

# Predoc Data Task Analysis

## Contents

<b>1</b>	<b>1. Load Data</b>	<b>1</b>
<b>2</b>	<b>2. Data Pipeline</b>	<b>2</b>
2.1	2.1 Recoding and Missing Data Approach (Overview) . . . . .	2
2.2	2.2 Eligibility Filters (Applied First) . . . . .	2
2.3	2.3 Data Quality Checks (Review) . . . . .	2
2.4	2.4 Extreme Value Review (Ages) . . . . .	6
2.5	2.5 Apply Extreme Value Decision . . . . .	7
2.6	2.6 Recode Outcomes and Controls . . . . .	7
2.7	2.7 Missingness After Recoding (Post-Filters) . . . . .	9
2.8	2.8 Final Sample Construction and Flow Table . . . . .	9
<b>3</b>	<b>3. Descriptive Statistics</b>	<b>10</b>
<b>4</b>	<b>4. Main Analysis</b>	<b>12</b>
<b>5</b>	<b>5. Robustness Checks</b>	<b>16</b>
<b>6</b>	<b>6. Exploratory: Blame - Sex Interaction</b>	<b>19</b>

## 1 1. Load Data

```
# Load the main data sheet (raw data is treated as read-only).
data_raw <- readxl::read_excel(input_path, sheet = "Data")

# Explicitly require the variables that the plan allows us to analyze.
required_vars <- c("ResponseId", "Finished", "passedattn", "consent", "IPAddress",
  "sex", "age", "real_imaginary", "blame_1", "outcome_binary1", "outcome_binary2")

# Fail loudly if any required variable is missing.
missing_vars <- setdiff(required_vars, names(data_raw))
if (length(missing_vars) > 0) {
  stop(paste("Missing required variables:", paste(missing_vars, collapse = ", ")))
}
```

## 2 2. Data Pipeline

### 2.1 2.1 Recoding and Missing Data Approach (Overview)

- This pipeline keeps `data_raw` read-only after import. There is no imputation of missing values.
- Missing values would be handled by listwise deletion, if any existed.
- In this dataset, post-recode missingness for core variables is 0.
  - Consequentially, no explicit missing-value exclusions are applied in the pipeline.

### 2.2 2.2 Eligibility Filters (Applied First)

```
# Exclusion flags follow the plan exactly. 1) Finished must be TRUE.
exclude_not_finished <- !(data_raw$Finished %in% TRUE)
# 2) Attention check must be 'yes'.
exclude_failed_attn <- !(data_raw$passedattn %in% "yes")
# 3) Consent must be AGREE (no explicit 'do not use' flag exists).
exclude_nonconsent <- (!is.na(data_raw$consent)) & (data_raw$consent != "AGREE")

keep_eligible <- !(exclude_not_finished | exclude_failed_attn | exclude_nonconsent)
data_eligible <- data_raw[keep_eligible, ]

cat("Eligible rows after filters (Finished, attention, consent):", nrow(data_eligible),
    "\n")
```

```
## Eligible rows after filters (Finished, attention, consent): 46
```

### 2.3 2.3 Data Quality Checks (Review)

```
# Confirm each respondent appears once (ResponseId should be unique).
if (anyDuplicated(data_eligible$ResponseId) > 0) {
  dup_ids <- data_eligible$ResponseId[duplicated(data_eligible$ResponseId)]
  stop(paste("Duplicate ResponseId values found:", paste(unique(dup_ids), collapse = ", ")))
}

# Additional respondent uniqueness check using IP address (trimmed).
ip_norm <- trimws(as.character(data_eligible$IPAddress))
if (any(is.na(ip_norm) | ip_norm == "")) {
  stop("Missing IPAddress values found; cannot check uniqueness by IP address.")
}
if (anyDuplicated(ip_norm) > 0) {
  dup_ips <- unique(ip_norm[duplicated(ip_norm)])
  dup_counts <- table(ip_norm)[dup_ips]
  cat("Duplicate IPAddress values found (continuing):\n")
  cat("Total distinct duplicate IPs:", length(dup_ips), "\n")
  for (ip in dup_ips) {
    cat("  -", ip, "count:", as.integer(dup_counts[[ip]]), "\n")
  }
}
```

```

# Print requested fields for all rows with duplicate IPs.
extra_vars <- c("IPAddress", "StartDate", "EndDate", "Duration (in seconds)",
  "Q26_Operating System", "describe")
missing_extra_vars <- setdiff(extra_vars, names(data_eligible))
if (length(missing_extra_vars) > 0) {
  stop(paste("Missing required fields for duplicate IP review:", paste(missing_extra_vars,
    collapse = ", ")))
}
dup_rows <- data_eligible[ip_norm %in% dup_ips, extra_vars]

# Table for timestamps, duration, and OS (plus IP for clarity).
table_vars <- c("IPAddress", "StartDate", "EndDate", "Duration (in seconds)",
  "Q26_Operating System")
dup_table <- dup_rows[, table_vars]
cat("Duplicate IP rows (timestamps, duration, OS):\n")
print(dup_table)

# List of describe fields for each duplicate row.
cat("Describe field entries for duplicate IP rows:\n")
for (i in seq_len(nrow(dup_rows))) {
  desc_text <- as.character(dup_rows$describe[i])
  desc_text <- ifelse(is.na(desc_text), "[NA]", desc_text)
  wrapped_desc <- strwrap(desc_text, width = 80)
  cat(" - IP:", dup_rows$IPAddress[i], "Describe:\n")
  for (line in wrapped_desc) {
    cat("    ", line, "\n")
  }
}
}

```

```

## Duplicate IPAddress values found (continuing):
## Total distinct duplicate IPs: 1
## - 50.237.5.2 count: 4
## Duplicate IP rows (timestamps, duration, OS):
## # A tibble: 4 x 5
##   IPAddress StartDate      EndDate      'Duration (in seconds)'
##   <chr>      <dtm>          <dtm>          <dbl>
## 1 50.237.5.2 2021-01-27 12:12:53 2021-01-27 12:16:56      243
## 2 50.237.5.2 2021-01-27 12:12:45 2021-01-27 12:17:46      300
## 3 50.237.5.2 2021-01-27 12:12:51 2021-01-27 12:18:01      310
## 4 50.237.5.2 2021-01-27 12:12:42 2021-01-27 12:18:35      352
## # i 1 more variable: 'Q26_Operating System' <chr>
## Describe field entries for duplicate IP rows:
## - IP: 50.237.5.2 Describe:
##   This colleague felt that I was not delivering a creative/out of the box enough
##   solution and I felt that I was providing as much as possible given the
##   constraints I was working with. There was no extreme argument but I could tell
##   that he was not pleased with my explanation.
## - IP: 50.237.5.2 Describe:
##   Work conflict -- he was my direct supervisor and always made unreasonable asks
##   of the team and treated people poorly (e.g. yelling at them). There was one
##   incident in particular where I had not thought critically enough about a
##   workstream that I was on, and he blew up at me.

```

```
## - IP: 50.237.5.2 Describe:
##   I was managing someone who was older and more educated than me (they were in
##   their 40s and had a PhD, and I was in my 20's and did not). They were being
##   generally very hostile in the workplace (towards many people), and they were
##   letting small frustrations lead to strong, negative emotional reactions. While
##   their negative emotional state was partly due to external factors, they were
##   also frustrated by what they saw was me not being qualified to be managing them
##   or telling them what to do.
## - IP: 50.237.5.2 Describe:
##   My colleague and I disagreed on the degrees of prioritization in implementing a
##   project.
```

```
# After analyzing the duplicates, I've decided to keep them. The reasoning
# will be in the report appendix.
```

```
# Inventory variables used (plan requires an explicit list).
```

```
used_vars <- required_vars
```

```
# Helper: numeric summaries for ranges/missingness.
```

```
summarize_numeric <- function(x) {
  x_num <- as.numeric(x)
  if (all(is.na(x_num))) {
    return(list(n = length(x_num), missing = sum(is.na(x_num)), mean = NA, sd = NA,
               min = NA, max = NA))
  }
  list(n = length(x_num), missing = sum(is.na(x_num)), mean = mean(x_num, na.rm = TRUE),
       sd = sd(x_num, na.rm = TRUE), min = min(x_num, na.rm = TRUE), max = max(x_num,
                                       na.rm = TRUE))
}
```

```
# Helper: categorical level inventory for observed values.
```

```
summarize_categorical <- function(x) {
  tab <- table(x, useNA = "ifany")
  as.data.frame(tab, stringsAsFactors = FALSE)
}
```

```
# Check for unexpected non-numeric entries in numeric fields.
```

```
count_non_numeric <- function(x) {
  x_num <- suppressWarnings(as.numeric(x))
  sum(!is.na(x) & is.na(x_num))
}
```

```
non_numeric_age <- count_non_numeric(data_eligible$age)
non_numeric_blame <- count_non_numeric(data_eligible$blame_1)
```

```
cat("Non-numeric entries (age):", non_numeric_age, "\n")
```

```
## Non-numeric entries (age): 0
```

```
cat("Non-numeric entries (blame_1):", non_numeric_blame, "\n")
```

```
## Non-numeric entries (blame_1): 0
```

```
# Categorical variables to list levels for checks.
categorical_vars <- c("Finished", "passedattn", "consent", "sex", "real_imaginary",
  "outcome_binary1", "outcome_binary2")
```

```
# Numeric summaries for checks.
age_stats <- summarize_numeric(data_eligible$age)
blame_stats <- summarize_numeric(data_eligible$blame_1)
```

```
# Print key structure and quality checks in the knitted output.
cat("Data checks: total eligible rows:", nrow(data_eligible), "\n")
```

```
## Data checks: total eligible rows: 46
```

```
cat("Data checks: unique ResponseId rows:", nrow(data_eligible), "\n")
```

```
## Data checks: unique ResponseId rows: 46
```

```
cat("Data checks: variables used:", paste(used_vars, collapse = ", "), "\n")
```

```
## Data checks: variables used: ResponseId, Finished, passedattn, consent, IPAddress, sex, age, real_im
```

```
cat("Data checks: age missing/mean/sd/min/max:", age_stats$missing, round(age_stats$mean,
  2), round(age_stats$sd, 2), age_stats$min, age_stats$max, "\n")
```

```
## Data checks: age missing/mean/sd/min/max: 0 33.63 19.95 18 149
```

```
cat("Data checks: blame_1 missing/mean/sd/min/max:", blame_stats$missing, round(blame_stats$mean,
  2), round(blame_stats$sd, 2), blame_stats$min, blame_stats$max, "\n")
```

```
## Data checks: blame_1 missing/mean/sd/min/max: 0 37.93 16.95 0 75
```

```
cat("Data checks: observed levels for categorical variables:\n")
```

```
## Data checks: observed levels for categorical variables:
```

```
for (v in categorical_vars) {
  tab <- summarize_categorical(data_eligible[[v]])
  levels_text <- paste(tab$x, tab$Freq, sep = ": ", collapse = "; ")
  cat(" -", v, "->", levels_text, "\n")
}
```

```
## - Finished -> TRUE: 46
```

```
## - passedattn -> yes: 46
```

```
## - consent -> AGREE: 46
```

```
## - sex -> Female: 26; Male: 20
```

```
## - real_imaginary -> I am imagining myself in a fictional situation where this is true (I can't think
```

```
## - outcome_binary1 -> I apologize first, then ${e://Field/initials} apologizes.: 36; Neither I nor ${e://Field/initials}
```

```
## - outcome_binary2 -> I apologize first, but ${e://Field/initials} does not apologize after that.: 20
```

## 2.4 2.4 Extreme Value Review (Ages)

```
# We review extreme ages because Section 2.3 showed a max age of 149. Identify
# extreme ages for review (above 100 only).
extreme_mask <- !is.na(data_eligible$age) & (data_eligible$age > 100)
extreme_count <- sum(extreme_mask)

cat("Number of extreme ages (>100):", extreme_count, "\n")
```

```
## Number of extreme ages (>100): 1
```

```
# Show the full rows for extreme ages so decisions can be made explicitly.
t(data_eligible[extreme_mask, ])
```

```
##                               [,1]
## StartDate                    "2021-01-27 12:12:43"
## EndDate                      "2021-01-27 12:16:46"
## Status                       "IP Address"
## IPAddress                    "72.83.222.239"
## Progress                     "100"
## Duration (in seconds)        "242"
## Finished                     "TRUE"
## RecordedDate                 "2021-01-27 12:16:48"
## ResponseId                   "R_bBJ75JWs3W8KSgF"
## RecipientLastName            NA
## RecipientFirstName           NA
## RecipientEmail               NA
## ExternalReference            NA
## LocationLatitude              "38.9489"
## LocationLongitude            "-76.9352"
## DistributionChannel           "anonymous"
## UserLanguage                 "EN"
## consent                      "AGREE"
## Q26_Browser                  "Chrome"
## Q26_Version                  "88.0.4324.104"
## Q26_Operating System         "Windows NT 10.0"
## Q26_Resolution               "1920x1200"
## QID54_First Click            "29.019"
## QID54_Last Click             "29.019"
## QID54_Page Submit            "42.399"
## QID54_Click Count            "1"
## real_imaginary               "I have thought of a real argument/conflict from my life where this is true"
## initials_box                 "TR"
## describe                     "I'm not comfortable writing this."
## feelings_youalone            "-30"
## feelings_bothyoufirst        "8"
## feelings_themalone           "1"
## feelings_boththemfirst       "20"
## feelings_neither             "-20"
## feelings_youaloneforgiven    "-30"
## feelings_D0_1                "2"
## feelings_D0_2                "1"
```

```
## feelings_D0_3          "3"
## feelings_D0_4          "5"
## feelings_D0_5          "4"
## feelings_D0_6          "6"
## feelings_exp           NA
## outcome_binary1        "I apologize first, then ${e://Field/initials} apologizes."
## outcome_binary1_D0_1    "2"
## outcome_binary1_D0_2    "1"
## outcome_binary2        "Neither I nor ${e://Field/initials} apologizes."
## outcome_binary2_D0_1    "1"
## outcome_binary2_D0_2    "2"
## blame_1                "31"
## attention_1            NA
## attention_1_TEXT        NA
## attention_2            NA
## attention_2_TEXT        NA
## attention_3            NA
## attention_3_TEXT        "Banana"
## target_sex             "Other"
## target_sex_3_TEXT       NA
## sex                    "Male"
## sex_3_TEXT             NA
## age                    "149"
## comments                "don't use these data; was just clicking through."
## mainControl             "showConsentOn showAttnOn"
## passedattn             "yes"
## lottery_draw            "37"
## winner                  "0"
## initials                "TR"
```

## 2.5 2.5 Apply Extreme Value Decision

```
# Rule: keep ages <= 100; drop ages > 100 because the flagged row includes a
# 'do not use' note.
data_no_extreme <- data_eligible[!extreme_mask, ]
cat("Rows dropped for age > 100:", nrow(data_eligible) - nrow(data_no_extreme), "\n")
```

```
## Rows dropped for age > 100: 1
```

## 2.6 2.6 Recode Outcomes and Controls

```
# Decision: sex is the primary explanatory variable. Only 'Female' and 'Male'
# are valid.
allowed_sex <- c("Female", "Male")
if (any(!is.na(data_no_extreme$sex) & !(data_no_extreme$sex %in% allowed_sex))) {
  # bad_vals captures unexpected sex entries; we stop so they are not
  # silently dropped or miscoded.
  bad_vals <- unique(data_no_extreme$sex[!is.na(data_no_extreme$sex) & !(data_no_extreme$sex %in%
    allowed_sex)])
  stop(paste("Unexpected values in sex:", paste(bad_vals, collapse = ", ")))
}
```

```

}

data_recoded <- data_no_extreme

# Set Female as the reference category for interpretation.
data_recoded$sex_factor <- factor(data_recoded$sex, levels = c("Female", "Male"))

# Decision: real_imaginary is used only as a control / robustness dimension.
real_label <- "I have thought of a real argument/conflict from my life where this is true"
imag_label <- "I am imagining myself in a fictional situation where this is true (I can't think of one)"

allowed_real_imag <- c(real_label, imag_label)
if (any(!is.na(data_recoded$real_imaginary) & !(data_recoded$real_imaginary %in%
allowed_real_imag))) {
  # bad_vals captures unexpected framing entries; we stop rather than coerce
  # to avoid misclassification.
  bad_vals <- unique(data_recoded$real_imaginary[!is.na(data_recoded$real_imaginary) &
!(data_recoded$real_imaginary %in% allowed_real_imag)])
  stop(paste("Unexpected values in real_imaginary:", paste(bad_vals, collapse = ", ")))
}

# Map to concise labels for modeling.
data_recoded$real_imaginary_factor <- factor(ifelse(data_recoded$real_imaginary ==
real_label, "real", ifelse(data_recoded$real_imaginary == imag_label, "imagined",
NA)), levels = c("real", "imagined"))

# Decision: outcome coding (1 = 'I apologize first...', 0 = 'Neither
# apologizes').
ob1_yes <- "I apologize first, then ${e://Field/initials} apologizes."
ob1_no <- "Neither I nor ${e://Field/initials} apologizes."

ob2_yes <- "I apologize first, but ${e://Field/initials} does not apologize after that."
ob2_no <- "Neither I nor ${e://Field/initials} apologizes."

if (any(!is.na(data_recoded$outcome_binary1) & !(data_recoded$outcome_binary1 %in%
c(ob1_yes, ob1_no)))) {
  # bad_vals captures outcome responses outside the two allowed strings; we
  # stop to prevent miscoding.
  bad_vals <- unique(data_recoded$outcome_binary1[!is.na(data_recoded$outcome_binary1) &
!(data_recoded$outcome_binary1 %in% c(ob1_yes, ob1_no))])
  stop(paste("Unexpected values in outcome_binary1:", paste(bad_vals, collapse = ", ")))
}
if (any(!is.na(data_recoded$outcome_binary2) & !(data_recoded$outcome_binary2 %in%
c(ob2_yes, ob2_no)))) {
  # bad_vals captures outcome responses outside the two allowed strings; we
  # stop to prevent miscoding.
  bad_vals <- unique(data_recoded$outcome_binary2[!is.na(data_recoded$outcome_binary2) &
!(data_recoded$outcome_binary2 %in% c(ob2_yes, ob2_no))])
  stop(paste("Unexpected values in outcome_binary2:", paste(bad_vals, collapse = ", ")))
}

# Create numeric outcomes for analysis and plotting.
data_recoded$outcome1 <- ifelse(data_recoded$outcome_binary1 == ob1_yes, 1, ifelse(data_recoded$outcome

```



Table 1: Missingness after filters, extreme-age drop, and recoding

variable	missing_count
sex	0
age	0
real_imaginary	0
blame_1	0
outcome_binary1	0
outcome_binary2	0
outcome1	0
outcome2	0

```
ob1_no, 0, NA))

data_recoded$outcome2 <- ifelse(data_recoded$outcome_binary2 == ob2_yes, 1, ifelse(data_recoded$outcome
ob2_no, 0, NA))
```

## 2.7 2.7 Missingness After Recoding (Post-Filters)

```
# Missingness for analysis variables after eligibility filters and extreme-age
# drop.
vars_for_missing <- c("sex", "age", "real_imaginary", "blame_1", "outcome_binary1",
  "outcome_binary2", "outcome1", "outcome2")

missing_post_recode <- sapply(data_recoded[vars_for_missing], function(x) sum(is.na(x)))
missing_table <- data.frame(variable = names(missing_post_recode), missing_count = as.integer(missing_p
  stringsAsFactors = FALSE)

if (knitr::is_latex_output()) {
  knitr::kable(missing_table, format = "latex", caption = "Missingness after filters, extreme-age drop
} else {
  knitr::kable(missing_table, format = "html", caption = "Missingness after filters, extreme-age drop
}
```

```
float_barrier()
```

```
## \clearpage
```

## 2.8 2.8 Final Sample Construction and Flow Table

```
analysis_main <- data_recoded

# Only keep needed columns for analysis and output.
analysis_vars <- c("ResponseId", "sex_factor", "age", "real_imaginary_factor", "blame_1",
  "outcome1", "outcome2")
analysis_main <- analysis_main[, analysis_vars]
```

Table 2: Sample flow counts

step	n
raw	61
after_eligibility_filters	46
after_extreme_age_drop	45

```
# Save cleaned analysis dataset to /output (raw data remains untouched).
clean_path <- file.path(output_dir, "cleaned_analysis_data.csv")
write.csv(analysis_main, clean_path, row.names = FALSE)

sample_flow <- data.frame(step = c("raw", "after_eligibility_filters", "after_extreme_age_drop"),
  n = c(nrow(data_raw), nrow(data_eligible), nrow(data_no_extreme)), stringsAsFactors = FALSE)

if (knitr::is_latex_output()) {
  knitr::kable(sample_flow, format = "latex", caption = "Sample flow counts")
} else {
  knitr::kable(sample_flow, format = "html", caption = "Sample flow counts")
}
```

```
float_barrier()
```

```
## \clearpage
```

### 3. Descriptive Statistics

```
# Descriptive table: mean, median, SD by sex group (overall, female, male).
# Includes age, outcomes, real_imaginary (coded 1=real, 0=imagined), and
# blame_1.
if (!is.numeric(analysis_main$blame_1)) {
  stop("blame_1 must be numeric to compute mean/median/SD. Check the raw data coding.")
}

analysis_main$real_binary <- ifelse(analysis_main$real_imaginary_factor == "real",
  1, ifelse(analysis_main$real_imaginary_factor == "imagined", 0, NA))

stats_by_group <- function(x, group_name) {
  data.frame(group = group_name, mean = mean(x, na.rm = TRUE), median = median(x,
    na.rm = TRUE), sd = sd(x, na.rm = TRUE))
}

build_desc_table <- function(var_name, x, data) {
  rbind(cbind(variable = var_name, stats_by_group(x, "Overall")), cbind(variable = var_name,
    stats_by_group(x[data$sex_factor == "Female"], "Female")), cbind(variable = var_name,
    stats_by_group(x[data$sex_factor == "Male"], "Male")))
}

desc_table <- rbind(build_desc_table("age", analysis_main$age, analysis_main), build_desc_table("outcome", analysis_main$outcome, analysis_main))
```

Table 3: Descriptive statistics by sex (mean, median, SD)

variable	group	mean	median	sd
age	Overall	31.067	29	9.882
age	Female	31.192	29	8.271
age	Male	30.895	29	11.986
outcome1	Overall	0.778	1	0.420
outcome1	Female	0.731	1	0.452
outcome1	Male	0.842	1	0.375
outcome2	Overall	0.467	0	0.505
outcome2	Female	0.462	0	0.508
outcome2	Male	0.474	0	0.513
real_imaginary_real	Overall	0.911	1	0.288
real_imaginary_real	Female	0.885	1	0.326
real_imaginary_real	Male	0.947	1	0.229
blame_1	Overall	38.089	40	17.112
blame_1	Female	34.308	34	13.356
blame_1	Male	43.263	46	20.472

```

analysis_main$outcome1, analysis_main), build_desc_table("outcome2", analysis_main$outcome2,
analysis_main), build_desc_table("real_imaginary_real", analysis_main$real_binary,
analysis_main), build_desc_table("blame_1", analysis_main$blame_1, analysis_main))

if (knitr::is_latex_output()) {
  knitr::kable(desc_table, digits = 3, format = "latex", caption = "Descriptive statistics by sex (mean, median, SD)")
} else {
  knitr::kable(desc_table, digits = 3, format = "html", caption = "Descriptive statistics by sex (mean, median, SD)")
}

```

```
float_barrier()
```

```
## \clearpage
```

```

# Print simple sex counts
sex_counts <- table(analysis_main$sex_factor, useNA = "ifany")

cat(sprintf("Sex counts: Female: %d, Male: %d\n", sex_counts["Female"], sex_counts["Male"]))

## Sex counts: Female: 26, Male: 19

# Print simple real/imaginary counts
ri_counts <- table(analysis_main$real_imaginary_factor, useNA = "ifany")

cat(sprintf("Conflict counts: Real: %d, Imagined: %d\n", ri_counts["real"], ri_counts["imagined"]))

## Conflict counts: Real: 41, Imagined: 4

```

Table 4: Difference-in-means (Male - Female)

outcome	mean_male	mean_female	diff_male_minus_female	ci_lower	ci_upper	p_value
outcome_binary1	0.8421	0.7308	0.1113	-0.1379	0.3606	0.3725
outcome_binary2	0.4737	0.4615	0.0121	-0.2999	0.3242	0.9376

## 4 4. Main Analysis

```
# Difference-in-means (Male minus Female) with 95% CI and p-values. Test
# choice: t.test() compares group means directly and is valid for binary
# outcomes as a difference-in-means test. Justification: this aligns with the
# linear probability model interpretation and keeps the contrast on the mean
# scale.
diff_in_means <- function(outcome_var, outcome_name) {
  male <- analysis_main[analysis_main$sex_factor == "Male", ]
  female <- analysis_main[analysis_main$sex_factor == "Female", ]

  mean_m <- mean(male[[outcome_var]], na.rm = TRUE)
  mean_f <- mean(female[[outcome_var]], na.rm = TRUE)

  test <- t.test(male[[outcome_var]], female[[outcome_var]])

  data.frame(outcome = outcome_name, mean_male = mean_m, mean_female = mean_f,
    diff_male_minus_female = mean_m - mean_f, ci_lower = test$conf.int[1], ci_upper = test$conf.int[2],
    p_value = test$p.value)
}

dim1 <- diff_in_means("outcome1", "outcome_binary1")
dim2 <- diff_in_means("outcome2", "outcome_binary2")

diff_table <- rbind(dim1, dim2)
if (knitr::is_latex_output()) {
  knitr::kable(diff_table, digits = 4, format = "latex", caption = "Difference-in-means (Male - Female)")
} else {
  knitr::kable(diff_table, digits = 4, format = "html", caption = "Difference-in-means (Male - Female)")
}
```

```
float_barrier()
```

```
## \clearpage
```

```
# Figure 1: Two-panel bar chart of outcome rates by sex with 95% CI error bars.
# Panel A: Reciprocal conciliation (outcome1). Panel B: Unilateral
# conciliation (outcome2). CI method: exact binomial confidence interval from
# binom.test().
```

```
compute_rate_ci <- function(x, group) {
  x_group <- x[group]
  n <- sum(!is.na(x_group))
  if (n == 0) {
```

```

    return(c(rate = NA_real_, lower = NA_real_, upper = NA_real_))
  }
  successes <- sum(x_group == 1, na.rm = TRUE)
  ci <- binom.test(successes, n)$conf.int
  c(rate = successes/n, lower = ci[1], upper = ci[2])
}

plot_rate_with_ci <- function(rate_ci, title_text) {
  bar_pos <- barplot(height = rate_ci["rate", ], names.arg = c("Female", "Male"),
    ylim = c(0, 1), main = title_text, ylab = "Proportion", xlab = "")
  arrows(x0 = bar_pos, y0 = rate_ci["lower", ], x1 = bar_pos, y1 = rate_ci["upper",
    ], angle = 90, code = 3, length = 0.05)
}

sex_is_female <- analysis_main$sex_factor == "Female"
sex_is_male <- analysis_main$sex_factor == "Male"

rate_ci_outcome1 <- cbind(Female = compute_rate_ci(analysis_main$outcome1, sex_is_female),
  Male = compute_rate_ci(analysis_main$outcome1, sex_is_male))
rate_ci_outcome2 <- cbind(Female = compute_rate_ci(analysis_main$outcome2, sex_is_female),
  Male = compute_rate_ci(analysis_main$outcome2, sex_is_male))

draw_figure_1 <- function(add_caption) {
  op <- par(no.readonly = TRUE)
  on.exit(par(op), add = TRUE)
  par(mfrow = c(1, 2), mar = c(4, 4, 3, 1), oma = c(2, 0, 0, 0), cex.main = 0.9,
    family = "serif")
  plot_rate_with_ci(rate_ci_outcome1, "Reciprocal conciliation by sex")
  plot_rate_with_ci(rate_ci_outcome2, "Unilateral conciliation by sex")
  if (add_caption) {
    mtext("Figure 1: Conciliation by Sex with 95% CIs", side = 1, outer = TRUE,
      line = 1, cex = 0.9)
  }
}

# Render for the knitted output (caption handled by fig.cap in PDF).
draw_figure_1(add_caption = FALSE)

```

```

# Also write a captioned PNG for the output folder.
png(filename = file.path(output_dir, "figure_1_conciliation.png"), width = 7, height = 3.5,
  units = "in", res = 300)
draw_figure_1(add_caption = TRUE)
dev.off()

```

```

## png
## 2

```

```
float_barrier()
```

```
## \clearpage
```



Figure 1: Conciliation by Sex with 95% CIs

```
# Main LPMs: outcome ~ sex + age (listwise deletion occurs naturally). Model
# choice: linear probability models are used as the main specification.
# Justification: LPM coefficients are directly interpretable as
# percentage-point changes in probability, which is the most transparent way to
# answer the research question in this short data task.
reg_main_out1 <- lm(outcome1 ~ sex_factor + age, data = analysis_main)
reg_main_out2 <- lm(outcome2 ~ sex_factor + age, data = analysis_main)

# Helper: HC3 robust standard error and p-value for a coefficient.
robust_coef_stats <- function(model, term) {
  vc <- sandwich::vcovHC(model, type = "HC3")
  ct <- lmtest::coeftest(model, vcov. = vc)
  if (!(term %in% rownames(ct))) {
    stop(paste("Term not found in model coefficients:", term))
  }
  list(estimate = ct[term, "Estimate"], se = ct[term, "Std. Error"], p = ct[term,
    "Pr(>|t|)"])
}

# Main results table (stargazer) for Section 4.
stargazer(reg_main_out1, reg_main_out2, type = if (knitr::is_latex_output()) "latex" else "html",
  title = "Main LPM Regressions", label = if (knitr::is_latex_output()) "tab:main_lpm" else NULL,
  header = FALSE, float = TRUE, table.placement = "htbp", se = list(sqrt(diag(sandwich::vcovHC(reg_main_out1,
    type = "HC3"))), sqrt(diag(sandwich::vcovHC(reg_main_out2, type = "HC3")))),
  dep.var.labels = c("Outcome 1", "Outcome 2"), covariate.labels = c("Male", "Age"),
  keep.stat = c("n"), digits = 3)

float_barrier()
```

Table 5: Main LPM Regressions

	<i>Dependent variable:</i>	
	Outcome 1	Outcome 2
	(1)	(2)
Male	0.112 (0.128)	0.010 (0.159)
Age	0.002 (0.007)	−0.006 (0.008)
Constant	0.672*** (0.214)	0.653** (0.257)
Observations	45	45
<i>Note:</i>	*p<0.1; **p<0.05; ***p<0.01	

## 5 5. Robustness Checks

```
# Influence/leverage check (concise): drop the single most influential
# observation and re-estimate. Diagnostic used: Cook's distance from the main
# LPM to identify the most influential row. Purpose: assess whether the main
# sex effect is sensitive to one high-leverage observation in a small sample.
refit_drop_top_influence <- function(model, data, label) {
  cooks <- cooks.distance(model)
  drop_idx <- which.max(cooks)
  refit <- lm(formula(model), data = data[-drop_idx, ])
  orig_stats <- robust_coef_stats(model, "sex_factorMale")
  refit_stats <- robust_coef_stats(refit, "sex_factorMale")
  se_orig <- orig_stats$se
  p_orig <- orig_stats$p
  se_refit <- refit_stats$se
  p_refit <- refit_stats$p
  coef_orig <- orig_stats$estimate
  coef_refit <- refit_stats$estimate
  coef_diff <- abs(coef_refit - coef_orig)
  se_diff <- abs(se_refit - se_orig)
  p_diff <- abs(p_refit - p_orig)
  cat("\nInfluence check:", label, "\n")
  cat("Diagnostic: Cook's distance (drop the single largest value).\n")
  cat("Dropped row index:", drop_idx, "\n")
  cat("Sex coefficient (original, refit):", round(coef_orig, 3), "->", round(coef_refit,
    3), "\n")
  cat("SE (original, refit):", round(se_orig, 3), "->", round(se_refit, 3), "\n")
  cat("p-value (original, refit):", round(p_orig, 3), "->", round(p_refit, 3),
    "\n")
  cat("Change summary (absolute): coef=", round(coef_diff, 3), ", SE=", round(se_diff,
    3), ", p-value=", round(p_diff, 3), "\n")
}

refit_drop_top_influence(reg_main_out1, analysis_main, "Main LPM - Outcome 1")

##
## Influence check: Main LPM - Outcome 1
## Diagnostic: Cook's distance (drop the single largest value).
## Dropped row index: 31
## Sex coefficient (original, refit): 0.112 -> 0.08
## SE (original, refit): 0.128 -> 0.127
## p-value (original, refit): 0.388 -> 0.534
## Change summary (absolute): coef= 0.032 , SE= 0.001 , p-value= 0.146

refit_drop_top_influence(reg_main_out2, analysis_main, "Main LPM - Outcome 2")

##
## Influence check: Main LPM - Outcome 2
## Diagnostic: Cook's distance (drop the single largest value).
## Dropped row index: 39
## Sex coefficient (original, refit): 0.01 -> 0.033
```



```
## SE (original, refit): 0.159 -> 0.166
## p-value (original, refit): 0.948 -> 0.844
## Change summary (absolute): coef= 0.022 , SE= 0.007 , p-value= 0.104

# Control robustness: add real_imaginary to the main specification.
# Justification: the plan specifies real/imaginary as a control or sample
# dimension; adding it tests whether the main sex effect is sensitive to
# framing.
reg_main_out1_real <- lm(outcome1 ~ sex_factor + age + real_imaginary_factor, data = analysis_main)
reg_main_out2_real <- lm(outcome2 ~ sex_factor + age + real_imaginary_factor, data = analysis_main)

# Control robustness: add blame_1 to the main specification. Justification:
# tests whether results are sensitive to including perceived blame as an
# additional control. blame_1: higher values indicate more blame assigned to
# the respondent lower values indicate more blame assigned to the other person.
reg_main_out1_blame <- lm(outcome1 ~ sex_factor + age + blame_1, data = analysis_main)
reg_main_out2_blame <- lm(outcome2 ~ sex_factor + age + blame_1, data = analysis_main)

# Across regressions, the gender effect remains small and non-significant.

# Robustness regression table (stargazer) comparing main spec to each check.
robust_models <- list(reg_main_out1, reg_main_out1_real, reg_main_out1_blame, reg_main_out2,
  reg_main_out2_real, reg_main_out2_blame)

stargazer(robust_models, type = if (knitr::is_latex_output()) "latex" else "html",
  title = "Robustness Regressions (Main + Controls)", label = if (knitr::is_latex_output()) "tab:robust",
  header = FALSE, float = TRUE, table.placement = "htbp", object.names = FALSE,
  se = lapply(robust_models, function(m) sqrt(diag(sandwich::vcovHC(m, type = "HC3")))),
  dep.var.labels = c("Outcome 1", "Outcome 2"), column.separate = c(3, 3), covariate.labels = c("Male",
  "Age", "Imagined (ref: Real)", "Blame"), keep.stat = c("n"), digits = 3)

float_barrier()
```

Table 6: Robustness Regressions (Main + Controls)

	<i>Dependent variable:</i>					
	Outcome 1			Outcome 2		
	(1)	(2)	(3)	(4)	(5)	(6)
Male	0.112 (0.128)	0.093 (0.133)	0.055 (0.136)	0.010 (0.159)	−0.010 (0.156)	−0.131 (0.142)
Age	0.002 (0.007)	−0.0003 (0.007)	0.002 (0.006)	−0.006 (0.008)	−0.008 (0.007)	−0.007 (0.011)
Imagined (ref: Real)		−0.289 (0.373)			−0.312 (0.317)	
Blame			0.006 (0.004)			0.016*** (0.003)
Constant	0.672*** (0.214)	0.772*** (0.240)	0.462* (0.248)	0.653** (0.257)	0.760*** (0.252)	0.129 (0.365)
Observations	45	45	45	45	45	45
<i>Note:</i>				*p<0.1; **p<0.05; ***p<0.01		

## 6 6. Exploratory: Blame - Sex Interaction

```
# Exploratory check: interaction between blame_1 and sex. Model includes main
# effects plus interaction, with age as a control.
reg_int_out1 <- lm(outcome1 ~ sex_factor * blame_1 + age, data = analysis_main)
reg_int_out2 <- lm(outcome2 ~ sex_factor * blame_1 + age, data = analysis_main)

stargazer(reg_int_out1, reg_int_out2, type = if (knitr::is_latex_output()) "latex" else "html",
  title = "Exploratory Interaction: Sex * Blame", label = if (knitr::is_latex_output()) "tab:interact",
  se = list(sqrt(diag(sandwich::vcovHC(reg_int_out1, type = "HC3"))), sqrt(diag(sandwich::vcovHC(reg_int_out2,
  type = "HC3")))), dep.var.labels = c("Outcome 1", "Outcome 2"), covariate.labels = c("Male",
  "Blame", "Male * Blame", "Age"), keep.stat = c("n"), digits = 3, header = FALSE,
  float = TRUE, table.placement = "htbp")
```

Table 7: Exploratory Interaction: Sex \* Blame

	<i>Dependent variable:</i>	
	Outcome 1	Outcome 2
	(1)	(2)
Male	0.443 (0.388)	0.295 (0.214)
Blame	0.013** (0.006)	0.023*** (0.004)
Male * Blame	0.002 (0.006)	-0.007 (0.010)
Age	-0.010 (0.008)	-0.011** (0.006)
Constant	0.243 (0.327)	-0.113 (0.307)
Observations	45	45

*Note:* \*p<0.1; \*\*p<0.05; \*\*\*p<0.01

```
float_barrier()
```