Power using Accuracy in Parameter Estimation

Erin M. Buchanan

Last Updated: 2021-08-25

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).

```r
#read in the ELP data
ELPmaster <- read.csv("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.

```r
#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)
#read.csv doesn't import as factors anymore
```

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

```r
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
```

```r
##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 se
## X1    1 40455 0.16 0.1   0.14    0.15 0.06 0.01 3.33  3.32 4.71    65.45  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.

```r
##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 se
```

```
## X1    1 40472 32.69 0.63     33   32.74   0  29  35     6 -0.91     1.45  0
```

```
##reduced sample size
psych::describe(summary_stats$samplesize)
```

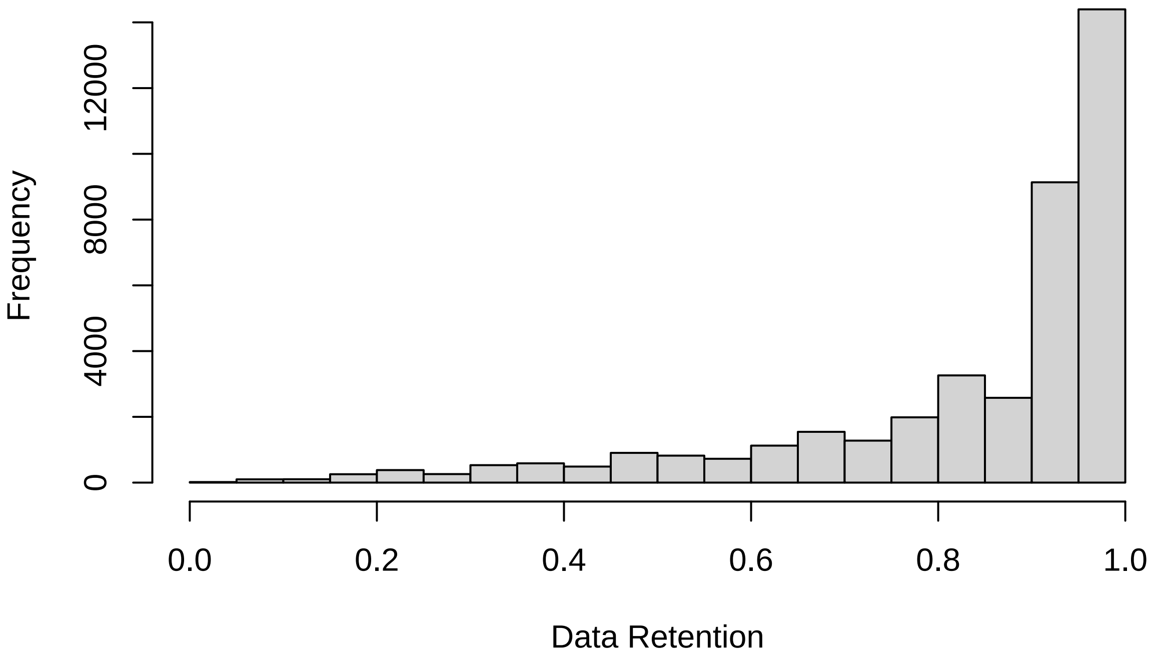
```
##    vars     n  mean   sd median trimmed  mad min max range  skew kurtosis   se
## X1    1 40472 27.41 6.43     30   28.64 2.97   1  35    34 -1.66      2.2 0.03
```

```
##percent retained
psych::describe(summary_stats$samplesize/summary_stats$n)
```

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

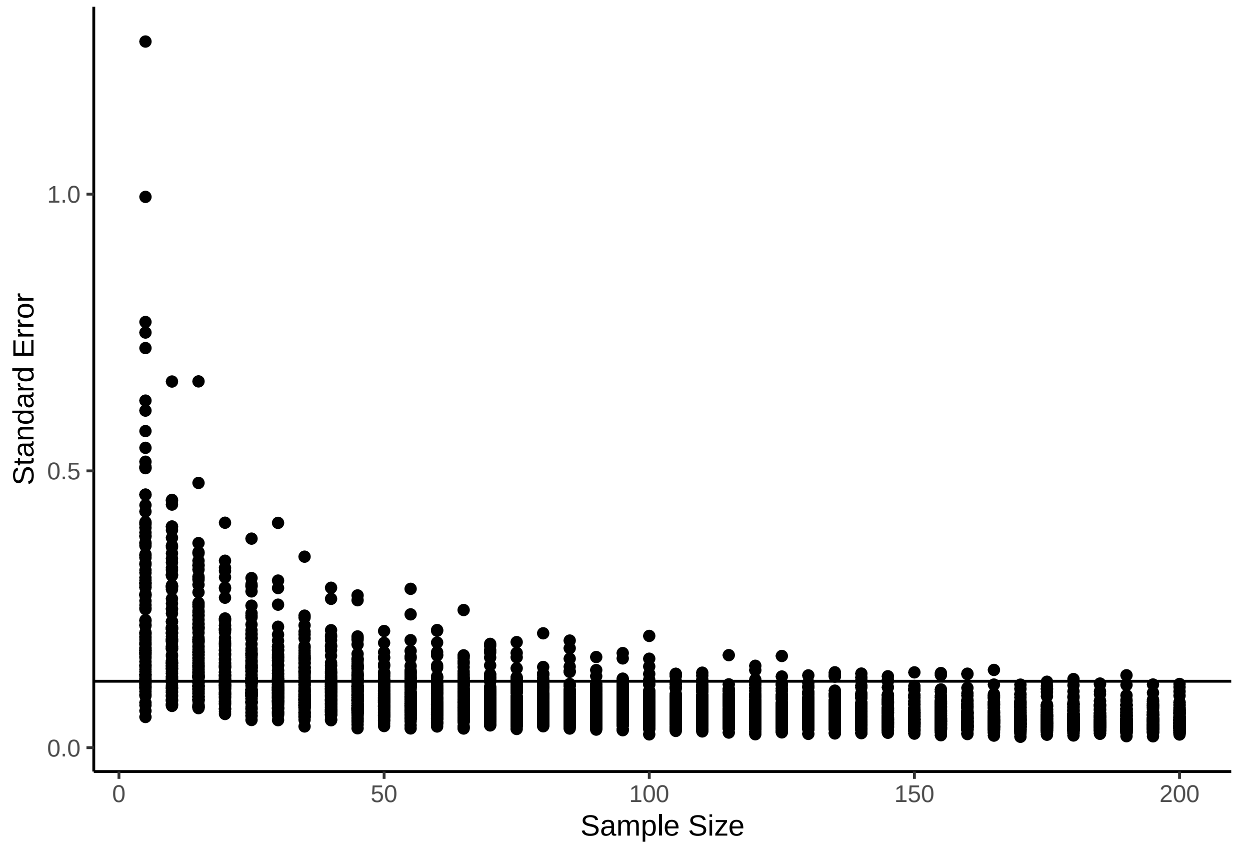
If I assume these data to be representative, what actual sample size might approximate SE = 0.12?

```
##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)
```

```r
##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:

```r
##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            14
##  2          10            22
##  3          15            33
##  4          20            38
##  5          25            49
##  6          30            66
##  7          35            64
##  8          40            75
##  9          45            70
## 10          50            82
## 11          55            83
## 12          60            87
## 13          65            86
## 14          70            88
## 15          75            91
## 16          80            92
## 17          85            94
## 18          90            97
## 19          95            96
## 20         100            95
## 21         105            96
## 22         110            97
## 23         115            99
## 24         120            97
## 25         125            98
## 26         130            99
## 27         135            96
## 28         140            97
## 29         145            97
## 30         150            99
```

```
## 31        155          98
## 32        160          98
## 33        165          99
## 34        170         100
## 35        175         100
## 36        180          99
## 37        185         100
## 38        190          99
## 39        195         100
## 40        200         100
```

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.

```r
##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)?

```r
##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 se
## X1    1 1661 0.06 0.01   0.06    0.06 0.01 0.04 0.16  0.12 2.19     11.4  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?

```r
##figure out the original sample sizes
original_SS <- SPPmaster %>% #data frame
  count(target) #count up the sample size
```

```r
##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   se
## X1    1 1661 255.2 1.36    255  255.26 1.48 251 258     7 -0.32    -0.15 0.03
```

```r
##reduced sample size
psych::describe(summary_stats$samplesize)
```

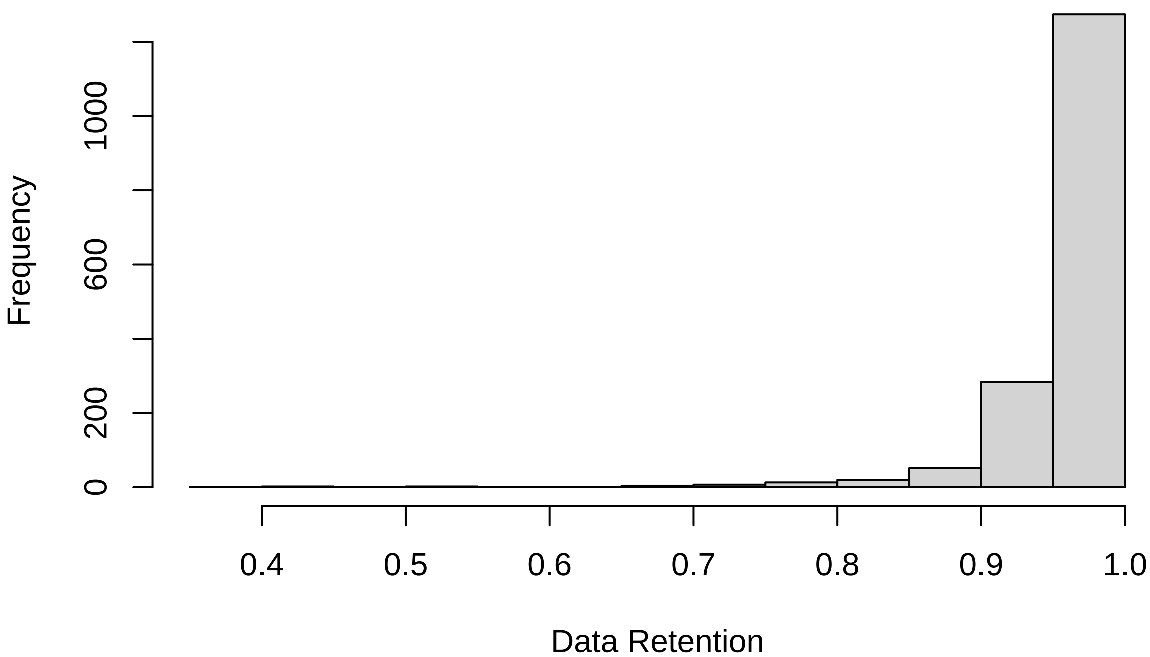
```
##    vars    n   mean    sd median trimmed  mad min max range  skew kurtosis   se
## X1    1 1661 244.23 12.64    247  246.51 4.45 101 257   156 -5.34    41.78 0.31
```

```r
##percent retained
psych::describe(summary_stats$samplesize/summary_stats$n)
```

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

```r
##show the retention rates
hist(summary_stats$samplesize/summary_stats$n,
     xlab = "Data Retention",
     main = "")
```



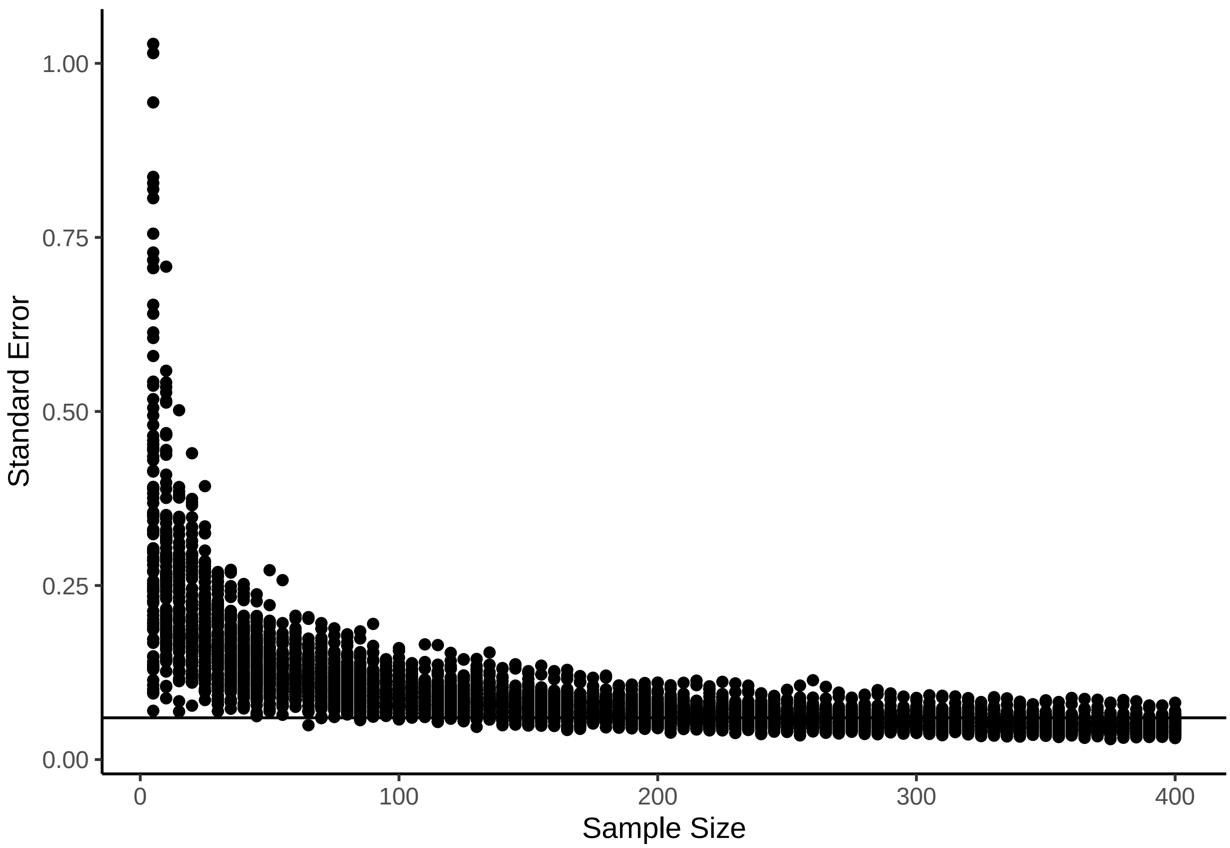
If I assume these data to be representative, what actual sample size would result in a SE = 0.06?

```r
##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)
```

```r
#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:

```r
##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?

```r
##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             0
##  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             1
## 14          70             1
## 15          75             0
## 16          80             0
## 17          85             1
## 18          90             0
## 19          95             0
## 20         100             1
## 21         105             0
## 22         110             0
## 23         115             3
## 24         120             2
## 25         125             3
## 26         130             4
## 27         135             5
## 28         140             9
## 29         145            10
## 30         150            12
## 31         155            14
## 32         160            11
## 33         165            13
## 34         170            19
## 35         175            14
## 36         180            18
## 37         185            28
## 38         190            28
## 39         195            31
## 40         200            26
## 41         205            35
## 42         210            41
## 43         215            37
## 44         220            40
## 45         225            38
## 46         230            37
## 47         235            48
## 48         240            46
## 49         245            56
## 50         250            60
## 51         255            59
## 52         260            59
## 53         265            59
## 54         270            66
## 55         275            63
## 56         280            69
## 57         285            71
## 58         290            68
## 59         295            69
## 60         300            73
## 61         305            71
## 62         310            78
## 63         315            79
## 64         320            78
## 65         325            86
## 66         330            81
## 67         335            84
## 68         340            88
## 69         345            85
## 70         350            85
## 71         355            87
```

```
## 72          360            89
## 73          365            92
## 74          370            88
## 75          375            93
## 76          380            92
## 77          385            93
## 78          390            94
## 79          395            94
## 80          400            92
```

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             4
##  2          10             8
##  3          15            12
##  4          20            11
##  5          25            16
##  6          30            26
##  7          35            38
##  8          40            41
##  9          45            43
## 10          50            53
## 11          55            55
## 12          60            66
## 13          65            65
## 14          70            69
## 15          75            69
## 16          80            70
## 17          85            76
## 18          90            79
## 19          95            78
## 20         100            82
## 21         105            87
## 22         110            84
## 23         115            87
## 24         120            88
## 25         125            88
## 26         130            90
## 27         135            91
## 28         140            90
## 29         145            93
## 30         150            91
## 31         155            94
## 32         160            95
## 33         165            93
## 34         170            95
## 35         175            94
## 36         180            95
## 37         185            96
## 38         190            97
## 39         195            98
## 40         200            96
```

```
##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   1
## 2    10   1
## 3    15   2
## 4    20   1
## 5    25   1
## 6    30   4
## 7    35   3
## 8    40   6
## 9    45   8
## 10   50  10
## 11   55  12
## 12   60   8
## 13   65  11
## 14   70  25
## 15   75  27
## 16   80  23
## 17   85  31
## 18   90  32
## 19   95  49
## 20  100  52
## 21  105  47
## 22  110  58
## 23  115  58
## 24  120  65
## 25  125  69
## 26  130  72
## 27  135  72
## 28  140  77
## 29  145  85
## 30  150  84
## 31  155  82
## 32  160  83
## 33  165  86
## 34  170  89
## 35  175  92
## 36  180  92
## 37  185  90
## 38  190  90
## 39  195  92
## 40  200  95
## 41  205  96
## 42  210  98
## 43  215  98
## 44  220  95
## 45  225  97
## 46  230  97
## 47  235  97
## 48  240  98
## 49  245  98
## 50  250  98
## 51  255  99
## 52  260  99
## 53  265  99
## 54  270  98
## 55  275 100
## 56  280  98
## 57  285  98
## 58  290  99
## 59  295  98
## 60  300 100
## 61  305  99
## 62  310  99
## 63  315  99
## 64  320 100
## 65  325 100
## 66  330 100
## 67  335 100
## 68  340 100
## 69  345 100
## 70  350 100
## 71  355 100
## 72  360 100
## 73  365 100
## 74  370 100
## 75  375 100
## 76  380 100
## 77  385 100
## 78  390 100
## 79  395 100
## 80  400 100
```

```r
##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
```

```r
##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
```