

d-prime_analyse1

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v dplyr      1.1.4      v readr      2.1.5
v forcats    1.0.1      v stringr    1.5.2
v ggplot2    4.0.0      v tibble     3.3.0
v lubridate  1.9.4      v tidyr      1.3.1
v purrr      1.1.0
```

```
-- Conflicts ----- tidyverse_conflicts() --
```

```
x dplyr::filter() masks stats::filter()
```

```
x dplyr::lag()     masks stats::lag()
```

```
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```
path <- "~/Desktop/Python/EM2-audio/experiment/control_experiment/data_detection/ny_data"
```

```
all_files <- fs::dir_ls(path, glob = "*.csv")
```

```
raw_data <- map_dfr(all_files, read_csv, .id = "source_file")
```

New names:

Rows: 224 Columns: 25

-- Column specification

----- Delimiter: "," chr

(11): signal_type, prime_file, correct_word, babbling_file, mask1_file, ... dbl

(11): compression_level, practice.thisRepN, practice.thisTrialN, practic... lgl

(3): thisRow.t, notes, ...25

i Use `spec()` to retrieve the full column specification for this data. i

Specify the column types or set `show_col_types = FALSE` to quiet this message.

* `` -> `...25`

```

tidy_data <- raw_data %>%

# --- Remove Practice Trials ---
# This logic is still correct. We keep rows where the 'main'
# loop was running (i.e., 'main.thisN' is not NA).
filter(!is.na(main.thisN)) %>%

# --- Select Only the Useful Columns ---
# We use the REAL column names from your glimpse() output.
# Use backticks `` for names with spaces like `Participant ID`.
select(
  # Participant info
  participant = `Participant ID`, # Rename `Participant ID` to `participant`
  session = Session,             # Rename `Session` to `session`

  # Trial info
  trial_num = main.thisN,        # This is the trial number

  # Experiment condition columns (from your data)
  signal_type,
  compression_level,
  prime_file,
  correct_word,
  babbling_file,
  mask1_file,
  mask2_file,

  # Dependent Variables (the participant's response)
  key_pressed = response_key, # Rename `response_key` to `key_pressed`
  rt,
  trial_outcome
) %>%

# --- Fix Data Types ---
# Convert columns to the correct type for analysis
mutate(
  # Convert RT to numeric
  rt = as.numeric(rt),

  # Convert key identifiers to factors (categorical variables)
  participant = as.factor(participant),
  session = as.factor(session),

```

```

    key_pressed = as.factor(key_pressed),

    # Convert valences to factors
    signal_type = as.factor(signal_type),
    trial_outcome = as.factor(trial_outcome)
  )

# 4. SHOW THE RESULT
#-----
# Print the first few rows of your new, clean data frame
print(tidy_data)

# A tibble: 224 x 13
  participant session trial_num signal_type compression_level prime_file
  <fct>      <fct>      <dbl> <fct>          <dbl> <chr>
1 johan_new  001          -1 absent          0.2 SILENCE
2 johan_new  001          -1 absent          0.6 SILENCE
3 johan_new  001          -1 present          0.7 redskab_compress~
4 johan_new  001          -1 absent          0.3 SILENCE
5 johan_new  001          -1 present          0.6 gade_compressed.~
6 johan_new  001          -1 absent          0.5 SILENCE
7 johan_new  001          -1 absent          0.1 SILENCE
8 johan_new  001          -1 present          0.5 s de_compressed.~
9 johan_new  001          -1 present          0.1 motor_compressed~
10 johan_new 001          -1 present          0.4 salat_compressed~
# i 214 more rows
# i 7 more variables: correct_word <chr>, babbling_file <chr>,
#   mask1_file <chr>, mask2_file <chr>, key_pressed <fct>, rt <dbl>,
#   trial_outcome <fct>

# Show a summary of the data frame structure
glimpse(tidy_data)

```

```

Rows: 224
Columns: 13
$ participant    <fct> johan_new, johan_new, johan_new, johan_new, johan_ne~
$ session       <fct> 001, 001, 001, 001, 001, 001, 001, 001, 001, 001, 00~
$ trial_num     <dbl> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, ~
$ signal_type   <fct> absent, absent, present, absent, present, absent, ab~
$ compression_level <dbl> 0.2, 0.6, 0.7, 0.3, 0.6, 0.5, 0.1, 0.5, 0.1, 0.4, 0.~
$ prime_file    <chr> "SILENCE", "SILENCE", "redskab_compressed.wav", "SIL~

```

```

$ correct_word      <chr> NA, NA, "redskab", NA, "gade", NA, NA, "sæde", "moto~
$ babbling_file    <chr> "babble_43.wav", "babble_18.wav", "babble_47.wav", "~
$ mask1_file       <chr> "pervers_reversed.wav", "fjendtlig_reversed.wav", "b~
$ mask2_file       <chr> "skuffe_reversed.wav", "bange_reversed.wav", "konkur~
$ key_pressed      <fct> z, z, z, z, m, z, z, z, m, m, z, m, z, z, z, z, m~
$ rt               <dbl> 3.4099970, 0.8748126, 2.0227927, 0.6059912, 0.501148~
$ trial_outcome    <fct> Correct Rejection, Correct Rejection, Miss, Correct ~

```

```

# Load the ggplot2 library (part of tidyverse)
library(ggplot2)

# Create the 100% stacked bar chart
# We use the 'tidy_data' data frame from the previous step

compression_plot <- ggplot(tidy_data, aes(x = factor(compression_level), fill = trial_outcome))

# geom_bar(position = "fill") creates a 100% stacked bar chart,
# which is perfect for comparing proportions across groups.
geom_bar(position = "fill") +

# Add clear labels
labs(
  title = "Proportion of Trial Outcomes by Compression Level",
  x = "Compression Level",
  y = "Proportion",
  fill = "Trial Outcome"
) +

# Format the y-axis to show percentages (e.g., 25%, 50%)
scale_y_continuous(labels = scales::percent_format()) +

# Apply a clean theme
theme_minimal()

# Save the plot to a file
ggsave("compression_outcomes_plot.png", plot = compression_plot, width = 10, height = 6)

# Print the plot object so it displays
print(compression_plot)

```

