AMS597 Project - Driven to Predict: Racing meets Statistics

Shubham Kishore Kale, Kashish Deepak Lalwani, Parth Satish Chavan, Naman Deep

Load Libraries

```
# List of required packages
required_packages <- c(
  "dplyr", "tidyr", "lubridate", "caret", "xgboost", "Matrix",
  "ggplot2", "ggthemes", "cowplot", "tidyverse", "readr",
  "cluster", "factoextra", "knitr", "kableExtra", "stats", "vcd", "binom"
)

# Function to install and load a package with full output suppression
install_and_load <- function(pkg) {
  if (!require(pkg, character.only = TRUE)) {
    invisible(capture.output(
      suppressMessages(suppressWarnings(install.packages(pkg, dependencies = TRUE)))
    ))
  }
  invisible(capture.output(
    suppressMessages(suppressWarnings(library(pkg, character.only = TRUE)))
  ))
}

# Apply to all required packages
invisible(lapply(required_packages, install_and_load))
```

Load Data

```
# Load datasets
results <- read.csv("results.csv")
drivers <- read.csv("drivers.csv")
constructors <- read.csv("constructors.csv")
races <- read.csv("races.csv")
circuits <- read.csv("circuits.csv")
lap_times <- read.csv("lap_times.csv")
pit_stops <- read.csv("pit_stops.csv")
```

Research Question 1: Lap time prediction

Using historical lap times, circuit characteristics (layout, length, altitude), and driver performance data (qualifying positions, constructor, driver standings, etc.), build a predictive model to estimate a driver's lap time for each lap of a race.

```
# Merge all necessary datasets
df <- lap_times %>%
  select(raceId, driverId, lap, milliseconds) %>%
  left_join(results %>% select(raceId, driverId, constructorId, grid),
            by = c("raceId", "driverId")) %>%
  left_join(drivers %>% select(driverId, dob), by = "driverId") %>%
  left_join(constructors %>% select(constructorId, constructorRef), by = "constructorId") %>%
  left_join(races %>% select(raceId, year, round, circuitId, date, time),
            by = "raceId") %>%
  left_join(circuits %>% select(circuitId, circuitRef, lat, lng, alt),
            by = "circuitId") %>%
  left_join(pit_stops %>% select(raceId, driverId, lap, stop, milliseconds),
            by = c("raceId", "driverId", "lap"), suffix = c("", "_pit"))
```

Data Exploration

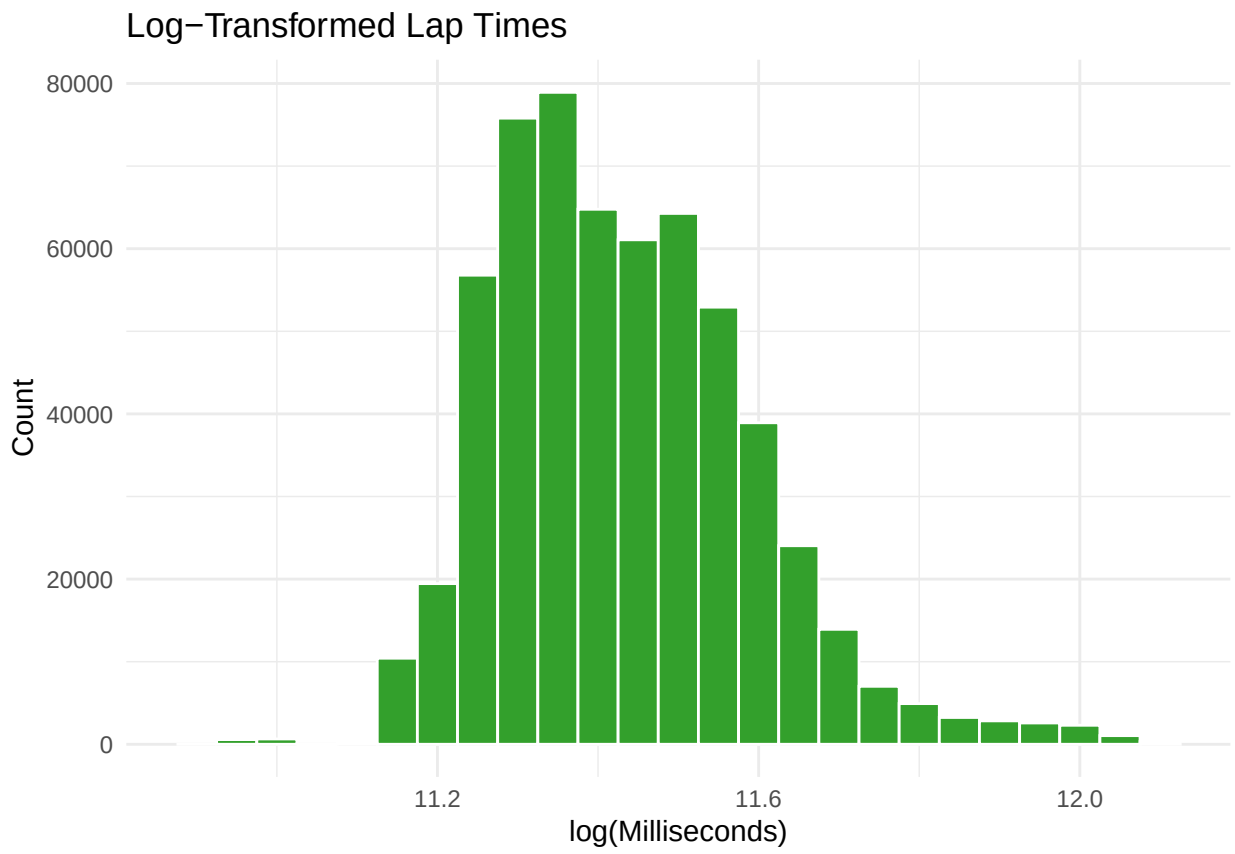
```
# Summary of target variable
print("Lap time (Target Variable) summary:")
```

```
## [1] "Lap time (Target Variable) summary:"
```

```
summary(df$milliseconds)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   55404   82041   90608   95799  101930 7507547
```

```
# Visual: Log transformation justification
ggplot(df %>% filter(milliseconds <= 180000), aes(x = log(milliseconds))) +
  geom_histogram(binwidth = 0.05, fill = "#33a02c", color = "white") +
  labs(title = "Log-Transformed Lap Times", x = "log(Milliseconds)", y = "Count") +
  theme_minimal()
```




```
# Missing values
na_summary <- sapply(df, function(x) sum(is.na(x)))
print(na_summary)
```

```
##           raceId         driverId              lap     milliseconds
##                0                0                0                0
##    constructorId             grid              dob   constructorRef
##                0                0                0                0
##             year            round        circuitId             date
##                0                0                0                0
##             time       circuitRef              lat              lng
##                0                0                0                0
##              alt             stop milliseconds_pit
##                0           577710           577710
```

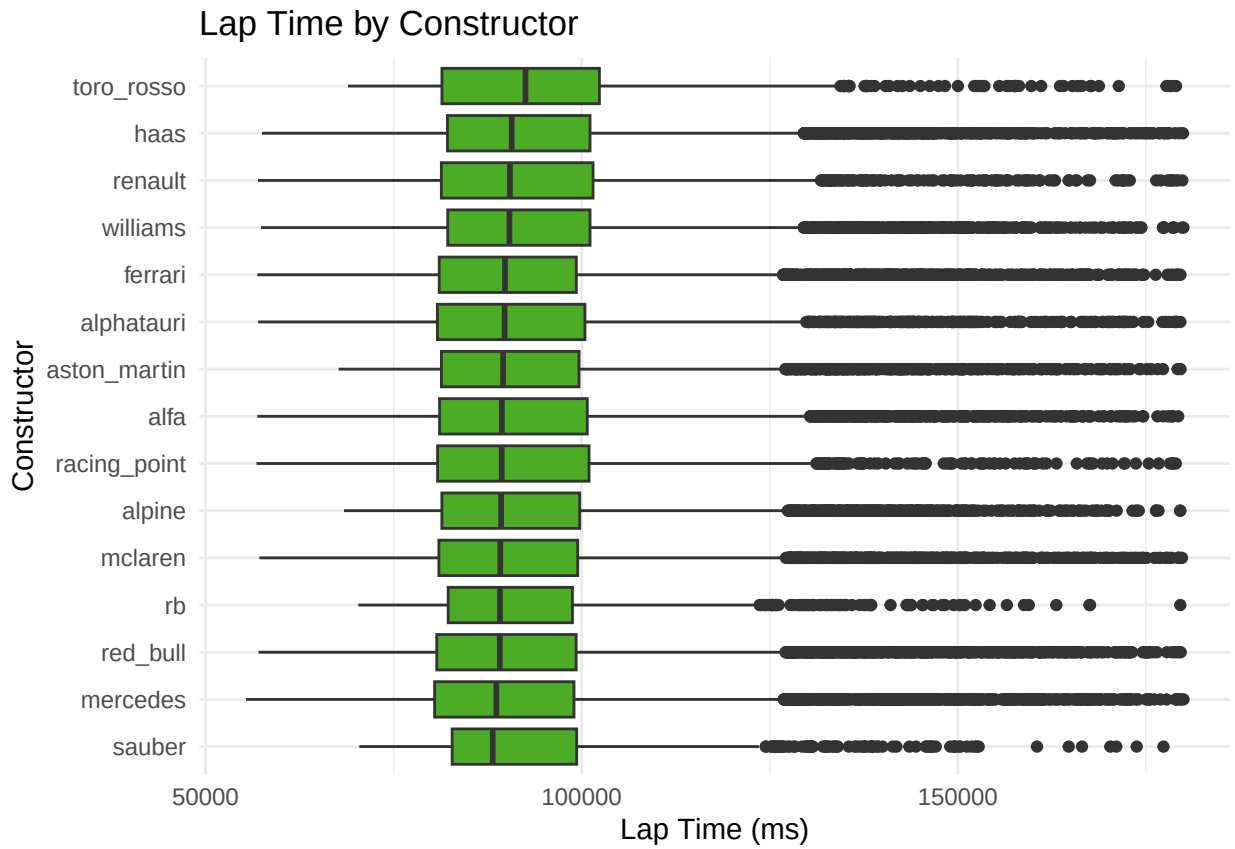
Cleaning and Feature Engineering

```
df <- df %>%
  filter(year >= 2019, milliseconds <= 180000) %>%  # Valid upper bound = 3 minutes
  rename(milliseconds_lap = milliseconds) %>%
  mutate(
    dob = ymd(dob),
    race_date = ymd(date),
```

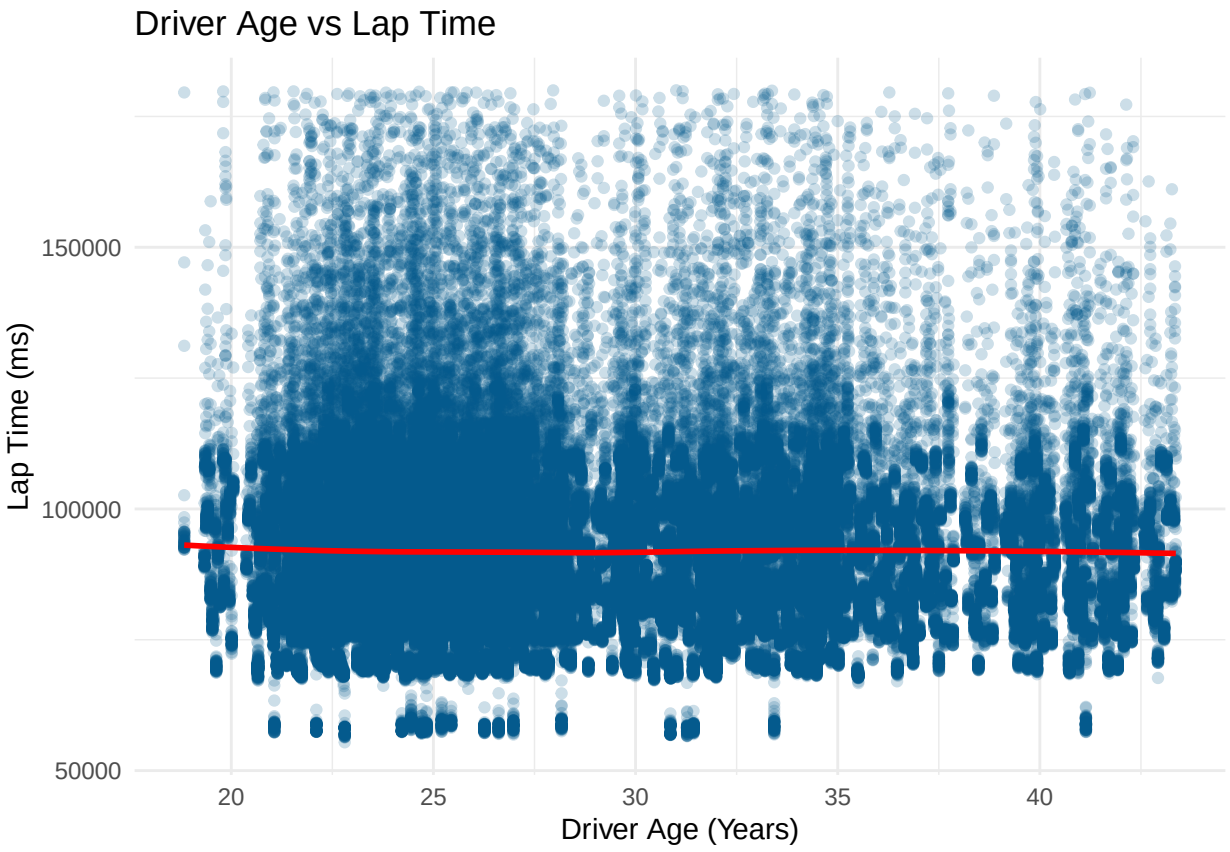
```
    race_time = hms(time),
    race_hour = hour(race_time),
    day_of_week = wday(race_date, label = TRUE),
    driver_age = as.numeric(difftime(race_date, dob, units = "days")) / 365.25,
    is_pit_lap = !is.na(milliseconds_pit),
    milliseconds_pit = ifelse(is.na(milliseconds_pit), 0, milliseconds_pit),
    stop = ifelse(is.na(stop), 0, stop),
    grid = ifelse(is.na(grid), 0, grid)
  )

# Lap context
df <- df %>%
  arrange(raceId, driverId, lap) %>%
  group_by(raceId, driverId) %>%
  mutate(
    lap_fraction = lap / max(lap),
    cumulative_pit = lag(cumsum(milliseconds_pit), default = 0),
    lap_time_delta = milliseconds_lap - lag(milliseconds_lap, default = milliseconds_lap[1]),
    lap_bin = cut(lap, breaks = c(0, 15, 35, Inf), labels = c("early", "mid", "late"))
  ) %>% ungroup()

# Global driver average
global_driver_avg <- df %>%
  group_by(driverId) %>%
  summarize(global_avg_lap = mean(milliseconds_lap, na.rm = TRUE))

df <- df %>%
  left_join(global_driver_avg, by = "driverId")

df <- df %>%
  arrange(raceId, driverId, lap) %>%
  group_by(raceId, driverId) %>%
  mutate(
    driver_avg_to_lap = lag(cummean(milliseconds_lap), default = first(global_avg_lap)),
    driver_delta = milliseconds_lap - driver_avg_to_lap
  ) %>% ungroup()

# Constructor average
constructor_avg <- df %>%
  group_by(constructorId) %>%
  summarize(constructor_avg_lap = mean(milliseconds_lap, na.rm = TRUE))

df <- df %>% left_join(constructor_avg, by = "constructorId")
```

EDA Visuals

```
# Boxplot of lap times by constructor
ggplot(df, aes(x = reorder(constructorRef, milliseconds_lap, median), y = milliseconds_lap)) +
  geom_boxplot(fill = "#4dac26") +
  coord_flip() +
  labs(title = "Lap Time by Constructor", x = "Constructor", y = "Lap Time (ms)") +
  theme_minimal()
```



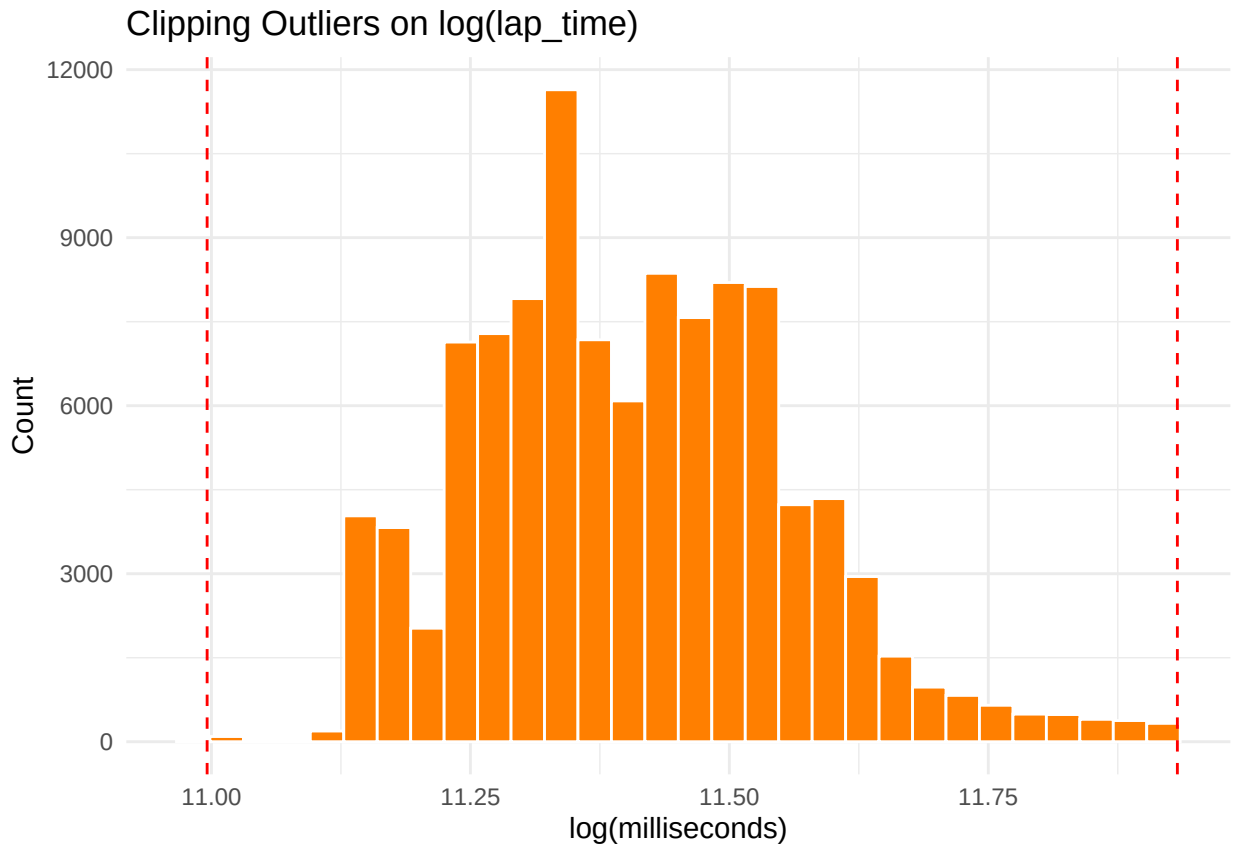

```r
# Driver age vs lap time
ggplot(df, aes(x = driver_age, y = milliseconds_lap)) +
  geom_point(alpha = 0.2, color = "#045a8d") +
  geom_smooth(method = "loess", se = FALSE, color = "red") +
  labs(title = "Driver Age vs Lap Time", x = "Driver Age (Years)", y = "Lap Time (ms)") +
  theme_minimal()
```



Final Modeling Dataset

```
model_df <- df %>%
  filter(!is_pit_lap) %>%
  select(milliseconds_lap, year, grid, lap, lap_fraction, cumulative_pit,
         lat, lng, alt, race_hour, day_of_week, driver_age,
         driver_avg_to_lap, constructor_avg_lap, driver_delta,
         lap_time_delta, lap_bin,
         driverId, constructorId, circuitRef) %>%
  drop_na()

model_df <- model_df %>%
  mutate(across(c(driverId, constructorId, circuitRef, day_of_week, lap_bin), as.factor)) %>%
  mutate(log_lap_time = log(milliseconds_lap))

# Train-test split
train_data <- model_df %>% filter(year %in% 2019:2023)
test_data  <- model_df %>% filter(year == 2024)

# Outlier clipping (even after filtering)
q_lo <- quantile(train_data$log_lap_time, 0.01)
q_hi <- quantile(train_data$log_lap_time, 0.99)
train_data <- train_data %>% filter(log_lap_time >= q_lo, log_lap_time <= q_hi)
```

```r
# Visual explanation for clipping
ggplot(train_data, aes(x = log_lap_time)) +
  geom_histogram(fill = "#ff7f00", color = "white", bins = 30) +
  geom_vline(xintercept = c(q_lo, q_hi), color = "red", linetype = "dashed") +
  labs(title = "Clipping Outliers on log(lap_time)", x = "log(milliseconds)", y = "Count") +
  theme_minimal()
```

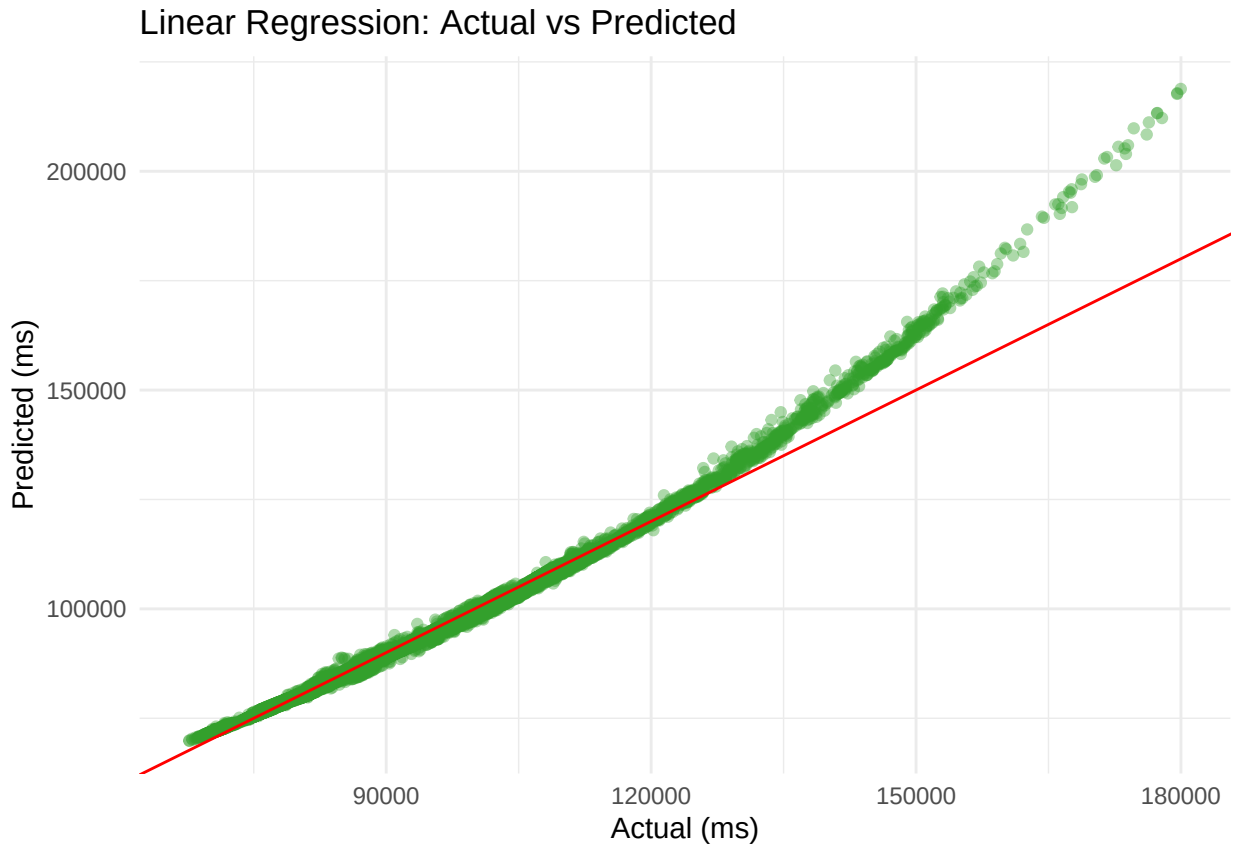


Encoding for Models

```r
encode_for_xgb <- function(df) {
  df_num <- df %>% select(-milliseconds_lap, -year)
  for (col in names(df_num)) {
    if (is.factor(df_num[[col]])) {
      df_num[[col]] <- as.numeric(as.factor(df_num[[col]]))
    }
  }
  return(df_num)
}

train_mat <- encode_for_xgb(train_data)
test_mat <- encode_for_xgb(test_data)
```

Baseline: Linear Regression

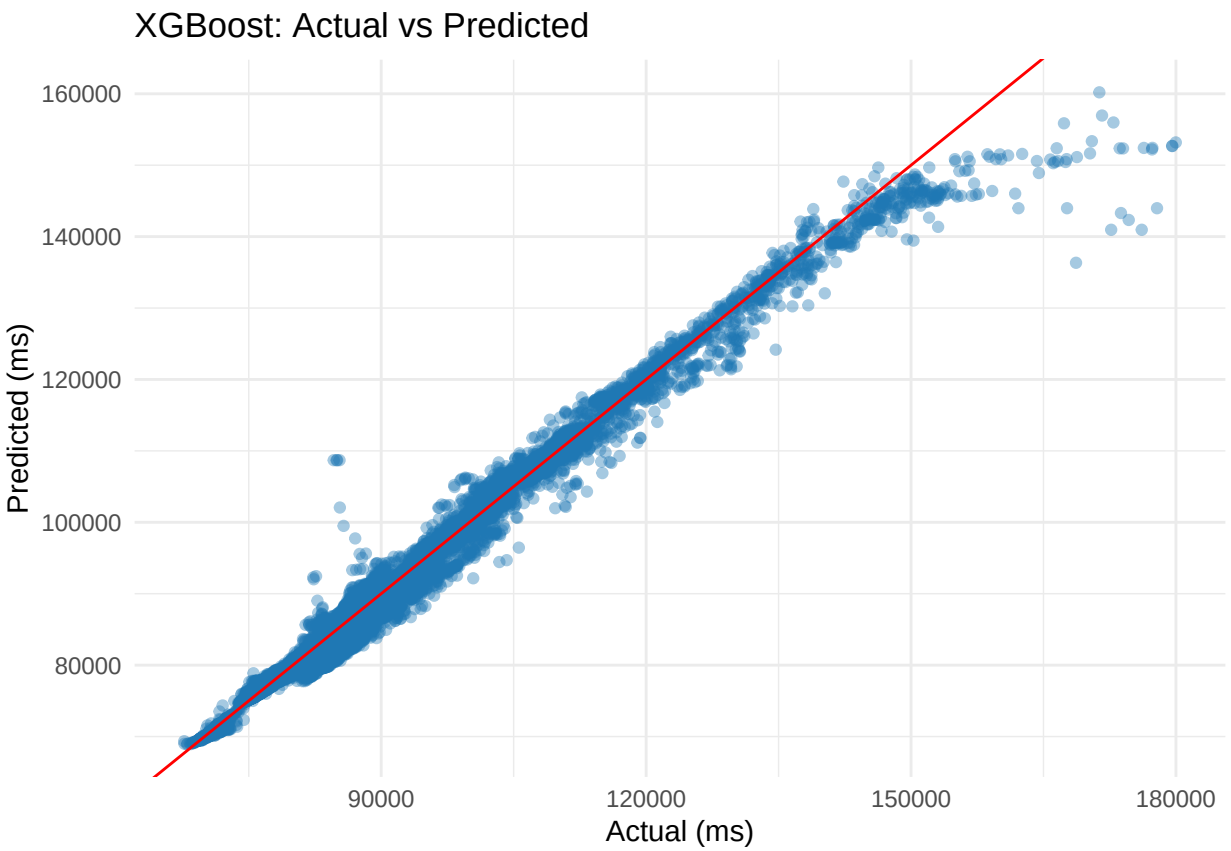
```
lm_model <- lm(log_lap_time ~ ., data = train_mat)
pred_log_lm <- predict(lm_model, newdata = test_mat)
pred_ms_lm <- exp(pred_log_lm)
rmse_lm <- RMSE(pred_ms_lm, test_data$milliseconds_lap)

# Plot predictions
ggplot(data.frame(Actual = test_data$milliseconds_lap, Predicted = pred_ms_lm),
       aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.4, color = "#33a02c") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "Linear Regression: Actual vs Predicted", x = "Actual (ms)", y = "Predicted (ms)") +
  theme_minimal()
```



XGBoost Regression

```
dtrain <- xgb.DMatrix(data = as.matrix(train_mat %>% select(-log_lap_time)),
                      label = train_mat$log_lap_time)
dtest <- xgb.DMatrix(data = as.matrix(test_mat %>% select(-log_lap_time)))

params <- list(
  objective = "reg:squarederror",
```

```r
  eval_metric = "rmse",
  eta = 0.03,
  max_depth = 6,
  subsample = 0.8,
  colsample_bytree = 0.8,
  min_child_weight = 5,
  gamma = 0.1
)

set.seed(42)
cv <- xgb.cv(params = params, data = dtrain, nrounds = 300,
             nfold = 5, early_stopping_rounds = 15, verbose = 0)

best_nrounds <- cv$best_iteration
final_model_xgb <- xgb.train(params = params, data = dtrain, nrounds = best_nrounds)

pred_log_xgb <- predict(final_model_xgb, dtest)
pred_ms_xgb <- exp(pred_log_xgb)
rmse_xgb <- RMSE(pred_ms_xgb, test_data$milliseconds_lap)

# XGBoost Prediction Plot
ggplot(data.frame(Actual = test_data$milliseconds_lap, Predicted = pred_ms_xgb),
       aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.4, color = "#1f78b4") +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "XGBoost: Actual vs Predicted", x = "Actual (ms)", y = "Predicted (ms)") +
  theme_minimal()
```



Model Comparison

```
cat("Model Performance (RMSE):\n")
```

```
## Model Performance (RMSE):
```

```
cat("Linear Regression: ", round(rmse_lm, 2), "\n")
```

```
## Linear Regression:  1934.93
```

```
cat("XGBoost          : ", round(rmse_xgb, 2), "\n")
```

```
## XGBoost          :  1600.35
```

Research Question 2: Clustering Drivers by Racing Style

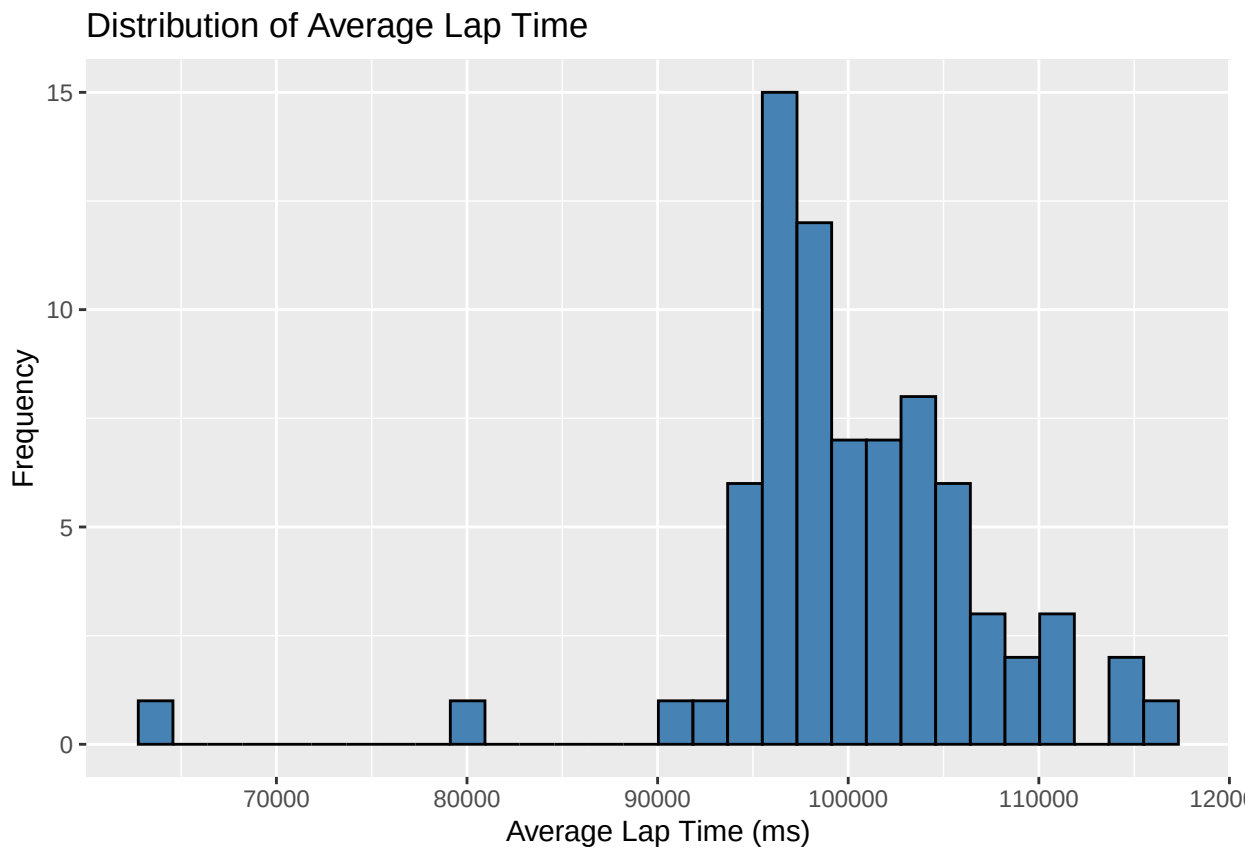
How can data-driven methods methods be used to identify different types of Formula 1 drivers based on their race performance, and what do these groupings reveal about common racing styles and strategies?

Feature Engineering

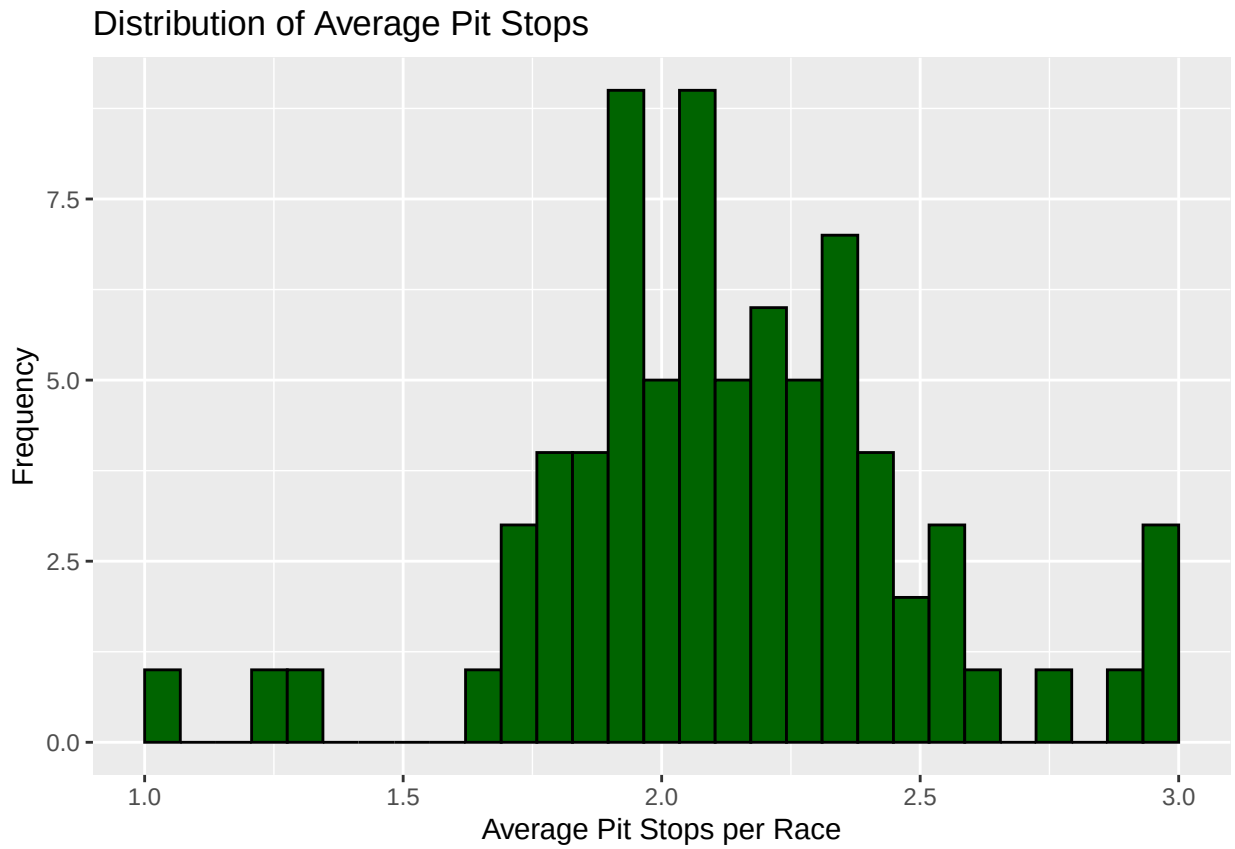
```
# Merge lap_times with races to add race details
races <- races %>%filter(year > 2010)
lap_races <- lap_times %>%
  inner_join(races, by = "raceId")

# Calculate average lap time and its standard deviation for each driver
lap_summary <- lap_races %>%
  group_by(driverId) %>%
  summarise(avg_lap_time = mean(milliseconds, na.rm = TRUE),
            sd_lap_time  = sd(milliseconds, na.rm = TRUE))

# Calculate pit stop frequency: average number of pit stops per race for each driver
pit_summary <- pit_stops %>%
  group_by(driverId, raceId) %>%
  summarise(num_pit_stops = n(), .groups = "drop") %>%
  group_by(driverId) %>%
  summarise(avg_pit_stops = mean(num_pit_stops, na.rm = TRUE))

# Calculate overtaking statistics: positions gained = grid position minus finishing position
overtake_summary <- results %>%
  mutate(positions_gained = grid - positionOrder) %>%
  group_by(driverId) %>%
  summarise(avg_positions_gained = mean(positions_gained, na.rm = TRUE))

# Merge all summaries with driver details
driver_features <- drivers %>%
  select(driverId, forename, surname, nationality) %>%
  inner_join(lap_summary, by = "driverId") %>%
  inner_join(pit_summary, by = "driverId") %>%
  inner_join(overtake_summary, by = "driverId") %>%
  # Create a single name column for clarity
  mutate(driver_name = paste(forename, surname))%>%
  select(-forename, -surname)
```

EDA

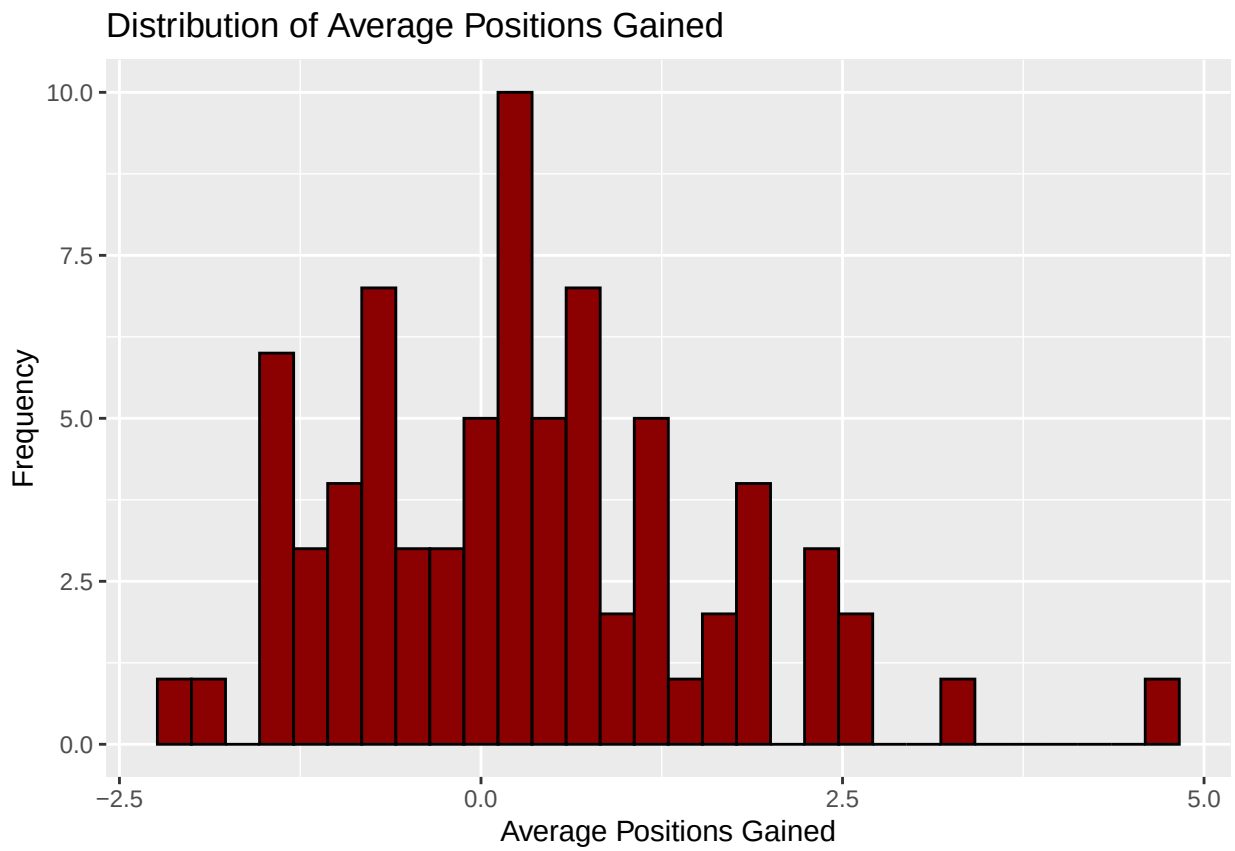
```
# Histogram for Average Lap Time
ggplot(driver_features, aes(x = avg_lap_time)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  labs(title = "Distribution of Average Lap Time", x = "Average Lap Time (ms)", y = "Frequency")
```




```
# Histogram for Average Pit Stop Frequency
ggplot(driver_features, aes(x = avg_pit_stops)) +
  geom_histogram(bins = 30, fill = "darkgreen", color = "black") +
  labs(title = "Distribution of Average Pit Stops", x = "Average Pit Stops per Race", y = "Frequency")
```




```r
# Histogram for Average Positions Gained (Overtaking Metric)
ggplot(driver_features, aes(x = avg_positions_gained)) +
  geom_histogram(bins = 30, fill = "darkred", color = "black") +
  labs(title = "Distribution of Average Positions Gained", x = "Average Positions Gained", y = "Frequen
```



Observations:

1. Average Lap Time

Distribution shape: Left-skewed.

Range: ~70,000 ms to 120,000 ms.

Insight: Most drivers cluster around 90,000–100,000 ms; very few extremely fast or slow drivers exist. Outliers: Some drivers have extremely low average lap times (e.g., ~72,000 ms) likely due to partial race data or anomalies.

2. Average Pit Stops

Distribution shape: Bell-like with mild skew.

Range: ~1.5 to ~3 pit stops per race.

Insight: Majority of drivers average around 2–2.5 pit stops, suggesting a common race strategy; a few edge cases have very high pit stops.

3. Average Positions Gained

Distribution shape: Roughly centered, with both positive and negative values.

Range: ~-2.5 to +5 positions gained per race.

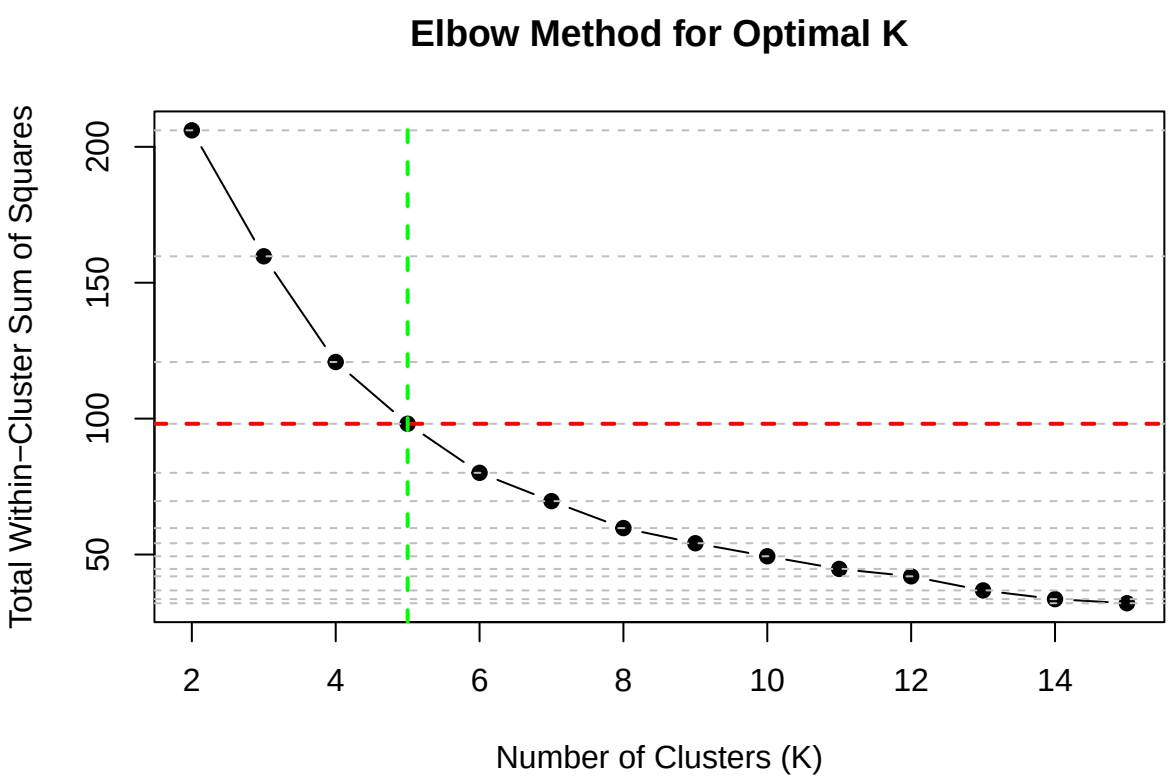
Insight: Most drivers hover around 0, indicating minimal positional change, but some consistently overtake more than 3–4 cars per race — suggesting aggressive racecraft.

K-means Clustering

```
# Select features for clustering analysis
features <- driver_features %>%
  select(avg_lap_time, sd_lap_time, avg_pit_stops, avg_positions_gained)

# Standardize features (zero mean, unit variance)
features_scaled <- scale(features)
```

Finding Optimal K value for clustering - Elbow Method

```
# Using the Elbow Method to identify the appropriate number of clusters
wss <- numeric(15)
for (k in 2:15) {
  set.seed(123)
  kmeans_model <- kmeans(features_scaled, centers = k, nstart = 50,iter.max = 100)
  wss[k] <- kmeans_model$tot.withinss
}
# Plot the Elbow Curve
plot(2:15, wss[2:15], type = "b", pch = 19,
     xlab = "Number of Clusters (K)", ylab = "Total Within-Cluster Sum of Squares",
     main = "Elbow Method for Optimal K")

for (y in wss[2:15]) {
  abline(h = y, lty = "dashed", col = "gray")
}

abline(h = wss[5], lty = "dashed", col = "red", lwd = 2)
abline(v = 5, lty = "dashed", col = "green", lwd = 2)
```



Observation:

Optimal Value of K - Method Used: Elbow Method

Plot: WSS (Within-Cluster Sum of Squares) vs. number of clusters k

A visible elbow point appears at k=5, where the reduction in WSS starts flattening.

A red dashed horizontal line at WSS for K = 5 and a green vertical line at k=5 visually emphasize this optimal point.

Hence chosen k=5 based on the Elbow Method's inflection point.

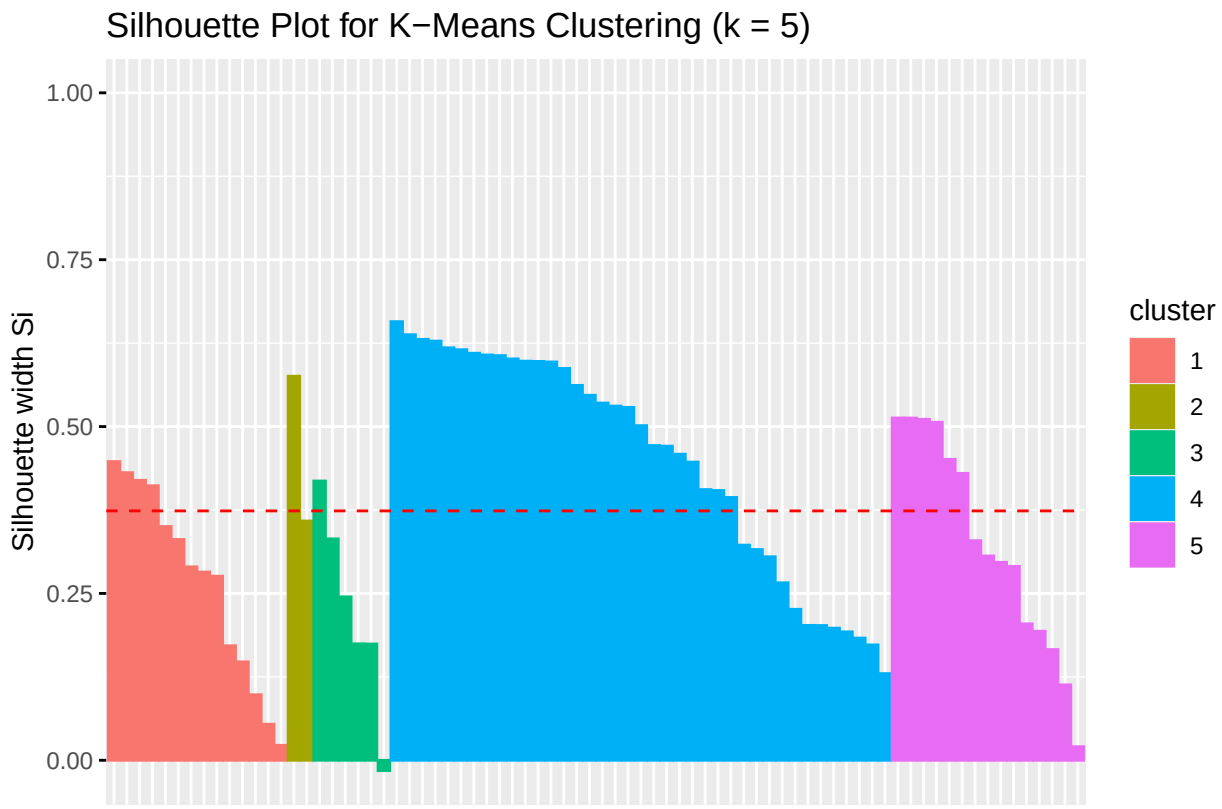
Cluster Validation and Visualisation

```
# Perform K-Means clustering with k = 5 clusters
set.seed(123)
kmeans_result <- kmeans(features_scaled, centers = 5, nstart = 25, iter.max = 100)

driver_features$cluster <- as.factor(kmeans_result$cluster)

sil <- silhouette(kmeans_result$cluster, dist(features_scaled))
mean_sil_width <- mean(sil[, 3])

cat("Average Silhouette Score for k = 5:", mean_sil_width, "\n")
```

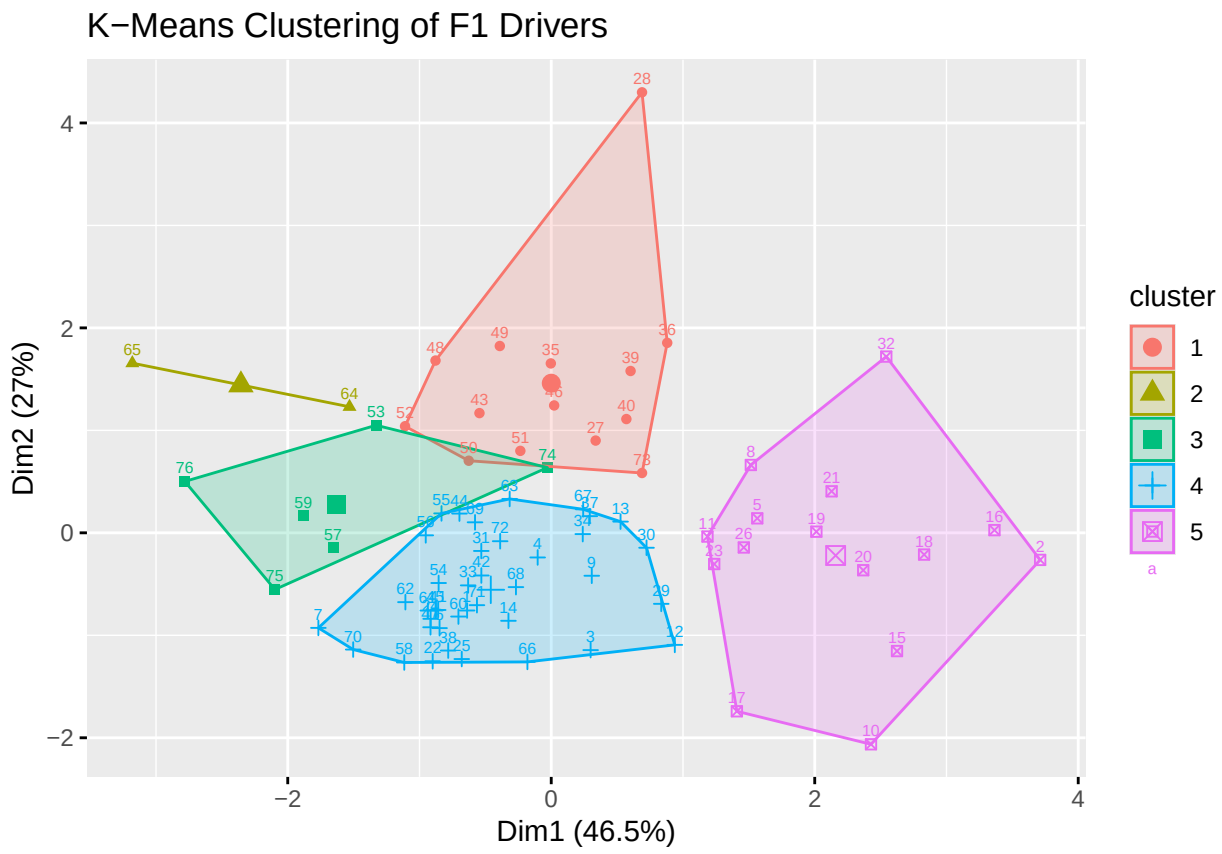
```
## Average Silhouette Score for k = 5: 0.3736497
```

```
print(fviz_silhouette(sil) +
  labs(title = "Silhouette Plot for K-Means Clustering (k = 5)"))
```

```
##   cluster size ave.sil.width
## 1       1   14          0.27
## 2       2    2          0.47
## 3       3    6          0.22
## 4       4   39          0.45
## 5       5   15          0.32
```




```
# Visualize the clustering result using PCA for dimensionality reduction
fviz_cluster(kmeans_result, data = features_scaled,
             ellipse.type = "convex",
             labelsize = 6,
             main = "K-Means Clustering of F1 Drivers")
```




```
# Cluster Characteristics describing each cluster
cluster_characteristics <- driver_features %>%
  group_by(cluster) %>%
  summarise(
    count = n(),
    mean_avg_lap_time = mean(avg_lap_time, na.rm = TRUE),
    mean_sd_lap_time = mean(sd_lap_time, na.rm = TRUE),
    mean_avg_pit_stops = mean(avg_pit_stops, na.rm = TRUE),
    mean_positions_gained = mean(avg_positions_gained, na.rm = TRUE)
  )

kable(cluster_characteristics,digits = 3, caption = "Cluster Characteristics")%>%
  kable_styling(font_size = 8)
```

Table 1: Cluster Characteristics

cluster	count	mean_avg_lap_time	mean_sd_lap_time	mean_avg_pit_stops	mean_positions_gained
1	14	101669.55	57707.78	2.339	1.902
2	2	72055.99	17598.05	3.000	0.750
3	6	96857.46	32705.50	1.439	1.260
4	39	98061.33	85974.30	2.044	-0.433
5	15	109244.72	219189.76	2.348	0.359

```
kable(driver_features,digits = 3, caption = "Driver-Level Features")%>%
kable_styling(font_size = 8)
```

Table 2: Driver-Level Features

driverId	nationality	avg_lap_time	sd_lap_time	avg_pit_stops	avg_positions_gained	driver_name	cluster
1	British	97150.49	79852.731	2.018	-0.719	Lewis Hamilton	4
2	German	114345.90	316219.790	2.500	0.614	Nick Heidfeld	5
3	German	101174.64	119204.758	2.243	-1.350	Nico Rosberg	4
4	Spanish	99371.30	105738.942	2.070	0.087	Fernando Alonso	4
5	Finnish	106103.98	171893.177	2.475	0.455	Heikki Kovalainen	5
8	Finnish	97348.30	64198.663	1.956	-0.963	Kimi Räikkönen	4
9	Polish	93804.59	15458.100	1.826	-1.051	Robert Kubica	4
10	German	108274.67	182992.281	2.083	1.768	Timo Glock	5
13	Brazilian	100831.07	102300.502	2.405	-0.620	Felipe Massa	4
15	Italian	111765.83	247326.133	2.235	-1.957	Jarno Trulli	5
16	German	104377.40	160310.469	2.373	0.273	Adrian Sutil	5
17	Australian	102291.36	144639.307	2.536	-1.525	Mark Webber	4
18	British	101869.52	118741.354	2.333	0.311	Jenson Button	4
20	German	98157.58	98495.653	2.067	-0.827	Sebastian Vettel	4
22	Brazilian	108803.16	233874.270	2.778	-1.451	Rubens Barrichello	5
24	Italian	116068.59	257550.893	2.562	0.642	Vitantonio Liuzzi	5
30	German	105526.32	184436.162	2.368	-2.013	Michael Schumacher	5
37	Spanish	114928.65	253501.126	2.176	0.785	Pedro de la Rosa	5
39	Indian	110550.59	221564.832	2.042	1.104	Narain Karthikeyan	5
67	Swiss	108222.47	244144.366	2.412	0.200	Sébastien Buemi	5
153	Spanish	106436.78	232906.570	2.389	1.261	Jaime Alguersuari	5
154	French	97862.24	57824.000	1.942	-1.436	Romain Grosjean	4
155	Japanese	106405.89	172272.424	2.140	0.263	Kamui Kobayashi	5
842	French	96090.07	82023.134	1.814	-0.558	Pierre Gasly	4
807	German	97901.10	71909.741	2.021	-1.435	Nico Hülkenberg	4
808	Russian	105919.85	177306.875	2.351	0.241	Vitaly Petrov	5
811	Brazilian	103325.62	80057.538	2.346	1.217	Bruno Senna	1
812	Indian	105481.09	8993.219	3.000	4.818	Karun Chandhok	1
813	Venezuelan	103471.59	133751.838	2.407	-0.844	Pastor Maldonado	4
814	British	103049.43	144158.476	2.190	0.271	Paul di Resta	4
815	Mexican	98009.26	79542.754	2.015	0.110	Sergio Pérez	4
816	Belgian	110940.72	231547.012	2.333	3.200	Jérôme d'Ambrosio	5
817	Australian	98227.58	78503.639	1.927	-0.257	Daniel Ricciardo	4
818	French	102204.68	93505.766	2.273	0.086	Jean-Éric Vergne	4
819	French	102787.49	69005.098	2.108	2.538	Charles Pic	1
820	British	104689.91	96820.775	2.545	2.400	Max Chilton	1
821	Mexican	102193.61	89478.060	2.357	0.203	Esteban Gutiérrez	4
822	Finnish	97318.85	79382.386	1.888	-1.105	Valtteri Bottas	4
823	Dutch	103390.91	48598.546	2.882	1.316	Giedo van der Garde	1
824	French	104306.19	99987.971	2.281	1.706	Jules Bianchi	1
825	Danish	96987.92	68862.146	1.919	-0.640	Kevin Magnussen	4
826	Russian	98504.52	73530.102	2.048	-0.304	Daniil Kvyat	4
828	Swedish	98732.87	61955.821	2.023	1.918	Marcus Ericsson	1
829	British	101773.12	15815.357	2.125	0.158	Will Stevens	4
830	Dutch	95693.99	73438.304	1.969	-0.689	Max Verstappen	4
831	Brazilian	101964.86	66908.304	2.256	1.750	Felipe Nasr	1
832	Spanish	96419.34	79302.398	1.817	-0.692	Carlos Sainz	4
833	Spanish	100870.10	15563.805	1.923	2.538	Roberto Merhi	1
834	American	102816.82	19231.380	2.200	2.400	Alexander Rossi	1
835	British	99547.57	41871.868	2.100	1.054	Jolyon Palmer	1
836	German	99908.99	71085.945	2.167	1.256	Pascal Wehrlein	1
837	Indonesian	94688.59	16819.292	2.273	1.167	Rio Haryanto	1
838	Belgian	96398.25	34325.612	1.718	2.000	Stoffel Vandoorne	3
839	French	97029.14	81019.450	1.779	-0.019	Esteban Ocon	4
840	Canadian	95922.82	77916.567	1.886	0.786	Lance Stroll	4

841	Italian	94894.40	78777.565	1.862	0.516	Antonio Giovinazzi	4
843	New Zealander	95627.92	17775.918	1.700	0.240	Brendon Hartley	3
844	Monegasque	95584.54	75469.055	1.750	-1.114	Charles Leclerc	4
845	Russian	93758.18	16442.890	1.632	0.762	Sergey Sirotkin	3
846	British	96429.63	91409.697	1.903	-0.594	Lando Norris	4
847	British	95362.81	75394.776	1.919	-0.617	George Russell	4
848	Thai	94918.97	65824.139	1.876	-0.495	Alexander Albon	4
849	Canadian	97352.13	96485.952	2.089	0.820	Nicholas Latifi	4
850	Brazilian	80792.57	24610.746	3.000	0.500	Pietro Fittipaldi	2
851	British	63319.40	10585.356	3.000	1.000	Jack Aitken	2
852	Japanese	98711.04	107669.930	2.099	-1.389	Yuki Tsunoda	4
853	Russian	99968.56	118356.438	2.211	0.636	Nikita Mazepin	4
854	German	98270.78	102623.020	2.051	-0.318	Mick Schumacher	4
855	Chinese	96468.96	77237.621	2.109	0.368	Guanyu Zhou	4
856	Dutch	92335.61	49414.467	1.900	-1.273	Nyck de Vries	4
857	Australian	96402.15	100168.453	1.953	-0.457	Oscar Piastri	4
858	American	97034.01	85572.570	2.182	0.056	Logan Sargeant	4
859	New Zealander	100862.63	111009.415	2.636	0.545	Liam Lawson	1
860	British	107227.07	110427.324	1.333	2.333	Oliver Bearman	3
861	Argentinian	96557.14	12142.161	1.250	0.222	Franco Colapinto	3
862	Australian	91576.25	5119.092	1.000	2.000	Jack Doohan	3

Interpreting F1 Driver Archetypes from Clustering and Silhouette Scores

The K-means clustering revealed five distinct groups of Formula 1 drivers based on their average lap time, lap time consistency, pit stop frequency, and positions gained per race. Below is a breakdown of each cluster, blending both performance characteristics and silhouette-based clustering quality:

Cluster 1: Efficient Overtakers Size: 14 drivers

Avg. Silhouette Width: 0.27

Traits:

Solid average lap times.

Slightly above-average pit stop usage.

Consistently gain positions during races, reflecting assertive on-track movement.

Interpretation: These drivers are effective at overtaking and making progress during races. However, the lower silhouette score suggests they share similarities with other clusters, likely due to mixed strategies or team dynamics, resulting in less clean separation.

Cluster 2: Short Run Strategists Size: 2 drivers

Avg. Silhouette Width: 0.47 (highest)

Traits:

Exceptionally fast lap times, the quickest among all clusters.

High pit stop frequency, possibly from sprint races or partial data.

Interpretation: Despite the small size, this cluster shows strong cohesion and clear distinction from others. These drivers are likely outliers or specialists with high performance in short runs. The high silhouette score confirms they are well-separated and internally consistent.

Cluster 3: Mid-Pack Movers Size: 6 drivers

Avg. Silhouette Width: 0.22 (lowest)

Traits:

Low pit stop counts.

Tight lap time consistency and positive position gains.

Interpretation: These drivers are strategic and efficient, making progress during races while minimizing variability. However, the low silhouette score indicates that they overlap with other driver types, possibly because of their balanced traits that straddle both aggressive and conservative styles.

Cluster 4: Baseline Finishers Size: 39 drivers (largest group)

Avg. Silhouette Width: 0.45

Traits:

Slightly negative positions gained → drivers tend to lose positions during races.

Lap times and pit stop behavior suggest average, consistent performance.

Interpretation: This is the most cohesive and well-defined cluster, forming the core baseline of the F1 driver pool. The high silhouette score validates this group as a well-separated, stable archetype—likely made up of regular midfield or slightly underperforming drivers.

Cluster 5: Slower but Stable Size: 15 drivers

Avg. Silhouette Width: 0.32

Traits:

Slowest average lap times and highest lap time variability.

Despite this, pit stop frequency remains moderate.

Interpretation: These drivers may represent rookies, backmarkers, or technically constrained cars. They are somewhat consistent in strategy but underperform on pace. The moderate silhouette score suggests internal cohesion with some overlap with neighboring clusters.

Research Question 3: Impact of Grid Position on Race Outcome

In Formula 1, the starting grid position (result of qualifying) is often considered crucial to race success. We want to statistically test whether starting in the Top 5 on the grid significantly increases the chances of winning compared to starting from a Lower Position.

Data Preprocessing

```
# Merge datasets by raceId to include year
data <- merge(results, races[, c("raceId", "year")], by = "raceId")

data_clean <- data %>%
  filter(!is.na(grid), !is.na(positionOrder), positionOrder > 0) %>%
  mutate(
```

```
    # Create binary variable for starting grid group
    GridGroup = ifelse(grid <= 5, "Top 5", "Lower"),
    # Create binary variable for race outcome
    Win = ifelse(positionOrder == 1, "Win", "Not Win"),
    # Podium finish: position 1 to 3
    PodiumFinish = ifelse(positionOrder <= 3, "Yes", "No"),
    # Front row: grid 1 or 2
    FrontRowStart = ifelse(grid <= 2, "Yes", "No"),
    # Midfield: grid 6-10
    MidfieldStart = ifelse(grid >= 6 & grid <= 10, "Yes", "No")
  )
```

Contingency Table

```
# This table compares Top 5 vs. Lower starters across Win/Not Win
table_result <- table(data_clean$GridGroup, data_clean$Win)
print(table_result)
```

```
##
##         Not Win  Win
##   Lower    4446   21
##   Top 5    1248  265
```

Chi-Square Test

```
chisq_test <- chisq.test(table_result)

# Print expected counts for assumption check
chisq_test$expected
```

```
##
##          Not Win       Win
##   Lower 4253.361 213.63913
##   Top 5 1440.639  72.36087
```

```
chisq_test <- chisq.test(table_result)
print(chisq_test)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_result
## X-squared = 717.29, df = 1, p-value < 2.2e-16
```

```
expected <- chisq_test$expected

# Cramér's V to measure effect size
```

```r
n <- sum(table_result)
k <- min(nrow(table_result), ncol(table_result))
cramers_v <- sqrt(chisq_test$statistic / (n * (k - 1)))
print(cramers_v)
```

```
## X-squared
## 0.3463361
```

Win Rate & Confidence Interval

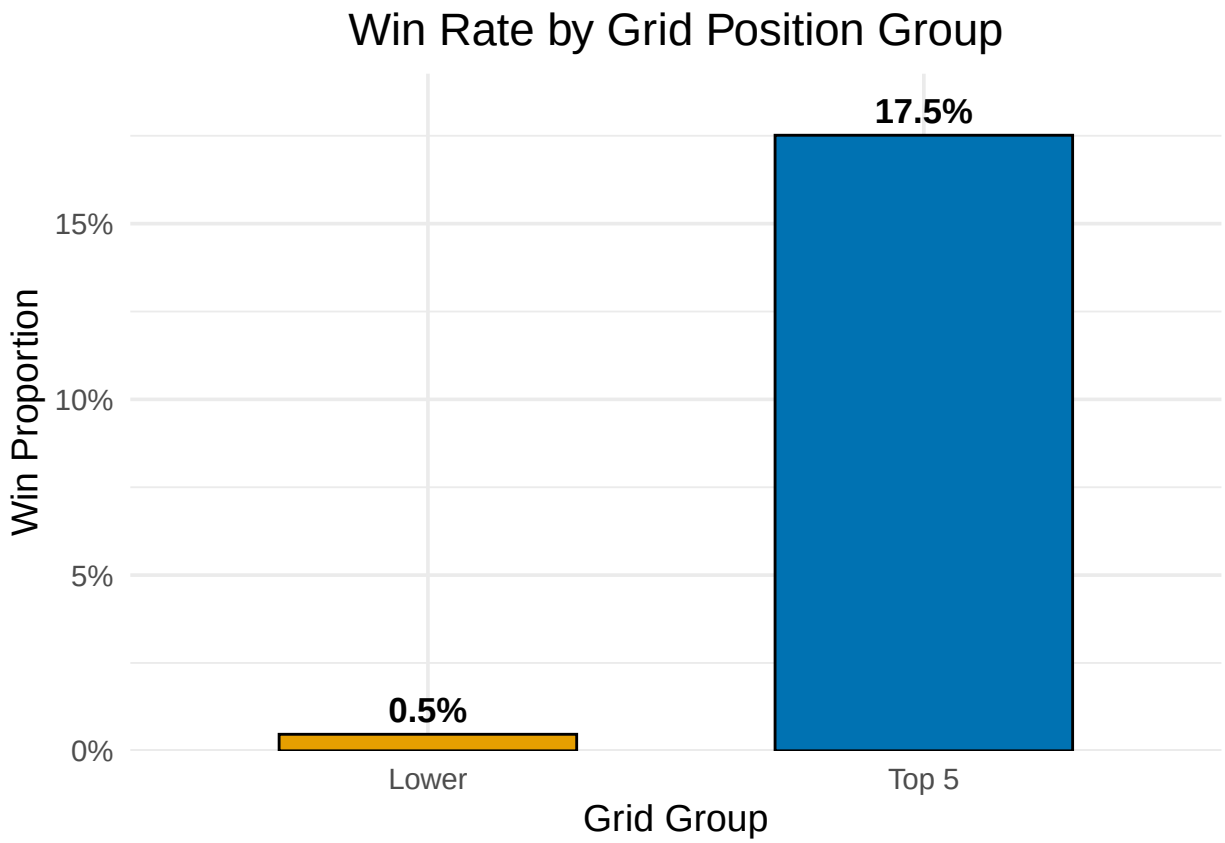
```r
# Calculate and compare Wilson confidence intervals for win rates of Top 5 vs Lower grid starters

top5 <- data_clean %>% filter(GridGroup == "Top 5")
lower <- data_clean %>% filter(GridGroup == "Lower")

top5_win_rate <- binom.confint(sum(top5$Win == "Win"), nrow(top5), method = "wilson")
lower_win_rate <- binom.confint(sum(lower$Win == "Win"), nrow(lower), method = "wilson")

print(top5_win_rate)
```

```
##   method   x    n      mean     lower     upper
## 1 wilson 265 1513 0.1751487 0.1568257 0.1951171
```

```r
print(lower_win_rate)
```

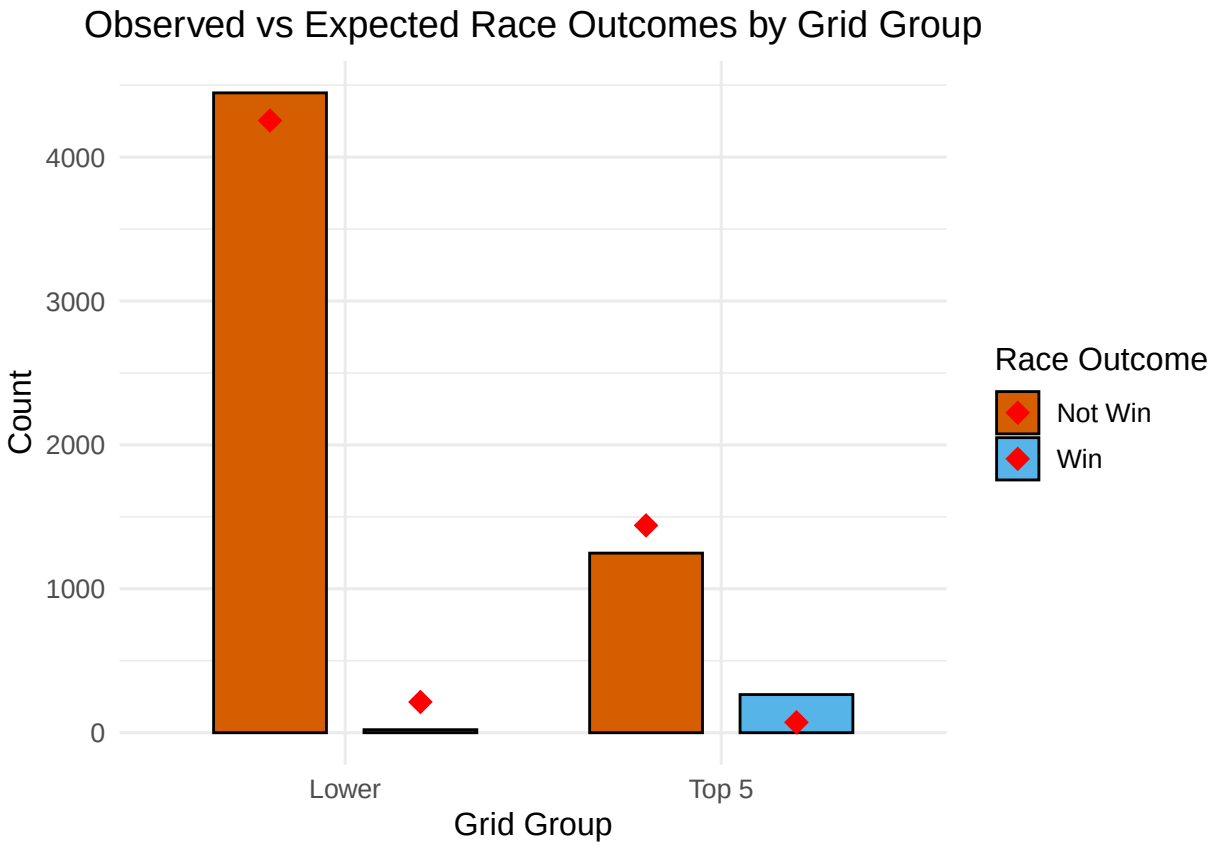
```
##   method  x    n        mean       lower       upper
## 1 wilson 21 4467 0.004701142 0.003076968 0.007176462
```

```r
# Compute win rates by grid group and visualize the comparison using a bar plot

win_summary <- data_clean %>%
  group_by(GridGroup) %>%
  summarise(WinRate = mean(Win == "Win"))

ggplot(win_summary, aes(x = GridGroup, y = WinRate, fill = GridGroup)) +
  geom_bar(stat = "identity", width = 0.6, color = "black") +
  geom_text(aes(label = scales::percent(WinRate, accuracy = 0.1)),
            vjust = -0.5, size = 4.5, fontface = "bold") +
  labs(title = "Win Rate by Grid Position Group",
       y = "Win Proportion",
       x = "Grid Group") +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1),
                     expand = expansion(mult = c(0, 0.1))) +
  scale_fill_manual(values = c("Top 5" = "#0072B2", "Lower" = "#E69F00")) +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text(),
        legend.position = "none")
```



Observed vs Expected

```
# Visualize observed vs. expected race outcomes by grid group
# using a grouped bar plot with expected values overlaid

observed <- as.data.frame(table_result)
expected_df <- as.data.frame(as.table(expected))
colnames(observed) <- c("GridGroup", "Win", "Observed")
colnames(expected_df) <- c("GridGroup", "Win", "Expected")
obs_exp <- merge(observed, expected_df, by = c("GridGroup", "Win"))

ggplot(obs_exp, aes(x = GridGroup, y = Observed, fill = Win)) +
  geom_bar(stat = "identity", position = position_dodge(0.8),
           width = 0.6, color = "black") +
  geom_point(aes(y = Expected), color = "red", shape = 18, size = 4,
             position = position_dodge(width = 0.8)) +
  labs(title = "Observed vs Expected Race Outcomes by Grid Group",
       x = "Grid Group", y = "Count", fill = "Race Outcome") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title = element_text()) +
  scale_fill_manual(values = c("Win" = "#56B4E9", "Not Win" = "#D55E00"))
```



In the "Top 5" grid group:

- The observed number of wins** is significantly higher than expected under independence.
- The red point (expected value) is much lower than the bar (actual count).

In the "Lower" grid group:

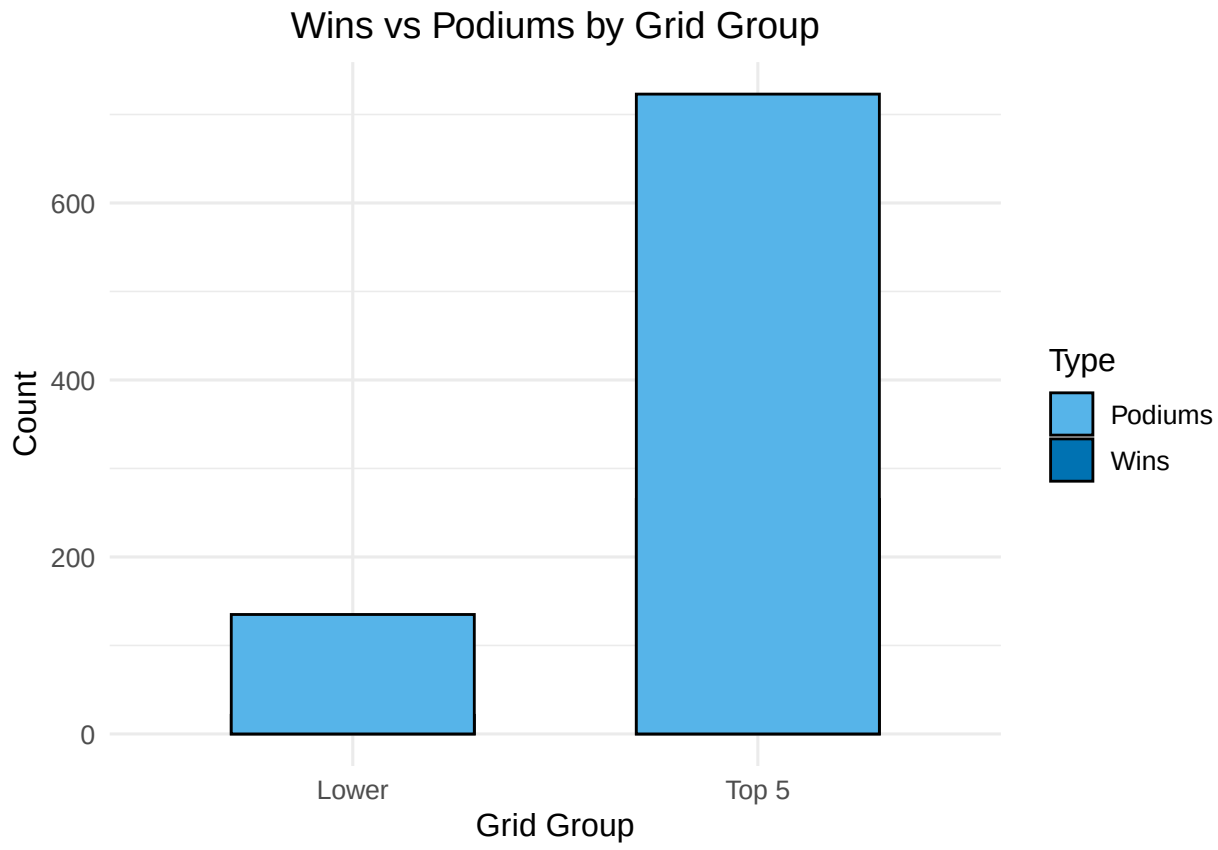
- The observed number of wins is much lower than expected.
- The red point (expected count) overshoots the bar, indicating fewer wins than predicted.

Podiums vs Wins

```
# Summarize key race statistics by grid group

summary_df <- data_clean %>%
  group_by(GridGroup) %>%
  summarise(
    Total = n(),
    Wins = sum(Win == "Win"),
    Podiums = sum(PodiumFinish == "Yes"),
    FrontRowStarts = sum(FrontRowStart == "Yes"),
    MidfieldStarts = sum(MidfieldStart == "Yes")
  )
print(summary_df)
```

```
## # A tibble: 2 x 6
##   GridGroup Total  Wins Podiums FrontRowStarts MidfieldStarts
##   <chr>     <int> <int>   <int>          <int>          <int>
## 1 Lower      4467    21     135              0           1426
## 2 Top 5      1513   265     723            655              0
```

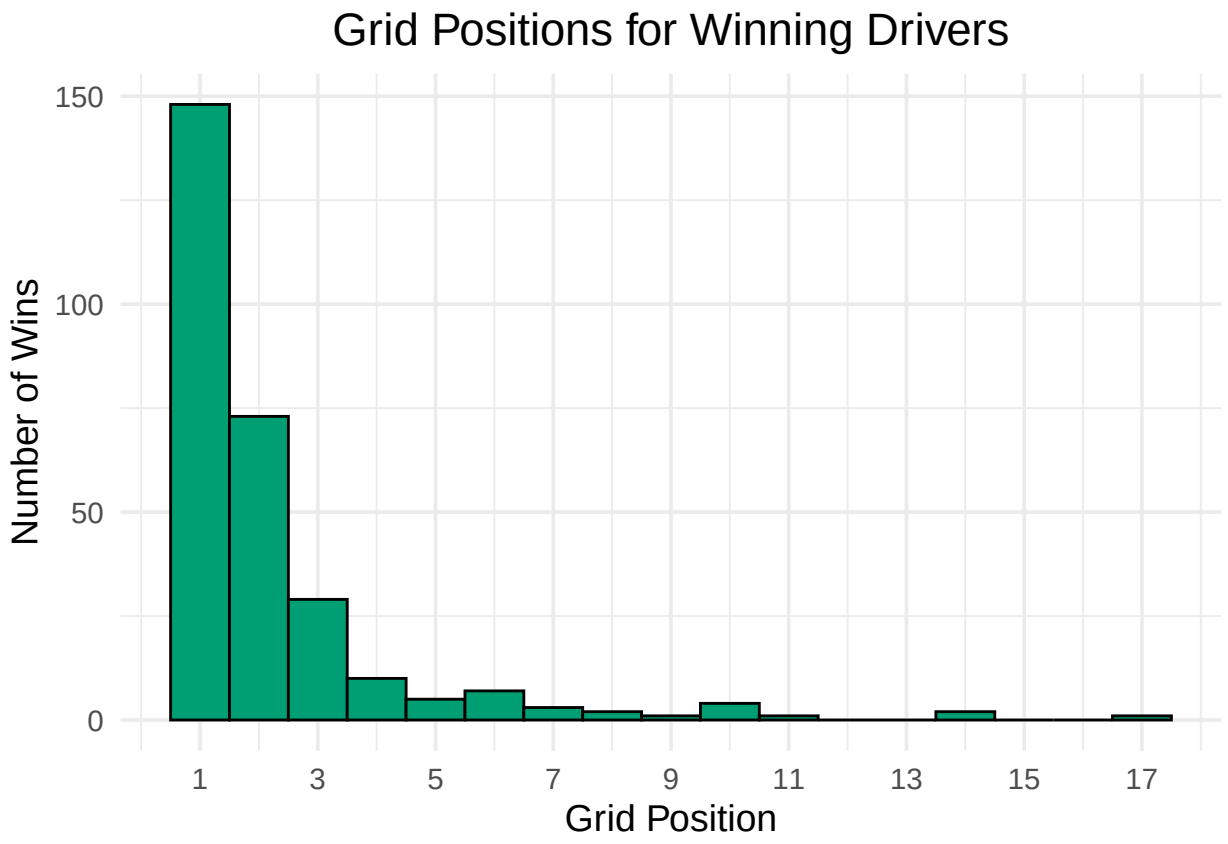
```
# Visualize comparison of Wins vs Podiums using grouped bar plot

ggplot(summary_df, aes(x = GridGroup)) +
  geom_bar(aes(y = Wins, fill = "Wins"), stat = "identity",
           position = position_dodge(width = 0.7),
           color = "black", width = 0.6) +
  geom_bar(aes(y = Podiums, fill = "Podiums"), stat = "identity",
           position = position_dodge(width = 0.7),
           color = "black", width = 0.6) +
  scale_fill_manual(name = "Type", values = c("Wins" = "#0072B2", "Podiums" = "#56B4E9")) +
  labs(title = "Wins vs Podiums by Grid Group", y = "Count", x = "Grid Group") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(hjust = 0.5))
```



Grid Position Distribution of Winners

```
# Filter winning drivers and plot a histogram of their starting grid positions
```

```r
winner_grid <- data_clean %>% filter(Win == "Win")

ggplot(winner_grid, aes(x = grid)) +
  geom_histogram(binwidth = 1, fill = "#009E73", color = "black") +
  labs(title = "Grid Positions for Winning Drivers",
       x = "Grid Position", y = "Number of Wins") +
  scale_x_continuous(breaks = seq(1, max(winner_grid$grid), by = 2)) +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5))
```



Final Interpretation:

1. The Chi-square test assesses whether there is a significant association between grid group and win outcome. Cramér's V 0.34, indicating a moderate effect size. Together, these results confirm that starting position has a meaningful impact on race outcomes in Formula 1.

2. The calculation uses the Wilson method to estimate 95% confidence intervals for win rates.

   Top 5 starters have an average win rate of 17.5% with a 95% CI of [15.68%, 19.51%].

   Lower grid starters have a win rate of 0.47% with a 95% CI of [0.30%, 0.71%].

3. Drivers starting in the top postitions have an average win rate of 17.5%, while those starting in Lower positions have a dramatically lower win rate of just 0.47%.

   This means that drivers in the Top 5 are more than 22 times more likely to win than those starting from P6 or lower.