467

2025-12-02

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.5.2
## v ggplot2   4.0.1     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:purrr':
##
##     flatten
```

```r
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(lmtest)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(dplyr)
library(tidyverse) #       dplyr
```

```
path_final <- "/Users/linuohu/Desktop/467/project/tmdb_5000_movies.csv"

data <- read_csv(path_final)
```

```
## Rows: 4803 Columns: 20
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (12): genres, homepage, keywords, original_language, original_title, ov...
## dbl   (7): budget, id, popularity, revenue, runtime, vote_average, vote_count
## date  (1): release_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
print(glimpse(data))
```

```
## Rows: 4,803
## Columns: 20
## $ budget               <dbl> 2.37e+08, 3.00e+08, 2.45e+08, 2.50e+08, 2.60e+08,~
## $ genres               <chr> "[{\"id\": 28, \"name\": \"Action\"}, {\"id\": 12~
## $ homepage             <chr> "http://www.avatarmovie.com/", "http://disney.go.~
## $ id                   <dbl> 19995, 285, 206647, 49026, 49529, 559, 38757, 998~
## $ keywords             <chr> "[{\"id\": 1463, \"name\": \"culture clash\"}, {\~
## $ original_language    <chr> "en", "en", "en", "en", "en", "en", "en", "en", "~
## $ original_title       <chr> "Avatar", "Pirates of the Caribbean: At World's E~
## $ overview             <chr> "In the 22nd century, a paraplegic Marine is disp~
## $ popularity           <dbl> 150.43758, 139.08262, 107.37679, 112.31295, 43.92~
## $ production_companies <chr> "[{\"name\": \"Ingenious Film Partners\", \"id\":~
## $ production_countries <chr> "[{\"iso_3166_1\": \"US\", \"name\": \"United Sta~
## $ release_date         <date> 2009-12-10, 2007-05-19, 2015-10-26, 2012-07-16, ~
## $ revenue              <dbl> 2787965087, 961000000, 880674609, 1084939099, 284~
## $ runtime              <dbl> 162, 169, 148, 165, 132, 139, 100, 141, 153, 151,~
## $ spoken_languages     <chr> "[{\"iso_639_1\": \"en\", \"name\": \"English\"},~
## $ status               <chr> "Released", "Released", "Released", "Released", "~
## $ tagline              <chr> "Enter the World of Pandora.", "At the end of the~
## $ title                <chr> "Avatar", "Pirates of the Caribbean: At World's E~
## $ vote_average         <dbl> 7.2, 6.9, 6.3, 7.6, 6.1, 5.9, 7.4, 7.3, 7.4, 5.7,~
## $ vote_count           <dbl> 11800, 4500, 4466, 9106, 2124, 3576, 3330, 6767, ~
```

```
## # A tibble: 4,803 x 20
##       budget genres    homepage     id keywords original_language original_title
##        <dbl> <chr>     <chr>     <dbl> <chr>    <chr>             <chr>
##  1 237000000 "[{\"id\~ http://~  19995 "[{\"id~ en                Avatar
##  2 300000000 "[{\"id\~ http://~    285 "[{\"id~ en                Pirates of th~
##  3 245000000 "[{\"id\~ http://~ 206647 "[{\"id~ en                Spectre
##  4 250000000 "[{\"id\~ http://~  49026 "[{\"id~ en                The Dark Knig~
##  5 260000000 "[{\"id\~ http://~  49529 "[{\"id~ en                John Carter
##  6 258000000 "[{\"id\~ http://~    559 "[{\"id~ en                Spider-Man 3
##  7 260000000 "[{\"id\~ http://~  38757 "[{\"id~ en                Tangled
##  8 280000000 "[{\"id\~ http://~  99861 "[{\"id~ en                Avengers: Age~
##  9 250000000 "[{\"id\~ http://~    767 "[{\"id~ en                Harry Potter ~
## 10 250000000 "[{\"id\~ http://~ 209112 "[{\"id~ en                Batman v Supe~
## # i 4,793 more rows
## # i 13 more variables: overview <chr>, popularity <dbl>,
## #   production_companies <chr>, production_countries <chr>,
## #   release_date <date>, revenue <dbl>, runtime <dbl>, spoken_languages <chr>,
## #   status <chr>, tagline <chr>, title <chr>, vote_average <dbl>,
## #   vote_count <dbl>
```

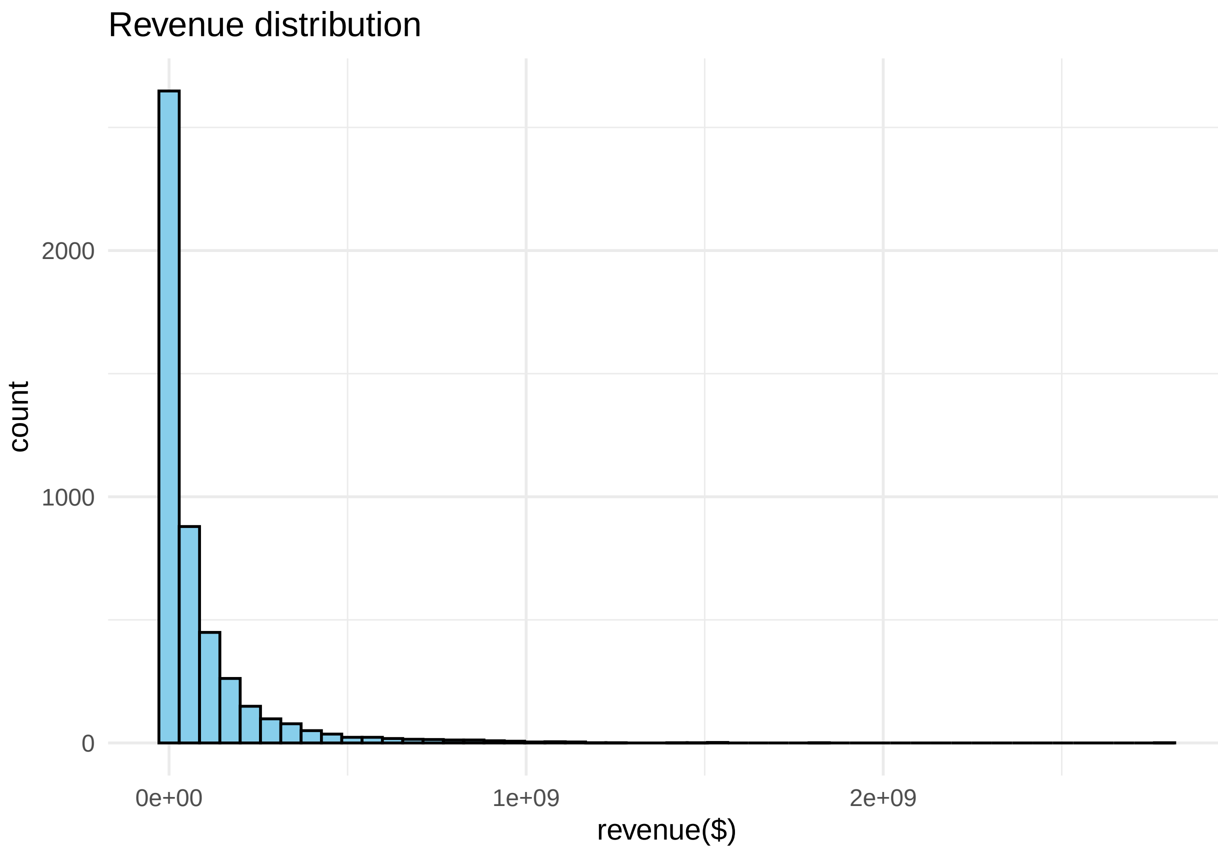
```r
response_variable <- data$revenue

summary(response_variable)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000e+00 0.000e+00 1.917e+07 8.226e+07 9.292e+07 2.788e+09
```

```r
sum(response_variable <= 0)
```

```
## [1] 1427
```

```r
ggplot(data, aes(x = revenue)) +
  geom_histogram(bins = 50, fill = "skyblue", color = "black") +
  labs(title = "Revenue distribution", x = "revenue($)") +
  theme_minimal()
```




```r
data_clean <- data %>%
  filter(revenue > 0 & budget > 0)

cat("Remaining observations after filtering zero revenue and budget:", nrow(data_clean), "\n")
```

```
## Remaining observations after filtering zero revenue and budget: 3229
```

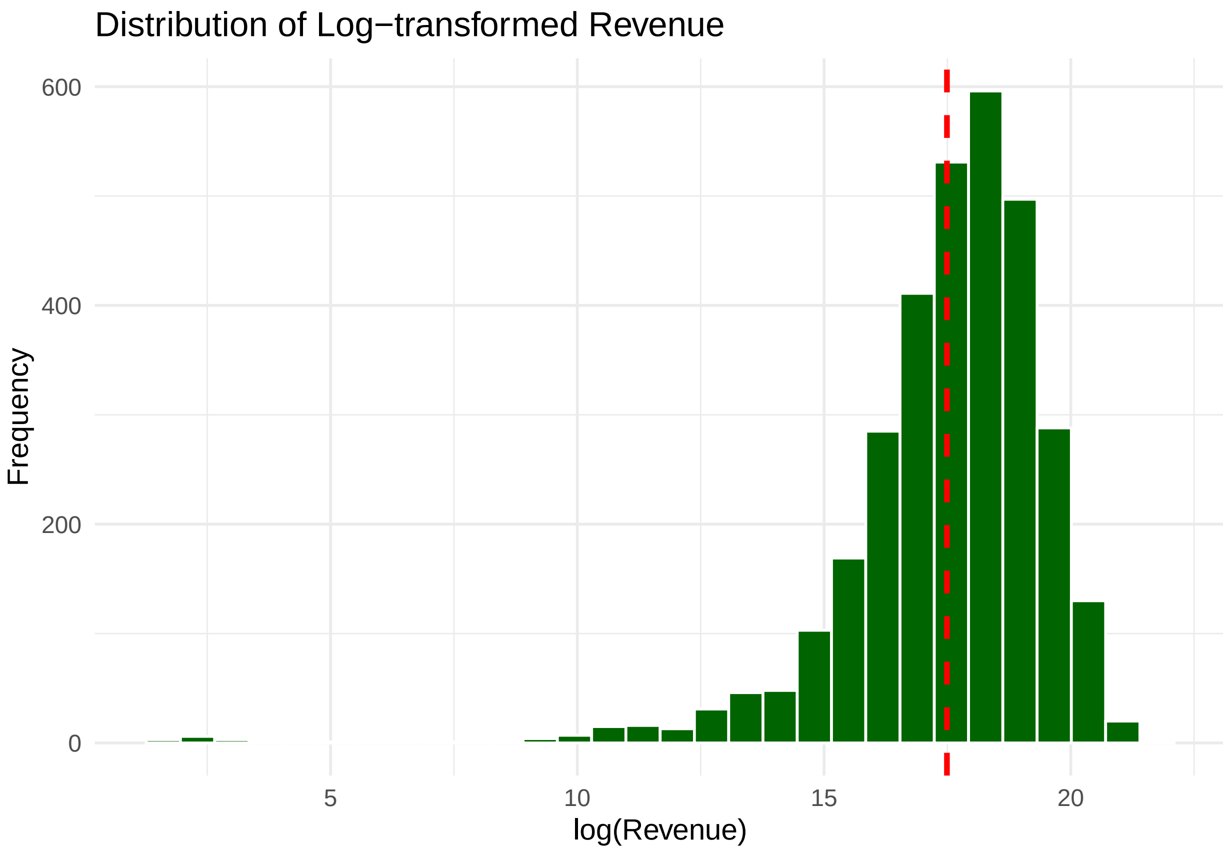
```r
data_clean <- data_clean %>%
  mutate(log_revenue = log(revenue))

cat("\nSummary of log(revenue):\n")
```

```
##
## Summary of log(revenue):
```

```r
print(summary(data_clean$log_revenue))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.609  16.649  17.826  17.491  18.801  21.749
```

```r
ggplot(data_clean, aes(x = log_revenue)) +
  geom_histogram(bins = 30, fill = "darkgreen", color = "white") +
  geom_vline(xintercept = mean(data_clean$log_revenue), linetype = "dashed", color = "red", linewidth =
  labs(title = "Distribution of Log-transformed Revenue",
       x = "log(Revenue)",
       y = "Frequency") +
  theme_minimal()
```




```
# ------------------------------------------------------------
#   1:   Budget ( )
# ------------------------------------------------------------
cat("\nSummary of original Budget:\n")
```

```
##
## Summary of original Budget:
```

```
#   NA     summary
print(summary(data_clean$budget))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##         1  10500000  25000000  40654445  55000000 380000000
```

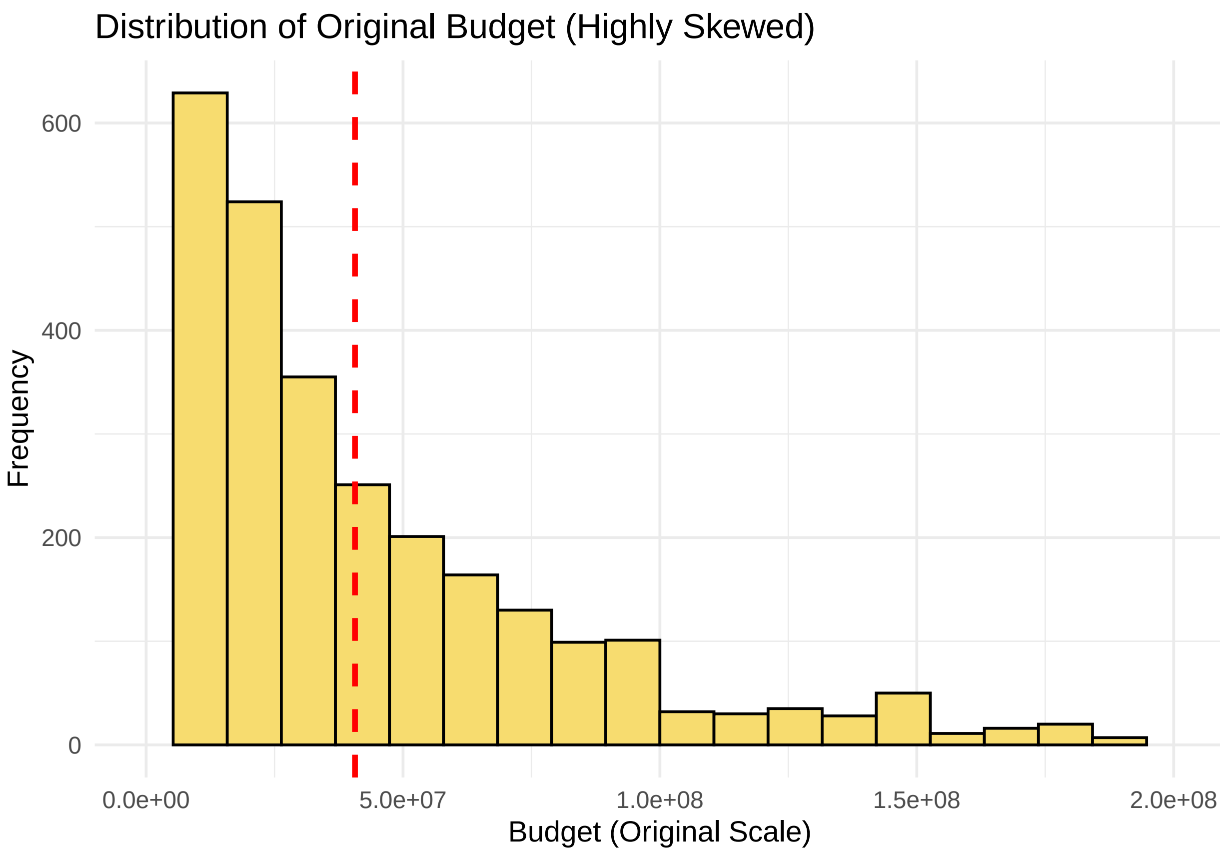
```
p1_original <- ggplot(data_clean, aes(x = budget)) +
  #   bins
  geom_histogram(bins = 20, fill = "#F7DC6F", color = "black") +
  #
  geom_vline(xintercept = mean(data_clean$budget, na.rm = TRUE), linetype = "dashed", color = "red", si
  labs(title = "Distribution of Original Budget (Highly Skewed)",
       x = "Budget (Original Scale)",
       y = "Frequency") +
  theme_minimal() +
  #   X
  xlim(0, quantile(data_clean$budget, 0.99, na.rm = TRUE))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
```

```
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
print(p1_original)
```

```
## Warning: Removed 27 rows containing non-finite outside the scale range
## (`stat_bin()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```




```r
# ------------------------------------------------------------
#   2:   Popularity (  )
# ------------------------------------------------------------
cat("\nSummary of original Popularity:\n")
```

```
##
## Summary of original Popularity:
```

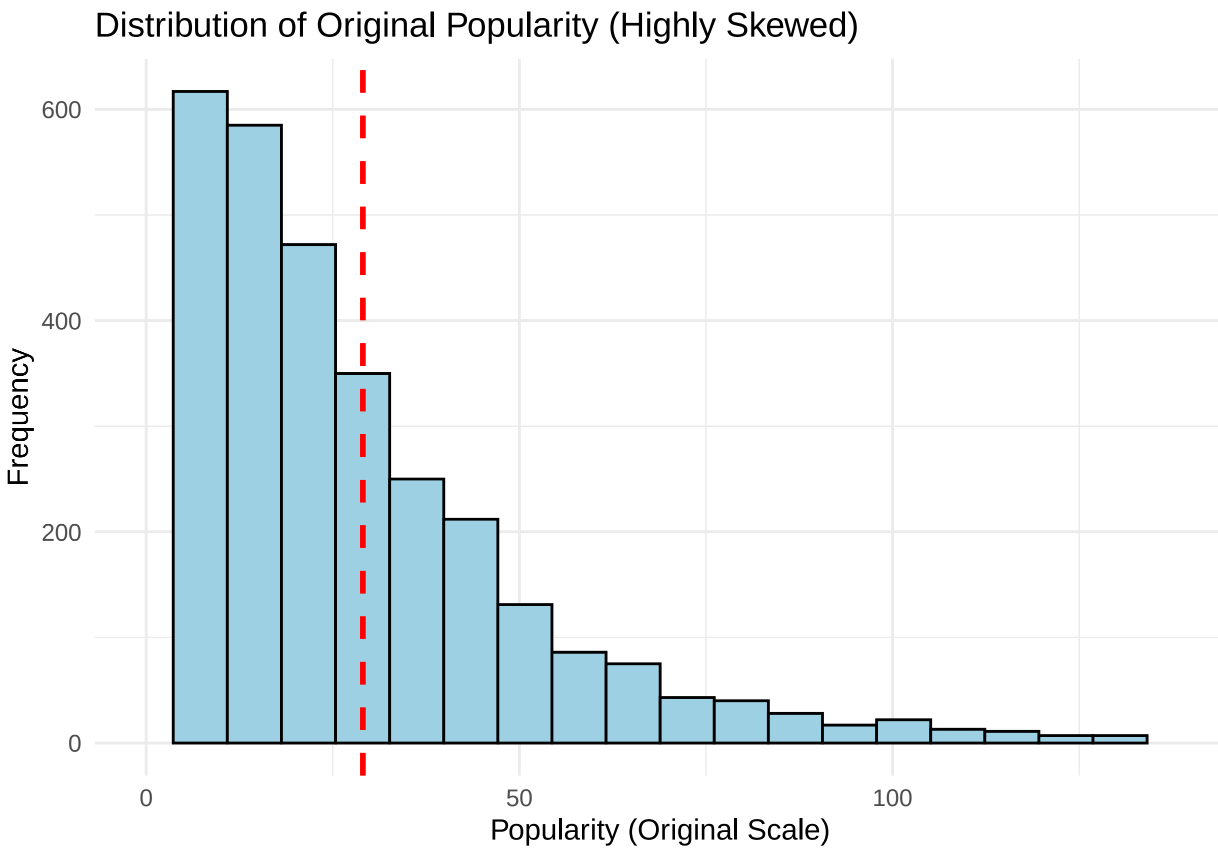
```r
print(summary(data_clean$popularity))
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
##    0.01998  10.44672  20.41035  29.03369  37.33572 875.58130
```

```r
p2_original <- ggplot(data_clean, aes(x = popularity)) +
  geom_histogram(bins = 20, fill = "#9DD0E3", color = "black") +
  geom_vline(xintercept = mean(data_clean$popularity, na.rm = TRUE), linetype = "dashed", color = "red"
  labs(title = "Distribution of Original Popularity (Highly Skewed)",
       x = "Popularity (Original Scale)",
       y = "Frequency") +
```

```r
  theme_minimal() +
  #   X
  xlim(0, quantile(data_clean$popularity, 0.99, na.rm = TRUE))
print(p2_original)
```

```
## Warning: Removed 33 rows containing non-finite outside the scale range (`stat_bin()`).
## Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```




```r
data_clean <- data_clean %>%
  mutate(log_budget = log(budget),
         log_popularity = log(popularity))

cat("\nSummary of log(budget):\n")
```

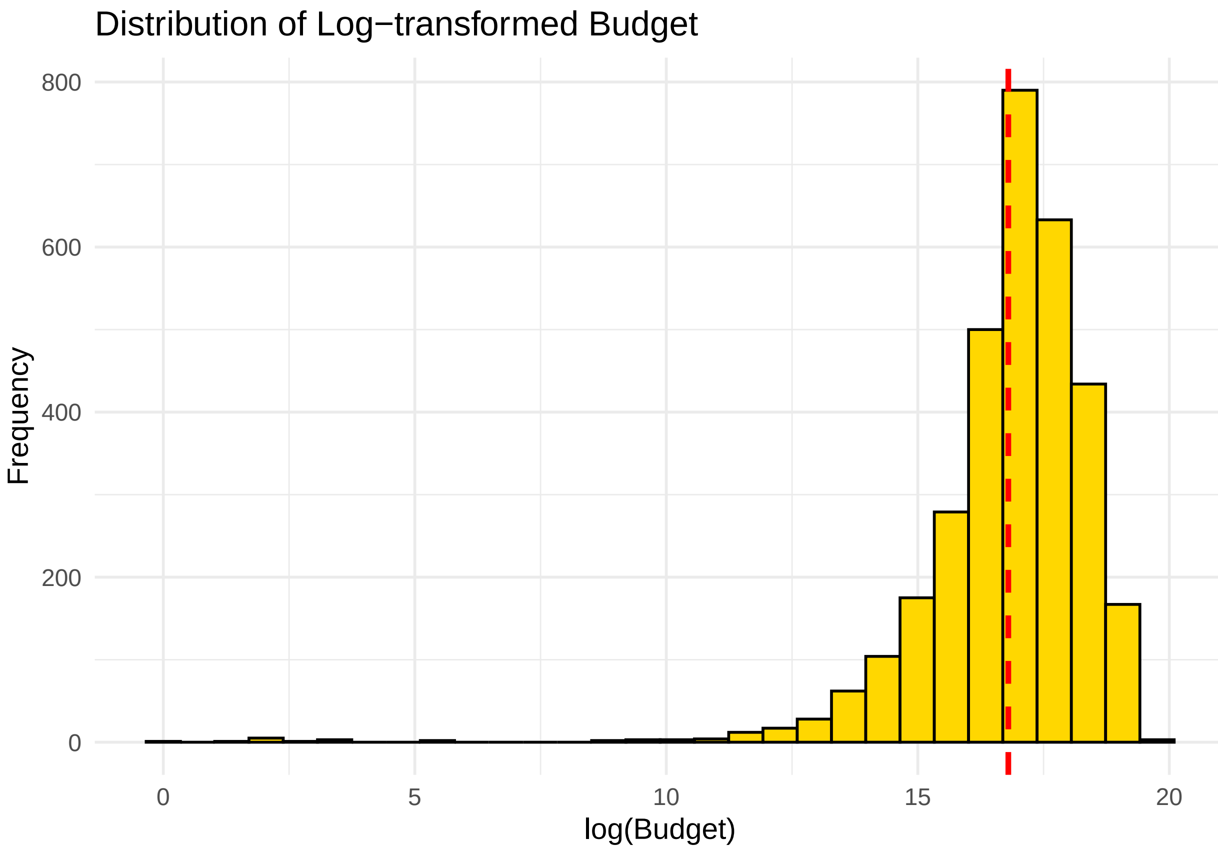
```
##
## Summary of log(budget):
```

```r
print(summary(data_clean$log_budget))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   16.17   17.03   16.80   17.82   19.76
```

```r
p1 <- ggplot(data_clean, aes(x = log_budget)) +
  geom_histogram(bins = 30, fill = "gold", color = "black") +
  geom_vline(xintercept = mean(data_clean$log_budget), linetype = "dashed", color = "red", size = 1) +
  labs(title = "Distribution of Log-transformed Budget",
       x = "log(Budget)",
```

```
      y = "Frequency") +
  theme_minimal()
print(p1)
```



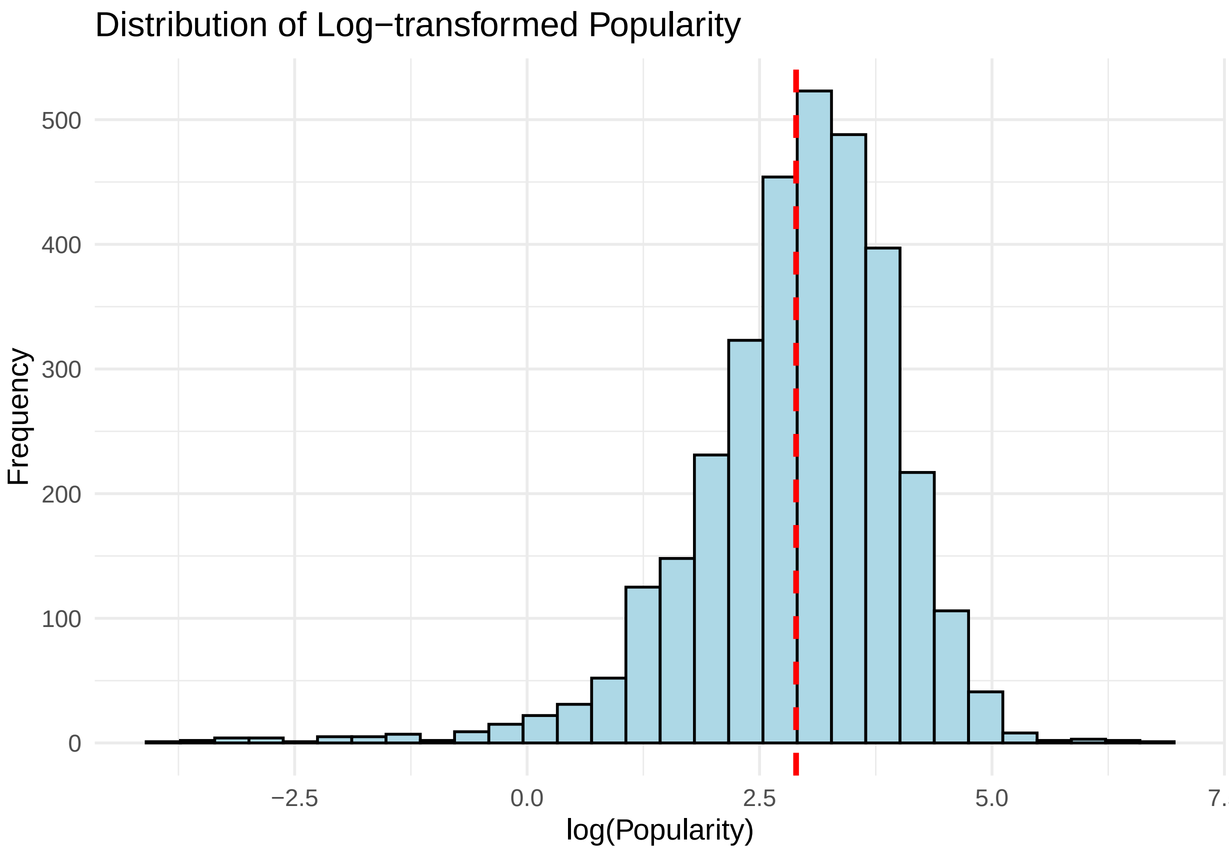

```
cat("\nSummary of log(popularity):\n")
```

```
##
## Summary of log(popularity):
```

```
print(summary(data_clean$log_popularity))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -3.913   2.346   3.016   2.893   3.620   6.775
```

```
p2 <- ggplot(data_clean, aes(x = log_popularity)) +
  geom_histogram(bins = 30, fill = "lightblue", color = "black") +
  geom_vline(xintercept = mean(data_clean$log_popularity), linetype = "dashed", color = "red", size = 1
  labs(title = "Distribution of Log-transformed Popularity",
       x = "log(Popularity)",
       y = "Frequency") +
  theme_minimal()
print(p2)
```



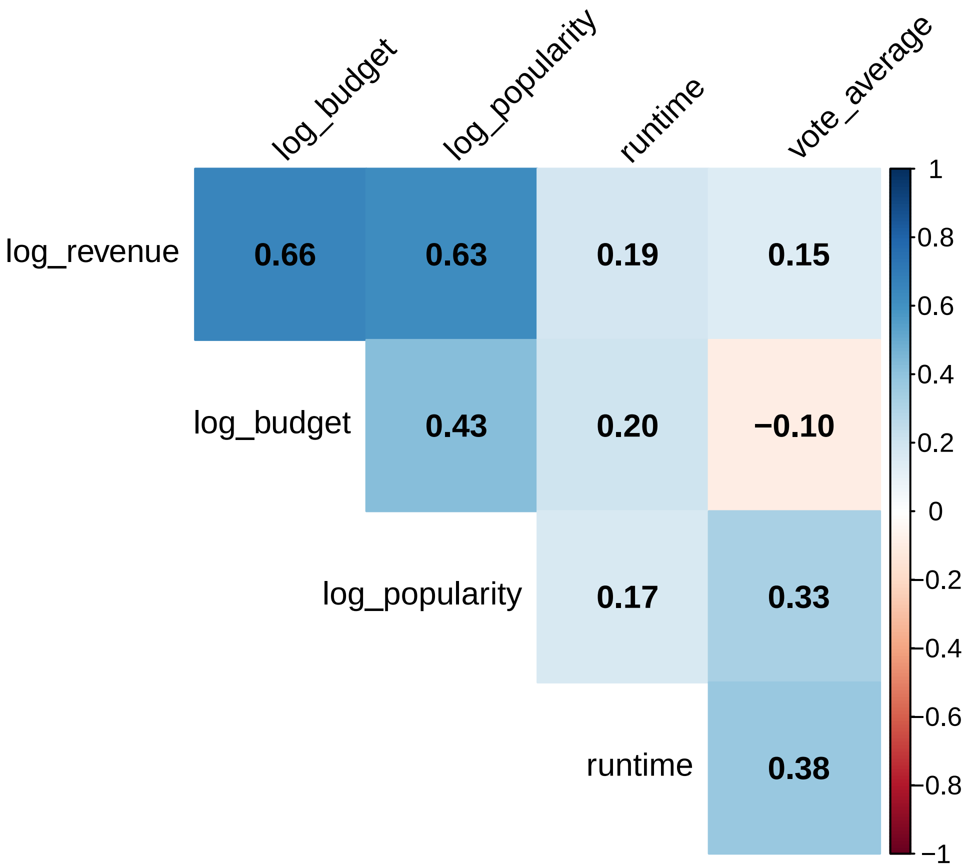

```r
df_for_corr <- data_clean %>%
  #   dplyr::select
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average) %>%
  na.omit()

correlation_matrix <- cor(df_for_corr)

cat("### Correlation Matrix of Key Quantitative Variables ###\n")
```

```
## ### Correlation Matrix of Key Quantitative Variables ###
```

```r
print(round(correlation_matrix, 3))
```

```
##                log_revenue log_budget log_popularity runtime vote_average
## log_revenue          1.000      0.657          0.630   0.190        0.147
## log_budget           0.657      1.000          0.425   0.205       -0.099
## log_popularity       0.630      0.425          1.000   0.168        0.330
## runtime              0.190      0.205          0.168   1.000        0.379
## vote_average         0.147     -0.099          0.330   0.379        1.000
```

```r
corrplot(correlation_matrix,
         method = "color",
         type = "upper",
         tl.col = "black",
         addCoef.col = "black",
         tl.srt = 45,
         diag = FALSE)
```




```
df_final <- data %>%
  filter(revenue > 0 & budget > 0) %>%
  mutate(log_revenue = log(revenue),
         log_budget = log(budget),
         log_popularity = log(popularity)) %>%
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average, original_language, genr
  na.omit()

key_vars <- c("log_revenue", "log_budget", "log_popularity", "runtime", "vote_average")

summary_stats <- df_final %>%
  pivot_longer(cols = all_of(key_vars), names_to = "Variable", values_to = "Value") %>%
  group_by(Variable) %>%
  summarise(
    N = n(),
    Mean = mean(Value),
    SD = sd(Value),
    Min = min(Value),
    Q1 = quantile(Value, 0.25),
    Median = median(Value),
    Q3 = quantile(Value, 0.75),
    Max = max(Value)
  ) %>%
  mutate(across(where(is.numeric), ~ round(., 2)))

cat("### Descriptive Statistics of Key Quantitative Variables (for Distribution Description) ###\n")
```

```
## ### Descriptive Statistics of Key Quantitative Variables (for Distribution Description) ###
```

```r
print(summary_stats)
```

```
## # A tibble: 5 x 9
##   Variable           N   Mean    SD   Min    Q1 Median     Q3    Max
##   <chr>          <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1 log_budget      3229   16.8  1.67  0     16.2   17.0   17.8   19.8
## 2 log_popularity  3229   2.89  1.11 -3.91  2.35   3.02   3.62   6.77
## 3 log_revenue     3229   17.5  2.08  1.61 16.6   17.8   18.8   21.8
## 4 runtime         3229 111.   21.0  41     96    107    121    338
## 5 vote_average    3229   6.31  0.87  0      5.8    6.3    6.9    8.5
```

```r
df_genres <- df_final %>%
  mutate(
    main_genre = str_extract(genres, '"name":\\s*"([^"]+)"', group = 1)
  )

top_genres <- df_genres %>%
  count(main_genre, sort = TRUE) %>%
  slice_head(n = 5)

cat("### Top 5 Main Genres ###\n")
```

```
## ### Top 5 Main Genres ###
```

```r
print(top_genres)
```

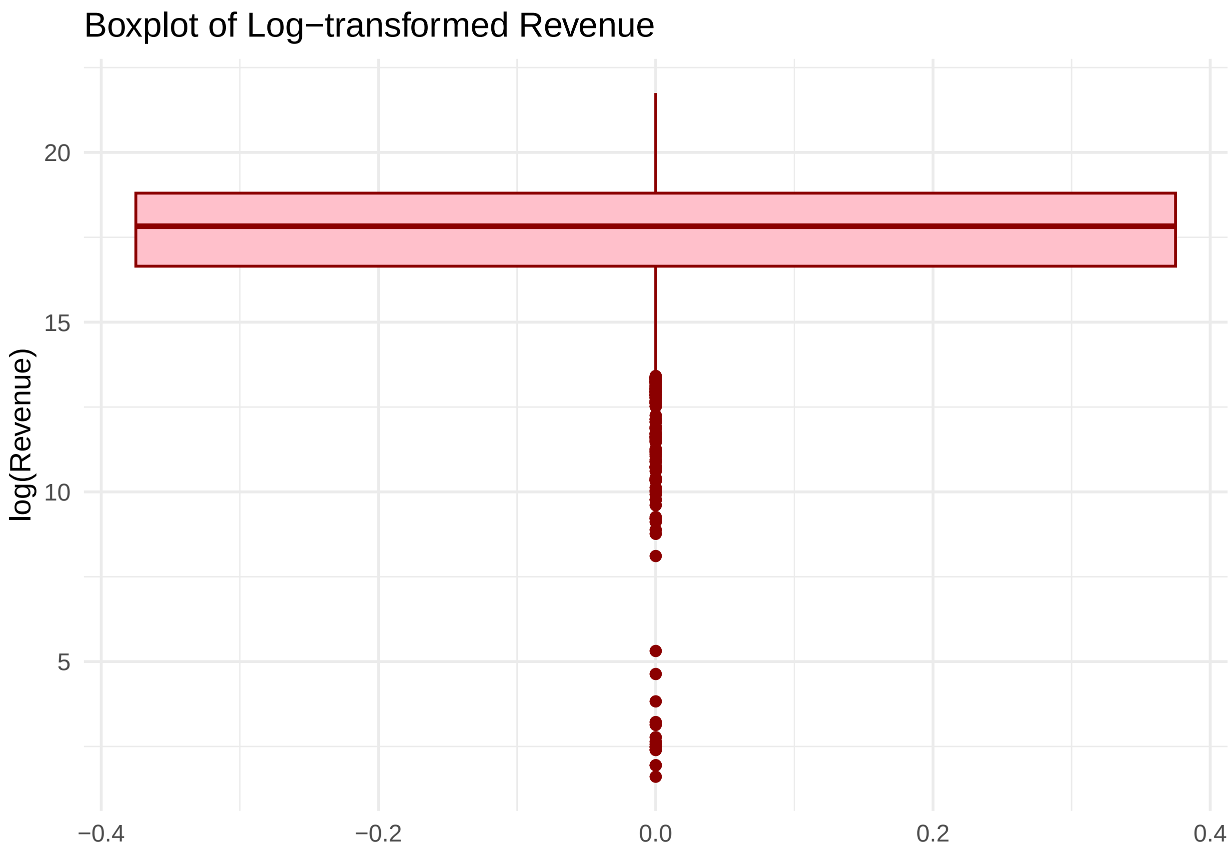
```
## # A tibble: 5 x 2
##   main_genre     n
##   <chr>      <int>
## 1 Drama        747
## 2 Comedy       634
## 3 Action       588
## 4 Adventure    288
## 5 Horror       197
```

```r
top_language <- df_final %>%
  count(original_language, sort = TRUE) %>%
  slice_head(n=5)

print(top_language)
```

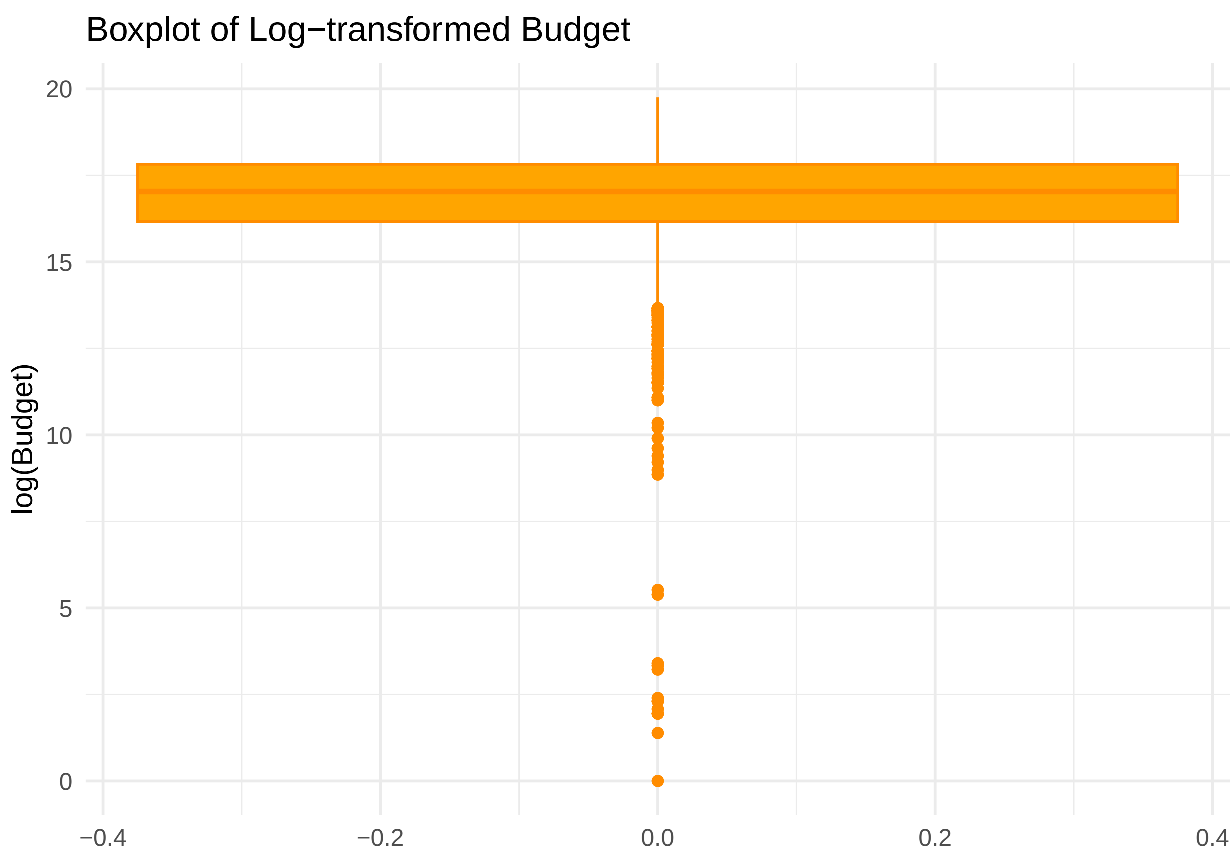
```
## # A tibble: 5 x 2
##   original_language     n
##   <chr>             <int>
## 1 en                 3102
## 2 fr                   25
## 3 es                   15
## 4 ja                   13
## 5 zh                   13
```

```r
p5 <- ggplot(df_final, aes(y = log_revenue)) +
  geom_boxplot(fill = "pink", color = "darkred") +
  labs(title = "Boxplot of Log-transformed Revenue",
       y = "log(Revenue)") +
  theme_minimal()
```

```r
print(p5)
```




```r
p6 <- ggplot(df_final, aes(y = log_budget)) +
  geom_boxplot(fill = "orange", color = "darkorange") +
  labs(title = "Boxplot of Log-transformed Budget",
       y = "log(Budget)") +
  theme_minimal()
print(p6)
```




```
df_genres <- df_final %>%
  mutate(
    main_genre = str_extract(genres, '"name":\\s*"([^"]+)"', group = 1)
  )

top_5_genres_names <- df_genres %>%
  count(main_genre, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(main_genre)


df_model_ready <- df_genres %>%
  mutate(
    Is_Drama = as.numeric(main_genre == top_5_genres_names[1]),
    Is_Comedy = as.numeric(main_genre == top_5_genres_names[2]),
    Is_Thriller = as.numeric(main_genre == top_5_genres_names[3]),
    Is_Action = as.numeric(main_genre == top_5_genres_names[4]),
    Is_Adventure = as.numeric(main_genre == top_5_genres_names[5]),

    Is_English = as.numeric(original_language == "en")
  ) %>%
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average,
         Is_Drama, Is_Comedy, Is_Thriller, Is_Action, Is_Adventure, Is_English)

cat("### Final Data Frame Variables for Model Fitting (First 3 Rows) ###\n")
```

```
## ### Final Data Frame Variables for Model Fitting (First 3 Rows) ###
```

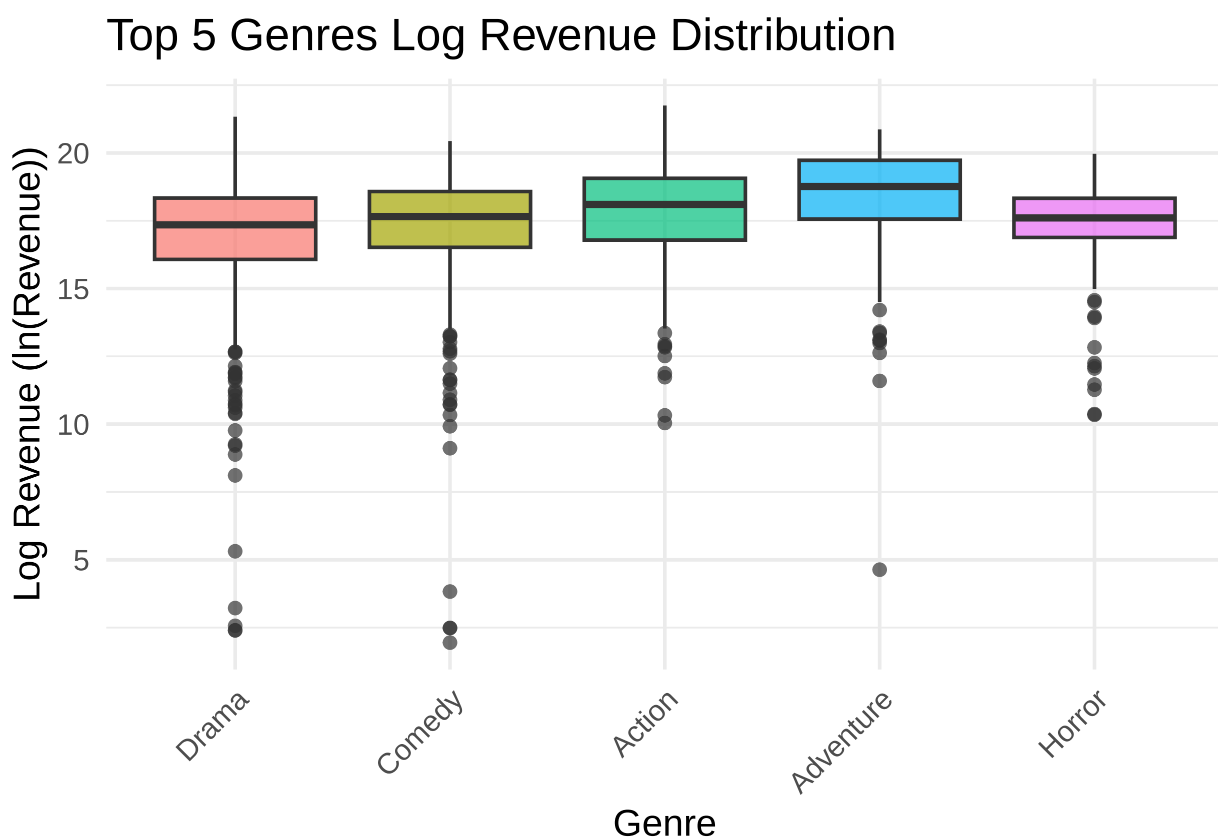
```r
print(head(df_model_ready, 3))
```

```
## # A tibble: 3 x 11
##   log_revenue log_budget log_popularity runtime vote_average Is_Drama Is_Comedy
##         <dbl>      <dbl>          <dbl>   <dbl>        <dbl>    <dbl>     <dbl>
## 1        21.7       19.3           5.01     162          7.2        0         0
## 2        20.7       19.5           4.94     169          6.9        0         0
## 3        20.6       19.3           4.68     148          6.3        0         0
## # i 4 more variables: Is_Thriller <dbl>, Is_Action <dbl>, Is_Adventure <dbl>,
## #   Is_English <dbl>
```

```r
# ------------------------------------------------------------
#   0 & 1:       (  )
# ------------------------------------------------------------
library(ggplot2)
library(dplyr)
library(stringr)

#     'data'

df_plot_ready <- data %>%
  filter(revenue > 0 & budget > 0) %>%
  mutate(log_revenue = log(revenue),
         #    Genre    (   JSON   )
         main_genre = str_extract(genres, '"name":\\s*"([^"]+)"', group = 1)) %>%
  dplyr::select(log_revenue, main_genre, original_language) %>%
  na.omit()

# ------------------------------------------------------------
#   2:   Top 10
# ------------------------------------------------------------
cat("---    Top 10    Log      ---\n")
```

```
## ---    Top 10    Log      ---
```

```r
#   Top 10
top_n_genres <- df_plot_ready %>%
  count(main_genre, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(main_genre)

#      (  )
df_genre_plot <- df_plot_ready %>%
  filter(main_genre %in% top_n_genres) %>%
  mutate(main_genre = factor(main_genre, levels = top_n_genres))

#
ggplot(df_genre_plot, aes(x = main_genre, y = log_revenue, fill = main_genre)) +
  geom_boxplot(alpha = 0.7) +
  labs(title = "Top 5 Genres Log Revenue Distribution",
       x = "Genre",
       y = "Log Revenue (ln(Revenue))") +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1), #   X
        legend.position = "none")
```

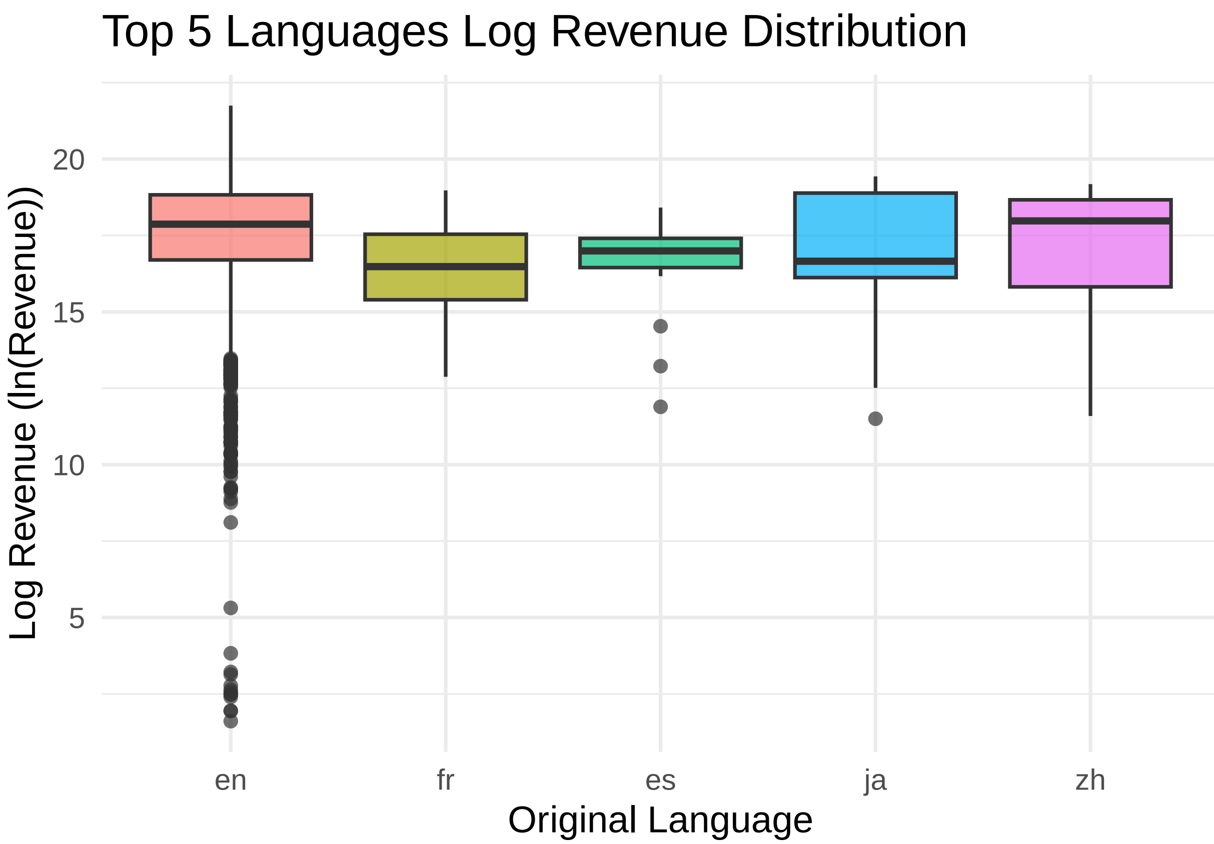



```
# ------------------------------------------------------------
#   0 & 1:       (  df_plot_ready  )
# ------------------------------------------------------------
library(ggplot2)
library(dplyr)
#              df_plot_ready
#          df_plot_ready

# ------------------------------------------------------------
#   2:   Top 5
# ------------------------------------------------------------
cat("---    Top 5    Log       ---\n")
```

```
## ---    Top 5    Log       ---
```

```
#   Top 5
top_n_languages <- df_plot_ready %>%
  count(original_language, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(original_language)

#      ( )
df_lang_plot <- df_plot_ready %>%
  filter(original_language %in% top_n_languages) %>%
  mutate(original_language = factor(original_language, levels = top_n_languages))

#
ggplot(df_lang_plot, aes(x = original_language, y = log_revenue, fill = original_language)) +
```

```r
  geom_boxplot(alpha = 0.7) +
  labs(title = "Top 5 Languages Log Revenue Distribution",
       x = "Original Language",
       y = "Log Revenue (ln(Revenue))") +
  theme_minimal(base_size = 14) +
  theme(legend.position = "none")
```




```r
# --------------------------------------------------------------
#   0:
# --------------------------------------------------------------
library(dplyr)
library(stringr)

#     'data'        revenue, budget, original_language, genres

# --------------------------------------------------------------
#   1:          (  Is_Adventure  )
# --------------------------------------------------------------
cat("---    1:     df_model_ready ---\n")
```

```
## ---     1:     df_model_ready ---
```

```r
# 1.       log
df_genres <- data %>%
  filter(revenue > 0 & budget > 0) %>%
  mutate(log_revenue = log(revenue),
         log_budget = log(budget),
         log_popularity = log(popularity),
```

```r
        #    Genre
        main_genre = str_extract(genres, '"name":\\s*"([^"]+)"', group = 1)
  )

# 2.    Top 5
top_5_genres_names <- df_genres %>%
  count(main_genre, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(main_genre)

cat("Top 5    : ")
```

```
## Top 5    :
```

```r
print(top_5_genres_names) #
```

```
## [1] "Drama"     "Comedy"    "Action"    "Adventure" "Horror"
```

```r
# 3.
df_model_ready <- df_genres %>%
  mutate(
    Is_Drama = as.numeric(main_genre == top_5_genres_names[1]),
    Is_Comedy = as.numeric(main_genre == top_5_genres_names[2]),
    Is_Thriller = as.numeric(main_genre == top_5_genres_names[3]),
    Is_Action = as.numeric(main_genre == top_5_genres_names[4]),
    Is_Adventure = as.numeric(main_genre == top_5_genres_names[5]), #    Is_Adventure

    Is_English = as.numeric(original_language == "en")
  ) %>%
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average,
         Is_Drama, Is_Comedy, Is_Thriller, Is_Action, Is_Adventure, Is_English) %>% #    Is_Adventure
  na.omit() #

cat("\n###    df_model_ready    ###\n")
```

```
##
## ###    df_model_ready    ###
```

```r
print(colnames(df_model_ready)) #     Is_Adventure
```

```
##  [1] "log_revenue"    "log_budget"     "log_popularity" "runtime"
##  [5] "vote_average"   "Is_Drama"       "Is_Comedy"      "Is_Thriller"
##  [9] "Is_Action"      "Is_Adventure"   "Is_English"
```

```r
cat("-----------------------------------------------------------\n")
```

```
## -----------------------------------------------------------
```

```r
# -----------------------------------------------------------
#    2:    Full Model (   Is_Adventure)
# -----------------------------------------------------------
cat("---    2:    Full Model (11    ) ---\n")
```

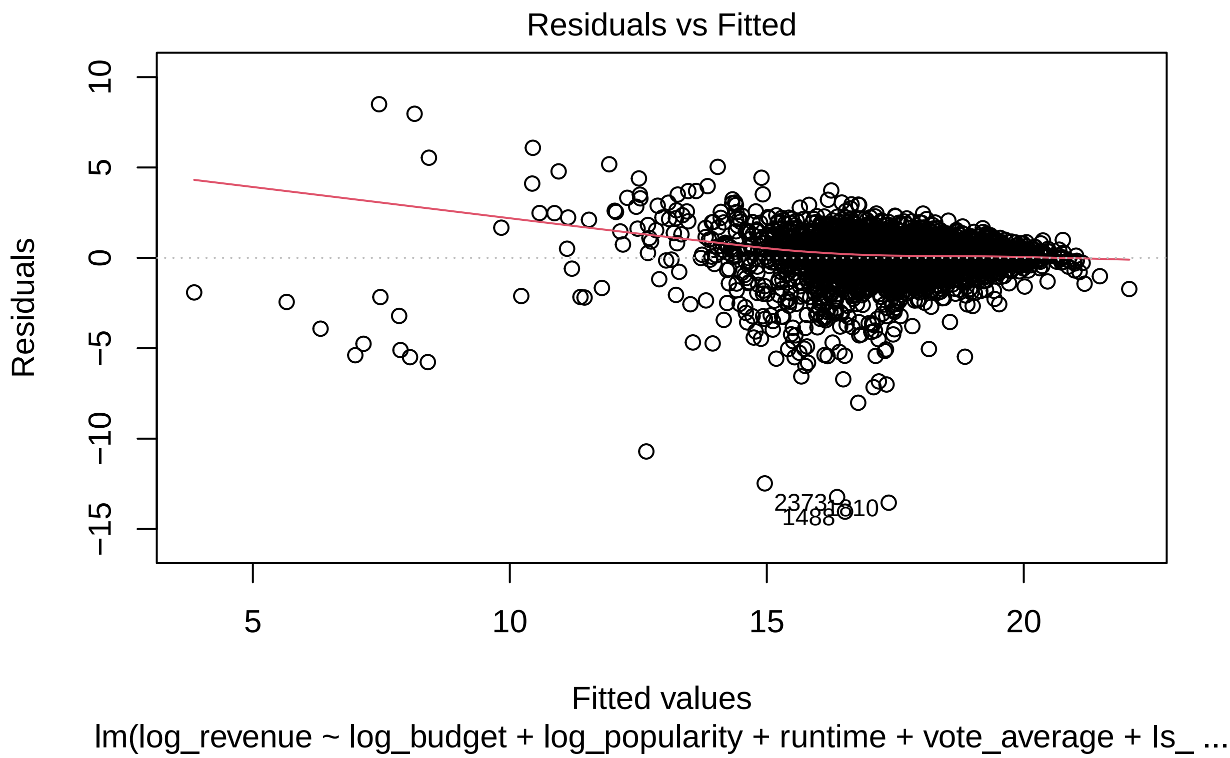
```
## ---    2:    Full Model (11    ) ---
```

```r
model_full <- lm(log_revenue ~ log_budget + log_popularity + runtime + vote_average +
                             Is_Drama + Is_Comedy + Is_Thriller + Is_Action + Is_Adventure + Is_English
                           data = df_model_ready)
```

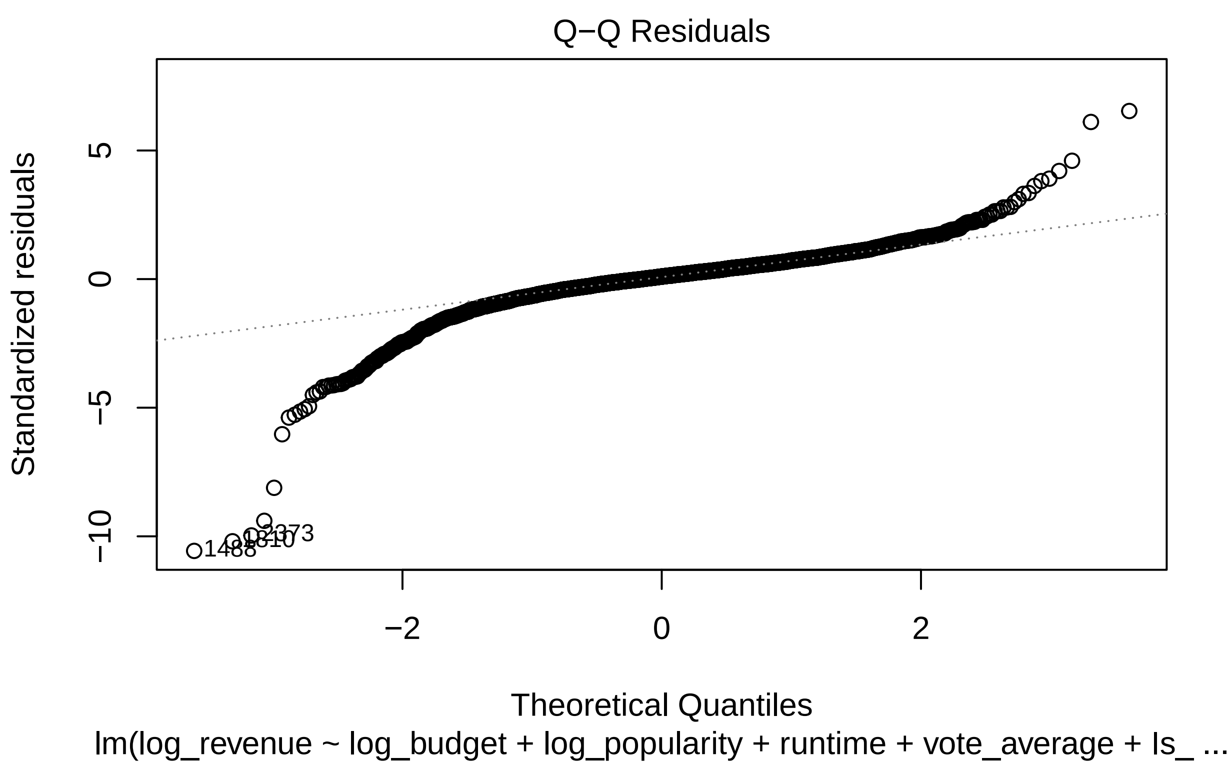
```r
summary(model_full)
```

```
##
## Call:
## lm(formula = log_revenue ~ log_budget + log_popularity + runtime +
##     vote_average + Is_Drama + Is_Comedy + Is_Thriller + Is_Action +
##     Is_Adventure + Is_English, data = df_model_ready)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0382  -0.4647   0.1325   0.6693   8.4997
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.074129   0.373457   8.232 2.65e-16 ***
## log_budget      0.627301   0.017389  36.075  < 2e-16 ***
## log_popularity  0.721439   0.026832  26.888  < 2e-16 ***
## runtime         0.001303   0.001303   1.000 0.317412
## vote_average    0.205524   0.033721   6.095 1.23e-09 ***
## Is_Drama       -0.077106   0.071231  -1.082 0.279125
## Is_Comedy       0.272767   0.072344   3.770 0.000166 ***
## Is_Thriller     0.012332   0.073994   0.167 0.867651
## Is_Action       0.204494   0.092653   2.207 0.027377 *
## Is_Adventure    0.560244   0.109532   5.115 3.32e-07 ***
## Is_English      0.269721   0.123394   2.186 0.028898 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.33 on 3217 degrees of freedom
## Multiple R-squared:  0.5926, Adjusted R-squared:  0.5913
## F-statistic: 467.9 on 10 and 3217 DF,  p-value: < 2.2e-16
```

```r
plot(model_full, which = 1)
```




```
plot(model_full, which = 2)
```




```
model_reduced <- lm(log_revenue ~ log_budget + log_popularity + vote_average +
                    Is_Comedy + Is_Action + Is_Adventure + Is_English,
                    data = df_model_ready)
```

```r
cat("### Reduced Model Fitting Results ###\n")
```

```
## ### Reduced Model Fitting Results ###
```

```r
summary(model_reduced)
```

```
##
## Call:
## lm(formula = log_revenue ~ log_budget + log_popularity + vote_average +
##     Is_Comedy + Is_Action + Is_Adventure + Is_English, data = df_model_ready)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0234  -0.4705   0.1361   0.6701   8.5043
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.07767    0.37135   8.288  < 2e-16 ***
## log_budget      0.63318    0.01673  37.837  < 2e-16 ***
## log_popularity  0.72560    0.02636  27.525  < 2e-16 ***
## vote_average    0.20853    0.03069   6.796 1.28e-11 ***
## Is_Comedy       0.28616    0.06125   4.672 3.10e-06 ***
## Is_Action       0.22492    0.08439   2.665  0.00774 **
## Is_Adventure    0.57089    0.10289   5.548 3.12e-08 ***
## Is_English      0.25960    0.12318   2.108  0.03515 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.33 on 3220 degrees of freedom
## Multiple R-squared:  0.5923, Adjusted R-squared:  0.5914
## F-statistic: 668.3 on 7 and 3220 DF,  p-value: < 2.2e-16
```

```r
anova_result <- anova(model_reduced, model_full)

cat("\n### F Test Results (ANOVA Comparison) ###\n")
```

```
##
## ### F Test Results (ANOVA Comparison) ###
```

```r
print(anova_result)
```

```
## Analysis of Variance Table
##
## Model 1: log_revenue ~ log_budget + log_popularity + vote_average + Is_Comedy +
##     Is_Action + Is_Adventure + Is_English
## Model 2: log_revenue ~ log_budget + log_popularity + runtime + vote_average +
##     Is_Drama + Is_Comedy + Is_Thriller + Is_Action + Is_Adventure +
##     Is_English
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1   3220 5696.4
## 2   3217 5692.4  3    3.9932 0.7522  0.521
```

```r
model_full_interaction <- lm(log_revenue ~ log_budget * (Is_Drama + Is_Comedy + Is_Thriller + Is_Action
                               log_popularity + vote_average + Is_English,
                             data = df_model_ready)
```

```r
cat("### Comprehensive Interaction ANCOVA Model (Log(Budget) x Top 4 Genres) ###\n")
```

```
## ### Comprehensive Interaction ANCOVA Model (Log(Budget) x Top 4 Genres) ###
```

```r
summary(model_full_interaction)
```

```
##
## Call:
## lm(formula = log_revenue ~ log_budget * (Is_Drama + Is_Comedy +
##     Is_Thriller + Is_Action + Is_Adventure) + log_popularity +
##     vote_average + Is_English, data = df_model_ready)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0812  -0.4637   0.1310   0.6648   8.5338
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.41583    0.54612   2.593  0.00957 **
## log_budget               0.72836    0.02862  25.452  < 2e-16 ***
## Is_Drama                 1.52782    0.62558   2.442  0.01465 *
## Is_Comedy                1.35007    0.79738   1.693  0.09053 .
## Is_Thriller              0.83637    0.95498   0.876  0.38120
## Is_Action                2.08118    1.02963   2.021  0.04333 *
## Is_Adventure             7.93218    0.96132   8.251 2.26e-16 ***
## log_popularity           0.69328    0.02697  25.705  < 2e-16 ***
## vote_average             0.23483    0.03162   7.427 1.42e-13 ***
## Is_English               0.25901    0.12266   2.112  0.03480 *
## log_budget:Is_Drama     -0.09500    0.03744  -2.537  0.01122 *
## log_budget:Is_Comedy    -0.06361    0.04755  -1.338  0.18106
## log_budget:Is_Thriller  -0.04942    0.05508  -0.897  0.36963
## log_budget:Is_Action    -0.10966    0.05880  -1.865  0.06228 .
## log_budget:Is_Adventure -0.45848    0.05931  -7.730 1.42e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.319 on 3213 degrees of freedom
## Multiple R-squared:  0.6001, Adjusted R-squared:  0.5983
## F-statistic: 344.3 on 14 and 3213 DF,  p-value: < 2.2e-16
```

```r
anova_result_joint_interaction <- anova(model_reduced, model_full_interaction)

cat("\n### F Test Results: Joint Significance of Log(Budget) and Top 4 Genres Interaction ###\n")
```

```
##
## ### F Test Results: Joint Significance of Log(Budget) and Top 4 Genres Interaction ###
```

```r
print(anova_result_joint_interaction)
```

```
## Analysis of Variance Table
##
## Model 1: log_revenue ~ log_budget + log_popularity + vote_average + Is_Comedy +
##     Is_Action + Is_Adventure + Is_English
## Model 2: log_revenue ~ log_budget * (Is_Drama + Is_Comedy + Is_Thriller +
##     Is_Action + Is_Adventure) + log_popularity + vote_average +
##     Is_English
```

```
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   3220 5696.4
## 2   3213 5588.1  7    108.26 8.8922 6.908e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# -------------------------------------------------------------
# COMPLETE R SCRIPT: OLS Prediction Intervals (Using Median Revenue Movie Data)
# -------------------------------------------------------------

# --- 0. Library Loading and Setup ---
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(dplyr)
library(stringr)
library(tidyverse)
set.seed(42)

# Helper function for currency formatting (for cleaner output)
format_currency <- function(x) {
    # Handles potential matrix output and formats to currency string
    format(round(as.numeric(x), 0), big.mark = ",", scientific = FALSE)
}


# --- 1. Data Loading and Feature Engineering (User's Code) ---
cat("--- 1. Data Loading and Preparation ---\n")
```

```
## --- 1. Data Loading and Preparation ---
```

```
path_final <- "/Users/linuohu/Desktop/467/project/tmdb_5000_movies.csv"
data <- read_csv(path_final)
```

```
## Rows: 4803 Columns: 20

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (12): genres, homepage, keywords, original_language, original_title, ov...
## dbl   (7): budget, id, popularity, revenue, runtime, vote_average, vote_count
## date  (1): release_date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Initial cleaning and log-transformation
df_final <- data %>%
  filter(revenue > 0 & budget > 0) %>%
  mutate(log_revenue = log(revenue),
         log_budget = log(budget),
         log_popularity = log(popularity)) %>%
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average, original_language, genre
```

```r
  na.omit()

# Genre processing and feature creation
df_genres <- df_final %>%
  mutate(main_genre = str_extract(genres, '"name":\\s*"([^"]+)"', group = 1))

top_5_genres_names <- df_genres %>%
  count(main_genre, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(main_genre)

# Final dataset for modeling (including actual revenue for output)
df_model_ready <- df_genres %>%
  mutate(
    Is_Drama = as.numeric(main_genre == top_5_genres_names[1]),
    Is_Comedy = as.numeric(main_genre == top_5_genres_names[2]),
    Is_Thriller = as.numeric(main_genre == top_5_genres_names[3]),
    Is_Action = as.numeric(main_genre == top_5_genres_names[4]),
    Is_English = as.numeric(original_language == "en")
  ) %>%
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average,
         Is_Drama, Is_Comedy, Is_Thriller, Is_Action, Is_English, revenue) %>%
    na.omit()


# --- 2. Define Input Data for Prediction (FINDING MEDIAN MOVIE) ---
cat("\n--- 2. Selecting Median Revenue Movie Data Point ---\n")
```

```
##
## --- 2. Selecting Median Revenue Movie Data Point ---
```

```r
# A. Find the movie closest to the median log_revenue
median_log_revenue <- median(df_model_ready$log_revenue)
median_index <- which.min(abs(df_model_ready$log_revenue - median_log_revenue))
representative_data <- df_model_ready[median_index, ]

# B. Extract features and actual revenue
actual_revenue_median <- representative_data$revenue
new_data <- representative_data %>%
    dplyr::select(log_budget, log_popularity, runtime, vote_average, Is_Drama, Is_Comedy, Is_Thriller, I

new_data_reduced <- new_data %>% dplyr::select(-runtime, -Is_Thriller)
cat(sprintf("Selected movie's Actual Revenue: %s\n", format_currency(actual_revenue_median)))
```

```
## Selected movie's Actual Revenue: 55,184,721
```

```r
# --- 3. OLS Modeling and Prediction ---
cat("\n--- 3. OLS Modeling and Prediction Intervals Calculation ---\n")
```

```
##
## --- 3. OLS Modeling and Prediction Intervals Calculation ---
```

```r
# Define Formulas
full_model_formula <- log_revenue ~ log_budget + log_popularity + runtime + vote_average +
                                    Is_Drama + Is_Comedy + Is_Thriller + Is_Action + Is_English
```

```r
reduced_model_formula <- log_revenue ~ log_budget + log_popularity + vote_average +
                                     Is_Drama + Is_Comedy + Is_Action + Is_English

# Fit OLS Models
model_full <- lm(full_model_formula, data = df_model_ready)
model_reduced <- lm(reduced_model_formula, data = df_model_ready)

# Calculate OLS Predictions (Prediction Interval)
pred_reduced <- predict(model_reduced, new_data_reduced, interval = "prediction", level = 0.95)
pred_full <- predict(model_full, new_data, interval = "prediction", level = 0.95)


# --- 4. Final Formatted Output ---
cat("\n=======================================================\n")
```

```
##
## =======================================================
```

```r
cat("  OLS 95% Prediction Intervals (PI) for Median Revenue Movie:\n")
```

```
##   OLS 95% Prediction Intervals (PI) for Median Revenue Movie:
```

```r
cat("=======================================================\n")
```

```
## =======================================================
```

```r
cat(sprintf("   - Actual Revenue of Selected Movie: %s\n", format_currency(actual_revenue_median)))
```

```
##    - Actual Revenue of Selected Movie: 55,184,721
```

```r
cat("-------------------------------------------------------\n")
```

```
## -------------------------------------------------------
```

```r
# --- OLS Reduced Model Output (USD Scale) ---
usd_reduced <- exp(pred_reduced)
cat("\n--- OLS REDUCED MODEL ---\n")
```

```
##
## --- OLS REDUCED MODEL ---
```

```r
cat(sprintf("   - Predicted Revenue (Point Estimate): %s\n", format_currency(usd_reduced[1, "fit"])))
```

```
##    - Predicted Revenue (Point Estimate): 7,555,168
```

```r
cat(sprintf("   - 95%% PI Lower Bound:               %s\n", format_currency(usd_reduced[1, "lwr"])))
```

```
##    - 95% PI Lower Bound:               548,653
```

```r
cat(sprintf("   - 95%% PI Upper Bound:               %s\n", format_currency(usd_reduced[1, "upr"])))
```

```
##    - 95% PI Upper Bound:               104,037,735
```

```r
cat(sprintf("   - PI Width:                        %s\n", format_currency(usd_reduced[1, "upr"] - usd_
```

```
##    - PI Width:                        103,489,082
```

```r
cat("-------------------------------------------------------\n")
```

```
## -------------------------------------------------------
```

```r
# --- OLS Full Model Output (USD Scale) ---
usd_full <- exp(pred_full)
cat("\n--- OLS FULL MODEL ---\n")
```

```
##
## --- OLS FULL MODEL ---
```

```r
cat(sprintf("   - Predicted Revenue (Point Estimate): %s\n", format_currency(usd_full[1, "fit"])))
```

```
##    - Predicted Revenue (Point Estimate): 7,622,889
```

```r
cat(sprintf("   - 95%% PI Lower Bound:                %s\n", format_currency(usd_full[1, "lwr"])))
```

```
##    - 95% PI Lower Bound:                553,605
```

```r
cat(sprintf("   - 95%% PI Upper Bound:                %s\n", format_currency(usd_full[1, "upr"])))
```

```
##    - 95% PI Upper Bound:                104,963,777
```

```r
cat(sprintf("   - PI Width:                          %s\n", format_currency(usd_full[1, "upr"] - usd_full
```

```
##    - PI Width:                          104,410,173
```

```r
cat("=======================================================\n")
```

```
## =======================================================
```

```r
# ------------------------------------------------------------
#   0:
# ------------------------------------------------------------
# ------------------------------------------------------------
#   0:
# ------------------------------------------------------------
library(MASS)     #   stepAIC
library(dplyr)
library(sandwich)
library(lmtest)
library(stringr)  #     genres
set.seed(42)      #

# ------------------------------------------------------------
#   1:         (  Is_Adventure)
# ------------------------------------------------------------
cat("---   1:       (  Is_Adventure) ---\n")
```

```
## ---   1:       (  Is_Adventure) ---
```

```r
#     'data'

#   5
df_temp <- data %>%
  filter(revenue > 0 & budget > 0) %>%
  mutate(main_genre = str_extract(genres, '"name":\\s*"([^"]+)"', group = 1))

top_5_genres_names <- df_temp %>%
  count(main_genre, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(main_genre) #   Top 5 Genre
```

```r
#      Is_Adventure
cat("Top 5   : ")
```

```
## Top 5   :
```

```r
print(top_5_genres_names)
```

```
## [1] "Drama"     "Comedy"    "Action"    "Adventure" "Horror"
```

```r
#       df_model_ready
df_model_ready <- df_temp %>%
  mutate(log_revenue = log(revenue),
         log_budget = log(budget),
         log_popularity = log(popularity),

         #       Top 5      (     )
         Is_Drama = as.numeric(main_genre == top_5_genres_names[1]),
         Is_Comedy = as.numeric(main_genre == top_5_genres_names[2]),
         Is_Thriller = as.numeric(main_genre == top_5_genres_names[3]),
         Is_Action = as.numeric(main_genre == top_5_genres_names[4]),
         Is_Adventure = as.numeric(main_genre == top_5_genres_names[5]), #   Is_Adventure

         Is_English = as.numeric(original_language == "en")
  ) %>%
  #      Is_Adventure     (      Is_Adventu  )
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average,
                Is_Drama, Is_Comedy, Is_Thriller, Is_Action, Is_Adventure, Is_English) %>%
  na.omit()

# -------------------------------------------------------------
#   2:
# -------------------------------------------------------------
cat("\n---   2:     ---\n")
```

```
##
## ---   2:     ---
```

```r
train_size <- floor(0.80 * nrow(df_model_ready))
train_indices <- sample(seq_len(nrow(df_model_ready)), size = train_size)
train_data <- df_model_ready[train_indices, ]

# 3. Full Model   (10   : 4     + 5    + 1   )
full_model_formula <- log_revenue ~ log_budget + log_popularity + runtime + vote_average +
                                    Is_Drama + Is_Comedy + Is_Thriller + Is_Action + Is_Adventure + Is_


# -------------------------------------------------------------
#   I: OLS Full Model    (   WLS    )
# -------------------------------------------------------------
cat("\n---   I: OLS Full Model      ---\n")
```

```
##
## ---   I: OLS Full Model      ---
```

```r
# 1.   OLS Full Model
ols_full <- lm(full_model_formula, data = train_data)
```

```r
# 2.     (  Full Model   )
res_sq_full <- residuals(ols_full)^2
res_sq_full[res_sq_full < 1e-6] <- 1e-6
variance_model_data <- cbind(train_data, res_sq = res_sq_full)

# 3.
variance_model_full <- lm(log(res_sq) ~ log_budget + log_popularity + runtime + vote_average,
                          data = variance_model_data)

# 4.   WLS
variance_estimates_full <- exp(predict(variance_model_full, newdata = train_data))
wls_weights_full <- 1 / variance_estimates_full

# -------------------------------------------------------------
#   II & III: WLS Full Model   & AIC
# -------------------------------------------------------------

# 1.   WLS Full Model
model_wls_full <- lm(full_model_formula, data = train_data, weights = wls_weights_full)

# 2.   stepAIC   WLS       (direction = "backward")
wls_step_model <- stepAIC(model_wls_full, direction = "backward", trace = 1)
```

```
## Start:  AIC=4268.63
## log_revenue ~ log_budget + log_popularity + runtime + vote_average +
##     Is_Drama + Is_Comedy + Is_Thriller + Is_Action + Is_Adventure +
##     Is_English
##
##                  Df Sum of Sq   RSS    AIC
## - runtime         1       0.1 13374 4266.6
## - Is_Action       1       0.4 13374 4266.7
## - Is_Thriller     1       8.7 13382 4268.3
## <none>                        13373 4268.6
## - Is_English      1      14.9 13388 4269.5
## - Is_Drama        1      25.8 13399 4271.6
## - Is_Comedy       1      57.1 13430 4277.6
## - Is_Adventure    1      59.3 13433 4278.1
## - vote_average    1     158.6 13532 4297.1
## - log_popularity  1    3279.7 16653 4832.9
## - log_budget      1    4004.5 17378 4942.9
##
## Step:  AIC=4266.65
## log_revenue ~ log_budget + log_popularity + vote_average + Is_Drama +
##     Is_Comedy + Is_Thriller + Is_Action + Is_Adventure + Is_English
##
##                  Df Sum of Sq   RSS    AIC
## - Is_Action       1       0.5 13374 4264.7
## - Is_Thriller     1       8.6 13382 4266.3
## <none>                        13374 4266.6
## - Is_English      1      14.8 13388 4267.5
## - Is_Drama        1      26.7 13400 4269.8
## - Is_Comedy       1      57.2 13431 4275.7
## - Is_Adventure    1      59.5 13433 4276.1
## - vote_average    1     184.4 13558 4300.0
```

```
## - log_popularity  1    3280.8 16654 4831.1
## - log_budget      1    4330.5 17704 4988.9
##
## Step:  AIC=4264.74
## log_revenue ~ log_budget + log_popularity + vote_average + Is_Drama +
##     Is_Comedy + Is_Thriller + Is_Adventure + Is_English
##
##                  Df Sum of Sq   RSS    AIC
## <none>                        13374 4264.7
## - Is_Thriller     1      12.2 13386 4265.1
## - Is_English      1      14.7 13389 4265.6
## - Is_Drama        1      32.3 13406 4269.0
## - Is_Adventure    1      59.5 13434 4274.2
## - Is_Comedy       1      60.0 13434 4274.3
## - vote_average    1     184.3 13558 4298.1
## - log_popularity  1    3284.3 16658 4829.7
## - log_budget      1    4404.6 17779 4997.8
```

```r
# trace = 1      AIC

# ------------------------------------------------------------
#   IV:
# ------------------------------------------------------------
cat("WLS Stepwise Selection :\n")
```

```
## WLS Stepwise Selection :
```

```r
print(formula(wls_step_model))
```

```
## log_revenue ~ log_budget + log_popularity + vote_average + Is_Drama +
##     Is_Comedy + Is_Thriller + Is_Adventure + Is_English
```

```r
#      HC3
hc_vcov_wls_final <- vcovHC(wls_step_model, type = "HC3")
print(coeftest(wls_step_model, vcov. = hc_vcov_wls_final))
```

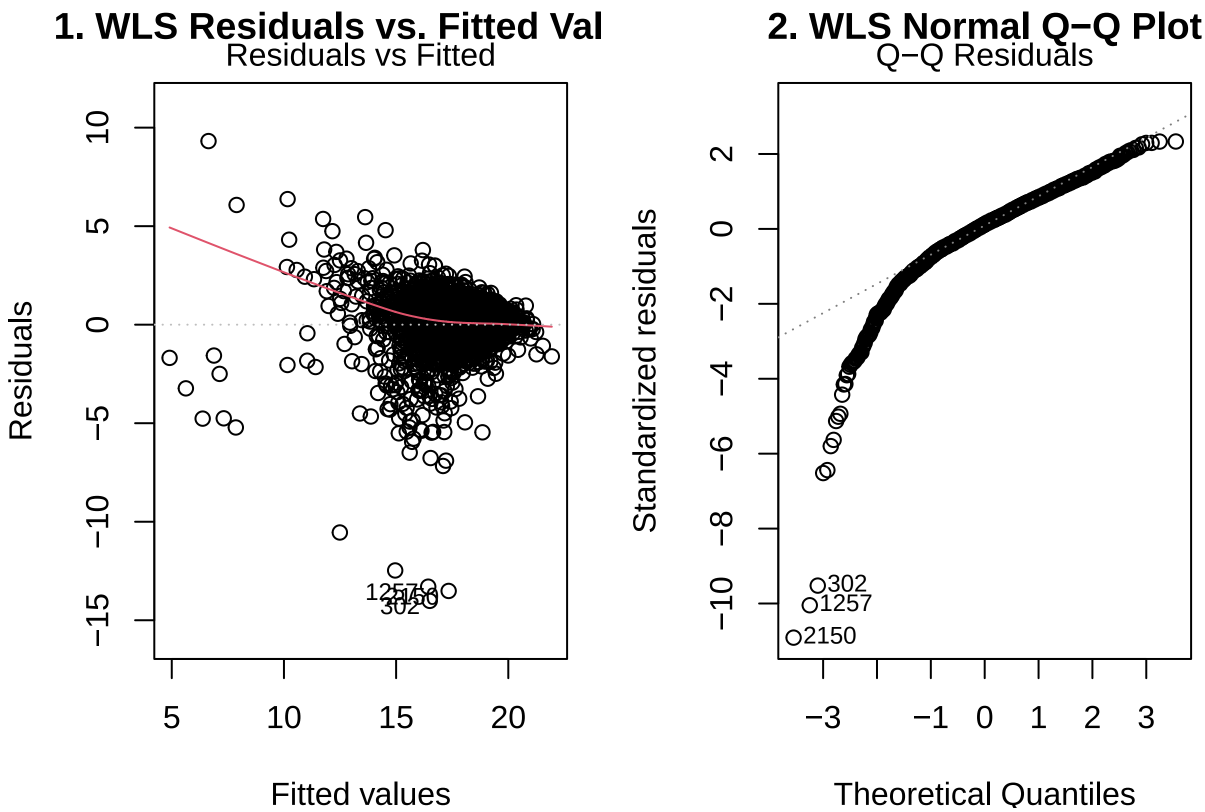
```
##
## t test of coefficients:
##
##                 Estimate Std. Error t value  Pr(>|t|)
## (Intercept)     2.656481   0.498100  5.3332 1.049e-07 ***
## log_budget      0.674046   0.025475 26.4586 < 2.2e-16 ***
## log_popularity  0.738797   0.034027 21.7118 < 2.2e-16 ***
## vote_average    0.176266   0.032803  5.3735 8.416e-08 ***
## Is_Drama       -0.151819   0.059522 -2.5506 0.0108107 *
## Is_Comedy       0.210269   0.061589  3.4141 0.0006499 ***
## Is_Thriller    -0.082480   0.051511 -1.6012 0.1094565
## Is_Adventure    0.355088   0.091403  3.8849 0.0001050 ***
## Is_English      0.236560   0.201633  1.1732 0.2408148
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# ------------------------------------------------------------
#   I:
# ------------------------------------------------------------
#   wls_step_model    AIC     WLS
```

```r
cat("--- WLS        ---\n")
```

```
## --- WLS        ---
```

```r
#   par()     1   x 2
par(mfrow = c(1, 2), mar = c(4.5, 4.5, 3, 1))

# ------------------------------------------------------------
#   II:       (   xlab   ylab)
# ------------------------------------------------------------

#   1: Residuals vs. Fitted (WLS   vs.   ) -
plot(wls_step_model, which = 1,
     main = "1. WLS Residuals vs. Fitted Values")
     #   xlab/ylab

#   2: Normal Q-Q Plot (  Q-Q  ) -
plot(wls_step_model, which = 2,
     main = "2. WLS Normal Q-Q Plot")
```




```r
     #   xlab/ylab

# ------------------------------------------------------------
#   III:
# ------------------------------------------------------------

#
par(mfrow = c(1, 1))
```

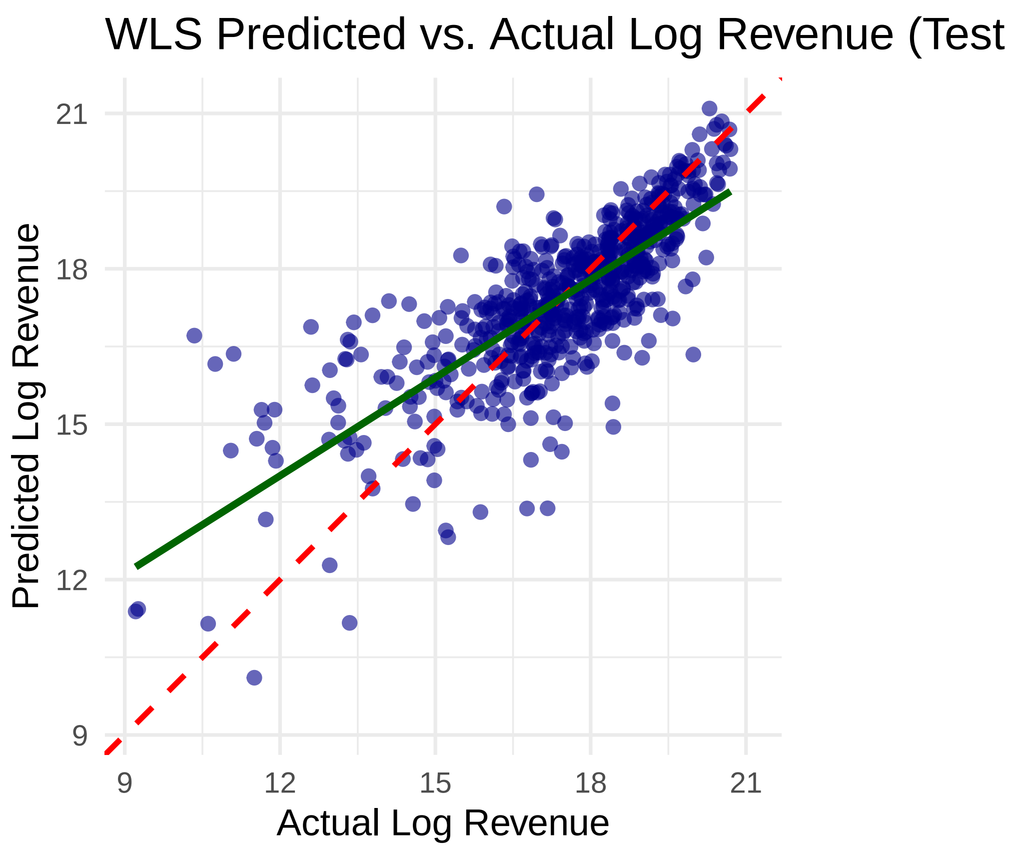
```r
cat("\nWLS        Q-Q    R      \n")
```

```
##
## WLS        Q-Q    R
```

```r
# ------------------------------------------------------------
#    :     'data'     R
# ------------------------------------------------------------
library(ggplot2)
library(dplyr)
library(stringr)
set.seed(42) #

# 1.
df_model_ready <- data %>%
  filter(revenue > 0 & budget > 0) %>%
  mutate(log_revenue = log(revenue),
         log_budget = log(budget),
         log_popularity = log(popularity),
         #
         Is_Drama = as.numeric(str_detect(genres, "Drama")),
         Is_Comedy = as.numeric(str_detect(genres, "Comedy")),
         Is_Thriller = as.numeric(str_detect(genres, "Thriller")),
         Is_Action = as.numeric(str_detect(genres, "Action")),
         Is_English = as.numeric(original_language == "en")
  ) %>%
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average,
              Is_Drama, Is_Comedy, Is_Thriller, Is_Action, Is_English) %>%
  na.omit()

# 2.       (  test_data  )
train_indices <- sample(seq_len(nrow(df_model_ready)), size = floor(0.80 * nrow(df_model_ready)))
train_data <- df_model_ready[train_indices, ]
test_data <- df_model_ready[-train_indices, ]

# 3.   WLS   (  wls_step_model   )
model_formula_simplified <- log_revenue ~ log_budget + log_popularity + runtime + vote_average +
                                          Is_Drama + Is_Comedy + Is_Thriller #    7

# WLS
ols_simplified <- lm(model_formula_simplified, data = train_data)
res_sq <- residuals(ols_simplified)^2
res_sq[res_sq < 1e-6] <- 1e-6
variance_model_data <- cbind(train_data, res_sq = res_sq)
variance_model <- lm(log(res_sq) ~ log_budget + log_popularity + runtime + vote_average, data = varianc
weights <- 1 / exp(predict(variance_model, newdata = train_data))
wls_step_model <- lm(model_formula_simplified, data = train_data, weights = weights)

# ------------------------------------------------------------
# 4.     vs.     (   )
# ------------------------------------------------------------

predicted_values_wls <- predict(wls_step_model, newdata = test_data)
```

```r
plot_data_wls <- data.frame(
  Actual = test_data$log_revenue,
  Predicted = predicted_values_wls
)

plot_range <- range(c(plot_data_wls$Actual, plot_data_wls$Predicted), na.rm = TRUE)

#
ggplot(plot_data_wls, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.6, color = "darkblue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red", size = 1) +
  geom_smooth(method = "lm", color = "darkgreen", se = FALSE) +
  labs(title = "WLS Predicted vs. Actual Log Revenue (Test Set)",
       x = "Actual Log Revenue",
       y = "Predicted Log Revenue") +
  coord_fixed(xlim = plot_range, ylim = plot_range) +
  theme_minimal(base_size = 14)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
# ------------------------------------------------------------
# COMPLETE R SCRIPT: WLS Prediction Intervals (Using Actual Median Revenue Movie)
# ------------------------------------------------------------

# --- 0. Library Loading and Setup ---
library(MASS)       # For stepAIC
library(dplyr)      # Data manipulation
```

```r
library(stringr)   # For processing 'genres' string
library(tidyverse) # Includes all necessary packages like readr

set.seed(42)       # For reproducibility

# Helper function for currency formatting
format_currency <- function(x) {
    format(round(as.numeric(x), 0), big.mark = ",", scientific = FALSE)
}

# ---  Data Loading Placeholder (User Provided)  ---
# Assuming 'data' object is loaded by the user's code before this script runs.
# We skip the explicit data loading lines here as they rely on a local path.

if (!exists("data") || nrow(data) == 0) {
    # This block is only for safety, assuming the user will load the data.
    stop("Error: 'data' object is missing or empty. Please load your dataset first.")
}

# --- 1. Data Preparation and Feature Engineering ---
cat("--- 1. Data Preparation and Feature Engineering ---\n")
```

```
## --- 1. Data Preparation and Feature Engineering ---
```

```r
# Filter, create log variables, and extract main genre
df_temp <- data %>%
  filter(revenue > 0 & budget > 0) %>%
  mutate(main_genre = str_extract(genres, '"name":\\s*"([^"]+)"', group = 1))

# Identify Top 5 genres
top_5_genres_names <- df_temp %>%
  filter(!is.na(main_genre)) %>%
  count(main_genre, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(main_genre)

# Create final modeling dataset
df_model_ready <- df_temp %>%
  mutate(
    log_revenue = log(revenue),
    log_budget = log(budget),
    log_popularity = log(popularity),
    # Create binary variables for Top 5 genres
    Is_Drama     = as.numeric(main_genre == top_5_genres_names[1]),
    Is_Comedy    = as.numeric(main_genre == top_5_genres_names[2]),
    Is_Thriller  = as.numeric(main_genre == top_5_genres_names[3]),
    Is_Action    = as.numeric(main_genre == top_5_genres_names[4]),
    Is_Adventure = as.numeric(main_genre == top_5_genres_names[5]),
    Is_English   = as.numeric(original_language == "en")
  ) %>%
  dplyr::select(log_revenue, log_budget, log_popularity, runtime, vote_average,
              Is_Drama, Is_Comedy, Is_Thriller, Is_Action, Is_Adventure, Is_English, revenue) %>%
  na.omit()
```

```r
# --- 2. Data Splitting ---
cat("\n--- 2. Data Splitting ---\n")
```

```
##
## --- 2. Data Splitting ---
```

```r
train_size <- floor(0.80 * nrow(df_model_ready))
train_indices <- sample(seq_len(nrow(df_model_ready)), size = train_size)
train_data <- df_model_ready[train_indices, ]
test_data <- df_model_ready[-train_indices, ] # Test data for prediction

# Full Model Formula
full_model_formula <- log_revenue ~ log_budget + log_popularity + runtime + vote_average +
                                    Is_Drama + Is_Comedy + Is_Thriller + Is_Action + Is_Adventure + Is_En


# --- 3. Stage I: OLS Full Model & Variance Model Estimation ---
cat("\n--- 3. WLS Stage I & II: Variance Model Estimation (on Training Data) ---\n")
```

```
##
## --- 3. WLS Stage I & II: Variance Model Estimation (on Training Data) ---
```

```r
# 1. Fit OLS Full Model (on training data)
ols_full <- lm(full_model_formula, data = train_data)

# 2. Estimate variance structure (squared residuals)
res_sq_full <- residuals(ols_full)^2
res_sq_full[res_sq_full < 1e-6] <- 1e-6 # Avoid log(0)
variance_model_data <- cbind(train_data, res_sq = res_sq_full)

# 3. Fit Variance Model (Predicts log(sigma_i^2) using continuous predictors)
variance_model_full <- lm(log(res_sq) ~ log_budget + log_popularity + runtime + vote_average,
                          data = variance_model_data)


# --- 4. Stage III & IV: WLS Modeling and Stepwise Selection ---
cat("\n--- 4. WLS Stage III & IV: Stepwise Selection ---\n")
```

```
##
## --- 4. WLS Stage III & IV: Stepwise Selection ---
```

```r
# 1. Calculate WLS weights (w_i = 1 / sigma_i^2)
variance_estimates_full <- exp(predict(variance_model_full, newdata = train_data))
wls_weights_full <- 1 / variance_estimates_full

# 2. Fit WLS Full Model
model_wls_full <- lm(full_model_formula, data = train_data, weights = wls_weights_full)

# 3. Perform stepAIC backward selection to get the final WLS model
wls_step_model <- stepAIC(model_wls_full, direction = "backward", trace = 0)

cat("Final WLS Model Formula:\n")
```

```
## Final WLS Model Formula:
```

```r
print(formula(wls_step_model))
```

```
## log_revenue ~ log_budget + log_popularity + vote_average + Is_Drama +
##     Is_Comedy + Is_Thriller + Is_Adventure + Is_English
```

```r
# --- 5. WLS Prediction Interval (PI) Calculation on Test Data ---
cat("\n--- 5. WLS Prediction Interval Calculation (on entire Test Data) ---\n")
```

```
##
## --- 5. WLS Prediction Interval Calculation (on entire Test Data) ---
```

```r
# A. Point Prediction and SE of the Mean Prediction (SE(Y_hat))
wls_predictions <- predict(
    wls_step_model,
    newdata = test_data,
    se.fit = TRUE
)

# B. Estimate Observation Error Variance (sigma_i^2) for Test Data
estimated_variance <- exp(predict(
    variance_model_full,
    newdata = test_data
))

# C. Calculate Full SE of the Prediction Error
se_prediction_error <- sqrt(
    wls_predictions$se.fit^2 + estimated_variance
)

# D. Determine t-score
alpha <- 0.05
df <- wls_step_model$df.residual
t_score <- qt(1 - alpha/2, df = df)

# E. Calculate 95% PI (Log Scale)
PI_lower_log <- wls_predictions$fit - t_score * se_prediction_error
PI_upper_log <- wls_predictions$fit + t_score * se_prediction_error

# F. Combine and Convert to Dollar Scale (exp())
wls_pi_results <- data.frame(
    Actual_Revenue = test_data$revenue, # Actual value in dollar scale
    log_revenue_actual = test_data$log_revenue,
    Predicted_log = wls_predictions$fit,
    PI_Lower_log = PI_lower_log,
    PI_Upper_log = PI_upper_log
) %>%
    mutate(
        Predicted_Revenue = exp(Predicted_log),
        PI_Lower_Revenue = exp(PI_Lower_log),
        PI_Upper_Revenue = exp(PI_Upper_log)
    )


# --- 6. Final Single-Point Representative Output (Actual Median Movie) ---
cat("\n 6. Final Single-Point Representative Output (Actual Median Movie) ---\n")
```

```
##
## 6. Final Single-Point Representative Output (Actual Median Movie) ---
```

```r
# 1. Find the observation in the test set closest to the actual median log_revenue
median_actual_log <- median(wls_pi_results$log_revenue_actual)
median_index <- which.min(abs(wls_pi_results$log_revenue_actual - median_actual_log))
representative_row <- wls_pi_results[median_index, ]

# 2. Format output
predicted_usd <- representative_row$Predicted_Revenue
lower_usd <- representative_row$PI_Lower_Revenue
upper_usd <- representative_row$PI_Upper_Revenue
actual_usd <- representative_row$Actual_Revenue


cat("WLS 95% Prediction Interval for Median Actual Revenue Movie:\n")
```

```
## WLS 95% Prediction Interval for Median Actual Revenue Movie:
```

```r
cat(sprintf("   - Actual Revenue of Selected Movie: %s\n", format_currency(actual_usd)))
```

```
##    - Actual Revenue of Selected Movie: 53,208,180
```

```r
cat(sprintf("   - Predicted Revenue (Point Estimate): %s\n", format_currency(predicted_usd)))
```

```
##    - Predicted Revenue (Point Estimate): 57,464,030
```

```r
cat(sprintf("   - 95%% PI Lower Bound:               %s\n", format_currency(lower_usd)))
```

```
##    - 95% PI Lower Bound:               22,668,093
```

```r
cat(sprintf("   - 95%% PI Upper Bound:               %s\n", format_currency(upper_usd)))
```

```
##    - 95% PI Upper Bound:               145,672,368
```

```r
cat(sprintf("   - PI Width:                         %s\n", format_currency(upper_usd - lower_usd)))
```

```
##    - PI Width:                         123,004,275
```