# Final Project

Aniket Rattan

2024-11-08

## 1. Introduction

The aim of this analysis is to explore and understand the patterns in video game sales data, with a focus on the relationships between sales, user and critic ratings, and other game features like genre and platform. The dataset includes information about video games, their sales across various regions, and ratings provided by both critics and users.

### 1.1 Data Description

The dataset contains the following variables:

```
# Display the dataset
variable_description_table %>%
    kable(format = "latex", booktabs = TRUE, align = "l") %>%
    kable_styling(latex_options = c("striped", "hold_position", "scale_down"), font_size = 12)
```

| Variable | Description |
|---|---|
| Name | The title of the video game |
| Platform | The gaming platform (e.g., PlayStation, Xbox, Nintendo) |
| Year | The year when the game was released |
| Genre | The type of game (e.g., Action, Adventure) |
| Publisher | The company that published the game |
| NA_Sales | The sales of the game in North America (in millions of units) |
| EU_Sales | The sales of the game in Europe (in millions of units) |
| JP_Sales | The sales of the game in Japan (in millions of units) |
| Other_Sales | The sales of the game in other regions (in millions of units) |
| Global_Sales | The total sales of the game across the world (in millions of units) |
| User_Score | The score given to the game by users |
| User_Count | The number of users who gave User_Score |
| Rating | The rating assigned to the game by the ESRB (E : Everyone, E10+ : Everyone 10+, M : Mature, T : Teens) |

## 2. Data Overview and Cleaning

Before we dive into any analysis, it's important to get a good look at the data and clean it up. This initial step is crucial—messy data can lead to messy results, so we want to make sure everything is accurate and organized. Here, we'll load the data, take an initial look to understand its structure, and clean it up so it's ready for analysis. By the end of this section, we'll have a clean, reliable dataset that we can use confidently for exploring trends and building models.

### 2.1 Load Data and Initial Inspection

Let's start by loading the dataset and taking a quick look at what we're working with. Here, we'll check out some basic details about the variables, spot any obvious issues like missing or duplicated values, and get a feel for the overall shape of the data. This initial inspection will set the stage for a smooth cleaning process.

```
# Adjusting width so that the code stays within margins
options(width = 100)

# Load the video game sales data
video_games <- read.csv(here("video_games_sales.csv"))

# Display a summary of the data to get an overview of each variable
summary(video_games)
```

```
##      Name              Platform             Year             Genre             Publisher
##  Length:7501        Length:7501        Length:7501        Length:7501        Length:7501
##  Class :character   Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##     NA_Sales          EU_Sales          JP_Sales        Other_Sales        Global_Sales
##  Min.   : 0.0000   Min.   : 0.0000   Min.   :0.00000   Min.   : 0.00000   Min.   : 0.0100
##  1st Qu.: 0.0600   1st Qu.: 0.0200   1st Qu.:0.00000   1st Qu.: 0.01000   1st Qu.: 0.1100
##  Median : 0.1500   Median : 0.0600   Median :0.00000   Median : 0.02000   Median : 0.2800
##  Mean   : 0.3795   Mean   : 0.2265   Mean   :0.05927   Mean   : 0.08015   Mean   : 0.7456
##  3rd Qu.: 0.3800   3rd Qu.: 0.2000   3rd Qu.:0.01000   3rd Qu.: 0.07000   3rd Qu.: 0.7200
##  Max.   :41.3600   Max.   :28.9600   Max.   :6.50000   Max.   :10.57000   Max.   :82.5300
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3          NA's   :3
##    User_Score      User_Count         Rating
##  Min.   :0.000   Min.   :    4.0   Length:7501
##  1st Qu.:6.400   1st Qu.:   10.0   Class :character
##  Median :7.500   Median :   24.0   Mode  :character
##  Mean   :7.127   Mean   :  162.9
##  3rd Qu.:8.200   3rd Qu.:   81.0
##  Max.   :9.700   Max.   :10665.0
##  NA's   :3       NA's   :3
```

```
# Display the first few rows of the data in a table format for a quick
# inspection
head(video_games) %>%
    kable(format = "latex") %>%
    kable_styling(latex_options = c("scale_down", "HOLD_position"))
```

| Name | Platform | Year | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales | User_Score | User_Count | Rating |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Wii Sports | Wii | 2006 | Sports | Nintendo | 41.36 | 28.96 | 3.77 | 8.45 | 82.53 | 8.0 | 322 | E |
| Mario Kart Wii | Wii | 2008 | Racing | Nintendo | 15.68 | 12.76 | 3.79 | 3.29 | 35.52 | 8.3 | 709 | E |
| Grand Theft Auto V | PS3 | 2013 | Action | Take-Two Interactive | 7.02 | 9.09 | 0.98 | 3.96 | 21.04 | 8.2 | 3994 | M |
| Grand Theft Auto V | X360 | 2013 | Action | Take-Two Interactive | 9.66 | 5.14 | 0.06 | 1.41 | 16.27 | 8.1 | 3711 | M |
| Wii Sports Resort | Wii | 2009 | Sports | Nintendo | 15.61 | 10.93 | 3.28 | 2.95 | 32.77 | 8.0 | 192 | E |
| New Super Mario Bros. | DS | 2006 | Platform | Nintendo | 11.28 | 9.14 | 6.50 | 2.88 | 29.80 | 8.5 | 431 | E |

### 2.2 Cleaning the Data

From the head() function, we can see that Grand Theft Auto V appeared two times. This is because many games have duplicate entries due to being launched on various platforms. Thus, we will convert the data from long format to wide format based on their platform.

```
# Convert platform indicators from long to wide format, creating binary columns for each platform
video_games_clean <- video_games %>%
  group_by(Name) %>%
  mutate(
    DS = any(Platform == "DS"), PC = any(Platform == "PC"), PS2 = any(Platform == "PS2"),
    PS3 = any(Platform == "PS3"), XB = any(Platform == "XB"), PSP = any(Platform == "PSP"),
    X360 = any(Platform == "X360"))

# Summarize the data by game name, aggregating values across platforms for each game
video_games_clean <- video_games_clean %>%
  group_by(Name) %>%
  summarise(
    Year = min(Year, na.rm = TRUE), # Take the earliest release year
    Genre = first(Genre),           # Retain the first genre
    Publisher = first(Publisher),   # Retain the first publisher
    NA_Sales = sum(NA_Sales, na.rm = TRUE),
    EU_Sales = sum(EU_Sales, na.rm = TRUE),
    JP_Sales = sum(JP_Sales, na.rm = TRUE),
    Other_Sales = sum(Other_Sales, na.rm = TRUE),
    Global_Sales = sum(Global_Sales, na.rm = TRUE),
    User_Score = mean(User_Score, na.rm = TRUE),    # Average user score
    User_Count = sum(User_Count, na.rm = TRUE),     # Total user count
    Rating = first(Rating),        # First rating occurrence
    across(DS:X360, any)           # Combine platform indicators
  ) %>%
  filter(DS | PC | PS2 | PS3 | XB | PSP | X360)
```

We can see from the summary that some of the variables are coded incorrectly. So we will recode those.

```
# Adjusting width so that the code stays within margins
options(width = 100)

# Convert `Year` to integer type
video_games_clean$Year <- suppressWarnings(as.integer(video_games_clean$Year))

# Converting categorical variables into factors
video_games_clean$Genre <- as.factor(video_games_clean$Genre)
video_games_clean$Publisher <- as.factor(video_games_clean$Publisher)
video_games_clean$Rating <- as.factor(video_games_clean$Rating)

# Lastly, removing any rows with NA values
video_games_clean <- na.omit(video_games_clean)

# Summary of the cleaned dataset
summary(video_games_clean)
```

```
##      Name                Year                Genre                          Publisher
```

```
##  Length:3486        Min.   :1985   Action      :673   Electronic Arts            : 364
##  Class :character   1st Qu.:2004   Role-Playing:448   Sony Computer Entertainment: 230
##  Mode  :character   Median :2007   Shooter     :439   Ubisoft                    : 230
##                     Mean   :2007   Sports      :437   Activision                 : 198
##                     3rd Qu.:2009   Racing      :299   THQ                        : 153
##                     Max.   :2016   Simulation  :216   Sega                       : 141
##                                    (Other)     :974   (Other)                    :2170
##     NA_Sales          EU_Sales          JP_Sales         Other_Sales       Global_Sales
##  Min.   : 0.0000   Min.   : 0.000   Min.   :0.00000   Min.   : 0.0000   Min.   : 0.010
##  1st Qu.: 0.0700   1st Qu.: 0.020   1st Qu.:0.00000   1st Qu.: 0.0100   1st Qu.: 0.130
##  Median : 0.2000   Median : 0.080   Median :0.00000   Median : 0.0300   Median : 0.410
##  Mean   : 0.6489   Mean   : 0.389   Mean   :0.07734   Mean   : 0.1474   Mean   : 1.263
##  3rd Qu.: 0.6600   3rd Qu.: 0.330   3rd Qu.:0.03000   3rd Qu.: 0.1200   3rd Qu.: 1.240
##  Max.   :23.8400   Max.   :23.420   Max.   :6.50000   Max.   :10.7100   Max.   :56.570
##
##    User_Score      User_Count       Rating         DS              PC              PS2
##  Min.   :0.000   Min.   :    4.0   E   :1006   Mode :logical   Mode :logical   Mode :logical
##  1st Qu.:6.581   1st Qu.:   14.0   E10+: 413   FALSE:2973      FALSE:2768      FALSE:2256
##  Median :7.500   Median :   37.0   M   : 723   TRUE :513       TRUE :718       TRUE :1230
##  Mean   :7.211   Mean   :  305.4   T   :1344
##  3rd Qu.:8.200   3rd Qu.:  128.0
##  Max.   :9.700   Max.   :24807.0
##
##     PS3             XB              PSP             X360
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:2629      FALSE:2906      FALSE:3065      FALSE:2537
##  TRUE :857       TRUE :580       TRUE :421       TRUE :949
##
##
##
##
```

## 3. Univariate analysis

Before we start with the analysis, we need to use log transformation on Global_Sales and User_Count. These variables are highly skewed because:

Most of the games have sales $< 1$ million units. This creates a highly right skewed graph.

Most of the games have User_Count of 300-400 but a few games have User_Count as high as 24000.

Thus we will use log transformation for this discrepancy. We will also use a histogram to check if the log transformation helped or not.

```r
# Add log transformation for global sales to address skewness
video_games_clean$Log_Global_Sales <- log(video_games_clean$Global_Sales + 1)

# Visualize the distribution before and after log transformation
par(mfrow = c(1, 2))

# First histogram
hist(video_games_clean$Global_Sales,
     main = "Global Sales", col = "skyblue", border = "white", breaks = 25,
     xlab = "Global Sales (in millions)")
```

```r
# Second histogram
hist(video_games_clean$Log_Global_Sales,
     main = "Log-Transformed Global Sales", col = "darkseagreen3", border = "white", breaks = 25,
     xlab = "Log of Global Sales")
```

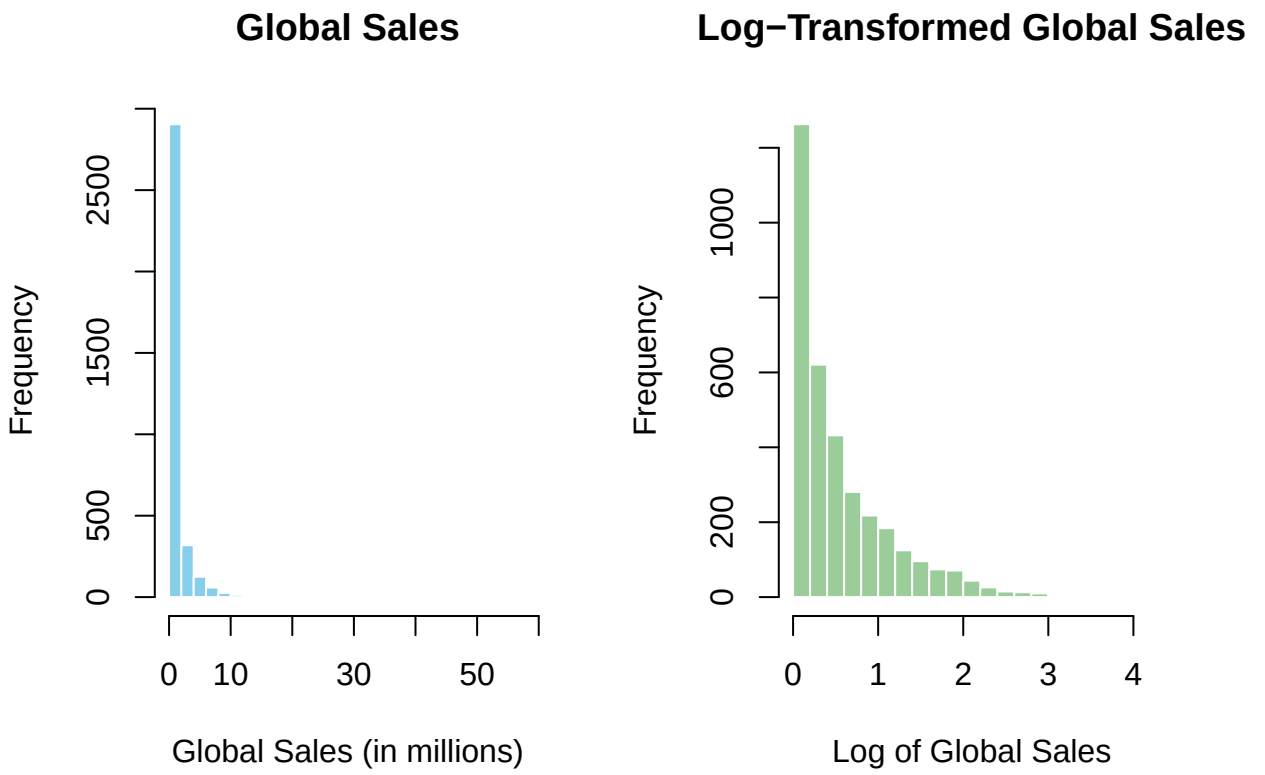


Figure 1: Distribution of Global Sales Before and After Log Transformation – The histogram on the left shows the original distribution of global sales, which is heavily right-skewed, with most games having sales under 1 million units and a few outliers with very high sales. After applying a log transformation, as seen in the histogram on the right, the data distribution becomes more normalized.

Looking at the histograms, the original global sales data is extremely skewed to the right—most games don't make it big, and only a handful pull in huge sales. This kind of skew can make modeling tricky because those few big sellers can throw things off. By applying a log transformation, we smooth out the distribution, making it more bell-shaped. This transformation helps our models perform better and gives us a clearer view of the typical game sales.

```r
# Add log transformation for global sales to address skewness
video_games_clean$Log_User_Count <- log(video_games_clean$User_Count)

# Visualize the distribution before and after log transformation
par(mfrow = c(1, 2))

# First histogram
hist(video_games_clean$User_Count,
     main = "User Count", col = "skyblue", border = "white", breaks = 20, xlab = "User Count")

# Second histogram
hist(video_games_clean$Log_User_Count,
```

```
    main = "Log-Transformed User Count", col = "darkseagreen3", border = "white", breaks = 20,
    xlab = "Log of User Count")
```

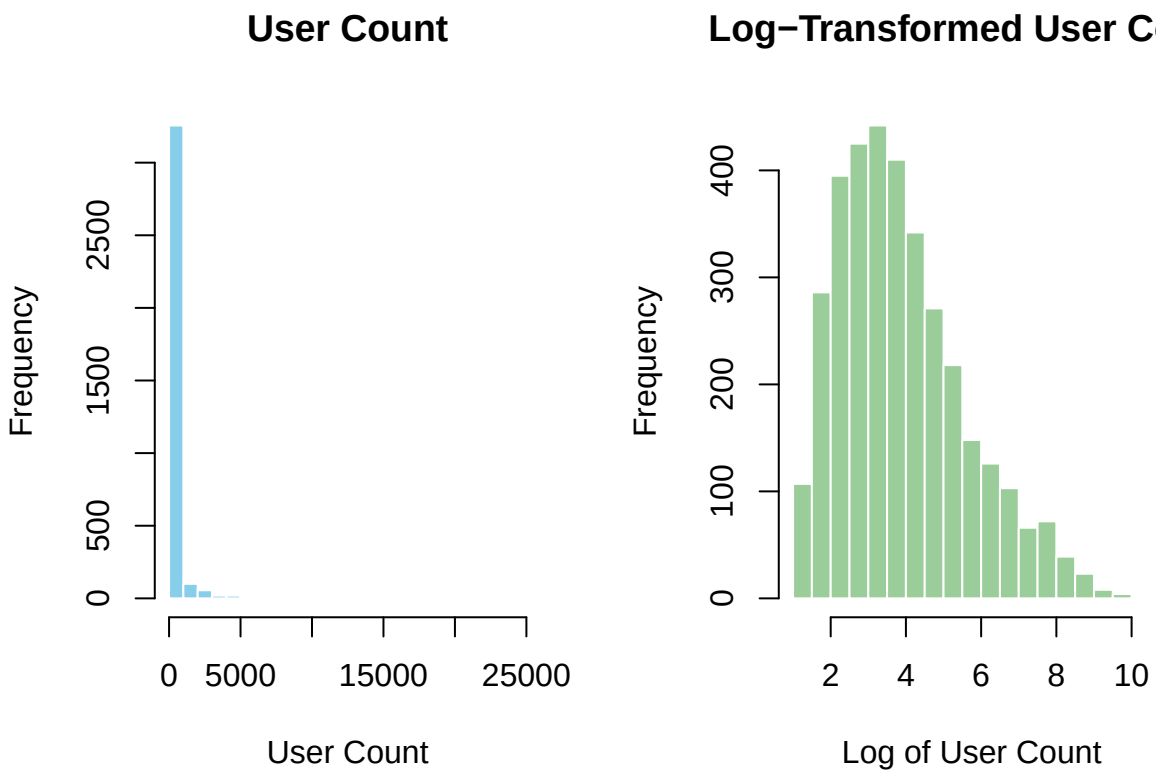


Figure 2: Distribution of User Count Before and After Log Transformation – In the left histogram, the original distribution of user count is highly skewed, with most games having low user feedback and a few games showing extremely high user counts. The right histogram shows the log-transformed data, which results in a more symmetrical distribution.

The user count data tells a similar story—most games get reviews from a relatively small group, but a select few attract a massive number of user ratings. This imbalance can distort our analysis, making it hard to spot patterns. By log-transforming the user counts, we reduce this skew, resulting in a more even spread. This adjustment makes it easier to compare games and ensures our models aren't overly influenced by the outliers.

Okay. So, now with that out of the way, let's start doing some analysis:

### 3.1

To kick off our analysis, let's see where most of the games are being released. Are certain platforms more popular than others for game launches? We'll dive into a bar plot to get a clearer picture of the number of games available on each platform.

```
# Adjusting width so that the code stays within margins
options(width = 65)

# Summarize the number of games for each platform
platform_counts <- video_games_clean %>%
  summarise(
    DS = sum(DS), PC = sum(PC), PS2 = sum(PS2), PS3 = sum(PS3),
```

```
    XB = sum(XB), PSP = sum(PSP), X360 = sum(X360)) %>%
  pivot_longer(cols = DS:X360, names_to = "Platform", values_to = "Count")

# Create a bar plot of the number of games by platform
ggplot(platform_counts, aes(x = Platform, y = Count, fill = Platform)) +
  geom_bar(stat = "identity", color = "black") +
  labs(title = "Number of Games by Platform", x = "Platform", y = "Number of Games") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(size = 14, face = "bold"),
        legend.position = "none")
```

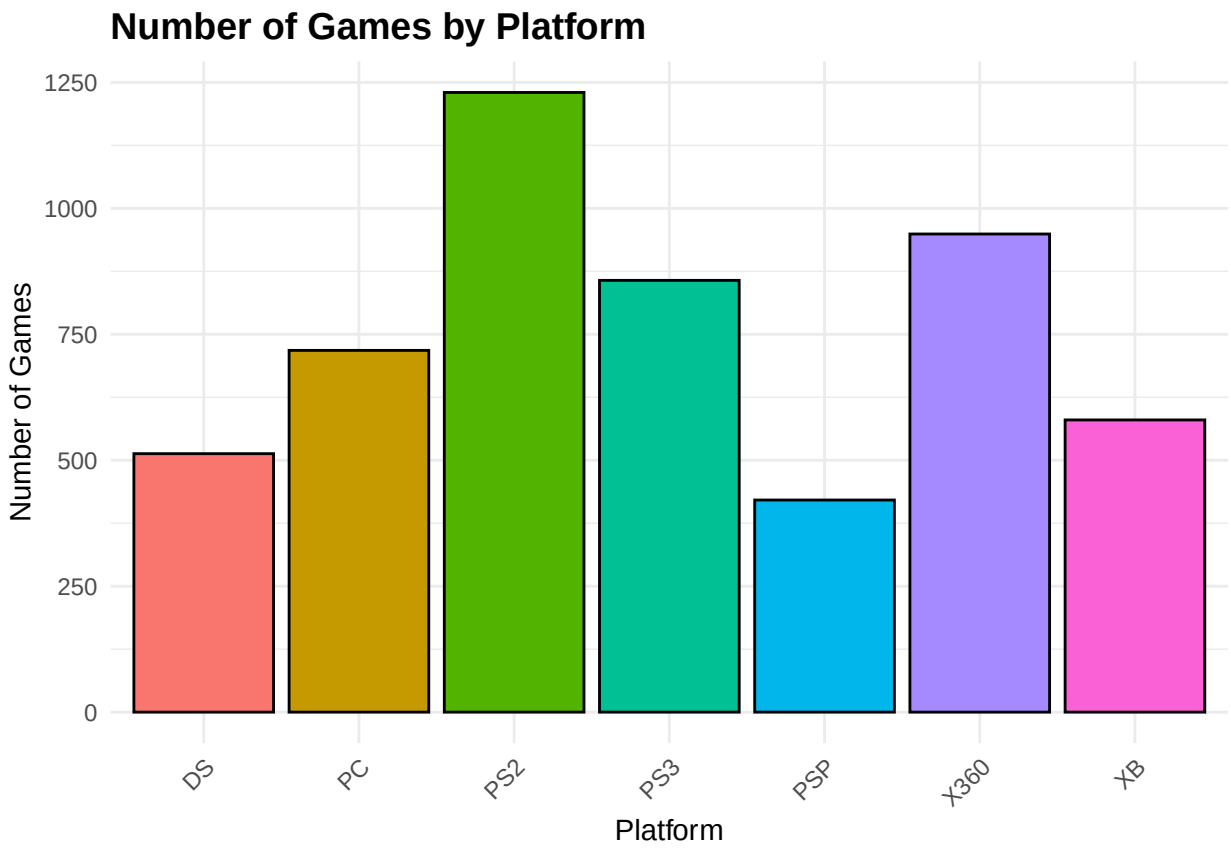


Figure 3.1: Count of Games per Platform – This bar chart highlights which gaming platforms have the most game releases.

From the chart, it's clear that platforms like PS2 and X360 have a larger game library compared to others, indicating their popularity among developers during their peak years.

### 3.2

Now, let's dig into what types of games are out there. From action-packed adventures to brain-teasing puzzles, different genres dominate the market. This bar plot gives us a sense of which genres have the most titles—do action games really rule the roost, or is there a hidden gem genre?

```
# Bar plot to show the distribution of games across different genres
ggplot(video_games_clean, aes(x = Genre)) +
```

```
  geom_bar(fill = "coral", color = "black") +
  labs(title = "Number of Games by Genre", x = "Genre", y = "Number of Games") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(size = 14, face = "bold"))
```

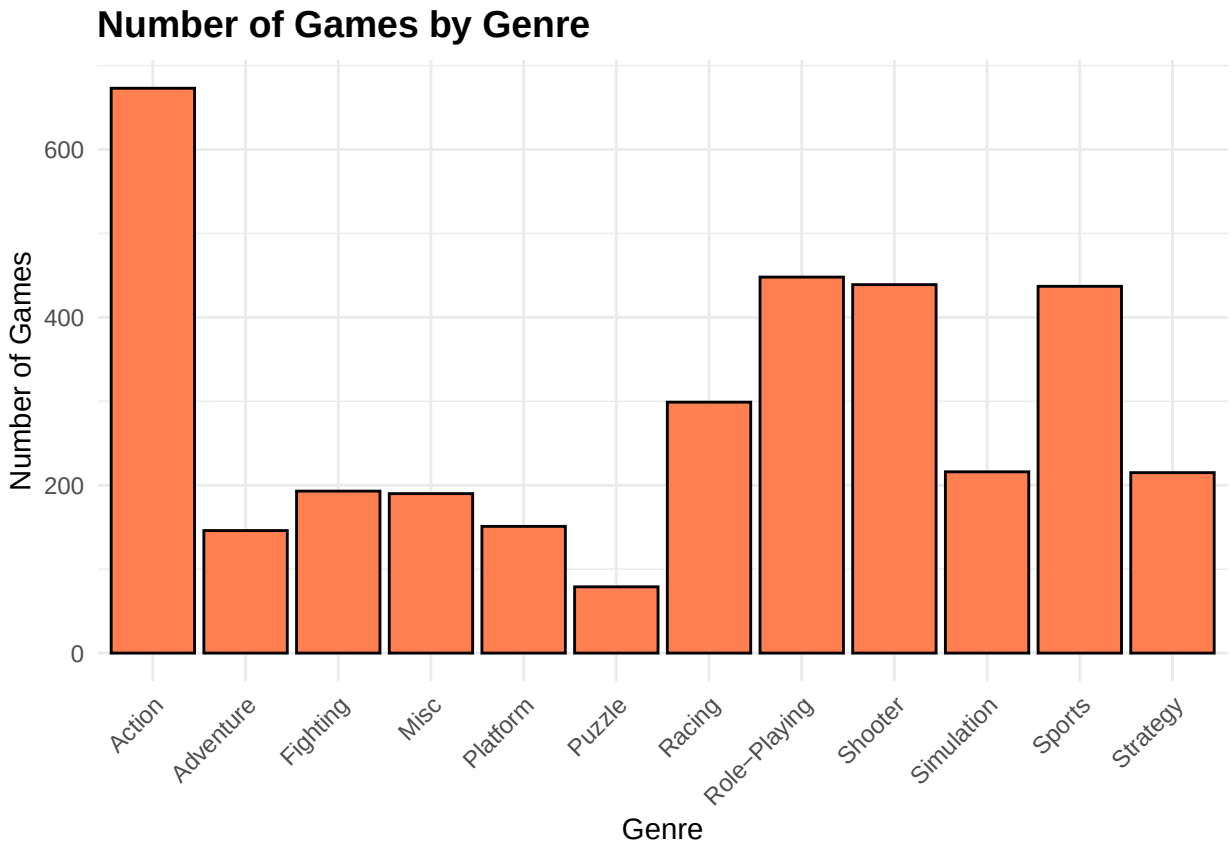


Figure 3.2: Game Genre Breakdown – A closer look at the number of games available in each genre.

The chart suggests that Action genres are the most prevalent, reflecting their wide appeal to a broad audience, while genres like Adventure and Puzzle are less common.

### 3.3

Finally, let's look at how games are rated in terms of age appropriateness. Are most games aimed at all ages, or do we see a lot more mature content? This bar chart helps us understand the age rating distribution of games in our dataset.

```
# Define rating labels with full forms
rating_labels <- c(E = "E (Everyone)", `E10+` = "E10+ (Everyone 10+)",
    M = "M (Mature 17+)", T = "T (Teen)")

# Plot for Number of Games by Rating with full forms
ggplot(video_games_clean, aes(x = Rating)) + geom_bar(fill = "steelblue",
    color = "black") + labs(title = "Number of Games by Rating", x = "Rating",
    y = "Number of Games") + scale_x_discrete(labels = rating_labels) +
    theme_minimal() + theme(axis.text.x = element_text(angle = 45,
    hjust = 1), plot.title = element_text(size = 14, face = "bold"))
```

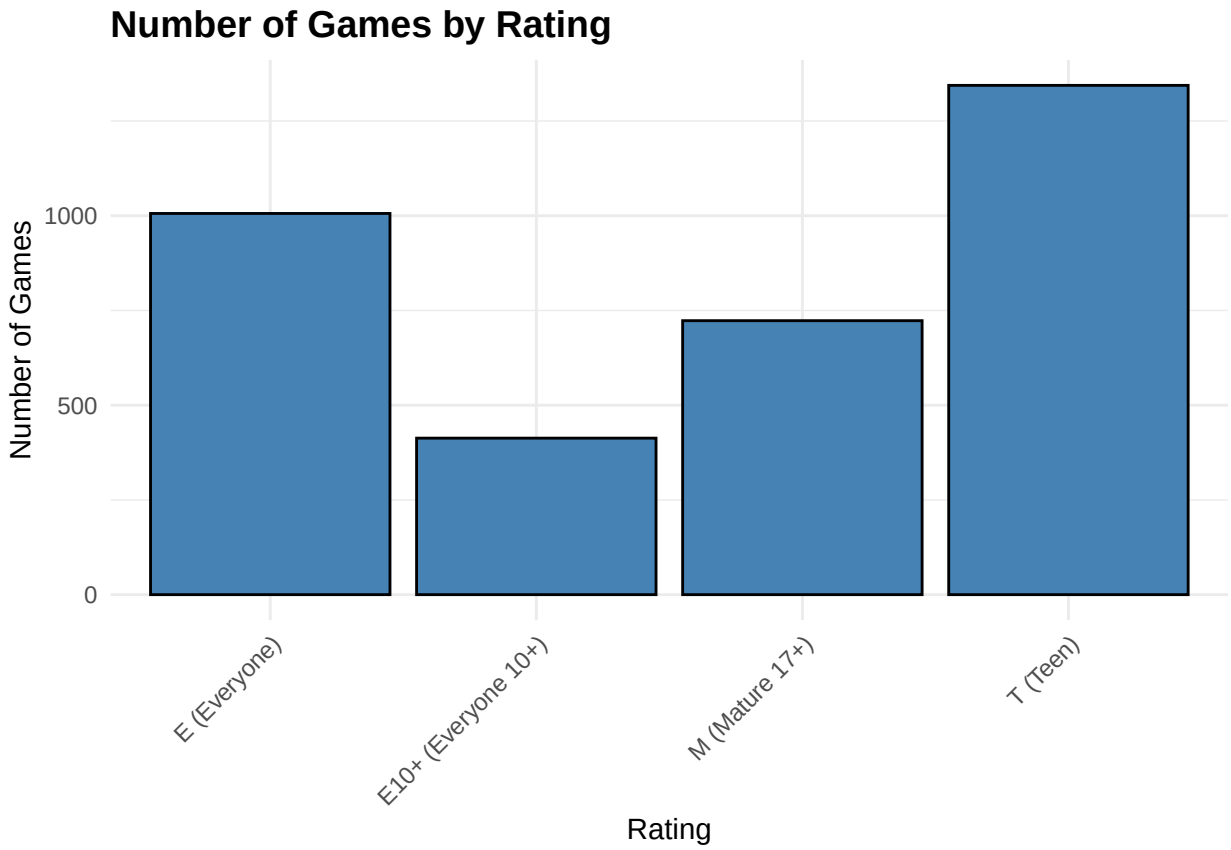


Figure 3.3: Age Rating Distribution – A view of how games stack up across different ESRB ratings.

The majority of games fall into the E and T categories, suggesting that game publishers often aim for broader audience appeal. However, the significant number of M-rated games shows that there's also a strong market for mature content.

# 4. Bivariate Analysis

Now that we've got a good sense of the data from our univariate analysis in Section 3, it's time to take it a step further. In this section, we'll start pairing variables to see how they interact with each other. By looking at relationships between variables like platform and sales, or genre and revenue, we can uncover more meaningful patterns. This deeper look will help us understand not just the "what," but also the "why" behind trends we observed earlier. Let's dive in and see how our findings from Section 3 play out when we start connecting the dots!

### 4.1

Now that we know which platforms host the most games, let's shift our focus to the revenue side of things. Do platforms with more games necessarily mean higher average sales? This bar plot shows the average global sales per platform to shed some light on this.

```
# Calculate the average global sales for each platform across
# all games
avg_sales_by_platform <- video_games_clean %>%
    summarise(DS = mean(Global_Sales[DS == 1], na.rm = TRUE), PC = mean(Global_Sales[PC ==
```

```
        1], na.rm = TRUE), PS2 = mean(Global_Sales[PS2 == 1], na.rm = TRUE),
        PS3 = mean(Global_Sales[PS3 == 1], na.rm = TRUE), XB = mean(Global_Sales[XB ==
            1], na.rm = TRUE), PSP = mean(Global_Sales[PSP == 1],
            na.rm = TRUE), X360 = mean(Global_Sales[X360 == 1], na.rm = TRUE)) %>%
    pivot_longer(cols = DS:X360, names_to = "Platform", values_to = "Average_Sales")

# Bar plot showing the average global sales by platform
ggplot(avg_sales_by_platform, aes(x = Platform, y = Average_Sales)) +
    geom_bar(stat = "identity", fill = viridis(1, option = "C", alpha = 0.7),
        color = "black") + labs(title = "Average Global Sales by Platform Presence",
    x = "Platform", y = "Average Global Sales (Millions)") + theme_classic() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1), plot.title = element_text(size = 14,
        face = "bold"), axis.line = element_blank())
```

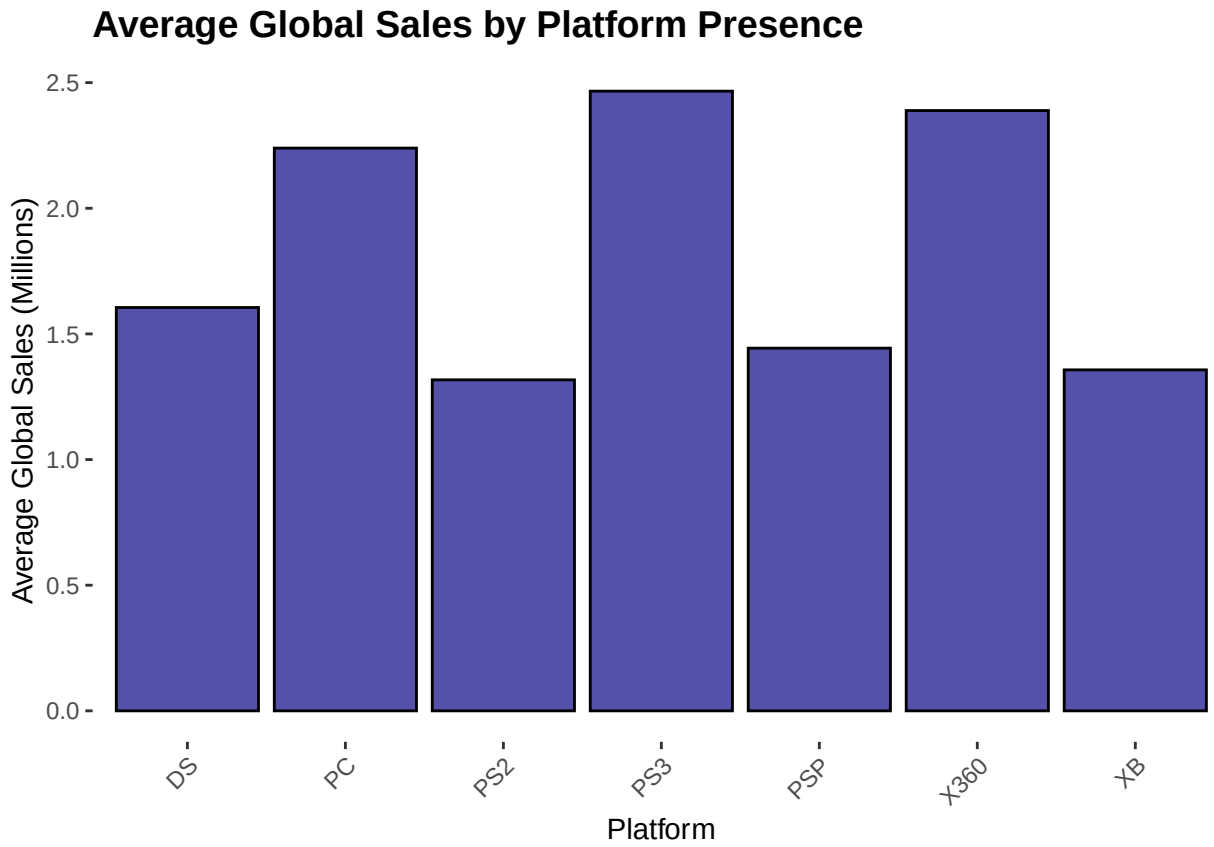


Figure 4.1: Average Global Sales by Platform – This bar chart displays the average global sales for games available on each platform.

Although the PS2 had the highest number of game releases (as seen in Figure 3.1), it's actually the PS3 that boasts higher average sales per game. This suggests that while PS2 had a broader library, the PS3 games might have been more popular or higher priced, leading to better average revenue. Also, X360 and PC platforms generate a considerate revenue as well.

### 4.2

Next, let's look at how different game genres perform when it comes to revenue. Are certain genres not just popular in terms of number of games but also in terms of sales? Here, a box plot for log-transformed global sales helps us understand the variability in revenue across genres.

```r
# Box plot to show the distribution of log-transformed global sales by genre
ggplot(video_games_clean, aes(x = Genre, y = Log_Global_Sales, fill = Genre)) +
  geom_boxplot() + coord_cartesian(ylim = c(0, 1.5)) +
  labs(
    title = "Log-Transformed Global Sales by Genre",
    x = "Genre",
    y = "Log of Global Sales (Millions)") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(size = 14, face = "bold"),
    legend.position = "none")
```

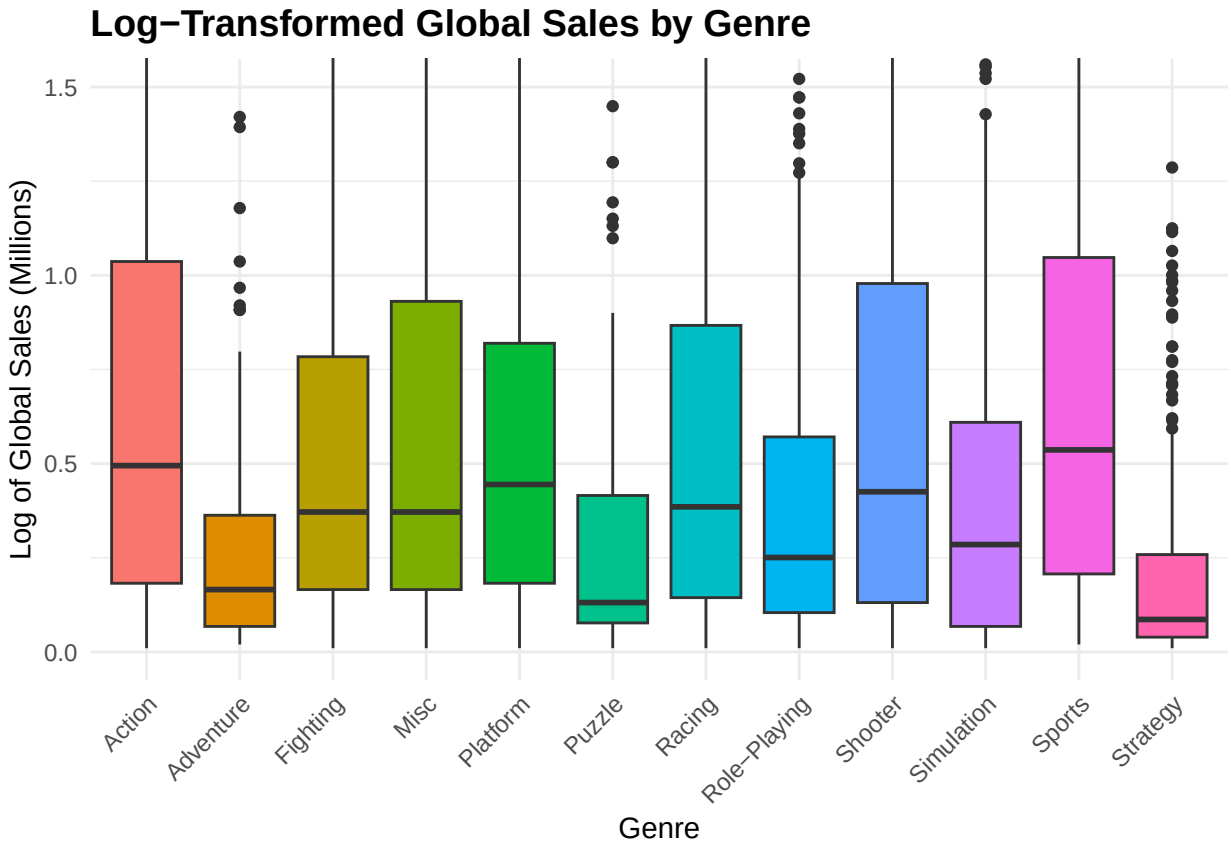


Figure 4.2: Log-Transformed Global Sales by Genre – The box plot compares the distribution of global sales across various genres, showing both the median and the range of sales.

Even though Action games were significantly more popular than any other genre (as seen in Figure 3.2), the log-transformed data indicates that genres like Shooter and Sports tend to have almost the same median sales. This suggests that while Action games have a large presence, Shooter and Sports games might drive more consistent revenue per title.

### 4.3

Now, let's explore the relationship between a game's ESRB rating and its sales. Are games with a particular rating more successful in terms of sales? This box plot of log-transformed global sales helping us understand if a game's rating correlates with its commercial success.

```r
# Box plot to display log-transformed global sales by rating category
ggplot(video_games_clean, aes(x = Rating, y = Log_Global_Sales, fill = Rating)) +
  geom_boxplot() + coord_cartesian(ylim = c(0, 1.5)) +
  labs(
    title = "Log-Transformed Global Sales by Rating",
    x = "Rating",
    y = "Log of Global Sales (Millions)") +
  scale_x_discrete(labels = rating_labels) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(size = 14, face = "bold"),
    legend.position = "none")
```

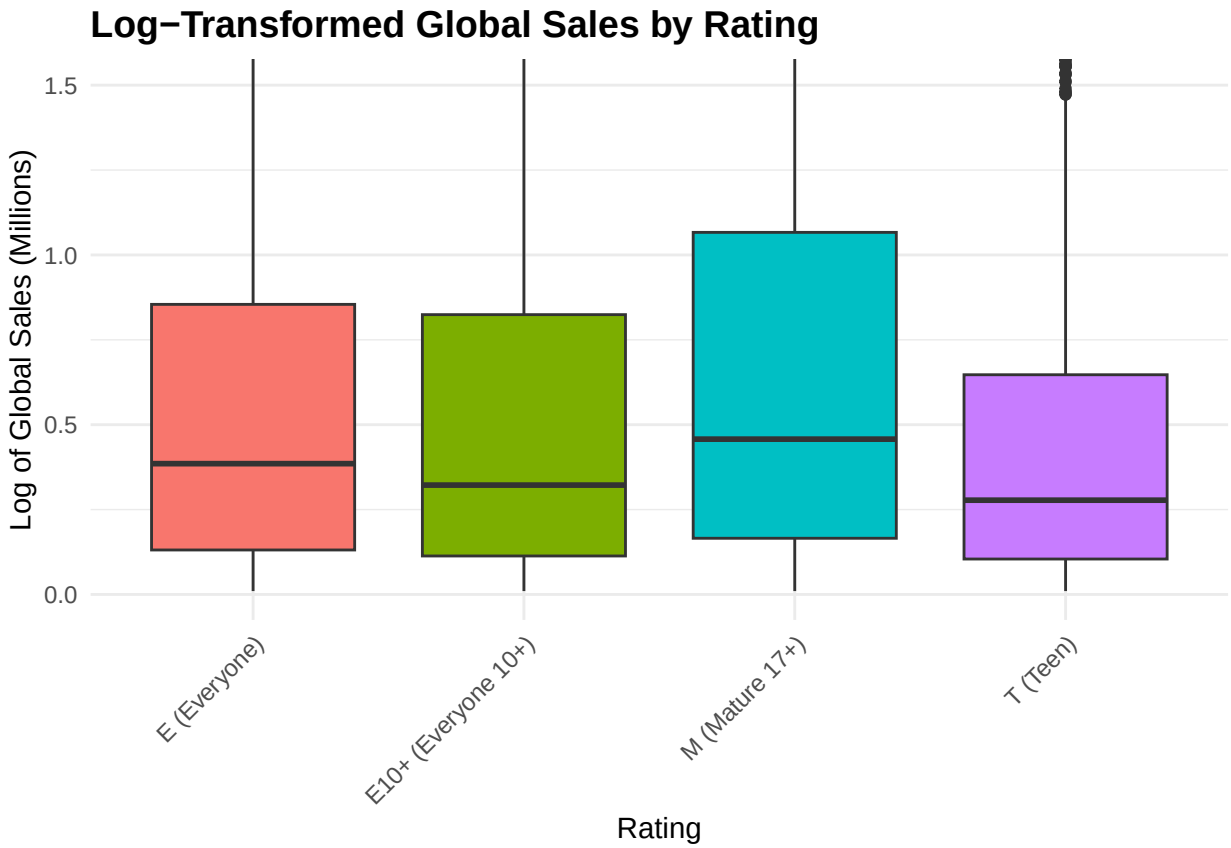


Figure 4.3: Log-Transformed Global Sales by Rating – This box plot shows the distribution of global sales for games with different ESRB ratings.

While Figure 3.3 revealed that most games are rated E or T, the sales data here suggests that M-rated games tend to have higher median sales, even though they are fewer in number. This indicates that while E and T-rated games cater to a broader audience, M-rated games might be more targeted but highly lucrative.

### 4.4

Let's investigate if there's a connection between user ratings and a game's commercial performance. Does a higher user score mean better sales? We use a scatter plot to visualize this relationship, highlighting trends in sales based on user feedback.

```r
# Scatter plot to show the relationship between user score and log-transformed global sales
ggplot(video_games_clean, aes(x = User_Score, y = Log_Global_Sales, color = User_Score)) +
  geom_point(alpha = 0.7, size = 2) + scale_color_viridis_c(option = "D") +
  labs(title = "User Score vs. Global Sales", x = "User Score",
       y = "Global Sales (Millions)", color = "User Score") +
  theme_minimal() +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10),
        legend.position = "right")
```

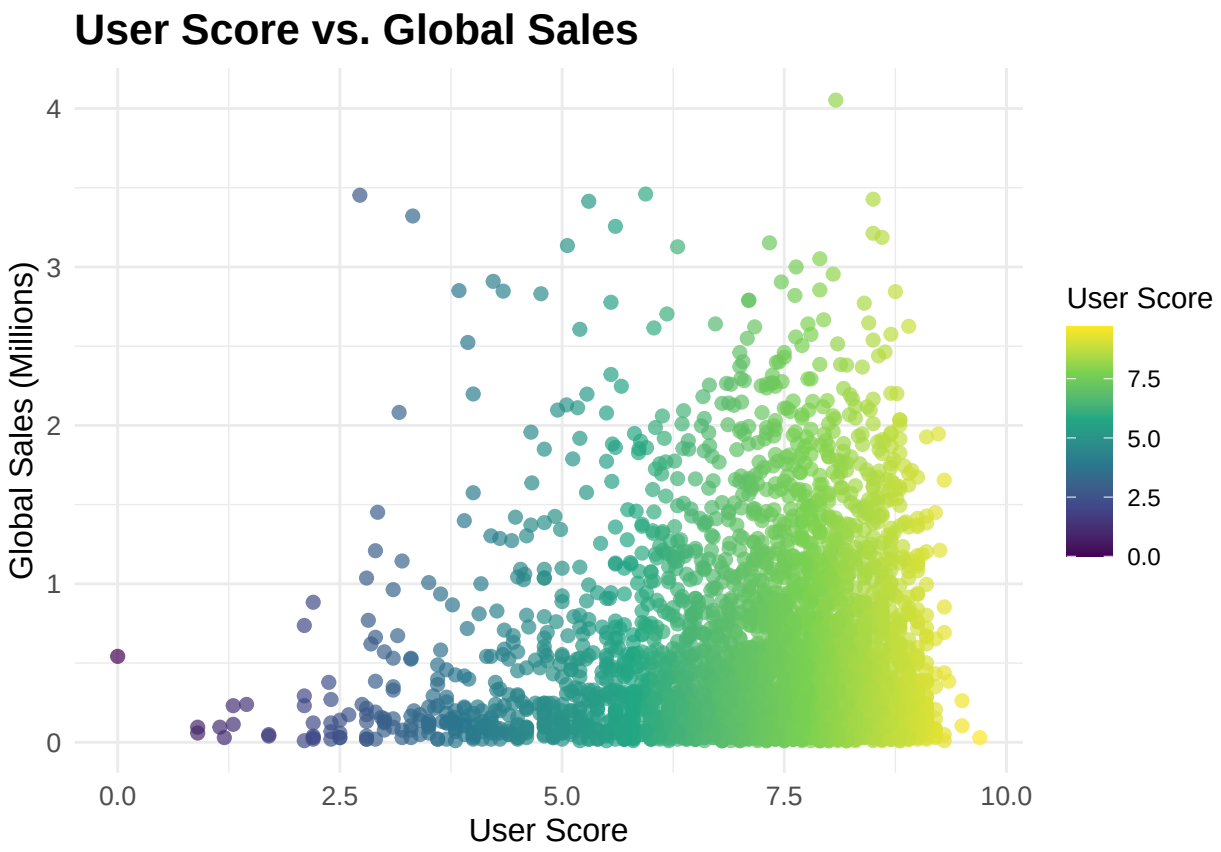


Figure 4.4: User Score vs. Global Sales – The scatter plot shows how user scores correlate with global sales.

There's a positive trend, suggesting that games with higher user scores tend to have better sales, but there's also considerable variability, indicating that other factors might influence sales as well.

### 4.5

Finally, we compare games released exclusively on a single platform versus those released on multiple platforms. This analysis helps us understand whether cross-platform releases impact a game's average sales.

```r
# Create a new column to categorize games as 'Single Platform' or 'Multiple Platform'
video_games_clean <- video_games_clean %>%
  mutate(Platform_Type = ifelse(
    rowSums(select(., DS, PC, PS2, PS3, XB, PSP, X360)) == 1, "Single Platform",
    "Multiple Platform"))
```

```r
# Calculate average global sales by platform type
avg_global_sales_platform_type <- video_games_clean %>%
  group_by(Platform_Type) %>%
  summarize(Average_Global_Sales = mean(Global_Sales, na.rm = TRUE))

# Create a bar plot for average global sales by platform type
ggplot(avg_global_sales_platform_type, aes(x = Platform_Type, y = Average_Global_Sales,
  fill = Platform_Type)) + geom_bar(stat = "identity", color = "black") +
  labs(title = "Average Global Sales for Single vs. Multiple Platform Games",
       x = "Platform Type",
       y = "Average Global Sales (Millions)") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14, face = "bold"),
        axis.title = element_text(size = 12),
        legend.position = "none")
```

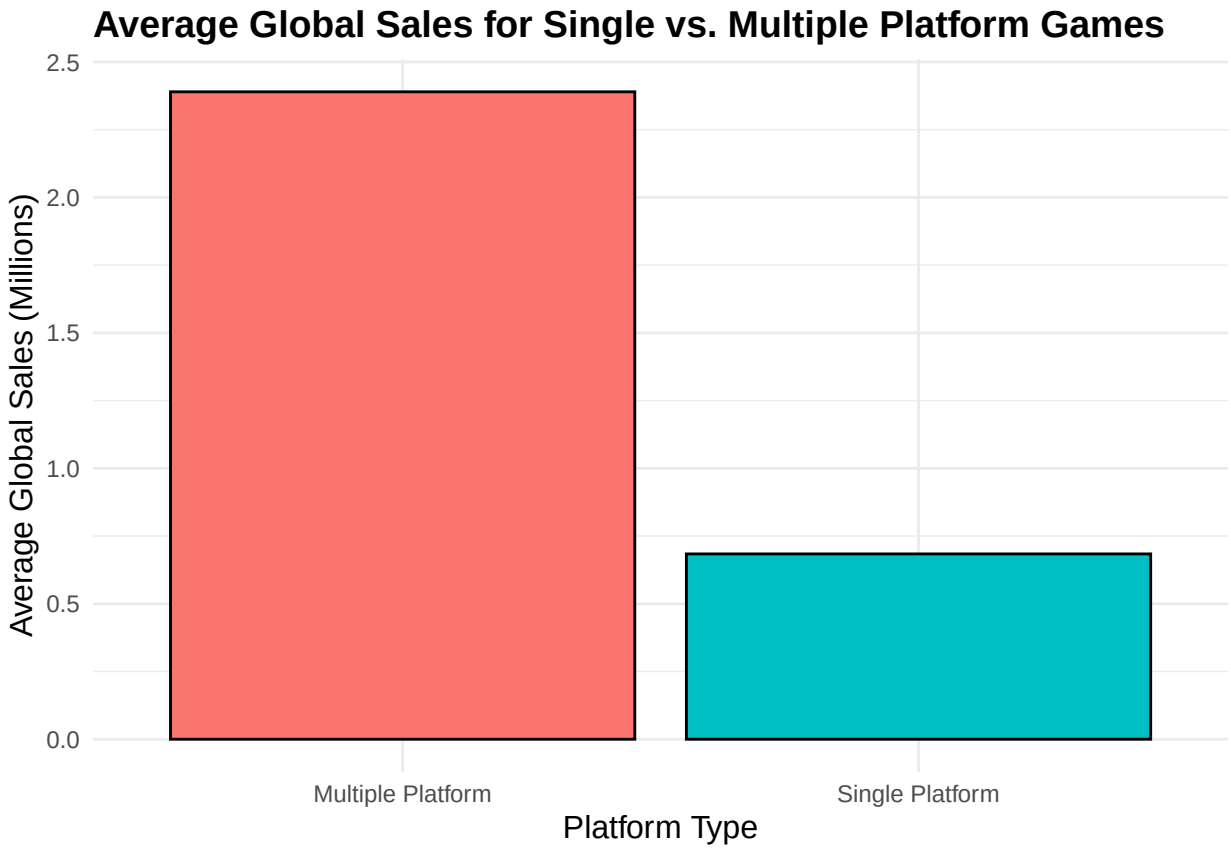


Figure 4.5: Average Global Sales for Single vs. Multiple Platform Games – This bar chart compares the average sales of games that are released on multiple platforms versus those that are exclusive to a single platform.

As expected, games released on multiple platforms tend to have higher average sales, likely due to their broader reach and accessibility to a wider audience.

# 5. Trivariate Analysis

### 5.1

To understand how game sales have evolved over time, we focus on the trends in total global sales for the most popular genres between 1995 and 2015. This will help us see if certain genres had peak years or have remained consistently popular over time.

```
# Adjusting width so that the code stays within margins
options(width = 55)

# Aggregate global sales by year and genre
sales_trend <- video_games_clean %>%
  group_by(Year, Genre) %>%
  summarize(Total_Global_Sales = sum(Global_Sales, na.rm = TRUE), .groups = "drop")

# Filter data to focus on the top 5 genres and years 1995-2015 for trend analysis
filtered_sales_trend <- sales_trend %>%
  filter(Genre %in% c("Action", "Misc", "Racing", "Shooter", "Sports") &
         Year >= 1995 & Year <= 2015)

# Create a line plot for the selected genres over the specified year range
ggplot(filtered_sales_trend, aes(x = Year, y = Total_Global_Sales, color = Genre)) +
  geom_line(linewidth = 1.2) +
  labs(
    title = "Global Sales Trend Over Time (1995-2015) for Top 5 Genres",
    x = "Year",
    y = "Total Global Sales (Millions)",
    color = "Genre") +
  scale_color_brewer(palette = "Set2") +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "right")
```

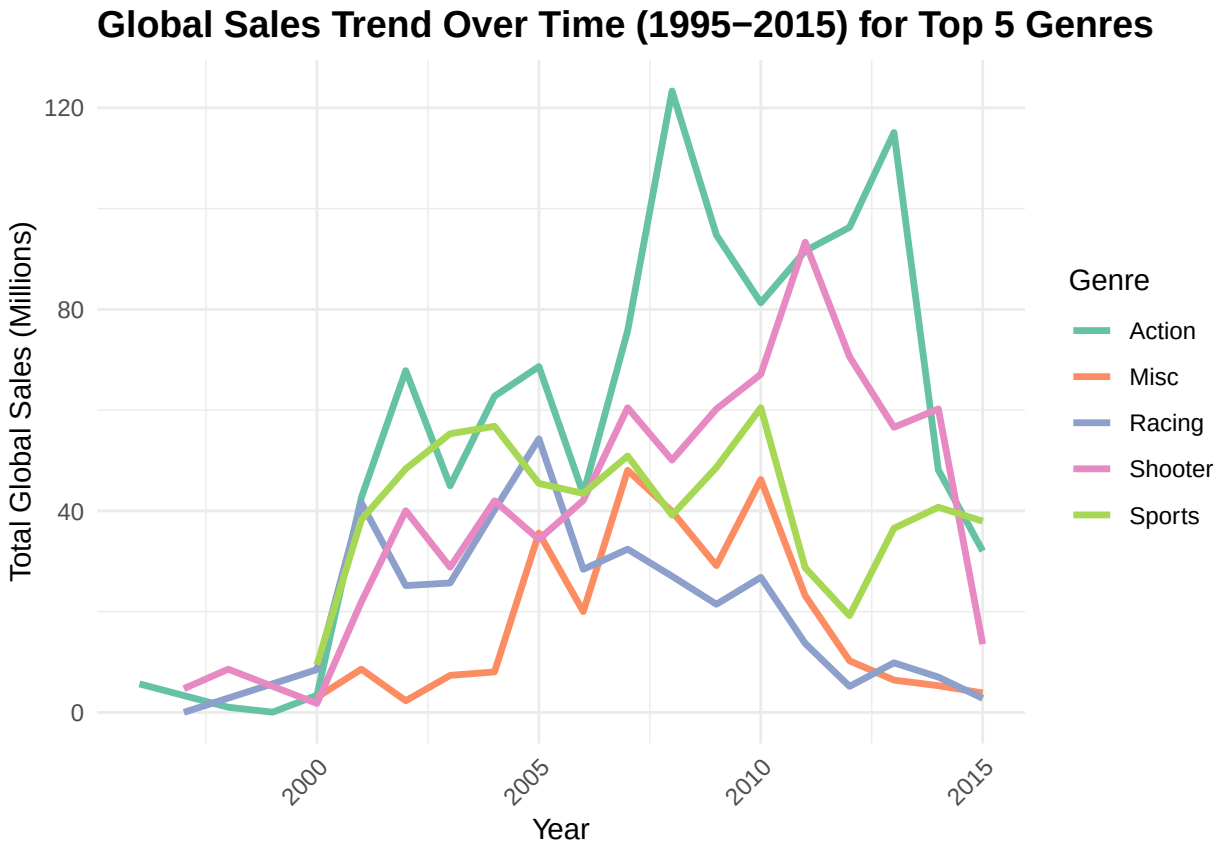


Figure 5.1: Global Sales Trend Over Time (1995-2015) for Top 5 Genres – This line plot illustrates how global sales for Action, Misc, Racing, Shooter, and Sports genres have changed over the years.

We can observe that genres like Action and Sports had more consistent sales over time, while genres like Shooter saw a significant rise, particularly in the 2005-2010 period. This suggests a surge in popularity for shooter games during that time, possibly tied to key game releases and technological advancements in gaming. Comparing this to the distribution of game genres in Figure 3.2, we notice that even though Action genre has more titles, Shooter games have gained traction in sales over time. This indicates that while Action games have always been popular, the Shooter genre caught up in sales due to successful game franchises or rising interest in competitive gaming during this period.

# 6. Predicting with straight lines

### 6.1 Linear Regression

Let's see if success in North America translates to global success. To find out, we run a simple linear regression, comparing North American sales to global sales. It's a way to test the theory that if a game does well in one region, it might just be a hit everywhere.

```
# Adjusting width so that the code stays within margins
options(width = 80)

#Add log transformation for NA_Sales to address skewness
video_games_clean$Log_NA_Sales <- log(video_games_clean$NA_Sales + 1)

# Calculate adjusted global sales by removing NA_Sales from Global_Sales
```

```r
# This prevents overfitting, as Global_Sales initially includes NA_Sales.
video_games_clean$Adjusted_Global_Sales <- video_games_clean$Global_Sales -
  video_games_clean$NA_Sales

# Log-transform the adjusted global sales to reduce skewness
video_games_clean$Log_Adjusted_Global_Sales <- log(video_games_clean$Adjusted_Global_Sales + 1)

# Fit a linear regression model using Log_NA_Sales to predict Log_Adjusted_Global_Sales
model_adjusted <- lm(Log_Adjusted_Global_Sales ~ Log_NA_Sales, data = video_games_clean)

# Generate predictions for Log_Adjusted_Global_Sales from the model
video_games_clean$predicted_Log_Adjusted_Global_Sales <- predict(model_adjusted,
    newdata = video_games_clean)
```

```r
# Adjusting width so that the code stays within margins
options(width = 55)

# Plot actual vs. predicted log-adjusted global sales
ggplot(video_games_clean) +
  geom_point(
    aes(x = Log_NA_Sales, y = Log_Adjusted_Global_Sales, color = "Actual Values"),
    size = 2, alpha = 0.6) +
  geom_point(
    aes(x = Log_NA_Sales, y = predicted_Log_Adjusted_Global_Sales, color =
          "Predicted Values"),
    size = 2, alpha = 0.6) +
  labs(
    title = "Actual vs. Predicted Log Adjusted Global Sales for Model 1",
    x = "Log NA Sales",
    y = "Log Adjusted Global Sales",
    color = "Legend") +
  scale_color_manual(
    values = c("Actual Values" = "dodgerblue", "Predicted Values" = "orange")) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    legend.position = "top",
    legend.title = element_blank())
```

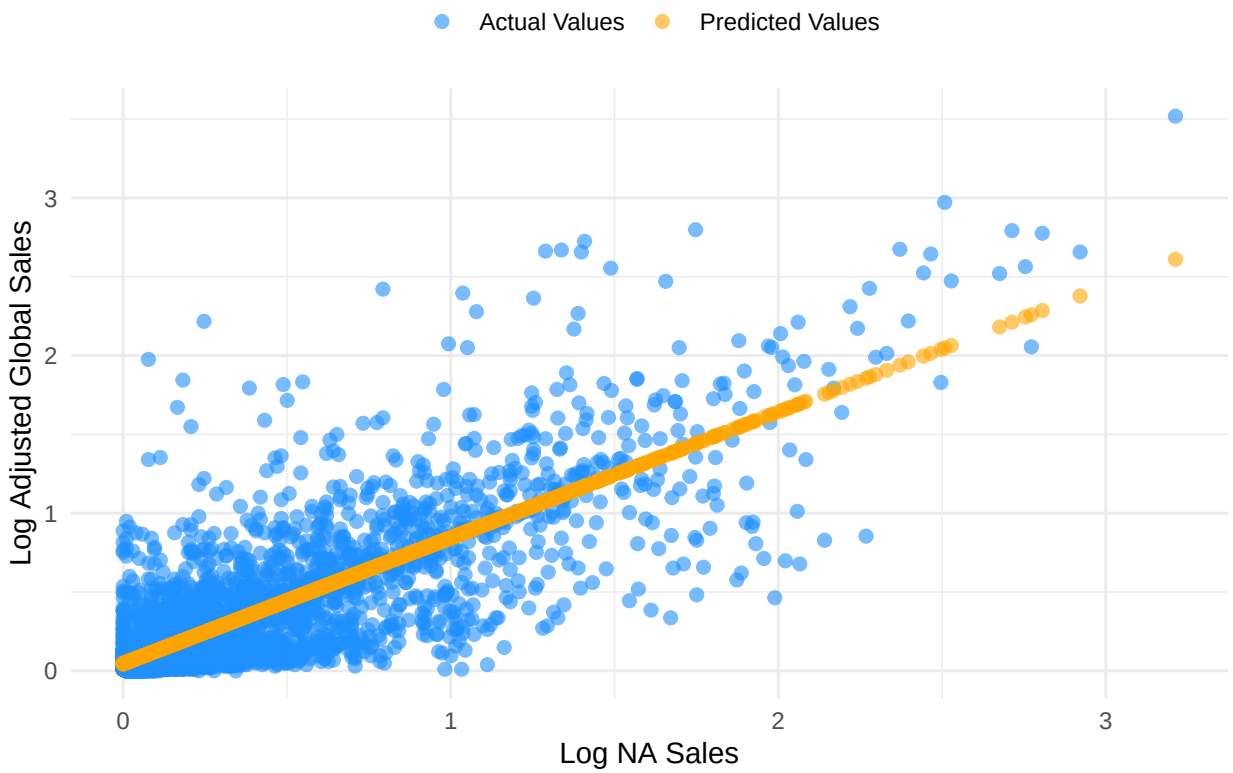


Figure 6.1: Actual vs. Predicted Log Global Sales for Model 1 – This plot compares the actual global sales against the predictions based on North American sales.

The blue points show the real values, while the yellow ones represent what the model predicts. We can see a positive trend which indicates that higher sales in North America generally lead to higher global sales.

### 6.2 Multiple Regression

Now, let's take it up a notch and see what really drives a game's global success. We use a multiple regression model, bringing in all kinds of factors—like genre, rating, release year, and platform. This helps us predict global sales and see which variables truly make a difference.

```
# Adjusting width
options(width = 85)

# Generate the initial model using a variety of independent variables
initial_model <- lm(Log_Global_Sales ~ Genre + Rating + Year + DS + PC + PS2 + PS3 + XB + PSP + X360,
data = video_games_clean)

# Using Anova to remove any variables with P - value > 0.05.
suppressWarnings(Anova(initial_model))
```

```
## Anova Table (Type II tests)
##
## Response: Log_Global_Sales
##           Sum Sq   Df  F value    Pr(>F)
```

```
## Genre      20.08   11   7.4207  1.04e-12 ***
## Rating      9.07    3  12.2856  5.41e-08 ***
## Year        0.18    1   0.7386    0.3902
## DS         52.87    1 214.9333 < 2.2e-16 ***
## PC         62.40    1 253.6720 < 2.2e-16 ***
## PS2        66.52    1 270.4375 < 2.2e-16 ***
## PS3        62.25    1 253.0936 < 2.2e-16 ***
## XB         29.74    1 120.8876 < 2.2e-16 ***
## PSP        20.89    1  84.9170 < 2.2e-16 ***
## X360       30.42    1 123.6909 < 2.2e-16 ***
## Residuals 851.81 3463
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Removed year and made the final model
model_2 <- lm(Log_Global_Sales ~ Genre + Rating + DS + PC + PS2 + PS3 + XB + PSP + X360,
data = video_games_clean)

# Generate predictions using model_2
video_games_clean$predicted_Log_Global_Sales_2 <- predict(model_2, newdata = video_games_clean)
```

```r
# Adjusting width so that the code stays within margins
options(width = 55)

# Create a ggplot for actual vs. predicted values for model_2 with a legend
ggplot(video_games_clean) +
  geom_point(
    aes(x = Log_Global_Sales, y = predicted_Log_Global_Sales_2, color = "Predicted Values"),
    size = 2, alpha = 0.6) +
  geom_point(
    aes(x = Log_Global_Sales, y = Log_Global_Sales, color = "Actual Values"),
    size = 2, alpha = 0.6) +
  labs(
    title = "Actual vs. Predicted Log Global Sales for Model 2",
    x = "Actual Log Global Sales",
    y = "Predicted Log Global Sales",
    color = "Legend") +
  scale_color_manual(
    values = c("Actual Values" = "dodgerblue", "Predicted Values" = "orange"),
    labels = c("Actual Values", "Predicted Values")) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 14, face = "bold"),
    legend.position = "top",
    legend.title = element_blank())
```

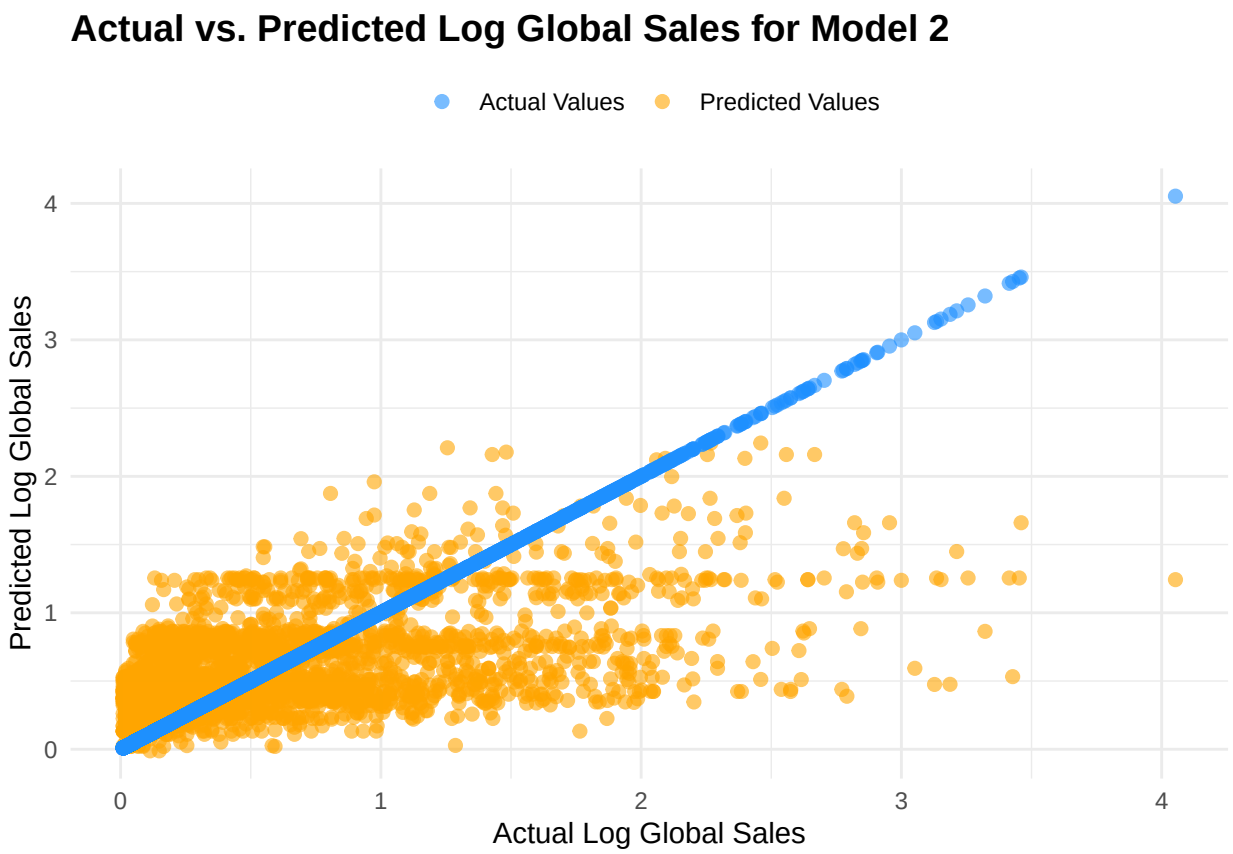


Figure 6.2: Actual vs. Predicted Log Global Sales for Model 2 – This scatter plot shows the comparison between the actual global sales and the model's predictions using multiple factors.

Blue points represent the actual sales values, while yellow ones show the predictions. The closer the points align along the diagonal line, the better the model's predictions. While many predictions are close, some outliers indicate that not every game's performance can be neatly captured by the chosen features.

# 7. Classification and Cross-Validation

In this section, we shift our focus from predicting continuous sales figures to a classification problem. We will use random forest model to predict global sales. To ensure our model is working as expected, we will use confusion matrix and ROC curves

### 7.1 Preparing Data for Classification

First we will create a new column to categorize games into "High Sales" or "Low Sales" based on their global sales. This binary classification helps us understand which factors are most important in determining whether a game will be a top seller. We have chosen 0.4 million to spilt the data because the median of global sales was close to 0.4. We do this so that the new column created has balanced data.

```
# Adjusting width
options(width = 100)

# Games with Global_Sales > 0.4 million are classified as 'High Sales'
video_games_clean$Sales_Category <- ifelse(video_games_clean$Global_Sales > 0.4, "High", "Low")
```

```r
# Convert the target variable into a factor for classification
video_games_clean$Sales_Category <- as.factor(video_games_clean$Sales_Category)
```

Again, here we are ensuring that the newly created column is balanced.

```r
# Ensuring that low and high values are almost equal in number
summary(video_games_clean$Sales_Category)
```

```
## High  Low
## 1749 1737
```

### 7.2 Training - Testing Data

Now we will create training and testing data sets. 80% of the data will be randomly split to form the training set whereas the rest will be the testing set.

```r
# Set seed for reproducibility
set.seed(123)

# Create a random split of the data
data_split <- initial_split(video_games_clean)

# Extract training and testing sets
train_data <- training(data_split)
test_data <- testing(data_split)
```

Now, coming to the current topic at hand, we will use random forest model for our classification.

```r
# Train a random forest model
model_rf <- randomForest(Sales_Category ~ Year + User_Score + User_Count + Genre + Rating + DS + PC +
    PS2 + PS3 + PSP + XB + X360, data = train_data, importance = TRUE)
```

### 7.3 Confusion Matrix

After training our random forest model, we assess its performance using a confusion matrix. This matrix allows us to see how often the model correctly classifies games into "High" or "Low" sales categories, giving us a clearer picture of its accuracy and effectiveness.

```r
# Predict the sales category on the test set
predictions_rf <- predict(model_rf, newdata = test_data)

# Generate confusion matrix
confusionMatrix(predictions_rf, test_data$Sales_Category)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction High Low
##       High  339  65
```

```
##       Low   113 355
##
##                Accuracy : 0.7959
##                  95% CI : (0.7676, 0.8222)
##     No Information Rate : 0.5183
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5928
##
##  Mcnemar's Test P-Value : 0.000427
##
##             Sensitivity : 0.7500
##             Specificity : 0.8452
##          Pos Pred Value : 0.8391
##          Neg Pred Value : 0.7585
##              Prevalence : 0.5183
##          Detection Rate : 0.3888
##    Detection Prevalence : 0.4633
##       Balanced Accuracy : 0.7976
##
##        'Positive' Class : High
##
```

Our random forest model achieved a sensitivity of 0.75 and a specificity of 0.845. In simple terms, sensitivity measures how well the model identifies games with high sales, while specificity indicates its accuracy in spotting games with low sales. Together, these values give us a good sense of the model's balance in correctly classifying both high and low sales categories.

### 7.4 ROC Curve and AUC

Finally, we generate a ROC curve to evaluate the trade-off between true positives and false positives at different classification thresholds.

```
# Adjusting width so that the code stays within margins
options(width = 60)

# Predict probabilities for the test data
probabilities <- predict(model_rf, newdata = test_data, type = "prob")

# Generate the ROC curve for the random forest model
roc_curve <- roc(response = test_data$Sales_Category, predictor =
  probabilities[, "High"], quiet = TRUE)

# Prepare ROC data for plotting
roc_data <- data.frame(
  FPR = 1 - roc_curve$specificities, # False Positive Rate
  TPR = roc_curve$sensitivities # True Positive Rate
)

# Create the ROC plot
ggplot(roc_data, aes(x = FPR, y = TPR)) +
  geom_line(color = "blue", linewidth = 1.2) +
  geom_abline(linetype = "dashed", color = "gray") +
```

```
  labs(title = "ROC Curve for Random Forest Model",
       x = "1 - Specificity",
       y = "Sensitivity") +
  coord_equal() +
  theme_minimal() +
  theme(plot.title = element_text(size = 14, face = "bold"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))
```

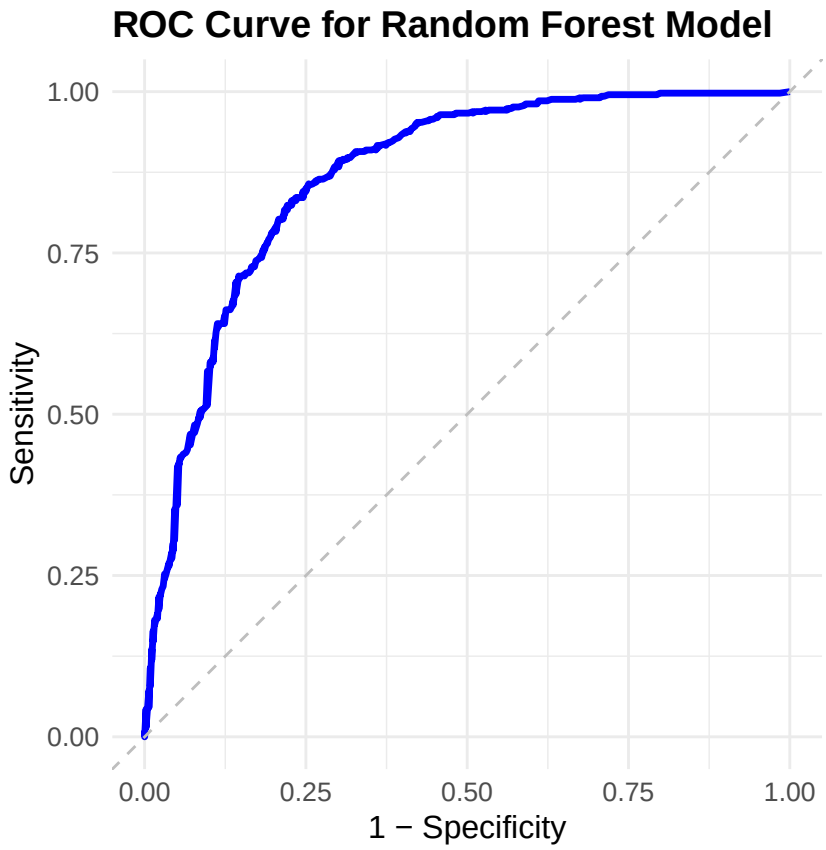


Figure 7.1: This ROC curve demonstrates the model's ability to classify games as having high or low sales. The curve rises sharply, showing a strong balance between sensitivity and specificity, with a clear gap from the diagonal line, indicating the model performs significantly better than random guessing.

```
# Calculate AUC
auc(roc_curve)
```

```
## Area under the curve: 0.8694
```

The Area Under the Curve (AUC) score gives us a single number to summarize the model's overall performance. We get an AUC 0.8694. This high score is another indication that the model is very effective at predicting whether a game will have high or low sales

## 8. Predicting with curved lines

Now, let's get a bit more flexible with our predictions by exploring non-linear models. Linear models can be limiting, so we'll try something different to capture complex relationships in the data. We'll start by

prepping the data, then dive into a logistic regression model to get a sense of how well we can classify games based on their probability of high or low sales. Finally, we'll use a calibration plot to see if the model's predicted probabilities match up with reality.

### 8.1 Data Pre-Processing Recipe

Before we dive into modeling, we need to make sure our data is ready to go. This step involves normalizing numbers and converting categorical values into a form that works with the models. Proper pre-processing helps us get the best results from our models.

```
# Adjusting width so that the code stays within margins
options(width = 80)

# Set up a recipe for the classification task
game_sales_recipe <- recipe(Sales_Category ~ Year + Genre + Rating + DS + PC + PS2 + PS3
+ PSP + XB + X360, data = train_data) %>%
  # Step 1: Normalize numeric predictors
  step_normalize(all_numeric(), -all_outcomes())%>%
  # Step 2: Convert categorical predictors with multiple levels to dummy variables
  step_dummy(all_nominal(), -all_outcomes())

# Prepare the recipe using the training data
game_sales_prep <- prep(game_sales_recipe)

# Create pre-processed training set using bake()
game_sales_train_preproc <- bake(game_sales_prep, new_data = train_data)

# Create pre-processed testing set using bake()
game_sales_test_preproc <- bake(game_sales_prep, new_data = test_data)
```

### 8.2 Logistic Regression

With our data prepared, we start with logistic regression. This model predicts the probability that each game will fall into the high sales category. Instead of a simple yes or no, we get a sense of how likely each game is to succeed, giving us a more detailed look at our predictions.

```
# Define logistic regression model specification
logistic_spec <- logistic_reg() %>%
    set_engine("glm") %>%
    set_mode("classification")

# Create a workflow for logistic regression with the prepped data
logistic_workflow <- workflow() %>%
    add_recipe(game_sales_recipe) %>%
    add_model(logistic_spec)

# Fit the logistic regression model on the entire training data
final_logistic_fit <- fit(logistic_workflow, data = train_data)

# Generate probability predictions for the test data
logistic_probabilities <- predict(final_logistic_fit, new_data = test_data, type = "prob")
```

### 8.3

Finally, we check if our model's predictions are on target by creating a calibration plot. This plot lets us compare the predicted probabilities of high sales with how often high sales actually happen, giving us insight into the model's accuracy.

```
# Adjusting width so that the code stays within margins
options(width = 65)

# Create calibration data by binning predicted probabilities
calibration_data <- test_data %>%
  bind_cols(logistic_probabilities %>% select(.pred_High)) %>%
  rename(predicted_prob_high = .pred_High) %>%
  mutate(pred_bin = ntile(predicted_prob_high, 7)) %>%
  group_by(pred_bin) %>%
  summarize(
    avg_pred_prob = mean(predicted_prob_high),
    observed_prop = mean(Sales_Category == "High")
)

# Create the calibration plot
ggplot(calibration_data, aes(x = avg_pred_prob, y = observed_prop)) +
  geom_line(color = "blue", linewidth = 1.2) +
  geom_point(color = "red", size = 2.5) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
  labs(
    title = "Calibration Plot for Logistic Regression Model",
       x = "Predicted Probability of High Sales",
       y = "Observed Proportion of High Sales") +
  theme_minimal() +
  theme(plot.title = element_text(size = 14, face = "bold"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10))
```

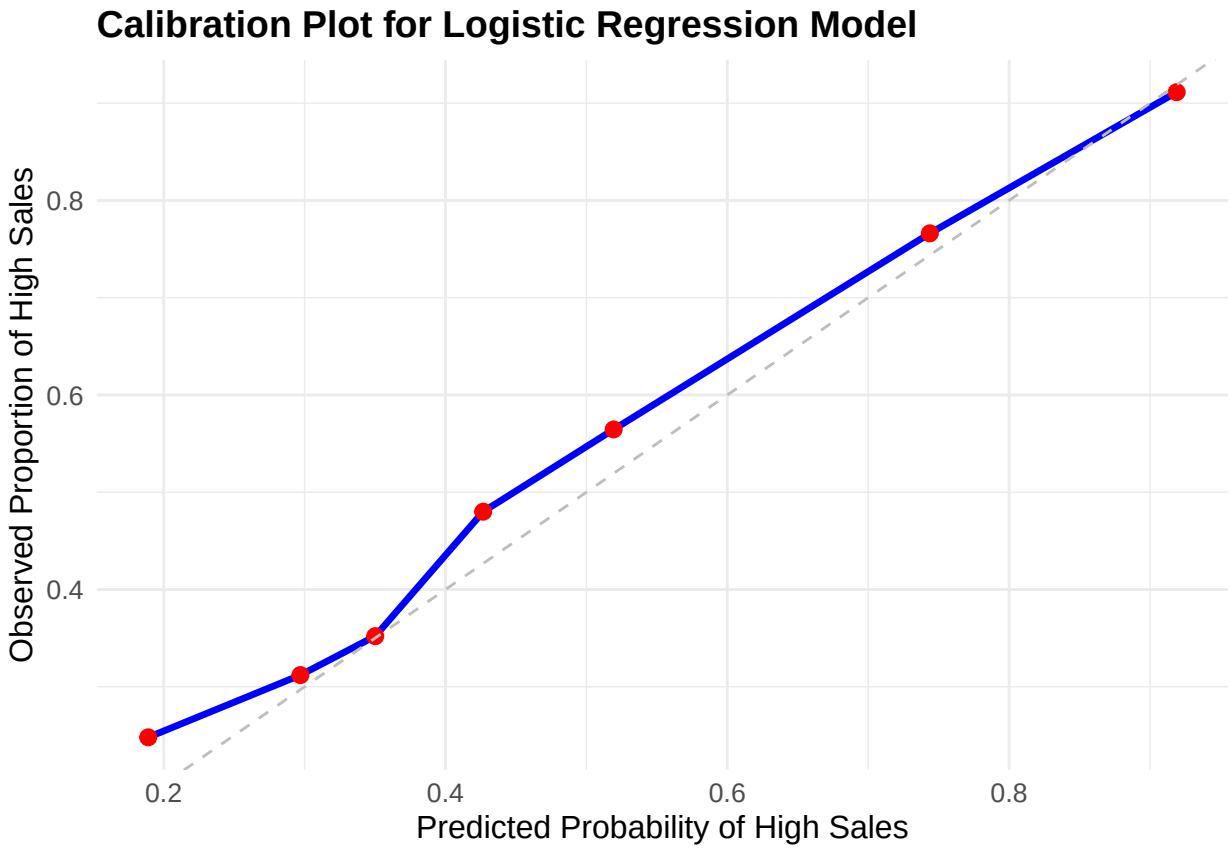


Figure 8.1: Calibration Plot for Logistic Regression Model – This plot shows if the model's predicted probabilities line up with real outcomes, with points near the diagonal indicating well-matched predictions.

The points on the calibration plot mostly hug the diagonal line, which is a good sign. This means that the model's predicted probabilities align pretty closely with actual high sales occurrences. So, if the model predicts a high chance of strong sales, it's usually on the mark—giving us confidence in its accuracy.

# 9. Predicting with fancy lines

Building on the non-linear approach from Section 8, we now dive into more advanced non-linear models: a classification tree and k-nearest neighbors (KNN). These models let us explore different ways to capture patterns in game sales, especially when relationships between features are complex. A classification tree uses branching decisions to predict sales categories, while KNN classifies a game by comparing it to its closest neighbors. After training both models, we'll evaluate and compare their performance to see which one better identifies high- and low-sales games.

### 9.1 Classification Tree

With the data prepared, we start with a classification tree model. This model creates a series of branching decisions based on game attributes, helping us predict if a game falls into the "High Sales" or "Low Sales" category.

```
# Specify the classification tree model with tuning
tree_spec <- decision_tree(cost_complexity = tune()) %>%
    set_engine("rpart") %>%
```

```
    set_mode("classification")

# Create a workflow for the tree model with prepped data
tree_workflow <- workflow() %>%
    add_recipe(game_sales_recipe) %>%
    add_model(tree_spec)

# Define cross-validation and tuning grid
cv_folds <- vfold_cv(train_data, v = 5)
cc_grid <- grid_regular(cost_complexity(), levels = 50)

# Tune the tree model
tree_results <- tune_grid(tree_workflow, resamples = cv_folds, grid = cc_grid,
    metrics = metric_set(roc_auc))

# Select best cost complexity and finalize the model
best_cc <- select_best(tree_results, metric = "roc_auc")
final_tree_workflow <- finalize_workflow(tree_workflow, best_cc)

# Fit the final model to the preprocessed training data
final_tree_fit <- fit(final_tree_workflow, data = train_data)

# Generate probability predictions for the test data
tree_probabilities <- predict(final_tree_fit, new_data = test_data,
    type = "prob")
```

### 9.2 K-Nearest Neighbors (KNN)

Next up is KNN, which places each game in high or low sales based on its similarity to other games. By looking at the "neighbors" with the closest attributes, KNN classifies a game based on trends among similar titles.

```
# Define KNN model specification with tuning
knn_spec <- nearest_neighbor(neighbors = tune()) %>%
    set_engine("kknn") %>%
    set_mode("classification")

# Create a workflow for KNN with the prepped data
knn_workflow <- workflow() %>%
    add_recipe(game_sales_recipe) %>%
    add_model(knn_spec)

# Tune the KNN model
knn_results <- tune_grid(knn_workflow, resamples = cv_folds, grid = tibble(neighbors = seq(1,
    100, by = 2)), metrics = metric_set(accuracy, roc_auc))

# Select the best k and finalize the model
best_k <- select_best(knn_results, metric = "roc_auc")
final_knn_workflow <- finalize_workflow(knn_workflow, best_k)

# Fit the final KNN model to the preprocessed training data
final_knn_fit <- fit(final_knn_workflow, data = train_data)
```

```r
# Generate probability predictions for the test data
knn_probabilities <- predict(final_knn_fit, new_data = test_data,
    type = "prob")
```

### 9.3 Comparing Classification Tree and KNN Model based on accuracy

Before diving into the ROC analysis, we first examine the basic accuracy of our two models: the classification tree and KNN. By generating predictions on the test data and calculating the accuracy, we get a straightforward measure of each model's ability to correctly classify games into high and low sales categories. This initial comparison gives us a baseline sense of which model might perform better before we analyze their performance more deeply with ROC curves.

```r
# Generate predictions for the classification tree model
tree_predictions <- predict(final_tree_fit, new_data = test_data) %>%
  bind_cols(test_data) %>%
  mutate(predicted = .pred_class)

# Calculate accuracy for the classification tree model
tree_accuracy <- tree_predictions %>%
  metrics(truth = Sales_Category, estimate = predicted) %>%
  filter(.metric == "accuracy") %>%
  pull(.estimate)

# Generate predictions for the KNN model
knn_predictions <- predict(final_knn_fit, new_data = test_data) %>%
  bind_cols(test_data) %>%
  mutate(predicted = .pred_class)

# Calculate accuracy for the KNN model
knn_accuracy <- knn_predictions %>%
  metrics(truth = Sales_Category, estimate = predicted) %>%
  filter(.metric == "accuracy") %>%
  pull(.estimate)

# Print the results
print(paste("Classification Tree Accuracy:", round(tree_accuracy, 3)))
```

```
## [1] "Classification Tree Accuracy: 0.688"
```

```r
print(paste("K-NN Accuracy:", round(knn_accuracy, 3)))
```

```
## [1] "K-NN Accuracy: 0.686"
```

The accuracy results show that the classification tree achieves an accuracy of 68.8%, while the KNN model achieves 68.6%. Although the difference is minimal, the classification tree has a slight edge in accuracy, suggesting it may be marginally more effective for this dataset.

### 9.4 Evaluating Classification Tree and KNN Model with ROC Analysis

To see which model does a better job, we compare the ROC curves for both the classification tree and KNN. This comparison helps us see where each model's strengths lie, giving us a clear view of which one captures the sales patterns more accurately.

```r
# Generate ROC curve data for the Classification Tree
tree_roc <- roc(response = test_data$Sales_Category, predictor = tree_probabilities$.pred_High,
    levels = c("Low", "High"), quiet = TRUE)

# Generate ROC curve data for the KNN Model
knn_roc <- roc(response = test_data$Sales_Category, predictor = knn_probabilities$.pred_High,
    levels = c("Low", "High"), quiet = TRUE)

# Convert ROC data to data frames for ggplot
tree_roc_data <- data.frame(FPR = 1 - tree_roc$specificities, TPR = tree_roc$sensitivities,
    Model = "Classification Tree")

# Generate ROC curve data for the KNN Model
knn_roc_data <- data.frame(FPR = 1 - knn_roc$specificities, TPR = knn_roc$sensitivities,
    Model = "KNN")

# Combine both ROC data frames
combined_roc_data <- rbind(tree_roc_data, knn_roc_data)

# Plot combined ROC curves
ggplot(combined_roc_data, aes(x = FPR, y = TPR, color = Model)) +
    geom_line(linewidth = 1.2) + geom_abline(linetype = "dashed",
    color = "gray") + labs(title = "ROC Curve for Classification Tree and KNN Models",
    x = "1 - Specificity", y = "Sensitivity") + theme_minimal() +
    theme(legend.position = "bottom")
```

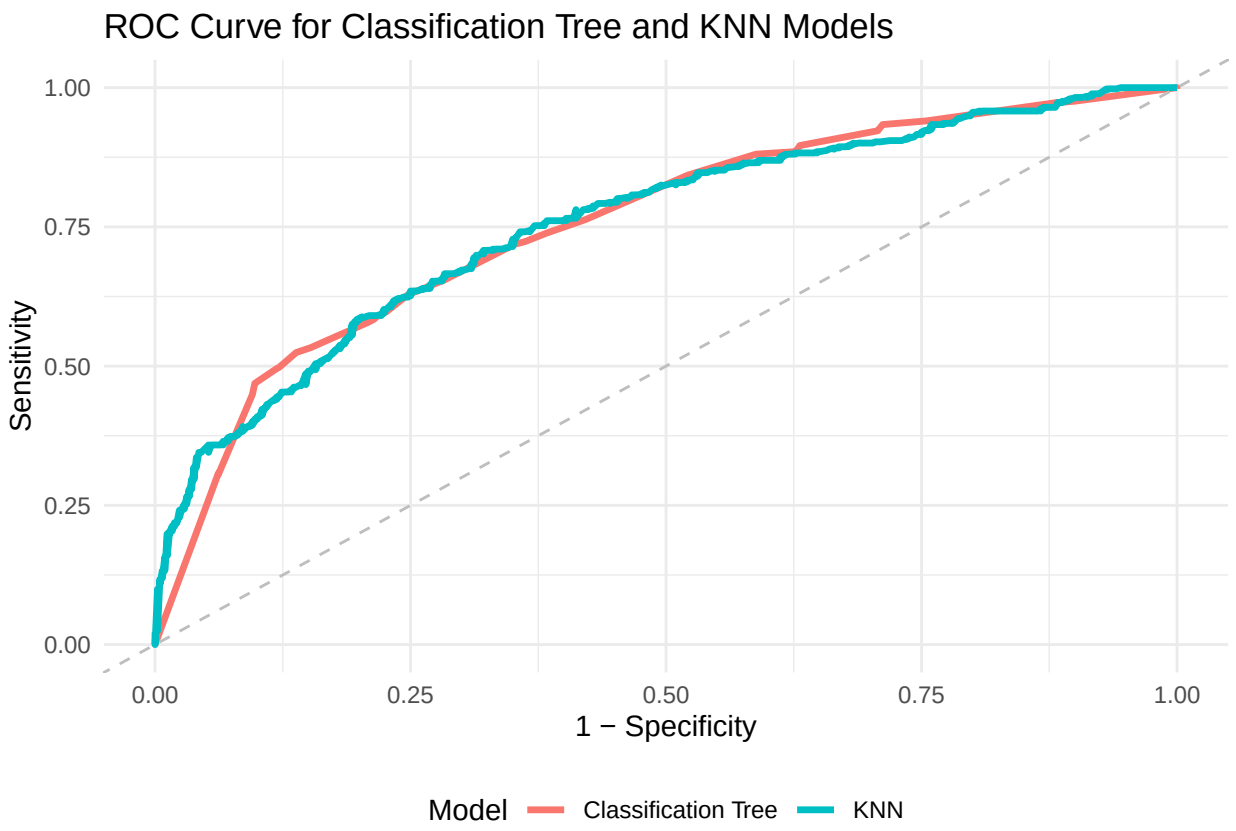


Figure 9.1: ROC Curve Comparison for Classification Tree and KNN Models – This combined ROC curve

shows how each model performs in distinguishing high- from low-sales games, with steeper curves indicating stronger classification.

Although both models are neck and neck, the classification tree does manage to pull ahead in a few areas of the ROC curve. This small edge lines up with what we saw in Section 9.3, where the classification tree had a slightly higher accuracy score. It's a close call, but these results suggest that the classification tree might just be the more reliable option for predicting high and low sales games.

# 10. Conclusion

In this project, we set out to dig into video game sales data to uncover trends and build models that could help us predict a game's sales potential. We began with a thorough data cleaning and transformation process, turning our dataset into a reliable source that reflects unique games instead of repeated entries. This step was essential to ensure accuracy and give us a strong foundation for analysis.

From there, we explored the data, starting with simple counts and distributions. We looked at the platforms with the most games, the genres that dominated the market, and the ratings that targeted different age groups. These initial explorations gave us a clear picture of the industry's structure. For example, we found that while action games were highly popular, genres like shooter and sports often generated more consistent revenue per title.

Next, we moved into modeling, starting with linear and multiple regression to predict global sales based on factors like North American sales, genre, and platform. These models helped us see which factors had the strongest influence on sales, highlighting that certain platforms and genres really do stand out when it comes to revenue potential.

Then came the shift to classification, where we categorized games as either high or low sales and applied a random forest model. This approach allowed us to identify which features were most helpful in predicting high sales games. Continuing with classification, we explored non-linear models like logistic regression and calibration plots to get more precise probability estimates.

Finally, we tried more advanced models with classification trees and K-nearest neighbors (KNN). Both of these methods provided valuable insights, but in the end, the classification tree had a slight edge. Its better accuracy and ROC curve showed it could capture patterns in our data effectively, making it a solid choice for this kind of prediction.

In summary, this report showcases a journey through data cleaning, exploratory analysis, and a range of predictive models, each bringing new insights into the video game market. We've seen that factors like genre, platform, and region play significant roles in a game's sales, and we've built models that can leverage these insights for future predictions. The classification tree, with its balance of accuracy and interpretability, stands out as a top performer—but more importantly, this project highlights the power of combining multiple approaches to get a fuller picture of the data.