Data Processing

Erin M. Buchanan

Last Update 2021-08-25

Libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(psych)
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following objects are masked from 'package:reshape':
##
##     expand, smiths
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

Data Processing Example

```
##import overall dataset
master <- read.csv("input_data/example_data.csv", stringsAsFactors = F)

##create demographics only data
demos <- master %>% #data frame
  filter(sender == "Demographics Form") #filter out only demographics lines

##we are only interested in the real trials
real_trials <- master %>% #data frame
  filter(sender == "Stimulus Real") #filter out only the real stimuli
```

Trial Data

Each language will be saved in a separate file with an item specific trial identification number to allow for matching concepts across languages (i.e., cat → katze → gatta).

Participants are expected to incorrectly answer trials, and these trials will be marked for potential exclusion. Further, computer errors or trials due to missing data (i.e., participant inattentiveness and timeout trials, internet disconnection, computer crashes) will be marked as such in the final data with NA values.

```
##mark the trials as correct or incorrrect
##could add this to the experiment file so it would be there
##when you downloaded but here's how to code it
real_trials$correct <- real_trials$class_real == real_trials$response

##specifically mark time out or other computer problems
real_trials$correct[real_trials$ended_on == "timeout"] <- NA
```

The response latencies from each participant's session will be z-scored in line with recommendations from Faust, Balota, Spieler, and Ferraro (1999).

```
##Separate out NA data for the z-score part
##this mostly controls for timeouts
real_trials_NA <- real_trials %>% #data frame
  filter(is.na(correct)) #filter out NA corrects

#set all Z_RTs to NA for time outs
real_trials_NA$Z_RT = NA
#set all duration to NA for time outs
real_trials_NA$duration = NA

##this section z-scores the rest of the data
##you do include incorrect trials for the z-score
##just not time outs
real_trials_nonNA <-
  real_trials %>% #data frame
  group_by(openLabId) %>% #group by participant
  filter(!is.na(correct)) %>% #take out the NA timeouts
  mutate(Z_RT = scale(duration)) #create a z-score RT
```

Next, we will merge these back together with the demographics.

```
##put the time outs with the answered trials
real_trials <- bind_rows(real_trials_NA, real_trials_nonNA)

##mark the demo column names
demo_columns <- c("gender", "year", "edu", "language", "openLabId")

##mark the real column names
real_columns <- c(setdiff(colnames(real_trials), demo_columns), "openLabId")

##merge the demographics with teh real trials
##the column name code above just keeps us from having a bunch
##of doubled demographic column names
real_trials <- merge(demos[ , demo_columns ],
                     real_trials[ , real_columns],
                     by = "openLabId",
                     all = T)

##write out raw trial data
write.csv(real_trials, "output_data/trial_data.csv", row.names = F)
```

Item Data

The item file will contain lexical information about all stimuli (length, frequency, orthographic neighborhood, bigram frequency). These files will be merged with the stimuli that were created in the previous step (for this example, using a small example of what that stimuli dataset might look like).

The descriptive statistics calculated from the trial level data will then be included: average response latency, average standardized response latency, sample size, standard errors of response latencies, and accuracy rate. For averages and standard errors, the incorrect and missing trials will be excluded.

```
##read in stimuli data
stimuli_data <- read.csv("../04_Procedure/example_stimuli.csv", stringsAsFactors = F)

##create item level data by summarizing
item_data <- real_trials %>% #data frame
  group_by(word_real) %>%  #group by word
  summarize(avgRT = mean(duration, na.rm = T), #average RT
            avgZ_RT = mean(Z_RT, na.rm = T), #average Z RT
            samplesize = length(na.omit(Z_RT)), #sample size correct
            seRT = sd(duration, na.rm = T)/sqrt(length(na.omit(duration))), #SE RT
```

```
            seZ_RT = sd(Z_RT, na.rm = T)/sqrt(length(na.omit(Z_RT))), #SE Z RT
            accuracy = length(na.omit(Z_RT))/length(Z_RT) #accuracy
            ) %>%
  filter(!is.na(word_real)) #take out blank participants with no trials
```

No data will be excluded for being a potential outlier, however, we will recommend cut off criterion for z-score outliers at 2.5 and 3.0 and will calculate these same statistics with those subsets of trials excluded.

```
##example outlier exclusion for Z > 2.5
##same as above with one extra filter
item_data_2.5 <- real_trials %>%
  filter(abs(Z_RT) < 2.5) %>% #take out trials above 2.5 z scores
  group_by(word_real) %>%
  summarize(avgRT = mean(duration, na.rm = T),
            avgZ_RT = mean(Z_RT, na.rm = T),
            samplesize = length(na.omit(Z_RT)),
            seRT = sd(duration, na.rm = T)/sqrt(length(na.omit(duration))),
            seZ_RT = sd(Z_RT, na.rm = T)/sqrt(length(na.omit(Z_RT)))) %>%
  filter(!is.na(word_real))

##make new column names for these calculations
colnames(item_data_2.5)[-1] <- paste("Z2.5_", colnames(item_data_2.5)[-1], sep = "")

#merge together two z score calculations
item_data_combo <- merge(item_data, item_data_2.5, by = "word_real")

##example outlier exclusion for Z > 3.0
##same as above with one extra filter
item_data_3.0 <- real_trials %>%
  filter(abs(Z_RT) < 3.0) %>% #take out trials above 2.5 z scores
  group_by(word_real) %>%
  summarize(avgRT = mean(duration, na.rm = T),
            avgZ_RT = mean(Z_RT, na.rm = T),
            samplesize = length(na.omit(Z_RT)),
            seRT = sd(duration, na.rm = T)/sqrt(length(na.omit(duration))),
            seZ_RT = sd(Z_RT, na.rm = T)/sqrt(length(na.omit(Z_RT)))) %>%
  filter(!is.na(word_real))

##make new column names for these calculations
colnames(item_data_3.0)[-1] <- paste("Z3.0_", colnames(item_data_3.0)[-1], sep = "")

#merge together two z score calculations
item_data_combo <- merge(item_data, item_data_2.5, by = "word_real")
item_data_combo <- merge(item_data_combo, item_data_3.0, by = "word_real")
```

For all real words, the age of acquisition, imageability, concreteness, valence, dominance, arousal, and familiarity values will be indicated because these values do not exist for nonwords.

```
##merge with stimuli data
item_data_combo <- merge(item_data_combo,
                         stimuli_data, #merge all stimuli information
                         by.x = "word_real",
                         by.y = "string")

##write out item level data
write.csv(item_data_combo, "output_data/item_data.csv", row.names = F)
```

Priming Data

Priming is defined as the subtraction of average z-scored related response latency for an item from the corresponding item in the unrelated condition. Also, we've included the calculation for non-scaled data, but the z-score calculation is recommended.

```
##create a data frame with the information about trial order
real_pairs <- real_trials %>% #data frame
  mutate(two.words = paste(word_real, lead(word_real), sep = "-")) #word 1-word2

##create a list of trial types
##this will be better in the future with coding embedded in the experiment
##additionally pairs can only be shown one-way, the demo just randomized
trial_types <- stimuli_data %>% #data frame
  filter(stim_type != "nonword") %>% #take out nonwords, no priming
  select(string, stim_type, unrelated, related) %>% #only take some columns
  #paste the trial order together
  mutate(unrelated_pair = paste(string, unrelated, sep = "-")) %>%
  mutate(unrelated_pair2 = paste(unrelated, string, sep = "-")) %>%
  mutate(related_pair = paste(string, related, sep = "-")) %>%
```

```
  mutate(related_pair2 = paste(related, string, sep = "-"))

##subset out only the trial names and orders
trial_types <- trial_types[ , grepl("pair", colnames(trial_types))]
##melt into long format
trial_types <- melt(trial_types, measure.vars = colnames(trial_types))
##just call them related and unrelated
##this section will be unnecessary when the exp is fully coded
trial_types$variable <- gsub("2", "", trial_types$variable)

##mark if the pair is related or unrelated or nonword
##merge the trial type into the real data
real_pairs <- merge(real_pairs, trial_types,
                    by.x = "two.words", by.y = "value")

##use only the target words, not the first priming word
target_words <- subset(stimuli_data,
                       target == 1 & stim_type == "real", #target is yes (1, second word) and stimuli is real word
                       select = "string")$string

##take the real pairs of only target words
targets_only <- real_pairs %>%
  ##filter out the priming words and nonwords
  filter(word_real %in% target_words) %>%
  ##group them by target word and condition related/unrelated
  group_by(word_real, variable) %>%
  ##create average scores by condition
  summarize(avgRT = mean(duration, na.rm = T),
            avgZ_RT = mean(Z_RT, na.rm = T),
            samplesize = length(na.omit(Z_RT)),
            seRT = sd(duration, na.rm = T)/sqrt(length(na.omit(duration))),
            seZ_RT = sd(Z_RT, na.rm = T)/sqrt(length(na.omit(Z_RT)))) %>%
  ##spread that into wide format so we can subtract
  pivot_wider(names_from = "variable",
              values_from = c("avgRT", "avgZ_RT", "samplesize", "seRT", "seZ_RT")) %>%
  ##create the priming scores by subtracting unrelated - related for that target word only
  mutate(avgRT_prime = avgRT_unrelated_pair - avgRT_related_pair) %>%
  mutate(avgZ_prime = avgZ_RT_unrelated_pair - avgZ_RT_related_pair)
```

```
## `summarise()` has grouped output by 'word_real'. You can override using the `.groups` argument.
```

```
## this process will be repeated for 2.5 and 3.0 z score outliers excluded
targets_only_no2.5 <- real_pairs %>%
  ##filter out the priming words and nonwords
  filter(word_real %in% target_words) %>%
  ##filter out z score outliers
  filter(Z_RT < 2.50) %>%
  ##group them by target word and condition related/unrelated
  group_by(word_real, variable) %>%
  ##create average scores by condition
  summarize(avgRT = mean(duration, na.rm = T),
            avgZ_RT = mean(Z_RT, na.rm = T),
            samplesize = length(na.omit(Z_RT)),
            seRT = sd(duration, na.rm = T)/sqrt(length(na.omit(duration))),
            seZ_RT = sd(Z_RT, na.rm = T)/sqrt(length(na.omit(Z_RT)))) %>%
  ##spread that into wide format so we can subtract
  pivot_wider(names_from = "variable",
              values_from = c("avgRT", "avgZ_RT", "samplesize", "seRT", "seZ_RT")) %>%
  ##create the priming scores by subtracting unrelated - related for that target word only
  mutate(avgRT_prime = avgRT_unrelated_pair - avgRT_related_pair) %>%
  mutate(avgZ_prime = avgZ_RT_unrelated_pair - avgZ_RT_related_pair)
```

```
## `summarise()` has grouped output by 'word_real'. You can override using the `.groups` argument.
```

```
targets_only_no3.0 <- real_pairs %>%
  ##filter out the priming words and nonwords
  filter(word_real %in% target_words) %>%
  ##filter out z score outliers
  filter(Z_RT < 3.00) %>%
  ##group them by target word and condition related/unrelated
  group_by(word_real, variable) %>%
  ##create average scores by condition
  summarize(avgRT = mean(duration, na.rm = T),
            avgZ_RT = mean(Z_RT, na.rm = T),
            samplesize = length(na.omit(Z_RT)),
            seRT = sd(duration, na.rm = T)/sqrt(length(na.omit(duration))),
            seZ_RT = sd(Z_RT, na.rm = T)/sqrt(length(na.omit(Z_RT)))) %>%
  ##spread that into wide format so we can subtract
  pivot_wider(names_from = "variable",
              values_from = c("avgRT", "avgZ_RT", "samplesize", "seRT", "seZ_RT")) %>%
  ##create the priming scores by subtracting unrelated - related for that target word only
```

```
  mutate(avgRT_prime = avgRT_unrelated_pair - avgRT_related_pair) %>%
  mutate(avgZ_prime = avgZ_RT_unrelated_pair - avgZ_RT_related_pair)
```

```
## `summarise()` has grouped output by 'word_real'. You can override using the `.groups` argument.
```

The similarity scores calculated during stimuli selection will be included, as well as other popular measures of similarity if they are available in that language.

```
##merge target information with the similarity scores
targets_only <- merge(targets_only,
                      stimuli_data[ , c("string", "cosine.Buchanan", "fsg.SWOW")],
                      by.x = "word_real",
                      by.y = "string")

targets_only_no2.5 <- merge(targets_only_no2.5,
                            stimuli_data[ , c("string", "cosine.Buchanan", "fsg.SWOW")],
                            by.x = "word_real",
                            by.y = "string")

targets_only_no3.0 <- merge(targets_only_no3.0,
                            stimuli_data[ , c("string", "cosine.Buchanan", "fsg.SWOW")],
                            by.x = "word_real",
                            by.y = "string")

##write out the priming data
write.csv(targets_only, "output_data/prime_data.csv", row.names = F)
write.csv(targets_only_no2.5, "output_data/prime_data_no2.5.csv", row.names = F)
write.csv(targets_only_no3.0, "output_data/prime_data_no3.0.csv", row.names = F)
```