Italian Age of Acquisition Norms for a Large Set of Words (ItAoA)

Mahmoud M.Elsherif

2023-03-28

### Vignette Setup:

knitr::opts\_chunk$set(echo = TRUE)  
  
# Set a random seed  
set.seed(3898934)  
  
# Libraries necessary for this vignette  
library(rio)  
library(flextable)  
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(tidyr)  
library(psych)  
  
# Function for simulation  
item\_power <- function(data, # name of data frame  
 dv\_col, # name of DV column as a character  
 item\_col, # number of items column as a character  
 sample\_start = 20,   
 sample\_stop = 200,   
 sample\_increase = 5,  
 decile = .5){  
   
 DF <- cbind.data.frame(  
 "dv" = data[ , dv\_col],  
 "items" = data[ , item\_col]  
 )  
   
 # just in case  
 colnames(DF) <- c("dv", "items")  
   
 # figure out the "sufficiently narrow" ci value  
 SE <- tapply(DF$dv, DF$items, function (x) { sd(x)/sqrt(length(x)) })  
 cutoff <- quantile(SE, probs = decile)  
   
 # sequence of sample sizes to try  
 samplesize\_values <- seq(sample\_start, sample\_stop, sample\_increase)  
  
 # create a blank table for us to save the values in   
 sim\_table <- matrix(NA,   
 nrow = length(samplesize\_values),   
 ncol = length(unique(DF$items)))  
  
 # make it a data frame  
 sim\_table <- as.data.frame(sim\_table)  
  
 # add a place for sample size values   
 sim\_table$sample\_size <- NA  
  
 # loop over sample sizes  
 for (i in 1:length(samplesize\_values)){  
   
 # temp that samples and summarizes  
 temp <- DF %>%   
 group\_by(items) %>%   
 sample\_n(samplesize\_values[i], replace = T) %>%   
 summarize(se = sd(dv)/sqrt(length(dv)))  
   
 # dv on items  
 colnames(sim\_table)[1:length(unique(DF$items))] <- temp$items  
 sim\_table[i, 1:length(unique(DF$items))] <- temp$se  
 sim\_table[i, "sample\_size"] <- samplesize\_values[i]  
   
 }  
  
 # figure out cut off  
 final\_sample <- sim\_table %>%   
 pivot\_longer(cols = -c(sample\_size)) %>%   
 rename(item = name, se = value) %>%   
 group\_by(sample\_size) %>%   
 summarize(Percent\_Below = sum(se <= cutoff)/length(unique(DF$items)))   
   
 # multiply by correction   
 final\_sample$new\_sample <- round(39.369 + 0.700\*final\_sample$sample\_size + 0.003\*cutoff - 0.694\*length(unique(DF$items)))  
   
 return(list(  
 SE = SE,   
 cutoff = cutoff,   
 DF = DF,   
 sim\_table = sim\_table,   
 final\_sample = final\_sample  
 ))  
  
}

### Project/Data Title:

Italian Age of Acquisition Norms for a Large Set of Words (ItAoA)

Data provided by: Ettore Ambrosini

### Project/Data Description:

Age of acquisition (AoA) represents the age at which a word is learned. This measure has been shown to affect performance in a large variety of cognitive tasks (see reviews by Juhasz, 2005; Johnston and Barry, 2006; Brysbaert and Ellis, 2016), with faster reaction times for words learned early in life compared with those learned later.

There are two main approaches to derive AoA data. First, objective AoA measures can be obtained by analysis of children’s production (Chalard et al., 2003; Álvarez and Cuetos, 2007; Lotto et al., 2010; Grigoriev and Oshhepkov, 2013). Within this approach, children (classified by age) are asked to name the picture of common objects and activities. The AoA of a given word is computed as the mean age of the group of children in which at least 75% of them can name the picture correctly. Alternatively, subjective AoA can be obtained by using adult estimates (Barca et al., 2002; Ferrand et al., 2008; Moors et al., 2013). Here, adult participants are asked to provide ratings of AoA on either a Likert scale (Schock et al., 2012; Alonso et al., 2015; Borelli et al., 2018) or directly in years, by indicating the number corresponding to the age they thought they had learned a given word (Stadthagen-Gonzalez and Davis, 2006; Ferrand et al., 2008; Moors et al., 2013). Compared to the use of a Likert scale, this latter method is easier for participants to use and it does not restrict the response range artificially, instead providing more precise information about the words’ AoA (Ghyselinck et al., 2000). It has been shown that the AoA estimates obtained from the two different methods are highly correlated (Morrison et al., 1997; Ghyselinck et al., 2000; Pind et al., 2000; Lotto et al., 2010; see also Brysbaert, 2017; Brysbaert and Biemiller, 2017) and this correlation still remains significant when other variables, such as familiarity, frequency, and phonological length, are controlled for (Bonin et al., 2004).

Only two sets of Italian norms with objective AoA (Rinaldi et al., 2004) and subjective AoA (Borelli et al., 2018) include abstract and concrete words and different word classes (adjective, noun, and verb), but they are limited to a relatively small number of word stimuli (519 and 512 words, respectively). Unfortunately, the lack of overlap between AoA (Dell’Acqua et al., 2000; Barca et al., 2002; Barbarotto et al., 2005; Della Rosa et al., 2010; Borelli et al., 2018) and semantic-affective norms (Zannino et al., 2006; Kremer and Baroni, 2011; Montefinese et al., 2013b, 2014; Fairfield et al., 2017) for Italian words has prevented the direct comparison of different lexical-semantic dimensions to establish the extent to which they overlap or complement each other in word processing. An important motivation of the present study is to extend previous Italian norms by collecting AoA ratings for a much larger range of Italian words for which concreteness and semantic-affective norms are now available thus ensuring greater coverage of words varying along these dimensions.

### Methods Description:

A total of 507 native Italian speakers were enrolled to participate in an online study (436 females and 81 males; mean age: 20.82 years, SD = 2.22; mean education: 15.16 years, SD = 1.11). We selected 1,957 Italian words from our Italian adaptations of the original ANEW (Montefinese et al., 2014; Fairfield et al., 2017) and from available Italian semantic norms (Zannino et al., 2006; Kremer and Baroni, 2011; Montefinese et al., 2013). The set of stimuli included 76% of nouns, 16% of adjectives, and 8% of verbs. The word stimuli were presented in the same verbal form as the previous Italian norms (e.g., the verbs were presented in the infinitive form) to preserve the consistency with these data collections (Montefinese et al., 2014; Fairfield et al., 2017). Word stimuli were distributed over 20 lists containing 97–98 words each. In order to avoid primacy or recency effects, the order in which words appeared in the list was randomized for each participant separately. All lists were roughly matched for word length, word frequency, number of orthographic neighbors, and mean frequency of orthographic neighbors. For each list, an online form was created using Google modules. Participants were asked to estimate the age (in years) at which they thought they had learned the word, specifying that this information should indicate the age at which, for the first time they understood the word when somebody else used it in their presence, even when they did not use the word themselves. These instructions and the examples provided to the participants closely matched those used in a large number of previous studies (Ghyselinck et al., 2000; Stadthagen-Gonzalez and Davis, 2006; Kuperman et al., 2012; Moors et al., 2013; Łuniewska et al., 2016). The task lasted about 40 min.

### Data Location:

Included with the vignette and Data Location: <https://osf.io/rzycf/>

DF <- import("ambrosini\_data.csv")  
  
DF <- DF %>%  
 arrange(Eng\_Word) %>% #orders the rows of the data by the target\_name column  
 group\_by(Eng\_Word) %>% #group by the target name  
 transform(items = as.numeric(factor(Eng\_Word)))%>% #transform target name into a item  
 select(items, Eng\_Word, Ita\_Word, everything()  
 ) #select all variables from items and target\_name   
  
DF <- DF %>%   
 group\_by(Eng\_Word) %>%  
 filter (Rating != 'Unknown')  
  
str(DF)

## gropd\_df [48,772 × 5] (S3: grouped\_df/tbl\_df/tbl/data.frame)  
## $ items : num [1:48772] 334 334 334 334 334 334 334 334 334 334 ...  
## $ Eng\_Word: chr [1:48772] "Colt" "Colt" "Colt" "Colt" ...  
## $ Ita\_Word: chr [1:48772] "rivoltella" "rivoltella" "rivoltella" "rivoltella" ...  
## $ SS\_ID : int [1:48772] 104 105 106 107 108 109 110 111 113 114 ...  
## $ Rating : chr [1:48772] "19" "14" "14" "10" ...  
## - attr(\*, "groups")= tibble [1,904 × 2] (S3: tbl\_df/tbl/data.frame)  
## ..$ Eng\_Word: chr [1:1904] "Colt" "Epiphany" "abortion" "absurd" ...  
## ..$ .rows : list<int> [1:1904]   
## .. ..$ : int [1:23] 1 2 3 4 5 6 7 8 9 10 ...  
## .. ..$ : int [1:25] 24 25 26 27 28 29 30 31 32 33 ...  
## .. ..$ : int [1:25] 49 50 51 52 53 54 55 56 57 58 ...  
## .. ..$ : int [1:25] 74 75 76 77 78 79 80 81 82 83 ...  
## .. ..$ : int [1:25] 99 100 101 102 103 104 105 106 107 108 ...  
## .. ..$ : int [1:25] 124 125 126 127 128 129 130 131 132 133 ...  
## .. ..$ : int [1:25] 149 150 151 152 153 154 155 156 157 158 ...  
## .. ..$ : int [1:25] 174 175 176 177 178 179 180 181 182 183 ...  
## .. ..$ : int [1:25] 199 200 201 202 203 204 205 206 207 208 ...  
## .. ..$ : int [1:25] 224 225 226 227 228 229 230 231 232 233 ...  
## .. ..$ : int [1:25] 249 250 251 252 253 254 255 256 257 258 ...  
## .. ..$ : int [1:25] 274 275 276 277 278 279 280 281 282 283 ...  
## .. ..$ : int [1:25] 299 300 301 302 303 304 305 306 307 308 ...  
## .. ..$ : int [1:50] 324 325 326 327 328 329 330 331 332 333 ...  
## .. ..$ : int [1:25] 374 375 376 377 378 379 380 381 382 383 ...  
## .. ..$ : int [1:25] 399 400 401 402 403 404 405 406 407 408 ...  
## .. ..$ : int [1:25] 424 425 426 427 428 429 430 431 432 433 ...  
## .. ..$ : int [1:25] 449 450 451 452 453 454 455 456 457 458 ...  
## .. ..$ : int [1:25] 474 475 476 477 478 479 480 481 482 483 ...  
## .. ..$ : int [1:25] 499 500 501 502 503 504 505 506 507 508 ...  
## .. ..$ : int [1:25] 524 525 526 527 528 529 530 531 532 533 ...  
## .. ..$ : int [1:25] 549 550 551 552 553 554 555 556 557 558 ...  
## .. ..$ : int [1:25] 574 575 576 577 578 579 580 581 582 583 ...  
## .. ..$ : int [1:25] 599 600 601 602 603 604 605 606 607 608 ...  
## .. ..$ : int [1:25] 624 625 626 627 628 629 630 631 632 633 ...  
## .. ..$ : int [1:25] 649 650 651 652 653 654 655 656 657 658 ...  
## .. ..$ : int [1:25] 674 675 676 677 678 679 680 681 682 683 ...  
## .. ..$ : int [1:25] 699 700 701 702 703 704 705 706 707 708 ...  
## .. ..$ : int [1:25] 724 725 726 727 728 729 730 731 732 733 ...  
## .. ..$ : int [1:25] 749 750 751 752 753 754 755 756 757 758 ...  
## .. ..$ : int [1:25] 774 775 776 777 778 779 780 781 782 783 ...  
## .. ..$ : int [1:25] 799 800 801 802 803 804 805 806 807 808 ...  
## .. ..$ : int [1:25] 824 825 826 827 828 829 830 831 832 833 ...  
## .. ..$ : int [1:25] 849 850 851 852 853 854 855 856 857 858 ...  
## .. ..$ : int [1:25] 874 875 876 877 878 879 880 881 882 883 ...  
## .. ..$ : int [1:25] 899 900 901 902 903 904 905 906 907 908 ...  
## .. ..$ : int [1:25] 924 925 926 927 928 929 930 931 932 933 ...  
## .. ..$ : int [1:25] 949 950 951 952 953 954 955 956 957 958 ...  
## .. ..$ : int [1:25] 974 975 976 977 978 979 980 981 982 983 ...  
## .. ..$ : int [1:25] 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 ...  
## .. ..$ : int [1:25] 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 ...  
## .. ..$ : int [1:25] 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 ...  
## .. ..$ : int [1:25] 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 ...  
## .. ..$ : int [1:24] 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 ...  
## .. ..$ : int [1:25] 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 ...  
## .. ..$ : int [1:25] 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 ...  
## .. ..$ : int [1:25] 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 ...  
## .. ..$ : int [1:25] 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 ...  
## .. ..$ : int [1:25] 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 ...  
## .. ..$ : int [1:25] 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 ...  
## .. ..$ : int [1:25] 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 ...  
## .. ..$ : int [1:25] 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 ...  
## .. ..$ : int [1:25] 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 ...  
## .. ..$ : int [1:25] 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 ...  
## .. ..$ : int [1:25] 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 ...  
## .. ..$ : int [1:25] 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 ...  
## .. ..$ : int [1:18] 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 ...  
## .. ..$ : int [1:25] 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 ...  
## .. ..$ : int [1:25] 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 ...  
## .. ..$ : int [1:25] 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 ...  
## .. ..$ : int [1:25] 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 ...  
## .. ..$ : int [1:25] 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 ...  
## .. ..$ : int [1:25] 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 ...  
## .. ..$ : int [1:25] 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 ...  
## .. ..$ : int [1:25] 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 ...  
## .. ..$ : int [1:25] 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 ...  
## .. ..$ : int [1:25] 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 ...  
## .. ..$ : int [1:25] 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 ...  
## .. ..$ : int [1:25] 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 ...  
## .. ..$ : int [1:24] 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 ...  
## .. ..$ : int [1:25] 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 ...  
## .. ..$ : int [1:25] 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 ...  
## .. ..$ : int [1:25] 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 ...  
## .. ..$ : int [1:25] 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 ...  
## .. ..$ : int [1:25] 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 ...  
## .. ..$ : int [1:25] 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 ...  
## .. ..$ : int [1:25] 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 ...  
## .. ..$ : int [1:25] 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 ...  
## .. ..$ : int [1:25] 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 ...  
## .. ..$ : int [1:25] 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 ...  
## .. ..$ : int [1:25] 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 ...  
## .. ..$ : int [1:25] 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 ...  
## .. ..$ : int [1:25] 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 ...  
## .. ..$ : int [1:25] 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 ...  
## .. ..$ : int [1:25] 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 ...  
## .. ..$ : int [1:25] 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 ...  
## .. ..$ : int [1:25] 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 ...  
## .. ..$ : int [1:25] 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 ...  
## .. ..$ : int [1:25] 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 ...  
## .. ..$ : int [1:25] 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 ...  
## .. ..$ : int [1:25] 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 ...  
## .. ..$ : int [1:25] 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 ...  
## .. ..$ : int [1:25] 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 ...  
## .. ..$ : int [1:25] 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 ...  
## .. ..$ : int [1:24] 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 ...  
## .. ..$ : int [1:25] 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 ...  
## .. ..$ : int [1:25] 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 ...  
## .. ..$ : int [1:25] 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 ...  
## .. ..$ : int [1:50] 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 ...  
## .. .. [list output truncated]  
## .. ..@ ptype: int(0)   
## ..- attr(\*, ".drop")= logi TRUE

### Date Published:

2019-02-13

### Dataset Citation:

Montefinese, M., Vinson, D., Vigliocco, G., & Ambrosini, E. (2018, November 26). Italian age of acquisition norms for a large set of words (ItAoA). <https://doi.org/10.17605/OSF.IO/3TRG2>

### Keywords:

age of acquisition, word, lexicon, Italian language, cross-linguistic comparison, subjective rating

### Use License:

CC-By Attribution 4.0 International

### Geographic Description - City/State/Country of Participants:

Italy

### Column Metadata:

metadata <- import("ambrosini\_metadata.xlsx")  
  
flextable(metadata) %>% autofit()

| Variable Name | Variable Description | Type (numeric, character, logical, etc.) |
| --- | --- | --- |
| items | Item number | Numeric |
| Eng\_Word | English translation of the item | Character |
| Ita\_Word | Italian translation of the item | Character |
| SS\_ID | Subject ID Number | Numeric |
| Rating | Age of acquisition rating | Numeric |

### AIPE Analysis:

Note that the data is already in long format (each item has one row), and therefore, we do not need to restructure the data.

#### Stopping Rule

In this dataset, we have 48772 individual words to select from for our research study. You would obviously not use all these within one study. Let’s say we wanted participants to rate 75 word pairs during our study (note: this selection is completely arbitrary).

random\_items <- unique(DF$items)[sample(unique(DF$items), size = 75)]  
  
DF <- DF %>%   
 filter(items %in% random\_items)  
  
# Function for simulation  
var1 <- item\_power(data = DF, # name of data frame  
 dv\_col = "Rating", # name of DV column as a character  
 item\_col = "items", # number of items column as a character  
 sample\_start = 20,   
 sample\_stop = 100,   
 sample\_increase = 5,  
 decile = .5)

What the usual standard error for the data that could be considered for our stopping rule using the 50% decile?

# individual SEs  
var1$SE

## 20 23 25 95 100 105 107 110   
## 0.3702252 0.4690416 0.4261455 0.2943920 0.5833238 0.3581434 0.4875791 0.3572114   
## 114 197 283 341 374 392 437 449   
## 0.4744119 0.4038892 0.4136021 0.5528110 0.6650313 0.4800000 0.6504358 0.3651484   
## 456 467 481 498 507 519 553 560   
## 0.4834598 0.2833137 0.5400000 0.4371117 0.6225753 0.4062840 0.6605049 0.3564641   
## 621 628 635 677 707 780 809 816   
## 0.3872983 0.4777726 0.5196152 0.6129709 0.4680456 0.5486954 0.6294972 0.5002666   
## 822 840 841 898 912 918 924 938   
## 0.3700450 0.5529316 0.4431704 0.4949074 0.6236452 0.4977282 1.0734990 0.4491000   
## 985 986 1008 1022 1034 1041 1062 1089   
## 0.5822943 0.6968501 0.3409790 0.3026549 0.5131601 0.3083288 0.3629509 0.5840091   
## 1096 1155 1187 1223 1227 1248 1256 1282   
## 0.3316625 0.3265986 0.5416026 0.4758151 0.4565085 0.4461689 0.6084954 0.5043808   
## 1341 1359 1395 1429 1435 1467 1469 1515   
## 0.5121198 0.3261901 0.5577335 0.6017389 0.2628054 0.4358899 0.3572114 0.4644710   
## 1522 1528 1538 1606 1617 1642 1650 1655   
## 0.2141142 0.3280244 0.2196968 0.3517575 0.3953058 0.2835489 0.3330666 0.3785939   
## 1660 1835 1865   
## 0.5461380 0.6380178 0.3553402

var1$cutoff

## 50%   
## 0.464471

Using our 50% decile as a guide, we find that 0.464 is our target standard error for an accurately measured item.

#### Minimum Sample Size

To estimate minimum sample size, we should figure out what number of participants it would take to achieve 80%, 85%, 90%, and 95% of the SEs for items below our critical score of 0.464?

flextable(var1$final\_sample %>%   
 filter(Percent\_Below >= .80) %>%   
 arrange(Percent\_Below, )) %>%   
 autofit()

| sample\_size | Percent\_Below | new\_sample |
| --- | --- | --- |
| 40 | 0.8933333 | 15 |
| 45 | 0.9333333 | 19 |
| 50 | 0.9333333 | 22 |
| 55 | 0.9466667 | 26 |
| 70 | 0.9733333 | 36 |
| 60 | 0.9866667 | 29 |
| 65 | 0.9866667 | 33 |
| 75 | 0.9866667 | 40 |
| 80 | 0.9866667 | 43 |
| 85 | 0.9866667 | 47 |
| 90 | 0.9866667 | 50 |
| 95 | 0.9866667 | 54 |
| 100 | 0.9866667 | 57 |

Our minimum sample size is small at 80% (*n* = 15 as the minimum). This value is likely deflated, as the original sample sizes per word are around 26 per word. We could consider using 90% (*n* = 19) or 95% 36. Potentially, these low values are due to the large number of items.

If we make our criterion more strict (by lowering the decile), we find similar values.

# Function for simulation  
var2 <- item\_power(data = DF, # name of data frame  
 dv\_col = "Rating", # name of DV column as a character  
 item\_col = "items", # number of items column as a character  
 sample\_start = 20,   
 sample\_stop = 100,   
 sample\_increase = 5,  
 decile = .3)  
  
flextable(var2$final\_sample %>%   
 filter(Percent\_Below >= .80) %>%   
 arrange(Percent\_Below, new\_sample)) %>%   
 autofit()

| sample\_size | Percent\_Below | new\_sample |
| --- | --- | --- |
| 60 | 0.8133333 | 29 |
| 65 | 0.8800000 | 33 |
| 70 | 0.8800000 | 36 |
| 85 | 0.9466667 | 47 |
| 75 | 0.9600000 | 40 |
| 90 | 0.9600000 | 50 |
| 80 | 0.9733333 | 43 |
| 95 | 0.9733333 | 54 |
| 100 | 0.9866667 | 57 |

#### Maximum Sample Size

While there are many considerations for maximum sample size (time, effort, resources), if we consider a higher value just for estimation sake, we could use *n* = 29 at 98%.

#### Final Sample Size

In any estimate for sample size, you should also consider the potential for missing data and/or unusable data due to any other exclusion criteria in your study (i.e., attention checks, speeding, getting the answer right, etc.). In this study, these values may be influenced by the other variables that we used to select the stimuli in the study.