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

Hana Kwon, WB Macleod

Nov 22024

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

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.

```
### 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
```

```
library(readr)
library(ggplot2)

file_data <- "../../Data/Embrey_2018a_new_data.txt"
data <- read.table(file_data, 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 ...
```

```
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, Preprocessing, and PD Difficulty Setting

- **Objective:** Prepare player and opponent data frames to align cooperation values and payoff values for each round. Classify games by PD difficulty (EasyPD or HardPD) based on payoff values.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

```
###Updated
# Set normalized payoff values
data <- data %>%
  mutate(
    r = 1.0,              # Reward for mutual cooperation
    s = -l,                 # Sucker's payoff
    t = 1.0 + g,            # Temptation payoff
    p = 0.0                 # Punishment for mutual defection
  )

# 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)

# Merge player and opponent data
df_merged <- df_self %>%
  left_join(df_opp, by = c("player_id", "opponent_id", "supergame", "round", "horizon"))


# Add PD difficulty column
df_merged <- df_merged %>%
  mutate(pd_difficulty = case_when(
    abs(t - 2) < 0.1 & abs(s + 1.41) < 0.1 ~ "EasyPD",
    abs(t - 4) < 0.1 & abs(s + 2.8) < 0.1 ~ "HardPD",
    TRUE ~ NA_character_
  ))
```

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.

```
#Updated

# Calculate the Payoff Column Based on Actions
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
  ))

# Calculate payoffs
#df_merged <- df_merged %>%
#  mutate(payoff = case_when(
#    player_coop == 1 & opponent_coop == 1 ~ r,
#    player_coop == 1 & opponent_coop == 0 ~ s,
#    player_coop == 0 & opponent_coop == 1 ~ t,
#    player_coop == 0 & opponent_coop == 0 ~ p,
#    TRUE ~ NA_real_
#  ))

# Verify that the 'payoff' column exists
if (!"payoff" %in% colnames(df_merged)) {
  stop("Error: 'payoff' column was not created.")
} else {
  print("Payoff column created successfully.")
}
```

```
## [1] "Payoff column created successfully."
```

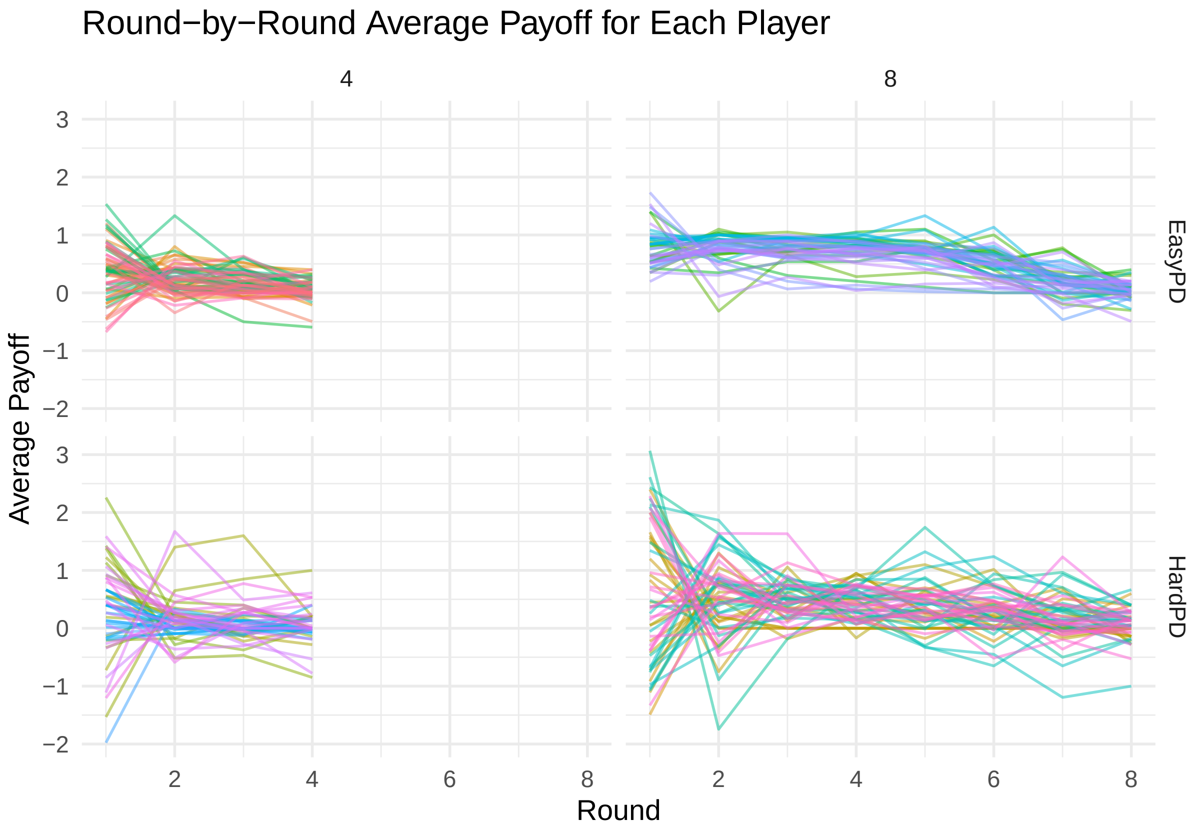
3.2 Round-by-Round Average Payoff Calculation

- **Objective:** Calculate the round-by-round average payoff for each player and visualize changes in performance over each round.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

```
# 3.2 Round-by-Round Average Payoff Calculation with 2x2 Design

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

print(round_avg_payoff)
```

```
## # A tibble: 1,248 x 5
##    player_id round horizon pd_difficulty avg_round_payoff
##        <int> <int>   <int> <chr>                    <dbl>
##  1         1     1       4 EasyPD                  0.829
##  2         1     2       4 EasyPD                  0.0875
##  3         1     3       4 EasyPD                 -0.0708
##  4         1     4       4 EasyPD                 -0.0708
##  5         2     1       4 EasyPD                  0.446
##  6         2     2       4 EasyPD                  0.387
##  7         2     3       4 EasyPD                  0.437
##  8         2     4       4 EasyPD                 -0.0417
##  9         3     1       4 EasyPD                  0.408
## 10         3     2       4 EasyPD                  0.129
## # i 1,238 more rows
```

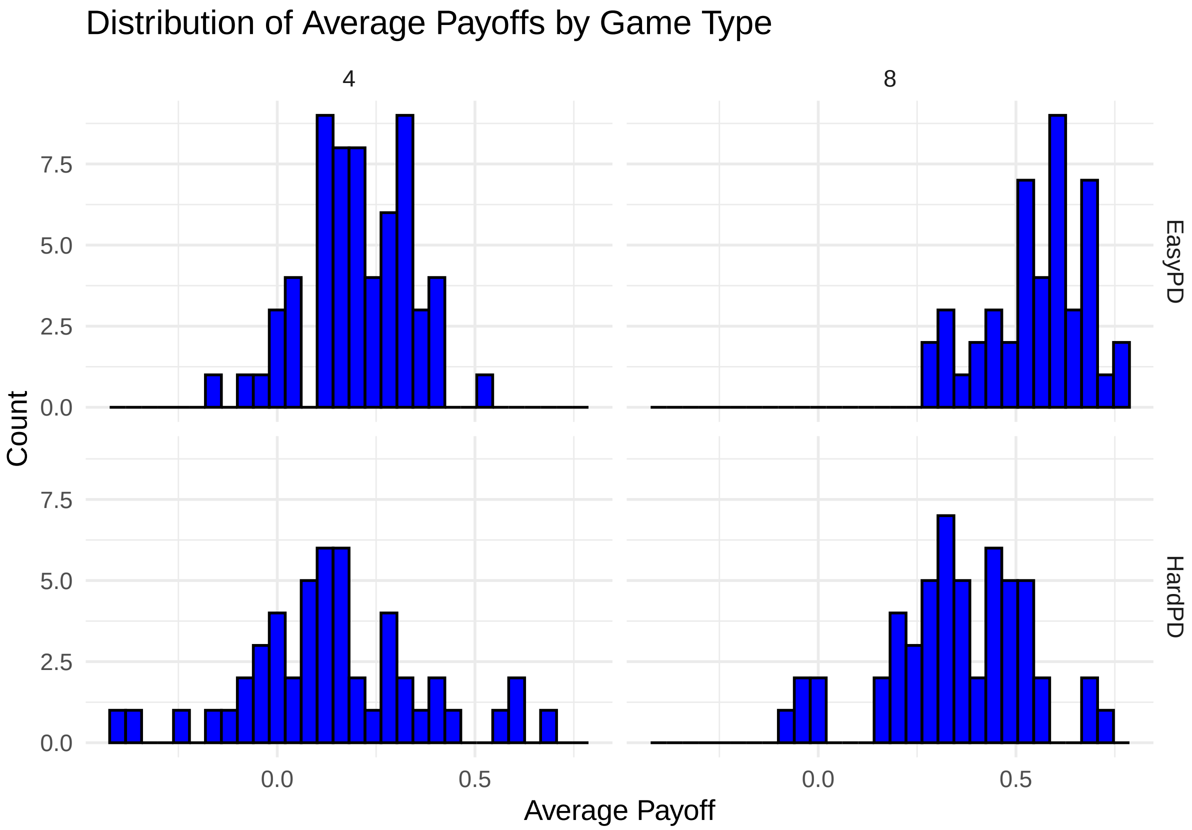
```
# Visualization: Round-by-Round Payoff Distribution for Each Player in 2x2 Design
ggplot(round_avg_payoff, aes(x = round, y = avg_round_payoff, color = factor(player_id), group = pl
  geom_line(alpha = 0.5) +
  labs(title = "Round-by-Round Average Payoff for Each Player", x = "Round", y = "Average Payoff")
  facet_grid(pd_difficulty ~ horizon) +  # 2x2 design: Difficulty by Game Length
  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.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

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

# Visualization with 2x2 Design
ggplot(avg_payoff_variance, aes(x = avg_payoff)) +
  geom_histogram(bins = 30, fill = "blue", color = "black") +
  facet_grid(pd_difficulty ~ horizon) +
  labs(title = "Distribution of Average Payoffs by Game Type", x = "Average Payoff", y = "Count") +
  theme_minimal()
```



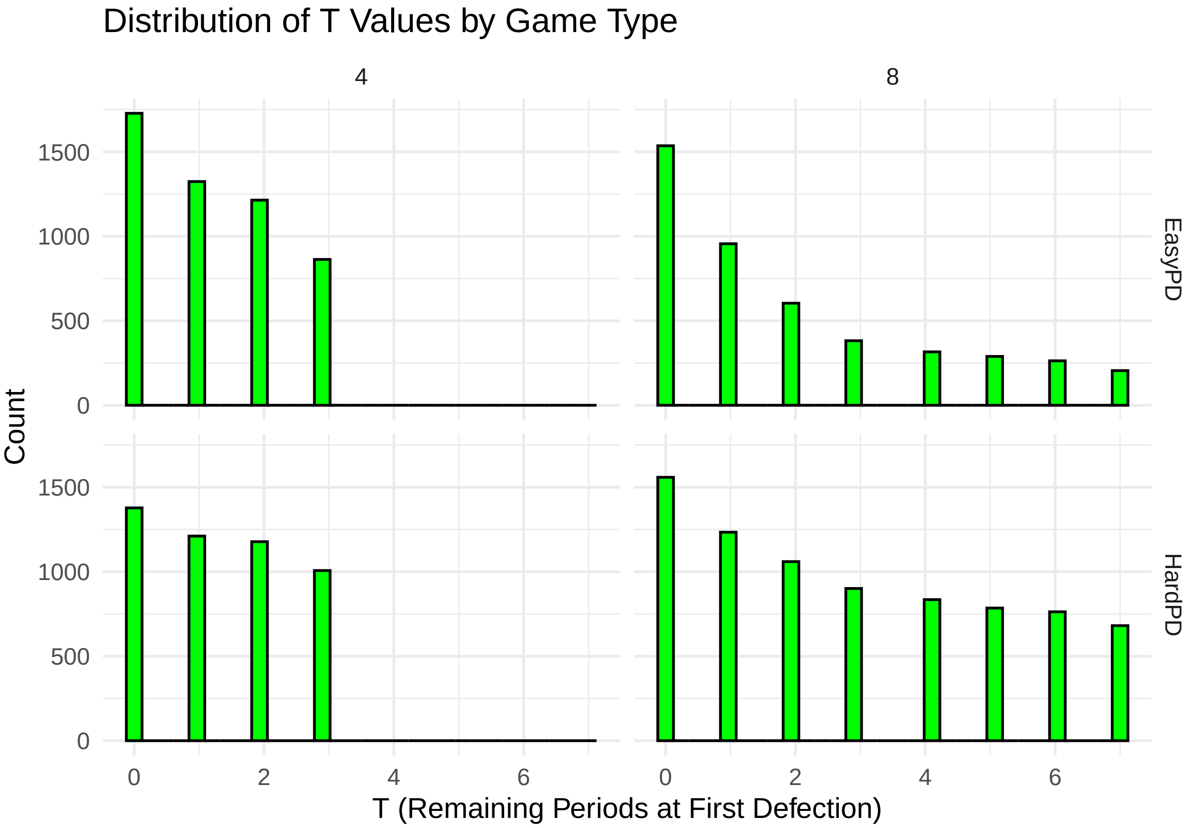
3.4 'T' Value Calculation and Distribution Visualization

- **Objective:** Calculate T value (the number of remaining rounds when a player first defects) and visualize the distribution of T values.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

```
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 ==
  ) %>%
  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
```

```
# Visualization: Distribution of T Values with 2x2 Design
ggplot(df_merged %>% filter(!is.na(T)), aes(x = T)) +
  geom_histogram(bins = 30, fill = "green", color = "black") +
  facet_grid(pd_difficulty ~ horizon) +
  labs(title = "Distribution of T Values by Game Type", x = "T (Remaining Periods at First Defectio
  theme_minimal()
```



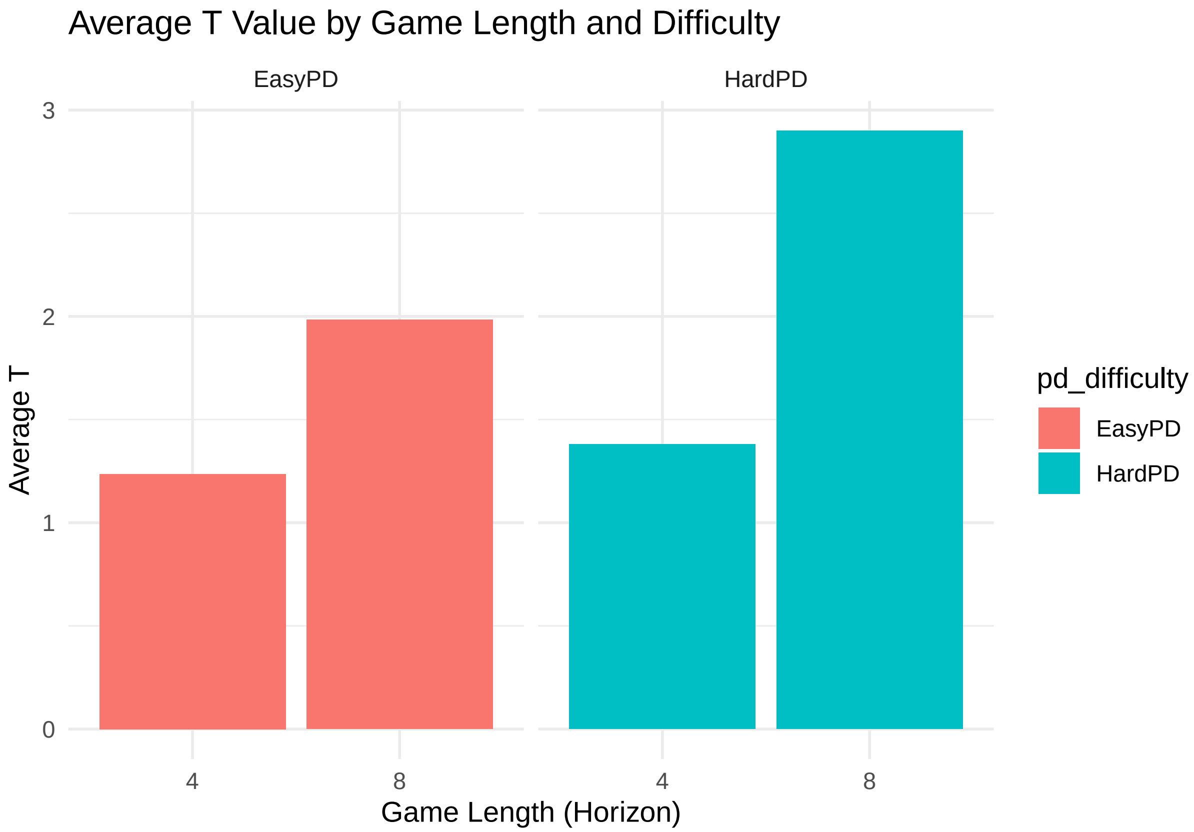
3.5 Cross-Tabulation of T Values by Game Length

- **Objective:** Summarize T values by game length (horizon) to show mean and standard deviation of T for each horizon and visualize these statistics.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

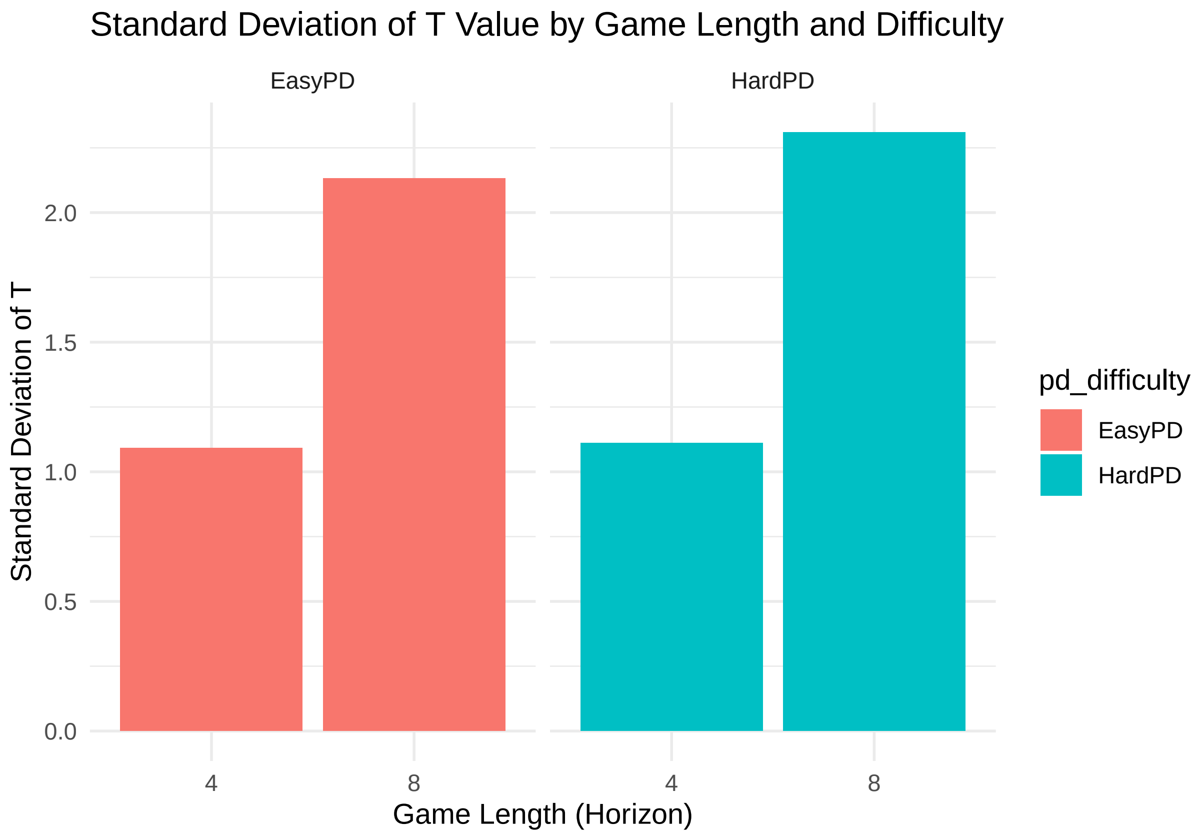
```
# Update T_summary to group by both horizon and pd_difficulty
T_summary <- df_merged %>%
  group_by(horizon, pd_difficulty) %>%
  summarise(
    mean_T = mean(T, na.rm = TRUE),
    sd_T = sd(T, na.rm = TRUE),
    count = n(),
    .groups = "drop"
  )

print(T_summary)
```

```
## # A tibble: 4 x 5
##   horizon pd_difficulty mean_T  sd_T count
##     <int> <chr>          <dbl> <dbl> <int>
## 1       4 EasyPD          1.24  1.09  6560
## 2       4 HardPD          1.38  1.11  5360
## 3       8 EasyPD          1.98  2.13  9920
## 4       8 HardPD          2.90  2.31 11520
```

```
# Visualization: Average T Value by Game Length and Difficulty
ggplot(T_summary, aes(x = factor(horizon), y = mean_T, fill = pd_difficulty)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ pd_difficulty) +
  labs(title = "Average T Value by Game Length and Difficulty", x = "Game Length (Horizon)", y = "A
  theme_minimal()
```



```
# Visualization: Standard Deviation of T Value by Game Length and Difficulty
ggplot(T_summary, aes(x = factor(horizon), y = sd_T, fill = pd_difficulty)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ pd_difficulty) +
  labs(title = "Standard Deviation of T Value by Game Length and Difficulty", x = "Game Length (Hor
  theme_minimal()
```



Step 4: Strategy Identification

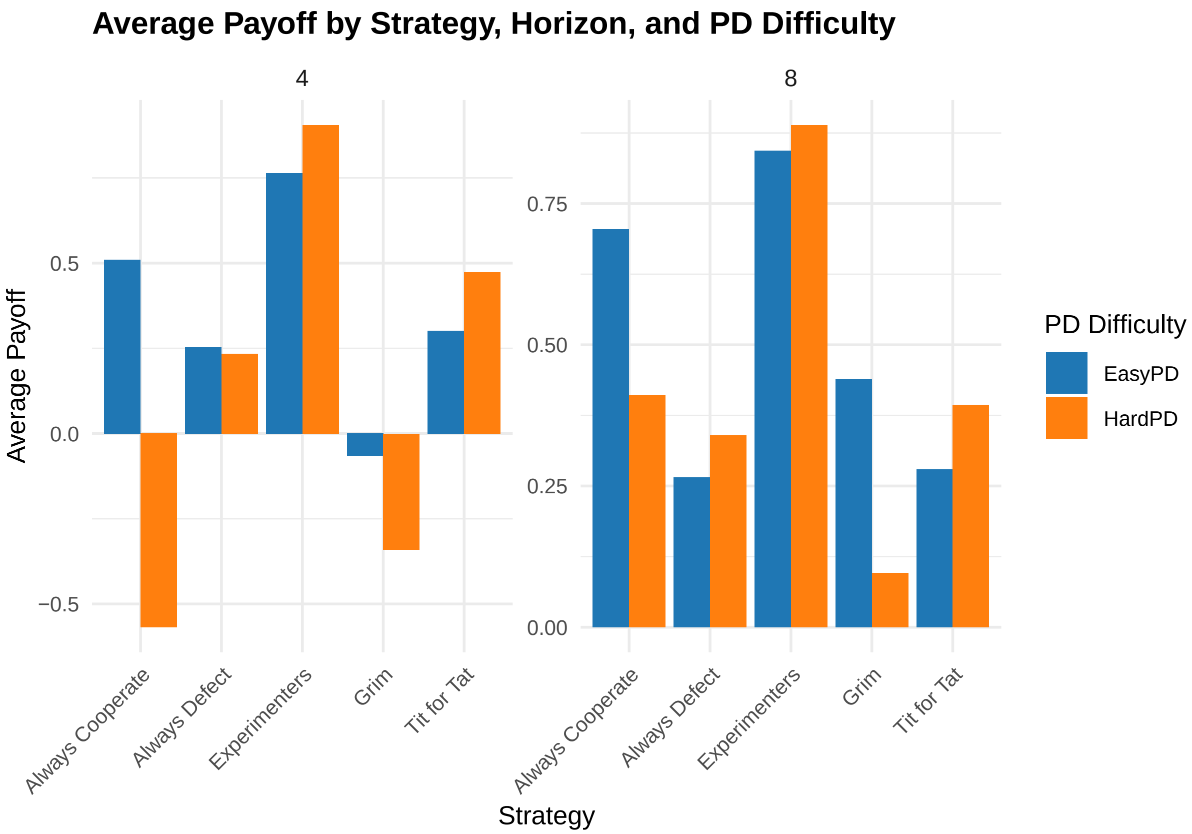
- **Objective:** Define a function to classify player strategies (Always Cooperate, Always Defect, Grim, Tit-for-Tat, and Experimenter) and merge the identified strategies with the main dataset.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

```r
# Define function for strategy classification
identify_strategy <- function(player_coop, opponent_coop) {

  # Check for Always Cooperate (C) - Classify first as per instruction
  if (all(player_coop == 1, na.rm = TRUE)) return("Always Cooperate")

  # Check for Always Defect (D) - Classify second
  else if (all(player_coop == 0, na.rm = TRUE)) return("Always Defect")

  # Check for Grim Strategy (G) - Classify third
  else if (all((player_coop == 1) | (cumsum(opponent_coop == 0) > 0), na.rm = TRUE)) return("Grim")

  # Check for Tit-for-Tat (TFT) - Classify last before Experimenters
  else if (length(player_coop) > 1 && all(player_coop[-1] == lag(opponent_coop)[-1], na.rm = TRUE))

  else return("Experimenters")  # Classify as "Experimenters" if no prior strategies match
}

# Apply function across dataset
df_merged <- df_merged %>%
  group_by(player_id, supergame) %>%
  mutate(strategy_label = identify_strategy(player_coop, opponent_coop)) %>%
  ungroup()
```

```r
# Calculate average payoff by strategy, horizon, and PD difficulty
strategy_payoff_difficulty <- df_merged %>%
  group_by(strategy_label, horizon, pd_difficulty) %>%
  summarise(
    avg_payoff = mean(payoff, na.rm = TRUE),
    payoff_variance = var(payoff, na.rm = TRUE),
    count = n(),
    .groups = "drop"
  )
# Print summary table
print(strategy_payoff_difficulty)
```

```
## # A tibble: 20 x 6
##    strategy_label   horizon pd_difficulty avg_payoff payoff_variance count
##    <chr>              <int> <chr>              <dbl>           <dbl> <int>
##  1 Always Cooperate       4 EasyPD            0.510            0.949   212
##  2 Always Cooperate       4 HardPD           -0.568            3.59     88
##  3 Always Cooperate       8 EasyPD            0.704            0.629   376
##  4 Always Cooperate       8 HardPD            0.410            1.92    208
##  5 Always Defect          4 EasyPD            0.253            0.442  3076
##  6 Always Defect          4 HardPD            0.235            0.884  3768
##  7 Always Defect          8 EasyPD            0.266            0.461  1032
##  8 Always Defect          8 HardPD            0.340            1.24   3344
##  9 Experimenters          4 EasyPD            0.764            1.19    768
## 10 Experimenters          4 HardPD            0.905            5.91    308
## 11 Experimenters          8 EasyPD            0.844            0.549  2928
## 12 Experimenters          8 HardPD            0.889            3.56   2544
## 13 Grim                   4 EasyPD           -0.0646           0.846  2440
## 14 Grim                   4 HardPD           -0.340            3.14   1152
## 15 Grim                   8 EasyPD            0.439            0.652  5464
## 16 Grim                   8 HardPD            0.0958           2.75   5064
## 17 Tit for Tat            4 EasyPD            0.302            1.66     64
## 18 Tit for Tat            4 HardPD            0.473            6.38     44
## 19 Tit for Tat            8 EasyPD            0.279            1.02    120
## 20 Tit for Tat            8 HardPD            0.394            4.03    360
```

```r
#####========Updated_241101===================================================#####

# Visualization: Average Payoff by Strategy, Horizon, and PD Difficulty with adjusted text size
ggplot(strategy_payoff_difficulty, aes(x = strategy_label, y = avg_payoff, fill = pd_difficulty)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  facet_wrap(~ horizon, scales = "free") +
  scale_fill_manual(values = c("EasyPD" = "#1f77b4", "HardPD" = "#ff7f0e", "NA" = "grey70")) +
  labs(title = "Average Payoff by Strategy, Horizon, and PD Difficulty",
       x = "Strategy",
       y = "Average Payoff",
       fill = "PD Difficulty") +
  theme_minimal() +
  theme(
    text = element_text(size = 10),  # Decrease overall text size
    axis.text.x = element_text(angle = 45, hjust = 1, size = 8),  # Rotate and reduce x-axis labels
    strip.text = element_text(size = 9),  # Decrease size of facet labels
    plot.title = element_text(size = 12, face = "bold")  # Title size adjustment
  )
```

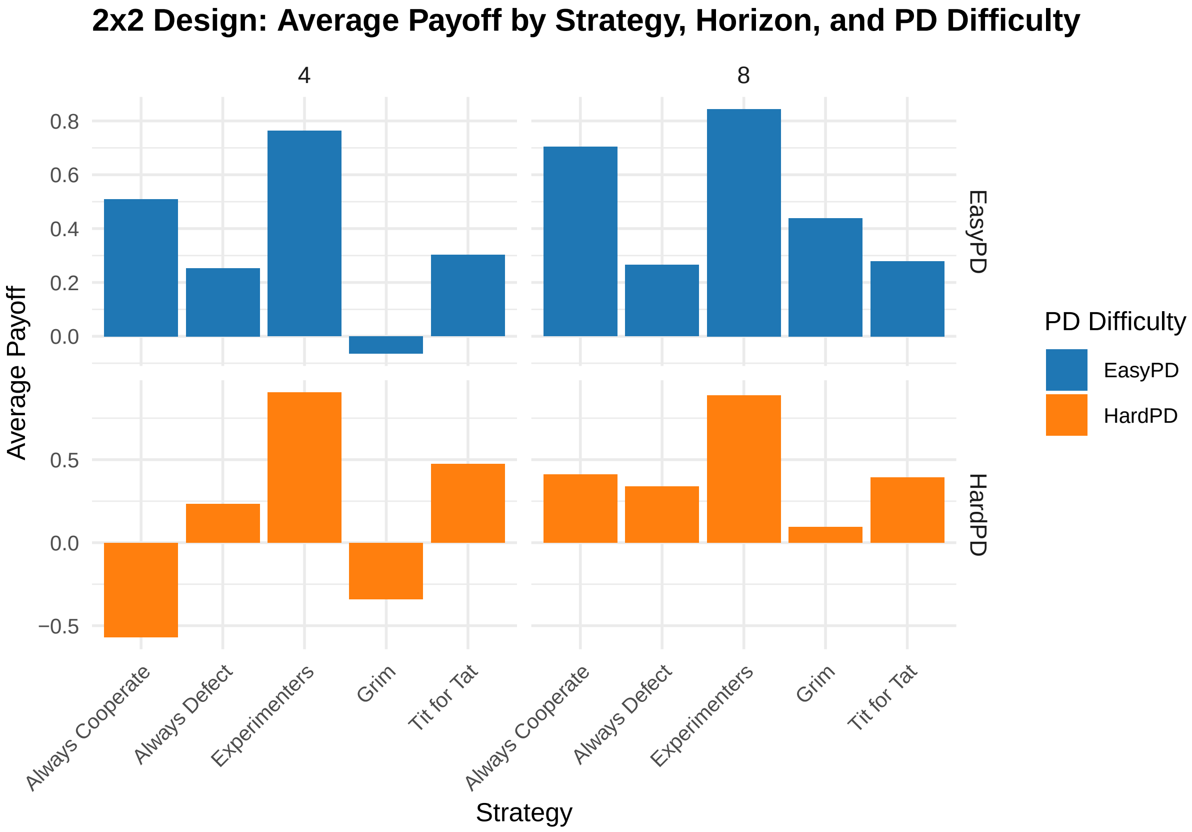


```r
#==========================================================================#
```

Step 5: Strategy Payoff Analysis

- **Objective:** Analyze the average payoff and payoff variance for each identified strategy, segmented by game length (horizon) and PD difficulty level (EasyPD or HardPD). This step provides insights into the effectiveness of each strategy in different game conditions and helps identify which strategies yield higher payoffs across varying difficulty levels and game lengths.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

```r
# Calculate average payoff by strategy, horizon, and PD difficulty
strategy_payoff_summary <- df_merged %>%
  group_by(strategy_label, horizon, pd_difficulty) %>%
  summarise(
    avg_strategy_payoff = mean(payoff, na.rm = TRUE),
    payoff_variance = var(payoff, na.rm = TRUE),
    count = n(),
    .groups = "drop"
  )

# 2x2 Design Visualization with adjusted text size
ggplot(strategy_payoff_summary, aes(x = strategy_label, y = avg_strategy_payoff, fill = pd_difficul
  geom_bar(stat = "identity", position = position_dodge()) +
  facet_grid(pd_difficulty ~ horizon, scales = "free") +
  scale_fill_manual(values = c("EasyPD" = "#1f77b4", "HardPD" = "#ff7f0e", "NA" = "grey70")) +
  labs(title = "2x2 Design: Average Payoff by Strategy, Horizon, and PD Difficulty",
       x = "Strategy",
       y = "Average Payoff",
       fill = "PD Difficulty") +
  theme_minimal() +
  theme(
    text = element_text(size = 10),  # Adjust overall text size
    axis.text.x = element_text(angle = 45, hjust = 1, size = 8),  # Rotate and resize x-axis labels
    strip.text = element_text(size = 9),  # Decrease facet label size
    plot.title = element_text(size = 12, face = "bold")  # Adjust title size and make bold
  )
```



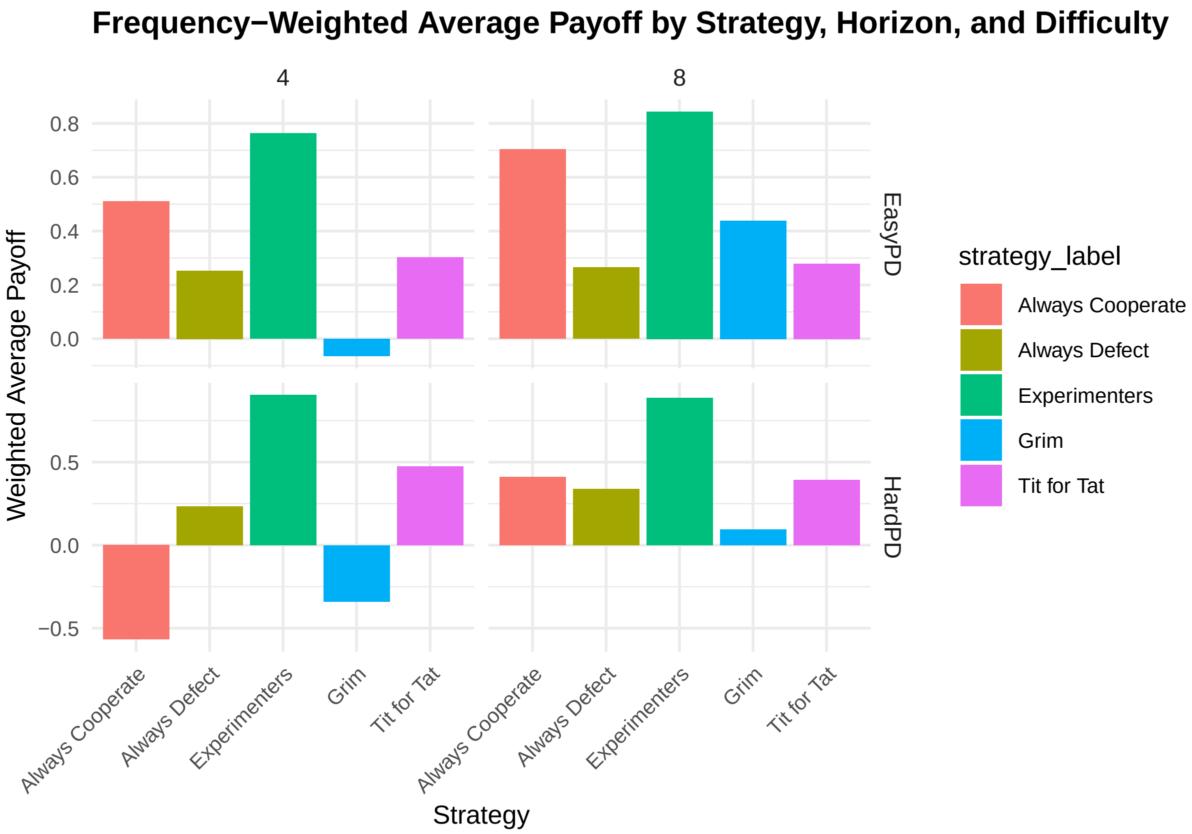
Extended Analysis

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.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

```r
weighted_payoff_summary <- strategy_payoff_summary %>%
  group_by(strategy_label, horizon, pd_difficulty) %>%
  mutate(weighted_avg_payoff = avg_strategy_payoff * (count / sum(count)))
```

```
# Visualization with 2x2 Design and adjusted text size
ggplot(weighted_payoff_summary, aes(x = strategy_label, y = weighted_avg_payoff, fill = strategy_la
  geom_bar(stat = "identity", position = "dodge") +
  facet_grid(pd_difficulty ~ horizon, scales = "free") +
  labs(title = "Frequency-Weighted Average Payoff by Strategy, Horizon, and Difficulty",
       x = "Strategy", y = "Weighted Average Payoff") +
  theme_minimal() +
  theme(
    text = element_text(size = 10),  # Set overall text size
    axis.text.x = element_text(angle = 45, hjust = 1, size = 8),  # Rotate and resize x-axis labels
    strip.text = element_text(size = 9),  # Decrease facet label size
    plot.title = element_text(size = 12, face = "bold")  # Bold and adjust title size
  )
```


Frequency−Weighted Average Payoff by Strategy, Horizon, and Difficulty

Step 7: Performance Against Non-Fixed Strategies

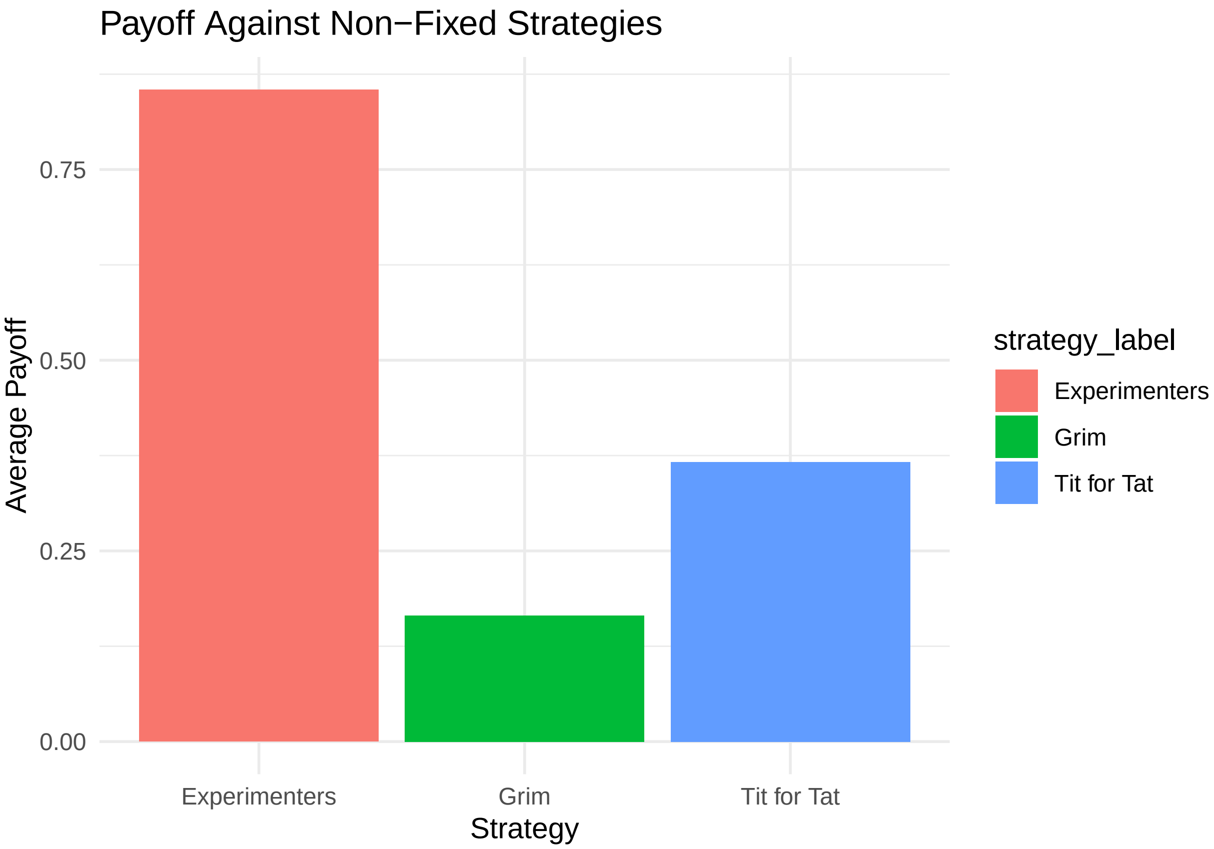
- **Objective:** This analysis compares each fixed strategy's performance when interacting with non-fixed strategies, highlighting adaptability.

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

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

```
## # A tibble: 3 x 3
##   strategy_label avg_payoff_against_non_fixed variance_against_non_fixed
##   <chr>                                 <dbl>                      <dbl>
## 1 Experimenters                         0.855                       2.05
## 2 Grim                                  0.165                       1.70
## 3 Tit for Tat                           0.366                       3.32
```

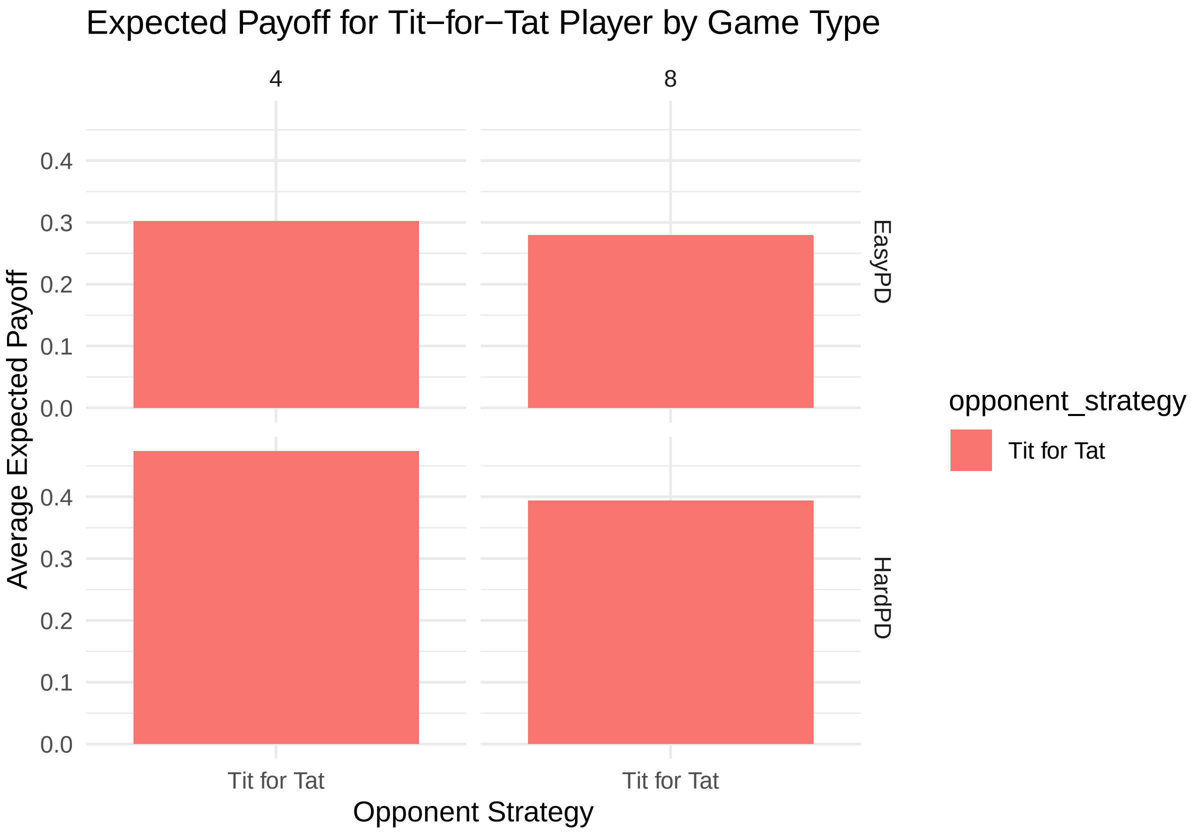
```r
# Visualization
ggplot(non_fixed_performance, aes(x = strategy_label, y = avg_payoff_against_non_fixed, fill = strategy_
  geom_bar(stat = "identity") +
  labs(title = "Payoff Against Non-Fixed Strategies", x = "Strategy", y = "Average Payoff") +
  theme_minimal()
```



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.
- **Code Updates (Nov 1, 2024) => 2x2 Design**

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

# Calculating Expected Payoff for Tit-for-Tat vs Opponent Strategies
hypothetical_tft_performance <- df_merged %>%
  filter(strategy_label == "Tit for Tat") %>%
  group_by(opponent_strategy = strategy_label, horizon, pd_difficulty) %>%
  summarise(
    avg_expected_payoff = mean(payoff, na.rm = TRUE),
    variance_expected_payoff = var(payoff, na.rm = TRUE),
    count = n(),
    .groups = "drop"
  )

# Visualization with 2x2 Design
ggplot(hypothetical_tft_performance, aes(x = opponent_strategy, y = avg_expected_payoff, fill = opp
  geom_bar(stat = "identity", position = "dodge") +
  facet_grid(pd_difficulty ~ horizon) +
  labs(title = "Expected Payoff for Tit-for-Tat Player by Game Type",
       x = "Opponent Strategy", y = "Average Expected Payoff") +
  theme_minimal()
```

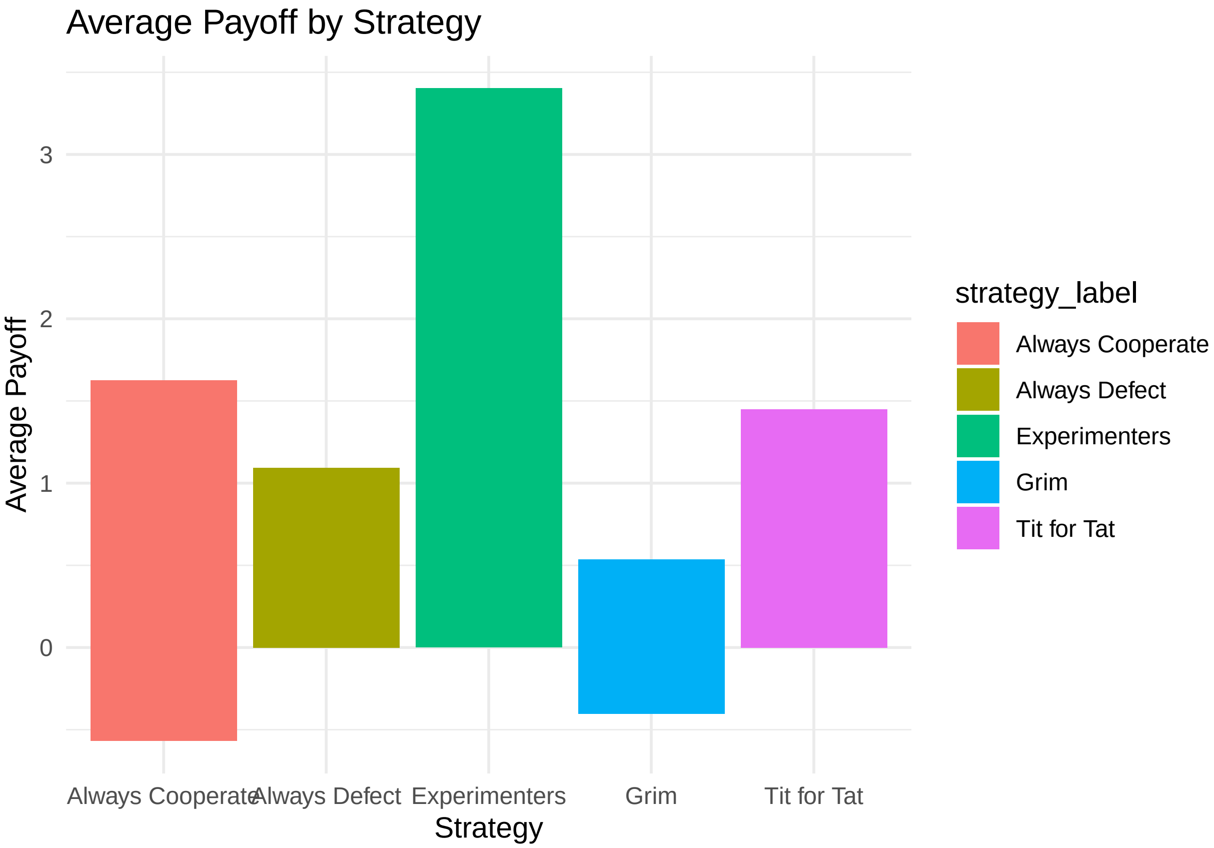


Step 9: Visualizations and Graphical Analysis

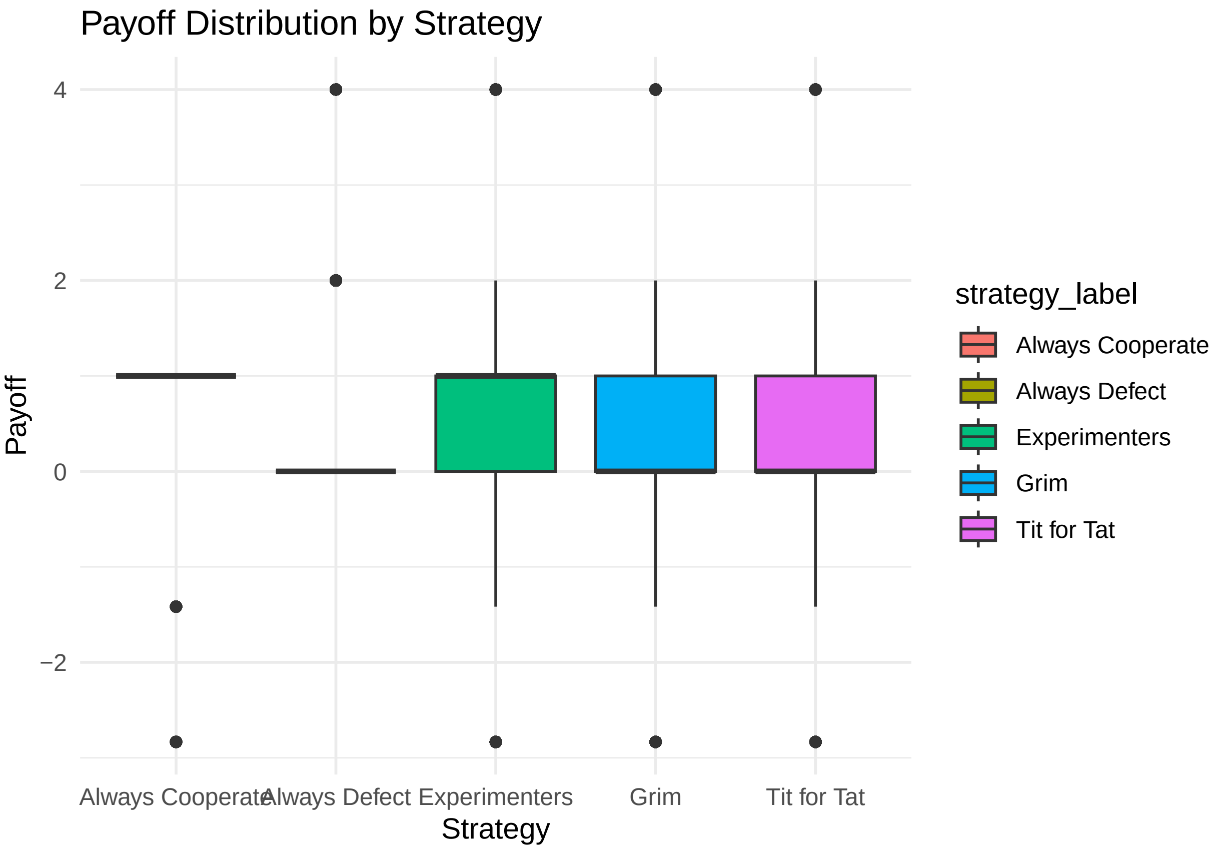
- **Objective:** Provides a visualization of strategy payoff distributions and strategy counts, offering a visual comparison of each strategy's performance and popularity.
- **Code Updates (Nov 1, 2024)**

```
### 9 Visualizations

library(ggplot2)
library(ggridges)

# 1. Bar Plot: Average Payoff by Strategy
```

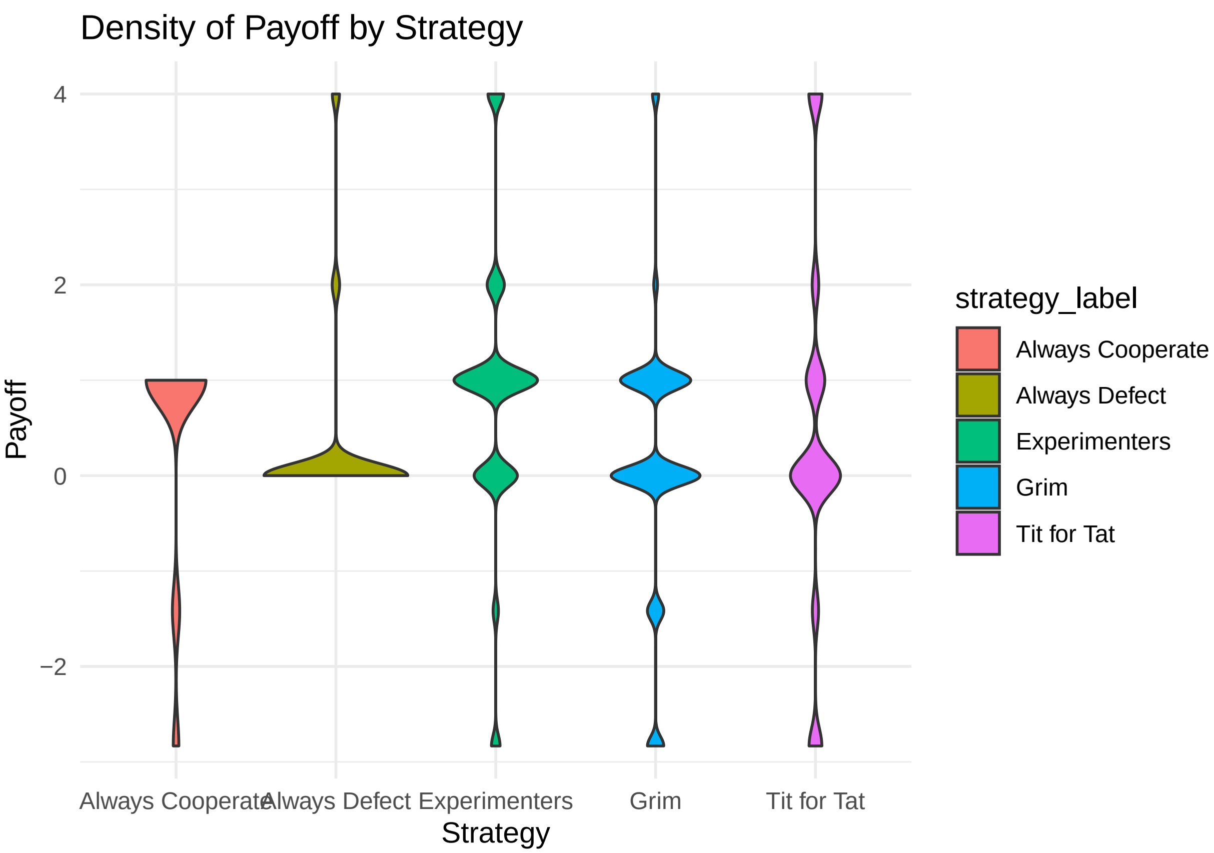
```r
ggplot(strategy_payoff_summary, aes(x = strategy_label, y = avg_strategy_payoff, fill = strategy_label)
  geom_bar(stat = "identity") +
  labs(title = "Average Payoff by Strategy", x = "Strategy", y = "Average Payoff") +
  theme_minimal()
```



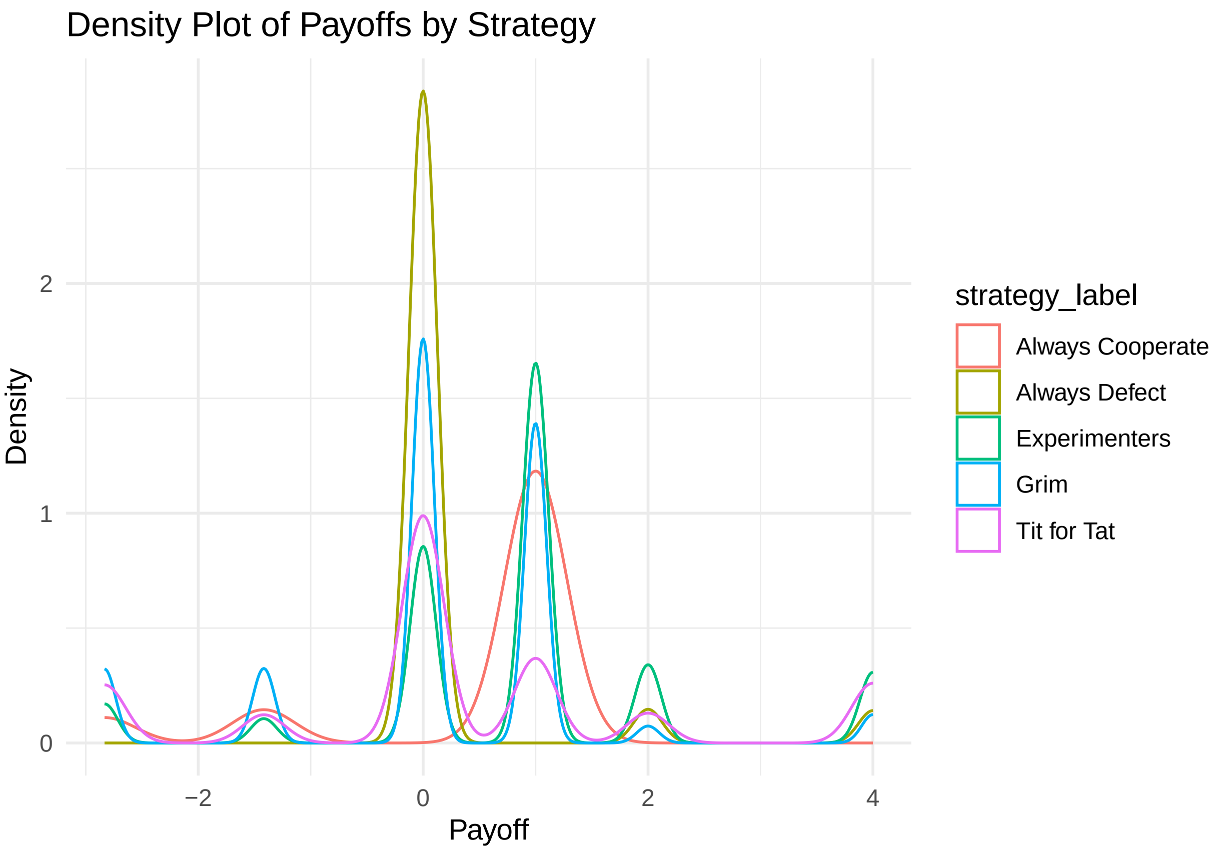
```r
# 2. Box Plot: Payoff Distribution by Strategy
ggplot(df_merged, aes(x = strategy_label, y = payoff, fill = strategy_label)) +
  geom_boxplot() +
  labs(title = "Payoff Distribution by Strategy", x = "Strategy", y = "Payoff") +
  theme_minimal()
```



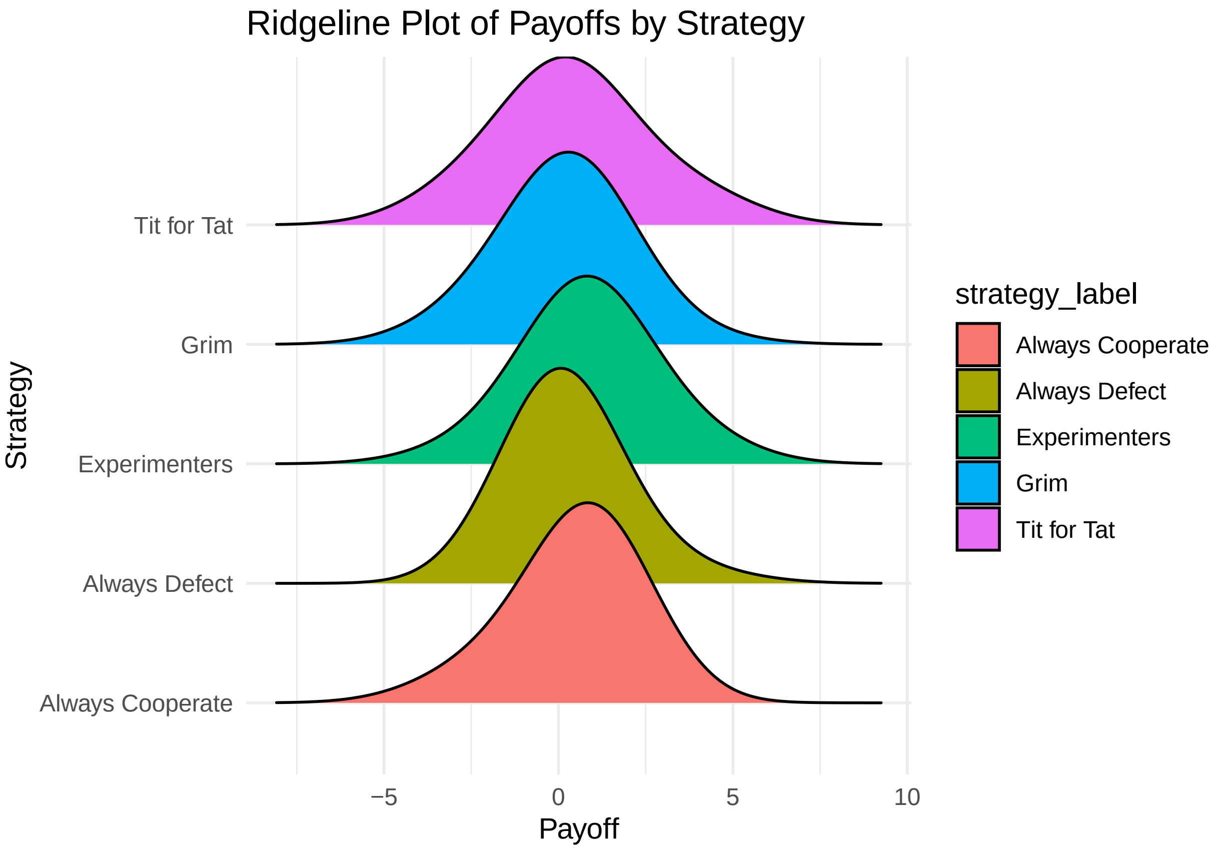
```
# 3. Violin Plot: Density of Payoff by Strategy
ggplot(df_merged, aes(x = strategy_label, y = payoff, fill = strategy_label)) +
  geom_violin() +
  labs(title = "Density of Payoff by Strategy", x = "Strategy", y = "Payoff") +
  theme_minimal()
```



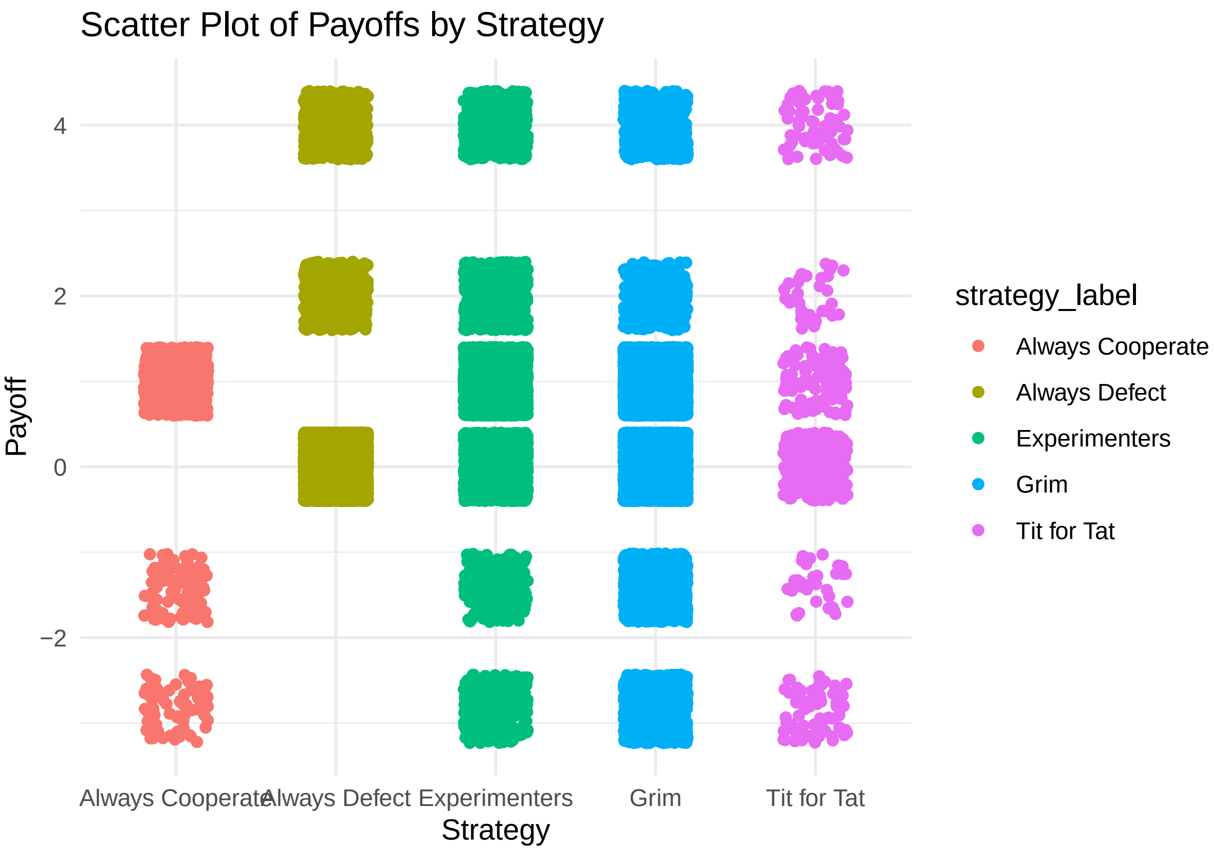
```r
# 4. Density Plot: Density of Payoffs by Strategy
ggplot(df_merged, aes(x = payoff, color = strategy_label)) +
  geom_density() +
  labs(title = "Density Plot of Payoffs by Strategy", x = "Payoff", y = "Density") +
  theme_minimal()
```



```r
# 5. Ridgeline Plot: Payoffs by Strategy
ggplot(df_merged, aes(x = payoff, y = strategy_label, fill = strategy_label)) +
  geom_density_ridges(bandwidth = 1.75) +
  labs(title = "Ridgeline Plot of Payoffs by Strategy", x = "Payoff", y = "Strategy") +
  theme_minimal()
```



```
# 6. Scatter Plot: Payoffs by Strategy
ggplot(df_merged, aes(x = strategy_label, y = payoff, color = strategy_label)) +
  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
4
2
0
−2
Always Cooperate
Always Defect
Experimenters
Grim
Tit for Tat
Strategy
strategy_label
● Always Cooperate
● Always Defect
● Experimenters
● Grim
● Tit for Tat