Untitled

Muhammad Enrizky Brillian_1009713712

2023-04-10

```
# 1. Merging Datasets from multiple CSVs
# loading libraries
library(tidyverse,dplyr,lubridate,knitr)
#Setting the working Directory
setwd("~/STAA57_Project")
# Create empty data frame
bikeshare_data <- data.frame()
# Loop through all CSV files in folder
for (file in list.files(pattern=".csv")) {
  temp_data <- read.csv(file, header=TRUE)  # Read in CSV file
  bikeshare_data <- rbind(bikeshare_data, temp_data)  # Bind data to the df
  rm(temp_data)  # Remove original data to free up memory}

# 2. Description of the Dataset
names(bikeshare_data)

# 3. The  Background of the Data: No Code in this Part

# 4. Overall Research Question: No Code in this Part

# 5 Tables
## 5.1 Trip Duration (in Seconds) Summary By User Type

# Create summary statistics of trip duration by user type
trip_duration_summary <- bikeshare_data %>% group_by(user_type) %>%
  summarise(
    Number_Trips = n(), Average_duration = mean(trip_duration_seconds),
    Median_duration = median(trip_duration_seconds),
    Min_duration = min(trip_duration_seconds),
    Max_duration = max(trip_duration_seconds))

# Display the summary table
kable(trip_duration_summary,
  caption = "Summary of Trip Duration (in seconds) by User Type",
  align=c("l","c","c","c","c"))

## 5.2 Table of Total Trips by Month for each user type

# Convert trip_start_time to a datetime object
bikeshare_data$trip_start_time <- as.POSIXct(bikeshare_data$trip_start_time,
                                             format = "%m/%d/%Y %H:%M")
```

```r
# Extract month from trip_start_time
bikeshare_data$trip_month <- month(bikeshare_data$trip_start_time, label = T)

# Create a table of trip count by month
trip_count_by_month <- bikeshare_data %>% drop_na() %>%
  count("Month"=trip_month, user_type) %>%
  pivot_wider(names_from = user_type, values_from = n)

# Print the table
kable(caption = "Total Trips by Month for each user type",trip_count_by_month)

## 5.3 The Top 10 Most Popular Startinh Stations

# Group the data by starting station and count the number of trips
starting_stations <- bikeshare_data %>%
  group_by("Starting Station"=from_station_name) %>%
  summarize(num_trips = n()) %>% arrange(desc(num_trips))

# Select the top 10 most popular starting stations
top_starting_stations <- head(starting_stations, 10)

# Create the table
kable(top_starting_stations, caption= "Top 10 Most Popular Starting stations")

## 5.4 Trip duration More than 30 minutes and More than 45 minutes

bikeshare_data %>%
  filter(trip_duration_seconds > 1800) %>% group_by(user_type) %>%
  summarise(">= 30 mins" = n()) %>% full_join(bikeshare_data %>%
      filter(trip_duration_seconds > 2700) %>% group_by(user_type) %>%
        summarise(">= 45 mins" = n()), by = "user_type") %>%
  kable(caption = "Trip Duration more than 30 minutes and 45 minutes.")

## 6.1 Line Chart of Total Trips by Hour

# Extract day and hour from trip_start_time
bikeshare_data <- bikeshare_data %>%
  mutate(trip_start_hour = hour(trip_start_time),
         trip_day = wday(trip_start_time, label = T, abbr = F))

# summarize data by hour of day
hourly_trips <- bikeshare_data %>%
  group_by(trip_start_hour) %>%
  summarize(total_trips = n())

# create line chart of total trips by hour
ggplot(hourly_trips, aes(x = trip_start_hour, y = total_trips)) +
  geom_line() + ggtitle("Total Trips by Hour") +
  xlab("Hours (0-24)") + ylab("Total Trips")

## 6.2 Stacked bar plot of trip volume by user type and month

# group by month and user type and calculate the total trips
```

```r
monthly_user_trips <- bikeshare_data %>% group_by(trip_month, user_type) %>%
  summarize(total_trips = n())

# create the stacked bar plot
ggplot(monthly_user_trips, aes(x=trip_month, y=total_trips, fill = user_type))
+ geom_bar(stat = "identity") +
  labs(title = "Monthly Trip Volume by User Type",
       x = "", y = "Total Trips", fill = "User Type")

## 6.3 Heatmap of trips by day of the week

# Aggregate the data by day of the week and hour
heatmap_data <- bikeshare_data %>% group_by(trip_start_hour, trip_day) %>%
  summarize(trip_count = n())

# Create the heatmap
ggplot(heatmap_data, aes(x=trip_start_hour, y=trip_day, fill = trip_count)) +
  geom_tile(color = "white", size = 0.5) +
  scale_fill_gradient(low = "yellow", high = "red") +
  labs(x = "Hour of Day", y = "", fill = "Number of Trips") + theme_bw() +
  theme(axis.text.x = element_text(angle = 0, hjust = 0.5),
        panel.grid.major = element_line(color = "black", size = 1),
        panel.grid.minor = element_line(color = "black", size = 0.5))

# 7. Hypothesis Testing
## Trip duration Between Trips taken on weekdays versus weekends.

# Subset data to separate trips taken on weekdays and weekends
weekdays <- subset(bikeshare_data, !trip_day %in% c("Saturday", "Sunday"))
weekends <- subset(bikeshare_data, trip_day %in% c("Saturday", "Sunday"))
# Conduct two-sample t-test
t.test(x=weekdays$trip_duration_seconds, y=weekends$trip_duration_seconds,
       var.equal = FALSE)

# 8. Bootstrapping
## Estimating the average trip duration for casual members

# Define a function to calculate the average trip duration
mean_duration <- function(data) {mean(data$trip_duration_seconds)}

# Subset data for casual members
casual_data <- bikeshare_data %>% filter(user_type == "Casual Member")

# Use bootstrapping to estimate the mean trip duration
set.seed(123)
n_boot <- 100
bootstrap_means <- replicate(n_boot, casual_data %>%
                             sample_n(nrow(casual_data), replace=TRUE) %>%
                             mean_duration())

# Calculate the 95% confidence interval
ci <- quantile(bootstrap_means, c(0.025, 0.975))
```

```r
# Print the results
cat("Mean trip duration for casual members:", mean(casual_data$trip_duration_seconds), "\n")
cat("95% confidence interval:","( ", ci[1], "-", ci[2]," )", "\n")

# 9. Non-Linear Regression Analysis
## Analyzing the relationship between the number of trips and hour of the day.

# Group the data by hour and count the number of trips for each hour
trips_by_hour <- bikeshare_data %>% group_by(trip_start_hour) %>%
  summarize(num_trips = n())

# Use Poly regression to analyze between the total trips and hour of the day
model <- lm(num_trips ~ trip_start_hour + I(trip_start_hour^2)
            + I(trip_start_hour^3), data = trips_by_hour)

# Print the summary of the model
summary(model)

# Print the summary of the model
summary(model)
# Plotting The Result
ggplot(trips_by_hour, aes(x=trip_start_hour, y=num_trips)) +
  geom_point() + stat_smooth()

# 10. Cross Validation
## Analyzing between Trip Duration (in hour) and hour of the day
# Calculate average trip duration (in hour) by hour of the day
avg_duration <- bikeshare_data %>% group_by(trip_start_hour) %>%
  summarize(avg_trip_duration = mean(trip_duration_seconds/3600))

# Define the model formula
formula <- avg_trip_duration ~ poly(trip_start_hour,degree = 3, raw = T)

# Define the number of folds for cross-validation
k <- 10

# Initialize a vector to store the mean squared errors for each fold
cv_mse <- rep(0, k)

# Define the indices for each fold
set.seed(123)
folds <- cut(seq(1, nrow(avg_duration)), breaks = k, labels = FALSE)

# Perform k-fold cross-validation
for (i in 1:k) {
  # Split the data into training and test sets for this fold
  test_indices <- which(folds == i, arr.ind = TRUE)
  test_set <- avg_duration[test_indices, ]
  train_set <- avg_duration[-test_indices, ]

  # Fit the model to the training set
  model <- lm(formula, data = train_set)
```

```r
  # Use the model to predict the test set
  predictions <- predict(model, newdata = test_set)

  # Calculate the mean squared error for this fold
  mse <- mean((predictions - test_set$avg_trip_duration)^2)

  # Store the mean squared error in the cv_mse vector
  cv_mse[i] <- mse}

# Calculate the average mean squared error over all folds
avg_mse <- mean(cv_mse)

# Print the average mean squared error over all folds
cat("The Average MSE is:",avg_mse,"\n")
```