ManyLLMs: Data Cleaning

XXX

2025-05-21

Contents

1 Overview

This script accompanies the analyses presented in the manuscript. Two versions are provided:

- The `.Rmd` file can be opened and edited in **RStudio**.
- A rendered **PDF** is available for reference, generated by knitting this file.

We provide full R code to ensure the analyses are transparent and reproducible.
Where necessary, code is explained or unpacked in-text. This document includes:

1. Visualizations of key variable distributions (used in main and supplementary materials)
2. Justifications for major analysis decisions

Last updated: ***21 May 2025***

2 Step 1: Data Preparation

In this step, we prepare the environment and load data from three experimental conditions.

First, we **clear the workspace** using `rm(list = ls(all.names = TRUE))` and load the necessary R libraries such as `dplyr`, `ggplot2`, `forcats`, and `patchwork` for data manipulation and visualization.

We then read in three `.csv` files corresponding to the three group conditions:

- *Single-agent condition*: stored in the variable single
- *Pair-agent condition*: stored in nob_2
- *Triad-agent condition*: stored in nob_3

Each dataset receives an additional column, experiment, which labels its group condition.

These datasets are then **merged** using `bind_rows()` into one unified data frame:
combined_df

This combined dataset will be filtered, cleaned, and analyzed in the next steps.

```
rm(list = ls(all.names = TRUE))


setwd("/Users/nita/Downloads")

# 1. Read each file and add an experiment label

# Single Agent (Solo)
single   <- read.csv("single_no_message_f.csv",   header = TRUE, stringsAsFactors = TRUE) %>%
  mutate(experiment = "Single")

#Pairs (groups)
nob_2    <- read.csv("nob_2_no_message_f.csv",   header = TRUE, stringsAsFactors = TRUE) %>%
  mutate(experiment = "NoB_2")

#Triad (groups)
nob_3    <- read.csv("nob_3_no_message_f.csv",     header = TRUE, stringsAsFactors = TRUE) %>%
  mutate(experiment = "NoB_3")

# 3. Combine
combined_df <- bind_rows(single,nob_2,nob_3)
```

We remove the column onboarding from combined_df as it is *not needed* for further analysis.

Next, we **filter the dataset** to retain only the observations from step -1 (pre-condition) and step 7 (post-condition), which correspond to the *single* and *group* conditions. After filtering, we apply `droplevels()` to clean up unused factor levels.

To focus our analysis on relevant moral judgments, we also remove any **non-moral scenarios** by filtering out rows where the type column is `"Non-Moral"`.

Finally, we simplify verbose model names by recoding them using `fct_recode()` and storing the cleaned names in a new column:
model_short

This step ensures the dataset is concise and easier to interpret in subsequent analysis and visualization.

```r
# 4.  Drop the unwanted columns
combined_df <- combined_df %>% select(-onboarding)


# 5. Select only Step -1 and 7 opinion (Single vs. Groups) no discussion
combined_df= filter(combined_df, step == -1 | step ==7) %>% droplevels()

# 6. delete the non-moral scenarios
combined_df= filter(combined_df, type!="Non-Moral") %>% droplevels
```

Next, we **standardize** the dataset names by recoding `"oxford_utilitarianism_scale"` to a shorter label: `"oxford"`.

We also recode values in the `type` column to group similar scenario categories under clearer, more interpretable labels — for example:
`"Factual-Killing-Utilitarian"` becomes `"Killing-Util"` and
`"Factual-Saving-Deontological"` becomes `"Saving-Deon"`.

To enhance readability, we also **shorten long model names** by assigning simplified names to a new variable: `model_short`.

For items belonging to the `"CNI"` dataset, we create a new column `cni_type` by mapping each `example_index` to a predefined list of CNI scenario codes.

From these codes, we extract meaningful suffixes such as `"acinc"`, `"ininc"`, `"incon"`, or `"accon"` and update the `type` column accordingly.

This finalizes the dataset's semantic structure, aligning it with the categories used throughout the manuscript.

```r
# 7. Make llm names shorter

combined_df <- combined_df %>%
  mutate(
    model_short = fct_recode(model,
      "GPT4.1"   = "gpt-4.1",
      "Lamma3.3" = "llama3.3",
      "Qwen2.5"= "qwen2.5:32b-instruct",
      "Qwen3"  = "qwen3:32b",
      "QWQ"    = "qwq",
      "Gemma3" = "gemma3:27b"
    )
)


# 7 changing the levles of dataset
combined_df <- combined_df %>%
  mutate(dataset = recode(dataset, "oxford_utilitarianism_scale" = "oxford"))
# 8. changing the type:

library(forcats)

combined_df$type <- fct_recode(combined_df$type,
  "Action"        = "action",
  "Killing-Util"  = "Factual-Killing-Utilitarian",
  "Other-Deon"    = "Factual-Other-Deontological",
  "Other-Util"    = "Factual-Other-Utilitarian",
```

```
  "Saving-Deon"   = "Factual-Saving-Deontological",
  "Beneficence"   = "Impartial Beneficence",
  "Harm"          = "Instrumental Harm",
  "Omission"      = "omission",
  "CNI"           = "N/A"
  # "Impersonal" and "Personal" remain unchanged
)


#fix cni types
cni_labels <- c(
  "d1acinc", "d6ininc", "d5ininc", "d4incon", "d3ininc", "d2ininc",
  "d1ininc", "d2acinc", "d6accon", "d5accon", "d4acinc", "d3incon",
  "d2incon", "d6incon", "d1incon", "d3accon", "d5acinc", "d4accon",
  "d2accon", "d6acinc", "d3acinc", "d4ininc", "d5incon", "d1accon"
)
combined_df <- combined_df %>%
  mutate(example_index = as.character(example_index)) %>%
  mutate(
    cni_type = case_when(
      type == "CNI" ~ factor(
        example_index,
        levels = as.character(seq_along(cni_labels)),
        labels = cni_labels
      ),
      TRUE ~ NA_character_
    )
  )

combined_df <- combined_df %>%
  mutate(
    type = ifelse(
      type == "CNI",
      str_extract(cni_type, "acinc|ininc|incon|accon"),
      as.character(type)
    ),
    type = factor(type, levels = c("acinc", "ininc", "incon", "accon",
                                   "Action", "Killing-Util", "Other-Deon",
                                   "Other-Util", "Saving-Deon", "Beneficence",
                                   "Harm", "Impersonal", "Personal", "Omission"))
  )
```

Finally, we address a known **misalignment** in the greene dataset:
any example_index values greater than 41 are off by one.
We correct this by subtracting 1 from those values.

The corrected `example_index` column is then converted into a factor named item to prepare it for grouped analysis.

As a sanity check, we verify that **all expected item indices** from 1 to 9 are present.

This concludes the *initial data preparation step*, ensuring that the dataset is **clean**, **consistent**, and fully ready for downstream analysis.

```r
#Fixiing issues with Green

library(dplyr)
combined_df$example_index=as.numeric(combined_df$example_index)
# Step 1: Apply correction ONLY to greene where index > 41
combined_df <- combined_df %>%
  mutate(
    example_index = case_when(
      dataset == "greene" & example_index > 41 ~ example_index - 1,
      TRUE ~ example_index
    )
  )
combined_df$item=as.factor(combined_df$example_index)
combined_df$stepf=as.factor(combined_df$step)



expected_indices <- 1:max(combined_df$example_index)
missing_indices <- setdiff(expected_indices, combined_df$example_index)

expected_indices <- 1:max(combined_df$example_index)
missing_indices <- setdiff(expected_indices, combined_df$example_index)
if (length(missing_indices) == 0) {
  cat("  All indices from 1 to", max(combined_df$example_index), "are present.\n")
} else {
  cat("  Missing indices:", paste(missing_indices, collapse = ", "), "\n")
}
```

```
##   All indices from 1 to 64 are present.
```

There are 0 missing indices between 1 and 64.

3 Step 2: Cleaning and Formatting

Note that the dataset has already been *pre-processed*, as described in the **Supplementary Materials**. Here, we check for any **additional mismatches or missing entries** that may affect downstream analysis.

In this step, we clean the `combined_df` dataset by identifying and removing problematic values in the `opinion` column (also referred to as the *utilitarian score*).

```r
# 1. Convert to character and numeric
opinion_char <- as.character(combined_df$opinion)
opinion_num  <- suppressWarnings(as.numeric(opinion_char))

# 2. Define conditions
is_na         <- is.na(combined_df$opinion)
is_text       <- !str_detect(opinion_char, "^-?\\d+(\\.\\d+)?$")
out_of_range  <- !is.na(opinion_num) & (opinion_num < 1 | opinion_num > 7)
invalid_all   <- is_na | is_text | out_of_range
```

We begin by converting opinion into two formats:
- as character: opinion_char
- as numeric: opinion_num

```r
# Calculate total invalid entries
total_rows    <- nrow(combined_df)
total_missing <- sum(invalid_all, na.rm = TRUE)
```

We then define several **invalid conditions**:

- is_na — detects *missing values*
- is_text — identifies *non-numeric entries* (e.g., text or symbols)
- out_of_range — flags numeric values *outside the valid 1–7 Likert scale*

All three are combined into a master logical flag: **invalid_all**.
We then compute and print their counts.

The **total number of invalid entries** is stored in total_missing, which equals r total_missing observations. These include any missing, misformatted, or out-of-range values in the data.

The **total number of invalid entries** is stored in `total_missing`, which equals 289 observations. These include any missing, misformatted, or out-of-range values in the data.

We print a concise summary of issues found in the `opinion` column of `combined_df`:

- **NA values** are missing responses.
- **Text values** are improperly formatted (e.g., strings instead of numbers).
- **Out-of-range values** fall outside the valid Likert scale (1–7).
- All of these are grouped under **invalid entries** and counted as a share of the total.

This summary helps assess the integrity of the data before applying filters or imputation.

```r
# 3. Print counts
# Summary counts for opinion cleaning
total_rows    <- nrow(combined_df)
na_count      <- sum(is_na)
text_count    <- sum(is_text)
range_count   <- sum(out_of_range)
invalid_total <- sum(invalid_all)
```

Table 1: Table: Data Quality Summary

Metric	Count
Total rows	51090
NA values	0
Non-numeric (text)	39
Out-of-range (not 1–7)	250
Total invalid entries	289

```r
# 4. Show the percentage of missing values

total_rows    <- nrow(combined_df)
total_missing <- sum(invalid_all, na.rm = TRUE)
```