```
# R course for beginners
# Week 7
# assignment by Noam Gabay, id 208540815


### preprocessing ----


# creating raw data

subjects = dir("stroop_data")
df = data.frame()

for (subject in subjects)  {
  temp_data = read.csv(file.path("stroop_data", subject))
  df = rbind(df, temp_data)
}


# piping

library(dplyr)

df = df |>
  mutate(
    task = ifelse(grepl("word_reading", condition), "word_reading",
                  ifelse(grepl("ink_naming", condition), "ink_naming", NA)),

    congruency = ifelse(grepl("_cong", condition), "congruent",
                        ifelse(grepl("incong", condition), "incongruent", NA)),

  )

df = df |>
  mutate(
    accuracy = (participant_response == correct_response) * 1
  )


df = df |>
  select(subject, task, congruency, block, trial, accuracy, rt)


df = df |>
  mutate(
    subject = factor(subject),
    task = factor(task, levels = c("word_reading", "ink_naming")),
    congruency = factor(congruency, levels = c("congruent", "incongruent")),
  )


save(df, file = "raw_data.rdata")


### creating filtered data ----

# number of subjects
n_subjects = length(unique(df$subject))

# drop rows with NAs
df_filtered  = na.omit(df)

# condition for rt (assuming rt is in ms)
df_filtered = df_filtered |>
```

```
  filter(
    (df_filtered$rt > 300) & (df_filtered$rt < 3000)
  )


# trial percentage

n_trials_original = df |>
  group_by(subject) |>
  summarize(trials = n())

n_trials_left = df_filtered |>
  group_by(subject) |>
  summarize(trials = n())


trial_data = data.frame(
  subject = unique(df_filtered$subject),

  percent_remaining = n_trials_left$trials / n_trials_original$trials * 100 ,
  percent_removed = (n_trials_original$trials - n_trials_left$trials) /
n_trials_original$trials *100
  )

print(trial_data)

trial_summary = trial_data |>
  summarize(
    mean_removed = mean(percent_removed, na.rm = TRUE),
    sd_removed = sd(percent_removed, na.rm = TRUE)
  )

print(trial_summary)

save(df_filtered, file= "filtered_data.csv")


### descriptive statistics ----

summary_conditions = df_filtered |>
  group_by(task, congruency) |>
  summarize(
    mean_rt = mean(rt, na.rm = TRUE),
    sd_rt = sd(rt, na.rm = TRUE),
    mean_accuracy = mean(accuracy, na.rm = TRUE),
    sd_accuracy = sd(accuracy, na.rm = TRUE)
  )



### plotting ----


library(ggplot2)


#plot rt

ggplot(summary_conditions, aes(x = task, y = mean_rt, fill = congruency)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(aes(ymin = mean_rt - sd_rt, ymax = mean_rt + sd_rt),
                position = position_dodge(0.9), width = 0.25) +
  labs(
    title = "Mean Reaction Times by Task and Congruency",
    x = "Task",
```

```
    y = "Mean Reaction Time (ms)",
    fill = "Congruency"
  ) +
  theme_minimal()

# scatter plot

ggplot(df_filtered, aes(x = task, y = rt, color = congruency)) +
  geom_point(alpha = 0.5, position = position_jitter(width = 0.2)) +  # נקודות עם
רנדומליות קלה
  labs(
    title = "Reaction Times by Task and Congruency",
    x = "Task",
    y = "Reaction Time (ms)",
    color = "Congruency"
  ) +
  theme_minimal()


### stats ----


library(lme4)

# to account for the fact that each subject did all conditions
# I consulted with chatGPT on how to include subject as a random effect
# and got:
# (1 | subject) for variance in intercept between participants
# (task * congruency | subject) for variance in slope between participants

model = lmer(rt ~ task * congruency + (1 + task * congruency | subject), data =
df_filtered)

coef(model)
summary(model)
```