Data Processing

Erin M. Buchanan

9/14/2019

## Data Processing Example

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

##import overall dataset  
master <- read.csv("../exp\_demo/input\_data/example\_data\_task.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, "../exp\_demo/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("../exp\_demo/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")

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, "../exp\_demo/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 = "-"))  
  
library(reshape)

##   
## Attaching package: 'reshape'

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

##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  
  
##this thing is crazy  
library(tidyr)

##   
## Attaching package: 'tidyr'

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

##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)

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")  
  
##write out the priming data   
write.csv(targets\_only, "../exp\_demo/output\_data/prime\_data.csv", row.names = F)