Data science

Alireza Sadeghi

Table of contents

Introduction

In this project, I used RStudio to replicate a graph and an Excel table from the paper (Agranov, Eraslan and Tergiman, 2024) . The goal of this work was to demonstrate my ability to use R for data analysis, visualization, and data handling.

To complete the replication, I worked with three main R packages: tidyverse(Wickham *et al.*, 2019), ggplot2,(Wickham, 2016) and readxl (Wickham and Bryan, 2025) The *tidyverse* package was essential for organizing and cleaning the data, making it easier to process and analyse.

The *ggplot2* (Wickham, 2016) was used to recreate the graph, providing a clear and professional visualization. The *readxl* package (Wickham and Bryan, 2025) was applied to import and handle the Excel data efficiently within RStudio.

Finally, the *readxl* package (Wickham and Bryan, 2025) enabled me to import data directly from Excel files into R, which is a common and practical skill for real-world data projects.

By successfully reproducing the table and the graph, I demonstrated my ability to use R effectively for both data management and visualization tasks . This process also enhanced my understanding of how R can be applied to replicate results from published research, an essential aspect of ensuring reliable and transparent data analysis (Agranov, Eraslan and Tergiman, 2024)

Analysis

The data used for this project were obtained by following the instructions provided in the project's README file. The raw data were originally processed through several Stata `.do` files designed to prepare and merge experimental datasets. Since the analysis was conducted in RStudio, the Stata code was carefully translated into R scripts with the help of ChatGPT to ensure full compatibility.

Understanding the Workflow

After reviewing the README documentation, it was clear that the data construction process relied on two key Stata scripts:

`prepare_bargaining.do`

`prepare_chats.do`

These scripts perform data cleaning, variable transformations, and merging steps required to construct the final dataset used in the analysis.

Code Conversion and Adaptation in R

The Stata commands were rewritten in R, maintaining the same logic and structure.
Functions from the following R packages were primarily used:

`haven` — for reading and writing Stata `.dta` files

`dplyr` — for data manipulation (`filter`, `mutate`, `group_by`, `summarize`)

`janitor` — for frequency tables and descriptive statistics

lmtest and **sandwich** — for regression analysis with cluster-robust standard error.

Each Stata operation (e.g., `use`, `drop`, `gen`, `merge`, `probit`) was replaced with its equivalent R syntax to preserve data integrity.

The translated R scripts were executed sequentially in RStudio:

The `prepare_bargaining` script processed experimental bargaining session data.

```r
# Load the readxl package (install if needed)
install.packages("readxl")  # Run once if not installed
library(readxl)

# Import the Excel file, using the first row as column names
data <- read_excel("E:/My education/UNI/edu uni/Semester 2/Data science for Business/data.20


# Load necessary package
install.packages("dplyr")  # Run once if not installed
library(dplyr)

# Assuming your dataset is called 'data'
data <- data %>%
  # Create 'id' as a grouped identifier like egen group()
  mutate(id = as.numeric(factor(paste(sessioncode, memberid, sep = "_"))),

         # Generate treatment indicators
         m24 = (votingtreatment == "majority" & pietreatment == 24),
         m48 = (votingtreatment == "majority" & pietreatment == 48),
         m96 = (votingtreatment == "majority" & pietreatment == 96),

         u24 = (votingtreatment == "unanimity" & pietreatment == 24),
         u48 = (votingtreatment == "unanimity" & pietreatment == 48),
         u96 = (votingtreatment == "unanimity" & pietreatment == 96))
# Load packages
library(dplyr)
library(haven)   # for saving .dta files if you need to

# Assuming your dataset is called 'data'
data <- data %>%
  # Total yes votes
  mutate(
    totalyes = vote1 + vote2 + vote3,

    # Pass condition
    pass = case_when(
      is.na(totalyes) ~ NA,
      (totalyes == 3 & votingtreatment == "unanimity") |
        (totalyes >= 2 & votingtreatment == "majority") ~ TRUE,
      TRUE ~ FALSE
    ),

    # Treatment variable
    treatment = case_when(
      m24 == 1 ~ "m24",
      m48 == 1 ~ "m48",
      m96 == 1 ~ "m96",
      u24 == 1 ~ "u24",
      u48 == 1 ~ "u48",
      u96 == 1 ~ "u96",
      TRUE ~ ""
    ),

    # Unanimity dummy
    unanimity = (votingtreatment == "unanimity"),

    # Session assignment
    session = case_when(
      sessioncode %in% c("50catrnf","7f7v4l6r","js437fzd","m1","m5","q32zfvnq") ~ 1,
      sessioncode %in% c("9gzbr7vg","ax2x1ab5","ifm6r51c","krgxtmp1","m2","m6") ~ 2,
      sessioncode %in% c("6nf5jbi5","gx0695sj","jvzb19el","m3","p66q0jhd","w2npj4ao") ~ 3,
      sessioncode %in% c("9fuzcnjg","ixhiq3bn","job0mwmb","jwlbjm39","m4","wfwdfk01") ~ 4,
      TRUE ~ NA_real_
    )
  )

# Create unique session IDs
data <- data %>%
  mutate(
    uniquesession = case_when(
      votingtreatment == "majority" & pietreatment == 24 ~ as.numeric(paste0("240", session))
      votingtreatment == "majority" & pietreatment == 48 ~ as.numeric(paste0("480", session))
      votingtreatment == "majority" & pietreatment == 96 ~ as.numeric(paste0("960", session))
      votingtreatment == "unanimity" & pietreatment == 24 ~ as.numeric(paste0("241", session
      votingtreatment == "unanimity" & pietreatment == 48 ~ as.numeric(paste0("481", session
      votingtreatment == "unanimity" & pietreatment == 96 ~ as.numeric(paste0("961", session
      TRUE ~ NA_real_
    )
  )

# Save as Stata .dta file
write_dta(data, "E:/My education/UNI/edu uni/Semester 2/Data science for Business/Bargaining
library(readxl)
library(dplyr)
library(haven)
library(janitor)
library(stringr)

# Define paths
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
excel_file <- file.path(base_path, "risk.xlsx")
output_file <- file.path(base_path, "risk.dta")

# Function to import one treatment group (e.g., U1_s1-U1_s4)
import_group <- function(prefix, treatment, unanimity, pie = NA, n_sessions = 4) {
  bind_rows(lapply(1:n_sessions, function(i) {
    sheet_name <- paste0(prefix, "_s", i)
    message("Importing sheet: ", sheet_name)

    df <- read_excel(excel_file, sheet = sheet_name) %>%
      clean_names() %>%
      mutate(session = i,
             treatment = treatment,
             unanimity = unanimity,
             pie = pie)
    return(df)
  }))
}

# ---- Import all treatment groups ----
```

```
risk_data <- bind_rows(
  import_group("U1", "u24", unanimity = 1, pie = 24),
  import_group("U2", "u48", unanimity = 1, pie = 48),
  import_group("U4", "u96", unanimity = 1, pie = 96),
  import_group("M1", "m24", unanimity = 0, pie = 24),
  import_group("M2", "m48", unanimity = 0, pie = 48),
  import_group("M4", "m96", unanimity = 0, pie = 96)
)

# ---- Fix long variable names for Stata ----
names(risk_data) <- names(risk_data) %>%
  str_sub(1, 32)  # shorten to first 32 characters (Stata's max limit)

# ---- Clean up: drop rows with missing session_code ----
if ("session_code" %in% names(risk_data)) {
  risk_data <- risk_data %>%
    filter(!(is.na(session_code) | session_code == ""))
}

# ---- Save final combined dataset ----
write_dta(risk_data, output_file)
message("  Combined risk data saved to: ", output_file)
library(haven)

# Load the Stata dataset
risk_data <- read_dta("E:/My education/UNI/edu uni/Semester 2/Data science for Business/risk

# View the first few rows
head(risk_data)

# Optional: open in RStudio's data viewer
View(risk_data)
library(dplyr)
library(tidyr)
library(haven)

# Load the risk dataset
risk_data <- read_dta("E:/My education/UNI/edu uni/Semester 2/Data science for Business/risk

# --- Rename columns to match Stata logic ---
risk_data <- risk_data %>%
  rename(
    participantcoderisk = participant_code,        # correct column name
    memberid = participant_id_in_session,         # correct column name
    playerinvested = player_invested              # correct column name
  )

# --- Create riskmultiplier ---
risk_data <- risk_data %>%
  mutate(
    riskmultiplier = case_when(
      subsession_investment_multiplier == 3 ~ 2,
      subsession_investment_multiplier == 2.5 ~ 1,
      TRUE ~ NA_real_
    )
  )

# --- Create unique ID like Stata's egen group() ---
risk_data <- risk_data %>%
  mutate(id = as.numeric(factor(paste(unanimity, pie, session, memberid, sep = "_"))))

# --- Keep only relevant columns ---
risk_data <- risk_data %>%
  select(participantcoderisk, memberid, session_code, riskmultiplier,
         playerinvested, id, session, unanimity)

# --- Reshape wide: playerinvested by riskmultiplier ---
risk_data_wide <- risk_data %>%
  pivot_wider(
    names_from = riskmultiplier,
    values_from = playerinvested,
    names_prefix = "playerinvested"
  ) %>%
  rename(
    playerinvested25 = playerinvested1,
    playerinvested30 = playerinvested2
  ) %>%
  select(-id) %>%
  arrange(session_code, memberid)

# --- Save as Stata .dta file ---
write_dta(risk_data_wide, "E:/My education/UNI/edu uni/Semester 2/Data science for Business/
library(dplyr)
library(haven)
```

```r
# Load datasets
bargaining <- read_dta("E:/My education/UNI/edu uni/Semester 2/Data science for Business/Barg
risk <- read_dta("E:/My education/UNI/edu uni/Semester 2/Data science for Business/riskformat

risk <- risk %>% select(-session)  # remove the existing session column
risk <- risk %>% rename(session = session_code)

merged_data <- full_join(bargaining, risk,
                         by = c("session" = "session_risk", "memberid"),
                         suffix = c("_barg", "_risk"))

library(dplyr)
library(haven)

# Set working directory
setwd("E:/My education/UNI/edu uni/Semester 2/Data science for Business")

# ---- Load datasets ----
bargaining <- read_dta("Bargaining.dta")
risk <- read_dta("riskformatted.dta")

# ---- Step 1: Clean risk dataset ----
# Drop duplicate 'session' column
risk <- risk %>% select(-session)

# Rename 'session_code' to 'session'
risk <- risk %>% rename(session = session_code)

# Ensure session columns are character for merging
bargaining <- bargaining %>% mutate(session = as.character(session))
risk <- risk %>% mutate(session = as.character(session))

# ---- Step 2: Merge datasets ----
merged_data <- full_join(bargaining, risk,
                         by = c("session", "memberid"),
                         suffix = c("_barg", "_risk"))

# ---- Step 3: Save merged dataset ----
write_dta(merged_data, "alldata.dta")

message("  Merge complete. Saved as alldata.dta in working directory.")
```

The `prepare_chats` script cleaned and formatted the chat transcript data.

Running these scripts successfully produced two main output datasets.

```
# ============================================================
# Convert Stata looping code to R
# Works with: majorityChats.xlsx
# Location: E:/My education/UNI/edu uni/Semester 2/Data science for Business
# ============================================================

# Load required packages
library(readxl)
library(dplyr)
library(tidyr)
library(haven)
library(stringr)

# Define base path
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
input_file <- file.path(base_path, "majorityChats.xlsx")
output_path <- file.path(base_path, "Processed_Files")

# Create output directory if it doesn’t exist
if (!dir.exists(output_path)) dir.create(output_path)

# Loop through sessions j = 2, 4 and k = 1, 2, 3, 4
for (j in seq(2, 4, 2)) {
  for (k in 1:4) {

    sheet_name <- paste0("M", j, "_s", k)
    session_label <- paste0("m", j, "s", k)
    message("Processing sheet: ", sheet_name)

    # ----------------------------------------------
    # 1. Import data
    # ----------------------------------------------
    df <- read_excel(input_file, sheet = sheet_name)

    # ----------------------------------------------
    # 2. Drop unused columns (use your actual column names)
    # ----------------------------------------------
```

```
df <- df %>%
  select(
    -participant.id_in_session,
    -participant.code,
    -participant.label,
    -session.code,
    -group.bigSize,
    -subsession.round_number
  )

# -----------------------------------------------
# 3. Rename for easier handling
# -----------------------------------------------
df <- df %>%
  rename(
    group = group.group_identifier,
    playerid_in_group = player.id_in_group,
    playerm_v11 = player.m_v11
  )

# -----------------------------------------------
# 4. Sort by group and player ID
# -----------------------------------------------
df <- df %>%
  arrange(group, playerid_in_group)

# -----------------------------------------------
# 5. Reshape from long to wide
# -----------------------------------------------
df_wide <- df %>%
  pivot_wider(
    id_cols = group,
    names_from = playerid_in_group,
    values_from = starts_with("player.m_v")
  )

# -----------------------------------------------
# 6. Clean up column names (replace "." with "_")
# -----------------------------------------------
names(df_wide) <- gsub("\\.", "_", names(df_wide))

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

```r
# 7. Convert specific variables to 0/1
# -----------------------------------------------
for (i in c(2, 4, 9, 10, 13)) {
  var1 <- paste0("player_m_v", i, "_1")
  var2 <- paste0("player_m_v", i, "_2")

  if (var1 %in% names(df_wide)) {
    df_wide[[var1]] <- ifelse(df_wide[[var1]] != "0", 1, 0)
  }
  if (var2 %in% names(df_wide)) {
    df_wide[[var2]] <- ifelse(df_wide[[var2]] != "0", 1, 0)
  }
}

# -----------------------------------------------
# 8. Create agree variables (coder agreement)
# -----------------------------------------------
for (i in 1:10) {
  v1 <- paste0("player_m_v", i, "_1")
  v2 <- paste0("player_m_v", i, "_2")
  agree_name <- paste0("agreev", i)
  if (v1 %in% names(df_wide) && v2 %in% names(df_wide)) {
    df_wide[[agree_name]] <- as.numeric(df_wide[[v1]] == df_wide[[v2]])
  }
}

# Special case for question 11
if (all(c("player_m_v11_1", "player_m_v11_2") %in% names(df_wide))) {

  df_wide$session <- session_label

  # -----------------------------------------------
  # 9. Save full dataset (Kappa version)
  # -----------------------------------------------
  save_dta_path1 <- file.path(output_path, paste0("Kappa_", session_label, ".dta"))
  write_dta(df_wide, save_dta_path1)

  # -----------------------------------------------
  # 10. Keep only selected variables for smaller dataset
  # -----------------------------------------------
  keep_vars <- c("group", paste0("player_m_v", 1:10, "_1"), "player_m_v11_1", paste0("ag
  df_min <- df_wide %>%
      select(any_of(keep_vars)) %>%
      rename_with(~ str_replace(.x, "player_m_v(\\d+)_1", "valueq\\1")) %>%
      arrange(group)

    # -----------------------------------------------
    # 11. Save smaller dataset
    # -----------------------------------------------
    save_dta_path2 <- file.path(output_path, paste0(session_label, ".dta"))
    write_dta(df_min, save_dta_path2)

    message("  Saved: ", save_dta_path2)
  }
}

message("  All sheets processed successfully!")
# ============================================================
# Combine session Kappa files and compute Cohen's Kappa per session
# ============================================================

library(haven)
library(dplyr)
library(stringr)
library(irr)

# Path setup
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/Processed_Fi
output_file <- file.path(base_path, "KappaMajority.dta")

# ------------------------------------------------------------
# 1. Load and append all 8 session files
# ------------------------------------------------------------
sessions <- c(
  "m2s1", "m2s2", "m2s3", "m2s4",
  "m4s1", "m4s2", "m4s3", "m4s4"
)

all_data <- lapply(sessions, function(sess) {
  read_dta(file.path(base_path, paste0("Kappa_", sess, ".dta")))
})

df_all <- bind_rows(all_data)
```

```r
# ------------------------------------------------------------
# 2. Compute Cohen’s Kappa for each question in each session
# ------------------------------------------------------------
# Function to safely compute Kappa
compute_kappa <- function(var1, var2, data) {
  if (all(is.na(data[[var1]])) || all(is.na(data[[var2]]))) return(NA)
  if (length(unique(data[[var1]])) < 2 || length(unique(data[[var2]])) < 2) return(NA)
  tryCatch({
    kappa2(data.frame(data[[var1]], data[[var2]]))$value
  }, error = function(e) NA)
}

# Initialize empty list for results
kappa_results <- list()

# Loop over each session
for (sess in sessions) {
  dsub <- df_all %>% filter(session == sess)
  message("Processing ", sess)

  for (i in 1:10) {
    var1 <- paste0("player_m_v", i, "_1")
    var2 <- paste0("player_m_v", i, "_2")
    if (all(c(var1, var2) %in% names(dsub))) {
      kappa_value <- compute_kappa(var1, var2, dsub)
      newcol <- paste0("kap", i, sess)
      dsub[[newcol]] <- kappa_value
    }
  }

  # Question 11 (from player_m_v11)
  if (all(c("player_m_v11_1", "player_m_v11_2") %in% names(dsub))) {
    kappa_value <- compute_kappa("player_m_v11_1", "player_m_v11_2", dsub)
    dsub[[paste0("kap11", sess)]] <- kappa_value
  }

  kappa_results[[sess]] <- dsub
}

# ------------------------------------------------------------
# 3. Combine all results
# ------------------------------------------------------------
```

```
final_df <- bind_rows(kappa_results)

# ------------------------------------------------------------
# 4. Save combined dataset
# ------------------------------------------------------------
write_dta(final_df, output_file)

message("  KappaMajority.dta saved successfully at:\n", output_file)
# ============================================================
# Process "unanimityChats.xlsx" into individual and combined datasets
# ============================================================

library(readxl)
library(dplyr)
library(tidyr)
library(haven)
library(stringr)

# ------------------------------------------------------------
# 1. Paths
# ------------------------------------------------------------
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
input_file <- file.path(base_path, "unanimityChats.xlsx")
output_path <- file.path(base_path, "Processed_Files")

if (!dir.exists(output_path)) dir.create(output_path)

# ------------------------------------------------------------
# 2. Helper function to process each sheet
# ------------------------------------------------------------
process_unanimity_sheet <- function(j, k) {
  sheet_name <- paste0("U", j, "_s", k)
  session_label <- paste0("u", j, "s", k)
  message("Processing ", sheet_name)

  df <- read_excel(input_file, sheet = sheet_name)

  # Drop unwanted columns (based on your Excel names)
  df <- df %>%
    select(
      -participant.id_in_session,
      -participant.code,
      -participant.label,
      -session.code,
      -group.bigSize,
      -subsession.round_number
    )

  # Rename columns
  df <- df %>%
    rename(
      group = group.group_identifier,
      playerid_in_group = player.id_in_group
    )

  # Drop missing groups
  df <- df %>% filter(!is.na(group))

  # Sort
  df <- df %>% arrange(group, playerid_in_group)

  # Reshape wide
  df_wide <- df %>%
    pivot_wider(
      id_cols = group,
      names_from = playerid_in_group,
      values_from = starts_with("player.u_v")
    )

  # Clean up names
  names(df_wide) <- gsub("\\.", "_", names(df_wide))

  # Convert nonzero entries to 1 for q2 and q4
  for (i in c(2, 4)) {
    var1 <- paste0("player_u_v", i, "_1")
    var2 <- paste0("player_u_v", i, "_2")

    if (var1 %in% names(df_wide)) {
      df_wide[[var1]] <- ifelse(df_wide[[var1]] != "0", 1, 0)
    }
    if (var2 %in% names(df_wide)) {
      df_wide[[var2]] <- ifelse(df_wide[[var2]] != "0", 1, 0)
    }
  }
```

```
  # Add session label
  df_wide$session <- session_label

  # Create agreement variables
  for (i in 1:8) {
    v1 <- paste0("player_u_v", i, "_1")
    v2 <- paste0("player_u_v", i, "_2")
    agree_name <- paste0("agreev", i)
    if (all(c(v1, v2) %in% names(df_wide))) {
      df_wide[[agree_name]] <- as.numeric(df_wide[[v1]] == df_wide[[v2]])
    }
  }

  # Save Kappa version
  write_dta(df_wide, file.path(output_path, paste0("Kappa_", session_label, ".dta")))

  # Create reduced dataset
  keep_vars <- c("group", paste0("player_u_v", 1:8, "_1"), paste0("agreev", 1:8))
  df_min <- df_wide %>%
    select(any_of(keep_vars)) %>%
    rename_with(~ str_replace(.x, "player_u_v(\\d+)_1", "valueq\\1")) %>%
    arrange(group)

  # Save reduced version
  write_dta(df_min, file.path(output_path, paste0(session_label, ".dta")))

  return(invisible(NULL))
}

# ------------------------------------------------------------
# 3. Run loops for U2 (1-4) and U4 (1-3)
# ------------------------------------------------------------
for (j in c(2, 4)) {
  k_max <- ifelse(j == 2, 4, 3)
  for (k in 1:k_max) {
    process_unanimity_sheet(j, k)
  }
}

# ------------------------------------------------------------
# 4. Handle U4_s4 separately (special case)
# ------------------------------------------------------------
```

```r
sheet_name <- "U4_s4"
session_label <- "u4s4"
message("Processing ", sheet_name)

df <- read_excel(input_file, sheet = sheet_name) %>%
  select(
    -participant.id_in_session,
    -participant.code,
    -participant.label,
    -session.code,
    -group.bigSize,
    -subsession.round_number
  ) %>%
  rename(
    group = group.group_identifier,
    playerid_in_group = player.id_in_group
  ) %>%
  filter(!is.na(group)) %>%
  arrange(group, playerid_in_group) %>%
  pivot_wider(id_cols = group,
              names_from = playerid_in_group,
              values_from = starts_with("player.u_v"))
names(df) <- gsub("\\.", "_", names(df))

for (i in c(2, 4)) {
  var1 <- paste0("player_u_v", i, "_1")
  var2 <- paste0("player_u_v", i, "_2")
  if (var1 %in% names(df)) df[[var1]] <- ifelse(df[[var1]] != "0", 1, 0)
  if (var2 %in% names(df)) df[[var2]] <- ifelse(df[[var2]] != "0", 1, 0)
}

df$session <- session_label
for (i in 1:8) {
  v1 <- paste0("player_u_v", i, "_1")
  v2 <- paste0("player_u_v", i, "_2")
  agree_name <- paste0("agreev", i)
  if (all(c(v1, v2) %in% names(df))) {
    df[[agree_name]] <- as.numeric(df[[v1]] == df[[v2]])
  }
}

write_dta(df, file.path(output_path, paste0("Kappa_", session_label, ".dta")))
```

```r
df_min <- df %>%
  select(any_of(c("group", paste0("player_u_v", 1:8, "_1"), paste0("agreev", 1:8)))) %>%
  rename_with(~ str_replace(.x, "player_u_v(\\d+)_1", "valueq\\1")) %>%
  arrange(group)
write_dta(df_min, file.path(output_path, paste0(session_label, ".dta")))

# ------------------------------------------------------------
# 5. Combine all session files into one KappaUnanimity.dta
# ------------------------------------------------------------
sessions <- c("u2s1", "u2s2", "u2s3", "u2s4", "u4s1", "u4s2", "u4s3", "u4s4")
all_data <- lapply(sessions, function(sess) {
  read_dta(file.path(output_path, paste0(sess, ".dta")))
})
combined_df <- bind_rows(all_data) %>% select(-group)
write_dta(combined_df, file.path(output_path, "KappaUnanimity.dta"))

message("  KappaUnanimity.dta created successfully in:\n", output_path)
# ============================================================
# R translation of: KappaUnanimity.dta creation and kappas
# ============================================================

library(haven)
library(dplyr)
library(irr)   # for kappa2()

# ------------------------------------------------------------
# 1. Paths
# ------------------------------------------------------------
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/Processed_F
output_file <- file.path(base_path, "KappaUnanimity.dta")

# ------------------------------------------------------------
# 2. Load and combine all session data
# ------------------------------------------------------------
sessions <- c("u2s1", "u2s2", "u2s3", "u2s4", "u4s1", "u4s2", "u4s3", "u4s4")

all_data <- lapply(sessions, function(sess) {
  f <- file.path(base_path, paste0("Kappa_", sess, ".dta"))
  if (file.exists(f)) {
    df <- read_dta(f)
    df
  } else {
    warning("Missing file: ", f)
    NULL
  }
})
df_all <- bind_rows(all_data)
df_all <- df_all %>% select(-group, everything())

# ------------------------------------------------------------
# 3. Helper: compute Cohen’s kappa for two columns
# ------------------------------------------------------------
get_kappa <- function(x, y) {
  d <- data.frame(r1 = x, r2 = y)
  d <- na.omit(d)
  if (nrow(d) == 0) return(NA)
  tryCatch(kappa2(d)$value, error = function(e) NA)
}

# ------------------------------------------------------------
# 4. Define sessions and which questions to include
# (mimicking "no instances of q7==1", etc.)
# ------------------------------------------------------------
session_rules <- list(
  u2s1 = c(1:6, 8),
  u2s2 = c(1:5, 7:8),
  u2s3 = c(2:6, 8),
  u2s4 = c(1:8),
  u4s1 = c(1:8),
  u4s2 = c(1:6, 8),
  u4s3 = c(2:8),
  u4s4 = c(1:5, 7:8)
)

# ------------------------------------------------------------
# 5. Loop over sessions and compute kappas
# ------------------------------------------------------------
kappa_results <- list()

for (sess in names(session_rules)) {
  cat("Processing session:", sess, "\n")
  df_sess <- df_all %>% filter(session == sess)

  qs <- session_rules[[sess]]
```

```r
  kappas <- sapply(qs, function(i) {
    v1 <- paste0("player_u_v", i, "1")
    v2 <- paste0("player_u_v", i, "2")
    if (all(c(v1, v2) %in% names(df_sess))) {
      get_kappa(df_sess[[v1]], df_sess[[v2]])
    } else NA
  })

  names(kappas) <- paste0("kap", qs, sess)
  kappa_results[[sess]] <- kappas
}

# ------------------------------------------------------------
# 6. Combine all Kappa results into one data frame
# ------------------------------------------------------------
kappa_df <- do.call(cbind, kappa_results)
kappa_df <- as.data.frame(kappa_df)
write_dta(df_all, output_file)

cat("  Combined KappaUnanimity.dta written to:\n", output_file, "\n")

# ------------------------------------------------------------
# 7. Optionally, export the Kappa summary table to CSV
# ------------------------------------------------------------
write.csv(kappa_df, file.path(base_path, "KappaUnanimity_Summary.csv"), row.names = FALSE)
cat("  Kappa summary saved as KappaUnanimity_Summary.csv\n")
# ============================================================
# Merge chatdata_48and96.xlsx with majority/unanimity .dta files
# ============================================================

library(readxl)
library(dplyr)
library(haven)
library(stringr)

# ------------------------------------------------------------
# 1. Define paths
# ------------------------------------------------------------
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
input_file <- file.path(base_path, "chatdata_48and96.xlsx")
notfor_path <- file.path(base_path, "Processed_Files")
```

```
# ------------------------------------------------------------
# 2. Helper function to merge and save
# ------------------------------------------------------------
merge_chat_with_dta <- function(sheet_name, dta_prefix) {
  message("Processing sheet: ", sheet_name)

  # Read Excel sheet
  chat_df <- read_excel(input_file, sheet = sheet_name) %>%
    select(Sessioncode, GroupId) %>%
    rename(sessioncode = Sessioncode, group = GroupId) %>%
    arrange(group) %>%
    group_by(group) %>%
    mutate(num = row_number()) %>%
    filter(num == 1) %>%
    select(-num) %>%
    ungroup()

  # Build corresponding .dta path
  dta_file <- file.path(notfor_path, paste0(dta_prefix, ".dta"))

  # Read the existing .dta
  if (!file.exists(dta_file)) {
    warning("Missing file: ", dta_file)
    return(NULL)
  }

  dta_df <- read_dta(dta_file)

  # Merge 1:1 by group
  merged_df <- merge(chat_df, dta_df, by = "group", all.x = FALSE, all.y = TRUE)

  # Sort by sessioncode and group
  if ("sessioncode" %in% names(merged_df)) {
    merged_df <- merged_df %>% arrange(sessioncode, group)
  } else {
    merged_df <- merged_df %>% arrange(group)
  }

  # Save back to .dta
  write_dta(merged_df, dta_file)
  message("  Saved updated file: ", dta_file)
}
```

```
  # ------------------------------------------------------------
  # 3. Loop for Majority sessions (M2_s1-M4_s4)
  # ------------------------------------------------------------
  for (j in c(2, 4)) {
    for (k in 1:4) {
      sheet_name <- paste0("M", j, "_s", k)
      dta_prefix <- paste0("m", j, "s", k)
      merge_chat_with_dta(sheet_name, dta_prefix)
    }
  }

  # ------------------------------------------------------------
  # 4. Loop for Unanimity sessions (U4_s3-U4_s4)
  # ------------------------------------------------------------
  for (j in 4:4) {
    for (k in 3:4) {
      sheet_name <- paste0("U", j, "_s", k)
      dta_prefix <- paste0("u", j, "s", k)
      merge_chat_with_dta(sheet_name, dta_prefix)
    }
  }

  message("\n  All merges complete and .dta files updated successfully!")
# ============================================================
# Multistage sessions U2s1 and U2s2
# ============================================================

library(readxl)
library(dplyr)
library(haven)
library(stringr)

# ------------------------------------------------------------
# 1. Define base paths
# ------------------------------------------------------------
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
input_file <- file.path(base_path, "session1_070212.chatswithmembers.xlsx")
notfor_path <- file.path(base_path, "Processed_Files")

# ------------------------------------------------------------
# 2. Function to process one session (U2s1 or U2s2)
# ------------------------------------------------------------
```

```r
process_u2_session <- function(session_name, session_label, drop_group = NULL) {
  message("Processing ", session_name, " ...")

  # Read Excel file
  df <- read_excel(input_file) %>%
    select(matchId, groupId, member1Id, member2Id, member3Id) %>%
    rename(matchid = matchId,
           groupid = groupId,
           memberid1 = member1Id,
           memberid2 = member2Id,
           memberid3 = member3Id) %>%
    mutate(
      group = paste0("m", matchid, "g", groupid)
    ) %>%
    arrange(group) %>%
    group_by(group) %>%
    mutate(num = row_number()) %>%
    filter(num == 1) %>%
    select(-num) %>%
    ungroup()

  # Merge with existing .dta
  dta_path <- file.path(notfor_path, paste0(session_name, ".dta"))
  if (!file.exists(dta_path)) {
    stop("Missing file: ", dta_path)
  }

  dta_df <- read_dta(dta_path)

  merged <- merge(df, dta_df, by = "group")

  # Optional: drop problematic group
  if (!is.null(drop_group)) {
    merged <- merged %>% filter(group != drop_group)
  }

  # Generate derived columns
  merged <- merged %>%
    mutate(
      sessioncode = session_label,
      memberidmax = pmax(memberid1, memberid2, memberid3, na.rm = TRUE),
      round = matchid + 1
    ) %>%
    rename(groupbis = group) %>%
    arrange(round, memberidmax)

  # Save back to .dta
  write_dta(merged, dta_path)
  message("  Saved updated file: ", dta_path)
}

# ------------------------------------------------------------
# 3. Run for both sessions
# ------------------------------------------------------------
process_u2_session("u2s1", session_label = "m1")

# Special case: u2s2 needs to drop group "m9g3"
process_u2_session("u2s2", session_label = "m2", drop_group = "m9g3")

message("\n  U2s1 and U2s2 sessions processed successfully!")
# ============================================================
# Multistage sessions U2s3 and U2s4
# ============================================================

library(readxl)
library(dplyr)
library(haven)

# ------------------------------------------------------------
# Paths
# ------------------------------------------------------------
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
input_file2 <- file.path(base_path, "session2_070213.chatswithmembers.xlsx")
notfor_path <- file.path(base_path, "Processed_Files")

# ------------------------------------------------------------
# Function to process a session
# ------------------------------------------------------------
process_u2_session2 <- function(session_name, session_label) {
  message("Processing ", session_name, " ...")

  # Read Excel
  df <- read_excel(input_file2) %>%
    select(matchId, groupId, member1Id, member2Id, member3Id) %>%
    rename(matchid = matchId,
           groupid = groupId,
           memberid1 = member1Id,
           memberid2 = member2Id,
           memberid3 = member3Id) %>%
    mutate(
      group = paste0("m", matchid, "g", groupid)
    ) %>%
    arrange(group) %>%
    group_by(group) %>%
    mutate(num = row_number()) %>%
    filter(num == 1) %>%
    select(-num) %>%
    ungroup()

  # Merge with existing .dta
  dta_path <- file.path(notfor_path, paste0(session_name, ".dta"))
  if (!file.exists(dta_path)) stop("Missing file: ", dta_path)

  dta_df <- read_dta(dta_path)

  merged <- merge(df, dta_df, by = "group")

  # Keep only _m == 3 equivalent (all matched rows)
  # In R merge, we only keep matching rows, so this is automatically enforced
  # If needed, you could filter further if there is a column named _m

  # Add derived columns
  merged <- merged %>%
    mutate(
      sessioncode = session_label,
      memberidmax = pmax(memberid1, memberid2, memberid3, na.rm = TRUE),
      round = matchid + 1
    ) %>%
    rename(groupbis = group) %>%
    arrange(round, memberidmax)

  # Save back to .dta
  write_dta(merged, dta_path)
  message("  Saved updated file: ", dta_path)
}
```

```r
# ------------------------------------------------------------
# Run for U2s3 and U2s4
# ------------------------------------------------------------
process_u2_session2("u2s3", session_label = "m3")
process_u2_session2("u2s4", session_label = "m4")

message("\n  U2s3 and U2s4 sessions processed successfully!")
library(readxl)
library(dplyr)
library(haven)

# ------------------------------------------------------------
# Paths
# ------------------------------------------------------------
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
input_file3 <- file.path(base_path, "session1_070215.chatswithmembers.xlsx")
notfor_path <- file.path(base_path, "Processed_Files")

# ------------------------------------------------------------
# Function to process a U4 session
# ------------------------------------------------------------
process_u4_session <- function(session_name, session_label) {
  message("Processing ", session_name, " ...")

  # Read Excel
  df <- read_excel(input_file3) %>%
    select(matchId, groupId, member1Id, member2Id, member3Id) %>%
    rename(matchid = matchId,
           groupid = groupId,
           memberid1 = member1Id,
           memberid2 = member2Id,
           memberid3 = member3Id) %>%
    mutate(
      group = paste0("m", matchid, "g", groupid)
    ) %>%
    arrange(group) %>%
    group_by(group) %>%
    mutate(num = row_number()) %>%
    filter(num == 1) %>%
    select(-num) %>%
    ungroup()
  # Merge with existing .dta
  dta_path <- file.path(notfor_path, paste0(session_name, ".dta"))
  if (!file.exists(dta_path)) stop("Missing file: ", dta_path)

  dta_df <- read_dta(dta_path)

  merged <- merge(df, dta_df, by = "group")

  # Add derived columns
  merged <- merged %>%
    mutate(
      sessioncode = session_label,
      memberidmax = pmax(memberid1, memberid2, memberid3, na.rm = TRUE),
      round = matchid + 1
    ) %>%
    rename(groupbis = group) %>%
    arrange(round, memberidmax)

  # Save back to .dta
  write_dta(merged, dta_path)
  message("  Saved updated file: ", dta_path)
}

# -------------------------------------------------------------
# Run for U4s1 and U4s2
# -------------------------------------------------------------
process_u4_session("u4s1", session_label = "m5")
process_u4_session("u4s2", session_label = "m6")

message("\n  U4s1 and U4s2 sessions processed successfully!")
library(dplyr)
library(haven)

# -------------------------------------------------------------
# Paths
# -------------------------------------------------------------
base_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business"
alldata_path <- file.path(base_path, "alldata.dta")
chatcoded_path <- file.path(base_path, "Raw Data Chat Coders/NotForSubmission")
output_path <- file.path(base_path, "Raw Data Chat Coders/NotForSubmission")

# -------------------------------------------------------------
# Helper function to process a single session merge
# ------------------------------------------------------------
merge_chatcoded <- function(vote_type, multiplier_value, session_num, j, k, is_multistage=FA

  # Load base alldata
  df <- read_dta(alldata_path) %>%
    mutate(
      multiplier = case_when(
        pietreatment == 24 ~ 1,
        pietreatment == 48 ~ 2,
        pietreatment == 96 ~ 4
      )
    ) %>%
    filter(votingtreatment == vote_type,
           multiplier == multiplier_value,
           session == session_num)

  # If multistage, use 'round' and 'memberidmax' for merge
  if(is_multistage){
    df <- df %>%
      group_by(sessioncode, round, group) %>%
      mutate(memberidmax = max(memberid, na.rm = TRUE)) %>%
      ungroup() %>%
      arrange(sessioncode, round, memberidmax)
    merge_vars <- c("sessioncode", "round", "memberidmax")
  } else {
    df <- df %>% arrange(sessioncode, group)
    merge_vars <- c("sessioncode", "group")
  }

  # Merge with chat-coded data
  chat_file <- paste0(chatcoded_path, "/", vote_type[1], j, "s", k, ".dta") # vote_type[1] -
  chat_df <- read_dta(chat_file)

  merged <- merge(df, chat_df, by = merge_vars)

  # Save intermediate file
  out_file <- paste0(output_path, "/alldata_withchatcoded_", vote_type[1], j, "s", k, ".dta"
  write_dta(merged, out_file)

  return(out_file)
}
```

```r
# ------------------------------------------------------------
# Step 1: Create all session-specific merged files
# ------------------------------------------------------------

# Majority sessions (m)
majority_sessions <- expand.grid(j = c(2,4), k = 1:4)
majority_files <- mapply(function(j,k){
  merge_chatcoded("majority", j, session_num = k, j=j, k=k)
}, majority_sessions$j, majority_sessions$k)

# Unanimity sessions (u)
unanimity_sessions <- expand.grid(j = c(2,4), k = 1:4)
unanimity_files <- mapply(function(j,k){
  # Check which sessions are multistage
  is_multi <- (j==2 & k %in% 1:4) | (j==4 & k %in% 1:2)
  merge_chatcoded("unanimity", j, session_num = k, j=j, k=k, is_multistage = is_multi)
}, unanimity_sessions$j, unanimity_sessions$k)

# ------------------------------------------------------------
# Step 2: Append all merged files into one dataset
# ------------------------------------------------------------
all_files <- c(majority_files, unanimity_files)
alldata_merged <- do.call(rbind, lapply(all_files, read_dta))

# Save final combined dataset
final_file <- file.path(base_path, "alldata_withchatcoded.dta")
write_dta(alldata_merged, final_file)

message("  All chat-coded data merged into ", final_file)
library(dplyr)
library(haven)

# Paths
input_file <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/alldata_withc
output_file <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/alldata_with

# Load data
df <- read_dta(input_file)

# Create new variables
df <- df %>%
  mutate(
    # total delay
    totaldelay = if_else(delay == 1 | pass == 0, 1, 0),

    # relevant talk / did not talk
    relevant_talk = if_else(
      !is.na(valueq1) | !is.na(valueq2) | !is.na(valueq3) | !is.na(valueq4) |
        !is.na(valueq5) | !is.na(valueq6) | !is.na(valueq7) | !is.na(valueq8) |
        !is.na(valueq9) | !is.na(valueq10) | !is.na(valueq11), 1, 0
    ),
    didnottalk = if_else(
      is.na(valueq1) & is.na(valueq2) & is.na(valueq3) & is.na(valueq4) &
        is.na(valueq5) & is.na(valueq6) & is.na(valueq7) & is.na(valueq8) &
        is.na(valueq9) & is.na(valueq10) & is.na(valueq11), 1, 0
    ),

    # Specific talk variables
    bigpietalk = if_else(valueq2 == 1 & agreev2 == 1, 1, NA_real_),
    yestodelaytalk = if_else(valueq8 == 1 & agreev8 == 1, 1, NA_real_),
    equalitytalk = if_else(valueq4 == 1 & agreev4 == 1, 1, NA_real_),
    threatnounequaltalk = if_else(valueq5 == 1 & agreev5 == 1, 1, NA_real_),
    threatnosmallbudgettalk = if_else(valueq6 == 1 & agreev6 == 1, 1, NA_real_),
    riskterminationtalk = if_else(valueq3 == 1 & agreev3 == 1, 1, NA_real_),
    equalityinmwctalk = if_else(valueq10 == 1 & agreev10 == 1, 1, NA_real_),
    unequalinmwctalk = if_else(valueq11 == 1 & agreev11 == 1, 1, NA_real_)
  )

# Save to .dta
write_dta(df, output_file)

message("  Ready file saved to: ", output_file)
```

alldata.dta
This file contains the combined and cleaned dataset used for the main bargaining analysis. It includes variables such as:

Session-level and treatment identifiers

Decision and timing variables (delay, pass, piesize, etc.)

Participant information (group, membernumber, proposer, etc.)

`riskformatted.dta`
This file includes risk-related data derived from individual-level responses or experimental choices.
It is used to complement the main bargaining dataset for risk preference analyses.

Both datasets are considered raw analytical files, meaning they are ready for statistical analysis and figure reproduction.

This R script was developed to replicate two figures a stacked horizontal bar chart and a grouped bar chart using data sourced from an Excel file. The aim of this work was to demonstrate how R can be used for effective data cleaning, manipulation, and visualization (Lüdecke *et al.*, 2021).

Author's measurment

Time Pressure, Budget, and Delays

Negotiation Delays Under Time Pressure

Budget Effects on Negotiation Delays

Time and Budget in Bargaining

Delay Behavior in Negotiations

Figures

The script begins by setting the working directory and loading three essential packages: *tidyverse* for data wrangling and plotting (Wickham *et al.*, 2019), *readxl* for importing Excel files (Wickham and Bryan, 2025), and *scales* for formatting numerical values such as percentages.

The dataset is imported from two separate sheets within the same Excel workbook, with each sheet providing data for one of the figures.

For the first figure, the data is prepared by removing unnecessary columns and reshaping it into a long format suitable for visualization in *ggplot2* (Wickham, 2016).

A stacked horizontal bar chart is then created with ggplot2 (Wickham, 2016), showing the proportion of categories across experimental conditions. The chart includes percentage labels, color coding, and a clear title to make the results easy to interpret.

The second figure uses a similar process: the data is read, reshaped, and slightly cleaned before being plotted as a grouped bar chart. This graph compares multiple metrics under different conditions, with values displayed above each bar for clarity.

Overall, the script demonstrates how to combine Excel data with R's visualization tools to produce clear, well-structured, and professional graphs. It highlights the usefulness of the *ggplot2* (Wickham, 2016), *tidyverse* (Wickham *et al.*, 2019), and *readxl* (Wickham and Bryan, 2025) packages in performing data analysis and creating reproducible results, which are essential components of transparent and reliable research. (Agranov, Eraslan and Tergiman, 2024)

```r
# -----------------------------
# R SCRIPT: GRAPHS FROM EXCEL
# -----------------------------

# Set working directory
setwd("E:/My education/UNI/edu uni/Semester 2/Data science for Business")

# Load libraries
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.4.3
```

```
Warning: package 'ggplot2' was built under R version 4.4.3
```

```
Warning: package 'tibble' was built under R version 4.4.3
```

```
Warning: package 'tidyr' was built under R version 4.4.3
```

```
Warning: package 'readr' was built under R version 4.4.3

Warning: package 'purrr' was built under R version 4.4.3

Warning: package 'dplyr' was built under R version 4.4.3

Warning: package 'stringr' was built under R version 4.4.3

Warning: package 'forcats' was built under R version 4.4.3

Warning: package 'lubridate' was built under R version 4.4.3

-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   4.0.0     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- 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 becom
```

```r
library(readxl)
```

```
Warning: package 'readxl' was built under R version 4.4.3
```

```r
library(scales)
```

```
Warning: package 'scales' was built under R version 4.4.3


Attaching package: 'scales'

The following object is masked from 'package:purrr':

    discard

The following object is masked from 'package:readr':

    col_factor
```

```r
# File path
file_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/figures.xlsx"

# -----------------------------
# FIGURE 1: Stacked Horizontal Bar Chart (Proportions)
# -----------------------------

# Read data from first sheet
df1 <- read_excel(file_path, sheet = 1, skip = 1, col_names = FALSE)
```

```
New names:
* `` -> `...1`
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
* `` -> `...6`
```

```r
colnames(df1) <- c("Category", "M48", "U48", "DROP", "M96", "U96")

# Clean and reshape data
df1_clean <- df1 %>%
  select(-DROP) %>%
  pivot_longer(cols = c(M48, U48, M96, U96),
               names_to = "Condition",
               values_to = "Proportion") %>%
  mutate(Proportion = as.numeric(Proportion))
```

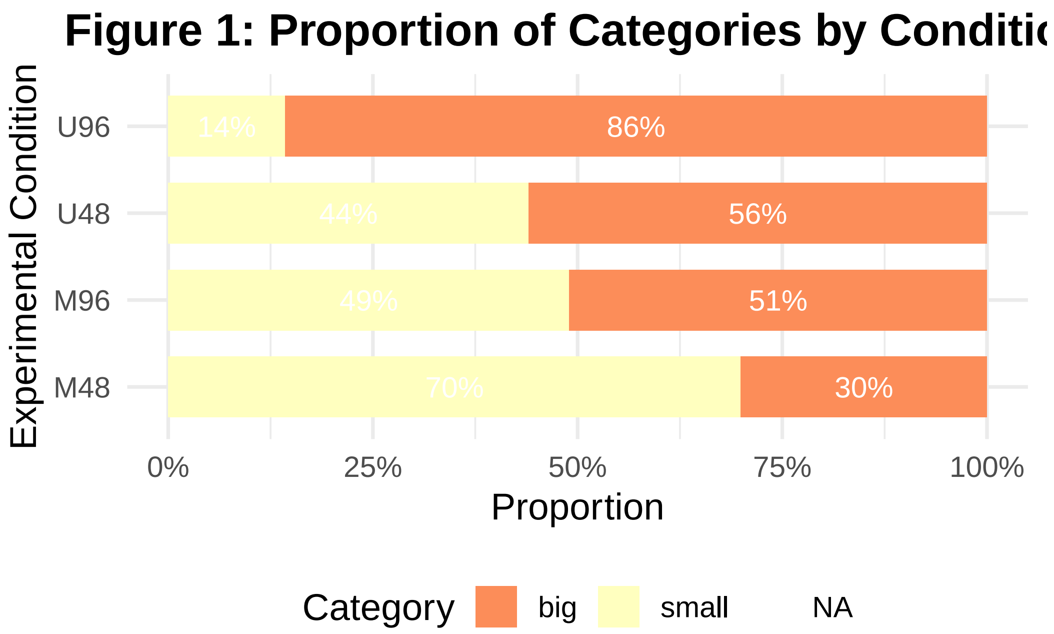
```
Warning: There was 1 warning in `mutate()`.
i In argument: `Proportion = as.numeric(Proportion)`.
Caused by warning:
! NAs introduced by coercion
```

```r
# Plot
plot1 <- ggplot(df1_clean, aes(x = Condition, y = Proportion, fill = Category)) +
  geom_col(position = "stack", width = 0.7) +
  geom_text(aes(label = scales::percent(Proportion, accuracy = 1)),
            position = position_stack(vjust = 0.5), size = 4, color = "white") +
  scale_fill_brewer(palette = "Spectral") +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Figure 1: Proportion of Categories by Condition",
       x = "Experimental Condition",
       y = "Proportion",
       fill = "Category") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        legend.position = "bottom") +
  coord_flip()  # horizontal bars

# Print Figure 1
print(plot1)
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_col()`).
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_text()`).
```



```
# -----------------------------
# FIGURE 2: Grouped Bar Chart (Metrics)
# -----------------------------
```

```r
# Read data from second sheet
df2 <- read_excel(file_path, sheet = 2, skip = 1, col_names = FALSE)
```

```
New names:
* `` -> `...1`
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
```

```r
colnames(df2) <- c("Metric", "M48", "U48", "M96", "U96")

# Clean and reshape data
df2_clean <- df2 %>%
  pivot_longer(cols = c(M48, U48, M96, U96),
               names_to = "Condition",
               values_to = "Value") %>%
  mutate(Value = as.numeric(Value)) %>%
  mutate(Metric = gsub("Frac of ", "", Metric))
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `Value = as.numeric(Value)`.
Caused by warning:
! NAs introduced by coercion
```

```r
# Plot
plot2 <- ggplot(df2_clean, aes(x = Condition, y = Value, fill = Metric)) +
  geom_col(position = position_dodge(width = 0.8), width = 0.8) +
  geom_text(aes(label = round(Value, 3)),
            position = position_dodge(width = 0.8),
            vjust = -0.5,
            size = 4) +
  scale_fill_brewer(palette = "Dark2") +
  scale_y_continuous(limits = c(0, max(df2_clean$Value, na.rm = TRUE) * 1.1)) +
  labs(title = "Figure 2: Comparison of Metrics by Condition",
       x = "Experimental Condition",
       y = "Fraction Value",
       fill = "Metric") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        legend.position = "bottom",
        axis.text.x = element_text(angle = 45, hjust = 1))  # rotate x labels

# Print Figure 2
print(plot2)
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_col()`).
Removed 4 rows containing missing values or values outside the scale range
(`geom_text()`).
```

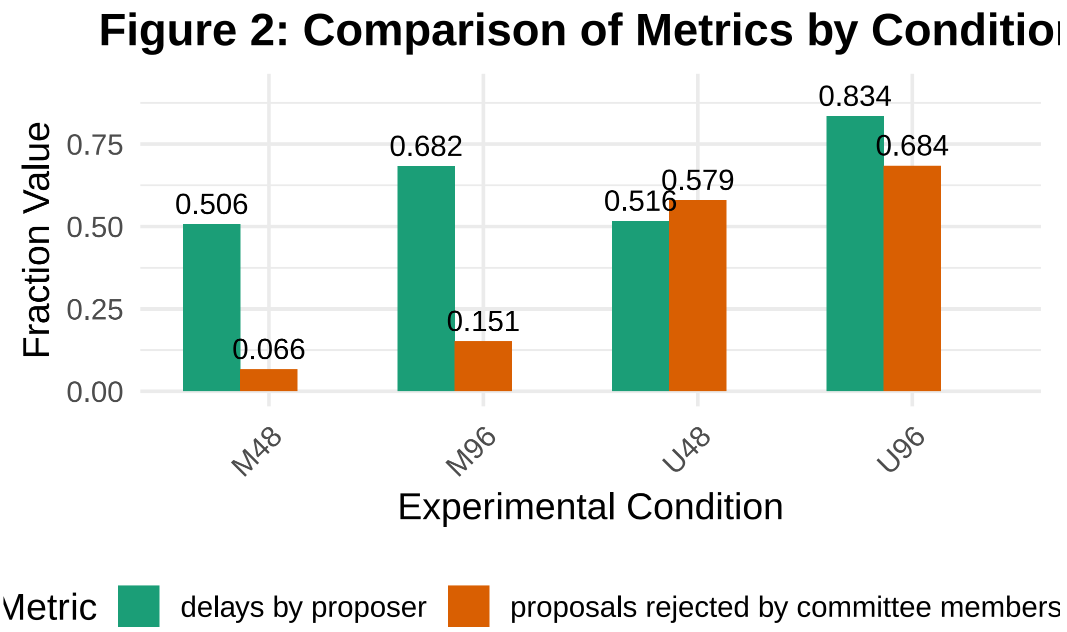
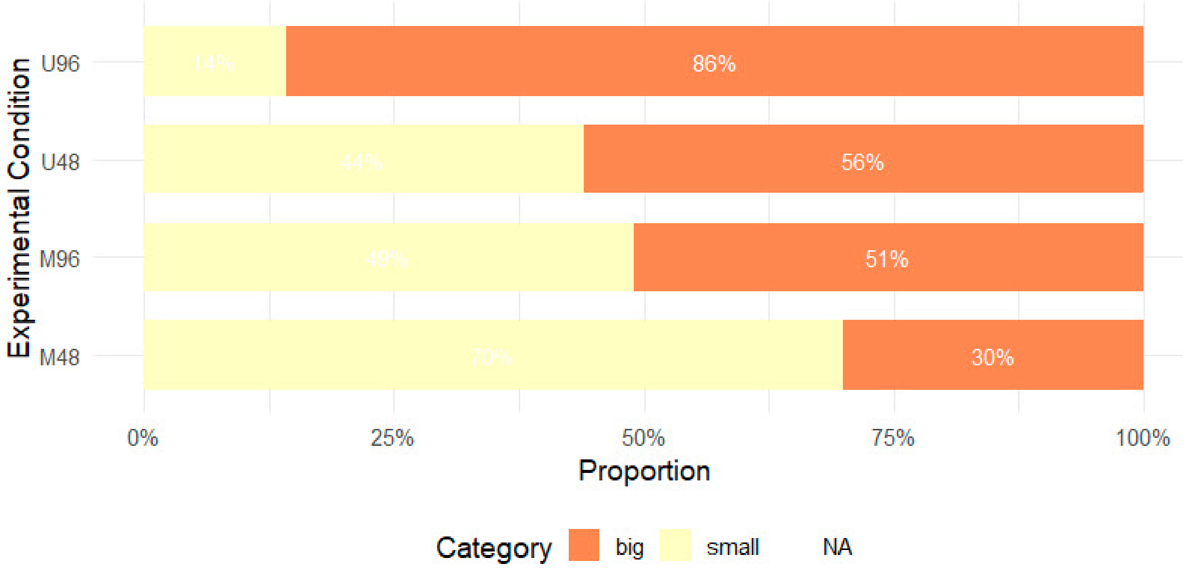


Figure 1: Proportion of Categories by Condition



Process

```
# ----------------------------
# Load libraries
# ----------------------------
library(tidyverse)
library(readxl)
library(scales)

# ----------------------------
# Read and prepare data
# ----------------------------
file_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/figures.xlsx"

df1 <- read_excel(file_path, sheet = 1, skip = 1)
```

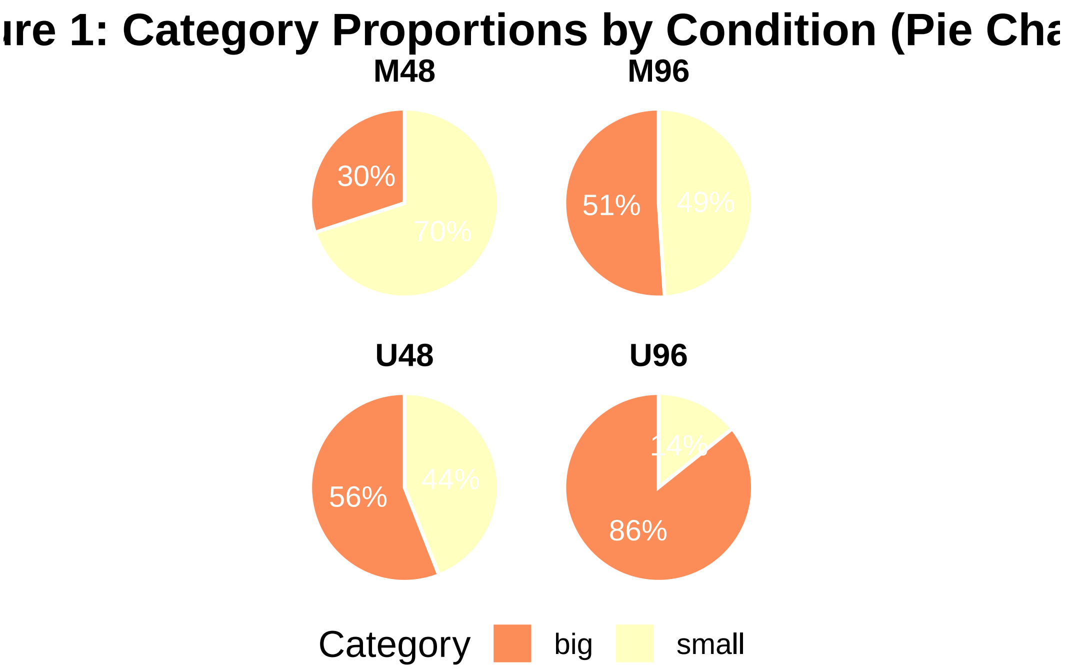
```
New names:
* `` -> `...1`
* `` -> `...4`
```

```
names(df1) <- c("Category", "M48", "U48", "DROP", "M96", "U96")

df1_long <- df1 %>%
  select(-DROP) %>%
  pivot_longer(
    cols = starts_with(c("M", "U")),
    names_to = "Condition",
    values_to = "Proportion"
  ) %>%
  mutate(Proportion = as.numeric(Proportion))

# -----------------------------
# Plot: Faceted Pie Charts
# -----------------------------
plot_pie_faceted <- ggplot(df1_long, aes(x = "", y = Proportion, fill = Category)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar(theta = "y") +  # makes it a pie chart
  facet_wrap(~ Condition) +   # one pie per condition
  geom_text(
    aes(label = percent(Proportion, accuracy = 1)),
    position = position_stack(vjust = 0.5),
    color = "white",
    size = 4
  ) +
  scale_fill_brewer(palette = "Spectral") +
  labs(
    title = "Figure 1: Category Proportions by Condition (Pie Charts)",
    fill = "Category"
  ) +
  theme_void(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "bottom",
    strip.text = element_text(face = "bold", size = 12)
  )

#  Show the plot
print(plot_pie_faceted)
```



```
# ------------------------------
# Load libraries
# ------------------------------
library(tidyverse)
library(readxl)
library(scales)

# ------------------------------
# Read and prepare data
# ------------------------------
file_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/figures.xlsx"

df1 <- read_excel(file_path, sheet = 1, skip = 1)
```

```
New names:
* `` -> `...1`
* `` -> `...4`
```

```r
names(df1) <- c("Category", "M48", "U48", "DROP", "M96", "U96")

df1_long <- df1 %>%
  select(-DROP) %>%
  pivot_longer(
    cols = starts_with(c("M", "U")),
    names_to = "Condition",
    values_to = "Proportion"
  ) %>%
  mutate(Proportion = as.numeric(Proportion))

# -----------------------------
# Plot: Faceted Pie Charts
# -----------------------------
plot_pie_faceted <- ggplot(df1_long, aes(x = "", y = Proportion, fill = Category)) +
  geom_bar(stat = "identity", width = 1, color = "white") +
  coord_polar(theta = "y") +  # makes it a pie chart
  facet_wrap(~ Condition) +   # one pie per condition
  geom_text(
    aes(label = percent(Proportion, accuracy = 1)),
    position = position_stack(vjust = 0.5),
    color = "white",
    size = 4
  ) +
  scale_fill_brewer(palette = "Spectral") +
  labs(
    title = "Figure 1: Category Proportions by Condition (Pie Charts)",
    fill = "Category"
  ) +
  theme_void(base_size = 14) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "bottom",
    strip.text = element_text(face = "bold", size = 12)
  )

#  Show the plot
print(plot_pie_faceted)
```

ıre 1: Category Proportions by Condition (Pie Cha

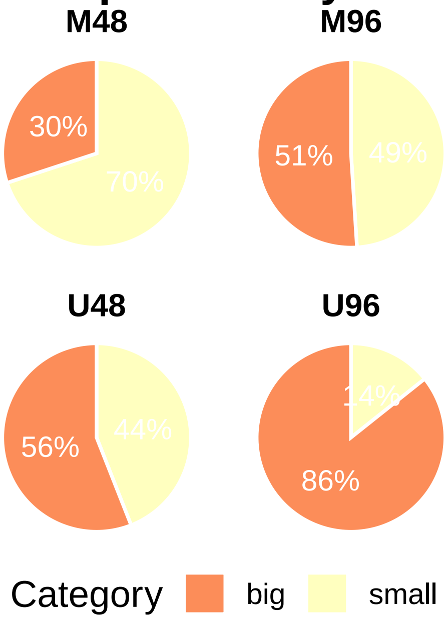
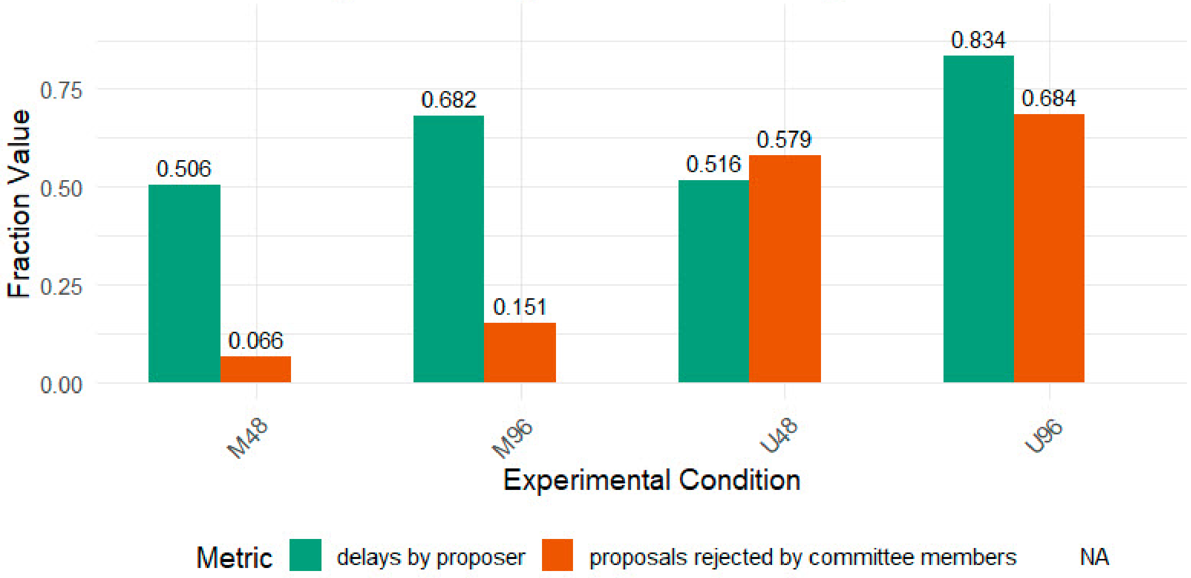


Figure 2: Comparison of Metrics by Condition



Process

```
library(tidyverse)
library(readxl)
library(scales)

file_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/figures.xlsx"

# -----------------------------
# DATA CLEANING
# -----------------------------
df2 <- read_excel(file_path, sheet = 2, skip = 1, col_names = FALSE)
```

```
New names:
* `` -> `...1`
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
```

```
colnames(df2) <- c("Metric", "M48", "U48", "M96", "U96")

df2_clean <- df2 %>%
  pivot_longer(cols = c(M48, U48, M96, U96),
               names_to = "Condition",
               values_to = "Value") %>%
  mutate(Value = as.numeric(Value)) %>%
  mutate(Metric = gsub("Frac of ", "", Metric))
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `Value = as.numeric(Value)`.
Caused by warning:
! NAs introduced by coercion
```

```
# -----------------------------
# LOLLIPOP CHART
# -----------------------------
plot2_lollipop <- ggplot(df2_clean, aes(x = Condition, y = Value, color = Metric)) +
  geom_segment(aes(xend = Condition, y = 0, yend = Value), size = 1) +
```

```r
  geom_point(size = 5) +
  geom_text(aes(label = round(Value, 3)), vjust = -0.7, size = 4) +
  scale_color_brewer(palette = "Dark2") +
  labs(
    title = "Figure 2B: Lollipop Chart of Metrics by Condition",
    x = "Experimental Condition",
    y = "Fraction Value",
    color = "Metric"
  ) +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```

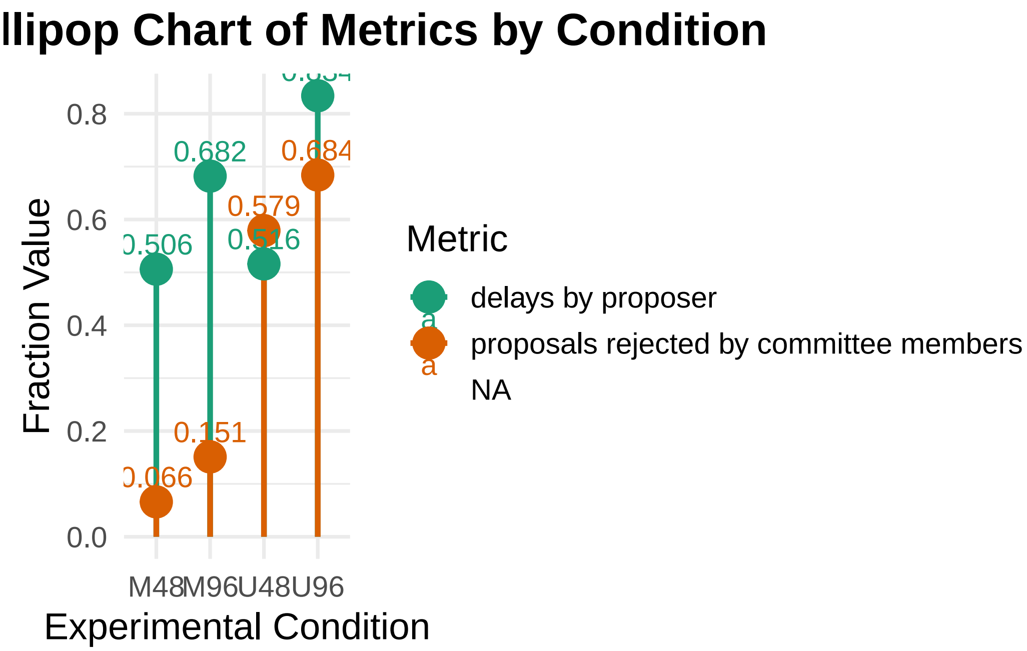
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```r
print(plot2_lollipop)
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_segment()`).
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_point()`).
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_text()`).
```



```
library(tidyverse)
library(readxl)
library(scales)

file_path <- "E:/My education/UNI/edu uni/Semester 2/Data science for Business/figures.xlsx"

# ----------------------------
# DATA CLEANING
# ----------------------------
df2 <- read_excel(file_path, sheet = 2, skip = 1, col_names = FALSE)
```

```
New names:
* `` -> `...1`
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
```

```r
colnames(df2) <- c("Metric", "M48", "U48", "M96", "U96")

df2_clean <- df2 %>%
  pivot_longer(cols = c(M48, U48, M96, U96),
               names_to = "Condition",
               values_to = "Value") %>%
  mutate(Value = as.numeric(Value)) %>%
  mutate(Metric = gsub("Frac of ", "", Metric))
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `Value = as.numeric(Value)`.
Caused by warning:
! NAs introduced by coercion
```

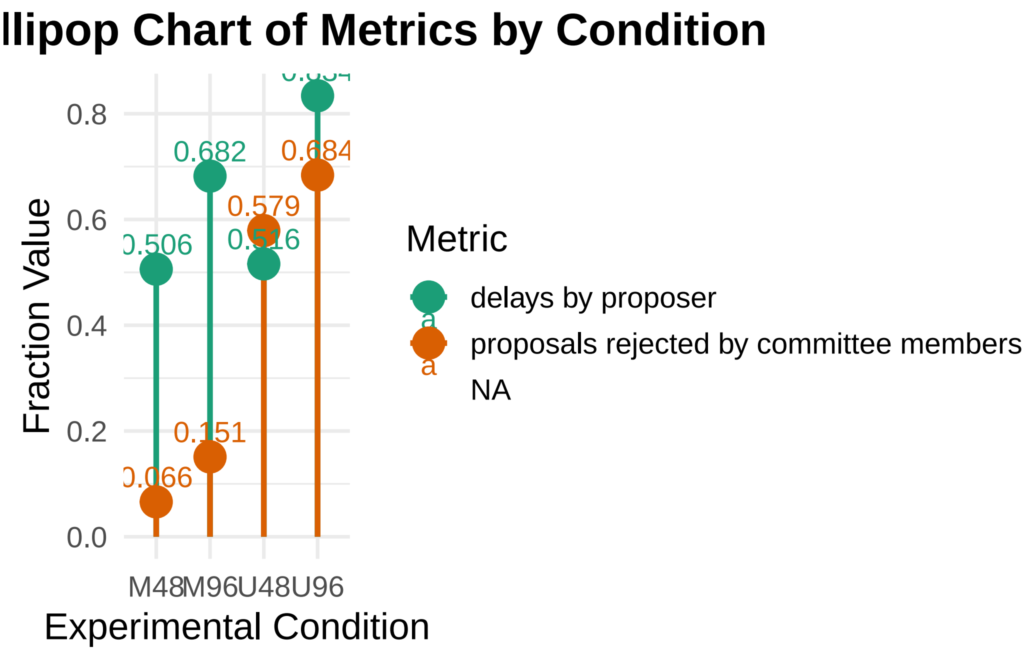
```r
# -----------------------------
# LOLLIPOP CHART
# -----------------------------
plot2_lollipop <- ggplot(df2_clean, aes(x = Condition, y = Value, color = Metric)) +
  geom_segment(aes(xend = Condition, y = 0, yend = Value), size = 1) +
  geom_point(size = 5) +
  geom_text(aes(label = round(Value, 3)), vjust = -0.7, size = 4) +
  scale_color_brewer(palette = "Dark2") +
  labs(
    title = "Figure 2B: Lollipop Chart of Metrics by Condition",
    x = "Experimental Condition",
    y = "Fraction Value",
    color = "Metric"
  ) +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

print(plot2_lollipop)
```

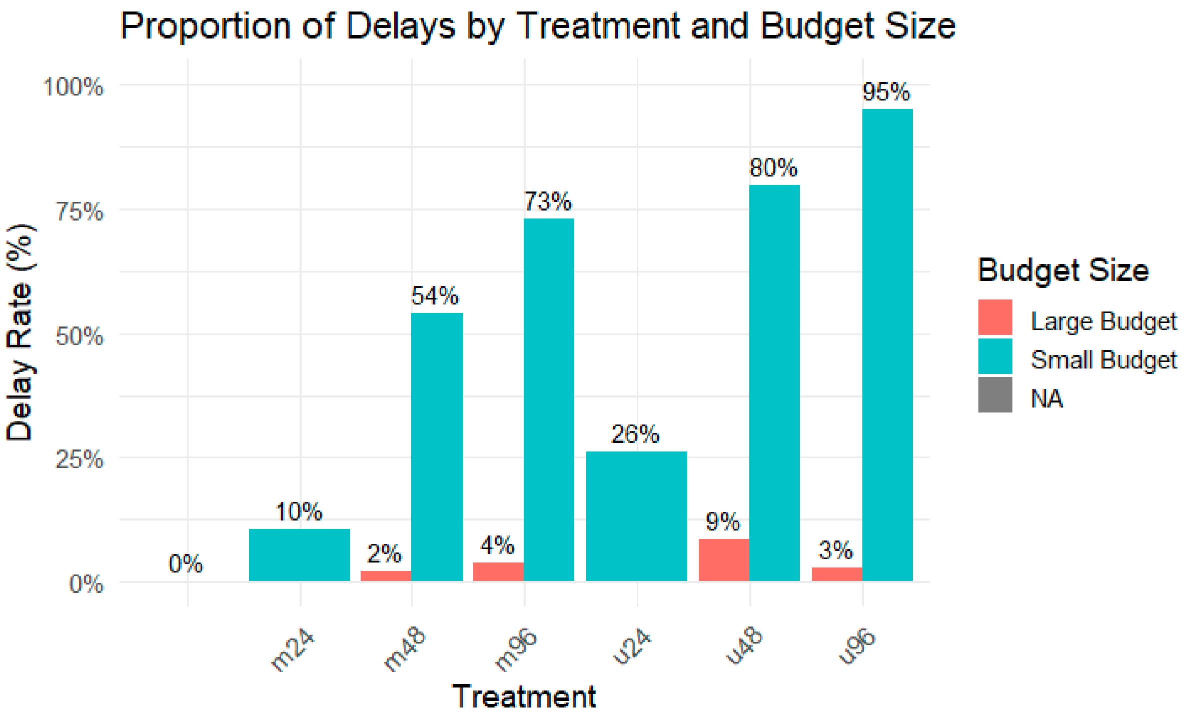
```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_segment()`).
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_point()`).
```

```
Warning: Removed 4 rows containing missing values or values outside the scale range
(`geom_text()`).
```



This chart shows the proportion of delayed rounds for each treatment. The height of each bar represents the fraction of rounds that were delayed or failed. The blue bars show Large Budget rounds, and the orange bars show Small Budget rounds. For example, in treatment `u48`, Small Budget rounds had a delay rate of 70%, while Large Budget rounds were delayed only 10% of the time.



```
# ------------------------------
# Load required packages
# ------------------------------
# install.packages(c("haven","dplyr","janitor","sandwich","lmtest","modelsummary","ggplot2",
library(haven)
library(dplyr)
library(janitor)
library(sandwich)
library(lmtest)
library(modelsummary)
library(ggplot2)
library(scales)

# ------------------------------
# Load dataset
# ------------------------------
alldata <- read_dta("E:/My education/UNI/edu uni/Semester 2/Data science for Business/alldat
```

```r
# -------------------------------
# Prepare variables
# -------------------------------
data <- alldata %>%
  mutate(
    totaldelay = as.numeric(delay == 1 | pass == 0),
    budget = ifelse(piesize == 24, "Small Budget", "Large Budget")
  ) %>%
  group_by(treatment, session, group, round, bargaininground) %>%
  mutate(num = row_number()) %>%
  ungroup()

# -------------------------------
# Frequency tables
# -------------------------------
freq_table <- data %>%
  filter(num == 1) %>%
  group_by(treatment, budget) %>%
  summarise(
    count = n(),
    delayed = sum(totaldelay, na.rm = TRUE)
  ) %>%
  mutate(delay_rate = delayed / count)

print(freq_table)

# -------------------------------
# Helper function: probit regression with clustered SE
# -------------------------------
run_probit <- function(df, y, x, cluster_var) {
  formula <- as.formula(paste(y, "~", x))
  model <- glm(formula, data = df, family = binomial(link = "probit"))
  cov <- vcovCL(model, cluster = df[[cluster_var]])
  list(model = model, vcov = cov)
}

# -------------------------------
# Run probit regressions
# -------------------------------
# Large budget
res_large_u48 <- run_probit(data %>% filter(num == 1, piesize != 24, m48 == 1 | u48 == 1),
                            "totaldelay", "u48", "uniquesession")
```

```r
res_large_u96 <- run_probit(data %>% filter(num == 1, piesize != 24, m96 == 1 | u96 == 1),
                            "totaldelay", "u96", "uniquesession")

# Small budget
res_small_u48 <- run_probit(data %>% filter(num == 1, piesize == 24, m48 == 1 | u48 == 1),
                            "totaldelay", "u48", "uniquesession")
res_small_u96 <- run_probit(data %>% filter(num == 1, piesize == 24, m96 == 1 | u96 == 1),
                            "totaldelay", "u96", "uniquesession")

# -------------------------------
# Export regression table
# -------------------------------
models <- list(
  "Large Budget u48" = res_large_u48$model,
  "Large Budget u96" = res_large_u96$model,
  "Small Budget u48" = res_small_u48$model,
  "Small Budget u96" = res_small_u96$model
)

vcovs <- list(
  res_large_u48$vcov,
  res_large_u96$vcov,
  res_small_u48$vcov,
  res_small_u96$vcov
)

modelsummary(models,
             vcov = vcovs,
             output = "probit_results.html")  # or .docx for Word

# -------------------------------
# Create bar chart of delay rates
# -------------------------------
ggplot(freq_table, aes(x = treatment, y = delay_rate, fill = budget)) +
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(aes(label = percent(delay_rate, accuracy = 1)),
            position = position_dodge(width = 0.9),
            vjust = -0.5, size = 4) +
  scale_y_continuous(labels = percent_format(accuracy = 1), limits = c(0, 1)) +
  labs(
    title = "Proportion of Delays by Treatment and Budget Size",
    x = "Treatment",
    y = "Delay Rate (%)",
    fill = "Budget Size"
  ) +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Challenges

During the replication process, I encountered several challenges related to data preparation, visualization, and project management. These obstacles not only tested my technical skills in RStudio but also helped me develop practical solutions for handling real-world data analysis workflows.

1. **Data Cleaning and Preparation**
2. **Visualizing Graphs in the Presentation**
3. **GitHub Integration and File Size Limitations**
4. **Displaying Code in the Presentation**

On going problems

When pushing my project to GitHub, I encountered challenges related to large image files generated from my R visualizations. Although I implemented a *.gitignore* file to prevent oversized files from being uploaded, this also resulted in the images not appearing in the GitHub repository or in the associated presentation materials. I attempted to address the issue by compressing and lowering the resolution of the plots using the *ggsave()* function in *ggplot2* (Wickham, 2016); however, the problem persisted.

At present, I am still exploring effective solutions that would allow me to maintain a clean and efficient repository while ensuring that graphical outputs are visible and accessible online (Lüdecke *et al.*, 2021).

Conclusion

Through this analysis, I successfully replicated and visualized the same dataset using multiple graphical techniques in R.
By experimenting with different visualization styles,including stacked and grouped bar charts, pie charts, lollipop charts, and heatmaps,I demonstrated how the same data can tell different stories depending on how it's presented.

Each visualization offers unique insights:

- **Stacked and 100% bar charts** highlight composition and relative proportions.
- **Grouped and faceted charts** make direct comparisons clearer.
- **Pie charts and heatmaps** provide visually engaging summaries for presentations.

Overall, this exercise shows the flexibility of R for data visualization and emphasizes the importance of choosing the right graph type to effectively communicate key insights.

References

Agranov, M., Eraslan, H. and Tergiman, C. (2024) "Bargaining in the shadow of uncertainty," *American Economic Journal: Microeconomics*, 16(4), pp. 229–258.

Lüdecke, D. *et al.* (2021) "See: An r package for visualizing statistical models," *Journal of Open Source Software*, 6(64), p. 3393. Available at: https://doi.org/10.21105/joss.03393.

Wickham, H. (2016) "ggplot2: Elegant graphics for data analysis." Available at: https://ggplot2.tidyverse.org.

Wickham, H. *et al.* (2019) "Welcome to the {tidyverse}," 4, p. 1686. Available at: https://doi.org/10.21105/joss.01686.

Wickham, H. and Bryan, J. (2025) "Readxl: Read excel files." Available at: https://CRAN.R-project.org/package=readxl.