Cellphone Billing Project

Harry Chang

2023-04-22

```
setwd("/Users/harrychang/Downloads/BT4211/Assignment/Assignment 2")
```

```
library(readxl)
cellphone = read_excel("cellphone_billing_student.xlsx")
subscriber = read_excel("subscriber_info_student.xlsx")
zip = read_excel("zip_info_student.xlsx")
```

Exploring the cellphone billing dataset

```
cellphone
```

```
## # A tibble: 195,878 x 11
##     cust_id bill_year bill_month churn plan_chosen plan1 plan2 plan3 plan4
##       <dbl>     <dbl>      <dbl> <dbl>       <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 19164958      2015          9     0           1     1     0     0     0
##  2 19164958      2015         10     0           1     1     0     0     0
##  3 19164958      2015         11     0           1     1     0     0     0
##  4 19164958      2015         12     0           1     1     0     0     0
##  5 19164958      2016          1     0           1     1     0     0     0
##  6 19164958      2016          2     0           1     1     0     0     0
##  7 19164958      2016          3     0           1     1     0     0     0
##  8 19164958      2016          4     0           1     1     0     0     0
##  9 19164958      2016          5     0           1     1     0     0     0
## 10 19164958      2016          6     0           1     1     0     0     0
## # ... with 195,868 more rows, and 2 more variables: total_minute_peak <dbl>,
## #   promo_lag1 <dbl>
```

```
summary(cellphone)
```

```
##     cust_id           bill_year      bill_month        churn
##  Min.   :19164958   Min.   :2015   Min.   : 1.00   Min.   :0.00000
##  1st Qu.:74944945   1st Qu.:2016   1st Qu.: 3.00   1st Qu.:0.00000
##  Median :81475998   Median :2016   Median : 5.00   Median :0.00000
##  Mean   :81935115   Mean   :2016   Mean   : 5.99   Mean   :0.02508
##  3rd Qu.:88623109   3rd Qu.:2016   3rd Qu.: 9.00   3rd Qu.:0.00000
##  Max.   :88705192   Max.   :2017   Max.   :12.00   Max.   :1.00000
##   plan_chosen        plan1            plan2            plan3
##  Min.   :1.000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
```

```
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :1.000   Median :1.0000   Median :0.00000   Median :0.0000
##  Mean   :2.021   Mean   :0.5096   Mean   :0.03194   Mean   :0.3859
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :4.000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##      plan4         total_minute_peak   promo_lag1
##  Min.   :0.00000   Min.   :   0.0    Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:  58.0    1st Qu.:0.0000
##  Median :0.00000   Median : 139.0    Median :0.0000
##  Mean   :0.07253   Mean   : 171.4    Mean   :0.1347
##  3rd Qu.:0.00000   3rd Qu.: 244.0    3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1500.0    Max.   :1.0000
```

Descriptive statistics

```
mean_total_minute_peak <- mean(cellphone$total_minute_peak)
median_total_minute_peak <- median(cellphone$total_minute_peak)
sd_total_minute_peak <- sd(cellphone$total_minute_peak)

cat("Mean of total_minute_peak:", mean_total_minute_peak, "\n")
```

```
## Mean of total_minute_peak: 171.4271
```

```
cat("Median of total_minute_peak:", median_total_minute_peak, "\n")
```

```
## Median of total_minute_peak: 139
```

```
cat("Standard deviation of total_minute_peak:", sd_total_minute_peak, "\n")
```

```
## Standard deviation of total_minute_peak: 150.8134
```

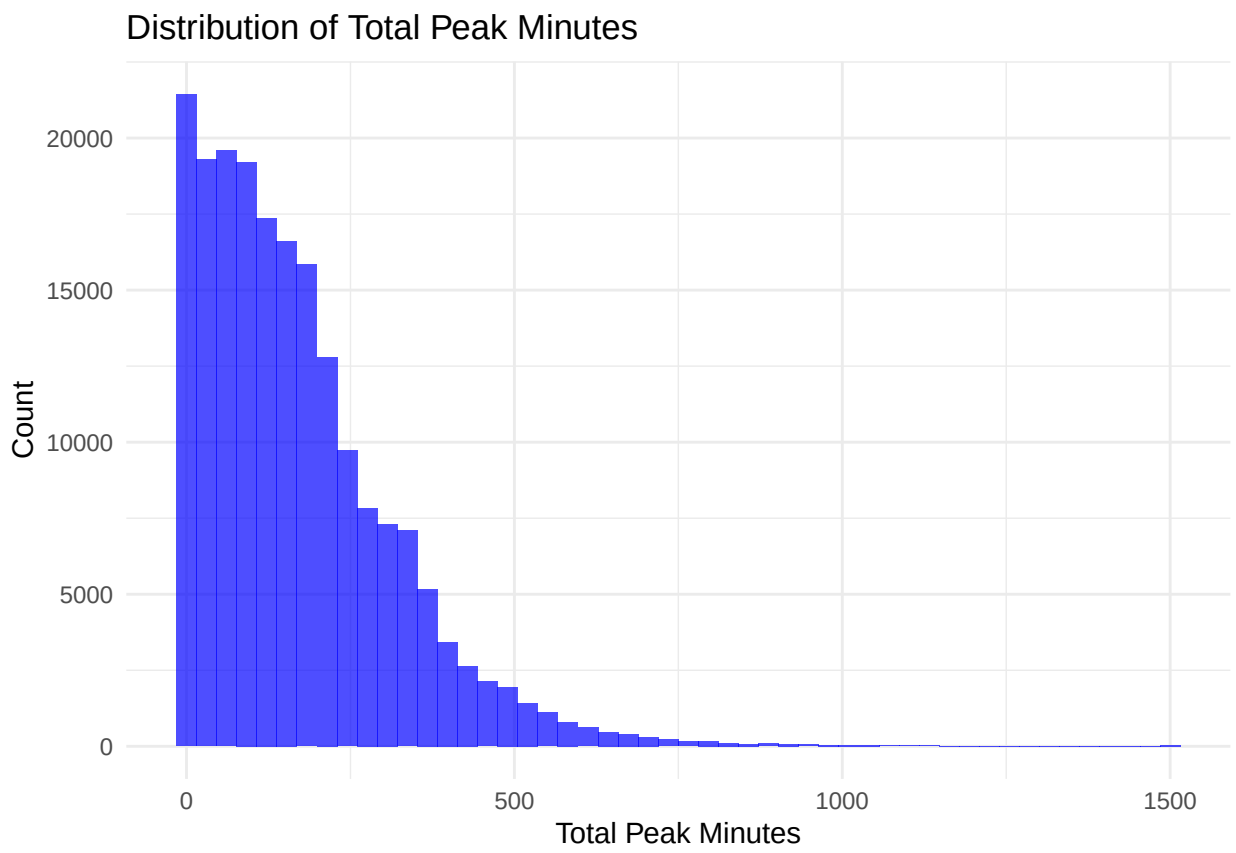
```
customer_count <- nrow(cellphone)
plan_proportions <- table(cellphone$plan_chosen) / customer_count
cat("Proportions of customers in each plan:\n")
```

```
## Proportions of customers in each plan:
```

```
print(plan_proportions)
```

```
##
##          1          2          3          4
## 0.50957739 0.03194335 0.38594942 0.07252984
```

```
library(ggplot2)
ggplot(cellphone, aes(x = total_minute_peak)) +
  geom_histogram(bins = 50, fill = "blue", alpha = 0.7) +
  theme_minimal() +
  labs(title = "Distribution of Total Peak Minutes", x = "Total Peak Minutes", y = "Count")
```



Churn analysis

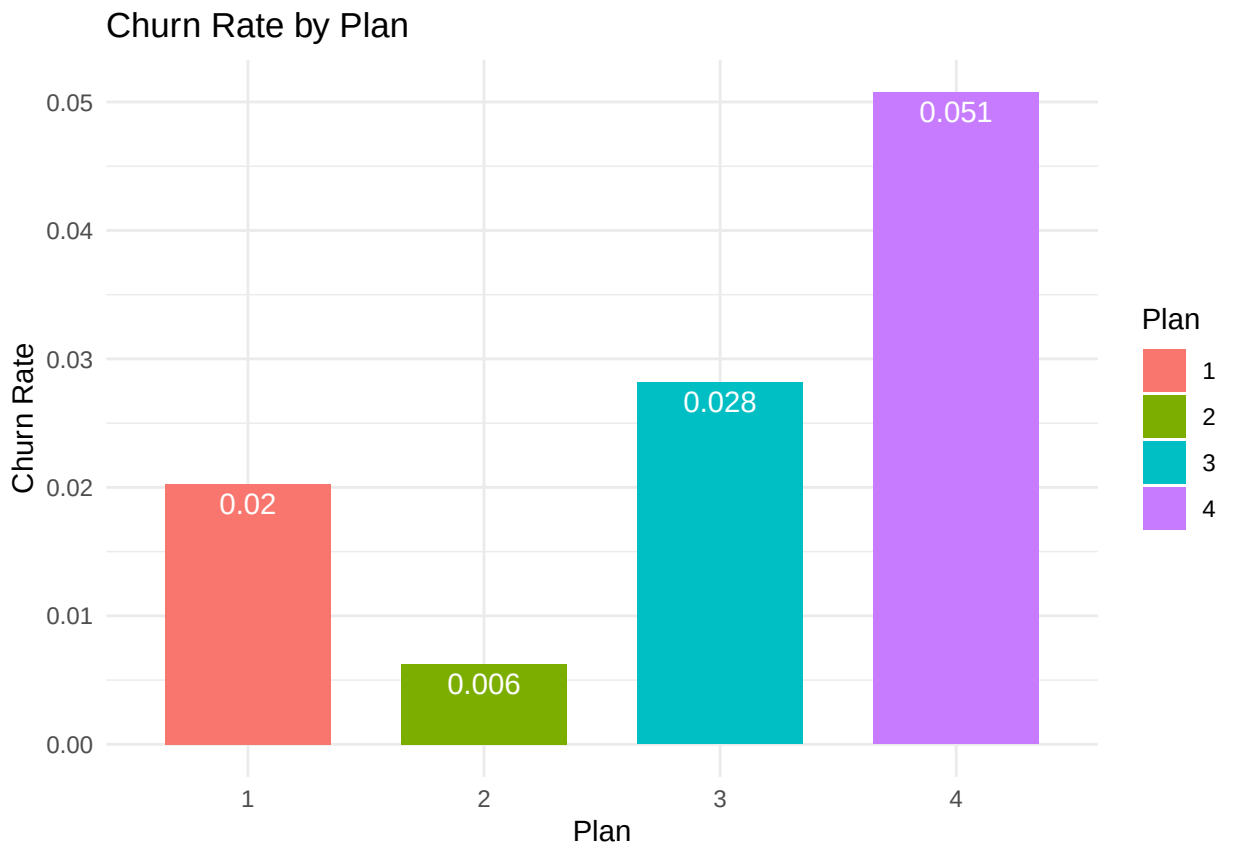
```
churn_rate_by_plan <- with(cellphone, tapply(churn, plan_chosen, function(x) mean(x == 1)))
cat("Churn rate by plan:\n")
```

```
## Churn rate by plan:
```

```
print(churn_rate_by_plan)
```

```
##           1           2           3           4
## 0.020257476 0.006233019 0.028174976 0.050749630
```

```
churn_rate_by_plan_df <- as.data.frame(churn_rate_by_plan)
churn_rate_by_plan_df$plan_chosen <- row.names(churn_rate_by_plan_df)

ggplot(churn_rate_by_plan_df, aes(x = plan_chosen, y = churn_rate_by_plan, fill = plan_chosen)) +
  geom_bar(stat = "identity", width = 0.7) +
  theme_minimal() +
  labs(title = "Churn Rate by Plan", x = "Plan", y = "Churn Rate") +
  geom_text(aes(label = round(churn_rate_by_plan,3)), vjust = 1.4, color = "white") +
  scale_fill_discrete(name = "Plan")
```



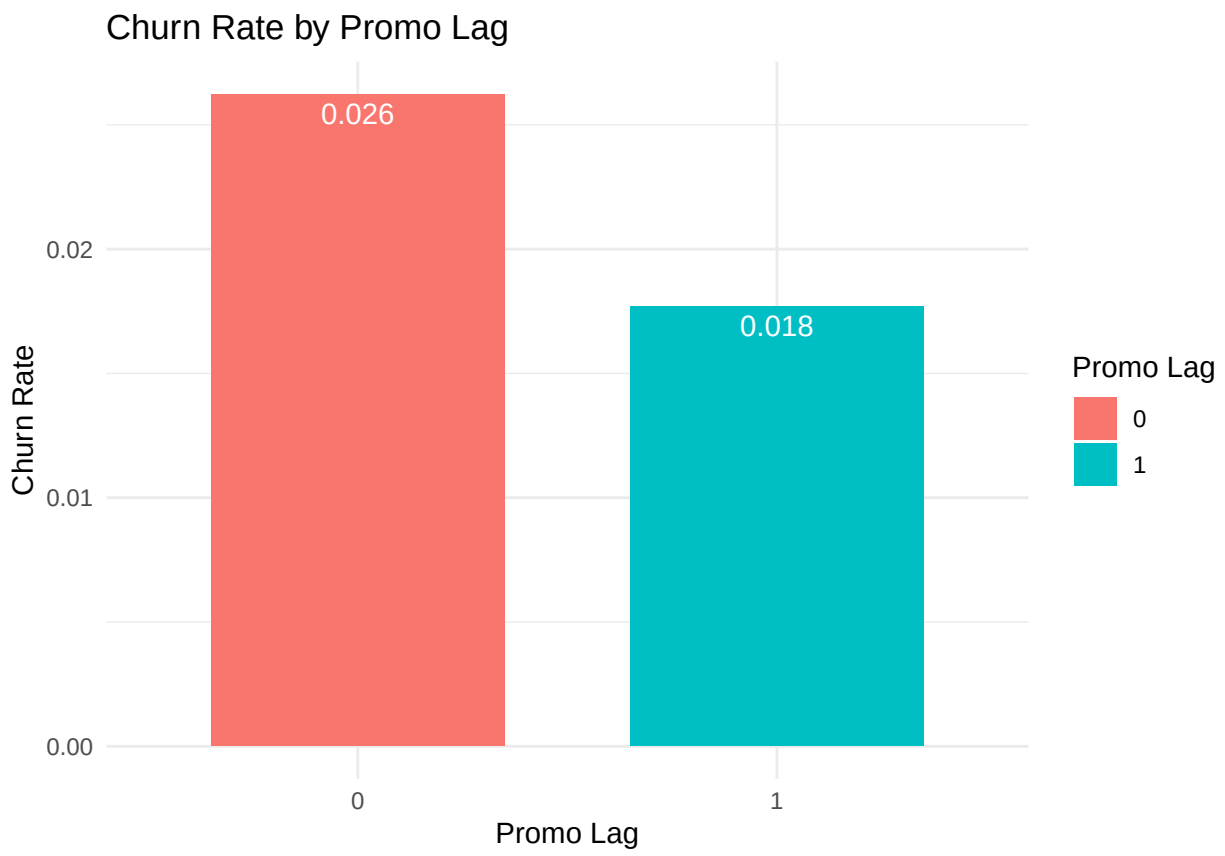

```
churn_rate_by_promo <- with(cellphone, tapply(churn, promo_lag1, function(x) mean(x == 1)))
cat("Churn rate by promo_lag1:\n")
```

```
## Churn rate by promo_lag1:
```

```
print(churn_rate_by_promo)
```

```
##          0          1
## 0.02622635 0.01769476
```

```
churn_rate_by_promo_df <- as.data.frame(churn_rate_by_promo)
churn_rate_by_promo_df$promo_lag1 <- row.names(churn_rate_by_promo_df)

ggplot(churn_rate_by_promo_df, aes(x = promo_lag1, y = churn_rate_by_promo, fill = promo_lag1)) +
  geom_bar(stat = "identity", width = 0.7) +
  theme_minimal() +
  labs(title = "Churn Rate by Promo Lag", x = "Promo Lag", y = "Churn Rate") +
  geom_text(aes(label = round(churn_rate_by_promo,3)), vjust = 1.4, color = "white") +
  scale_fill_discrete(name = "Promo Lag")
```



Customer segmentation

```
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
```

```
data_scaled <- cellphone %>%
  select(total_minute_peak, plan_chosen, promo_lag1) %>%
  scale()
```

```
set.seed(123) # For reproducibility
k <- 3 # Number of clusters
kmeans_result <- kmeans(data_scaled, centers = k)
```

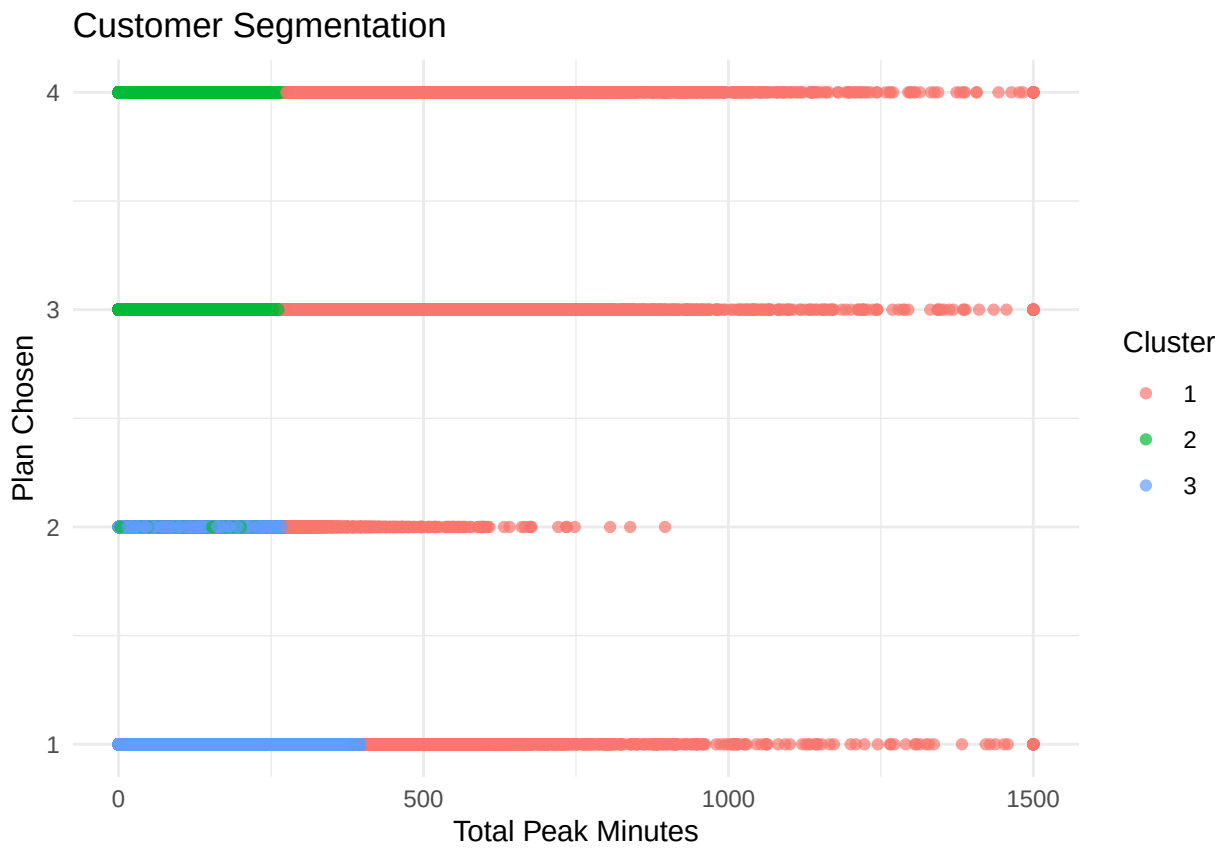
```r
cellphone$cluster <- kmeans_result$cluster
cluster_summary <- cellphone %>%
  group_by(cluster) %>%
  summarise(
    count = n(),
    avg_total_minute_peak = mean(total_minute_peak),
    avg_churn = mean(churn),
    avg_promo_lag1 = mean(promo_lag1),
    plan1_prop = mean(plan1),
    plan2_prop = mean(plan2),
    plan3_prop = mean(plan3),
    plan4_prop = mean(plan4)
  )

cat("Customer segmentation summary:\n")
```

```
## Customer segmentation summary:
```

```r
print(cluster_summary)
```

```
## # A tibble: 3 x 9
##   cluster  count avg_total_minut~ avg_churn avg_promo_lag1 plan1_prop plan2_prop
##     <int>  <int>            <dbl>     <dbl>          <dbl>      <dbl>      <dbl>
## 1       1  37530             408.    0.0245          0.144     0.0467     0.0359
## 2       2  55969             124.    0.0361          0.161     0          0.0106
## 3       3 102379             111.    0.0193          0.117     0.958      0.0422
## # ... with 2 more variables: plan3_prop <dbl>, plan4_prop <dbl>
```

```r
ggplot(cellphone, aes(x = total_minute_peak, y = plan_chosen, color = factor(cluster))) +
  geom_point(alpha = 0.7) +
  theme_minimal() +
  labs(title = "Customer Segmentation", x = "Total Peak Minutes", y = "Plan Chosen", color = "Cluster")
```



Detailed descriptive analysis

a) How many customers are there for each type of phone service plan?

```
customers_per_plan <- table(cellphone$plan_chosen)
cat("Number of customers for each plan:\n")
```

```
## Number of customers for each plan:
```

```
print(customers_per_plan)
```

```
##
##     1     2     3     4
## 99815  6257 75599 14207
```

b) What is the average number of customers acquired for each phone service plan across the months of August to December 2015?

```
# Filter customers acquired between August and December 2015
data_2015 <- cellphone %>% filter(bill_year == 2015 & bill_month >= 8 & bill_month <= 12)

# Remove duplicate customer IDs and count unique customers per plan
unique_customers_2015 <- data_2015 %>% distinct(cust_id, .keep_all = TRUE)
```

```
customers_per_plan_2015 <- table(unique_customers_2015$plan_chosen)

# Calculate the average number of customers acquired per plan
avg_customers_per_plan_2015 <- customers_per_plan_2015 / 5
cat("Average number of customers acquired for each plan (August to December 2015):\n")
```

```
## Average number of customers acquired for each plan (August to December 2015):
```

```
print(avg_customers_per_plan_2015)
```

```
##
##     1     3     4
## 459.2 853.0  81.2
```

c) What is the average number of months a customer stays with the cellular phone service company from the start till the end of a phone service subscription (i.e., only focus on customers who churned)?

```
churned_customers <- cellphone %>% filter(churn == 1)
churned_customers_months <- churned_customers %>% group_by(cust_id) %>% summarise(months_active = n())
avg_months_churned <- mean(churned_customers_months$months_active)
cat("Average number of months a churned customer stays with the company:", avg_months_churned, "\n")
```

```
## Average number of months a churned customer stays with the company: 1
```

d) For each type of phone service plan, what is the number of customers who churned?

```
churned_by_plan <- table(churned_customers$plan_chosen)
cat("Number of churned customers by plan:\n")
```

```
## Number of churned customers by plan:
```

```
print(churned_by_plan)
```

```
##
##    1    2    3    4
## 2022   39 2130  721
```

e) For each type of phone service plan, what is the average number of months a customer stays with the company from the start till the end of a service subscription (i.e., only focus on customers who churned)?

```
avg_months_churned_by_plan <- churned_customers %>% group_by(plan_chosen) %>% summarise(avg_months_activ
#cat("Average number of months churned customers stay with the company by plan:\n")
print(avg_months_churned_by_plan)
```

```
## # A tibble: 4 x 2
##   plan_chosen avg_months_active
##         <dbl>             <dbl>
## 1           1              2022
## 2           2                39
## 3           3              2130
## 4           4               721
```

f) What is the average number of peak minutes used in a month under each type of phone service plan?

```
avg_peak_minutes_by_plan <- cellphone %>% group_by(plan_chosen) %>% summarise(avg_peak_minutes = mean(to
#cat("Average number of peak minutes used in a month by plan:\n")
print(avg_peak_minutes_by_plan)
```

```
## # A tibble: 4 x 2
##   plan_chosen avg_peak_minutes
##         <dbl>            <dbl>
## 1           1             117.
## 2           2             185.
## 3           3             214.
## 4           4             320.
```

g) How many customers are over-utilizing their allocated maximum peak time minutes (by more than 5%) for each type of phone service plan?

```
# Calculate maximum peak minutes allowed for each plan
max_minutes <- c(200, 300, 350, 500)
cellphone$max_peak_minutes <- max_minutes[cellphone$plan_chosen]

# Identify customers who exceed their allocated maximum peak time minutes by more than 5%
cellphone$over_utilizing <- cellphone$total_minute_peak > cellphone$max_peak_minutes * 1.05

# Count the number of customers who over-utilize their allocated maximum peak time minutes for each pla
over_utilizing_customers_by_plan <- cellphone %>%
  filter(over_utilizing == TRUE) %>%
  group_by(plan_chosen) %>%
  summarise(count = n())

#cat("Number of customers over-utilizing their allocated maximum peak time minutes by plan:\n")
print(over_utilizing_customers_by_plan)
```

```
## # A tibble: 4 x 2
##   plan_chosen count
##         <dbl> <int>
## 1           1 13275
## 2           2   611
## 3           3 10365
## 4           4  2343
```

h) What is the average total phone bill (fixed subscription charge plus variable excess usage fees) per customer across all months under each type of phone service plan?

```
fixed_prices <- c(30, 35, 40, 50)
excess_prices <- 0.40

cellphone$fixed_charge <- fixed_prices[cellphone$plan_chosen]
cellphone$excess_usage <- pmax(0, cellphone$total_minute_peak - cellphone$max_peak_minutes) * excess_pri
cellphone$total_bill <- cellphone$fixed_charge + cellphone$excess_usage

avg_total_bill_by_plan <- cellphone %>%
```

```
  group_by(plan_chosen) %>%
  summarise(avg_total_bill = mean(total_bill))

#cat("Average total phone bill per customer across all months by plan:\n")
print(avg_total_bill_by_plan)
```

```
## # A tibble: 4 x 2
##   plan_chosen avg_total_bill
##         <dbl>          <dbl>
## 1           1           35.6
## 2           2           38.6
## 3           3           47.2
## 4           4           61.6
```

i) What is the average profit per customer across all months under each type of phone service plan?

```
profit_margin <- 0.53
cellphone$profit <- cellphone$total_bill * profit_margin
avg_profit_by_plan <- cellphone %>%
  group_by(plan_chosen) %>%
  summarise(avg_profit = mean(profit))

#cat("Average profit per customer across all months by plan:\n")
print(avg_profit_by_plan)
```

```
## # A tibble: 4 x 2
##   plan_chosen avg_profit
##         <dbl>      <dbl>
## 1           1       18.9
## 2           2       20.4
## 3           3       25.0
## 4           4       32.7
```

j) For customers who have churned, what is the average customer lifetime value under each type of phone service plan? Assume a monthly discount rate of 1% and compute the lifetime value as of the month of customer acquisition onward.

```
monthly_discount_rate <- 0.01

churned_customers <- cellphone %>% filter(churn == 1)
churned_customers <- churned_customers %>%
  group_by(cust_id, plan_chosen) %>%
  summarise(
  months_active = n(),
  total_profit = sum(profit)
)
```

```
## `summarise()` has grouped output by 'cust_id'. You can override using the
## `.groups` argument.
```

```r
churned_customers$lifetime_value <- churned_customers$total_profit / ((1 - (1 + monthly_discount_rate)^

avg_lifetime_value_by_plan <- churned_customers %>%
  group_by(plan_chosen) %>%
  summarise(avg_lifetime_value = mean(lifetime_value))

#cat("Average customer lifetime value for churned customers by plan:\n")
print(avg_lifetime_value_by_plan)
```

```
## # A tibble: 4 x 2
##   plan_chosen avg_lifetime_value
##         <dbl>              <dbl>
## 1           1               20.9
## 2           2               21.4
## 3           3               27.3
## 4           4               33.3
```

Creating aggregated dataset using the 3 originally provided datasets

```r
# Aggregate the panel-level customer billing data into a cross-sectional one, keeping the last observat
billing_data_cross_sectional <- cellphone %>%
  group_by(cust_id) %>%
  slice_tail(n = 1)

# Compute average monthly peak minutes used for each customer (ave_minute_peak):
average_minutes <- cellphone %>%
  group_by(cust_id) %>%
  summarise(ave_minute_peak = mean(total_minute_peak))
billing_data_cross_sectional <- left_join(billing_data_cross_sectional, average_minutes, by = "cust_id"

# Compute the percentage of months a customer received marketing promotions (promo_pct):
promo_percentage <- cellphone %>%
  group_by(cust_id) %>%
  summarise(promo_pct = mean(promo_lag1))
billing_data_cross_sectional <- left_join(billing_data_cross_sectional, promo_percentage, by = "cust_id

# Merge the cross-sectional billing data with the subscriber information data:
merged_data <- left_join(billing_data_cross_sectional, subscriber, by = "cust_id")

# Create the customer age variable (age):
current_year <- 2017
merged_data$age <- current_year - merged_data$birth_year

# Further merge the data set created in Step 5 above with the ZIP code data:
merged_data <- left_join(merged_data, zip, by = "zip_code")

# Create the commercial ZIP code type dummy variable (zip_comm):
merged_data$zip_comm <- ifelse(merged_data$zip_type == 2, 1, 0)

merged_data
```

```
## # A tibble: 12,495 x 33
## # Groups:   cust_id [12,495]
##     cust_id bill_year bill_month churn plan_chosen plan1 plan2 plan3 plan4
##       <dbl>     <dbl>      <dbl> <dbl>       <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 19164958      2017          4     1           1     1     0     0     0
##  2 39244924      2016          2     1           3     0     0     1     0
##  3 39578413      2016          4     1           3     0     0     1     0
##  4 40992265      2016          5     1           3     0     0     1     0
##  5 43061957      2016         10     1           3     0     0     1     0
##  6 47196850      2017          5     0           2     0     1     0     0
##  7 51236987      2017          2     1           3     0     0     1     0
##  8 51326773      2016          4     1           3     0     0     1     0
##  9 54271247      2016         11     1           1     1     0     0     0
## 10 70765025      2016         11     1           3     0     0     1     0
## # ... with 12,485 more rows, and 24 more variables: total_minute_peak <dbl>,
## #   promo_lag1 <dbl>, cluster <int>, max_peak_minutes <dbl>,
## #   over_utilizing <lgl>, fixed_charge <dbl>, excess_usage <dbl>,
## #   total_bill <dbl>, profit <dbl>, ave_minute_peak <dbl>, promo_pct <dbl>,
## #   birth_year <dbl>, birth_month <dbl>, zip_code <dbl>, newcell <dbl>,
## #   oldcell <dbl>, prizm_code <dbl>, age <dbl>, ruca2 <dbl>, zip_type <dbl>,
## #   zip_popn5 <dbl>, state_code <dbl>, state_abbr <chr>, zip_comm <dbl>
```

```
# Check dimensions of merged dataset
dim(merged_data)
```

```
## [1] 12495    33
```

Using logit and probit models on merged dataset

(a) Estimate the binomial logit and probit models of customer churn decision:

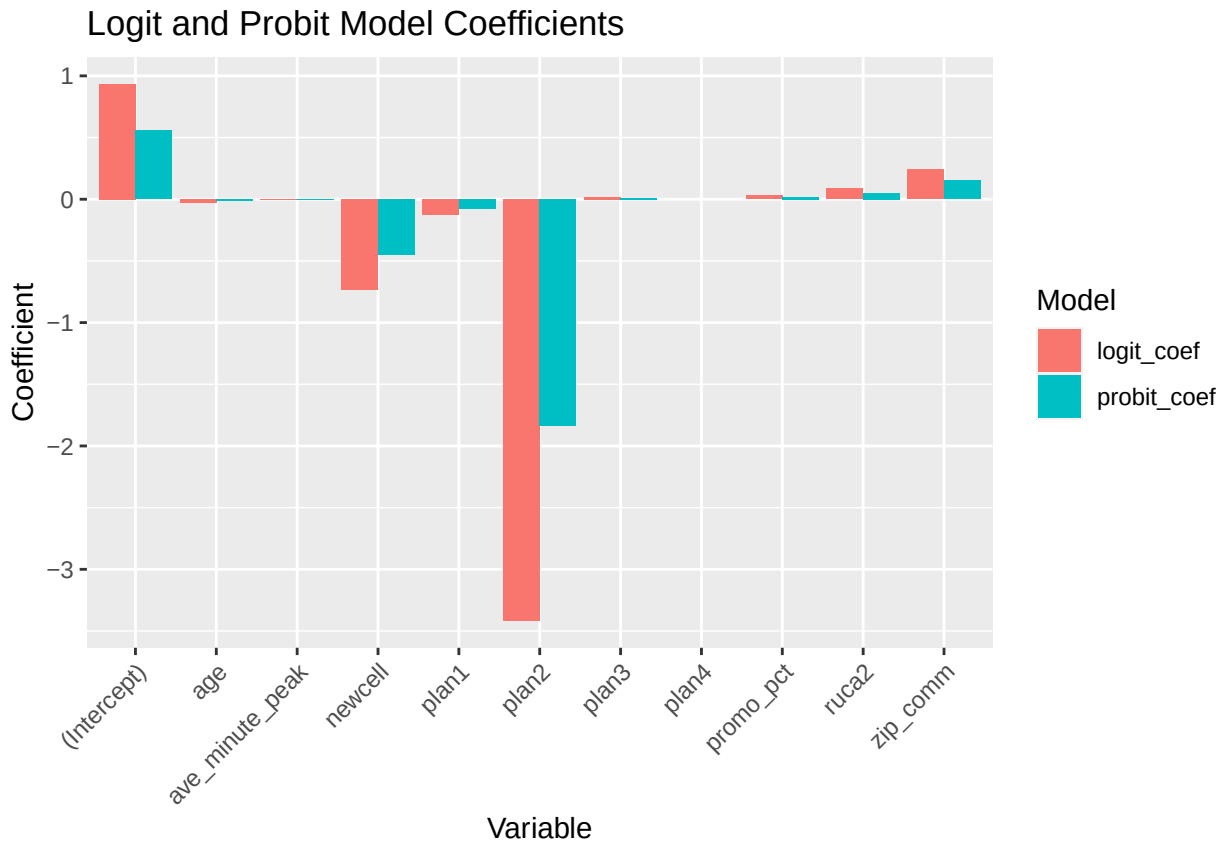
```
library(car)
```

```
## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode
```

```
# Binomial Logit Model
logit_model <- glm(churn ~ plan1 + plan2 + plan3 + plan4 + ave_minute_peak +
                   promo_pct + age + newcell + ruca2 + zip_comm,
                   family = binomial(link = "logit"), data = merged_data)

# Binomial Probit Model
probit_model <- glm(churn ~ plan1 + plan2 + plan3 + plan4 + ave_minute_peak +
                    promo_pct + age + newcell + ruca2 + zip_comm,
                    family = binomial(link = "probit"), data = merged_data)
```

```
# Create a dataframe with coefficients and model names
coefs <- data.frame(
  variable = names(logit_model$coefficients),
  logit_coef = logit_model$coefficients,
  probit_coef = probit_model$coefficients
)

coefs <- reshape2::melt(coefs, id.vars = "variable", variable.name = "model", value.name = "coefficient

# Plot coefficients
ggplot(coefs, aes(x = variable, y = coefficient, fill = model)) +
  geom_col(position = "dodge") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Logit and Probit Model Coefficients", x = "Variable", y = "Coefficient", fill = "Model"
```

```
## Warning: Removed 2 rows containing missing values (`geom_col()`).
```



(b) Compare the model fit using AIC and accuracy:

```
library(caret)
```

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

```r
# Split data into training and testing sets
set.seed(123)
train_index <- createDataPartition(merged_data$churn, p = 0.8, list = FALSE)
train_data <- merged_data[train_index, ]
test_data <- merged_data[-train_index, ]

# Fit the models on the training data
logit_model <- glm(churn ~ plan1 + plan2 + plan3 + ave_minute_peak + promo_pct + age + newcell + ruca2 +

probit_model <- glm(churn ~ plan1 + plan2 + plan3 + ave_minute_peak + promo_pct + age + newcell + ruca2

# Predictions on the test data
logit_pred <- ifelse(predict(logit_model, newdata = test_data, type = "response") > 0.5, 1, 0)
probit_pred <- ifelse(predict(probit_model, newdata = test_data, type = "response") > 0.5, 1, 0)

# Confusion matrices
logit_conf_mat <- table(Predicted = as.factor(logit_pred), Actual = as.factor(test_data$churn))
probit_conf_mat <- table(Predicted = as.factor(probit_pred), Actual = as.factor(test_data$churn))

# Accuracy
logit_accuracy <- sum(diag(logit_conf_mat)) / sum(logit_conf_mat)
probit_accuracy <- sum(diag(probit_conf_mat)) / sum(probit_conf_mat)
```

```r
# Create a dataframe with AIC and accuracy
model_comparison <- data.frame(
  model = c("Logit", "Probit"),
  AIC = c(AIC(logit_model), AIC(probit_model)),
  accuracy = c(logit_accuracy, probit_accuracy)
)

model_comparison
```

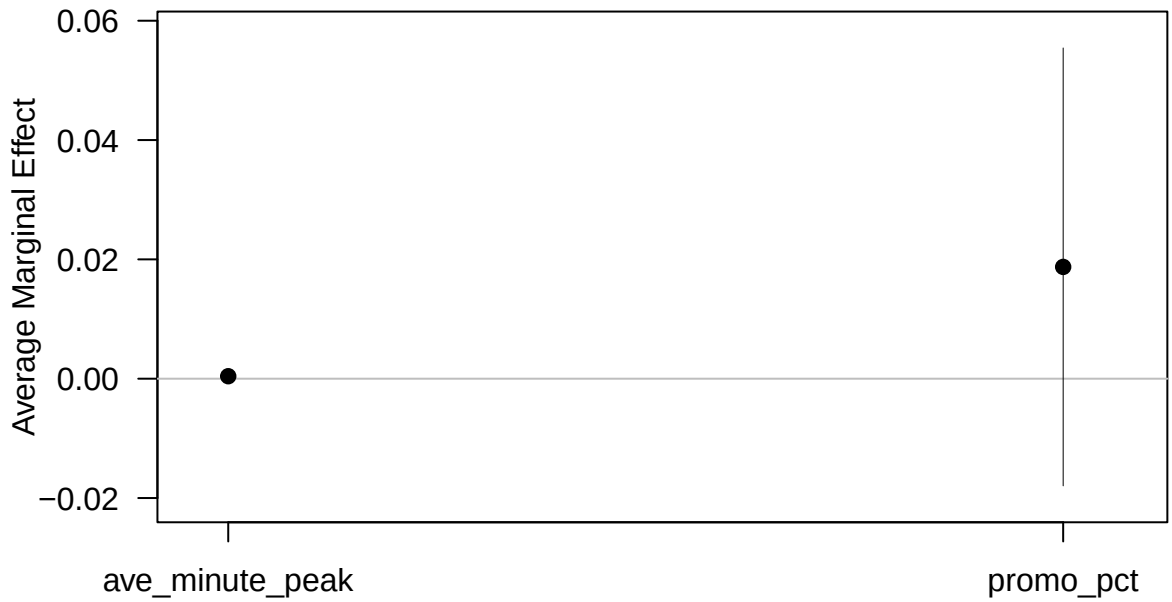
```
##    model      AIC  accuracy
## 1  Logit 11391.22 0.6562373
## 2 Probit 11388.33 0.6562373
```

(c) Marginal effects of ave_minute_peak and promo_pct on the probability of customer churn:

```r
library(margins)

# Calculate marginal effects for the best-fit model (e.g., logit_model)
marginal_effects <- margins(logit_model, variables = c("ave_minute_peak", "promo_pct"))

# Summary of marginal effects
marginal_effects_summary <- summary(marginal_effects)
marginal_effects_summary
```

```
##           factor    AME     SE      z      p   lower  upper
##  ave_minute_peak 0.0004 0.0000 8.7750 0.0000  0.0003 0.0005
##        promo_pct 0.0187 0.0187 1.0008 0.3169 -0.0180 0.0554
```

```r
# Plot marginal effects
plot(marginal_effects)
```



(d) Multinomial logit model of customer plan choice decision:

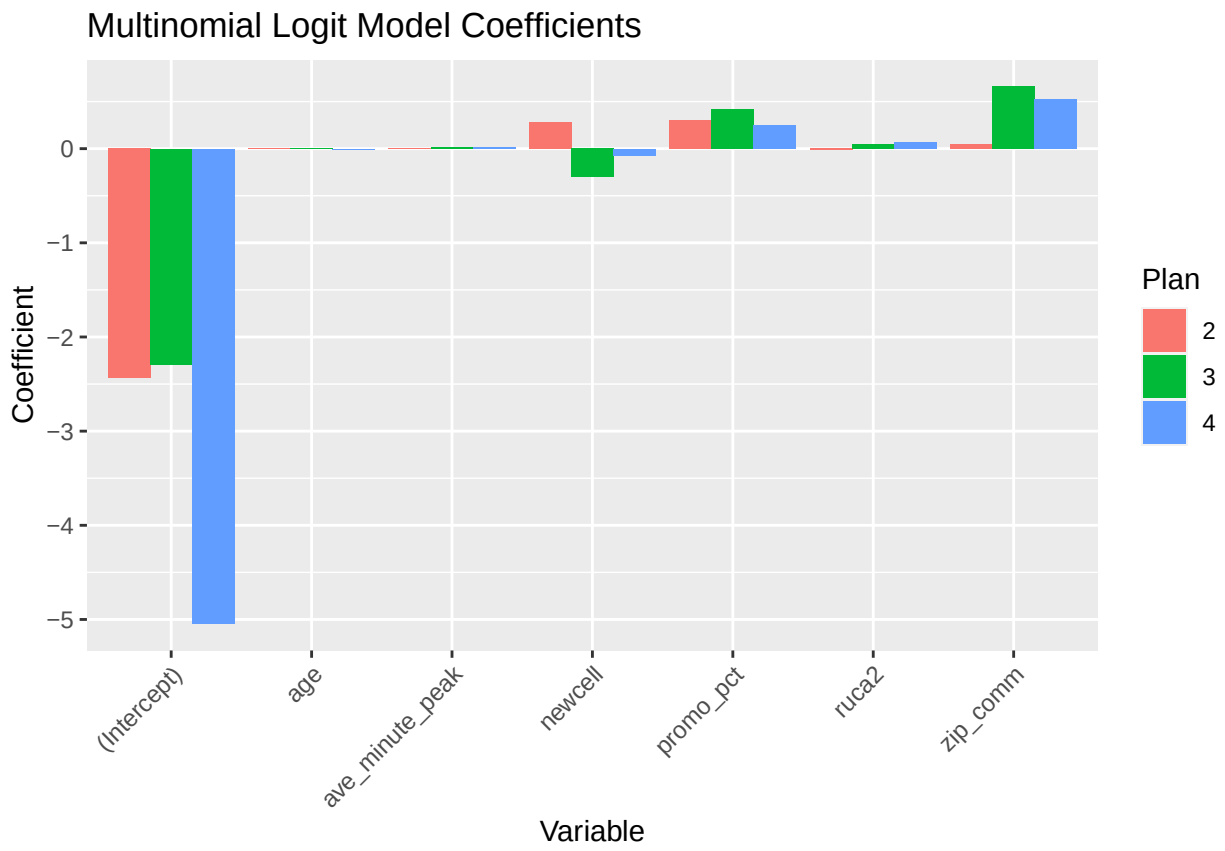
```r
library(nnet)

# Create a new variable for plan choice
merged_data$plan_choice <- as.factor(ifelse(merged_data$plan1 == 1, 1, ifelse(merged_data$plan2 == 1, 2

# Remove missing data (if any)
merged_data_clean <- merged_data %>%
  na.omit()

multinom_model <- multinom(plan_choice ~ ave_minute_peak + promo_pct + age + newcell + ruca2 + zip_comm
```

```
## # weights:  32 (21 variable)
## initial  value 15791.279068
## iter  10 value 12580.890145
## iter  20 value 11644.332942
## iter  30 value 11518.606463
## final  value 11518.584009
## converged
```

```r
# Summary of the multinomial logit model
multinom_summary <- summary(multinom_model)
multinom_summary
```

```
## Call:
## multinom(formula = plan_choice ~ ave_minute_peak + promo_pct +
##     age + newcell + ruca2 + zip_comm, data = merged_data_clean)
##
## Coefficients:
##   (Intercept) ave_minute_peak promo_pct           age     newcell       ruca2
## 2   -2.434356     0.007756583 0.2996368  0.0005083711  0.27851183 -0.01439087
## 3   -2.292736     0.014357492 0.4232658 -0.0055970647 -0.30487260  0.04846221
## 4   -5.044267     0.021095249 0.2469391 -0.0103360633 -0.07173525  0.06775428
##     zip_comm
## 2 0.04336654
## 3 0.65914560
## 4 0.52568588
##
## Std. Errors:
##   (Intercept) ave_minute_peak promo_pct         age    newcell      ruca2
## 2   0.1644395    0.0004134632 0.1286716 0.002206126 0.07786722 0.02431938
## 3   0.1307836    0.0003395906 0.1018426 0.001775667 0.06958558 0.01772544
## 4   0.2086960    0.0004423015 0.1609257 0.002970345 0.10919213 0.02932189
##    zip_comm
## 2 0.3612414
## 3 0.2510011
## 4 0.3739297
##
## Residual Deviance: 23037.17
## AIC: 23079.17
```

```r
# Prepare the coefficients data for visualization
coefs <- coef(multinom_model)
coefs <- as.data.frame(t(coefs))
coefs$Variable <- row.names(coefs)
coefs_long <- reshape2::melt(coefs, id.vars = "Variable", variable.name = "Plan", value.name = "Coeffic

# Visualize the coefficients
ggplot(coefs_long, aes(x = Variable, y = Coefficient, fill = Plan)) +
  geom_col(position = "dodge") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Multinomial Logit Model Coefficients", x = "Variable", y = "Coefficient", fill = "Plan")
```



Model estimations for duration models

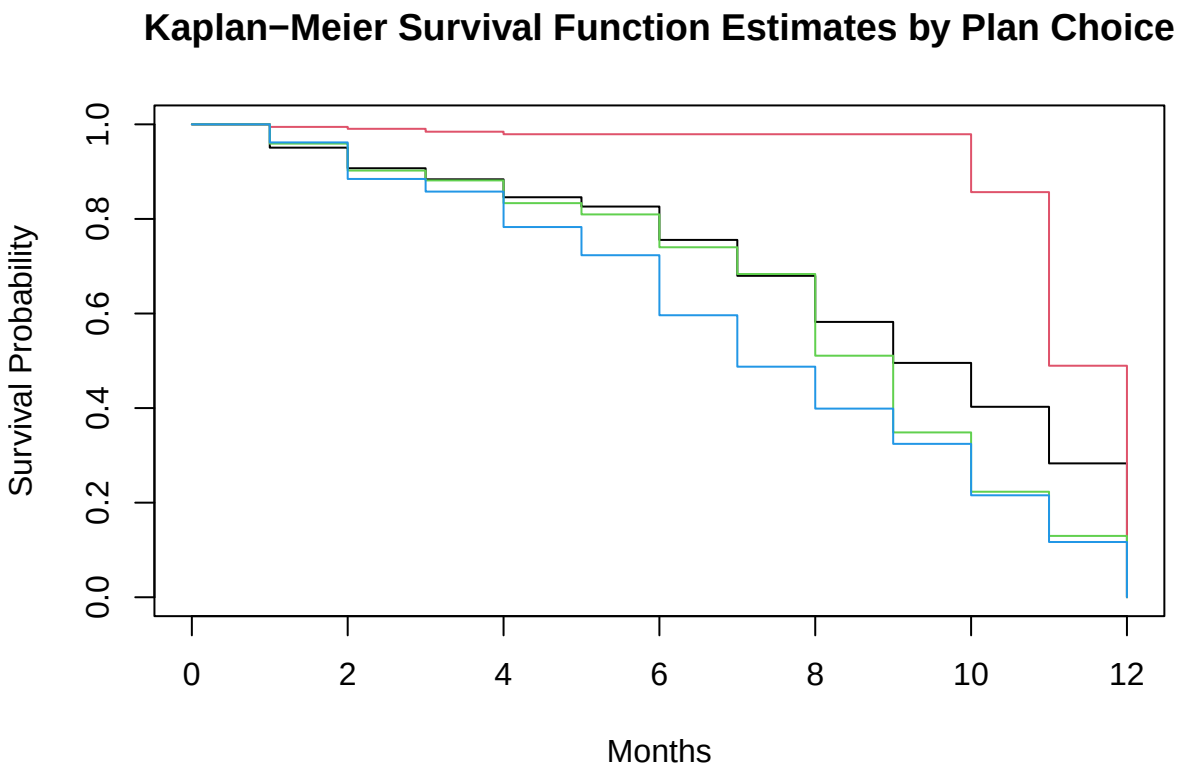
(a) Plot the Kaplan-Meier survival function estimates for each plan choice on the same graph

```
#install.packages("survival")
library(survival)
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
# Create a Surv object for survival analysis
surv_obj <- Surv(time = merged_data$bill_month, event = merged_data$churn)

# Compute the Kaplan-Meier survival function estimates for each plan choice
km_estimates <- survfit(surv_obj ~ merged_data$plan_chosen)

# Plot the Kaplan-Meier survival function estimates for each plan choice on the same graph
plot(km_estimates, col = 1:4, xlab = "Months", ylab = "Survival Probability", main = "Kaplan-Meier Surv
```



(b) Estimate the Cox proportional hazard (PH) type duration data models using a semi-parametric approach.

```
# Estimate the Cox PH model
cox_ph_model <- coxph(surv_obj ~ plan_chosen + ave_minute_peak + promo_pct + age + newcell + ruca2 + zip

# Display the results
summary(cox_ph_model)
```

```
## Call:
## coxph(formula = surv_obj ~ plan_chosen + ave_minute_peak + promo_pct +
##     age + newcell + ruca2 + zip_comm, data = merged_data)
##
##   n= 12319, number of events= 4848
##    (176 observations deleted due to missingness)
##
##                      coef  exp(coef)   se(coef)      z Pr(>|z|)
## plan_chosen      0.1120465  1.1185649  0.0156654  7.152 8.52e-13 ***
## ave_minute_peak  0.0002078  1.0002079  0.0001454  1.429  0.15290
## promo_pct        0.1226752  1.1305172  0.0593211  2.068  0.03864 *
## age             -0.0109161  0.9891432  0.0010917 -9.999  < 2e-16 ***
## newcell         -0.1348020  0.8738889  0.0488949 -2.757  0.00583 **
## ruca2            0.0275484  1.0279314  0.0089252  3.087  0.00202 **
## zip_comm         0.2008662  1.2224612  0.0998352  2.012  0.04422 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                 exp(coef) exp(-coef) lower .95 upper .95
## plan_chosen        1.1186     0.8940    1.0847    1.1534
## ave_minute_peak    1.0002     0.9998    0.9999    1.0005
## promo_pct          1.1305     0.8846    1.0064    1.2699
## age                0.9891     1.0110    0.9870    0.9913
## newcell            0.8739     1.1443    0.7940    0.9618
## ruca2              1.0279     0.9728    1.0101    1.0461
## zip_comm           1.2225     0.8180    1.0052    1.4867
##
## Concordance= 0.585  (se = 0.006 )
## Likelihood ratio test= 278.9  on 7 df,   p=<2e-16
## Wald test            = 272.8  on 7 df,   p=<2e-16
## Score (logrank) test = 274.5  on 7 df,   p=<2e-16
```

```
# Compute the AIC for the Cox PH model
cox_ph_aic <- AIC(cox_ph_model)
```

```
cox_ph_aic
```

```
## [1] 77001.51
```

(c) Estimate the accelerated failure time (AFT) type duration data models using the exponential and Weibull distributions.

```
# Estimate the AFT model with the exponential distribution
aft_exp_model <- survreg(surv_obj ~ plan_chosen + ave_minute_peak + promo_pct + age + newcell + ruca2 +

# Display the results
summary(aft_exp_model)
```

```
##
## Call:
## survreg(formula = surv_obj ~ plan_chosen + ave_minute_peak +
##     promo_pct + age + newcell + ruca2 + zip_comm, data = merged_data,
##     dist = "exponential")
##                     Value Std. Error     z       p
## (Intercept)      2.207795   0.075942 29.07 < 2e-16
## plan_chosen     -0.048248   0.015889 -3.04  0.0024
## ave_minute_peak -0.000925   0.000143 -6.45 1.1e-10
## promo_pct       -0.025705   0.059485 -0.43  0.6656
## age              0.013740   0.001094 12.56 < 2e-16
## newcell          0.416840   0.048551  8.59 < 2e-16
## ruca2           -0.043480   0.008944 -4.86 1.2e-06
## zip_comm        -0.185351   0.099721 -1.86  0.0631
##
## Scale fixed at 1
##
## Exponential distribution
## Loglik(model)= -17481.8   Loglik(intercept only)= -17725.5
##  Chisq= 487.45 on 7 degrees of freedom, p= 4e-101
## Number of Newton-Raphson Iterations: 5
## n=12319 (176 observations deleted due to missingness)
```

```
# Compute the AIC for the AFT model with the exponential distribution
aft_exp_aic <- AIC(aft_exp_model)

# Estimate the AFT model with the Weibull distribution
aft_weibull_model <- survreg(surv_obj ~ plan_chosen + ave_minute_peak + promo_pct + age + newcell + ruca

# Display the results
summary(aft_weibull_model)
```

```
##
## Call:
## survreg(formula = surv_obj ~ plan_chosen + ave_minute_peak +
##     promo_pct + age + newcell + ruca2 + zip_comm, data = merged_data,
##     dist = "weibull")
##                     Value Std. Error      z       p
## (Intercept)      2.11e+00   3.20e-02  65.95 < 2e-16
## plan_chosen     -2.94e-02   6.63e-03  -4.43 9.3e-06
## ave_minute_peak -1.77e-04   6.17e-05  -2.86 0.00426
## promo_pct       -3.16e-02   2.50e-02  -1.26 0.20645
## age              4.45e-03   4.66e-04   9.54 < 2e-16
## newcell          9.20e-02   2.07e-02   4.45 8.6e-06
## ruca2           -1.41e-02   3.78e-03  -3.74 0.00018
## zip_comm        -8.56e-02   4.21e-02  -2.03 0.04218
## Log(scale)      -8.62e-01   1.07e-02 -80.75 < 2e-16
##
## Scale= 0.422
##
## Weibull distribution
## Loglik(model)= -15403   Loglik(intercept only)= -15528
##  Chisq= 249.94 on 7 degrees of freedom, p= 2.9e-50
## Number of Newton-Raphson Iterations: 6
## n=12319 (176 observations deleted due to missingness)
```

```
# Compute the AIC for the AFT model with the Weibull distribution
aft_weibull_aic <- AIC(aft_weibull_model)
```

```
aft_weibull_aic
```

```
## [1] 30823.97
```

(d) Evaluate the estimated model fit and performance using the AIC measure.

```
# Compare the AIC values
cat("AIC values:\nCox PH model:", cox_ph_aic, "\nAFT Exponential model:", aft_exp_aic, "\nAFT Weibull m
```

```
## AIC values:
## Cox PH model: 77001.51
## AFT Exponential model: 34979.55
## AFT Weibull model: 30823.97
```

(e) Interpret either the hazard ratio (for PH model) or time ratio (for AFT model) associated with plan2 to plan 4 dummies (relative to plan 1) and age on the probability of or time to churning.

```r
# Compute time ratios for the best model (AFT Weibull)
time_ratios <- exp(coef(aft_weibull_model))
time_ratios
```

```
##     (Intercept)     plan_chosen ave_minute_peak       promo_pct             age
##       8.2692361       0.9710389       0.9998235       0.9688519       1.0044574
##         newcell           ruca2        zip_comm
##       1.0964031       0.9859539       0.9179447
```

```r
# Compute 95% confidence intervals for time ratios
conf_ints <- confint(aft_weibull_model, level = 0.95)
ci_lower <- exp(conf_ints[, 1])
ci_upper <- exp(conf_ints[, 2])

# Combine time ratios and their confidence intervals
time_ratios_ci <- data.frame(TimeRatio = time_ratios, LowerCI = ci_lower, UpperCI = ci_upper)
time_ratios_ci
```

```
##                 TimeRatio   LowerCI   UpperCI
## (Intercept)     8.2692361 7.7660449 8.8050309
## plan_chosen     0.9710389 0.9585009 0.9837410
## ave_minute_peak 0.9998235 0.9997025 0.9999445
## promo_pct       0.9688519 0.9224392 1.0176000
## age             1.0044574 1.0035396 1.0053760
## newcell         1.0964031 1.0528486 1.1417594
## ruca2           0.9859539 0.9786799 0.9932821
## zip_comm        0.9179447 0.8451745 0.9969805
```

Using random forest classifier to predict churn

```r
# Load the necessary library
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## 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
merged_data_clean$churn <- as.factor(merged_data_clean$churn)

# Split the data into training and test sets
set.seed(123)
trainIndex <- sample(1:nrow(merged_data_clean), round(0.7 * nrow(merged_data_clean)), replace = FALSE)
trainData <- merged_data_clean[trainIndex, ]
testData <- merged_data_clean[-trainIndex, ]

# Train the model
rf_model <- randomForest(churn ~., data = trainData, importance = TRUE)

# Make predictions on the test set
rf_pred <- predict(rf_model, testData)

# Evaluate the model performance
rf_cm <- table(Predicted = rf_pred, Actual = testData$churn)
rf_accuracy <- sum(diag(rf_cm)) / sum(rf_cm)
print(paste0("Random Forest Classifier Accuracy: ", round(rf_accuracy * 100, 2), "%"))
```

```
## [1] "Random Forest Classifier Accuracy: 100%"
```

```r
# Plot variable importance
varImpPlot(rf_model)
```