[Tit-for-Tat] Strategy Identification and Payoff Analysis

Hana Kwon

Oct 27, 2024

Introduction

This document details the approach used to identify player strategies *(Tit-for-Tat, Grim, Always Cooperate, and Always Defect)* and calculate payoffs in an experimental dataset.

Definitions and Assumptions

- **Payoff Values**
  - `r`: *(Reward)* Payoff for mutual cooperation
  - `s`: *(Sucker)* Payoff when one cooperates and the other defects
  - `t`: *(Temptation)* Payoff for defection when the other cooperates
  - `p`: *(Punishment)* Payoff for mutual defection
- **Strategies**
  1. **Always Cooperate**: Player cooperates in all rounds.
  2. **Always Defect**: Player defects in all rounds.
  3. **Grim**: Player starts by cooperating and defects permanently if the opponent defects.
  4. **Tit-for-Tat**: Player mirrors the opponent's previous move.
- **Dataset Information**
  - `session`: Experiment session number.
  - `id`: Participant ID.
  - `oid`: Partner's ID.
  - `supergame`: Match number.
  - `round`: Round number within a supergame.
  - `horizon`: Length of supergame.
  - `coop`: Cooperation indicator (1 if cooperated, 0 otherwise).
  - `r`, `s`, `t`, `p`: Payoff values based on cooperation and defection as described above.

Analysis Overview

- **Core Analysis**
  1. Data Loading and Initial Exploration
  2. Data Preparation and Preprocessing

3. Payoff Analysis
   3.1 Calculate Payoff Based on Actions
   3.2 Round-by-Round Average Payoff Calculation
   3.3 Average Payoff and Variance by Player
   3.4 Calculation and Visualization of T Value
   3.5 Cross-Tabulation of T Values by Game Length
4. Strategy Identification
5. Strategy Payoff Analysis

- **Extended Analysis**

5. Frequency-Weighted Payoff Calculation
6. Performance Against Non-Fixed Strategies
7. Expected Payoff Simulation for Hypothetical Tit-for-Tat Player
8. Visualizations and Graphical Analysis

Core Analysis

Step 1: Data Loading, Initial Exploration, and Preparation

- **Objective:** Load the dataset and examine its structure to ensure successful data import and check for any missing values.

```r
### 1.1 Load Necessary Libraries and Dataset
library(dplyr)
```

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

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

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

```r
library(readr)
library(ggplot2)

file_path <- "~/Desktop/[Tit for Tat] Prof.MacLeod_Hana Kwon/Embrey_2018a_new_data.txt"
data <- read.table(file_path, header = TRUE, sep = "\t", stringsAsFactors = FALSE)

### 1.2 Initial Data Exploration
str(data)      # Check the structure of the data
```

```
## 'data.frame':    33360 obs. of  14 variables:
##  $ id       : int  73 73 73 73 73 73 73 73 73 73 ...
##  $ oid      : int  77 80 75 78 81 86 83 85 74 82 ...
##  $ supergame: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ round    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ horizon  : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ r        : int  51 51 51 51 51 51 51 51 51 51 ...
##  $ s        : int  22 22 22 22 22 22 22 22 22 22 ...
##  $ t        : int  63 63 63 63 63 63 63 63 63 63 ...
##  $ p        : int  39 39 39 39 39 39 39 39 39 39 ...
##  $ g        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ l        : num  1.42 1.42 1.42 1.42 1.42 ...
##  $ sizebad  : num  0.191 0.191 0.191 0.191 0.191 ...
##  $ session  : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ coop     : int  0 0 0 0 1 1 0 0 1 1 ...
```

```r
summary(data)  # Summary of data to examine distributions and any NA values
```

```
##        id             oid          supergame        round      
##  Min.   :  1.0   Min.   :  1.0   Min.   : 1.00   Min.   :1.000  
##  1st Qu.: 98.0   1st Qu.: 98.0   1st Qu.: 7.00   1st Qu.:2.000  
##  Median :150.0   Median :150.0   Median :14.00   Median :3.000  
##  Mean   :152.3   Mean   :152.3   Mean   :14.28   Mean   :3.785  
##  3rd Qu.:212.0   3rd Qu.:212.0   3rd Qu.:21.00   3rd Qu.:5.000  
##  Max.   :284.0   Max.   :284.0   Max.   :30.00   Max.   :8.000  
##                                                                 
##     horizon            r            s              t               p     
##  Min.   :4.000   Min.   :51   Min.   : 5.0   Min.   :63.00   Min.   :39  
##  1st Qu.:4.000   1st Qu.:51   1st Qu.: 5.0   1st Qu.:63.00   1st Qu.:39  
##  Median :8.000   Median :51   Median : 5.0   Median :87.00   Median :39  
##  Mean   :6.571   Mean   :51   Mean   :13.4   Mean   :75.14   Mean   :39  
##  3rd Qu.:8.000   3rd Qu.:51   3rd Qu.:22.0   3rd Qu.:87.00   3rd Qu.:39  
##  Max.   :8.000   Max.   :51   Max.   :22.0   Max.   :87.00   Max.   :39  
##                                                                          
##        g               l            sizebad          session      
##  Min.   :1.000   Min.   :1.417   Min.   :0.191   Min.   : 1.000  
##  1st Qu.:1.000   1st Qu.:1.417   1st Qu.:0.415   1st Qu.: 5.000  
##  Median :3.000   Median :2.833   Median :0.415   Median : 7.000  
##  Mean   :2.012   Mean   :2.133   Mean   :0.504   Mean   : 6.954  
##  3rd Qu.:3.000   3rd Qu.:2.833   3rd Qu.:0.415   3rd Qu.: 9.000  
##  Max.   :3.000   Max.   :2.833   Max.   :1.000   Max.   :12.000  
##                                  NA's   :27700                    
##       coop       
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.3589  
##  3rd Qu.:1.0000  
##  Max.   :1.0000  
## 
```

Step 2: Data Preparation and Preprocessing

- **Objective:** Prepare player and opponent data frames to align cooperation values and payoff values for each round.

```
### 2.1 Create Player and Opponent Data Frames
df_self <- data %>%
  select(id, oid, supergame, round, horizon, coop, r, s, t, p) %>%
  rename(player_id = id, opponent_id = oid, player_coop = coop)

df_opp <- data %>%
  select(id, oid, supergame, round, horizon, coop) %>%
  rename(opponent_id = id, player_id = oid, opponent_coop = coop)

### 2.2 Merge Player and Opponent Data
df_merged <- df_self %>%
  left_join(df_opp, by = c("player_id", "opponent_id", "supergame", "round", "horizon"))

# Check structure of merged dataframe
str(df_merged)
```

```
## 'data.frame':    33360 obs. of  11 variables:
##  $ player_id    : int  73 73 73 73 73 73 73 73 73 73 ...
##  $ opponent_id  : int  77 80 75 78 81 86 83 85 74 82 ...
##  $ supergame    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ round        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ horizon      : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ player_coop  : int  0 0 0 0 1 1 0 0 1 1 ...
##  $ r            : int  51 51 51 51 51 51 51 51 51 51 ...
##  $ s            : int  22 22 22 22 22 22 22 22 22 22 ...
##  $ t            : int  63 63 63 63 63 63 63 63 63 63 ...
##  $ p            : int  39 39 39 39 39 39 39 39 39 39 ...
##  $ opponent_coop: int  0 0 1 1 1 0 1 1 1 1 ...
```

Step 3: Descriptive Payoff Analysis

3.1 Payoff Calculation Based on Actions

- **Objective:** Calculate the payoff based on cooperation and defection combinations for each round and assign values to the `payoff` column.

```
df_merged <- df_merged %>%
  mutate(payoff = case_when(
    player_coop == 1 & opponent_coop == 1 ~ r,  # Both Cooperate
    player_coop == 1 & opponent_coop == 0 ~ s,  # Only Player Cooperates
    player_coop == 0 & opponent_coop == 1 ~ t,  # Only Player Defects
    player_coop == 0 & opponent_coop == 0 ~ p,  # Both Defect
    TRUE ~ NA_real_                             # Default value for any unspecified cases
  ))
```

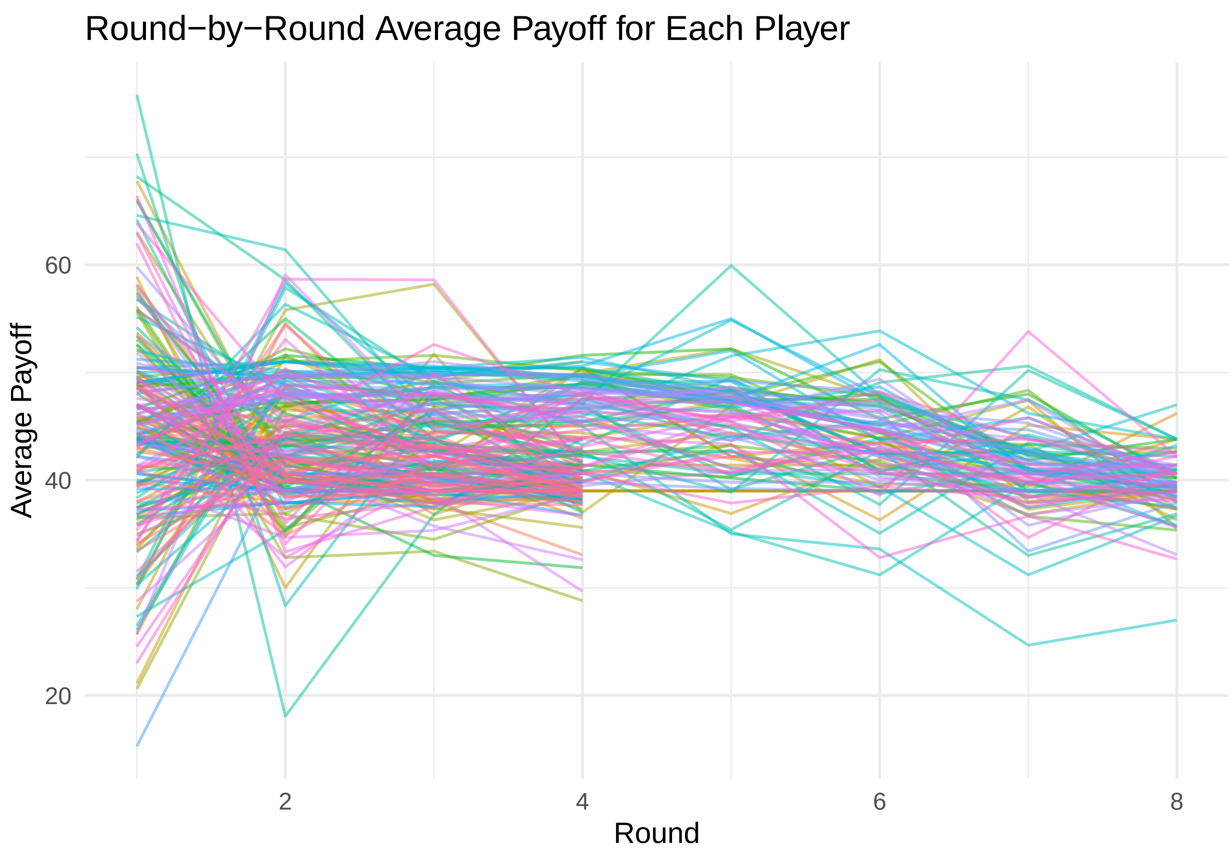
3.2 Round-by-Round Average Payoff Calculation

- **Objective:** Calculate the round-by-round average payoff for each player and visualize changes in performance over each round.

```r
round_avg_payoff <- df_merged %>%
  group_by(player_id, round) %>%
  summarize(avg_round_payoff = mean(payoff, na.rm = TRUE), .groups = "drop")

print(round_avg_payoff)
```

```
## # A tibble: 1,248 x 3
##    player_id round avg_round_payoff
##        <int> <int>            <dbl>
##  1         1     1             49.0
##  2         1     2             40.0
##  3         1     3             38.2
##  4         1     4             38.2
##  5         2     1             44.4
##  6         2     2             43.6
##  7         2     3             44.2
##  8         2     4             38.5
##  9         3     1             43.9
## 10         3     2             40.6
## # i 1,238 more rows
```

```r
# Visualization: Round-by-Round Payoff Distribution for Each Player with Grouping and Color
ggplot(round_avg_payoff, aes(x = round, y = avg_round_payoff, color = factor(player_id), group = player_
  geom_line(alpha = 0.5) +
  labs(title = "Round-by-Round Average Payoff for Each Player", x = "Round", y = "Average Payoff") +
  theme_minimal() +
  theme(legend.position = "none")
```



3.3 Average Payoff and Variance Calculation per Player

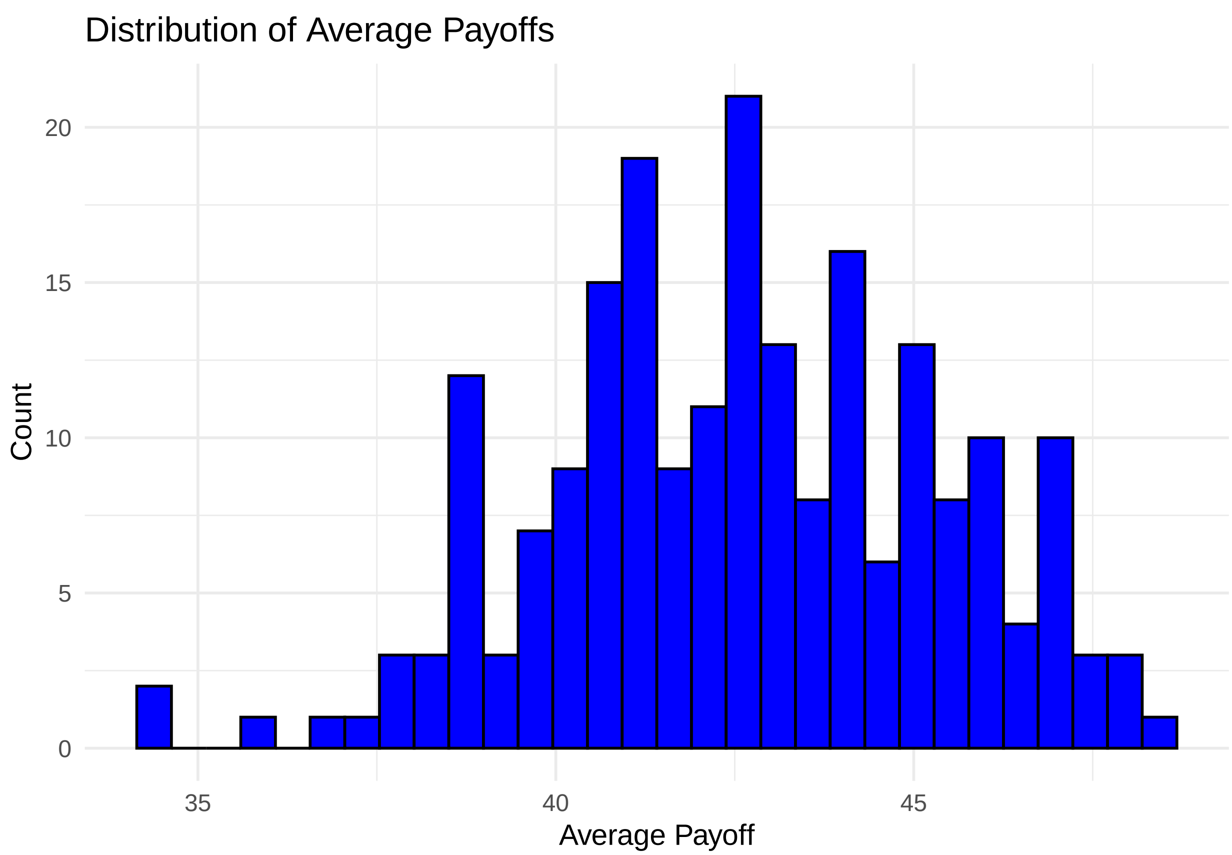
- **Objective:** Calculate the overall average payoff and payoff variance for each player to understand the distribution of player performance.

```
avg_payoff_variance <- df_merged %>%
  group_by(player_id) %>%
  summarize(
    avg_payoff = mean(payoff, na.rm = TRUE),
    payoff_variance = var(payoff, na.rm = TRUE)
  )

print(avg_payoff_variance)
```

```
## # A tibble: 212 x 3
##    player_id avg_payoff payoff_variance
##        <int>      <dbl>           <dbl>
##  1         1       41.3           104.
##  2         2       42.7           145.
##  3         3       41.0            94.4
##  4         4       42.2           105.
##  5         5       37.9           150.
##  6         6       43.4           115.
##  7         7       42.9            79.4
##  8         8       40.4            85.5
```

```
## 9          9       41.8            68.7
## 10        10       42.6            74.4
## # i 202 more rows
```

```
# Visualization: Distribution of Average Payoffs
ggplot(avg_payoff_variance, aes(x = avg_payoff)) +
  geom_histogram(bins = 30, fill = "blue", color = "black") +
  labs(title = "Distribution of Average Payoffs", x = "Average Payoff", y = "Count") +
  theme_minimal()
```



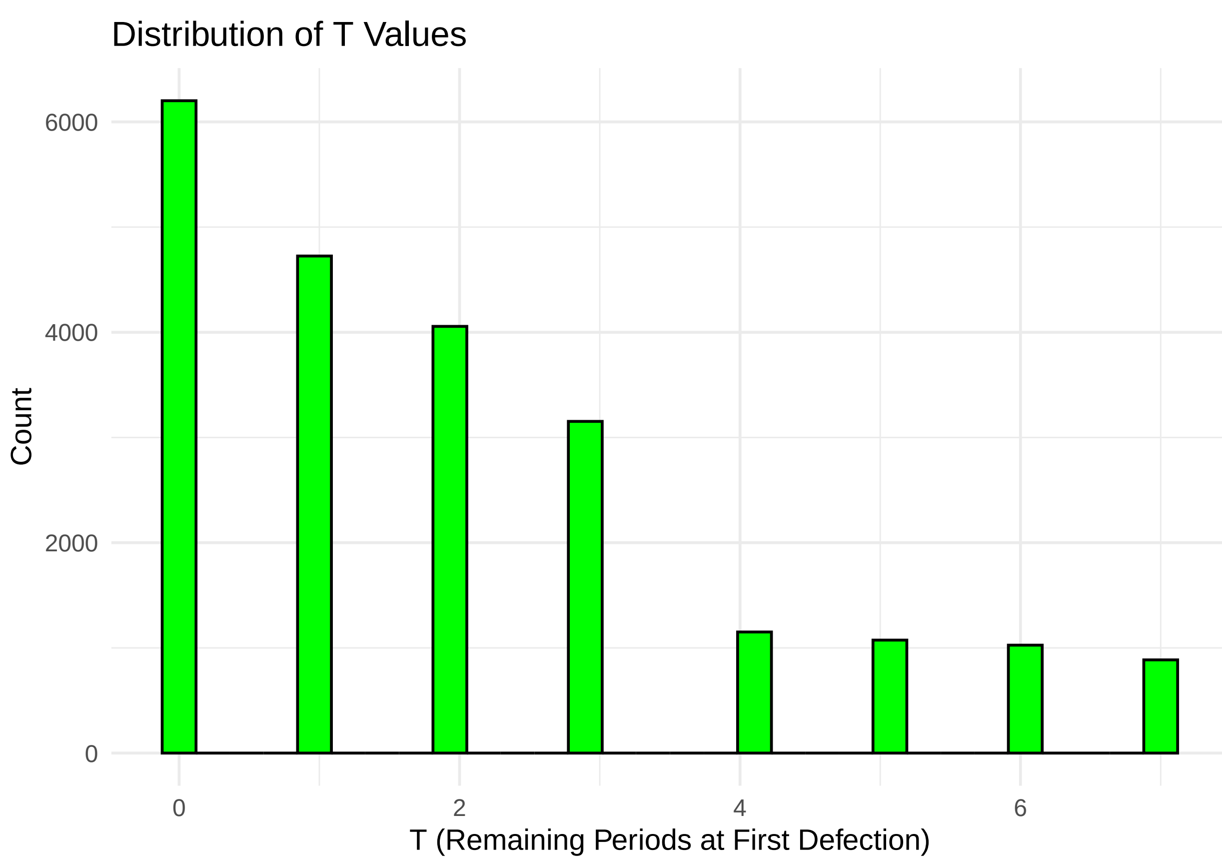
3.4 'T' Value Calculation and Distribution Visualization

- **Objective:** Calculate T value (the number of remaining rounds when a player first defects) and visualize the distribution of T values.

```
df_merged <- df_merged %>%
  group_by(player_id, supergame) %>%
  mutate(
    first_defect_round = ifelse(player_coop == 0 & !is.na(player_coop), round, NA),
    T = ifelse(!is.na(first_defect_round), horizon - first_defect_round, ifelse(all(player_coop == 1, na
  ) %>%
  ungroup()

print(df_merged %>% select(player_id, supergame, round, player_coop, horizon, T))
```

```
## # A tibble: 33,360 x 6
##    player_id supergame round player_coop horizon     T
##        <int>     <int> <int>       <int>   <int> <dbl>
##  1        73         1     1           0       8     7
##  2        73         2     1           0       8     7
##  3        73         3     1           0       8     7
##  4        73         4     1           0       8     7
##  5        73         5     1           1       8     0
##  6        73         6     1           1       8    NA
##  7        73         7     1           0       8     7
##  8        73         8     1           0       8     7
##  9        73         9     1           1       8    NA
## 10        73        10     1           1       8    NA
## # i 33,350 more rows
```

```r
# Visualization: Distribution of T Values
ggplot(df_merged %>% filter(!is.na(T)), aes(x = T)) +
  geom_histogram(bins = 30, fill = "green", color = "black") +
  labs(title = "Distribution of T Values", x = "T (Remaining Periods at First Defection)", y = "Count")
  theme_minimal()
```



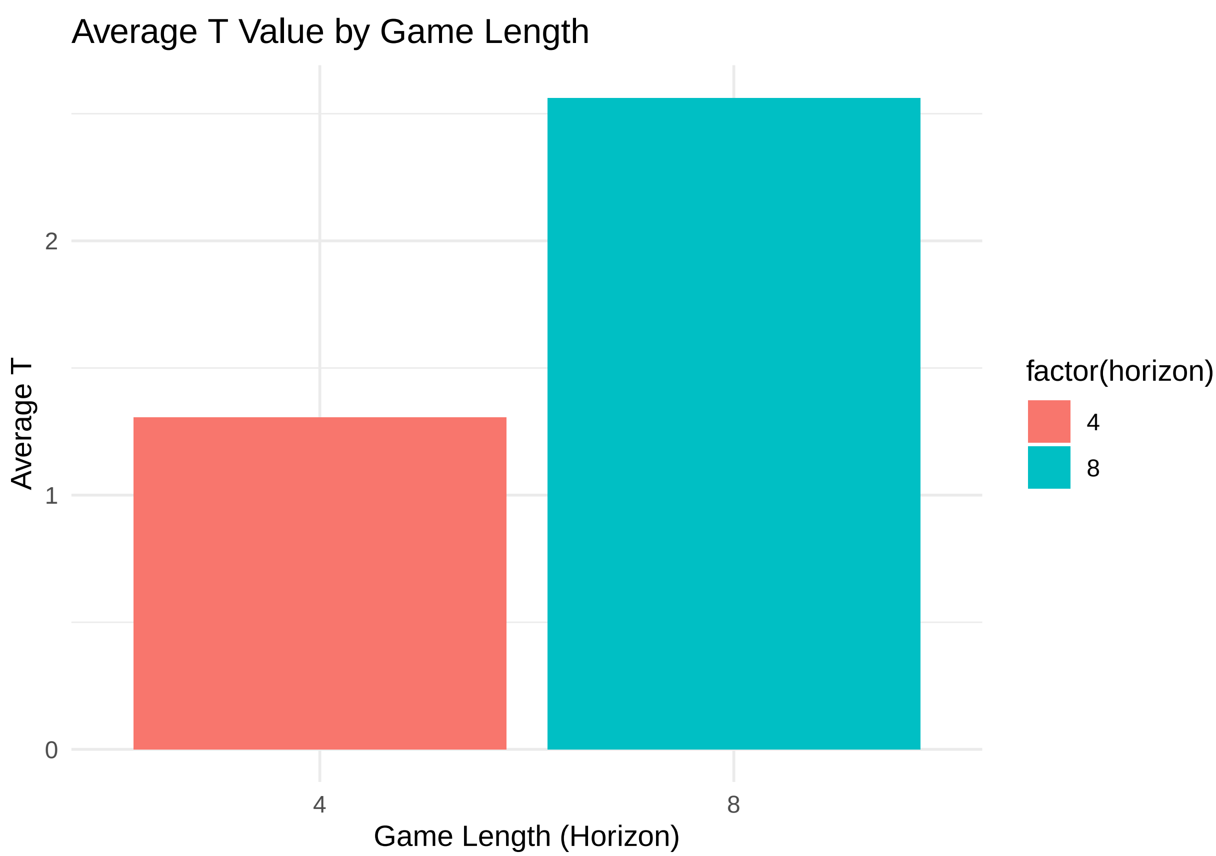
3.5 Cross-Tabulation of T Values by Game Length

- **Objective:** Summarize T values by game length (horizon) to show mean and standard deviation of T for each horizon and visualize these statistics.

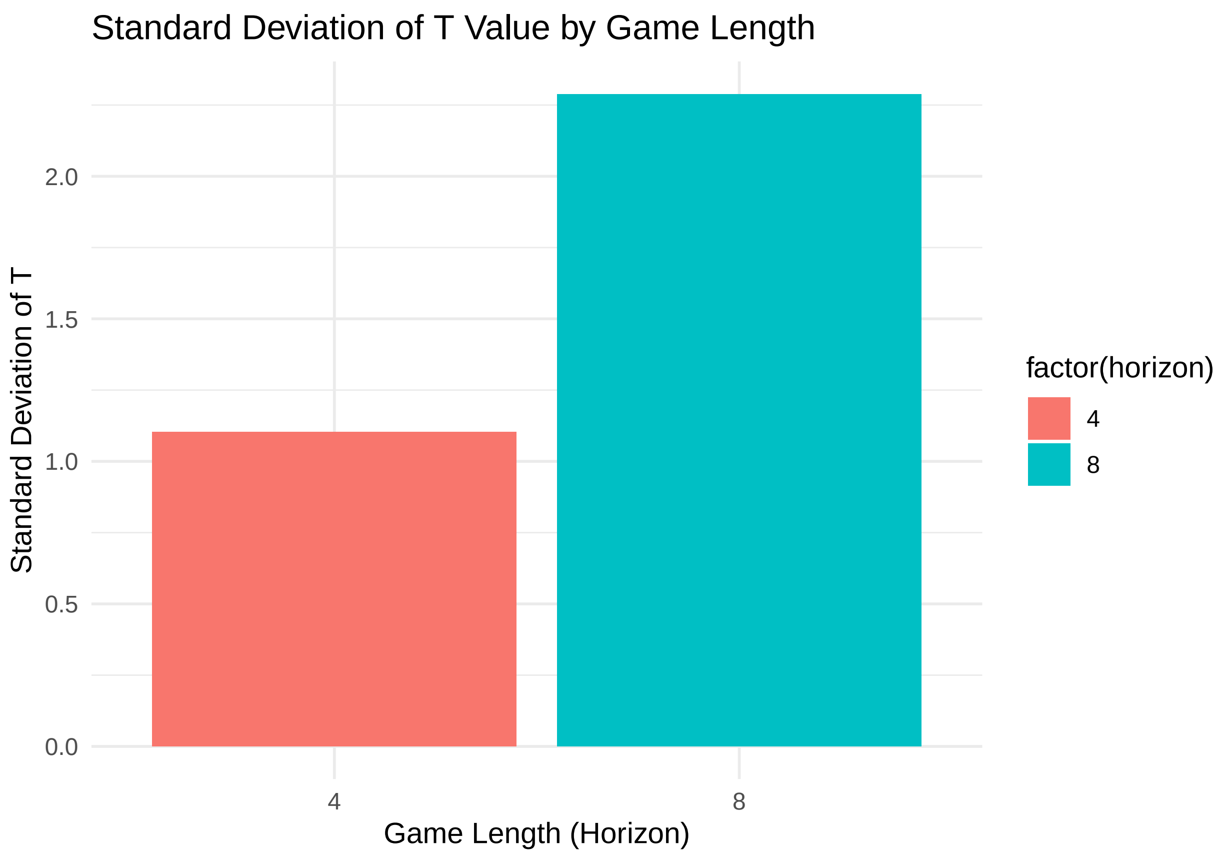
```r
T_summary <- df_merged %>%
  group_by(horizon) %>%
  summarise(
    mean_T = mean(T, na.rm = TRUE),
    sd_T = sd(T, na.rm = TRUE),
    count = n()
  )

print(T_summary)
```

```
## # A tibble: 2 x 4
##   horizon mean_T  sd_T count
##     <int>  <dbl> <dbl> <int>
## 1       4   1.31  1.10 11920
## 2       8   2.56  2.29 21440
```

```r
### Visualization:

# Average T Value by Game Length Visualization
ggplot(T_summary, aes(x = factor(horizon), y = mean_T, fill = factor(horizon))) +
  geom_bar(stat = "identity") +
  labs(title = "Average T Value by Game Length", x = "Game Length (Horizon)", y = "Average T") +
  theme_minimal()
```



```r
# Standard Deviation of T Value by Game Length Visualization
ggplot(T_summary, aes(x = factor(horizon), y = sd_T, fill = factor(horizon))) +
  geom_bar(stat = "identity") +
  labs(title = "Standard Deviation of T Value by Game Length", x = "Game Length (Horizon)", y = "Standa
  theme_minimal()
```



Step 4: Strategy Identification

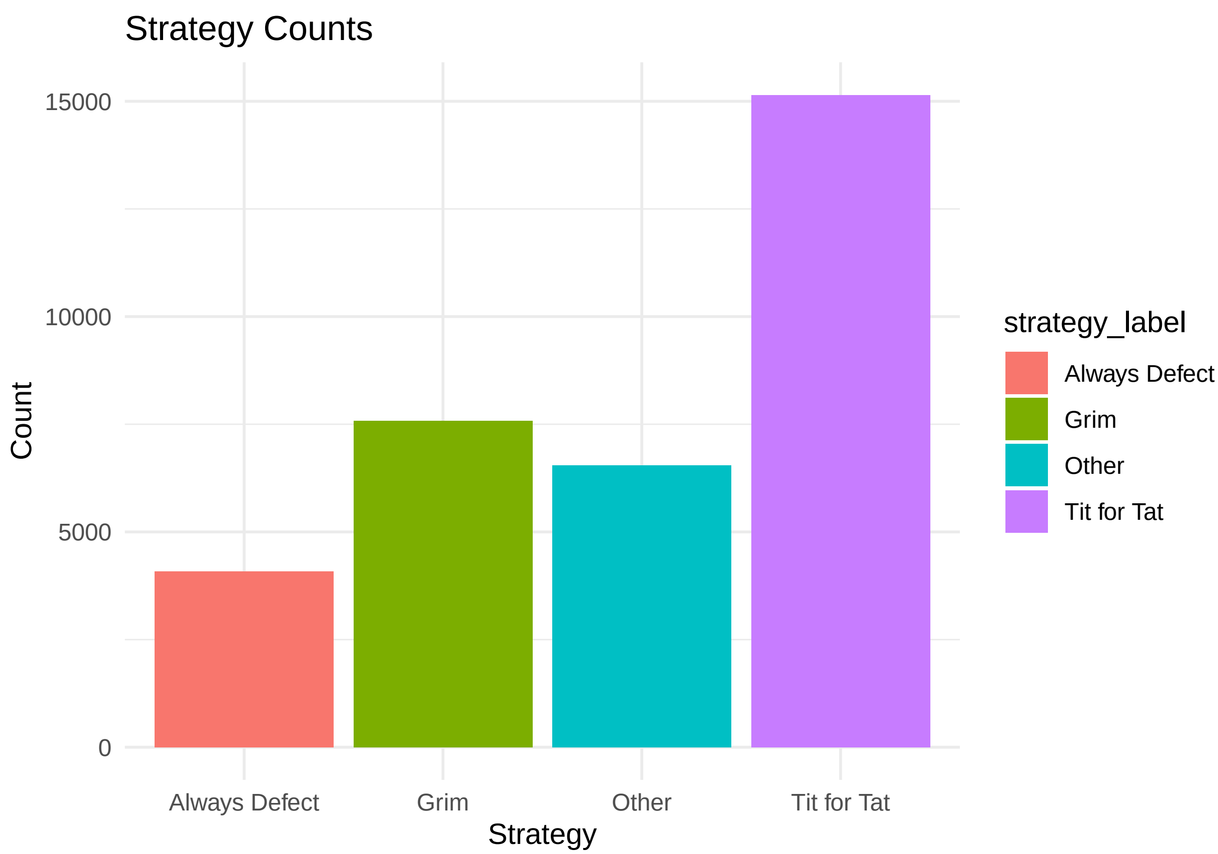
- **Objective:** Define a function to classify player strategies, and then merge the identified strategies with the main dataset and calculate the average payoff and payoff variance for each strategy.

```r
# Load ggplot2 library for visualization
library(ggplot2)

### 4.1 Define Strategy Identification Function
identify_strategy <- function(player_coop, opponent_coop) {
  # Tit for Tat
  tit_for_tat <- if (length(player_coop) > 1) all(player_coop[-1] == lag(opponent_coop)[-1], na.rm = TRU
  # Grim Strategy
  grim <- all((player_coop == 1) | (cumsum(opponent_coop == 0) > 0), na.rm = TRUE)
  # Always Cooperate
  always_cooperate <- all(player_coop == 1)
  # Always Defect
  always_defect <- all(player_coop == 0)
```

```r
  ## Determine Strategy
  if (tit_for_tat) return("Tit for Tat")
  else if (grim) return("Grim")
  else if (always_cooperate) return("Always Cooperate")
  else if (always_defect) return("Always Defect")
  else return("Other")
}

# Apply Strategy Identification to df_merged
df_merged <- df_merged %>%
  group_by(player_id, supergame) %>%
  mutate(strategy_label = identify_strategy(player_coop, opponent_coop)) %>%
  ungroup()

# Group by strategy_label and count
strategy_counts <- df_merged %>%
  group_by(strategy_label) %>%
  summarize(count = n())

print(strategy_counts)
```

```
## # A tibble: 4 x 2
##   strategy_label count
##   <chr>          <int>
## 1 Always Defect   4088
## 2 Grim            7576
## 3 Other           6548
## 4 Tit for Tat    15148
```

```r
#==================================================================================#

# Visualize Strategy Counts
ggplot(strategy_counts, aes(x = strategy_label, y = count, fill = strategy_label)) +
  geom_bar(stat = "identity") +
  labs(title = "Strategy Counts", x = "Strategy", y = "Count") +
  theme_minimal()
```



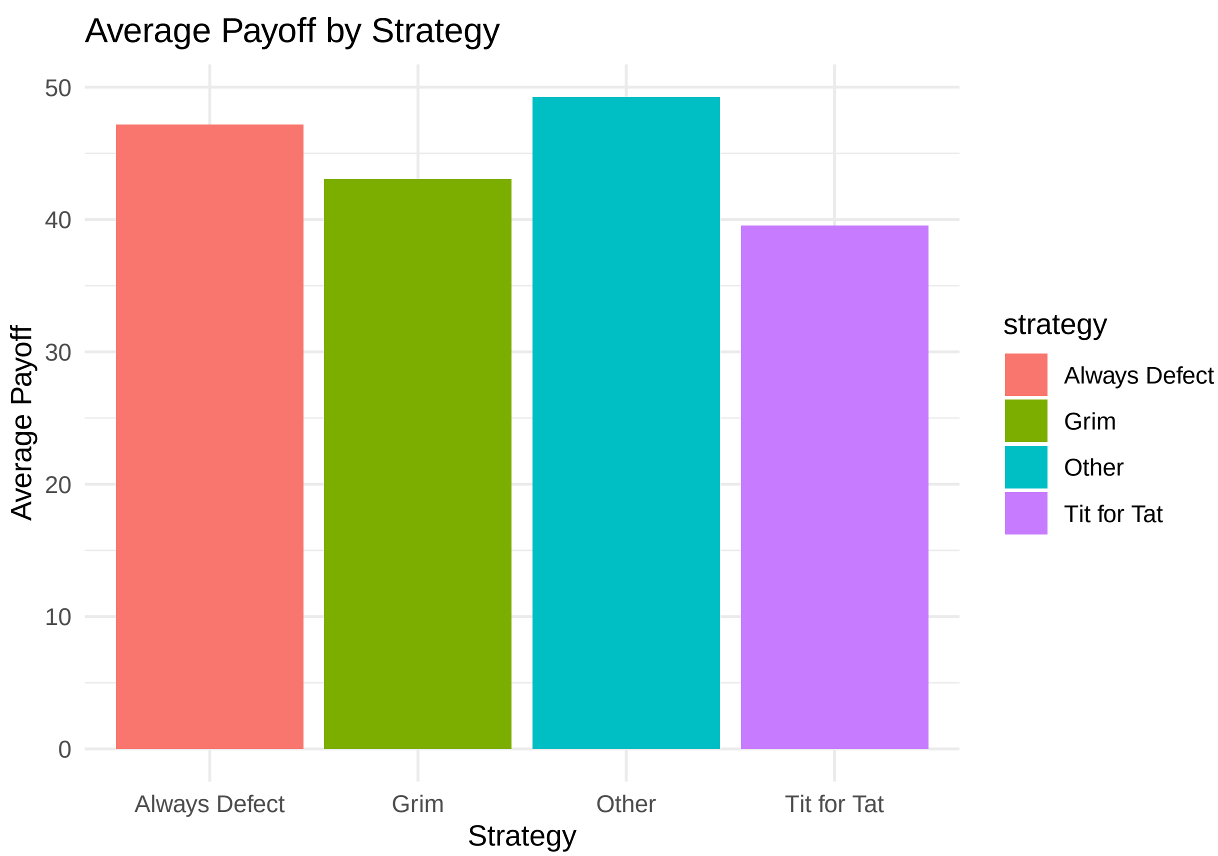
Step 5: Strategy Payoff Analysis

- **Objective:** Merge the identified strategies with the main dataset and calculate the average payoff and payoff variance for each strategy.

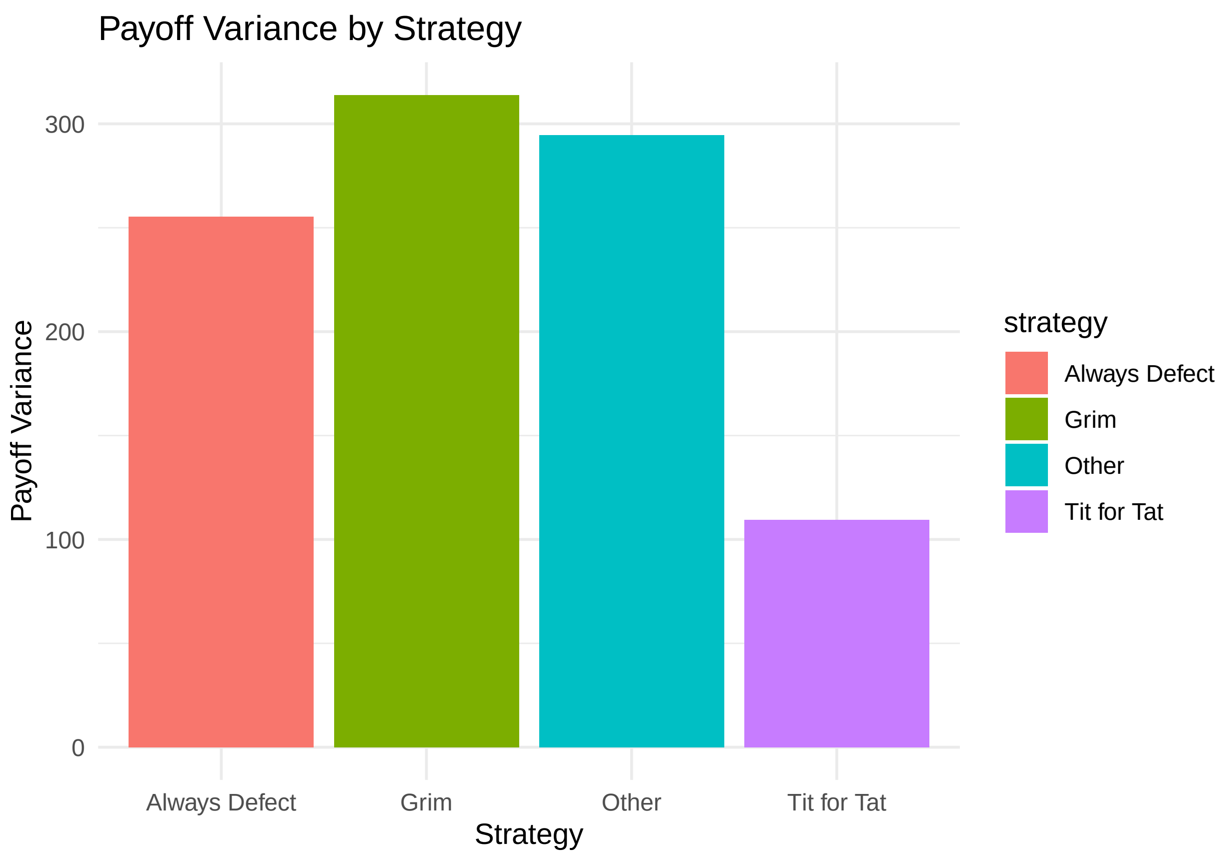
```
### 5.1 Apply Strategy Identification to Each Player and Supergame
df_strategies <- df_merged %>%
  group_by(player_id, supergame) %>%
  summarise(strategy = identify_strategy(player_coop, opponent_coop), .groups = 'drop')

### 5.2 Merge Identified Strategies with Original Data
df_merged <- df_merged %>%
  left_join(df_strategies, by = c("player_id", "supergame"))

### 5.3 Calculate Average Payoff by Strategy
strategy_payoff_summary <- df_merged %>%
  group_by(strategy) %>%
  summarise(
    avg_strategy_payoff = mean(payoff, na.rm = TRUE),
    payoff_variance = var(payoff, na.rm = TRUE),
    count = n()
  )

# View Results
print(strategy_payoff_summary)
```

```
## # A tibble: 4 x 4
##   strategy      avg_strategy_payoff payoff_variance count
##   <chr>                       <dbl>           <dbl> <int>
## 1 Always Defect                47.2            255.  4088
## 2 Grim                         43.0            314.  7576
## 3 Other                        49.3            295.  6548
## 4 Tit for Tat                  39.5            109. 15148
```

```r
#=================================================================================#

## Visualize Average Payoff by Strategy & Payoff Variance by Strategy

# Average Payoff by Strategy
ggplot(strategy_payoff_summary, aes(x = strategy, y = avg_strategy_payoff, fill = strategy)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Payoff by Strategy", x = "Strategy", y = "Average Payoff") +
  theme_minimal()
```




```r
# Payoff Variance by Strategy
ggplot(strategy_payoff_summary, aes(x = strategy, y = payoff_variance, fill = strategy)) +
  geom_bar(stat = "identity") +
  labs(title = "Payoff Variance by Strategy", x = "Strategy", y = "Payoff Variance") +
  theme_minimal()
```



Extended Analysis

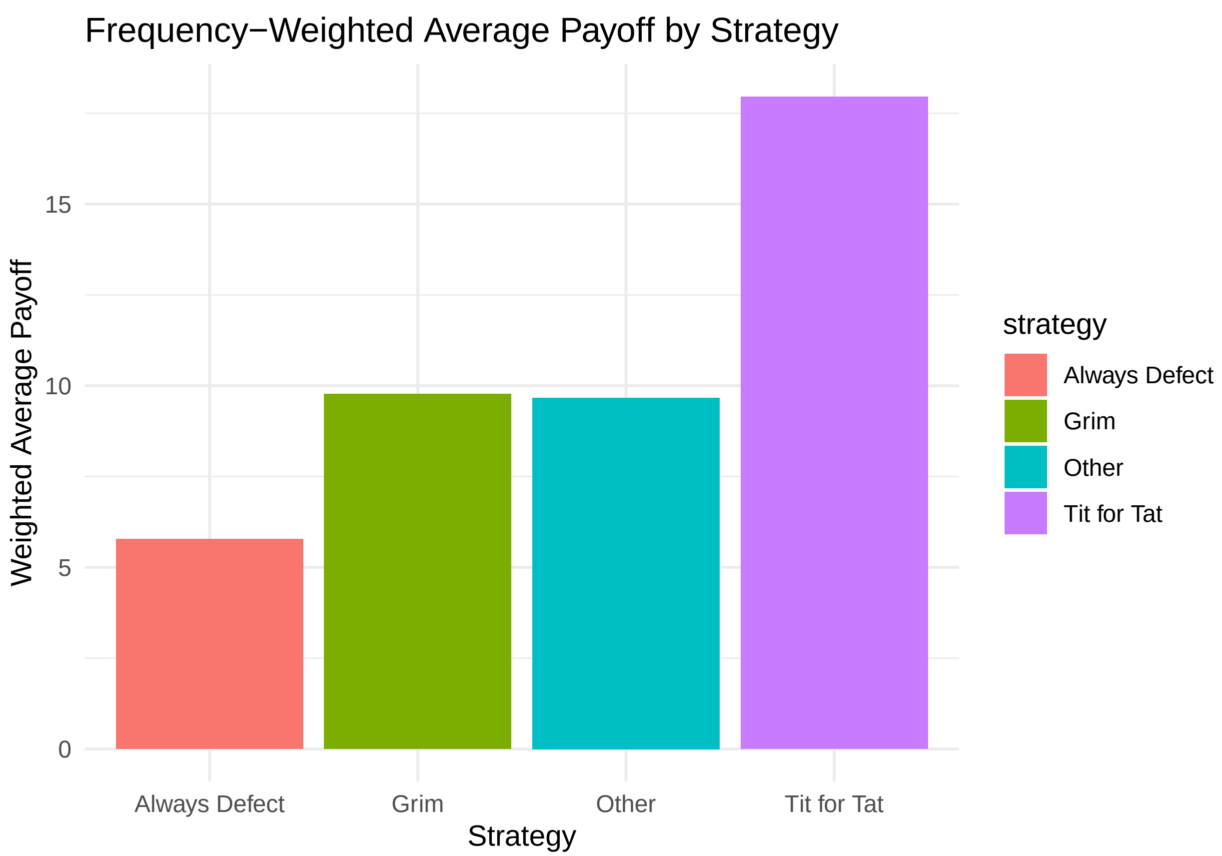
Step 6: Frequency-Weighted Payoff Calculation

- **Objective:** This calculation weights each strategy's average payoff by its frequency, providing a clearer view of strategy effectiveness.

```
### 6 Frequency-Weighted Average Payoff Calculation
weighted_payoff_summary <- strategy_payoff_summary %>%
  mutate(weighted_avg_payoff = avg_strategy_payoff * (count / sum(count)))

# Print Weighted Results
print(weighted_payoff_summary)
```

```
## # A tibble: 4 x 5
##   strategy      avg_strategy_payoff payoff_variance count weighted_avg_payoff
##   <chr>                       <dbl>           <dbl> <int>               <dbl>
## 1 Always Defect                47.2            255.  4088                5.78
## 2 Grim                         43.0            314.  7576                9.78
## 3 Other                        49.3            295.  6548                9.67
## 4 Tit for Tat                  39.5            109. 15148               18.0
```

```r
# Visualization
ggplot(weighted_payoff_summary, aes(x = strategy, y = weighted_avg_payoff, fill = strategy)) +
  geom_bar(stat = "identity") +
  labs(title = "Frequency-Weighted Average Payoff by Strategy", x = "Strategy", y = "Weighted Average Pa
  theme_minimal()
```



Step 7: Performance Against Non-Fixed Strategies

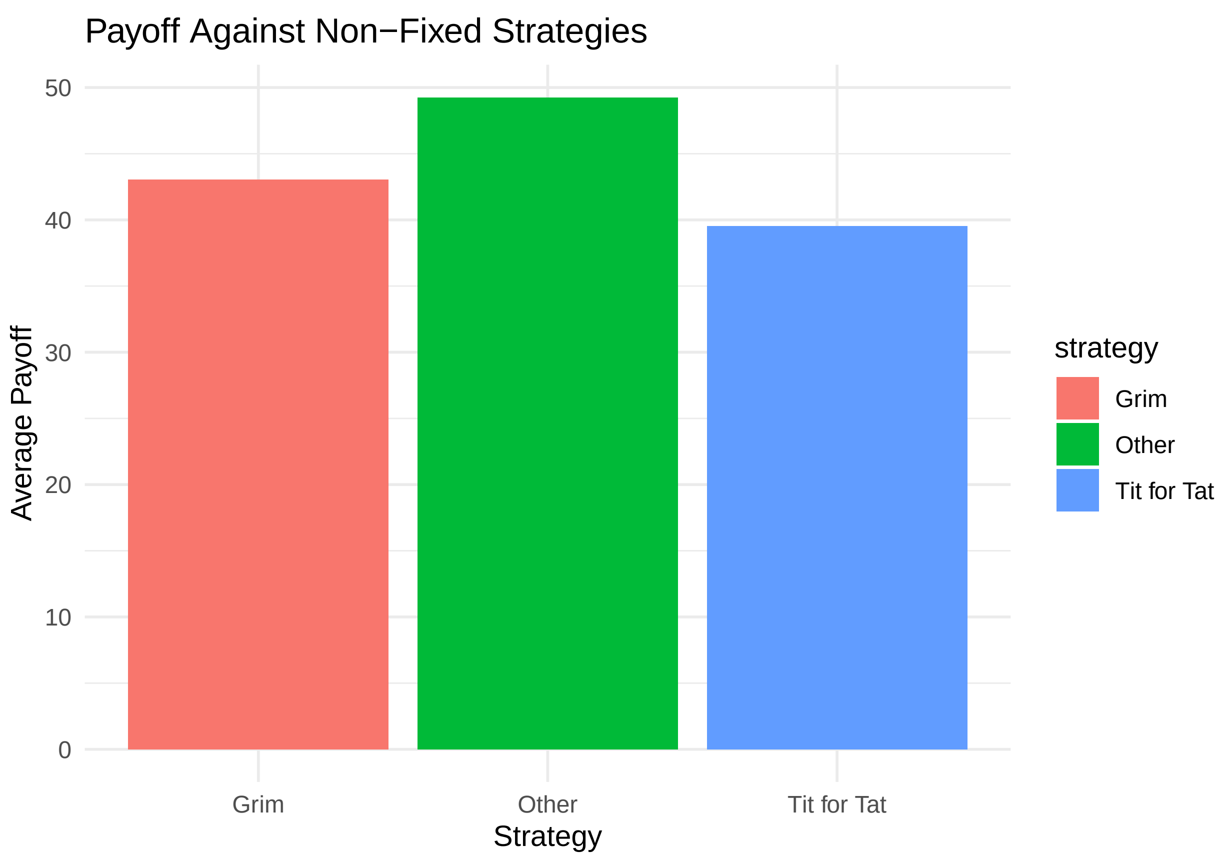
- **Objective:** This analysis compares each fixed strategy's performance when interacting with non-fixed strategies, highlighting adaptability.

```r
### 7 Payoff Analysis Against Non-Fixed Strategies
non_fixed_performance <- df_merged %>%
  filter(strategy != "Always Cooperate" & strategy != "Always Defect") %>%
  group_by(strategy) %>%
  summarise(
    avg_payoff_against_non_fixed = mean(payoff, na.rm = TRUE),
    variance_against_non_fixed = var(payoff, na.rm = TRUE)
  )

# Print Non-Fixed Performance Results
print(non_fixed_performance)
```

```
## # A tibble: 3 x 3
```

```
##   strategy    avg_payoff_against_non_fixed variance_against_non_fixed
##   <chr>                              <dbl>                      <dbl>
## 1 Grim                                43.0                       314.
## 2 Other                               49.3                       295.
## 3 Tit for Tat                         39.5                       109.
```

```r
# Visualization
ggplot(non_fixed_performance, aes(x = strategy, y = avg_payoff_against_non_fixed, fill = strategy)) +
  geom_bar(stat = "identity") +
  labs(title = "Payoff Against Non-Fixed Strategies", x = "Strategy", y = "Average Payoff") +
  theme_minimal()
```



Step 8: Expected Payoff Simulation for Hypothetical Tit-for-Tat Player

- **Objective:** In this step, we estimate the expected payoff for a hypothetical Tit-for-Tat player when competing against various other strategies. This analysis provides insights into how a Tit-for-Tat strategy might perform on average against other identified strategies.

```r
### 8 Simulating Tit-for-Tat Performance

# Calculating Expected Payoff for Tit-for-Tat vs Opponent Strategies
hypothetical_tft_performance <- df_merged %>%
  filter(strategy == "Tit for Tat") %>%
  group_by(opponent_strategy = strategy) %>%
```

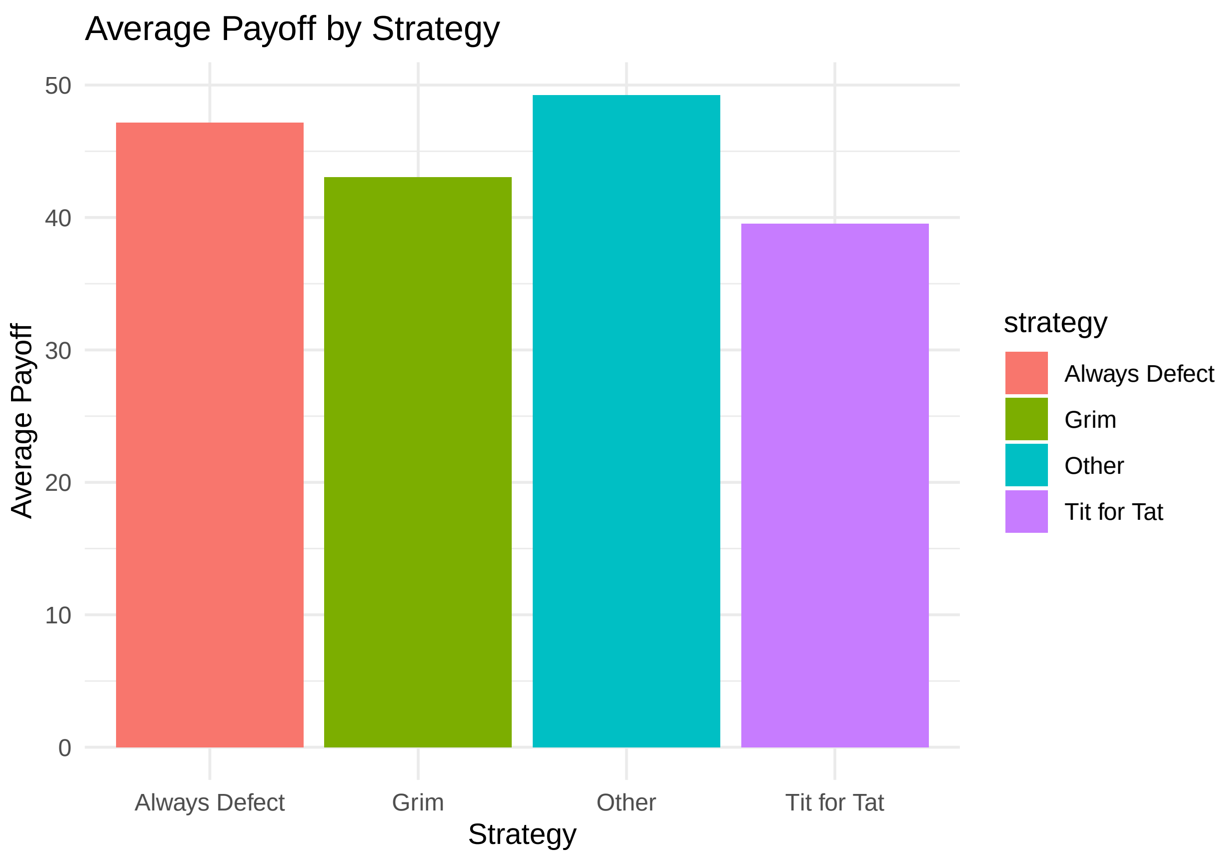
```r
  summarise(
    avg_expected_payoff = mean(payoff, na.rm = TRUE),
    variance_expected_payoff = var(payoff, na.rm = TRUE),
    count = n()
  )

# View Results as a Table
print(hypothetical_tft_performance)
```

```
## # A tibble: 1 x 4
##   opponent_strategy avg_expected_payoff variance_expected_payoff count
##   <chr>                           <dbl>                    <dbl> <int>
## 1 Tit for Tat                      39.5                     109. 15148
```

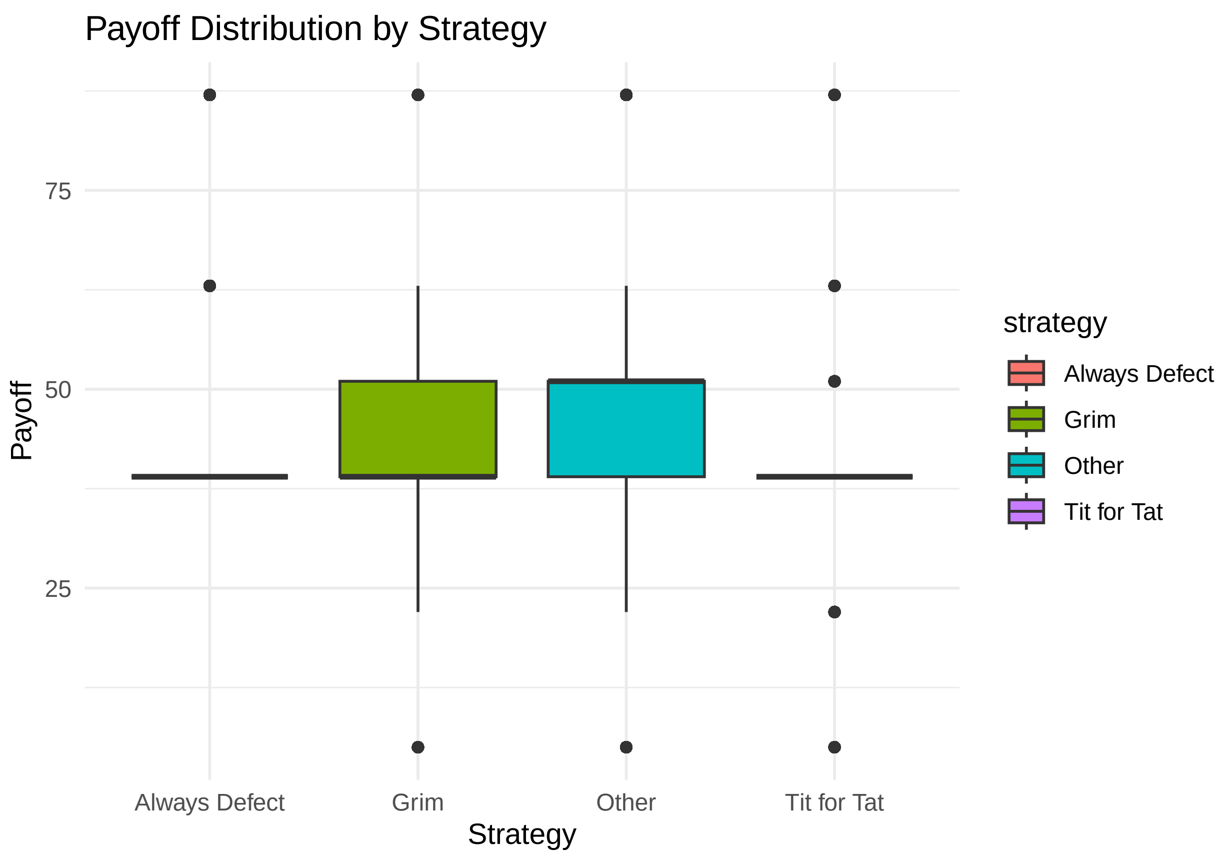
Step 9: Visualizations and Graphical Analysis

- **Objective:** Provides a visualization of strategy payoff distributions and strategy counts, offering a visual comparison of each strategy's performance and popularity.

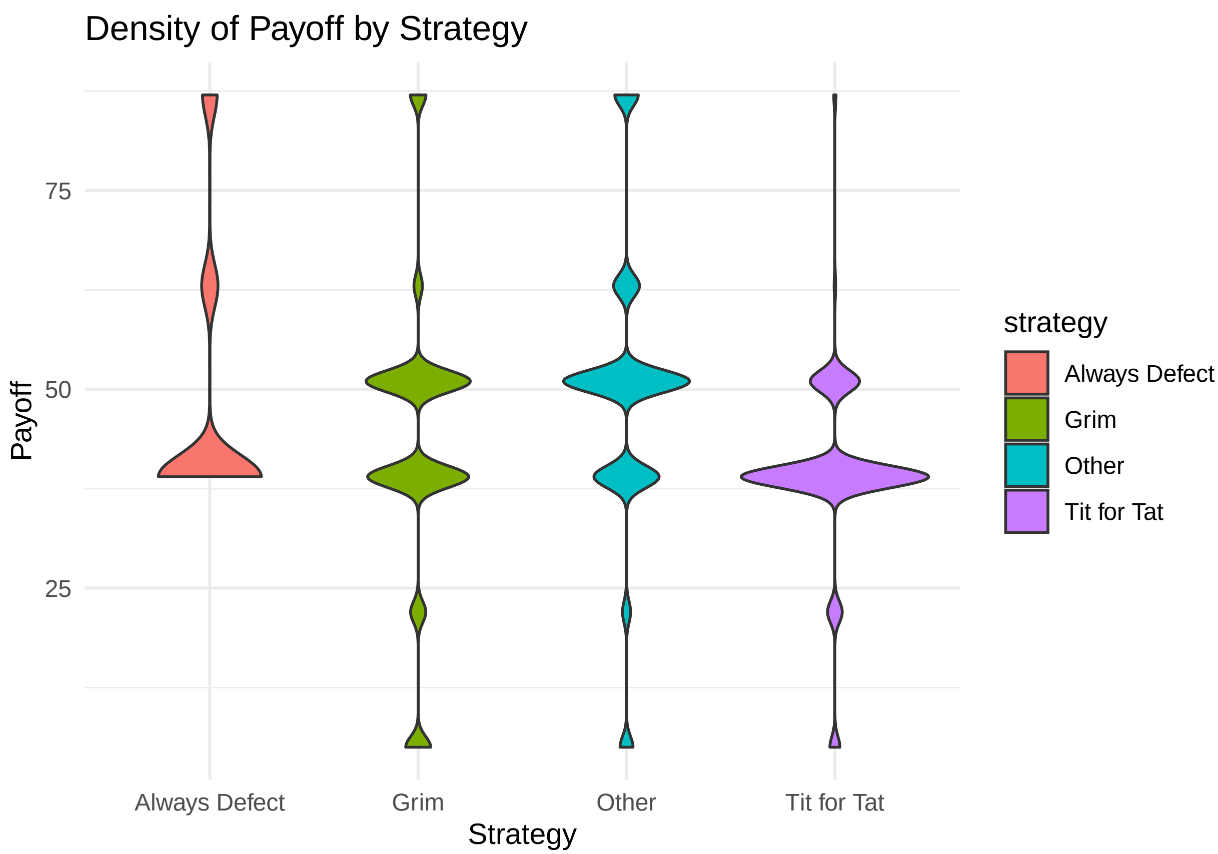
```r
### 9 Visualizations

library(ggplot2)
library(ggridges)

# 1. Bar Plot: Average Payoff by Strategy
# This bar plot shows the average payoff for each strategy, allowing a straightforward comparison of av
ggplot(strategy_payoff_summary, aes(x = strategy, y = avg_strategy_payoff, fill = strategy)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Payoff by Strategy", x = "Strategy", y = "Average Payoff") +
  theme_minimal()
```



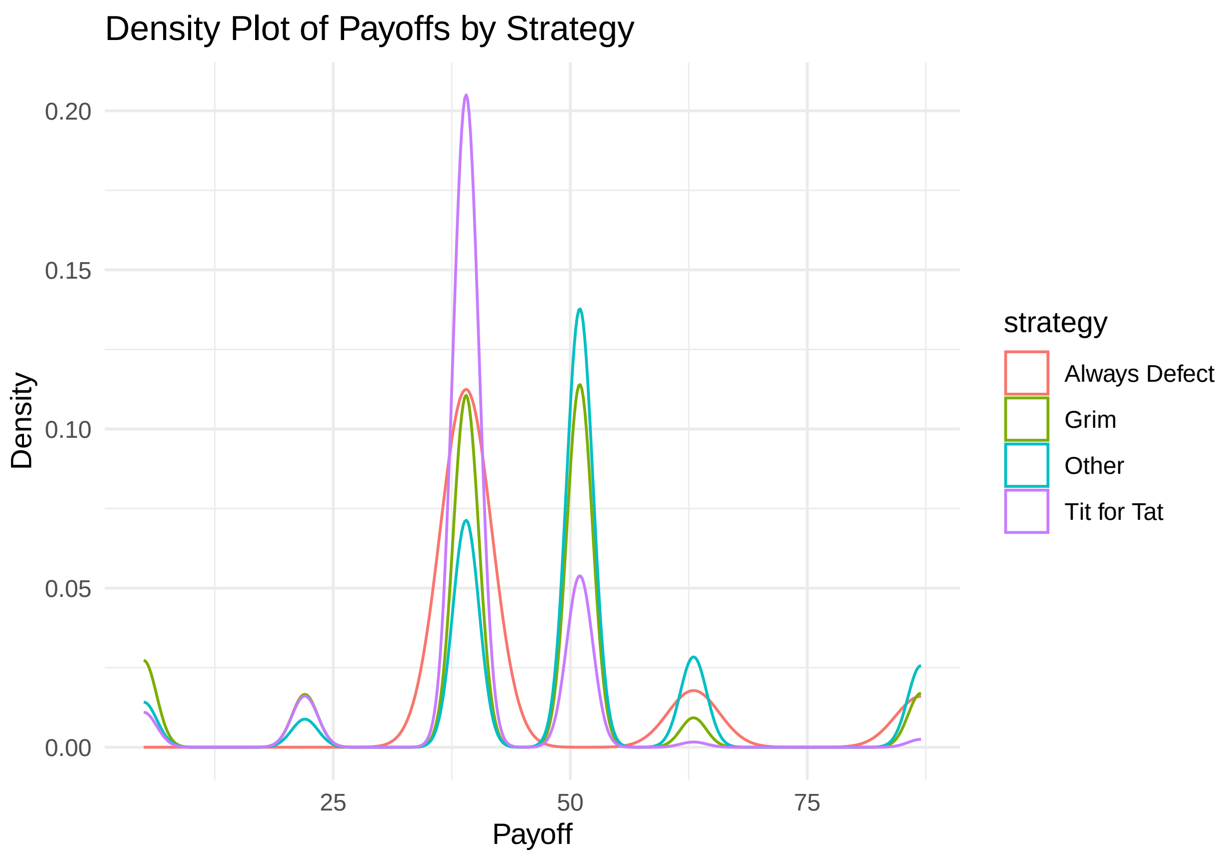

```r
# 2. Box Plot: Payoff Distribution by Strategy
# This box plot displays the distribution of payoffs for each strategy, highlighting median, quartiles,
ggplot(df_merged, aes(x = strategy, y = payoff, fill = strategy)) +
  geom_boxplot() +
  labs(title = "Payoff Distribution by Strategy", x = "Strategy", y = "Payoff") +
  theme_minimal()
```



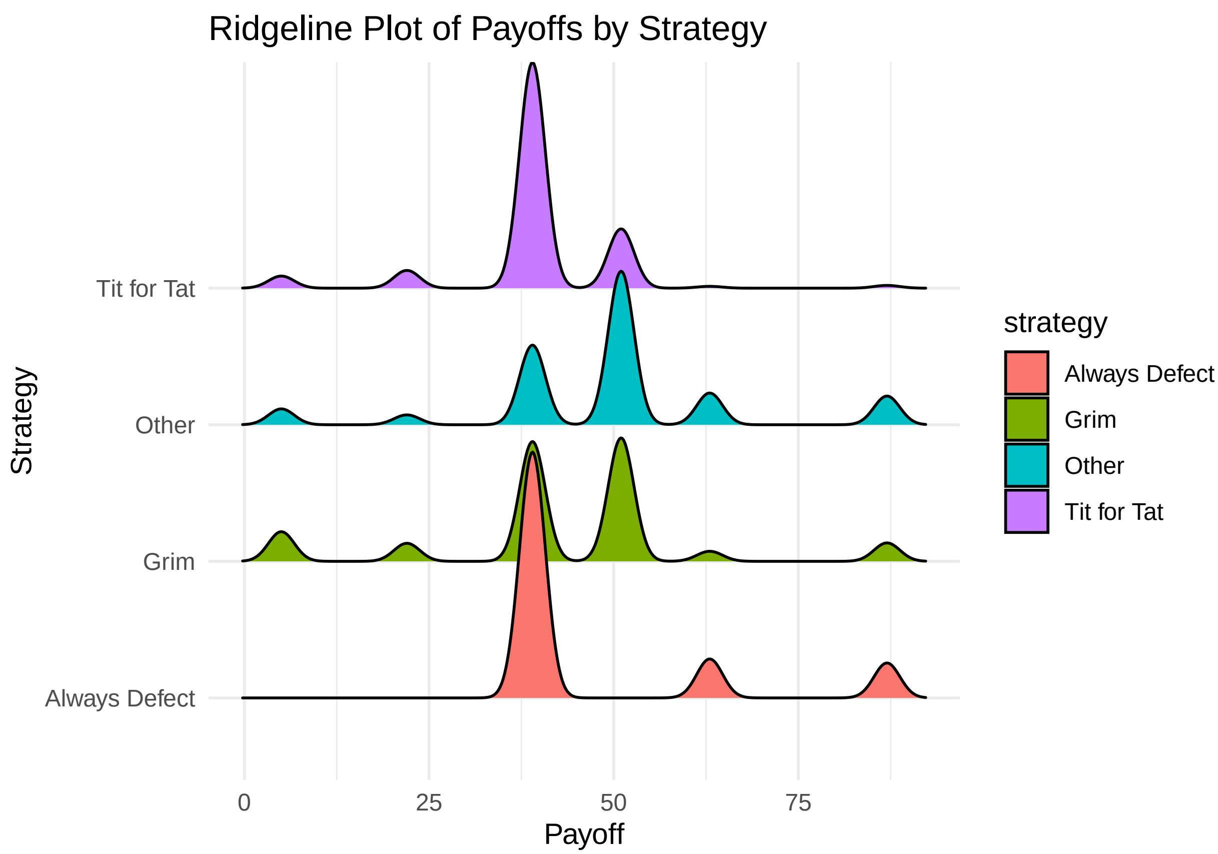

```r
# 3. Violin Plot: Density of Payoff by Strategy
# The violin plot shows the density distribution of payoffs for each strategy, combining box plot and d
ggplot(df_merged, aes(x = strategy, y = payoff, fill = strategy)) +
  geom_violin() +
  labs(title = "Density of Payoff by Strategy", x = "Strategy", y = "Payoff") +
  theme_minimal()
```



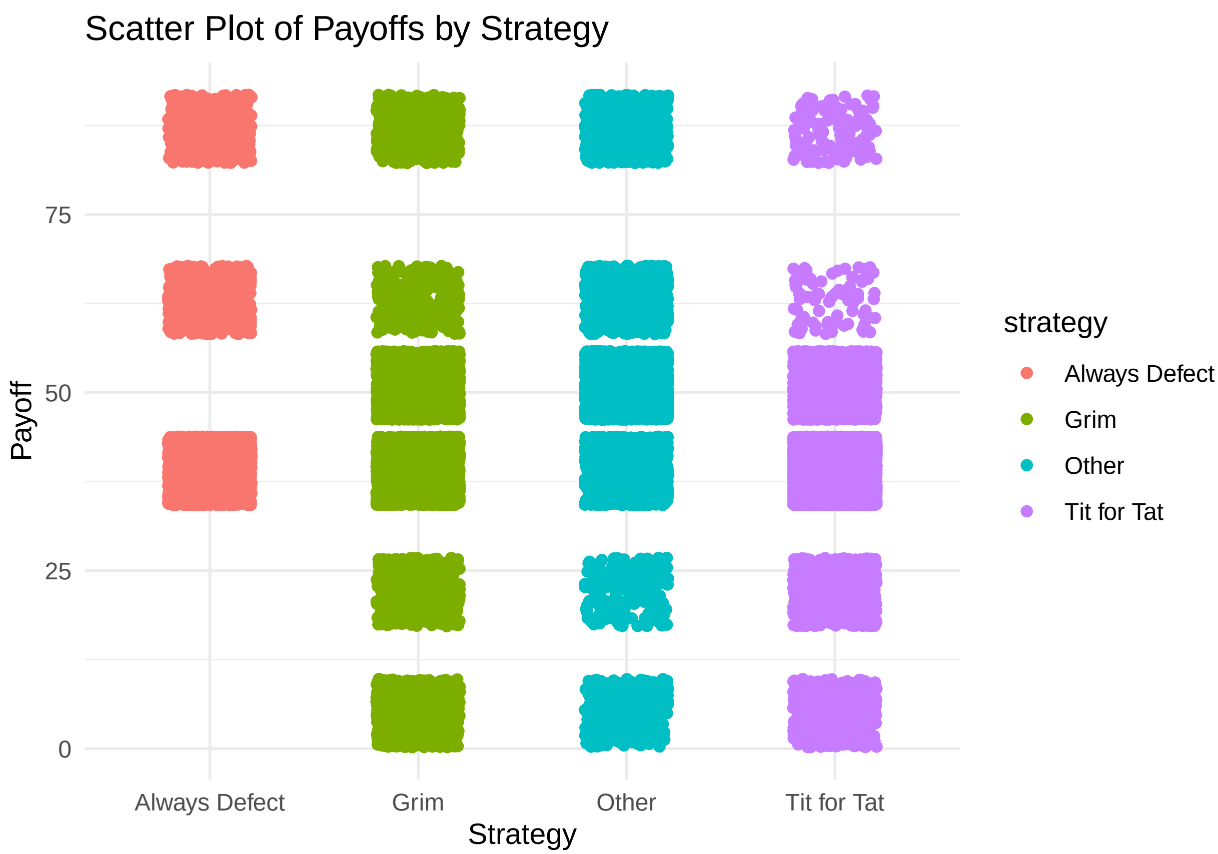

```
# 4. Density Plot: Density of Payoffs by Strategy
# This density plot presents the probability density of payoffs across strategies, allowing visual comp
ggplot(df_merged, aes(x = payoff, color = strategy)) +
  geom_density() +
  labs(title = "Density Plot of Payoffs by Strategy", x = "Payoff", y = "Density") +
  theme_minimal()
```




```
# 5. Ridgeline Plot: Payoffs by Strategy
# The ridgeline plot provides a layered view of payoff distributions by strategy, making it easier to c
ggplot(df_merged, aes(x = payoff, y = strategy, fill = strategy)) +
  geom_density_ridges(bandwidth = 1.75) +
  labs(title = "Ridgeline Plot of Payoffs by Strategy", x = "Payoff", y = "Strategy") +
  theme_minimal()
```




```r
# 6. Scatter Plot: Payoffs by Strategy
# This scatter plot shows individual payoff points for each strategy, adding some noise with width to r
ggplot(df_merged, aes(x = strategy, y = payoff, color = strategy)) +
  geom_jitter(width = 0.2) +
  labs(title = "Scatter Plot of Payoffs by Strategy", x = "Strategy", y = "Payoff") +
  theme_minimal()
```


Scatter Plot of Payoffs by Strategy
Payoff
75
50
25
0
Always Defect
Grim
Other
Tit for Tat
Strategy
strategy
Always Defect
Grim
Other
Tit for Tat