Parameter Estimation

Erin M. Buchanan

8/21/2019

## A Brief Overview

This document supports a proposal to collect data in collaboration with the Psychological Science Accelerator. The purpose of this project is provide semantic priming data across many languages, inspired by the Semantic Priming Project (which is only in English; <http://spp.montana.edu/>; Hutchison et al., 2013). Big data sets are currency for those who do research in psycholinguistics, computational linguistics, natural language processing, and cognitive modeling. These data sets encourage controlled methodology and new scientific questions.

Cue words are defined as those shown first in a priming task, while target words are shown after the cue word. Semantic priming occurs when the target word is facilitated (i.e., responded to faster) in a related pair condition (DOCTOR-NURSE) versus an unrelated pair condition (TREE-NURSE). Therefore, in a priming task, one subject might see DOCTOR-NURSE, while another subject might see TREE-NURSE paired together. The two instances of NURSE will then be compared in an item analysis to see if the subjects who saw the related pairs responded to NURSE faster than the subjects who saw the unrelated pairs (however, some studies simply compare the average response latency of the unrelated condition to the related condition for a group level analysis).

One concern is how to estimate sample size necessary for any particular target word. The magic *N* = 30 has often been used, in an attempt to at least meet some perceived minimum criteria for the central limit theorem. Sample size planning has been promoted when there is a specific parameter goal, such as power to find X effect at specified alpha levels, but no good method has been suggested for knowing when the data around a single word has “settled”. In this power / sample size analysis, we will focus on the lexical decision task in particular, wherein participants are simply asked if a concept presented to them is a word (NURSE) or nonsense word (LURSE). The dependent variable in this study is response latency, and we will use the data from the English Lexicon Project (<http://elexicon.wustl.edu/>; Balota et al., 2007) and the Semantic Priming Project (<http://spp.montana.edu/>; Hutchison et al., 2013) as the metric for our analysis.

Herein, we will also use concepts in accuracy in parameter estimates (AIPE) to think about how we can have the confidence intervals be “sufficiently narrow” (Kelley, 2007; Kelley, Darku, & Chattopadhyay, 2018; Maxwell, Kelley, & Rausch, 2008). Usually, AIPE power/sample size analysis focuses on the standardized mean difference, but here we instead want to know that the estimation of the response latency does not vary by some particular amount. Therefore, it seems that we actually want to focus on the standard error of the response latency, as this determines the width of the confidence interval.

## Examining The English Lexicon Project (ELP)

The English Lexicon Project collected lexical decision (word or nonsense word) and naming (reading the word aloud) data for over 40,000 words. These data provide a good metric for the variability in base response latencies across words, which should allow for the estimation of the number of participants a study should use if the focus is on the standard error of response latencies.

Another issue to consider is that each participant likely has a somewhat arbitrary response latency factor. Usually, you would control for within-subject variance with a random intercept value in a multilevel type analysis, but another suggestion has been to standardize each participant’s responses within a data collection session (Faust et al., 1999).

#read in the ELP data  
ELPmaster <- read.csv("./ldt\_raw/ELPDecisionData.csv")  
  
#use the ave function to create a z-score of each participant  
#they only did one session   
ELPmaster$ZScore <- ave(ELPmaster$RT, #dependent variable  
 ELPmaster$Participant, #group variable  
 FUN = scale) #function, scale is z-scoring  
  
#view the data   
head(ELPmaster)

## Trial Type Accuracy RT Stimulus Participant ZScore  
## 1 1 1 0 707 bookie participant1 0.1030453  
## 2 2 0 1 769 gandbrake participant1 0.4896557  
## 3 3 1 1 526 philosophical participant1 -1.0256075  
## 4 4 0 0 510 umbeaten participant1 -1.1253779  
## 5 5 1 1 512 belonging participant1 -1.1129066  
## 6 6 1 1 626 lowliest participant1 -0.4020424

Let’s first remove all the inaccurate responses (i.e., they decided word/nonsense word incorrectly) and non-words because they do not represent the target words we wish to collect.

#exclude 0 accuracy for incorrect  
#exclude 0 type, which is non-words  
#subset is like filter in tidyverse  
ELPcorrect <- subset(ELPmaster, #data frame  
 Accuracy > 0 & Type > 0) #logical rules to subset by  
  
#droplevels simply excludes the non-word labels that we just dropped  
ELPcorrect$Stimulus <- droplevels(ELPcorrect$Stimulus)

What is the average standard error for our standardized response latencies?

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

##summarize the dataframe to see what the average SE is  
summary\_stats <- ELPcorrect %>% #data frame  
 select(ZScore, Stimulus) %>% #pick the columns  
 group\_by(Stimulus) %>% #put together the stimuli  
 summarize(SES = sd(ZScore)/sqrt(length(ZScore)), samplesize = length(ZScore)) #create SE and the sample size for below   
  
##give descriptives of the SEs  
psych::describe(summary\_stats$SES)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 40455 0.16 0.1 0.14 0.15 0.06 0.01 3.33 3.32 4.71 65.45  
## se  
## X1 0

From this output, we can see that the average and median SE hover around 0.14 to 0.16.

What is the average sample size after data loss due to incorrect answers? Note that there are several real words that only had one participant answer correctly, so they are included in the original sample sizes below, but not in the SE estimate. This explains why total *n* increases between the two code sections here.

##figure out the original sample sizes  
original\_SS <- ELPmaster %>% #data frame  
 count(Stimulus) #count up the sample size  
  
##add the original sample size to the data frame  
summary\_stats <- merge(summary\_stats, original\_SS, by = "Stimulus")  
  
##original sample size average  
psych::describe(summary\_stats$n)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 40472 32.69 0.63 33 32.74 0 29 35 6 -0.91 1.45  
## se  
## X1 0

##reduced sample size  
psych::describe(summary\_stats$samplesize)

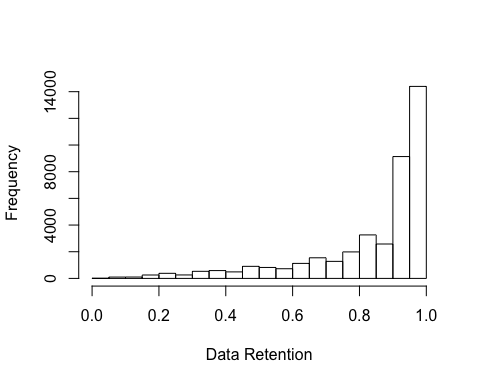
## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 40472 27.41 6.43 30 28.64 2.97 1 35 34 -1.66 2.2  
## se  
## X1 0.03

##percent retained  
psych::describe(summary\_stats$samplesize/summary\_stats$n)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 40472 0.84 0.2 0.91 0.88 0.09 0.03 1 0.97 -1.68 2.25  
## se  
## X1 0

We can see that on average, the ELP usually contained ~32 participants per word. The reduced sample size was about ~27 per word with an average retention rate of 84%. There are many weird words in the ELP (see the histogram of retention rates below, creating a skewed distribution), so the median retention rate might be a better estimation of data loss at ~ .91.

##show the retention rates  
hist(summary\_stats$samplesize/summary\_stats$n,  
 xlab = "Data Retention",  
 main = "")



Let’s look at only the data above the magic *N* = 30 for the best estimate of what level of SE to use as our point at which we would consider the parameter accurate:

##average SE for words with at least n = 30  
summary\_stats %>% #data frame  
 filter(samplesize >=30) %>% #filter out lower sample sizes  
 summarize(avgSES = mean(SES)) #create the mean

## avgSES  
## 1 0.1229559

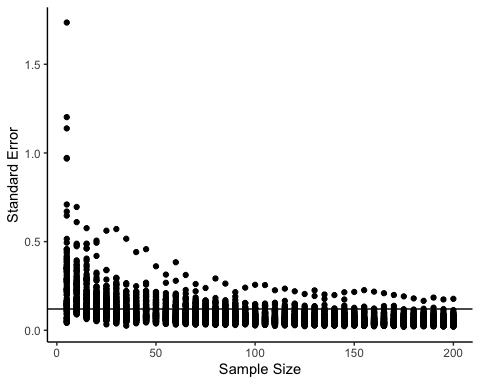
So, potentially, we could set the SE of the ZScore for an item to .12 as our metric of when to stop collecting data.

If I assume these data to be representative, what actual sample size might approximate SE = 0.12?

##pick 100 random words with sample sizes above 30  
targets <- summary\_stats %>% #data frame  
 filter(samplesize >=30) %>% #filter out sample sizes  
 select(Stimulus) %>% #select only stimuli  
 sample\_n(100) %>% #get 100  
 pull(Stimulus) #return a vector  
targets <- as.character(targets)  
  
##this section creates a sequence of sample sizes to estimate at  
#5, 10, 15, etc.   
samplesize\_values <- seq(5, 200, 5)   
#create a blank table for us to save the values in   
sim\_table <- matrix(NA, nrow = length(samplesize\_values), ncol = length(targets))  
#create column names based on the current targets  
colnames(sim\_table) <- targets  
#make it a data frame  
sim\_table <- as.data.frame(sim\_table)  
#add those sample size values   
sim\_table$sample\_size <- samplesize\_values  
  
##loop over all the target words randomly selected  
for (i in 1:length(targets)){  
   
 ##loop over sample sizes  
 for (q in 1:length(samplesize\_values)){  
   
 ##temporarily save a data frame of Zscores   
 temp <- ELPcorrect %>% #data frame  
 filter(Stimulus == targets[i]) %>% #pick rows that are the current target word  
 sample\_n(samplesize\_values[q], replace = T) %>% #select sample size number of rows   
 pull(ZScore)  
   
 #put that in the table  
 #find the sample size row and column we are working with  
 #calculate SE sd/sqrt(n)  
 sim\_table[sim\_table$sample\_size == samplesize\_values[q], targets[i]] <- sd(temp)/sqrt(length(temp))  
   
 }  
   
 }

Obviously, each run of this exercise will be different because it’s randomly selected, but let’s graph the data:

##load some libraries  
library(ggplot2)  
library(reshape)  
  
##melt down the data into long format for ggplot2  
sim\_table\_long <- melt(sim\_table,   
 id = "sample\_size")  
  
##create a graph of the sample size by SE value  
ggplot(sim\_table\_long, aes(sample\_size, value)) +   
 theme\_classic() +  
 xlab("Sample Size") +  
 ylab("Standard Error") +   
 geom\_point() +   
 geom\_hline(yintercept = .12) #mark here .12 occurs



At what point is 80% of the data below .12?

##calculate the percent below .12  
sim\_table\_long %>% #data frame  
 group\_by(sample\_size) %>% #group by sample size  
 summarize(Percent\_Below = sum(value<=.12)) %>% #is it less than .12  
 print(n = nrow(.))

## # A tibble: 40 x 2  
## sample\_size Percent\_Below  
## <dbl> <int>  
## 1 5 18  
## 2 10 18  
## 3 15 33  
## 4 20 40  
## 5 25 45  
## 6 30 57  
## 7 35 61  
## 8 40 65  
## 9 45 70  
## 10 50 72  
## 11 55 75  
## 12 60 76  
## 13 65 82  
## 14 70 76  
## 15 75 85  
## 16 80 88  
## 17 85 88  
## 18 90 89  
## 19 95 93  
## 20 100 89  
## 21 105 92  
## 22 110 95  
## 23 115 93  
## 24 120 92  
## 25 125 97  
## 26 130 93  
## 27 135 96  
## 28 140 98  
## 29 145 96  
## 30 150 98  
## 31 155 97  
## 32 160 97  
## 33 165 96  
## 34 170 97  
## 35 175 99  
## 36 180 99  
## 37 185 98  
## 38 190 98  
## 39 195 99  
## 40 200 99

Looks like the answer is ~ 50 give or take different variations of this random sampling. This estimate would be the minimum sample size per word.

*Note*: I took several runs of this simulation and the one below for the estimates listed - they may not perfectly match the current compiled version of this document, but are representative of the larger set of values I estimated.

## Examining The Semantic Priming Project (SPP)

In the SPP, participants were given a lexical decision task with a priming cue word first. So, the task is the same as the ELP, however, they first saw a prime word, then made the lexical decision on the target word. The sample size of target words is 1661. We are using the already z-scored data for the response latencies. In the SPP, they provide an item level analysis of the average z-score priming (i.e., average z-score for the target word in the related minus unrelated condition). However, that data does not allow you to estimate when the priming estimate would be stable, as it’s just one value for each prime-target pair. As mentioned in the full proposal, we would expect priming to be variable - it should be predicted by other psycholinguistic variables. Therefore, we should aim to create stable estimates for the z-scored response latencies in both the related and unrelated conditions. This aim would allow us to know that at least the response latencies are reliable, and variability in the final subtracted priming can be investigated for predictors.

##read in the SPP data  
SPPmaster <- read.csv("subjectdataLDT.csv")  
  
##drop the nonwords and non accurate, this has already been z scored  
SPPcorrect <- subset(SPPmaster, #data frame  
 target.ACC > 0 & lexicality == 1) #make sure it's accurate and lexicality = 1 are real words   
  
##drop all unused stimuli and words  
SPPcorrect$target <- droplevels(SPPcorrect$target)  
  
#remove NAs from the Z target  
SPPcorrect <- subset(SPPcorrect, #data frame  
 !is.na(Ztarget.RT)) #is not NA, only keep non-NA values

What is the average SE for our standardized response latencies for words in a priming task (rather than no priming lexical decision)?

##summarize the dataframe to see what the average SE is  
summary\_stats <- SPPcorrect %>% #data frame  
 select(Ztarget.RT, target) %>% #pick the columns  
 group\_by(target) %>% #put together the stimuli  
 summarize(SES = sd(Ztarget.RT)/sqrt(length(Ztarget.RT)), samplesize = length(Ztarget.RT)) #create SE and the sample size for below   
  
##give descriptives of the SEs  
psych::describe(summary\_stats$SES)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 1661 0.06 0.01 0.06 0.06 0.01 0.04 0.16 0.12 2.19 11.4  
## se  
## X1 0

In this study, they used a smaller subset of words (1661) as compared to the much larger set of English in ELP (40,000+). These words are likely to be similar to the words chosen for the study - because they are mostly somewhat frequent nouns, the variability in response latency is less than above. Here, we see it’s about 0.06 for the standard error.

What is the average sample size after data loss due to incorrect answers?

##figure out the original sample sizes  
original\_SS <- SPPmaster %>% #data frame  
 count(target) #count up the sample size  
  
##add the original sample size to the data frame  
summary\_stats <- merge(summary\_stats, original\_SS, by = "target")  
  
##original sample size average  
psych::describe(summary\_stats$n)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 1661 255.2 1.36 255 255.26 1.48 251 258 7 -0.32 -0.15  
## se  
## X1 0.03

##reduced sample size  
psych::describe(summary\_stats$samplesize)

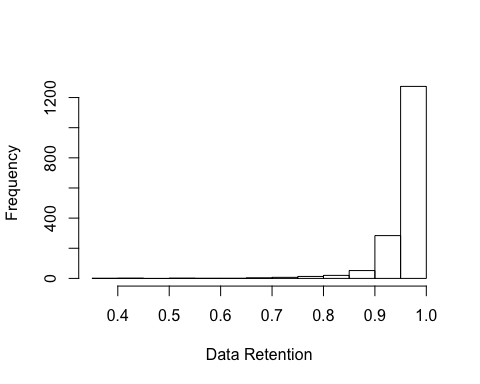
## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 1661 244.23 12.64 247 246.51 4.45 101 257 156 -5.34 41.78  
## se  
## X1 0.31

##percent retained  
psych::describe(summary\_stats$samplesize/summary\_stats$n)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 1661 0.96 0.05 0.97 0.97 0.02 0.39 1 0.61 -5.43 42.82  
## se  
## X1 0

The original sample sizes are approximately ~256 participants, which is *n* = 32 for each of the eight possible conditions in the study. We are not going to use those conditions, so the entire data was collapsed for this analysis. The data retention is much better in this analysis at around 96%-97%, likely because the dataset includes fewer “weird” words.

##show the retention rates  
hist(summary\_stats$samplesize/summary\_stats$n,  
 xlab = "Data Retention",  
 main = "")

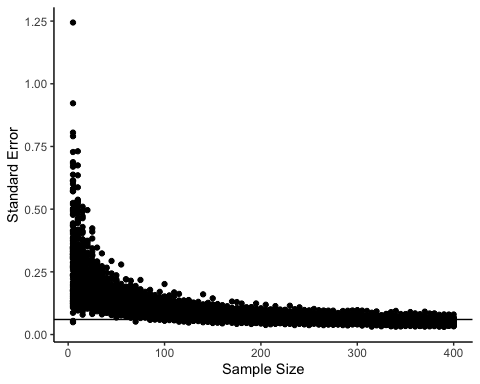


If I assume these data to be representative, what actual sample size would result in a SE = 0.06?

##pick 100 random words as all sample sizes are above 30  
targets <- summary\_stats %>% #data frame  
 select(target) %>% #select only stimuli  
 sample\_n(100) %>% #get 100  
 pull(target) #make it a vector  
targets <- as.character(targets)  
  
##this section creates a sequence of sample sizes to estimate at  
#5, 10, 15, etc.   
samplesize\_values <- seq(5, 400, 5)   
#create a blank table for us to save the values in   
sim\_table <- matrix(NA, nrow = length(samplesize\_values), ncol = length(targets))  
#create column names based on the current targets  
colnames(sim\_table) <- targets  
#make it a data frame  
sim\_table <- as.data.frame(sim\_table)  
#add those sample size values   
sim\_table$sample\_size <- samplesize\_values  
  
##loop over all the target words randomly selected  
for (i in 1:length(targets)){  
   
 ##loop over sample sizes  
 for (q in 1:length(samplesize\_values)){  
   
 ##temporarily save a data frame of Zscores   
 temp <- SPPcorrect %>% #data frame  
 filter(target == targets[i]) %>% #pick rows that are the current target word  
 sample\_n(samplesize\_values[q], replace = T) %>% #select sample size number of rows   
 pull(Ztarget.RT)  
   
 #put that in the table  
 #find the sample size row and column we are working with  
 #calculate SE sd/sqrt(n)  
 sim\_table[sim\_table$sample\_size == samplesize\_values[q], targets[i]] <- sd(temp)/sqrt(length(temp))  
   
 }  
   
 }

A graph of this data:

##melt down the data into long format for ggplot2  
sim\_table\_long2 <- melt(sim\_table,   
 id = "sample\_size")  
  
##create a graph of the sample size by SE value  
ggplot(sim\_table\_long2, aes(sample\_size, value)) +   
 theme\_classic() +  
 xlab("Sample Size") +  
 ylab("Standard Error") +   
 geom\_point() +   
 geom\_hline(yintercept = .06) #mark here .06 occurs



At what point is 80% of the data below .06?

##calculate the percent below .06  
sim\_table\_long2 %>% #data frame  
 group\_by(sample\_size) %>% #group by sample size  
 summarize(Percent\_Below = sum(value<=.06)) %>% #is it less than .06  
 print(n = nrow(.))

## # A tibble: 80 x 2  
## sample\_size Percent\_Below  
## <dbl> <int>  
## 1 5 2  
## 2 10 0  
## 3 15 0  
## 4 20 0  
## 5 25 0  
## 6 30 0  
## 7 35 0  
## 8 40 0  
## 9 45 0  
## 10 50 0  
## 11 55 0  
## 12 60 0  
## 13 65 0  
## 14 70 1  
## 15 75 0  
## 16 80 0  
## 17 85 2  
## 18 90 1  
## 19 95 1  
## 20 100 2  
## 21 105 2  
## 22 110 2  
## 23 115 1  
## 24 120 3  
## 25 125 3  
## 26 130 8  
## 27 135 5  
## 28 140 5  
## 29 145 9  
## 30 150 9  
## 31 155 10  
## 32 160 8  
## 33 165 14  
## 34 170 17  
## 35 175 13  
## 36 180 20  
## 37 185 20  
## 38 190 27  
## 39 195 22  
## 40 200 22  
## 41 205 29  
## 42 210 34  
## 43 215 28  
## 44 220 30  
## 45 225 42  
## 46 230 39  
## 47 235 40  
## 48 240 50  
## 49 245 48  
## 50 250 49  
## 51 255 57  
## 52 260 55  
## 53 265 52  
## 54 270 51  
## 55 275 60  
## 56 280 69  
## 57 285 64  
## 58 290 67  
## 59 295 66  
## 60 300 68  
## 61 305 72  
## 62 310 74  
## 63 315 77  
## 64 320 75  
## 65 325 75  
## 66 330 78  
## 67 335 80  
## 68 340 79  
## 69 345 85  
## 70 350 80  
## 71 355 82  
## 72 360 82  
## 73 365 83  
## 74 370 85  
## 75 375 89  
## 76 380 90  
## 77 385 89  
## 78 390 89  
## 79 395 90  
## 80 400 86

Here the required number of participants per word would be ~320 participants.

## Summary and Suggestions

In each session, participants would judge multiple words. In the SPP, each person judged 800 words per session, while in the ELP included 1,200 words per session. This facet should be considering for timing of the experiment, especially fatigue.

Estimation formulas:

##how many words per session  
##go a little less since it's a boring task  
words\_per\_session <- 600  
  
##words are assigned 25% related, 25% unrelated, 50% nonwords  
##this keeps relatedness to 50/50 for real words, which is what SPP did  
##also keeps yes/no lexical decision to 50/50  
##also remember you will rate the prime word but it doesn't count  
usable\_words\_per\_session <- words\_per\_session \* .50 / 2   
  
##each word has to be collected in both unrelated and related conditions  
conditions <- 2  
  
##estimated participants from above   
lower\_est <- 50  
upper\_est <- 320  
  
##data loss conservative estimate from ELP, since online studies may have more  
data\_loss <- .9   
  
##target word goal  
#number of targets we wish to achieve  
number\_of\_targets <- 1000  
  
##total estimated participants  
((1/data\_loss) \* #incorporate data loss  
 lower\_est \* #number of participants needed for each word  
 conditions \* #number of conditions each word has to appear in   
 number\_of\_targets) / #number of total words  
 usable\_words\_per\_session

## [1] 740.7407

##total estimated participants  
((1/data\_loss) \* #incorporate data loss  
 upper\_est \* #number of participants needed for each word  
 conditions \* #number of conditions each word has to appear in   
 number\_of\_targets) / #number of total words  
 usable\_words\_per\_session

## [1] 4740.741

The formula works as follows: We will incorporate expected data loss by multiplying by a percent increase one would need to accomodate that loss. This score is then multiplied by the estimates of persons per word for accuracy in parameter estimation. Each word must be seen in the related and unrelated condition, and these are not repeated within-subjects, therefore, we will double the estimate for the two conditions. This number is then multiplied by a desired number of target words, and 1000 words is the goal for this study. That value is divided by the useable number of words per session from a participant. In priming studies, you need to control for relatedness proprotions by keeping a balance of unrelated and related target words, as well as the balance of yes/no answers for the lexical decision task. Therefore, they are allocated at 25% for each of the real words (related/unrelated) and 50% for non-words. Therefore, each participant only provides 50% useable words, which is then further divided by two to only capture target words (i.e., ignoring prime words).

The estimates indicate that between 741 and 4741 participants would be necessary to gather 1000 real word targets in related and unrelated conditions for the study. This value would be the target sample size for each of the languages in the study.

## Stopping Procedure

Because the variability of the sample size is quite large, we will employ a stopping procedure to ensure participant time and effort is maximized, and data collection is minimized. The minimum sample size will be 50 participants per concept or 741 total participants. After 50 participants, each concept will be examined for standard error, and data collection for that concept will be stopped when the standard error reaches an average of the two metrics found in this exploration (0.06, 0.12) or 0.09. This process will be automated online and checked in a daily subroutine, and words that meet the stopping rule criteria will be removed from further data collection. From the current simulations, this approximates to 100 to 150 participants per word, and 1482 to 2223 participants per language total. The maximum number of participants per word will be *n* = 320 from estimations above.

##calculate the percent below .09  
sim\_table\_long %>% #data frame  
 group\_by(sample\_size) %>% #group by sample size  
 summarize(Percent\_Below = sum(value<=.09)) %>% #is it less than .09  
 print(n = nrow(.))

## # A tibble: 40 x 2  
## sample\_size Percent\_Below  
## <dbl> <int>  
## 1 5 12  
## 2 10 0  
## 3 15 14  
## 4 20 18  
## 5 25 20  
## 6 30 34  
## 7 35 38  
## 8 40 44  
## 9 45 51  
## 10 50 50  
## 11 55 52  
## 12 60 61  
## 13 65 57  
## 14 70 62  
## 15 75 66  
## 16 80 72  
## 17 85 68  
## 18 90 74  
## 19 95 76  
## 20 100 75  
## 21 105 79  
## 22 110 81  
## 23 115 79  
## 24 120 80  
## 25 125 81  
## 26 130 85  
## 27 135 84  
## 28 140 83  
## 29 145 83  
## 30 150 87  
## 31 155 87  
## 32 160 87  
## 33 165 87  
## 34 170 90  
## 35 175 90  
## 36 180 90  
## 37 185 92  
## 38 190 94  
## 39 195 95  
## 40 200 94

##calculate the percent below .09  
sim\_table\_long2 %>% #data frame  
 group\_by(sample\_size) %>% #group by sample size  
 summarize(Percent\_Below = sum(value<=.09)) %>% #is it less than .09  
 print(n = nrow(.))

## # A tibble: 80 x 2  
## sample\_size Percent\_Below  
## <dbl> <int>  
## 1 5 3  
## 2 10 0  
## 3 15 1  
## 4 20 0  
## 5 25 2  
## 6 30 0  
## 7 35 4  
## 8 40 2  
## 9 45 8  
## 10 50 8  
## 11 55 12  
## 12 60 7  
## 13 65 15  
## 14 70 14  
## 15 75 18  
## 16 80 22  
## 17 85 28  
## 18 90 30  
## 19 95 37  
## 20 100 45  
## 21 105 50  
## 22 110 48  
## 23 115 49  
## 24 120 56  
## 25 125 59  
## 26 130 66  
## 27 135 75  
## 28 140 76  
## 29 145 77  
## 30 150 82  
## 31 155 80  
## 32 160 86  
## 33 165 84  
## 34 170 84  
## 35 175 88  
## 36 180 89  
## 37 185 91  
## 38 190 91  
## 39 195 89  
## 40 200 94  
## 41 205 95  
## 42 210 95  
## 43 215 97  
## 44 220 98  
## 45 225 97  
## 46 230 98  
## 47 235 98  
## 48 240 98  
## 49 245 98  
## 50 250 99  
## 51 255 99  
## 52 260 99  
## 53 265 99  
## 54 270 99  
## 55 275 99  
## 56 280 100  
## 57 285 99  
## 58 290 99  
## 59 295 99  
## 60 300 99  
## 61 305 99  
## 62 310 99  
## 63 315 100  
## 64 320 99  
## 65 325 100  
## 66 330 100  
## 67 335 100  
## 68 340 99  
## 69 345 100  
## 70 350 100  
## 71 355 100  
## 72 360 100  
## 73 365 100  
## 74 370 99  
## 75 375 100  
## 76 380 100  
## 77 385 100  
## 78 390 100  
## 79 395 100  
## 80 400 100

##how many words per session  
##go a little less since it's a boring task  
words\_per\_session <- 600  
  
##words are assigned 25% related, 25% unrelated, 50% nonwords  
##this keeps relatedness to 50/50 for real words, which is what SPP did  
##also keeps yes/no lexical decision to 50/50  
##also remember you will rate the prime word but it doesn't count  
usable\_words\_per\_session <- words\_per\_session \* .50 / 2   
  
##each word has to collected in both unrelated and related conditions  
conditions <- 2  
  
##estimated participants from above   
lower\_est <- 100  
upper\_est <- 150  
  
##data loss conservative estimate from ELP, since online studies may have more  
data\_loss <- .9   
  
##target word goal  
#number of targets we wish to achieve  
number\_of\_targets <- 1000  
  
##total estimated participants  
((1/data\_loss) \* #incorporate data loss  
 lower\_est \* #number of participants needed for each word  
 conditions \* #number of conditions each word has to appear in   
 number\_of\_targets) / #number of total words  
 usable\_words\_per\_session

## [1] 1481.481

##total estimated participants  
((1/data\_loss) \* #incorporate data loss  
 upper\_est \* #number of participants needed for each word  
 conditions \* #number of conditions each word has to appear in   
 number\_of\_targets) / #number of total words  
 usable\_words\_per\_session

## [1] 2222.222