05) + theme_hc()
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

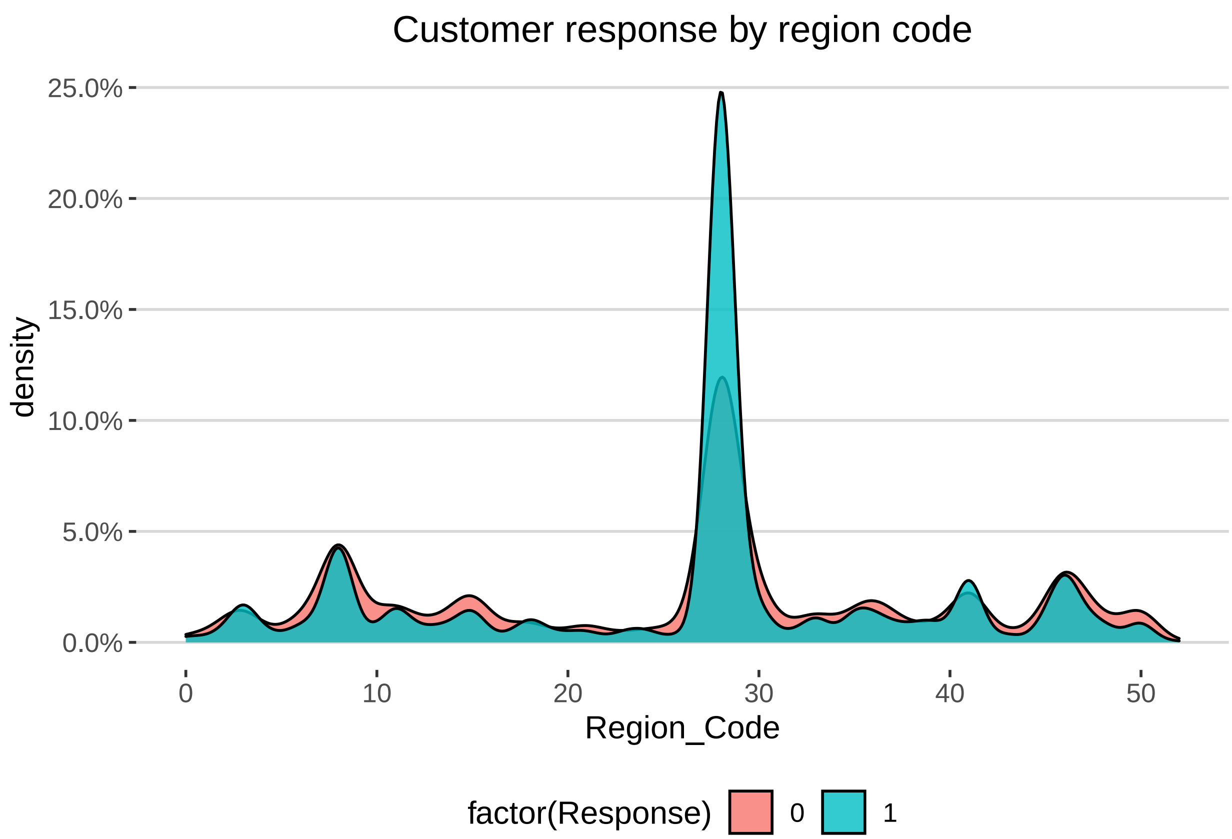
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



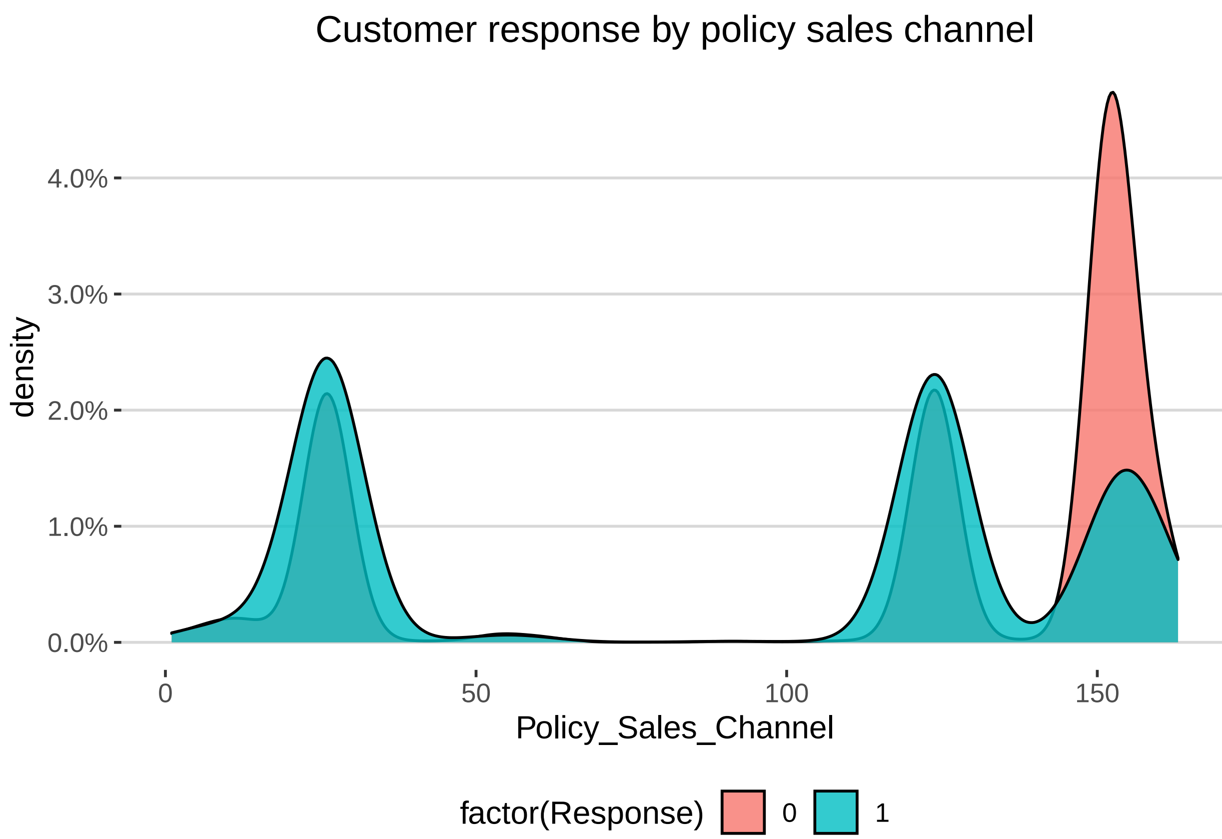

```r
# A density plot showing how customers' vintages are distributed within each
# response group. We can see that customer groups with different response to the
# new insurance are similar in their vintage. In other words, solely looking at a
# customer's vintage does not seem to tell us anything about his/her response.

ggplot(data, aes(Vintage, fill = factor(Response))) + geom_density(alpha = 0.8) +
    scale_y_continuous(labels = scales::percent) + labs(title = "Customer response by vintage") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```



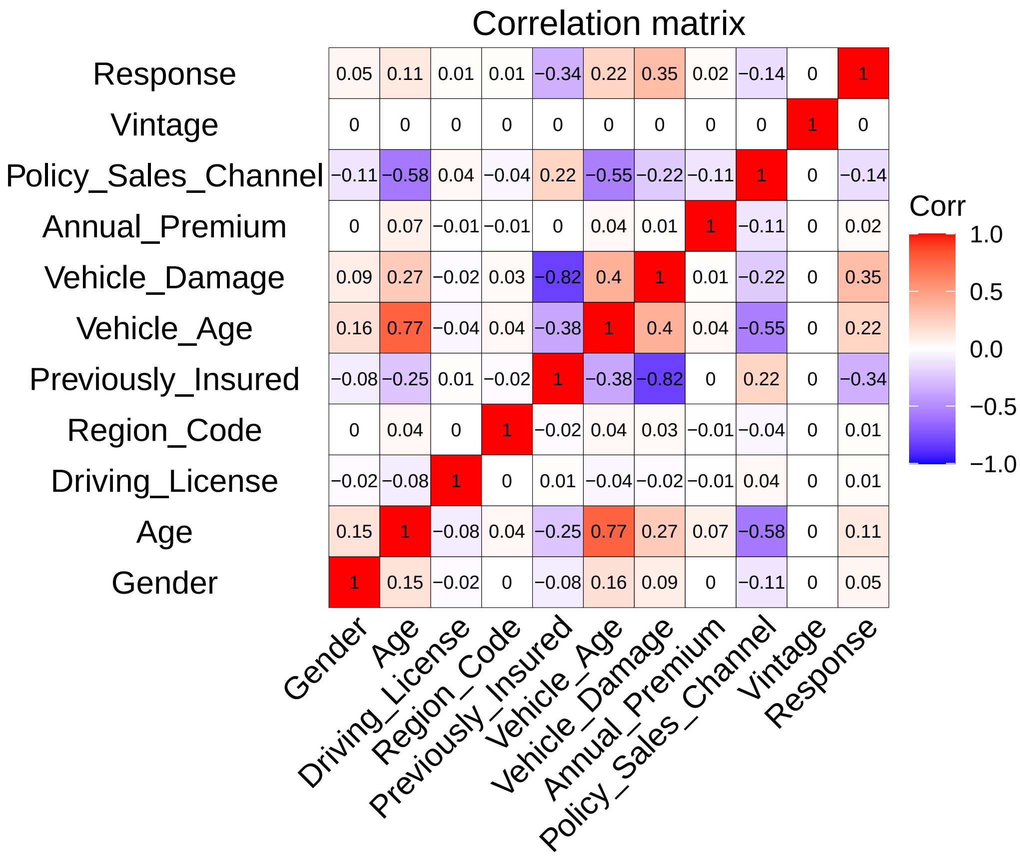

```r
# A density plot showing how customers of different response group are
# distributed within the country. We can see that, in most regions, more
# customers are not interested in the insurance. However, those customers in the
# region with code around 29 shows great interest. With these, the company can
# allocate its resources accordingly to target them.

ggplot(data, aes(x = Region_Code, fill = factor(Response))) + geom_density(alpha = 0.8) +
    scale_y_continuous(labels = scales::percent) + labs(title = "Customer response by region code") +
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```




```r
# A density plot showing how customers within each response group are distributed
# among different policy sales channel. We can see that, although there are many
# unique policy sales channel, most customers falls in a few of them.
# Additionally, most customers that falls under sales channel around 150 are much
# more likely to be interested in the insurance. With this information, the
# company can allocate its resources accordingly.

ggplot(data, aes(x = Policy_Sales_Channel, fill = factor(Response))) + geom_density(alpha = 0.8) +
    scale_y_continuous(labels = scales::percent) + labs(title = "Customer response by policy sales chan
    theme(plot.title = element_text(hjust = 0.5)) + theme_hc()
```




```r
# A correlation matrix showing the correlation between each pair of the
# variables. We can see that some variables are highly correlated. In this case,
# we may have to consider removing certain variables while modeling or using
# interaction.

data_matrix = data
for (i in 1:ncol(data_matrix)) {
    data_matrix[, i] = as.numeric(data_matrix[, i])
}

ggcorrplot(cor(data_matrix), method = "square", type = "full", lab = TRUE, ggtheme = theme_void,
    lab_size = 2.5, outline.color = "black", color = c("blue", "white", "red"), title = "Correlation ma
    theme(plot.title = element_text(hjust = 0.5))
```




```
# Based on the EDA, we can see that there are many values under the 'Region_Code'
# variable. Since most of them only occurred a few times, and most rows have
# certain region code, it may be a good idea to leave those frequently occurred
# region code as they are, and group the rest into a big group. Doing so may
# sacrifice a little model accuracy, but can boost the modeling efficiency very
# much. Some models, such as random forest, cannot even handle that many distinct
# categorical values. With these, I leave the top 4 region codes as they are in
# the original data, while group all other region codes into a big group called
# 'region_group'

region_code_group = data %>%
    count(Region_Code) %>%
    arrange(desc(n)) %>%
    slice_max(n, n = 4) %>%
    select(Region_Code)
data[!(data$Region_Code %in% region_code_group$Region_Code), "Region_Code"] = "region_group"
```

```
# Based on the same logic, I leave the top 4 policy sales channel as they are in
# the original dataset, while group all others into a big group called
# 'channel_group'

policy_group = data %>%
    count(Policy_Sales_Channel) %>%
    arrange(desc(n)) %>%
    slice_max(n, n = 4) %>%
    select(Policy_Sales_Channel)
```

```r
data[!(data$Policy_Sales_Channel %in% policy_group$Policy_Sales_Channel), "Policy_Sales_Channel"] = "cha
```

—————————————————- Logistic Regression Model ——————————————————-

```r
# Before building the model, we convert the data that should not be treated like numeric variable to fa

data[, 'Response'] = as.factor(data[, 'Response'])
data[, 'Gender'] = as.factor(data[, 'Gender'])
data[, 'Driving_License'] = as.factor(data[, 'Driving_License'])
data[, 'Region_Code'] = as.factor(data[, 'Region_Code'])
data[, 'Previously_Insured'] = as.factor(data[, 'Previously_Insured'])
data[, 'Vehicle_Damage'] = as.factor(data[, 'Vehicle_Damage'])
data[, 'Policy_Sales_Channel'] = as.factor(data[, 'Policy_Sales_Channel'])
```

```r
# Randomly assign 80% of data to the training set and train the logistic
# regression model using data. Then, test the model using the 20% data left to
# check the prediction accuracy. Also, during the EDA process, we see that the
# target variable is imbalanced. This means that if we use the original data to
# train the model, the resulting model will target much on predicting those
# customers who are not interested in the insurance (those with a response of 0).
# To solve the data imbalance problem, I used a technique called over sampling.
# What it does is that the technique randomly sample the response that is under
# represented, and increase its occurrence until the number of occurrence of the
# two responses equal to each other. This way, the model will have a better
# performance predicting the outcome of our interest.

set.seed(123)
indices = sample(nrow(data), 0.8 * nrow(data))
train_set = data[indices, ]
test_set = data[-indices, ]

train_set = ovun.sample(Response ~ ., data = train_set, method = "over", N = 535112)$data
```

```r
# Build a logistic regression model based on previous analysis

glm1 = glm(Response ~ Gender + Age * Vehicle_Age * Policy_Sales_Channel + Driving_License +
    Region_Code + Previously_Insured * Vehicle_Damage + Annual_Premium + Vintage,
    data = train_set, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(glm1)
```

```
##
## Call:
## glm(formula = Response ~ Gender + Age * Vehicle_Age * Policy_Sales_Channel +
##     Driving_License + Region_Code + Previously_Insured * Vehicle_Damage +
##     Annual_Premium + Vintage, family = "binomial", data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -3.4394  -0.2601   0.0000   0.7635   3.7802
##
## Coefficients: (1 not defined because of singularities)
##                                                    Estimate Std. Error
## (Intercept)                                      -4.604e+00  2.185e-01
## Gender1                                           7.990e-02  7.426e-03
## Age                                               1.081e-01  7.041e-03
## Vehicle_Age1                                      4.065e+00  2.040e-01
## Vehicle_Age2                                      3.820e+00  2.349e-01
## Policy_Sales_Channel152                          -3.071e-01  2.186e-01
## Policy_Sales_Channel160                          -2.081e+00  2.683e-01
## Policy_Sales_Channel26                           -1.621e+00  2.921e-01
## Policy_Sales_Channelchannel_group                 1.514e+00  2.200e-01
## Driving_License1                                  1.068e+00  8.245e-02
## Region_Code41                                     2.345e-01  1.906e-02
## Region_Code46                                    -9.348e-02  1.828e-02
## Region_Code8                                     -2.555e-01  1.427e-02
## Region_Coderegion_group                          -9.986e-02  8.971e-03
## Previously_Insured1                              -4.326e+00  5.220e-02
## Vehicle_Damage1                                   1.936e+00  1.676e-02
## Annual_Premium                                    1.380e-06  2.223e-07
## Vintage                                          -8.532e-07  4.314e-05
## Age:Vehicle_Age1                                 -1.407e-01  7.077e-03
## Age:Vehicle_Age2                                 -1.344e-01  7.389e-03
## Age:Policy_Sales_Channel152                      -2.742e-02  7.834e-03
## Age:Policy_Sales_Channel160                       1.982e-02  1.053e-02
## Age:Policy_Sales_Channel26                        4.552e-02  1.054e-02
## Age:Policy_Sales_Channelchannel_group            -7.758e-02  7.694e-03
## Vehicle_Age1:Policy_Sales_Channel152              4.758e-01  2.816e-01
## Vehicle_Age2:Policy_Sales_Channel152              1.731e+02  4.785e+02
## Vehicle_Age1:Policy_Sales_Channel160             -8.496e-01  4.648e-01
## Vehicle_Age2:Policy_Sales_Channel160             -1.126e+01  1.970e+02
## Vehicle_Age1:Policy_Sales_Channel26               1.915e+00  2.964e-01
## Vehicle_Age2:Policy_Sales_Channel26               2.809e+00  3.337e-01
## Vehicle_Age1:Policy_Sales_Channelchannel_group   -1.653e+00  2.248e-01
## Vehicle_Age2:Policy_Sales_Channelchannel_group   -2.008e+00  2.749e-01
## Previously_Insured1:Vehicle_Damage1               9.188e-01  6.940e-02
## Age:Vehicle_Age1:Policy_Sales_Channel152         -1.668e-03  8.887e-03
## Age:Vehicle_Age2:Policy_Sales_Channel152         -4.306e+00  1.200e+01
## Age:Vehicle_Age1:Policy_Sales_Channel160          4.810e-03  1.327e-02
## Age:Vehicle_Age2:Policy_Sales_Channel160                 NA         NA
## Age:Vehicle_Age1:Policy_Sales_Channel26          -4.718e-02  1.059e-02
## Age:Vehicle_Age2:Policy_Sales_Channel26          -5.681e-02  1.094e-02
## Age:Vehicle_Age1:Policy_Sales_Channelchannel_group  8.062e-02  7.754e-03
## Age:Vehicle_Age2:Policy_Sales_Channelchannel_group  8.384e-02  8.249e-03
##                                                  z value Pr(>|z|)
## (Intercept)                                      -21.066  < 2e-16 ***
## Gender1                                           10.761  < 2e-16 ***
## Age                                               15.351  < 2e-16 ***
## Vehicle_Age1                                      19.925  < 2e-16 ***
## Vehicle_Age2                                      16.263  < 2e-16 ***
## Policy_Sales_Channel152                           -1.405 0.160037
## Policy_Sales_Channel160                           -7.756 8.76e-15 ***
## Policy_Sales_Channel26                            -5.551 2.85e-08 ***
```

```
## Policy_Sales_Channelchannel_group                        6.880 5.99e-12 ***
## Driving_License1                                        12.958  < 2e-16 ***
## Region_Code41                                           12.303  < 2e-16 ***
## Region_Code46                                           -5.114 3.15e-07 ***
## Region_Code8                                           -17.904  < 2e-16 ***
## Region_Coderegion_group                                -11.132  < 2e-16 ***
## Previously_Insured1                                    -82.869  < 2e-16 ***
## Vehicle_Damage1                                        115.527  < 2e-16 ***
## Annual_Premium                                           6.208 5.36e-10 ***
## Vintage                                                 -0.020 0.984219
## Age:Vehicle_Age1                                       -19.879  < 2e-16 ***
## Age:Vehicle_Age2                                       -18.183  < 2e-16 ***
## Age:Policy_Sales_Channel152                             -3.500 0.000465 ***
## Age:Policy_Sales_Channel160                              1.882 0.059789 .
## Age:Policy_Sales_Channel26                               4.317 1.58e-05 ***
## Age:Policy_Sales_Channelchannel_group                  -10.083  < 2e-16 ***
## Vehicle_Age1:Policy_Sales_Channel152                     1.690 0.091083 .
## Vehicle_Age2:Policy_Sales_Channel152                     0.362 0.717585
## Vehicle_Age1:Policy_Sales_Channel160                    -1.828 0.067588 .
## Vehicle_Age2:Policy_Sales_Channel160                    -0.057 0.954394
## Vehicle_Age1:Policy_Sales_Channel26                      6.462 1.03e-10 ***
## Vehicle_Age2:Policy_Sales_Channel26                      8.416  < 2e-16 ***
## Vehicle_Age1:Policy_Sales_Channelchannel_group          -7.352 1.95e-13 ***
## Vehicle_Age2:Policy_Sales_Channelchannel_group          -7.305 2.77e-13 ***
## Previously_Insured1:Vehicle_Damage1                     13.238  < 2e-16 ***
## Age:Vehicle_Age1:Policy_Sales_Channel152                -0.188 0.851090
## Age:Vehicle_Age2:Policy_Sales_Channel152                -0.359 0.719834
## Age:Vehicle_Age1:Policy_Sales_Channel160                 0.363 0.716967
## Age:Vehicle_Age2:Policy_Sales_Channel160                    NA       NA
## Age:Vehicle_Age1:Policy_Sales_Channel26                 -4.454 8.42e-06 ***
## Age:Vehicle_Age2:Policy_Sales_Channel26                 -5.193 2.07e-07 ***
## Age:Vehicle_Age1:Policy_Sales_Channelchannel_group  10.397  < 2e-16 ***
## Age:Vehicle_Age2:Policy_Sales_Channelchannel_group  10.163  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 741823  on 535111  degrees of freedom
## Residual deviance: 458123  on 535072  degrees of freedom
## AIC: 458203
##
## Number of Fisher Scoring iterations: 10
```

```r
# Setting a proper threshold, we get an overall out of sample prediction accuracy
# of 87.7%.

mean(ifelse(predict(glm1, test_set, type = "response") > 0.91, 1, 0) == test_set$Response)
```

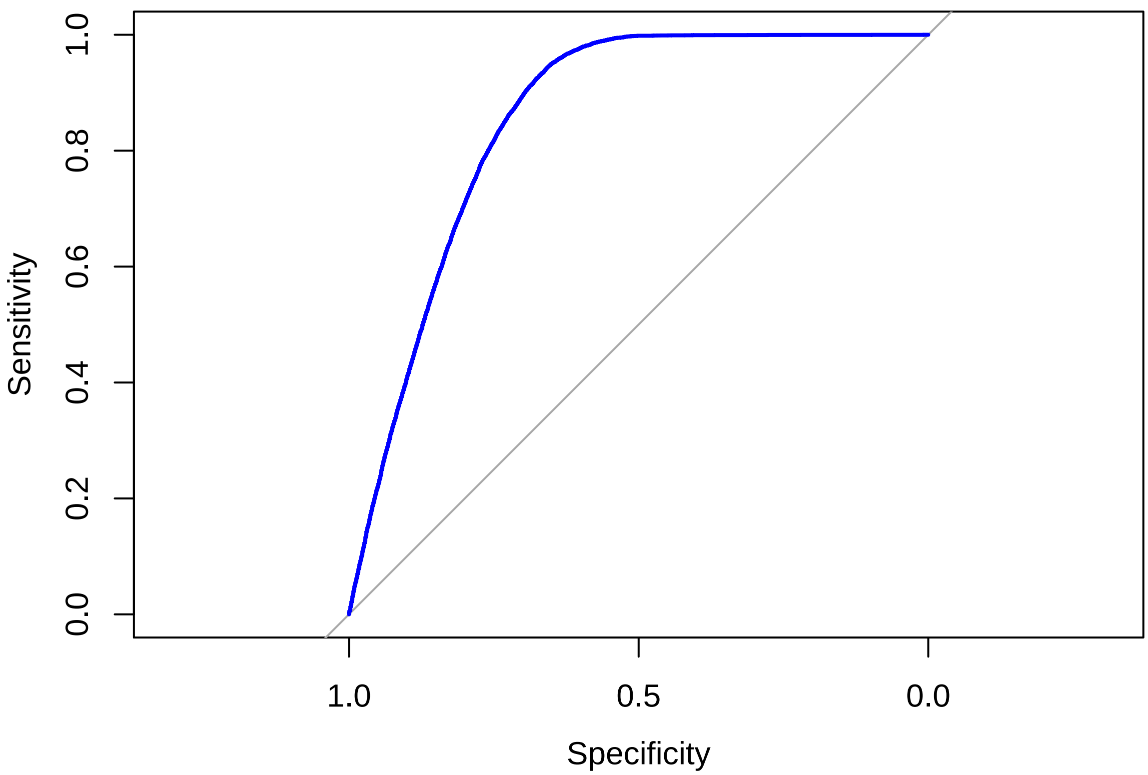
```
## [1] 0.8765238
```

```r
# Here we use a visualization tool called ROC curve. It provides a way to better
# understand the model's ability to distinguish between 0 and 1 target variable
```

```r
roc1 = roc(test_set$Response, predict(glm1, test_set, type = "response"))
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Setting direction: controls < cases
```

```r
plot(roc1, col = "blue")
```




```r
# Calculate the area under the ROC curve. The closer the area is to one, the
# better the model performs. Here, we get 85.29% as our AUC.

auc(roc1)
```

```
## Area under the curve: 0.8529
```

Random Forest

```r
# Different models have different fundamental logic. Here, we use a random forest
# model to see how it performs in predicting the target response

rf = randomForest(Response ~ ., data = train_set, ntree = 250)
print(rf)
```

```
##
## Call:
##  randomForest(formula = Response ~ ., data = train_set, ntree = 250)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 17.91%
## Confusion matrix:
##        0      1 class.error
## 0 182477  85079  0.31798577
## 1  10754 256802  0.04019345
```

```
# A confusion matrix shows how the random forest model performs. We can see that,
# although the overall accuracy is 71.25%, random forest does a very good job
# predicting the target variable that we are interested in (predicting customers
# who are interested in the insurance). How to choose between two models depends
# on our goal of building the model. Given our interest, we may want to use
# random forest for predicting purpose.

confusionMatrix(predict(rf, test_set), test_set$Response)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 45724   850
##          1 21057  8578
##
##                Accuracy : 0.7125
##                  95% CI : (0.7093, 0.7158)
##     No Information Rate : 0.8763
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3096
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6847
##             Specificity : 0.9098
##          Pos Pred Value : 0.9817
##          Neg Pred Value : 0.2895
##              Prevalence : 0.8763
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.6111
##       Balanced Accuracy : 0.7973
##
##        'Positive' Class : 0
##
```