Advertising_EDA_Report

2025-01-30

Libraries and Imports

Data Import

```
#load staging data from SQL analysis
path <- "C:\\Users\\jbeas\\OneDrive\\Desktop\\Projects\\Advertising\\ad_staging.csv"
orig <- read.csv(path)

#additional stats from SQL analysis
stats_path <- "C:\\Users\\jbeas\\OneDrive\\Desktop\\Projects\\Advertising\\summary_stats.csv"
freq_path <- "C:\\Users\\jbeas\\OneDrive\\Desktop\\Projects\\Advertising\\freq_dist.csv"
mvals_path <- "C:\\Users\\jbeas\\OneDrive\\Desktop\\Projects\\Advertising\\missing_vals.csv"

data_stats <- read.csv(stats_path)
data_freq <- read.csv(freq_path)
data_mvals <- read.csv(mvals_path)

# verify data imported
head(orig)
```

```
##   Ad_ID Ad_Type Visual_Complexity Clicks Time_Spent Engagement_Score Age_Group
## 1     1      AR              High    238         41               52     18-24
## 2     2      2D            Medium    116         44               87     35-44
## 3     3      AR            Medium    300         23               90     25-34
## 4     4      3D              High     65        120               61     18-24
## 5     5      AR               Low     92         65               93     25-34
## 6     6      2D              High    273         63               80     25-34
##   Gender Device_Type Conversion_Rate Bounce_Rate   CTR Frame_Data
## 1      F     Desktop            8.72       25.92  5.09    frame_5
## 2      M      Mobile            7.99        9.05  9.02    frame_5
## 3      F      Mobile            1.65       11.32  7.57   frame_10
## 4      M     Desktop            6.22       39.11  3.64    frame_4
## 5      F      Tablet            8.31       15.94 14.97    frame_8
## 6      M     Desktop            8.06       25.38  2.92    frame_4
##   User_Movement_Data Age_Group_Numeric Movement_Numeric
## 1        no movement                 1                1
## 2     gaze, movement                 3                3
## 3      movement only                 2                2
## 4      movement only                 1                2
## 5        no movement                 2                1
## 6        no movement                 2                1
```

```
##   Visual_Complexity_Numeric Ad_Type_Numeric
## 1                         3               3
## 2                         2               1
## 3                         2               3
## 4                         3               2
## 5                         1               3
## 6                         3               1
```

```
#create copy of data
data <- orig
```

EDA

Summary Stats

```
#check for missing vals
data_mvals
```

```
##   Total_rows Missing_Ad_ID Missing_Ad_Type Missing_Visual_Complexity
## 1       1000             0               0                         0
##   Missing_Clicks Missing_Time_Spent Missing_Engagement_Score Missing_Age_Group
## 1              0                  0                        0                 0
##   Missing_Gender Missing_Device_Type Missing_Conversion_Rate
## 1              0                   0                       0
##   Missing_Bounce_Rate Missing_CTR Missing_Frame_Data Missing_User_Movement_Data
## 1                   0           0                  0                          0
```

```
#summary stats
data_stats
```

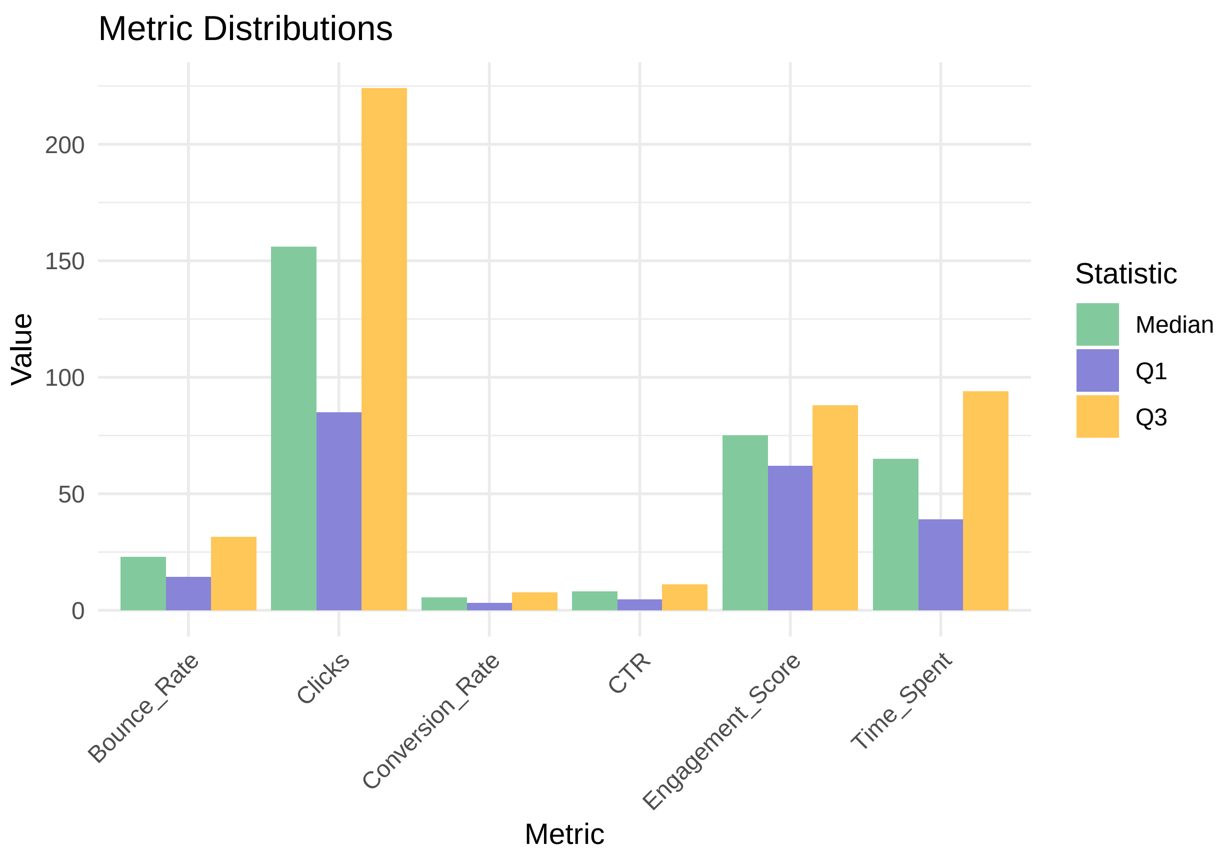
```
##             Metric   Mean Std_Dev Min_Value    Q1 Median     Q3 Max_Value
## 1      Bounce_Rate  22.73    9.91      5.00 14.21  22.90  31.48     39.99
## 2           Clicks 154.91   82.56     10.00 85.00 156.00 224.00    300.00
## 3  Conversion_Rate   5.47    2.60      1.00  3.17   5.48   7.71      9.98
## 4              CTR   7.97    3.93      1.03  4.64   8.03  11.09     14.99
## 5 Engagement_Score  74.87   14.80     50.00 62.00  75.00  88.00    100.00
## 6       Time_Spent  65.81   31.86     10.00 39.00  65.00  94.00    120.00
```

```
data_freq
```

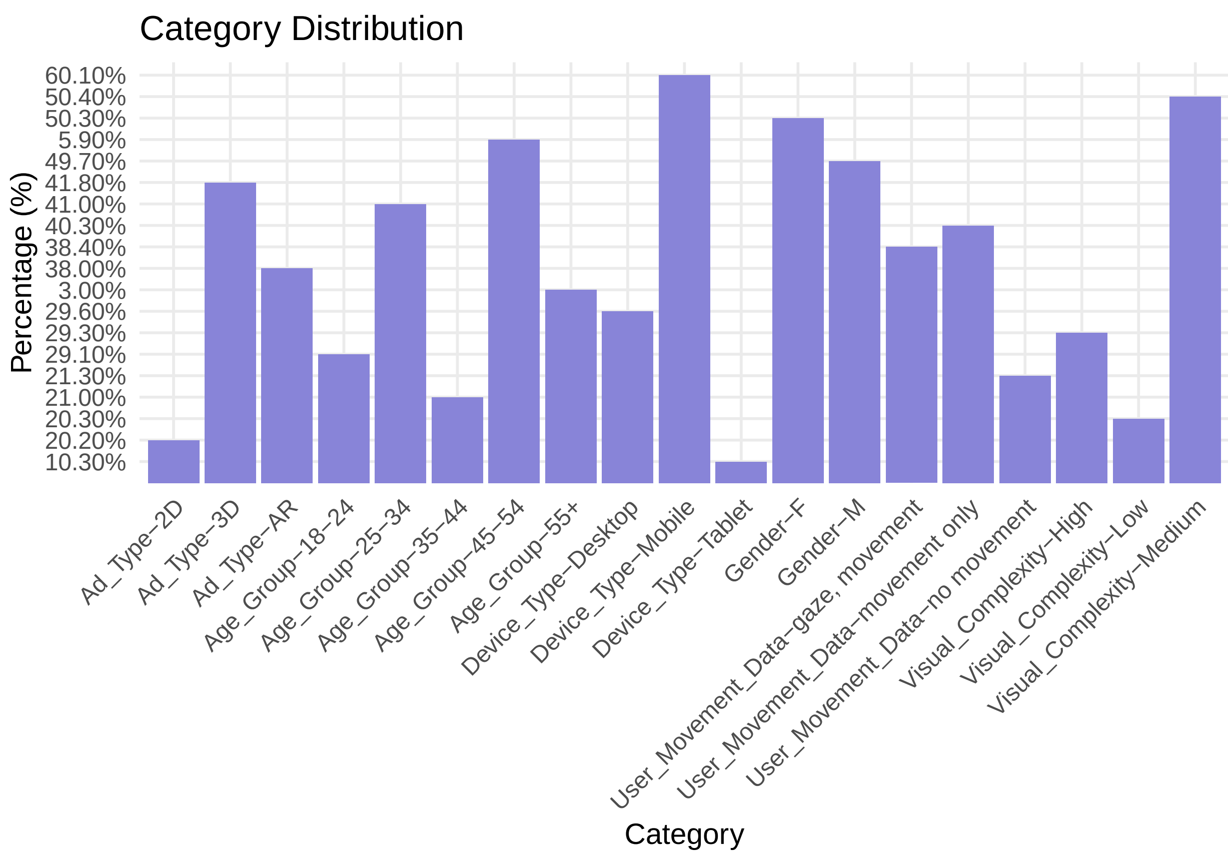
```
##              Category         Value Frequency Percentage
## 1             Ad_Type            3D       418     41.80%
## 2             Ad_Type            AR       380     38.00%
## 3             Ad_Type            2D       202     20.20%
## 4           Age_Group         25-34       410     41.00%
## 5           Age_Group         18-24       291     29.10%
## 6           Age_Group         35-44       210     21.00%
## 7           Age_Group         45-54        59      5.90%
## 8           Age_Group           55+        30      3.00%
```

```
## 9         Device_Type         Mobile       601      60.10%
## 10        Device_Type        Desktop       296      29.60%
## 11        Device_Type         Tablet       103      10.30%
## 12             Gender              F       503      50.30%
## 13             Gender              M       497      49.70%
## 14 User_Movement_Data  movement only       403      40.30%
## 15 User_Movement_Data gaze, movement       384      38.40%
## 16 User_Movement_Data    no movement       213      21.30%
## 17  Visual_Complexity         Medium       504      50.40%
## 18  Visual_Complexity           High       293      29.30%
## 19  Visual_Complexity            Low       203      20.30%
```

```r
#create copies for plotting
stats <- data_stats
freq <- data_freq

#metric distribution
metrics_long <- data_stats %>%
  dplyr::select(dplyr::all_of(c("Metric", "Q1", "Median", "Q3"))) %>%
  tidyr::pivot_longer(
    cols = dplyr::all_of(c("Q1", "Median", "Q3")),
    names_to = "Statistic",
    values_to = "Value"
  )

p1 <- ggplot(metrics_long, aes(x = Metric, y = Value, fill = Statistic)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Metric Distributions",
       x = "Metric",
       y = "Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_fill_manual(values = c("#82CA9D", "#8884D8", "#FFC658"))

#frequency distribution
p2 <- ggplot(freq, aes(x = paste(Category, Value, sep = "-"), y = Percentage)) +
  geom_bar(stat = "identity", fill = "#8884D8") +
  theme_minimal() +
  labs(title = "Category Distribution",
       x = "Category",
       y = "Percentage (%)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Print plots
print(p1)
```


Metric Distributions
Value
200
150
100
50
0
Bounce_Rate
Clicks
Conversion_Rate
CTR
Engagement_Score
Time_Spent
Metric
Statistic
Median
Q1
Q3

```r
print(p2)
```




```
#save
ggsave("visualizations\\metric_distributions.png", p1, width = 10, height = 6,)
ggsave("visualizations\\category_distribution.png", p2, width = 10, height = 6)
```

At a glance:

- Bounce Rate, Conversion Rate, CTR and Engagement Score all have fairly even distributions
- Clicks and Time Spent have a much more variance (higher Q1, Q3 values compared to median)
- We can use 'Q' values as a *'performance benchmark'* for future campaigns
  - If a campaign's metrics fall *below the Q1 value*, it is **underperforming**
  - If a campaign's metrics fall *above the Q3 value*, it is **successful**
  - anything in between means the ad is performing as expected

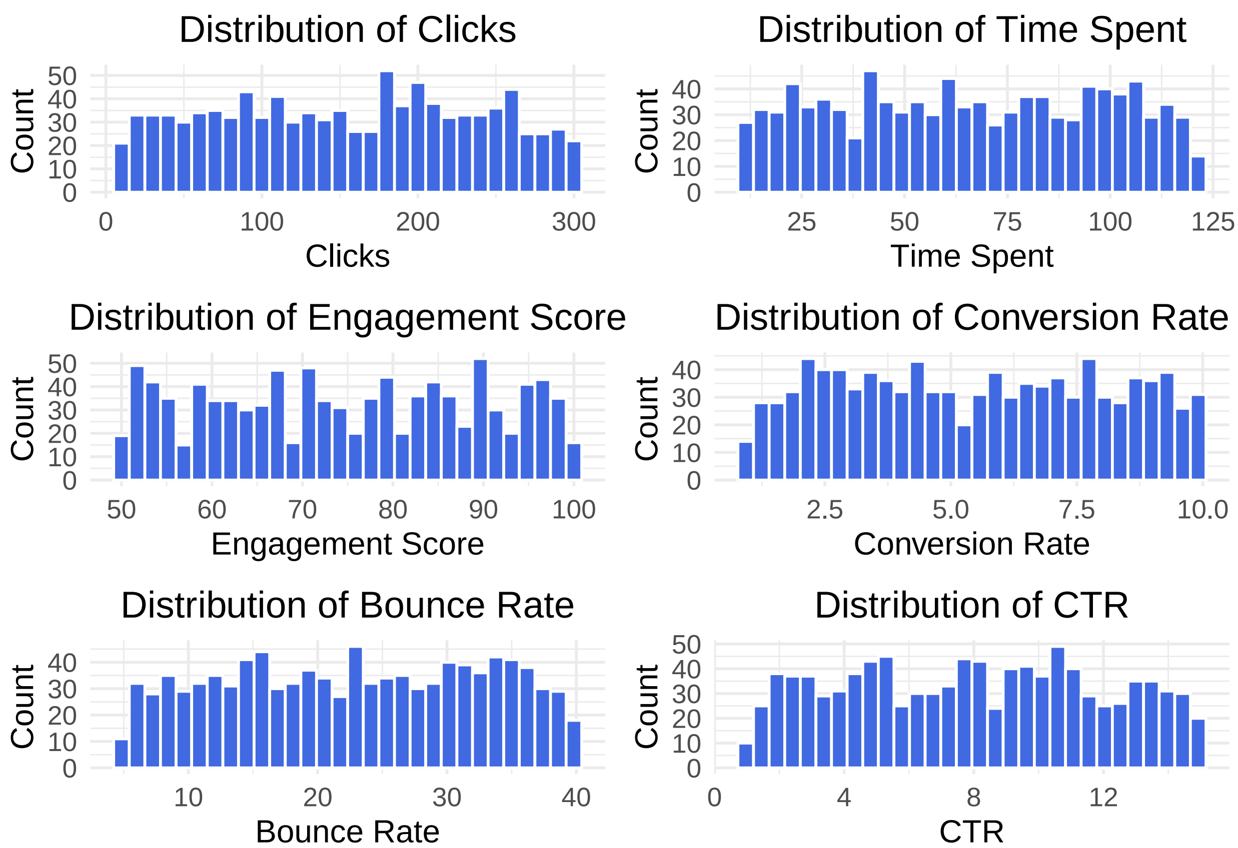
Areas to Investigate:

- **METRICS:** Clicks, Time Spent, Engagement Score, Conversion Rate, Bounce Rate, CTR
- 3D and AR ads are the most common Ad Types, but are they more successful than 2D?
- What impact does Visual Complexity have on our performance metrics?
- What is the relationship between Visual Complexity and User Movement, does this effect our metrics?

- What characteristics of an ad yields the highest metrics?
  - What characteristics effect Clicks, Time Spent Conversion Rate Bounce Rate and CTR the most?

Bar Plots

```r
#numerical data
# Create a barplot function
create_barplot <- function(data, column_name) {
  ggplot(data, aes(x = !!sym(column_name))) +
    geom_histogram(fill = "#4169E1", color = "white", bins = 30) +
    theme_minimal() +
    labs(
      title = paste("Distribution of", gsub("_", " ", column_name)),
      x = gsub("_", " ", column_name),
      y = "Count"
    ) +
    theme(
      plot.title = element_text(hjust = 0.5, size = 14),
      axis.title = element_text(size = 12),
      axis.text = element_text(size = 10)
    )
}

numerical_columns <- c(
  "Clicks",
  "Time_Spent",
  "Engagement_Score",
  "Conversion_Rate",
  "Bounce_Rate",
  "CTR"
)

plots <- list()
for (col in numerical_columns) {
  plots[[col]] <- create_barplot(data, col)
}

#plot
grid.arrange(
  plots[["Clicks"]],
  plots[["Time_Spent"]],
  plots[["Engagement_Score"]],
  plots[["Conversion_Rate"]],
  plots[["Bounce_Rate"]],
  plots[["CTR"]],
  ncol = 2
)
```




```
#save
ggsave("visualizations\\numerical_distributions.png", width = 10, height = 8)
```

```
# Function to calculate goodness-of-fit statistics
calculate_gof <- function(x, dist_name, params) {
  # Ensure x is positive for certain distributions
  x_adj <- if(min(x) <= 0) x - min(x) + 0.01 else x

  # Add small random noise to break ties
  x_jitter <- x + rnorm(length(x), 0, sd(x)/1000)
  x_adj_jitter <- x_adj + rnorm(length(x_adj), 0, sd(x_adj)/1000)

  # Kolmogorov-Smirnov test with jittered data
  if(dist_name == "normal") {
    ks_test <- suppressWarnings(ks.test(x_jitter, "pnorm", mean = params$mean, sd = params$sd))
  } else if(dist_name == "lognormal") {
    ks_test <- suppressWarnings(ks.test(x_adj_jitter, "plnorm", meanlog = params$meanlog, sdlog = param
  } else if(dist_name == "weibull") {
    ks_test <- suppressWarnings(ks.test(x_adj_jitter, "pweibull", shape = params$shape, scale = params$
  } else if(dist_name == "uniform") {
    ks_test <- suppressWarnings(ks.test(x_jitter, "punif", min = params$min, max = params$max))
  }

  # Return test statistics
  return(list(
    ks_stat = ks_test$statistic,
```

```
    ks_p = ks_test$p.value
  ))
}

# Enhanced distribution analysis function
analyze_distribution <- function(data, column_name) {
  # Get the data from the column
  x <- data[[column_name]]

  # Basic error checking
  if(is.null(x)) {
    stop(paste("Column", column_name, "not found in data"))
  }

  # Calculate basic statistics
  mu <- mean(x, na.rm = TRUE)
  sigma <- sd(x, na.rm = TRUE)

  # Create sequence for curves
  x_range <- seq(min(x, na.rm = TRUE), max(x, na.rm = TRUE), length.out = 200)

  # Fit distributions and calculate goodness-of-fit
  tryCatch({
    # Normal
    normal_params <- list(mean = mu, sd = sigma)
    normal_y <- dnorm(x_range, mean = mu, sd = sigma)
    normal_gof <- calculate_gof(x, "normal", normal_params)

    # Log-normal
    x_lognorm <- if(min(x) <= 0) x - min(x) + 0.01 else x
    lognorm_fit <- fitdistr(x_lognorm, "lognormal")
    lognorm_params <- list(
      meanlog = lognorm_fit$estimate[1],
      sdlog = lognorm_fit$estimate[2]
    )
    lognorm_y <- dlnorm(x_range, meanlog = lognorm_params$meanlog,
                        sdlog = lognorm_params$sdlog)
    lognorm_gof <- calculate_gof(x, "lognormal", lognorm_params)

    # Weibull
    weibull_fit <- fitdistr(x - min(x) + 0.01, "weibull")
    weibull_params <- list(
      shape = weibull_fit$estimate[1],
      scale = weibull_fit$estimate[2]
    )
    weibull_y <- dweibull(x_range - min(x_range) + 0.01,
                          shape = weibull_params$shape,
                          scale = weibull_params$scale)
    weibull_gof <- calculate_gof(x, "weibull", weibull_params)

    # Uniform
    uniform_params <- list(min = min(x), max = max(x))
    uniform_y <- dunif(x_range, min = min(x), max = max(x))
```

```
uniform_gof <- calculate_gof(x, "uniform", uniform_params)

# Create dataframe for all distributions
dist_df <- data.frame(
  x = rep(x_range, 4),
  y = c(normal_y, lognorm_y, weibull_y, uniform_y),
  Distribution = factor(rep(c("Normal", "Log-normal", "Weibull", "Uniform"),
                            each = length(x_range)))
)

# Find best fitting distribution
gof_stats <- data.frame(
  Distribution = c("Normal", "Log-normal", "Weibull", "Uniform"),
  KS_stat = c(normal_gof$ks_stat, lognorm_gof$ks_stat,
              weibull_gof$ks_stat, uniform_gof$ks_stat),
  P_value = c(normal_gof$ks_p, lognorm_gof$ks_p,
              weibull_gof$ks_p, uniform_gof$ks_p)
)
best_fit <- gof_stats[which.max(gof_stats$P_value), "Distribution"]

# Create goodness-of-fit results dataframe
gof_results <- data.frame(
  Distribution = c("Normal", "Log-normal", "Weibull", "Uniform"),
  KS_stat = c(normal_gof$ks_stat, lognorm_gof$ks_stat,
              weibull_gof$ks_stat, uniform_gof$ks_stat),
  P_value = c(normal_gof$ks_p, lognorm_gof$ks_p,
              weibull_gof$ks_p, uniform_gof$ks_p)
)

# Print results to console
cat("\nGoodness-of-fit Test Results for", column_name, "\n")
cat("========================================\n")
print(gof_results)
cat("\nBest fitting distribution:", best_fit, "\n")

# Create formatted text for plot annotation
gof_text <- paste(
  "Goodness-of-fit Statistics",
  sprintf("%-12s  D     p-value", "Distribution"),
  sprintf("%-12s %.3f  %.2e", "Normal", normal_gof$ks_stat, normal_gof$ks_p),
  sprintf("%-12s %.3f  %.2e", "Log-normal", lognorm_gof$ks_stat, lognorm_gof$ks_p),
  sprintf("%-12s %.3f  %.2e", "Weibull", weibull_gof$ks_stat, weibull_gof$ks_p),
  sprintf("%-12s %.3f  %.2e", "Uniform", uniform_gof$ks_stat, uniform_gof$ks_p),
  sprintf("\nBest fit: %s", best_fit),
  sep = "\n"
)

# Create the plot with error handling
p <- tryCatch({
  ggplot() +
  # Histogram with density
  geom_histogram(data = data.frame(x = x), aes(x = x, y = ..density..),
                 fill = "grey80", color = "white", alpha = 0.7,
```

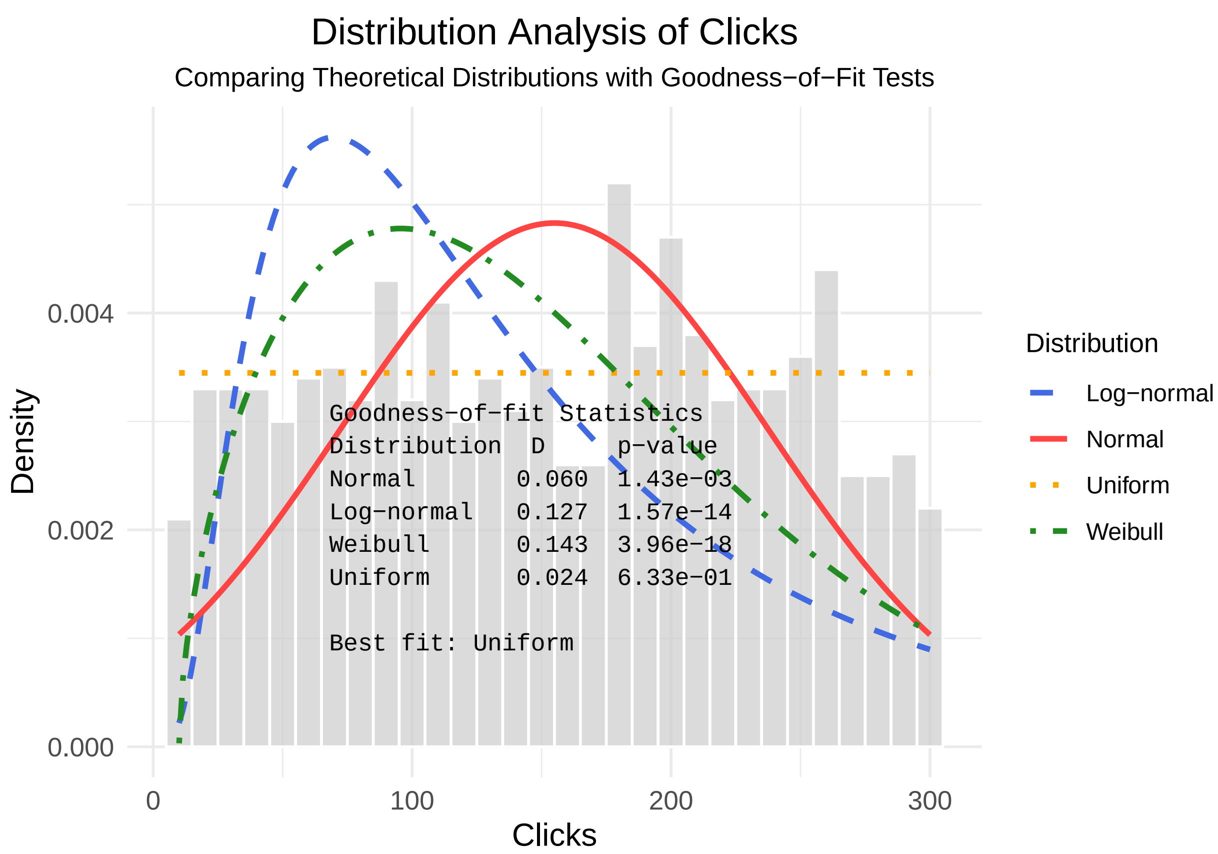
```r
                    bins = 30) +
    # Add theoretical distributions
    geom_line(data = dist_df,
              aes(x = x, y = y, color = Distribution, linetype = Distribution),
              linewidth = 1) +
    # Customize colors and linetypes
    scale_color_manual(values = c("Normal" = "#FF4444",
                                  "Log-normal" = "#4169E1",
                                  "Weibull" = "#228B22",
                                  "Uniform" = "#FFA500")) +
    scale_linetype_manual(values = c("Normal" = "solid",
                                     "Log-normal" = "dashed",
                                     "Weibull" = "dotdash",
                                     "Uniform" = "dotted")) +
    # Add statistics box
    # Add statistics in a text box using annotate
    annotate("text",
             x = min(x) + (max(x) - min(x)) * 0.2,  # Position at 20% of x range
             y = max(density(x)$y) * 0.8,           # Position at 80% of max height
             label = gof_text,
             hjust = 0,
             vjust = 1,
             size = 3,
             family = "mono",
             color = "black",
             box.color = "black",
             box.padding = unit(0.5, "lines"),
             box.margin = unit(0.5, "lines")) +
    # Theme and labels
    theme_minimal() +
    labs(
      title = paste("Distribution Analysis of", gsub("_", " ", column_name)),
      subtitle = "Comparing Theoretical Distributions with Goodness-of-Fit Tests",
      x = gsub("_", " ", column_name),
      y = "Density"
    ) +
    theme(
      plot.title = element_text(hjust = 0.5, size = 14),
      plot.subtitle = element_text(hjust = 0.5, size = 10),
      axis.title = element_text(size = 12),
      axis.text = element_text(size = 10),
      legend.position = "right",
      legend.title = element_text(size = 10),
      legend.text = element_text(size = 9)
    )

  }, error = function(e) {
    message("Error in plot creation: ", e$message)
    return(NULL)
  })

  if (!is.null(p)) {
    return(p)
```

```r
    } else {
      # If plot creation failed, create a simple error plot
      return(ggplot() +
             annotate("text", x = 0.5, y = 0.5,
                      label = "Error creating distribution plot",
                      size = 5) +
             theme_void() +
             xlim(0, 1) + ylim(0, 1))
    }

  }, error = function(e) {
    message("Error in distribution fitting: ", e$message)
    return(NULL)
  })
}



# Function to plot one column
plot_one <- function(data, column_name) {
  p <- analyze_distribution(data, column_name)
  if (!is.null(p)) {
    print(p)
    ggsave(
      filename = paste0("visualizations\\distribution_analysis_", tolower(column_name), ".png"),
      plot = p,
      width = 12,
      height = 7,
      dpi = 300
    )
  }
}
```

```r
plot_one(data, "Clicks")
```

```
##
## Goodness-of-fit Test Results for Clicks
## ==========================================
##   Distribution    KS_stat      P_value
## 1       Normal 0.06017627 1.431230e-03
## 2   Log-normal 0.12743358 1.569417e-14
## 3      Weibull 0.14276566 3.957503e-18
## 4      Uniform 0.02360576 6.330958e-01
##
## Best fitting distribution: Uniform

## Warning in annotate("text", x = min(x) + (max(x) - min(x)) * 0.2, y =
## max(density(x)$y) * : Ignoring unknown parameters: `box.colour`, `box.padding`,
## and `box.margin`

## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
```

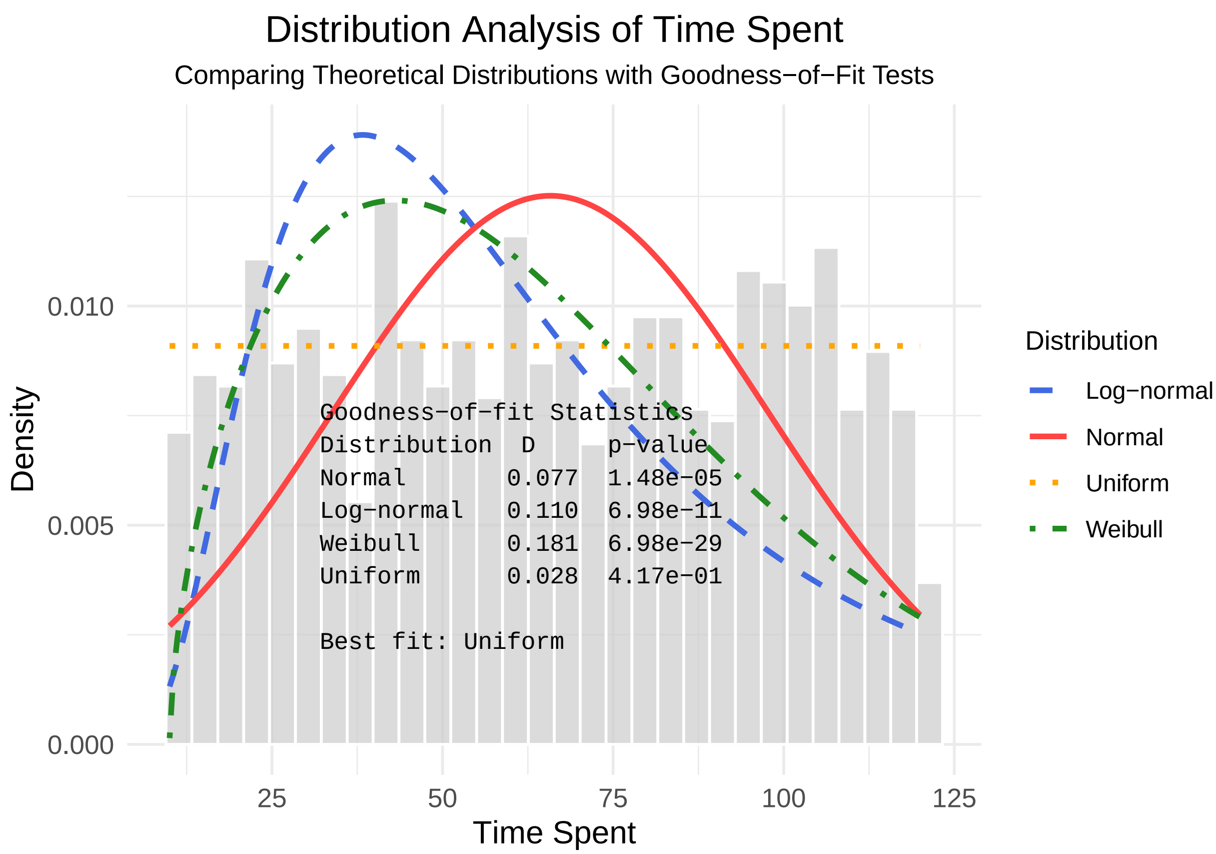
```
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```r
plot_one(data, "Time_Spent")
```

```
##
## Goodness-of-fit Test Results for Time_Spent
## ==========================================
##   Distribution    KS_stat      P_value
## 1       Normal 0.07685794 1.479623e-05
## 2   Log-normal 0.10972501 6.975675e-11
## 3      Weibull 0.18100459 6.978216e-29
## 4      Uniform 0.02791024 4.172009e-01
##
## Best fitting distribution: Uniform
```

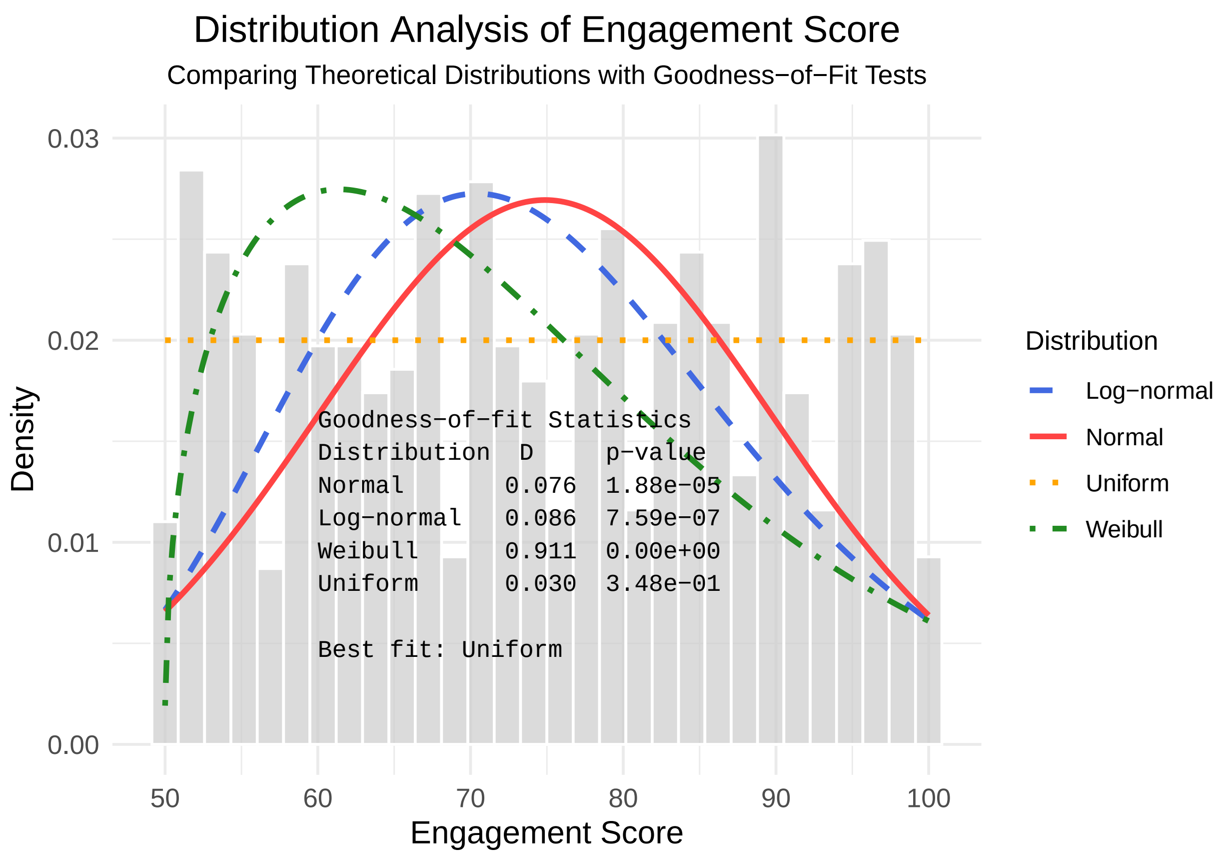
```
## Warning in annotate("text", x = min(x) + (max(x) - min(x)) * 0.2, y =
## max(density(x)$y) * : Ignoring unknown parameters: `box.colour`, `box.padding`,
## and `box.margin`
```




```r
plot_one(data, "Engagement_Score")
```

```
##
## Goodness-of-fit Test Results for Engagement_Score
## ==========================================
##   Distribution    KS_stat      P_value
## 1       Normal 0.07607241 1.881464e-05
## 2   Log-normal 0.08597759 7.590914e-07
## 3      Weibull 0.91107447 0.000000e+00
## 4      Uniform 0.02951776 3.482603e-01
##
## Best fitting distribution: Uniform
```

```
## Warning in annotate("text", x = min(x) + (max(x) - min(x)) * 0.2, y =
## max(density(x)$y) * : Ignoring unknown parameters: `box.colour`, `box.padding`,
## and `box.margin`
```

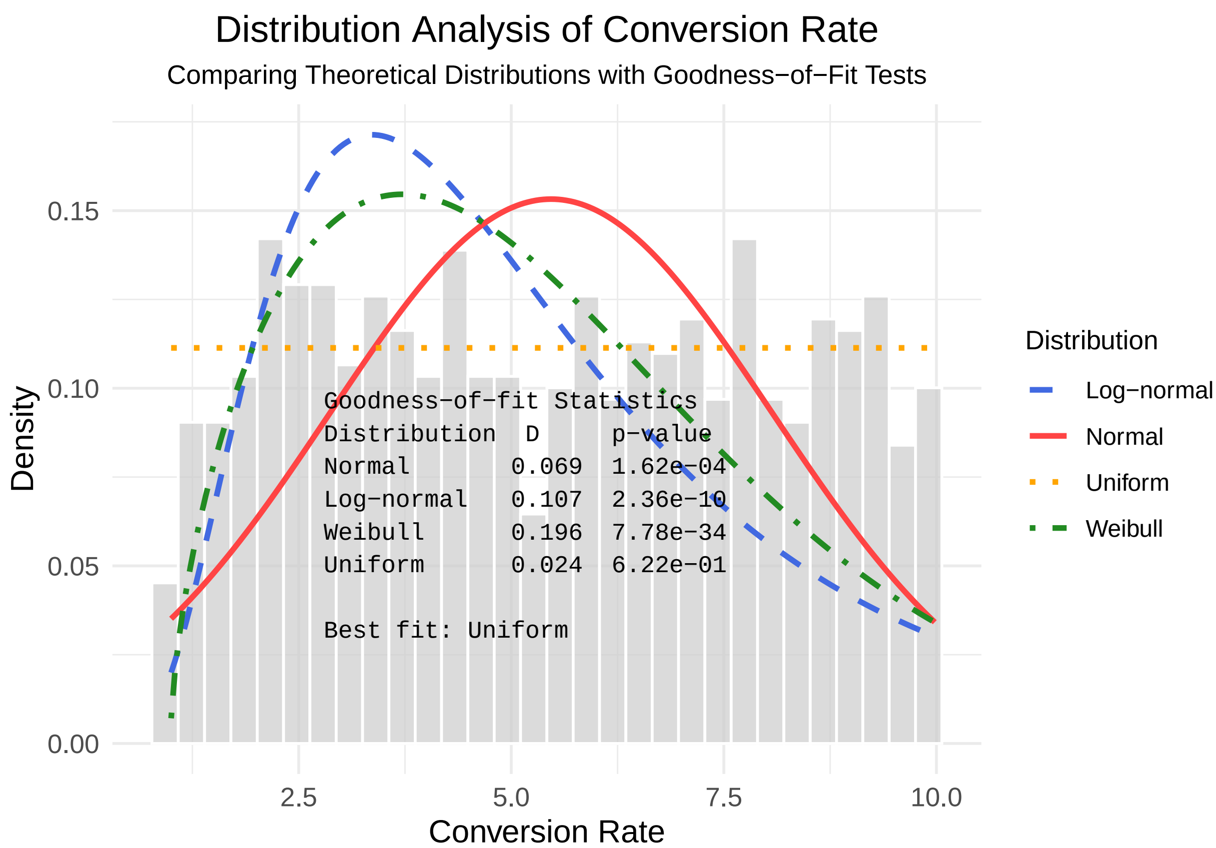



```r
plot_one(data, "Conversion_Rate")
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
##
## Goodness-of-fit Test Results for Conversion_Rate
## ==========================================
##   Distribution    KS_stat      P_value
## 1       Normal 0.06862982 1.621552e-04
## 2   Log-normal 0.10691421 2.357737e-10
## 3      Weibull 0.19612376 7.783288e-34
## 4      Uniform 0.02381761 6.218180e-01
##
## Best fitting distribution: Uniform
```

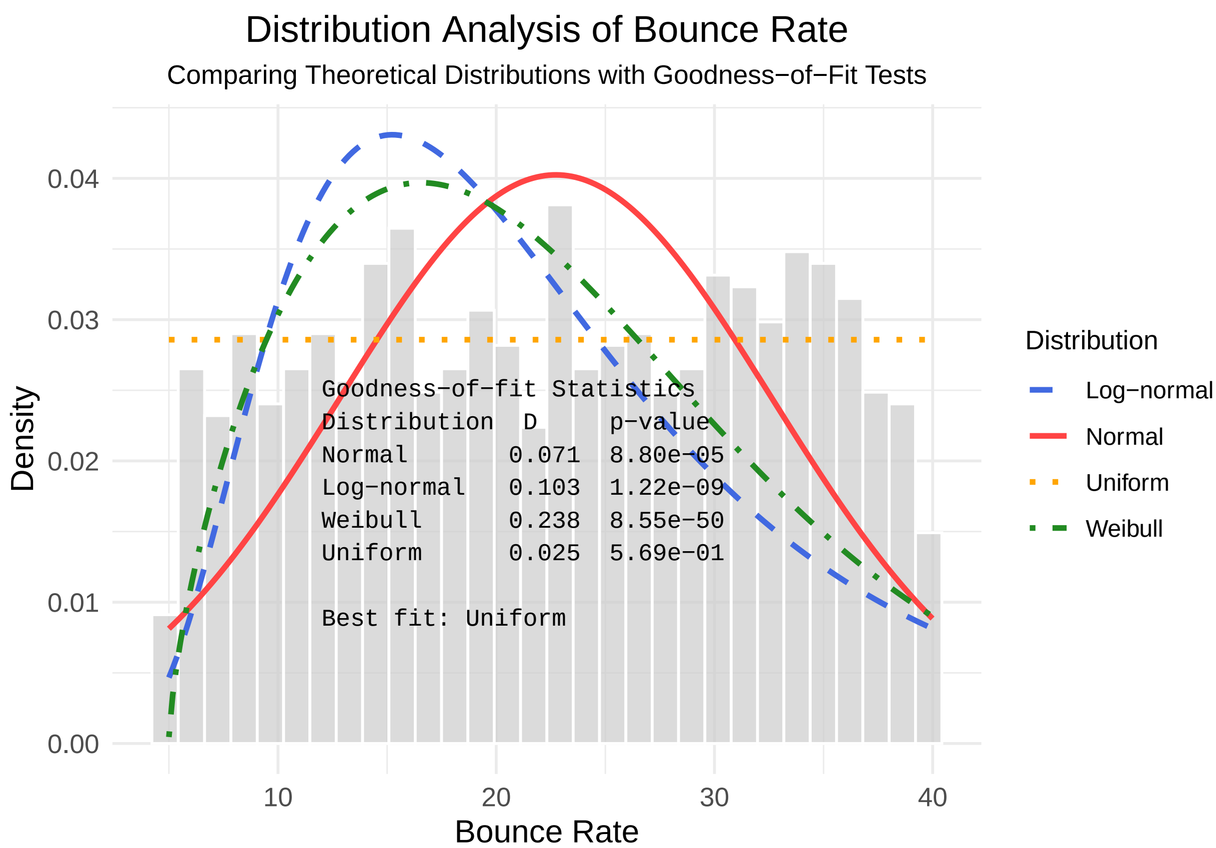
```
## Warning in annotate("text", x = min(x) + (max(x) - min(x)) * 0.2, y =
## max(density(x)$y) * : Ignoring unknown parameters: `box.colour`, `box.padding`,
## and `box.margin`
```




```r
plot_one(data, "Bounce_Rate")
```

```
##
## Goodness-of-fit Test Results for Bounce_Rate
## ==========================================
##   Distribution    KS_stat      P_value
## 1       Normal 0.07082074 8.801474e-05
## 2   Log-normal 0.10300658 1.216166e-09
## 3      Weibull 0.23840827 8.545416e-50
## 4      Uniform 0.02482560 5.686393e-01
##
## Best fitting distribution: Uniform
```

```
## Warning in annotate("text", x = min(x) + (max(x) - min(x)) * 0.2, y =
## max(density(x)$y) * : Ignoring unknown parameters: `box.colour`, `box.padding`,
## and `box.margin`
```

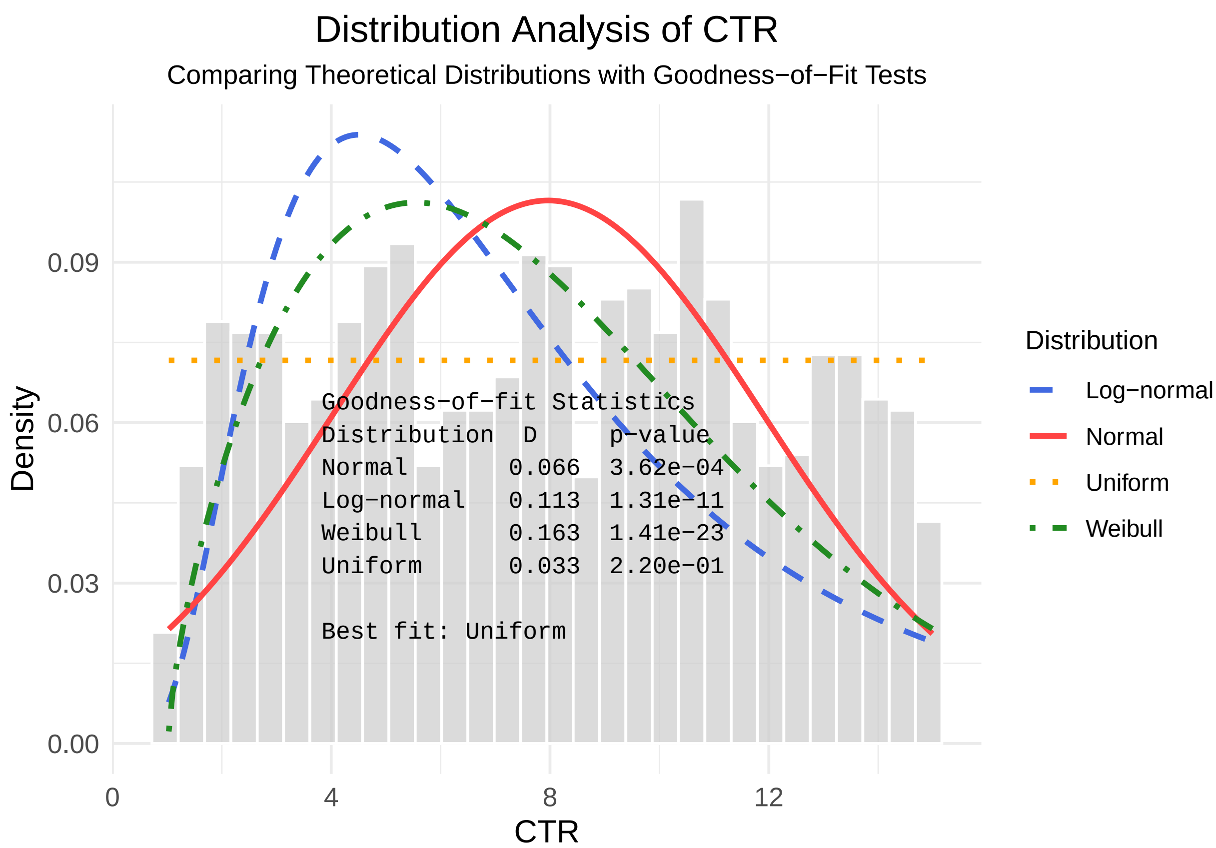



```r
plot_one(data, "CTR")
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
##
## Goodness-of-fit Test Results for CTR
## ==========================================
##   Distribution    KS_stat      P_value
## 1       Normal 0.06563953 3.619707e-04
## 2   Log-normal 0.11346766 1.312278e-11
## 3      Weibull 0.16326103 1.410967e-23
## 4      Uniform 0.03322416 2.196164e-01
##
## Best fitting distribution: Uniform
```

```
## Warning in annotate("text", x = min(x) + (max(x) - min(x)) * 0.2, y =
## max(density(x)$y) * : Ignoring unknown parameters: `box.colour`, `box.padding`,
## and `box.margin`
```



As we see from our Distribution Analysis, our data is Uniformly Distributed

Correlation Plot

```r
numerical_columns <- c("Clicks", "Time_Spent", "Engagement_Score",
                       "Conversion_Rate", "Bounce_Rate", "CTR")

numerical_data <- data %>%
  dplyr::select(dplyr::all_of(numerical_columns))

# Calculate correlation matrix
cor_matrix <- cor(numerical_data)

# Print matrix for inspection
print("Correlation Matrix:")
```

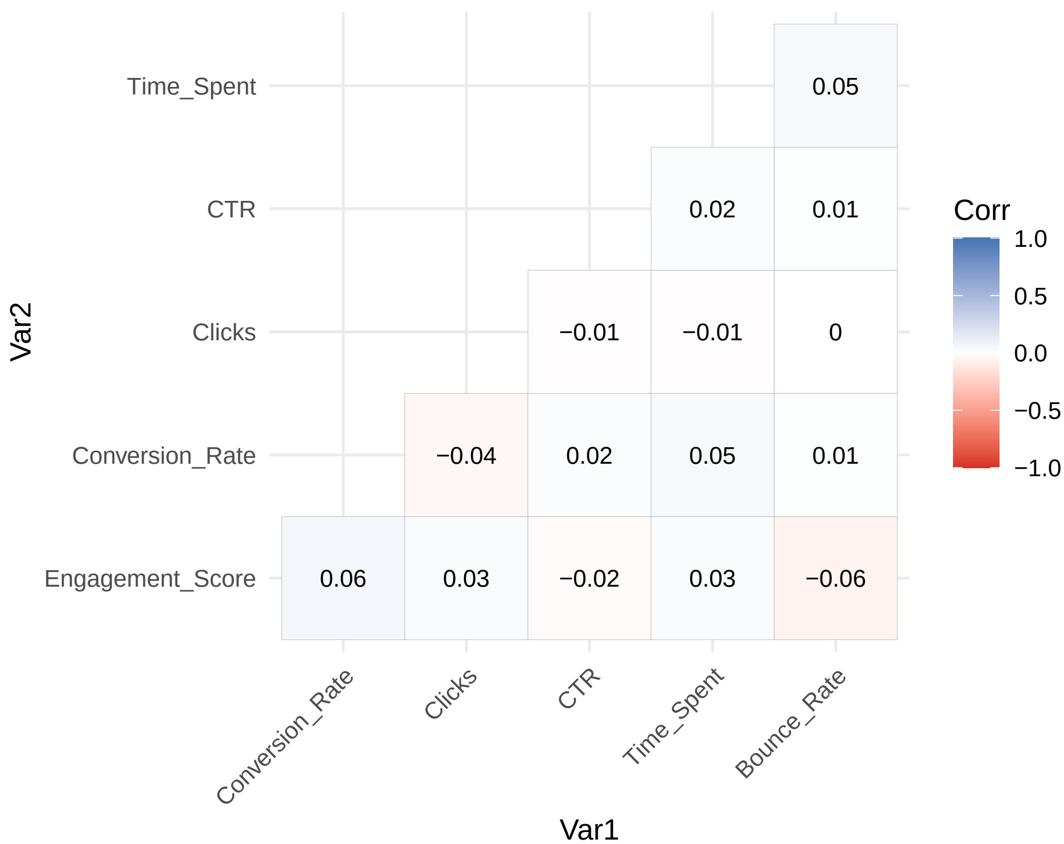
```
## [1] "Correlation Matrix:"
```

```r
print(cor_matrix)
```

```
##                      Clicks   Time_Spent Engagement_Score Conversion_Rate
## Clicks           1.000000000 -0.006126274       0.03056959    -0.044432411
## Time_Spent      -0.006126274  1.000000000       0.02564127     0.052200269
```

```
## Engagement_Score  0.030569585  0.025641273       1.00000000      0.055550416
## Conversion_Rate  -0.044432411  0.052200269       0.05555042      1.000000000
## Bounce_Rate       0.004451510  0.045156588      -0.06488862      0.007921938
## CTR              -0.006370837  0.015421482      -0.02109692      0.019890959
##                   Bounce_Rate          CTR
## Clicks            0.004451510 -0.006370837
## Time_Spent        0.045156588  0.015421482
## Engagement_Score -0.064888619 -0.021096916
## Conversion_Rate   0.007921938  0.019890959
## Bounce_Rate       1.000000000  0.012645491
## CTR               0.012645491  1.000000000
```

```r
# Create correlation plot
cor_plot <- ggcorrplot(cor_matrix,
                       hc.order = TRUE,
                       type = "lower",
                       lab = TRUE,
                       lab_size = 3,
                       colors = c("#D73027", "white", "#4575B4")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Display plot
print(cor_plot)
```



```r
# Save plot
ggsave("visualizations/correlation_matrix.png",
       plot = cor_plot,
       width = 10,
       height = 8)

# Optional: Return correlation statistics
correlation_summary <- data.frame(
  Variable1 = character(),
  Variable2 = character(),
  Correlation = numeric(),
  stringsAsFactors = FALSE
)
```

Looking at the Barplots and Correlation matrix:

- Our data is not normally distributed, so linear modeling/analysis will not be that useful
- There are no strong correlations between any of the numerical columns

    - We do not need to investigate the interactions between these columns further

Non-Linear Relationships

Ad Type vs. Engagement Score

Kruskall Test

- Null Hypothesis: 3D and AR ads are more effective than 2D ads
- Alternative Hypothesis: 3D and AR ads are not more successful than 2D ads

```r
# 2. Kruskal-Wallis Test with detailed interpretation
e_score <- kruskal.test(Engagement_Score ~ Ad_Type, data = data)
c_score <- kruskal.test(Clicks ~ Ad_Type, data = data)
t_score <- kruskal.test(Time_Spent ~ Ad_Type, data = data)
ctr_score <- kruskal.test(CTR ~ Ad_Type, data = data)

# Create results dataframe with more detailed information
results <- data.frame(
  Metric = c("Engagement Score", "Clicks", "Time Spent", "CTR"),
  p_value = c(e_score$p.value, c_score$p.value, t_score$p.value, ctr_score$p.value),
  statistic = c(e_score$statistic, c_score$statistic, t_score$statistic, ctr_score$statistic)
)

# Add interpretation columns
results <- results %>%
  mutate(
    Significance = case_when(
      p_value < 0.01 ~ "Highly Significant",
      p_value < 0.05 ~ "Significant",
      TRUE ~ "Not Significant"
```

```r
    ),
    Interpretation = case_when(
      p_value < 0.05 ~ "There are significant differences between ad types",
      TRUE ~ "No significant differences between ad types"
    )
  ) %>%
  mutate(
    p_value = round(p_value, 4),
    statistic = round(statistic, 2)
  )

# Print formatted results
print("Analysis of Ad Type Effects on Performance Metrics")
```

```
## [1] "Analysis of Ad Type Effects on Performance Metrics"
```

```r
print("------------------------------------------------")
```

```
## [1] "------------------------------------------------"
```

```r
for(i in 1:nrow(results)) {
  cat(sprintf("\nMetric: %s", results$Metric[i]))
  cat(sprintf("\n- P-value: %f", results$p_value[i]))
  cat(sprintf("\n- Chi-squared statistic: %f", results$statistic[i]))
  cat(sprintf("\n- Result: %s", results$Significance[i]))
  cat(sprintf("\n- Interpretation: %s\n", results$Interpretation[i]))
}
```

```
##
## Metric: Engagement Score
## - P-value: 0.213800
## - Chi-squared statistic: 3.090000
## - Result: Not Significant
## - Interpretation: No significant differences between ad types
##
## Metric: Clicks
## - P-value: 0.574300
## - Chi-squared statistic: 1.110000
## - Result: Not Significant
## - Interpretation: No significant differences between ad types
##
## Metric: Time Spent
## - P-value: 0.050900
## - Chi-squared statistic: 5.960000
## - Result: Not Significant
## - Interpretation: No significant differences between ad types
##
## Metric: CTR
## - P-value: 0.433200
## - Chi-squared statistic: 1.670000
## - Result: Not Significant
## - Interpretation: No significant differences between ad types
```

```r
# Create summary statement
significant_metrics <- results$Metric[results$p_value < 0.05]
cat("\nSummary:\n")
```

```
##
## Summary:
```

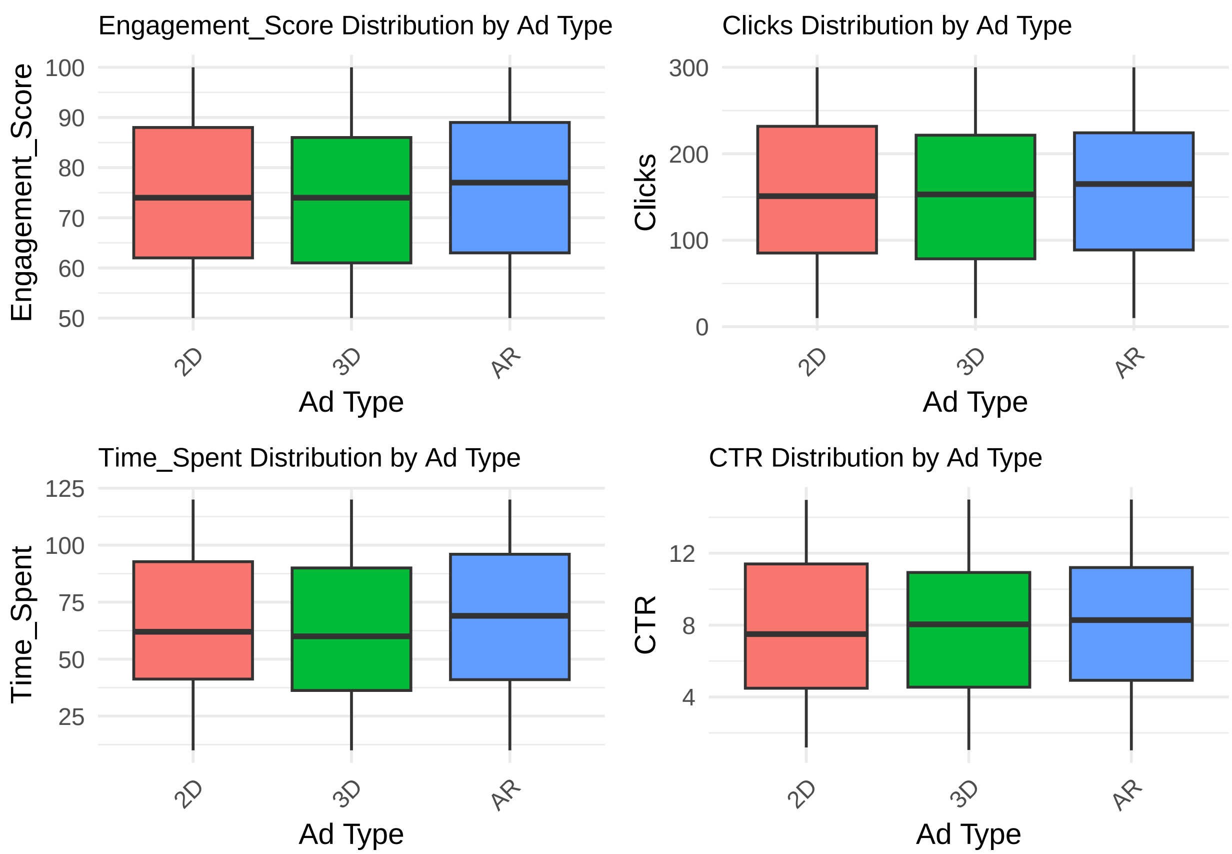
```r
if(length(significant_metrics) > 0) {
  cat("The following metrics show significant differences between ad types:\n")
  cat(paste("-", significant_metrics, collapse = "\n"))
  cat("\n\nThis suggests that ad type does influence these performance metrics.")
} else {
  cat("None of the metrics showed significant differences between ad types.\n")
  cat("This suggests that ad type may not be a determining factor in ad performance.")
}
```

```
## None of the metrics showed significant differences between ad types.
## This suggests that ad type may not be a determining factor in ad performance.
```

Boxplots

- Visualize our findings

```r
#function to create a boxplot for a given metric
create_boxplot <- function(data, metric_name) {
  ggplot(data, aes(x = Ad_Type, y = .data[[metric_name]])) +
    geom_boxplot(aes(fill = Ad_Type)) +
    theme_minimal() +
    labs(title = paste(metric_name, "Distribution by Ad Type"),
         x = "Ad Type",
         y = metric_name) +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1),
      plot.title = element_text(size = 10),
      legend.position = "none"
    )
}

#compare across all metrics
metrics <- c("Engagement_Score", "Clicks", "Time_Spent", "CTR")
plot_list <- lapply(metrics, function(metric) create_boxplot(data, metric))

#2x2 grid for plots
grid.arrange(
  plot_list[[1]], plot_list[[2]],
  plot_list[[3]], plot_list[[4]],
  ncol = 2
)
```



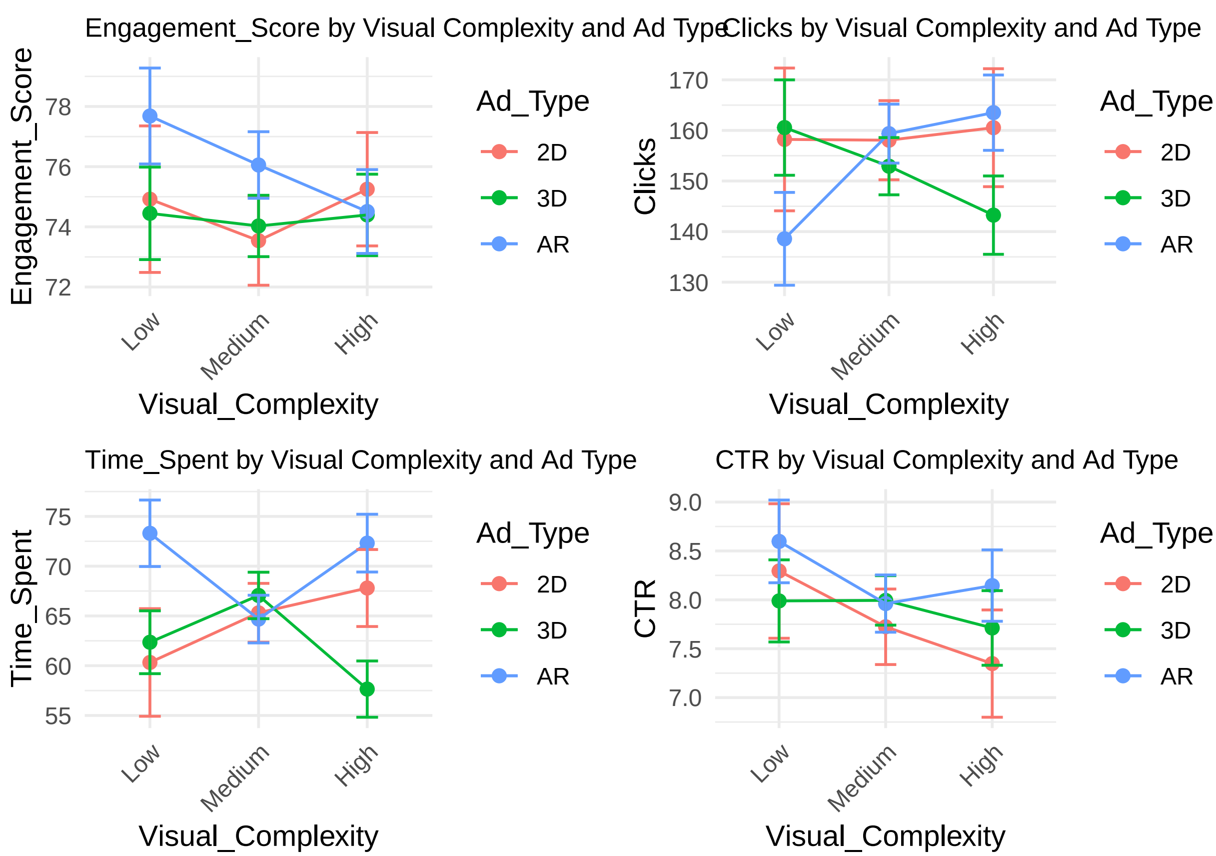

```r
#save
ggsave("visualizations\\ad_type_metrics_comparison.png",
       arrangeGrob(
         plot_list[[1]], plot_list[[2]],
         plot_list[[3]], plot_list[[4]],
         ncol = 2
       ),
       width = 12, height = 10)
```

- Based on our results, we see that while Ad Type does not have any significant effect on our target metrics, we still do know that 3D ads and AR ads are more popular based on the barplot from earlier.

Interactions between Ad Type and Visual Complexity

```r
# First, convert Visual_Complexity to factor with specified order
data$Visual_Complexity <- factor(data$Visual_Complexity,
                                 levels = c("Low", "Medium", "High"))

# Function to create interaction plot for a given metric
create_interaction_plot <- function(data, metric_name) {
  ggplot(data, aes(x = Visual_Complexity, y = .data[[metric_name]],
                   color = Ad_Type, group = Ad_Type)) +
```

```
    stat_summary(fun = mean, geom = "point", size = 2) +
    stat_summary(fun = mean, geom = "line") +
    stat_summary(fun.data = mean_se, geom = "errorbar", width = 0.2) +
    theme_minimal() +
    labs(title = paste(metric_name, "by Visual Complexity and Ad Type"),
         y = metric_name) +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1),
      plot.title = element_text(size = 10)
    )
}


# Create plots for all metrics
metrics <- c("Engagement_Score", "Clicks", "Time_Spent", "CTR")
interaction_plots <- lapply(metrics, function(metric) create_interaction_plot(data, metric))

# Arrange plots in a 2x2 grid
grid.arrange(
  interaction_plots[[1]], interaction_plots[[2]],
  interaction_plots[[3]], interaction_plots[[4]],
  ncol = 2
)
```



```r
#save
ggsave("visualizations\\interaction_plots.png",
       arrangeGrob(
         interaction_plots[[1]], interaction_plots[[2]],
         interaction_plots[[3]], interaction_plots[[4]],
         ncol = 2
       ),
       width = 12, height = 10)

# Create empty list to store results
results <- list()

# Run ANOVA for each metric and store results
for (metric in metrics) {
  # Create formula and run ANOVA
  formula <- as.formula(paste(metric, "~ Ad_Type * Visual_Complexity"))
  model <- aov(formula, data = data)

  # Store results
  results[[metric]] <- summary(model)[[1]]
}

# Print results in a clear format
cat("Interaction Analysis Results:\n")
```

```
## Interaction Analysis Results:
```

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

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

```r
for (metric in metrics) {
  cat(sprintf("Metric: %s\n", metric))
  cat("------------------\n")

  # Extract interaction p-value
  p_val <- results[[metric]]["Ad_Type:Visual_Complexity", "Pr(>F)"]
  f_val <- results[[metric]]["Ad_Type:Visual_Complexity", "F value"]

  cat(sprintf("F-value: %.3f\n", f_val))
  cat(sprintf("p-value: %.4f\n", p_val))
  cat(sprintf("Significant: %s\n\n", ifelse(p_val < 0.05, "Yes", "No")))
}
```

```
## Metric: Engagement_Score
## ------------------
## F-value: 0.473
## p-value: 0.7558
## Significant: No
##
## Metric: Clicks
## ------------------
```

```
## F-value: 1.663
## p-value: 0.1563
## Significant: No
##
## Metric: Time_Spent
## ------------------
## F-value: 3.536
## p-value: 0.0071
## Significant: Yes
##
## Metric: CTR
## ------------------
## F-value: 0.351
## p-value: 0.8434
## Significant: No
```

- This test tells us that how long users spend with an ad (Time Spent) depends on both the Ad Type AND Visual Complexity working together, the other metrics are not affected by Visual Complexity

Exploring the effects of Ad Type and Visual Complexity on Time Spent

```r
# 1. Focused Time Spent Interaction Plot with Enhanced Details
time_spent_plot <- ggplot(data, aes(x = Visual_Complexity, y = Time_Spent,
                                     color = Ad_Type, group = Ad_Type)) +
    stat_summary(fun = mean, geom = "point") +
    stat_summary(fun = mean, geom = "line") +
    stat_summary(fun.data = mean_se, geom = "errorbar", width = 0.2) +
    theme_minimal() +
    labs(title = "Interaction Effect: Ad Type and Visual Complexity on Time Spent",
         subtitle = "Error bars represent standard error of the mean",
         y = "Time Spent (seconds)") +
    scale_color_brewer(palette = "Set2") +
    theme(
      legend.position = "right",
      plot.title = element_text(size = 12, face = "bold"),
      axis.title = element_text(size = 10),
      legend.title = element_text(size = 10)
    )

# 2. Post-hoc analysis
# Perform Tukey's HSD test
tukey_model <- aov(Time_Spent ~ Ad_Type * Visual_Complexity, data = data)
tukey_results <- TukeyHSD(tukey_model, which = "Ad_Type:Visual_Complexity")

# Filter significant comparisons (p < 0.05)
sig_comparisons <- as.data.frame(tukey_results$`Ad_Type:Visual_Complexity`)
sig_comparisons$comparison <- rownames(sig_comparisons)
sig_comparisons <- sig_comparisons[sig_comparisons$`p adj` < 0.05, ]

# Print significant differences
cat("\nSignificant differences in Time Spent:\n")
```

```
##
## Significant differences in Time Spent:
```

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

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

```r
if(nrow(sig_comparisons) > 0) {
  for(i in 1:nrow(sig_comparisons)) {
    cat(sprintf("\n%s:\n", sig_comparisons$comparison[i]))
    cat(sprintf("Difference: %.2f seconds\n", sig_comparisons$diff[i]))
    cat(sprintf("Adjusted p-value: %.4f\n", sig_comparisons$`p adj`[i]))
  }
} else {
  cat("No pairwise comparisons were significant at p < 0.05\n")
}
```

```
##
## 3D:High-AR:Low:
## Difference: -15.66 seconds
## Adjusted p-value: 0.0197
##
## AR:High-3D:High:
## Difference: 14.67 seconds
## Adjusted p-value: 0.0115
```

```r
# Additional insight: Check effect sizes within each Visual Complexity level
cat("\nEffect Sizes within Visual Complexity Levels:\n")
```

```
##
## Effect Sizes within Visual Complexity Levels:
```

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

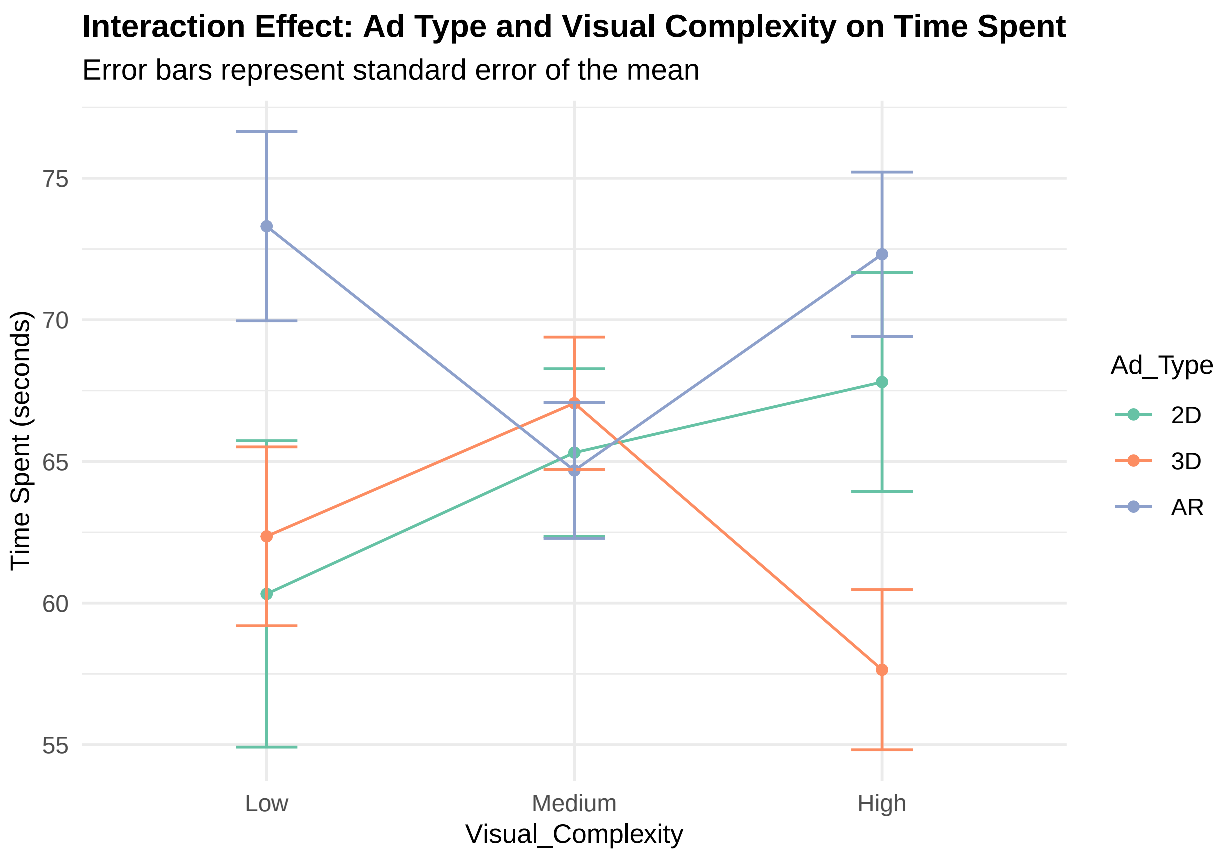
```
## ==========================================
```

```r
for(complexity in levels(data$Visual_Complexity)) {
  subset_data <- data[data$Visual_Complexity == complexity, ]
  effect_size <- summary(aov(Time_Spent ~ Ad_Type, data = subset_data))[[1]]
  cat(sprintf("\nVisual Complexity: %s\n", complexity))
  cat(sprintf("F-value: %.3f\n", effect_size$`F value`[1]))
  cat(sprintf("p-value: %.4f\n", effect_size$`Pr(>F)`[1]))
}
```

```
##
## Visual Complexity: Low
## F-value: 3.593
## p-value: 0.0293
##
## Visual Complexity: Medium
## F-value: 0.274
```

```
## p-value: 0.7603
##
## Visual Complexity: High
## F-value: 6.928
## p-value: 0.0012
```

```
# Display the focused plot
print(time_spent_plot)
```



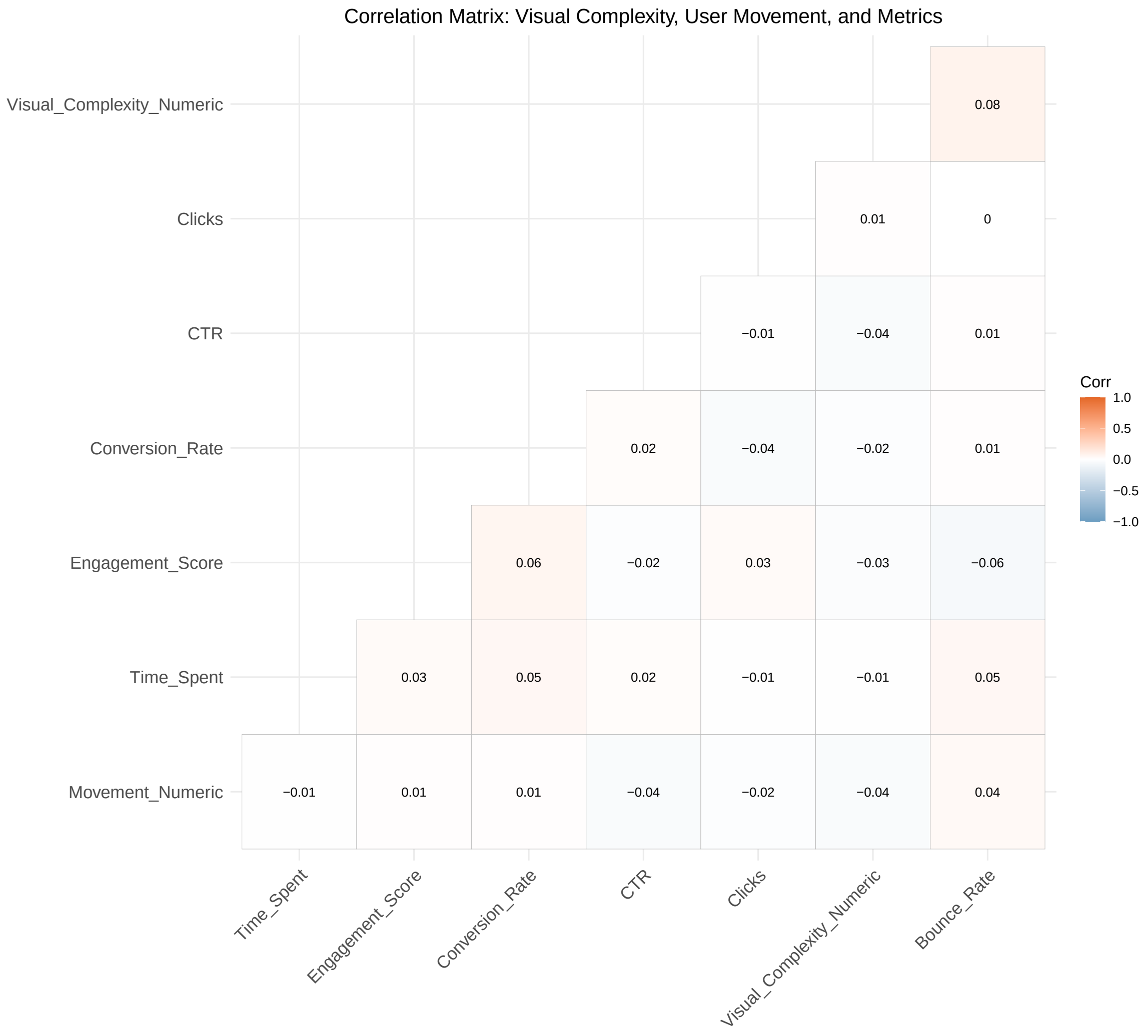

```
#save
ggsave("visualizations\\time_spent_interaction_plot.png", time_spent_plot, width = 10, height = 6)
```

- Looking further we see that AR Ads are most effective at Low and High Complexities
- At Medium Complexity, each Ad Type performs similarly

Verifying from our Tukeys HSD test: > * Low Complexity: Moderate effect (F = 3.593, p = 0.0293) - Ad types do differ significantly > * Medium Complexity: No significant effect (F = 0.274, p = 0.7603) - Ad types perform similarly > * High Complexity: Strongest effect (F = 6.928, p = 0.0012) - Ad types show very significant differences

Relationship between Visual Complexity and User Movement

```r
# Select relevant columns for correlation
correlation_cols <- c("Visual_Complexity_Numeric", "Movement_Numeric",
                      "Clicks", "Time_Spent", "Engagement_Score",
                      "Conversion_Rate", "Bounce_Rate", "CTR")

correlation_data <- data %>%
  dplyr::select(dplyr::all_of(correlation_cols))


# Calculate correlation matrix
cor_matrix <- cor(correlation_data)

# Create correlation plot
ggcorrplot(cor_matrix,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           colors = c("#6D9EC1", "white", "#E46726"),
           title = "Correlation Matrix: Visual Complexity, User Movement, and Metrics",
           ggtheme = theme_minimal()) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14),
    axis.text.x = element_text(angle = 45, hjust = 1),
    axis.text.y = element_text(hjust = 1)
  )
```




```r
# Add statistical significance tests
correlation_tests <- data.frame(
  Variable1 = character(),
  Variable2 = character(),
  Correlation = numeric(),
  P_Value = numeric()
)

variables <- colnames(correlation_data)
for(i in 1:(length(variables)-1)) {
  for(j in (i+1):length(variables)) {
    test <- cor.test(correlation_data[[variables[i]]],
                    correlation_data[[variables[j]]])
    correlation_tests <- rbind(correlation_tests,
                              data.frame(Variable1 = variables[i],
                                        Variable2 = variables[j],
                                        Correlation = test$estimate,
                                        P_Value = test$p.value))
  }
}
```

```r
# Display significant correlations (p < 0.05)
cat("\nStatistically Significant Correlations (p < 0.05):\n\n")
```

```
##
## Statistically Significant Correlations (p < 0.05):
```

```r
significant_cors <- correlation_tests %>%
  filter(P_Value < 0.05) %>%
  arrange(P_Value) %>%
  mutate(
    Correlation = round(Correlation, 3),
    P_Value = round(P_Value, 4)
  )

# Print each correlation clearly
for(i in 1:nrow(significant_cors)) {
  cat(sprintf("%s and %s:\n",
              significant_cors$Variable1[i],
              significant_cors$Variable2[i]))
  cat(sprintf("  Correlation: %.3f\n", significant_cors$Correlation[i]))
  cat(sprintf("  P-value: %.4f\n\n", significant_cors$P_Value[i]))
}
```

```
## Visual_Complexity_Numeric and Bounce_Rate:
##   Correlation: 0.082
##   P-value: 0.0093
##
## Engagement_Score and Bounce_Rate:
##   Correlation: -0.065
##   P-value: 0.0402
```

Looking at the correlation plot, we don't see any strong correlations between Visual Complexity and User Movement or any of the other metrics

Visual Complexity and Bounce Rate: > * very weak positive correlation > * it is statistically significant ($p < 0.01$), however the practical effect is unnoticeable

Engagement Score and Bounce Rate > * very weak negative correlation > * it is statistically significant ($p < 0.05$), however again, the practical effect is unnoticeable

Ad Characteristics Analysis

Now here's the interesting part, lets see which characteristics have the greatest impact on our metrics

Analysis of Ad Type Performance

```r
# Create summary statistics for Ad Type
ad_type_summary <- data %>%
  group_by(Ad_Type) %>%
```

```r
  summarise(across(c(Clicks, Time_Spent, Engagement_Score,
                    Conversion_Rate, Bounce_Rate, CTR),
                  list(
                    mean = ~mean(.x, na.rm = TRUE),
                    sd = ~sd(.x, na.rm = TRUE)
                  )),
           n = n()) %>%
  ungroup()

# Print summary statistics
cat("Summary Statistics by Ad Type:\n")
```

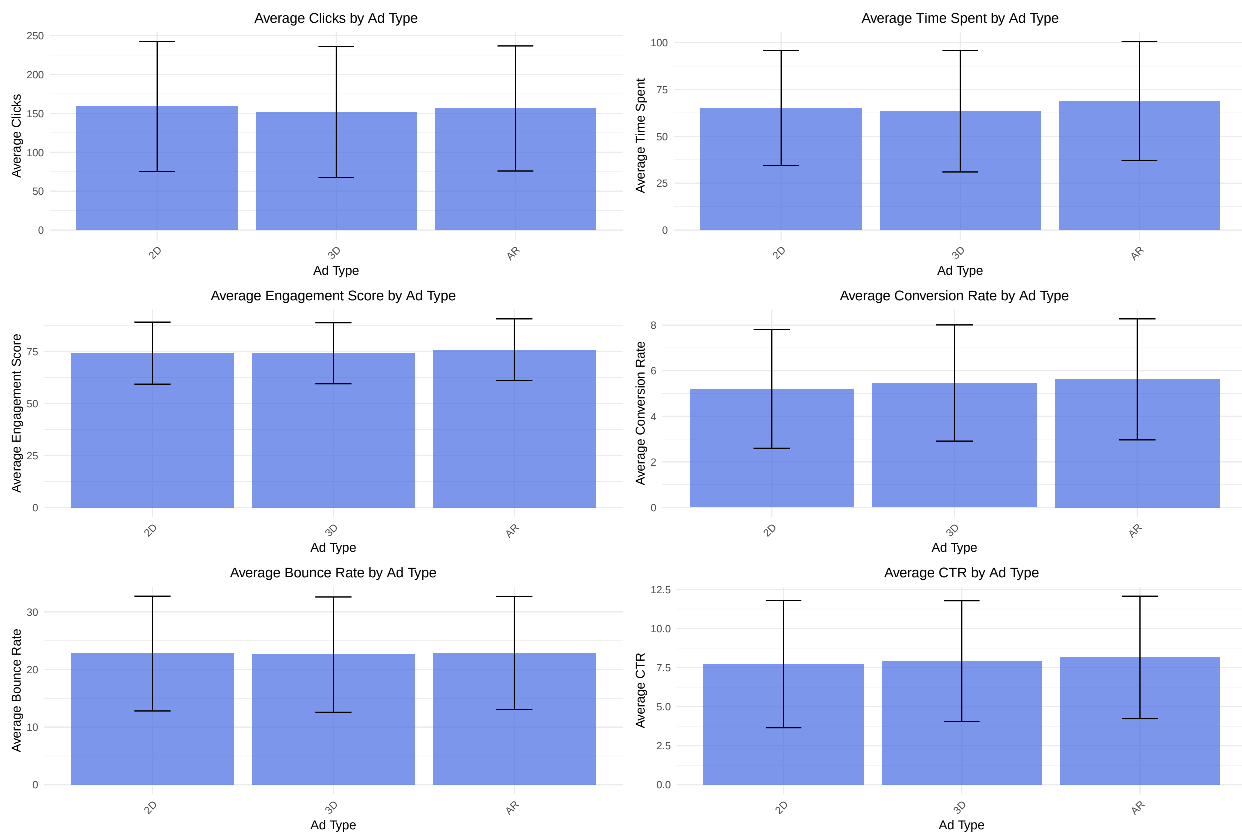
```
## Summary Statistics by Ad Type:
```

```r
print(ad_type_summary)
```

```
## # A tibble: 3 x 14
##   Ad_Type Clicks_mean Clicks_sd Time_Spent_mean Time_Spent_sd
##   <chr>         <dbl>     <dbl>           <dbl>         <dbl>
## 1 2D             159.      83.6            65.1          30.7
## 2 3D             152.      84.2            63.4          32.4
## 3 AR             156.      80.4            68.8          31.8
## # i 9 more variables: Engagement_Score_mean <dbl>, Engagement_Score_sd <dbl>,
## #   Conversion_Rate_mean <dbl>, Conversion_Rate_sd <dbl>,
## #   Bounce_Rate_mean <dbl>, Bounce_Rate_sd <dbl>, CTR_mean <dbl>, CTR_sd <dbl>,
## #   n <int>
```

```r
# Function to create a single metric plot
plot_metric <- function(data, metric_name) {
  ggplot(data, aes(x = Ad_Type, y = !!sym(paste0(metric_name, "_mean")))) +
    geom_bar(stat = "identity", fill = "#4169E1", alpha = 0.7) +
    geom_errorbar(aes(ymin = !!sym(paste0(metric_name, "_mean")) - !!sym(paste0(metric_name, "_sd")),
                      ymax = !!sym(paste0(metric_name, "_mean")) + !!sym(paste0(metric_name, "_sd"))),
                  width = 0.2) +
    theme_minimal() +
    labs(title = paste("Average", gsub("_", " ", metric_name), "by Ad Type"),
         x = "Ad Type",
         y = paste("Average", gsub("_", " ", metric_name))) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(hjust = 0.5, size = 12))
}

# Create plots for each metric
metrics <- c("Clicks", "Time_Spent", "Engagement_Score",
            "Conversion_Rate", "Bounce_Rate", "CTR")

plots <- lapply(metrics, function(metric) plot_metric(ad_type_summary, metric))

# Display plots in a grid with proper spacing
grid.arrange(grobs = plots,
            ncol = 2,
            widths = c(1, 1),
```

```
             heights = c(1, 1, 1),
             padding = unit(2, "line"))
```




```
# Perform ANOVA tests for each metric
cat("\nStatistical Analysis for Ad Type:\n")
```

```
##
## Statistical Analysis for Ad Type:
```

```
for (metric in metrics) {
  cat(paste("\n", gsub("_", " ", metric), "ANOVA Results:\n"))
  model <- aov(as.formula(paste(metric, "~ Ad_Type")), data = data)
  print(summary(model))

  # If ANOVA is significant, perform Tukey's test
  if (summary(model)[[1]]$"Pr(>F)"[1] < 0.05) {
    cat("\nTukey's HSD Test Results:\n")
    print(TukeyHSD(model))
  }
}
```

```
##
##  Clicks ANOVA Results:
##              Df  Sum Sq Mean Sq F value Pr(>F)
## Ad_Type       2    7932    3966   0.581   0.56
```

```
## Residuals   997 6807515    6828
##
##  Time Spent ANOVA Results:
##               Df  Sum Sq Mean Sq F value Pr(>F)
## Ad_Type        2    6033    3017   2.981 0.0512 .
## Residuals    997 1009009    1012
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Engagement Score ANOVA Results:
##               Df Sum Sq Mean Sq F value Pr(>F)
## Ad_Type        2    663   331.7   1.514  0.221
## Residuals    997 218463   219.1
##
##  Conversion Rate ANOVA Results:
##               Df Sum Sq Mean Sq F value Pr(>F)
## Ad_Type        2     23  11.704   1.729  0.178
## Residuals    997   6747   6.767
##
##  Bounce Rate ANOVA Results:
##               Df Sum Sq Mean Sq F value Pr(>F)
## Ad_Type        2     18    9.20   0.093  0.911
## Residuals    997  98149   98.44
##
##  CTR ANOVA Results:
##               Df Sum Sq Mean Sq F value Pr(>F)
## Ad_Type        2     26   12.99   0.841  0.431
## Residuals    997  15393   15.44
```

```r
# Create a summary table of best performing ad types
best_performers <- data.frame(
  Metric = character(),
  Best_Ad_Type = character(),
  Mean_Value = numeric(),
  Significant = character()
)

for (metric in metrics) {
  # Get best performing ad type
  best_idx <- which.max(ad_type_summary[[paste0(metric, "_mean")]])

  # Check significance
  model <- aov(as.formula(paste(metric, "~ Ad_Type")), data = data)
  is_significant <- summary(model)[[1]]$"Pr(>F)"[1] < 0.05

  best_performers <- rbind(best_performers, data.frame(
    Metric = metric,
    Best_Ad_Type = ad_type_summary$Ad_Type[best_idx],
    Mean_Value = round(ad_type_summary[[paste0(metric, "_mean")]][best_idx], 2),
    Significant = ifelse(is_significant, "Yes", "No")
  ))
}

cat("\nBest Performing Ad Types for Each Metric:\n")
```

```
##
## Best Performing Ad Types for Each Metric:
```

```r
print(best_performers)
```

```
##             Metric Best_Ad_Type Mean_Value Significant
## 1           Clicks           2D     158.78          No
## 2       Time_Spent           AR      68.84          No
## 3 Engagement_Score           AR      75.91          No
## 4  Conversion_Rate           AR       5.62          No
## 5      Bounce_Rate           AR      22.88          No
## 6              CTR           AR       8.15          No
```

Analysis of Age Group Performance

```r
# Create summary statistics for Ad Type
age_group_summary <- data %>%
  group_by(Age_Group) %>%
  summarise(across(c(Clicks, Time_Spent, Engagement_Score,
                     Conversion_Rate, Bounce_Rate, CTR),
                   list(
                     mean = ~mean(.x, na.rm = TRUE),
                     sd = ~sd(.x, na.rm = TRUE)
                   )),
            n = n()) %>%
  ungroup()

# Print summary statistics
cat("Summary Statistics by Age Group:\n")
```

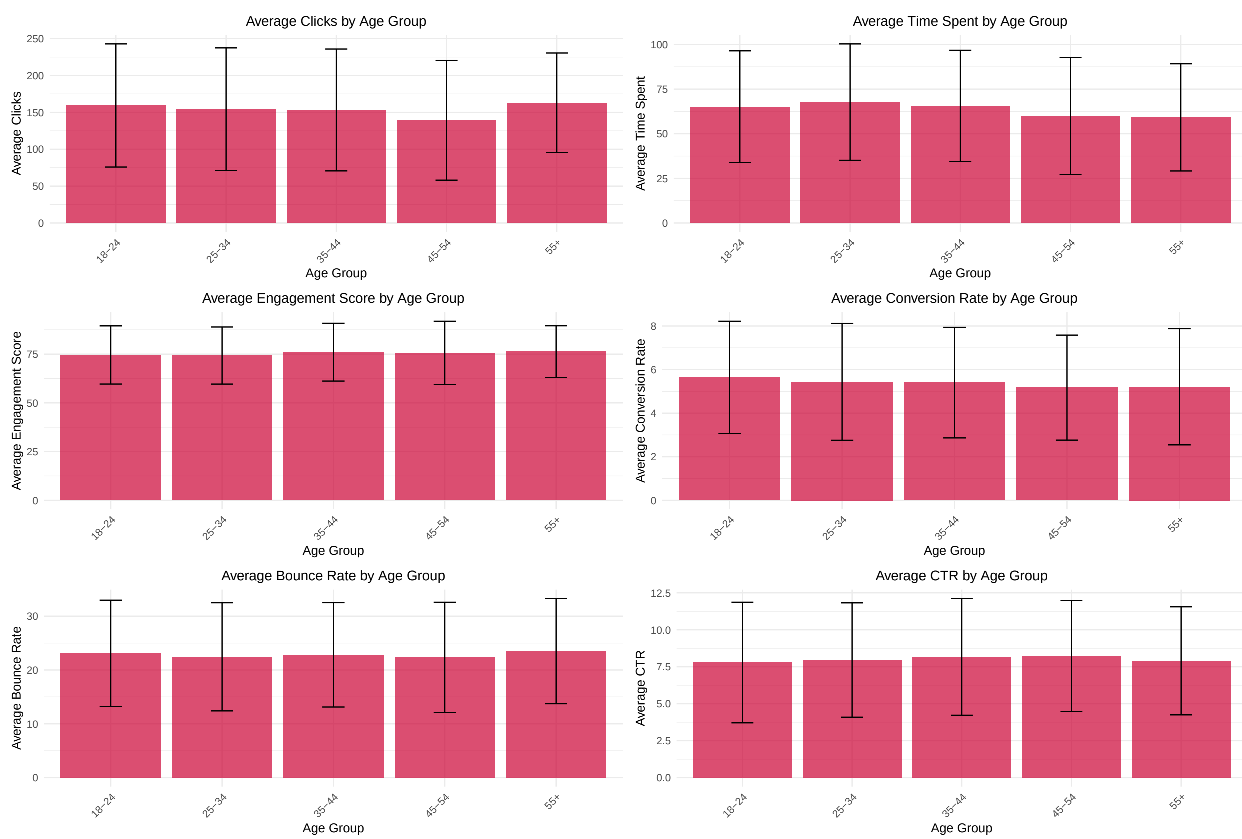
```
## Summary Statistics by Age Group:
```

```r
print(age_group_summary)
```

```
## # A tibble: 5 x 14
##   Age_Group Clicks_mean Clicks_sd Time_Spent_mean Time_Spent_sd
##   <chr>           <dbl>     <dbl>           <dbl>         <dbl>
## 1 18-24            159.      83.5            65.1          31.3
## 2 25-34            154.      83.2            67.7          32.6
## 3 35-44            153.      82.6            65.6          31.2
## 4 45-54            139.      81.2            59.9          32.8
## 5 55+              163.      67.5            59.2          30.0
## # i 9 more variables: Engagement_Score_mean <dbl>, Engagement_Score_sd <dbl>,
## #   Conversion_Rate_mean <dbl>, Conversion_Rate_sd <dbl>,
## #   Bounce_Rate_mean <dbl>, Bounce_Rate_sd <dbl>, CTR_mean <dbl>, CTR_sd <dbl>,
## #   n <int>
```

```r
# Function to create a single metric plot
plot_metric <- function(data, metric_name) {
  ggplot(data, aes(x = Age_Group, y = !!sym(paste0(metric_name, "_mean")))) +
```

```r
    geom_bar(stat = "identity", fill = "#CC0033", alpha = 0.7) +
    geom_errorbar(aes(ymin = !!sym(paste0(metric_name, "_mean")) - !!sym(paste0(metric_name, "_sd")),
                      ymax = !!sym(paste0(metric_name, "_mean")) + !!sym(paste0(metric_name, "_sd"))),
                  width = 0.2) +
    theme_minimal() +
    labs(title = paste("Average", gsub("_", " ", metric_name), "by Age Group"),
         x = "Age Group",
         y = paste("Average", gsub("_", " ", metric_name))) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(hjust = 0.5, size = 12))
}

# Create plots for each metric
metrics <- c("Clicks", "Time_Spent", "Engagement_Score",
            "Conversion_Rate", "Bounce_Rate", "CTR")

plots <- lapply(metrics, function(metric) plot_metric(age_group_summary, metric))

# Display plots in a grid with proper spacing
grid.arrange(grobs = plots,
            ncol = 2,
            widths = c(1, 1),
            heights = c(1, 1, 1),
            padding = unit(2, "line"))
```



```r
# Perform ANOVA tests for each metric
cat("\nStatistical Analysis for Age Group:\n")
```

```
##
## Statistical Analysis for Age Group:
```

```r
for (metric in metrics) {
  cat(paste("\n", gsub("_", " ", metric), "ANOVA Results:\n"))
  model <- aov(as.formula(paste(metric, "~ Age_Group")), data = data)
  print(summary(model))

  # If ANOVA is significant, perform Tukey's test
  if (summary(model)[[1]]$"Pr(>F)"[1] < 0.05) {
    cat("\nTukey's HSD Test Results:\n")
    print(TukeyHSD(model))
  }
}
```

```
##
##  Clicks ANOVA Results:
##              Df  Sum Sq Mean Sq F value Pr(>F)
## Age_Group     4   22808    5702   0.835  0.503
## Residuals   995 6792638    6827
##
##  Time Spent ANOVA Results:
##              Df  Sum Sq Mean Sq F value Pr(>F)
## Age_Group     4    4994    1249    1.23  0.296
## Residuals   995 1010048    1015
##
##  Engagement Score ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Age_Group     4    537   134.3   0.611  0.655
## Residuals   995 218590   219.7
##
##  Conversion Rate ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Age_Group     4     17   4.368   0.644  0.631
## Residuals   995   6753   6.787
##
##  Bounce Rate ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Age_Group     4     98   24.51   0.249  0.911
## Residuals   995  98069   98.56
##
##  CTR ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Age_Group     4     22   5.541   0.358  0.838
## Residuals   995  15397  15.474
```

```r
# Create a summary table of best performing ad types
best_performers <- data.frame(
  Metric = character(),
```

```r
  Best_Age_Group = character(),
  Mean_Value = numeric(),
  Significant = character()
)

for (metric in metrics) {
  # Get best performing ad type
  best_idx <- which.max(age_group_summary[[paste0(metric, "_mean")]])

  # Check significance
  model <- aov(as.formula(paste(metric, "~ Age_Group")), data = data)
  is_significant <- summary(model)[[1]]$"Pr(>F)"[1] < 0.05

  best_performers <- rbind(best_performers, data.frame(
    Metric = metric,
    Best_Age_Group = age_group_summary$Age_Group[best_idx],
    Mean_Value = round(age_group_summary[[paste0(metric, "_mean")]][best_idx], 2),
    Significant = ifelse(is_significant, "Yes", "No")
  ))
}

cat("\nMost Impacted Age Groups for Each Metric:\n")
```

```
##
## Most Impacted Age Groups for Each Metric:
```

```r
print(best_performers)
```

```
##             Metric Best_Age_Group Mean_Value Significant
## 1           Clicks            55+     162.97          No
## 2       Time_Spent          25-34      67.72          No
## 3 Engagement_Score            55+      76.30          No
## 4  Conversion_Rate          18-24       5.64          No
## 5      Bounce_Rate            55+      23.50          No
## 6              CTR          45-54       8.23          No
```

Analysis of Device Type Performance

```r
# Create summary statistics for Device Type
device_type_summary <- data %>%
  group_by(Device_Type) %>%
  summarise(across(c(Clicks, Time_Spent, Engagement_Score,
                     Conversion_Rate, Bounce_Rate, CTR),
                   list(
                     mean = ~mean(.x, na.rm = TRUE),
                     sd = ~sd(.x, na.rm = TRUE)
                   )),
            n = n()) %>%
  ungroup()
```

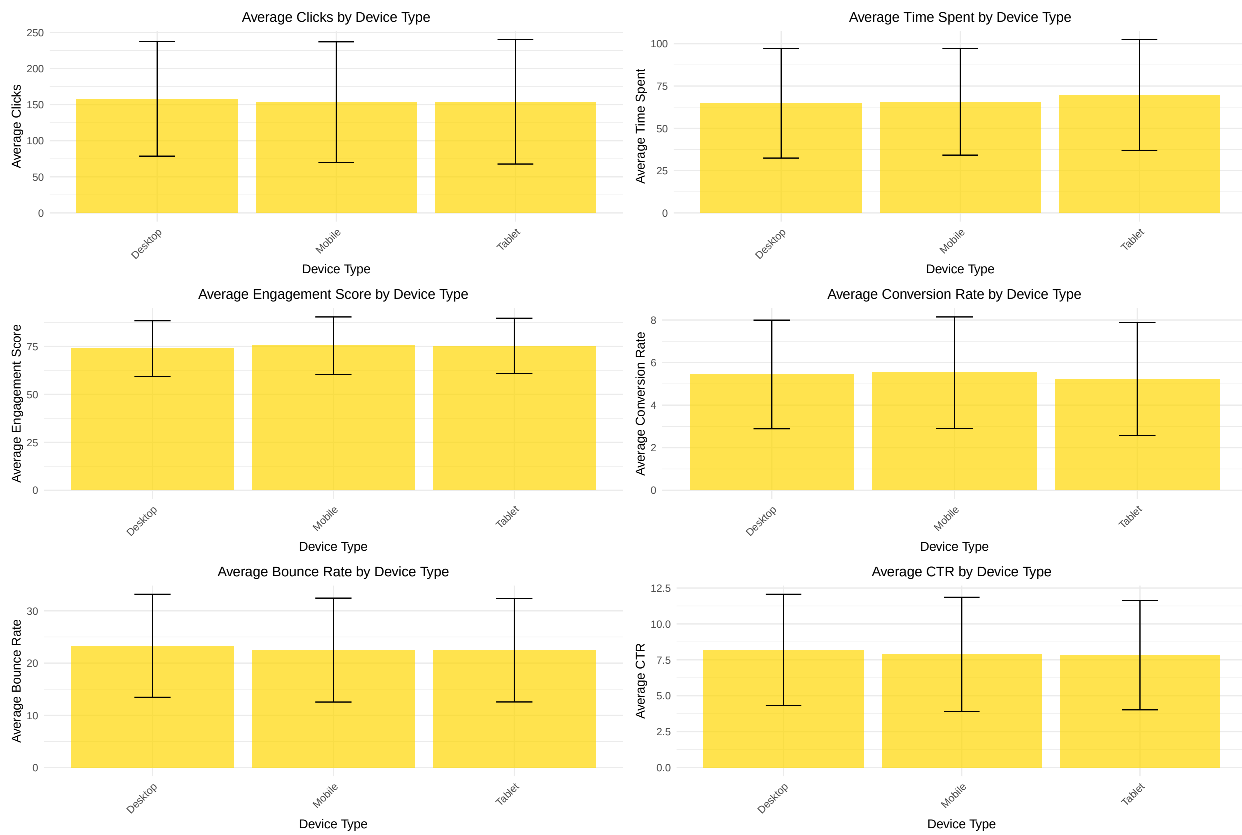
```r
# Print summary statistics
cat("Summary Statistics by Device Type:\n")
```

```
## Summary Statistics by Device Type:
```

```r
print(device_type_summary)
```

```
## # A tibble: 3 x 14
##   Device_Type Clicks_mean Clicks_sd Time_Spent_mean Time_Spent_sd
##   <chr>             <dbl>     <dbl>           <dbl>         <dbl>
## 1 Desktop            158.      79.5            64.8          32.3
## 2 Mobile             153.      83.6            65.7          31.5
## 3 Tablet             154.      86.2            69.7          32.8
## # i 9 more variables: Engagement_Score_mean <dbl>, Engagement_Score_sd <dbl>,
## #   Conversion_Rate_mean <dbl>, Conversion_Rate_sd <dbl>,
## #   Bounce_Rate_mean <dbl>, Bounce_Rate_sd <dbl>, CTR_mean <dbl>, CTR_sd <dbl>,
## #   n <int>
```

```r
# Function to create a single metric plot
plot_metric <- function(data, metric_name) {
  ggplot(data, aes(x = Device_Type, y = !!sym(paste0(metric_name, "_mean")))) +
    geom_bar(stat = "identity", fill = "#FFD700", alpha = 0.7) +  # Changed to yellow
    geom_errorbar(aes(ymin = !!sym(paste0(metric_name, "_mean")) - !!sym(paste0(metric_name, "_sd")),
                      ymax = !!sym(paste0(metric_name, "_mean")) + !!sym(paste0(metric_name, "_sd"))),
                  width = 0.2) +
    theme_minimal() +
    labs(title = paste("Average", gsub("_", " ", metric_name), "by Device Type"),
         x = "Device Type",
         y = paste("Average", gsub("_", " ", metric_name))) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(hjust = 0.5, size = 12))
}

# Create plots for each metric
metrics <- c("Clicks", "Time_Spent", "Engagement_Score",
            "Conversion_Rate", "Bounce_Rate", "CTR")

plots <- lapply(metrics, function(metric) plot_metric(device_type_summary, metric))

# Display plots in a grid with proper spacing
grid.arrange(grobs = plots,
            ncol = 2,
            widths = c(1, 1),
            heights = c(1, 1, 1),
            padding = unit(2, "line"))
```




```r
# Perform ANOVA tests for each metric
cat("\nStatistical Analysis for Device Type:\n")
```

```
##
## Statistical Analysis for Device Type:
```

```r
for (metric in metrics) {
  cat(paste("\n", gsub("_", " ", metric), "ANOVA Results:\n"))
  model <- aov(as.formula(paste(metric, "~ Device_Type")), data = data)
  print(summary(model))

  # If ANOVA is significant, perform Tukey's test
  if (summary(model)[[1]]$"Pr(>F)"[1] < 0.05) {
    cat("\nTukey's HSD Test Results:\n")
    print(TukeyHSD(model))
  }
}
```

```
##
##  Clicks ANOVA Results:
##              Df  Sum Sq Mean Sq F value Pr(>F)
## Device_Type   2    4362    2181   0.319  0.727
## Residuals   997 6811085    6832
##
##  Time Spent ANOVA Results:
##              Df  Sum Sq Mean Sq F value Pr(>F)
```

```
## Device_Type   2    1878   938.9   0.924  0.397
## Residuals   997 1013164  1016.2
##
##  Engagement Score ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Device_Type   2    481   240.4   1.096  0.334
## Residuals   997 218646   219.3
##
##  Conversion Rate ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Device_Type   2      8   3.957   0.583  0.558
## Residuals   997   6763   6.783
##
##  Bounce Rate ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Device_Type   2    143   71.30   0.725  0.484
## Residuals   997  98025   98.32
##
##  CTR ANOVA Results:
##              Df Sum Sq Mean Sq F value Pr(>F)
## Device_Type   2     22   10.85   0.703  0.495
## Residuals   997  15397   15.44
```

```r
# Create a summary table of best performing device types
best_performers <- data.frame(
  Metric = character(),
  Best_Device_Type = character(),
  Mean_Value = numeric(),
  Significant = character()
)

for (metric in metrics) {
  # Get best performing device type
  best_idx <- which.max(device_type_summary[[paste0(metric, "_mean")]])

  # Check significance
  model <- aov(as.formula(paste(metric, "~ Device_Type")), data = data)
  is_significant <- summary(model)[[1]]$"Pr(>F)"[1] < 0.05

  best_performers <- rbind(best_performers, data.frame(
    Metric = metric,
    Best_Device_Type = device_type_summary$Device_Type[best_idx],
    Mean_Value = round(device_type_summary[[paste0(metric, "_mean")]][best_idx], 2),
    Significant = ifelse(is_significant, "Yes", "No")
  ))
}

cat("\nBest Performing Device Types for Each Metric:\n")
```

```
##
## Best Performing Device Types for Each Metric:
```

```r
print(best_performers)
```

```
##             Metric Best_Device_Type Mean_Value Significant
## 1           Clicks          Desktop     158.13          No
## 2       Time_Spent           Tablet      69.68          No
## 3 Engagement_Score           Mobile      75.33          No
## 4  Conversion_Rate           Mobile       5.52          No
## 5      Bounce_Rate          Desktop      23.32          No
## 6              CTR          Desktop       8.19          No
```

Analysis of Visual Complexity Performance

```r
# Create summary statistics for Visual Complexity
visual_complexity_summary <- data %>%
  group_by(Visual_Complexity) %>%
  summarise(across(c(Clicks, Time_Spent, Engagement_Score,
                    Conversion_Rate, Bounce_Rate, CTR),
                  list(
                    mean = ~mean(.x, na.rm = TRUE),
                    sd = ~sd(.x, na.rm = TRUE)
                  )),
           n = n()) %>%
  ungroup()

# Print summary statistics
cat("Summary Statistics by Visual Complexity:\n")
```

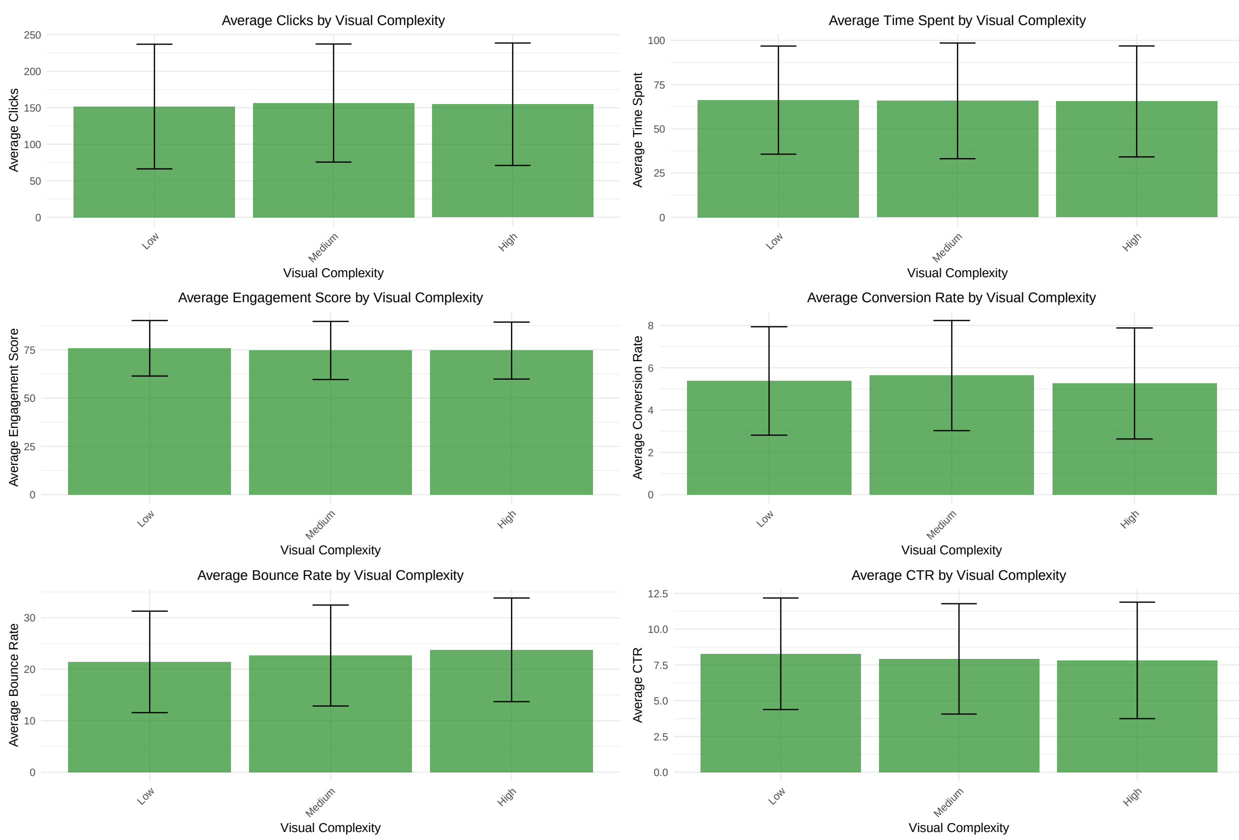
```
## Summary Statistics by Visual Complexity:
```

```r
print(visual_complexity_summary)
```

```
## # A tibble: 3 x 14
##   Visual_Complexity Clicks_mean Clicks_sd Time_Spent_mean Time_Spent_sd
##   <fct>                   <dbl>     <dbl>           <dbl>         <dbl>
## 1 Low                      152.      85.4            66.2          30.6
## 2 Medium                   156.      80.8            65.8          32.7
## 3 High                     155.      83.9            65.5          31.4
## # i 9 more variables: Engagement_Score_mean <dbl>, Engagement_Score_sd <dbl>,
## #   Conversion_Rate_mean <dbl>, Conversion_Rate_sd <dbl>,
## #   Bounce_Rate_mean <dbl>, Bounce_Rate_sd <dbl>, CTR_mean <dbl>, CTR_sd <dbl>,
## #   n <int>
```

```r
# Function to create a single metric plot
plot_metric <- function(data, metric_name) {
  ggplot(data, aes(x = Visual_Complexity, y = !!sym(paste0(metric_name, "_mean")))) +
    geom_bar(stat = "identity", fill = "#228B22", alpha = 0.7) +  # Changed to forest green
    geom_errorbar(aes(ymin = !!sym(paste0(metric_name, "_mean")) - !!sym(paste0(metric_name, "_sd")),
                      ymax = !!sym(paste0(metric_name, "_mean")) + !!sym(paste0(metric_name, "_sd"))),
                  width = 0.2) +
    theme_minimal() +
```

```r
    labs(title = paste("Average", gsub("_", " ", metric_name), "by Visual Complexity"),
         x = "Visual Complexity",
         y = paste("Average", gsub("_", " ", metric_name))) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(hjust = 0.5, size = 12))
}

# Create plots for each metric
metrics <- c("Clicks", "Time_Spent", "Engagement_Score",
            "Conversion_Rate", "Bounce_Rate", "CTR")

plots <- lapply(metrics, function(metric) plot_metric(visual_complexity_summary, metric))

# Display plots in a grid with proper spacing
grid.arrange(grobs = plots,
            ncol = 2,
            widths = c(1, 1),
            heights = c(1, 1, 1),
            padding = unit(2, "line"))
```




```r
# Perform ANOVA tests for each metric
cat("\nStatistical Analysis for Visual Complexity:\n")
```

```
##
## Statistical Analysis for Visual Complexity:
```

```r
for (metric in metrics) {
  cat(paste("\n", gsub("_", " ", metric), "ANOVA Results:\n"))
  model <- aov(as.formula(paste(metric, "~ Visual_Complexity")), data = data)
  print(summary(model))

  # If ANOVA is significant, perform Tukey's test
  if (summary(model)[[1]]$"Pr(>F)"[1] < 0.05) {
    cat("\nTukey's HSD Test Results:\n")
    print(TukeyHSD(model))
  }
}
```

```
##
##  Clicks ANOVA Results:
##                    Df  Sum Sq Mean Sq F value Pr(>F)
## Visual_Complexity   2    3349    1674   0.245  0.783
## Residuals         997 6812098    6833
##
##  Time Spent ANOVA Results:
##                    Df  Sum Sq Mean Sq F value Pr(>F)
## Visual_Complexity   2      68    33.9   0.033  0.967
## Residuals         997 1014974  1018.0
##
##  Engagement Score ANOVA Results:
##                    Df Sum Sq Mean Sq F value Pr(>F)
## Visual_Complexity   2    216   108.1   0.492  0.611
## Residuals         997 218911   219.6
##
##  Conversion Rate ANOVA Results:
##                    Df Sum Sq Mean Sq F value Pr(>F)
## Visual_Complexity   2     28  13.893   2.054  0.129
## Residuals         997   6743   6.763
##
##  Bounce Rate ANOVA Results:
##                    Df Sum Sq Mean Sq F value Pr(>F)
## Visual_Complexity   2    664   331.9   3.394  0.034 *
## Residuals         997  97503    97.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Tukey's HSD Test Results:
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = as.formula(paste(metric, "~ Visual_Complexity")), data = data)
##
## $Visual_Complexity
##                 diff        lwr      upr     p adj
## Medium-Low  1.236859 -0.6927145 3.166432 0.2891785
## High-Low    2.343086  0.2233881 4.462783 0.0259893
## High-Medium 1.106227 -0.5990504 2.811504 0.2806779
##
##
```

```
##  CTR ANOVA Results:
##                    Df Sum Sq Mean Sq F value Pr(>F)
## Visual_Complexity   2     28   13.80   0.894  0.409
## Residuals         997  15391   15.44
```

```r
# Create a summary table of best performing visual complexity levels
best_performers <- data.frame(
  Metric = character(),
  Best_Visual_Complexity = character(),
  Mean_Value = numeric(),
  Significant = character()
)

for (metric in metrics) {
  # Get best performing visual complexity level
  best_idx <- which.max(visual_complexity_summary[[paste0(metric, "_mean")]])

  # Check significance
  model <- aov(as.formula(paste(metric, "~ Visual_Complexity")), data = data)
  is_significant <- summary(model)[[1]]$"Pr(>F)"[1] < 0.05

  best_performers <- rbind(best_performers, data.frame(
    Metric = metric,
    Best_Visual_Complexity = visual_complexity_summary$Visual_Complexity[best_idx],
    Mean_Value = round(visual_complexity_summary[[paste0(metric, "_mean")]][best_idx], 2),
    Significant = ifelse(is_significant, "Yes", "No")
  ))
}

cat("\nBest Performing Visual Complexity Levels for Each Metric:\n")
```

```
##
## Best Performing Visual Complexity Levels for Each Metric:
```

```r
print(best_performers)
```

```
##             Metric Best_Visual_Complexity Mean_Value Significant
## 1           Clicks                 Medium     156.38          No
## 2       Time_Spent                    Low      66.25          No
## 3 Engagement_Score                    Low      75.79          No
## 4  Conversion_Rate                 Medium       5.63          No
## 5      Bounce_Rate                   High      23.77         Yes
## 6              CTR                    Low       8.28          No
```

Analysis of User Movement Performance

```r
# Create summary statistics for User Movement
user_movement_summary <- data %>%
  group_by(User_Movement_Data) %>%
  summarise(across(c(Clicks, Time_Spent, Engagement_Score,
                    Conversion_Rate, Bounce_Rate, CTR),
```

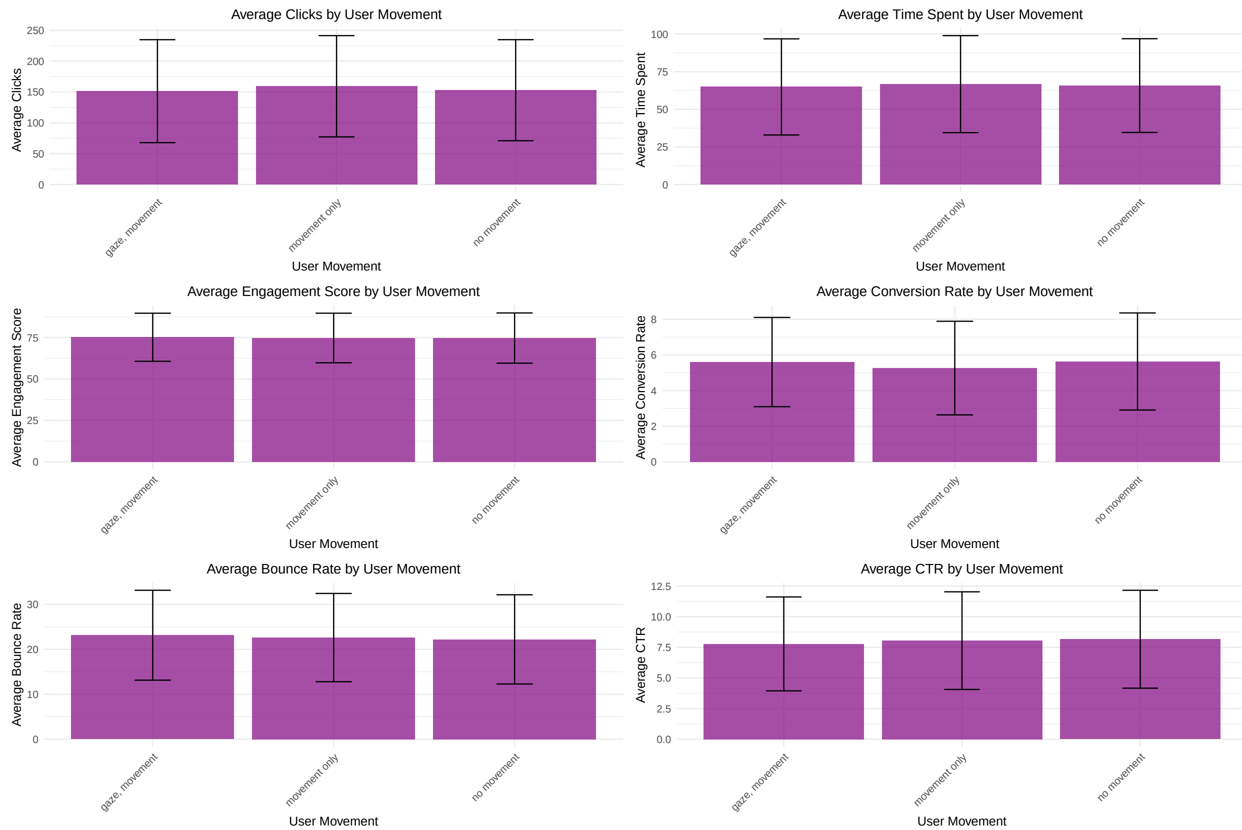
```r
                  list(
                    mean = ~mean(.x, na.rm = TRUE),
                    sd = ~sd(.x, na.rm = TRUE)
                  )),
           n = n()) %>%
  ungroup()

# Print summary statistics
cat("Summary Statistics by User Movement:\n")
```

```
## Summary Statistics by User Movement:
```

```r
print(user_movement_summary)
```

```
## # A tibble: 3 x 14
##   User_Movement_Data Clicks_mean Clicks_sd Time_Spent_mean Time_Spent_sd
##   <chr>                    <dbl>     <dbl>           <dbl>         <dbl>
## 1 gaze, movement            151.      83.5            64.9          32.0
## 2 movement only             159.      82.1            66.7          32.2
## 3 no movement               153.      81.9            65.8          31.1
## # i 9 more variables: Engagement_Score_mean <dbl>, Engagement_Score_sd <dbl>,
## #   Conversion_Rate_mean <dbl>, Conversion_Rate_sd <dbl>,
## #   Bounce_Rate_mean <dbl>, Bounce_Rate_sd <dbl>, CTR_mean <dbl>, CTR_sd <dbl>,
## #   n <int>
```

```r
# Function to create a single metric plot
plot_metric <- function(data, metric_name) {
  ggplot(data, aes(x = User_Movement_Data, y = !!sym(paste0(metric_name, "_mean")))) +
    geom_bar(stat = "identity", fill = "#800080", alpha = 0.7) +  # Changed to purple
    geom_errorbar(aes(ymin = !!sym(paste0(metric_name, "_mean")) - !!sym(paste0(metric_name, "_sd")),
                      ymax = !!sym(paste0(metric_name, "_mean")) + !!sym(paste0(metric_name, "_sd"))),
                  width = 0.2) +
    theme_minimal() +
    labs(title = paste("Average", gsub("_", " ", metric_name), "by User Movement"),
         x = "User Movement",
         y = paste("Average", gsub("_", " ", metric_name))) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(hjust = 0.5, size = 12))
}

# Create plots for each metric
metrics <- c("Clicks", "Time_Spent", "Engagement_Score",
            "Conversion_Rate", "Bounce_Rate", "CTR")

plots <- lapply(metrics, function(metric) plot_metric(user_movement_summary, metric))

# Display plots in a grid with proper spacing
grid.arrange(grobs = plots,
            ncol = 2,
            widths = c(1, 1),
            heights = c(1, 1, 1),
            padding = unit(2, "line"))
```




```r
# Perform ANOVA tests for each metric
cat("\nStatistical Analysis for User Movement:\n")
```

```
##
## Statistical Analysis for User Movement:
```

```r
for (metric in metrics) {
  cat(paste("\n", gsub("_", " ", metric), "ANOVA Results:\n"))
  model <- aov(as.formula(paste(metric, "~ User_Movement_Data")), data = data)
  print(summary(model))

  # If ANOVA is significant, perform Tukey's test
  if (summary(model)[[1]]$"Pr(>F)"[1] < 0.05) {
    cat("\nTukey's HSD Test Results:\n")
    print(TukeyHSD(model))
  }
}
```

```
##
##  Clicks ANOVA Results:
##                      Df  Sum Sq Mean Sq F value Pr(>F)
## User_Movement_Data    2   13464    6732   0.987  0.373
## Residuals           997 6801982    6822
##
##  Time Spent ANOVA Results:
##                      Df  Sum Sq Mean Sq F value Pr(>F)
```

```
## User_Movement_Data   2     684   342.2   0.336  0.714
## Residuals          997 1014358  1017.4
##
##  Engagement Score ANOVA Results:
##                     Df Sum Sq Mean Sq F value Pr(>F)
## User_Movement_Data   2     49   24.58   0.112  0.894
## Residuals          997 219078  219.74
##
##  Conversion Rate ANOVA Results:
##                     Df Sum Sq Mean Sq F value Pr(>F)
## User_Movement_Data   2     30  14.767   2.184  0.113
## Residuals          997   6741   6.761
##
##  Bounce Rate ANOVA Results:
##                     Df Sum Sq Mean Sq F value Pr(>F)
## User_Movement_Data   2    127   63.32   0.644  0.525
## Residuals          997  98041   98.34
##
##  CTR ANOVA Results:
##                     Df Sum Sq Mean Sq F value Pr(>F)
## User_Movement_Data   2     24   12.05    0.78  0.459
## Residuals          997  15395   15.44
```

```r
# Create a summary table of best performing user movement types
best_performers <- data.frame(
  Metric = character(),
  Best_User_Movement = character(),
  Mean_Value = numeric(),
  Significant = character()
)

for (metric in metrics) {
  # Get best performing user movement type
  best_idx <- which.max(user_movement_summary[[paste0(metric, "_mean")]])

  # Check significance
  model <- aov(as.formula(paste(metric, "~ User_Movement_Data")), data = data)
  is_significant <- summary(model)[[1]]$"Pr(>F)"[1] < 0.05

  best_performers <- rbind(best_performers, data.frame(
    Metric = metric,
    Best_User_Movement = user_movement_summary$User_Movement_Data[best_idx],
    Mean_Value = round(user_movement_summary[[paste0(metric, "_mean")]][best_idx], 2),
    Significant = ifelse(is_significant, "Yes", "No")
  ))
}

cat("\nBest Performing User Movement Types for Each Metric:\n")
```

```
##
## Best Performing User Movement Types for Each Metric:
```

```r
print(best_performers)
```

```
##             Metric Best_User_Movement Mean_Value Significant
## 1           Clicks      movement only     159.33          No
## 2       Time_Spent      movement only      66.73          No
## 3 Engagement_Score     gaze, movement      75.15          No
## 4  Conversion_Rate        no movement       5.63          No
## 5      Bounce_Rate     gaze, movement      23.14          No
## 6              CTR        no movement       8.16          No
```