Response Time in Semantic Memory in a Divergent Thinking Task

David Marcos Cuesta

2023-12-06

Libraries

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.3     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(qdapDictionaries)
library(here)
```

```
## here() starts at /Users/davidmarcoscuesta/Documents/GitHub/COGS 212 DATA SCIENCE /Data-Science-Project
```

```
library(readxl)
library(skimr)
```

INTRODUCTION

This analysis is motivated by a collaboration offered to me from the laboratory of Dr. Tyler Marghetis, in collaboration with my colleague Soran Malaie. Within this experiment there are many analyzes to be done and the possibility of working together arose since we have similar interests. The database is the result of two experiments that were carried out last year and with which a scientific article called: “Divergent and Convergent Creativity Are Different Kinds of Foraging” has just been published.

The paper investigates the evolutionary basis of human creativity, proposing that our creative capacities may have evolved from spatial foraging abilities. Through an experimental approach, the study demonstrates that tasks involving spatial searches can systematically influence subsequent creative thinking—divergent thinking is enhanced by spatially dispersed search, while convergent thinking benefits from a focused search pattern. These findings suggest a shared cognitive mechanism between spatial navigation and creative processes, supporting the idea that high-level cognitive functions may be grounded in more primitive, embodied experiences.

The task where we are going to perform the analyzes is called stem task, outlined by Malaie et al. (2023), participants are challenged with a word stem completion exercise, a method probing the expansiveness of semantic memory and divergent thinking capabilities. This task necessitates the rapid generation of words from provided two-letter prompts, reflecting the participant’s ability to creatively navigate their mental lexicon (Warrington & Weiskrantz, 1970; 1974). It’s a critical component in evaluating the cognitive processes related to creativity and semantic foraging, emphasizing the link between language and thought (Graf & Mandler, 1984).

This database offers an opportunity to better understand our semantic memory and our process of retrieving words from memory. Due to the nature of the data, causalities cannot be found, but it can serve as preliminary work for future lines of work. The main hypothesis is that we were going to find a relationship between the frequency with which those words are used in the English language and response time, as a measure of how quickly you retrieve that word from memory. This hypothesis aligns with existing research indicating that word frequency can modulate brain activation in language-related tasks (Sanchez, A., & Paz-Alonso, P. M., 2023).

METHODS

The methods that I used for this exploratory data analysis (EDAD) is based on Peng and Matsui's book "The Art of Data Science", retrieved from: https://bookdown.org/rdpeng/artofdatascience/ (https://bookdown.org/rdpeng/artofdatascience/).

They approach involves an "epicycle" of analysis, which includes setting expectations, collecting information, and comparing expectations to data. This cycle is applied throughout the data analysis process to refine questions and hypotheses, ensuring a thorough exploration and understanding of the data.

We have begun by doing an exploratory analysis of the original research dataset to understand how the variables have been recorded, the analysis possibilities it offers us and its limitations. Subsequently, the original database was cleaned, reducing the number of valid entries for our analysis from 15,830 to 4,635 (the experiments were done online, so many entries are wrong and are not useful to us. Based on the possibilities that the database offered, a hypothesis has been generated. After carrying out the relevant analyses, our initial hypothesis has been compared with the results of the analysis.

Raw Dataset

From the experiments for the paper: "Divergent and Convergent Creativity Are Different Kinds of Foraging" by Soran Malaie.

```
master_file <- read.csv(here("data", "master_file_01.csv"))
```

Basic EDA for the RAW Data Set "master_file_filtered"

We observe that the database has a number of columns 30 and a total number of responses of 15830. We find a number of missing entries of 15687 in total.

```
# First exploratory analyses, checking NA's and packaging
skim(master_file)
```

Data summary

Name	master_file
Number of rows	15830
Number of columns	30
_____	
Column type frequency:	
character	17
logical	7
numeric	6
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
failed_images	15687	0.01	2	2	0	1	0
failed_audio	15687	0.01	2	2	0	1	0
failed_video	15687	0.01	2	2	0	1	0
trial_type	0	1.00	6	23	0	9	0
internal_node_id	0	1.00	7	18	0	1479	0
subject_id	0	1.00	24	24	0	143	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
study_id	0	1.00	24	24	0	1	0
session_id	0	1.00	24	24	0	143	0
browser	15687	0.01	5	13	0	5	0
browser_version	15687	0.01	6	10	0	16	0
os	15687	0.01	5	11	0	5	0
stimulus	15401	0.03	20	301	0	4	0
response	2930	0.81	1	1080	0	2999	0
view_history	15115	0.05	38	79	0	709	0
png	12970	0.18	36	38	0	2860	0
Response	14142	0.11	7	7	0	1	0
accuracy	15401	0.03	2	2	0	1	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
success	15544	0.02	1.00	TRU: 286
timeout	15687	0.01	0.00	FAL: 143
webaudio	15687	0.01	1.00	TRU: 143
mobile	15687	0.01	0.00	FAL: 143
fullscreen	15687	0.01	1.00	TRU: 143
webcam	15687	0.01	0.76	TRU: 109, FAL: 34
microphone	15687	0.01	0.83	TRU: 119, FAL: 24

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
trial_index	0	1.00	59.55	39.78	0	27.00	55.00	87.00	242.00	
time_elapsed	0	1.00	1170881.80	814568.72	8	595671.50	1025090.00	1556878.00	4997597.00	
width	15687	0.01	1641.99	354.06	1120	1366.00	1536.00	1920.00	3440.00	
height	15687	0.01	935.10	174.55	657	768.00	900.00	1080.00	1440.00	
vsync_rate	15687	0.01	68.92	31.35	10	60.25	60.75	61.12	242.62	
rt	2117	0.87	19659.53	39808.27	4	2599.00	4837.00	20439.00	1234629.00	

Quick check of the first and last rows of the data set.

```
head(master_file)
```

```
##   success timeout failed_images failed_audio failed_video
## 1    TRUE   FALSE            []           []           []
## 2    TRUE      NA          <NA>         <NA>         <NA>
## 3      NA      NA          <NA>         <NA>         <NA>
## 4      NA      NA          <NA>         <NA>         <NA>
## 5      NA      NA          <NA>         <NA>         <NA>
## 6      NA      NA          <NA>         <NA>         <NA>
##               trial_type trial_index time_elapsed internal_node_id
## 1               preload           0           39          0.0-0.0
## 2            fullscreen           1       100353          0.0-1.0
## 3          browser-check           2       100771          0.0-2.0
## 4 html-keyboard-response           3       101996          0.0-3.0
## 5           instructions           4       108387          0.0-4.0
## 6           instructions           5       206823          0.0-5.0
##                subject_id                 study_id               session_id
## 1 55b2d3f2fdf99b525bc839aa 641387e815ec15c3ccbb8ea1 6419e61830db1f1bdf1b70dd
## 2 55b2d3f2fdf99b525bc839aa 641387e815ec15c3ccbb8ea1 6419e61830db1f1bdf1b70dd
## 3 55b2d3f2fdf99b525bc839aa 641387e815ec15c3ccbb8ea1 6419e61830db1f1bdf1b70dd
## 4 55b2d3f2fdf99b525bc839aa 641387e815ec15c3ccbb8ea1 6419e61830db1f1bdf1b70dd
## 5 55b2d3f2fdf99b525bc839aa 641387e815ec15c3ccbb8ea1 6419e61830db1f1bdf1b70dd
## 6 55b2d3f2fdf99b525bc839aa 641387e815ec15c3ccbb8ea1 6419e61830db1f1bdf1b70dd
##   width height webaudio browser browser_version mobile         os fullscreen
## 1    NA     NA       NA    <NA>            <NA>     NA       <NA>         NA
## 2    NA     NA       NA    <NA>            <NA>     NA       <NA>         NA
## 3  2560   1440     TRUE  chrome         111.0.0  FALSE Windows 10       TRUE
## 4    NA     NA       NA    <NA>            <NA>     NA       <NA>         NA
## 5    NA     NA       NA    <NA>            <NA>     NA       <NA>         NA
## 6    NA     NA       NA    <NA>            <NA>     NA       <NA>         NA
##   vsync_rate webcam microphone    rt
## 1         NA     NA         NA    NA
## 2         NA     NA         NA    NA
## 3      144.2   TRUE       TRUE    NA
## 4         NA     NA         NA  1225
## 5         NA     NA         NA  6391
## 6         NA     NA         NA 98436
##                                         stimulus response
## 1                                           <NA>     <NA>
## 2                                           <NA>     <NA>
## 3                                           <NA>     <NA>
## 4 Welcome to the experiment! Press any key to start.     <NA>
## 5                                           <NA>     <NA>
## 6                                           <NA>     <NA>
##                                                                view_history
## 1                                                                      <NA>
## 2                                                                      <NA>
## 3                                                                      <NA>
## 4                                                                      <NA>
## 5                                     [{"page_index":0,"viewing_time":6391}]
## 6 [{"page_index":0,"viewing_time":60372},{"page_index":1,"viewing_time":38064}]
##    png Response accuracy
## 1 <NA>     <NA>     <NA>
## 2 <NA>     <NA>     <NA>
## 3 <NA>     <NA>     <NA>
## 4 <NA>     <NA>     <NA>
## 5 <NA>     <NA>     <NA>
## 6 <NA>     <NA>     <NA>
```

```
tail(master_file)
```

```
##       success timeout failed_images failed_audio failed_video
## 15825      NA      NA          <NA>         <NA>         <NA>
## 15826      NA      NA          <NA>         <NA>         <NA>
## 15827      NA      NA          <NA>         <NA>         <NA>
## 15828      NA      NA          <NA>         <NA>         <NA>
## 15829      NA      NA          <NA>         <NA>         <NA>
## 15830      NA      NA          <NA>         <NA>         <NA>
##                   trial_type trial_index time_elapsed  internal_node_id
## 15825       survey-html-form         127      1964790 0.0-10.0-1.9-0.88
## 15826           instructions         128      1974885          0.0-11.0
## 15827                 survey         129      2050765          0.0-12.0
## 15828                 survey         130      2078205          0.0-13.0
## 15829                 survey         131      2113289          0.0-14.0
## 15830 html-keyboard-response         132      2121403          0.0-15.0
##                     subject_id                 study_id
## 15825 6415d236b70f6875edc68aad 641387e815ec15c3ccbb8ea1
## 15826 6415d236b70f6875edc68aad 641387e815ec15c3ccbb8ea1
## 15827 6415d236b70f6875edc68aad 641387e815ec15c3ccbb8ea1
## 15828 6415d236b70f6875edc68aad 641387e815ec15c3ccbb8ea1
## 15829 6415d236b70f6875edc68aad 641387e815ec15c3ccbb8ea1
## 15830 6415d236b70f6875edc68aad 641387e815ec15c3ccbb8ea1
##                     session_id width height webaudio browser browser_version
## 15825 6419cda09ff158f6bc32fd62    NA     NA       NA    <NA>            <NA>
## 15826 6419cda09ff158f6bc32fd62    NA     NA       NA    <NA>            <NA>
## 15827 6419cda09ff158f6bc32fd62    NA     NA       NA    <NA>            <NA>
## 15828 6419cda09ff158f6bc32fd62    NA     NA       NA    <NA>            <NA>
## 15829 6419cda09ff158f6bc32fd62    NA     NA       NA    <NA>            <NA>
## 15830 6419cda09ff158f6bc32fd62    NA     NA       NA    <NA>            <NA>
##       mobile   os fullscreen vsync_rate webcam microphone    rt
## 15825     NA <NA>         NA         NA     NA         NA    NA
## 15826     NA <NA>         NA         NA     NA         NA 10094
## 15827     NA <NA>         NA         NA     NA         NA 75822
## 15828     NA <NA>         NA         NA     NA         NA 27407
## 15829     NA <NA>         NA         NA     NA         NA 35073
## 15830     NA <NA>         NA         NA     NA         NA  8111
##
stimulus
## 15825
<NA>
## 15826
<NA>
## 15827
<NA>
## 15828
<NA>
## 15829
<NA>
## 15830 Please press Enter for your results to be saved. Do not close this window; you will be redirected to
the Prolific website automatically to get your credit after we received your results. It may take one minute,
or longer depending on your internet speed! <p><b>Thank you for your participation :) </b>
##
response
## 15825
<NA>
## 15826
<NA>
## 15827
{"Author":["J.R.R. Tolkien","John Grisham","Tom Clancy","Ernest Hemingway"]}
## 15828 {"Langauge":"Yes","2ndLanguage":"Spanish","2nd_lg_fluency":2,"SecondLanguage":"","3rd_lg_fluency":nu
ll,"age":"28","Highest_degree":"Doctoral degree (e.g., PhD, MD, JD)","Handedness":"Right","Race":["White","Sp
anish, Hispanic, or Latino"],"sexual_orientation":"straight (heterosexual)","gender":"man"}
## 15829
{"final_comment":"This study is not ideal for people using a desktop with a touchpad. "}
## 15830
```

```
enter
##                                 view_history  png Response accuracy
## 15825                                    <NA> <NA>  timeout     <NA>
## 15826 [{"page_index":0,"viewing_time":10092}] <NA>     <NA>     <NA>
## 15827                                    <NA> <NA>     <NA>       []
## 15828                                    <NA> <NA>     <NA>       []
## 15829                                    <NA> <NA>     <NA>       []
## 15830                                    <NA> <NA>     <NA>     <NA>
```

Checking the variables that we have registered in the data set

```
colnames(master_file)
```

```
##  [1] "success"          "timeout"          "failed_images"    "failed_audio"
##  [5] "failed_video"     "trial_type"       "trial_index"      "time_elapsed"
##  [9] "internal_node_id" "subject_id"       "study_id"         "session_id"
## [13] "width"            "height"           "webaudio"         "browser"
## [17] "browser_version"  "mobile"           "os"               "fullscreen"
## [21] "vsync_rate"       "webcam"           "microphone"       "rt"
## [25] "stimulus"         "response"         "view_history"     "png"
## [29] "Response"         "accuracy"
```

Interesting variables for our research

We are especially interested in analyzing the response time to understand if the frequency with which that word is used is interfering in the recovery process and thus be able to contribute new knowledge to how our semantic memory works.

To answer our research question, we are only interested in working with the columns: subject_id, stimulus, trial_index, time_elapsed, rt, and response.

```
# Filtering the variables of interest to work with
master_file_filtered <- read.csv(here("data", "master_file_01.csv")) %>%
  select(subject_id, stimulus, trial_index, time_elapsed, rt, response)
```

Here we created a function to clean the response text

```
clean_response <- function(data, pattern, cols_to_remove) {
  data %>%
    # Filter rows where the 'response' column contains the specified regex pattern.
    filter(str_detect(response, pattern)) %>%
    # Remove quotes, curly braces, and other characters from the 'response' column.
    mutate(response = str_replace_all(response, '[\"{}]', '')) %>%
    # Remove the substring 'let_' and any spaces from the 'response' column.
    mutate(response = str_remove_all(response, "let_| ")) %>%
    # Split the 'response' column into two separate columns: 'condition' and 'response' using ':' as the sepa
rator.
    separate(response, c("condition", "response"), sep = ":")
}
```

Our hypothesis is that the greater the frequency of the word, the less time it will take to retrieve that word from memory.

To do this, we are going to select the Corpus of Contemporary American English (COCA), the 60k version, retrieved from: https://www.wordfrequency.info/samples.asp (https://www.wordfrequency.info/samples.asp), to classify the words by word frequency ranks, with rank 1 being the most frequent word and rank 60,000 being the least frequent. Shows the frequency (raw frequency and frequency per million words) in each of the eight main genres: blogs, other web, TV/Movies, (more formal) spoken, fiction, magazine, newspaper, and academic.

```
df_word_freq <- read_excel('data/wordFrequency60k.xlsx',
                           sheet = 'lemmas')
```

Let's understand what is inside the COCA 60k version data set

We observe that there are 5050 rows and 25 columns.

```
skim(df_word_freq)
```

Data summary

Name	df_word_freq
Number of rows	5050
Number of columns	25
_____	
Column type frequency:	
character	2
numeric	23
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
lemma	0	1	1	16	0	4380	0
PoS	0	1	1	1	0	14	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
rank	0	1	2525.50	1457.95	1.00	1263.25	2525.50	3787.75	5050.00	▇▇▇▇▇
freq	0	1	164383.05	1201260.84	11875.00	18345.00	32254.50	76310.25	50033612.00	▇▁▁▁▁
perMil	0	1	165.52	1209.70	11.60	18.47	32.48	76.84	50385.16	▇▁▁▁▁
%caps	0	1	0.12	0.17	0.00	0.04	0.06	0.12	1.02	▇▁▁▁▁
%allC	0	1	0.01	0.04	0.00	0.00	0.01	0.01	1.00	▇▁▁▁▁
range	0	1	39028.65	57949.51	61.00	11117.50	18842.00	39692.00	482995.00	▇▁▁▁▁
disp	0	1	0.94	0.09	-4.46	0.93	0.95	0.97	1.01	▁▁▁▁▇
blog	0	1	21363.66	162766.09	0.00	2091.00	3884.00	9701.75	6266654.00	▇▁▁▁▁
web	0	1	21740.80	169179.41	0.00	2374.25	4348.00	10314.50	7095508.00	▇▁▁▁▁
TVM	0	1	22888.30	192625.00	0.00	901.25	2233.50	6043.25	8029744.00	▇▁▁▁▁
spok	0	1	22033.91	176544.31	0.00	1369.00	2847.50	7560.00	7025941.00	▇▁▁▁▁
fic	0	1	20149.06	152982.10	0.00	1238.25	3071.00	7619.50	6307838.00	▇▁▁▁▁
mag	0	1	20336.69	155239.89	0.00	2603.25	4723.00	10692.75	6801589.00	▇▁▁▁▁
news	0	1	19229.25	148779.20	0.00	2136.50	4241.00	9670.50	6579270.00	▇▁▁▁▁
acad	0	1	18917.16	159998.64	0.00	1732.25	4379.00	11036.25	7440931.00	▇▁▁▁▁

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
blogPM	0	1	171.94	1309.95	0.00	16.83	31.26	78.08	50434.35	▇▁▁▁▁
webPM	0	1	169.04	1315.41	0.00	18.46	33.80	80.20	55169.32	▇▁▁▁▁
TVMPM	0	1	178.71	1504.01	0.00	7.03	17.44	47.19	62695.87	▇▁▁▁▁
spokPM	0	1	174.68	1399.64	0.00	10.85	22.58	59.94	55701.50	▇▁▁▁▁
ficPM	0	1	170.29	1292.93	0.00	10.46	25.95	64.40	53310.74	▇▁▁▁▁
magPM	0	1	161.29	1231.17	0.00	20.64	37.45	84.80	53941.86	▇▁▁▁▁
newsPM	0	1	157.95	1222.09	0.00	17.55	34.84	79.44	54042.73	▇▁▁▁▁
acadPM	0	1	157.92	1335.65	0.00	14.46	36.56	92.13	62116.23	▇▁▁▁▁

Everything is as expected. The first ranks are the ones with the most frequency and the last words in the ranking are the ones with the least frequency.

```
head(df_word_freq)
```

```
## # A tibble: 6 × 25
##    rank lemma PoS       freq perMil `%caps` `%allC`  range  disp    blog     web
##   <dbl> <chr> <chr>    <dbl>  <dbl>   <dbl>   <dbl>  <dbl> <dbl>   <dbl>   <dbl>
## 1     1 the   a     50033612 50385.    0.11    0    482995  0.98 6266654 7095508
## 2     2 be    v     32394756 32623.    0.03    0.01 481177  0.99 5594001 5325829
## 3     3 and   c     24778098 24952.    0.09    0    478670  0.98 3205178 3453140
## 4     4 a     a     24225478 24396.    0.04    0.04 478204  0.99 3098338 3182822
## 5     5 of    i     23159162 23322.    0.01    0    477933  0.97 2897295 3419616
## 6     6 to    t     16770155 16888.    0.02    0    471197  1.01 2395858 2286438
## # ℹ 14 more variables: TVM <dbl>, spok <dbl>, fic <dbl>, mag <dbl>, news <dbl>,
## #   acad <dbl>, blogPM <dbl>, webPM <dbl>, TVMPM <dbl>, spokPM <dbl>,
## #   ficPM <dbl>, magPM <dbl>, newsPM <dbl>, acadPM <dbl>
```

```
tail(df_word_freq)
```

```
## # A tibble: 6 × 25
##    rank lemma   PoS    freq perMil `%caps` `%allC` range  disp  blog   web   TVM
##   <dbl> <chr>   <chr> <dbl>  <dbl>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  5045 hammer  n     11886   12.0    0.34    0.05  6776  0.92  1025  1153  1715
## 2  5046 keyboa… n     11885   12.0    0.05    0     5580  0.92  2464  2476   314
## 3  5047 persist v     11880   12.0    0.01    0     9554  0.94  1181  1620   170
## 4  5048 wheat   n     11877   12.0    0.15    0.01  5843  0.94  1287  1450   608
## 5  5049 predat… n     11876   12.0    0.14    0.01  6042  0.94  1166  1138  1041
## 6  5050 bizarre j     11875   12.0    0.04    0.01  9737  0.97  1794  1639  1083
## # ℹ 13 more variables: spok <dbl>, fic <dbl>, mag <dbl>, news <dbl>,
## #   acad <dbl>, blogPM <dbl>, webPM <dbl>, TVMPM <dbl>, spokPM <dbl>,
## #   ficPM <dbl>, magPM <dbl>, newsPM <dbl>, acadPM <dbl>
```

Analysis possibilities

Here we look at the possibilities we have. As we do not have data on the media most used by the subjects, we will be interested in using the general frequency ('freq').

```
colnames(df_word_freq)
```

```
##  [1] "rank"   "lemma"  "PoS"    "freq"   "perMil" "%caps"  "%allC"  "range"
##  [9] "disp"   "blog"   "web"    "TVM"    "spok"   "fic"    "mag"    "news"
## [17] "acad"   "blogPM" "webPM"  "TVMPM"  "spokPM" "ficPM"  "magPM"  "newsPM"
## [25] "acadPM"
```

Deciding the columns we are goint to use

We decided that to answer our research question, the columns we are going to select for our analysis are: ‘rank’, ‘lemma’, and ’freq.

```
df_word_freq_filtered <- read_excel('data/wordFrequency60k.xlsx',
                           sheet = 'lemmas')  %>%
  select(c('rank','lemma','freq'))
```

Before cleaning our data set, we have to select an English language dictionary to help us discern which answers are correct and which are not. We choose the ‘qdapDictionaries’ package.

Along with the data set, we were notified that certain answers had been accepted as valid in the experiment without them being in the dictionary, so we were given a list of the words that had been accepted as valid in the experiment. Which we integrated into the R code with the name “words_not_in_dic”.

In the following code we define the dictionary that we will use to clean the raw

database:

```
# The purpose of this function is to collect responses from subjects that are not included in the English
# language dictionary that we are using (qdapDictionaries) and add it.

words_not_in_dic = c("zoopathology", "zolt", "zoolagist", "zot", "zock", "zook", "zon",
                     "zosh", "zoze", "zoam", "zommer", "zozo", "zoomies", "zoop", "zope",
                     "zoetrope", "zoocology", "zote", "zoge", "zoed", "zomed", "zork",
                     "zomboni", "zotie", "zolac", "zoz", "zoadic", "zooplankton", "zoro",
                     "zong", "zop", "zoan", "zofia", "zoodle", "zoinks", "zowwys",
                     "zootropic",
                     "stoll", "steele","steamstream", "stuper","stong", "stupify", "steale", "stickly",
                     "stats", "stat", "steeler",
                     "protype", "pringle", "preporation", "preech", "pradae", "proactive", "practicum",
                     "priranha", "primede", "prine", "priviledge", "privelege", "priere", "pratt",
                     "predjudice", "pread", "procer", "prad", "probbe", "protone", "pronation", "prada",
                     "proffesional", "promo",
                     "mifted", "milf", "misadministration", "microplane", "milkshake", "minnce",
                     "millineter", "mit", "milor", "mitochondria", "minging", "millisecond", "millow",
                     "mittle", "mior","mich", "misunderstood", "miniatrure",
                     "leep", "lemer", "leem", "leen", "leahc", "leavor", "lepper", "letal", "leb",
                     "leir", "lerch", "lego",
                     "gray", "grinch", "greive", "grampa", "grungy", "grunge", "grandeous", "grap",
                     "grilla", "gret", "groot", "groope", "grimm", "growed",
                     "gluck", "glaven", "glute", "glag", "glicemic", "gle", "glam", "gleen", "glamping",
                     "glimp", "glack", "gloitter", "glintstone", "glep", "gleem", "glock", "glot",
                     "glay", "glup", "glantern", "glurp",
                     "eannagram", "eaw", "eaze", "eatily", "eal", "east", "eavesrop", "eads", "eab",
                     "eap", "eather", "easports",
                     "dagwood", "dat", "dans", "dawg", "dain", "datamine", "daquirri", "dask", "dallup",
                     "daith", "dack", "dax", "dall", "daly",
                     "abcess", "abs", "abba", "abscent", "abhorent", "abalon", "abott", "abcs", "abraid",
                     "abling", "abt", "abrash", "abdomnial", "abled", "abdjure", "abdominals", "aboot",
                     "abbhor", "abondant", "abicuss", "abrail", "ablebody", "abscomb", "abid", "abduction",
                     "abor", "abeed", "abacist"
                     )

# Adding words that considered non-word by dictionary's default, but are true words, and adding it to the "qd
apDictionaries" that I am using.
original_dict <- c(qdapDictionaries::GradyAugmented, words_not_in_dic)

# Removing non-words that considered words by dictionary's default:
filtered_dict = original_dict[!original_dict %in% c("st", "mi", "da")]

# Function used to check whether a given word exists in our dictionary:
is.word <- function(word, dictionary) {
  tolower(word) %in% dictionary
}
```

Data processing

We apply the ‘clean_response’ function, check if it is an existing word in our dictionary and select the columns that we are going to use for our analysis.

```
# Data processing pipeline
stem_task_data <- clean_response(master_file_filtered,
                                 'let_diverse|let_linear',
                                 c("stimulus")) %>%
  # Convert 'response' first two letters to lowercase for consistency
  mutate(letter_set = tolower(substr(response,
                                     start = 1,
                                     stop = 2))) %>%
  # Keep rows with specific two-letter sets for focused analysis
  filter(grepl('ab|da|ea|gl|gr|le|mi|pr|st|zo',
               letter_set, ignore.case = TRUE)) %>%
  # Ensure responses are valid words
  filter(is.word(response, filtered_dict)) %>%
  # Narrow down dataset to essential columns for analysis
  select(c('subject_id', 'time_elapsed',
           'rt', 'response', 'letter_set'))
```

Merge the datasets, adding the word frequency

With this line we manage to add a column with the frequency of each of the responses in our data set. Is going to be the final data set that we are going to use for the analysis.

```
combined_data <- merge(stem_task_data, df_word_freq_filtered, by.x = "response", by.y = "lemma")
```

Basic Stats

Our final database is composed of 143 subjects. The mean response time is: 19659.53, median: 4837 with a Standard Deviation of Response Time of: 39808.27.

```
# Number of Unique Subjects
num_subjects <- length(unique(combined_data$subject_id))
cat("Number of Subjects:", num_subjects, "\n")
```

```
## Number of Subjects: 120
```

```
# Range of Response Times
min_rt <- min(combined_data$rt, na.rm = TRUE)
max_rt <- max(combined_data$rt, na.rm = TRUE)
cat("Range of Response Times: Min =", min_rt, ", Max =", max_rt, "\n")
```

```
## Range of Response Times: Min = 358 , Max = 26749
```

```
# Mean and Median of Response Time
mean_rt <- mean(combined_data$rt, na.rm = TRUE)
median_rt <- median(combined_data$rt, na.rm = TRUE)
cat("Mean Response Time:", mean_rt, "\n")
```

```
## Mean Response Time: 3643.337
```

```
cat("Median Response Time:", median_rt, "\n")
```

```
## Median Response Time: 3056
```

```
# Standard Deviation of Response Time
sd_rt <- sd(combined_data$rt, na.rm = TRUE)
cat("Standard Deviation of Response Time:", sd_rt, "\n")
```

```
## Standard Deviation of Response Time: 2403.828
```

```
# Mean and Median of Word Frequency Rank
mean_freq <- mean(combined_data$freq, na.rm = TRUE)
median_freq <- median(combined_data$freq, na.rm = TRUE)
cat("Mean Word Frequency Rank:", mean_freq, "\n")
```

```
## Mean Word Frequency Rank: 170914.2
```

```
cat("Median Word Frequency Rank:", median_freq, "\n")
```

```
## Median Word Frequency Rank: 60710
```

Check Normality and Linearity

To choose which statistic we are going to use to analyze if there is a correlation between response time and word frequency, we are going to observe using a histogram how our variables are distributed, to see if they comply with normality and parametric tests can be used or if Applying non-parametric statistics does not comply.

Histogram to assess the distribution visually. The red line represents a kernel density estimate of your data. The dotted blue line is a normal distribution curve superimposed on the histogram, and is a visual representation of what your data distribution would look like if it followed a perfect normal distribution. X-axis limited to 20k for visualization purposes, the rows dropped out do not provide relevant information to analyze whether it comply with the normality curve.

Neither of the two variables follows normality, so we will have to use non-parametric statistics.

```
# Creating a histogram with mean and median lines, along with density and normal distribution curves
ggplot(combined_data, aes(x = rt)) +
  geom_histogram(
    aes(y = ..density..),
    binwidth = 100,
    fill = "#69b3a2",
    color = "black"
  ) +
  geom_vline(
    aes(xintercept = mean_rt),
    color = "blue",
    linetype = "dashed",
    size = 1,
    alpha = 0.7
  ) +
  geom_vline(
    aes(xintercept = median_rt),
    color = "green",
    linetype = "dashed",
    size = 1,
    alpha = 0.7
  ) +
  xlim(c(0, 15000)) +
  theme_minimal() +
  geom_density(
    color = "red",
    size = 1
  ) +
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(combined_data$rt, na.rm = TRUE),
      sd = sd(combined_data$rt, na.rm = TRUE)
    ),
    color = "blue",
    linetype = "dashed"
  ) +
  labs(
    title = "Histogram of Response Time with Mean and Median (Up to 10,000)",
    x = "Response Time",
    y = "Density"
  ) +
  geom_text(
    aes(x = mean_rt, y = 0, label = paste("Mean =", round(mean_rt, 2))),
    vjust = -25,
    color = "blue",
    hjust = 0,
    size = 4,
    alpha = 0.7
  ) +
  geom_text(
    aes(x = median_rt, y = 0, label = paste("Median =", round(median_rt, 2))),
    vjust = -29,
    color = "green",
    hjust = 0,
    size = 4,
    alpha = 0.7
  )
```

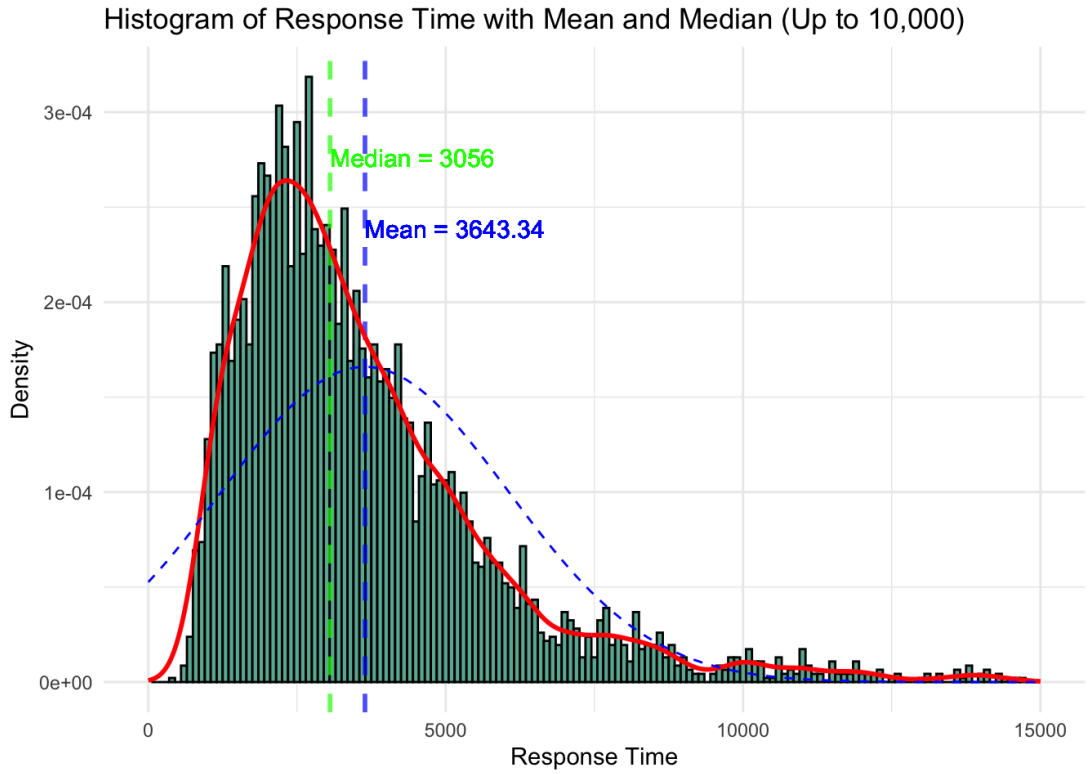
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 21 rows containing non-finite values (`stat_bin()`).
```

```
## Warning: Removed 21 rows containing non-finite values (`stat_density()`).
```

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```

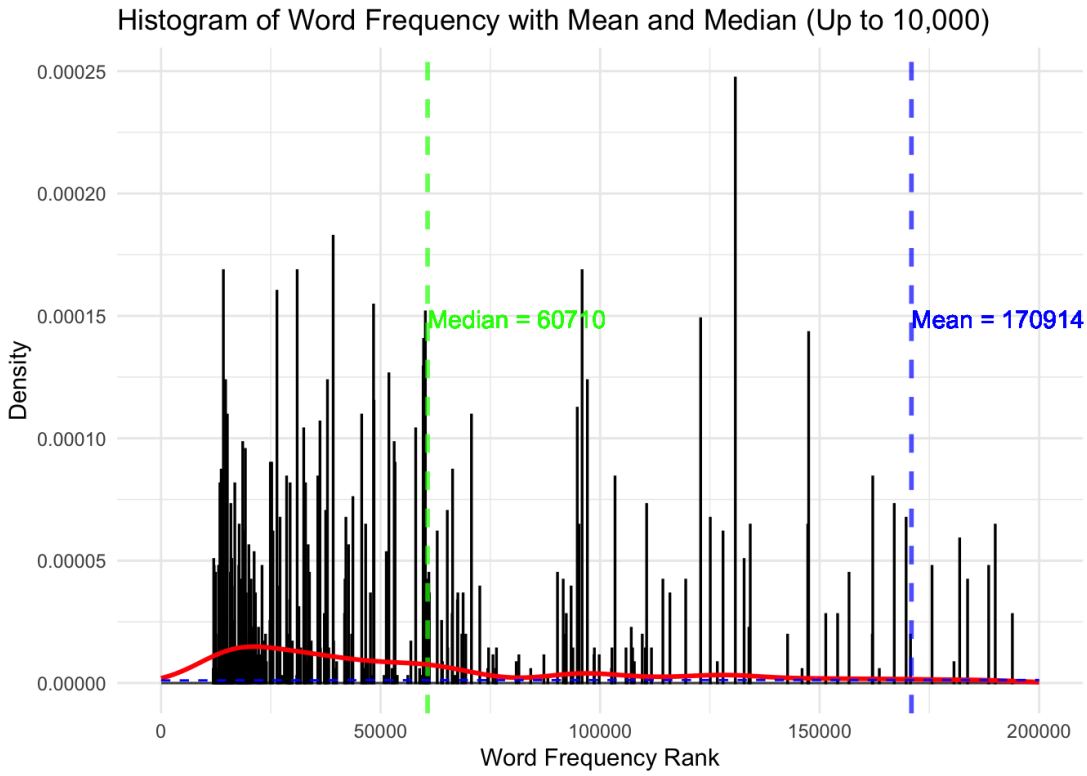


```
# Creating a histogram with mean and median lines, along with density and normal distribution curves
ggplot(combined_data, aes(x = freq)) +
  geom_histogram(
    aes(y = ..density..),
    binwidth = 100,
    fill = "#69b3a2",
    color = "black"
  ) +
  geom_vline(
    aes(xintercept = mean_freq),
    color = "blue",
    linetype = "dashed",
    size = 1,
    alpha = 0.7
  ) +
  geom_vline(
    aes(xintercept = median_freq),
    color = "green",
    linetype = "dashed",
    size = 1,
    alpha = 0.7
  ) +
  xlim(c(0, 200000)) +
  theme_minimal() +
  geom_density(
    color = "red",
    size = 1
  ) +
  stat_function(
    fun = dnorm,
    args = list(
      mean = mean(combined_data$freq, na.rm = TRUE),
      sd = sd(combined_data$freq, na.rm = TRUE)
    ),
    color = "blue",
    linetype = "dashed"
  ) +
  labs(
    title = "Histogram of Word Frequency with Mean and Median (Up to 10,000)",
    x = "Word Frequency Rank",
    y = "Density"
  ) +
  geom_text(
    aes(x = mean_freq, y = 0, label = paste("Mean =", round(mean_freq, 2))),
    vjust = -20,
    color = "blue",
    hjust = 0,
    size = 4,
    alpha = 0.7
  ) +
  geom_text(
    aes(x = median_freq, y = 0, label = paste("Median =", round(median_freq, 2))),
    vjust = -20,
    color = "green",
    hjust = 0,
    size = 4,
    alpha = 0.7
  )
```

```
## Warning: Removed 1076 rows containing non-finite values (`stat_bin()`).
```

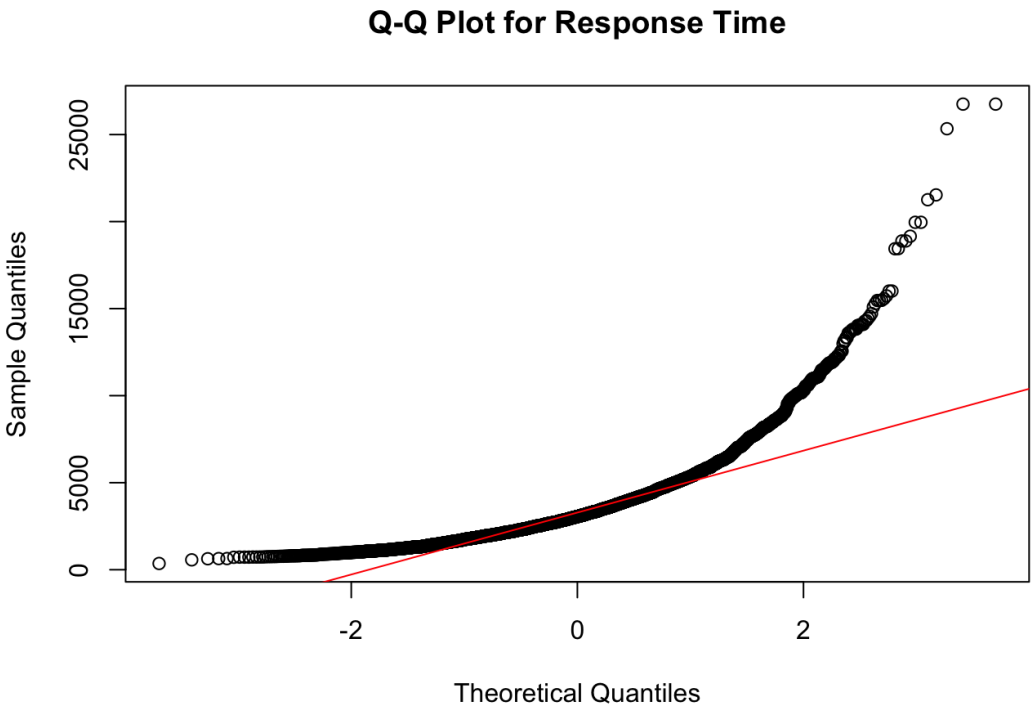
```
## Warning: Removed 1076 rows containing non-finite values (`stat_density()`).
```

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```

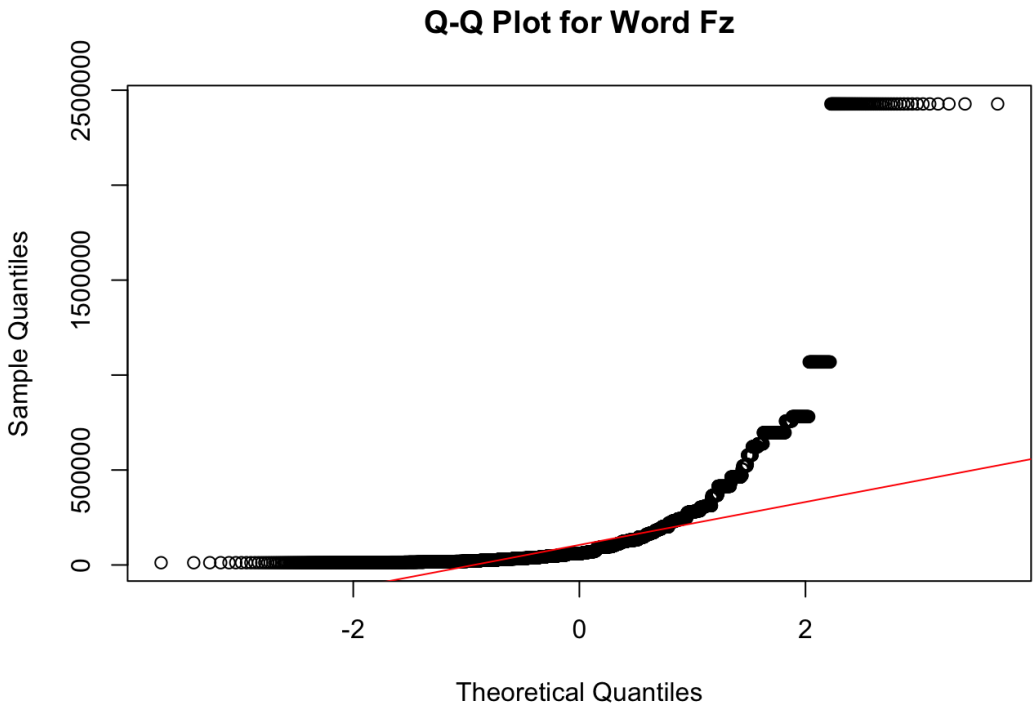


Q-Q Plots to check for deviations from normality

```
qqnorm(combined_data$rt, main = "Q-Q Plot for Response Time")
qqline(combined_data$rt, col = "red") # Color for visibility
```



```
qqnorm(combined_data$freq, main = "Q-Q Plot for Word Fz")
qqline(combined_data$freq, col = "red") # Color for visibility
```

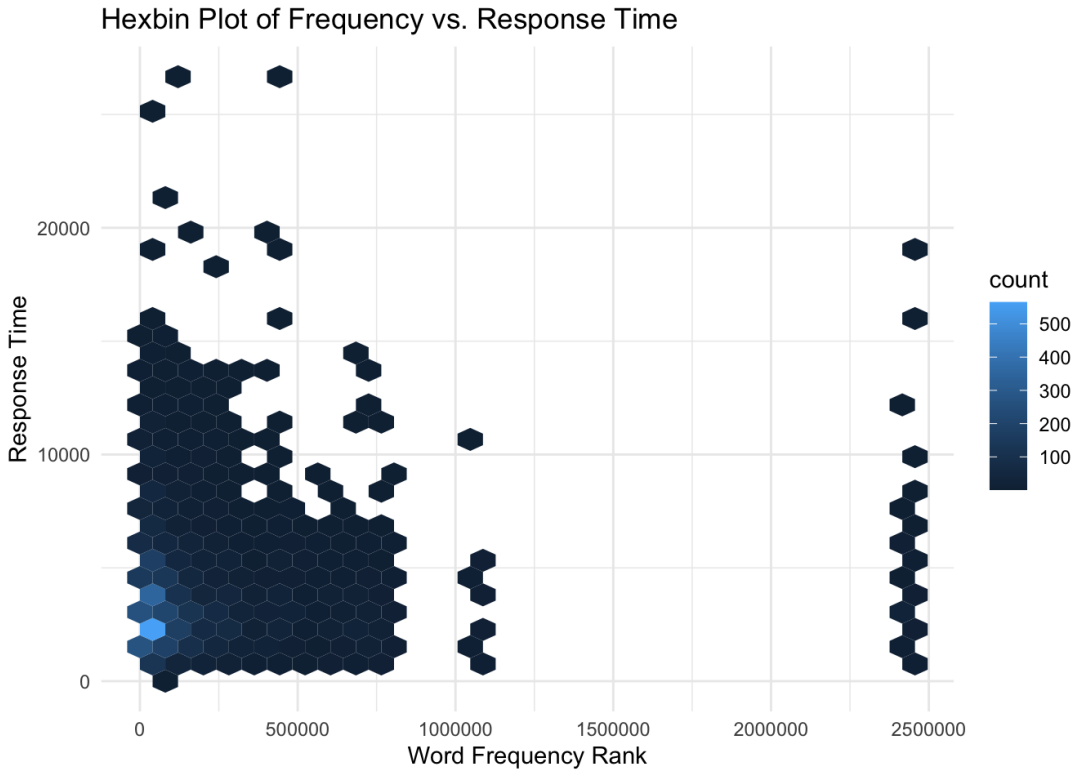


Visualization if there are any correlation between Word Frequenzy and Response Time

Before performing the Spearman correlation statistic, we want to visualize both variables to see if they show signs of correlation between them.

We have chosen Hexbinplot since with a scatter plot it was very difficult to visualize in which area there was more density due to the number of points in the same area. The Hexbin plot suggests that most responses have a high frequency but we can't observe any correlation between the frequency of a word and the time response.

```
ggplot(combined_data, aes(x = freq, y = rt)) +
  geom_hex() +
  labs(title = "Hexbin Plot of Frequency vs. Response Time",
       x = "Word Frequency Rank", y = "Response Time") +
  theme_minimal()
```



Correlation Test

Although there does not seem to be a correlation in the graph, we are going to use the non-parametric statistic Spearman's rank correlation coefficient to ensure that there is no significant correlation

```
cor_test_result <- cor.test(combined_data$rt, combined_data$freq, method="spearman")
```

```
## Warning in cor.test.default(combined_data$rt, combined_data$freq, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
print(cor_test_result)
```

```
##
##  Spearman's rank correlation rho
##
## data:  combined_data$rt and combined_data$freq
## S = 1.6487e+10, p-value = 0.6553
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##         rho
## 0.006558301
```

RESULTS

EDA Results

We have found a database with many NAs, which reduced the sample enormously. With data collection that has required meticulous processing, which can be improved for future research, such as the way of collecting the responses, where many errors have been found and instead of classifying the two experiments in a column and in another put the answers, we have found: {"let_linear":"able"}, which had to be separated into two columns and cleaned. Worrying response times have also been found that reduce the internal and external validity of the experiment, with a Mean Response Time: 3643.337 and a Standard Deviation of Response Time: 2403.828.

The Corpus of Contemporary American English (COCA) is a very interesting data set because it not only has the frequency of words in general, but also classifies them by categories, giving rise to interesting future research.

Correlation Results

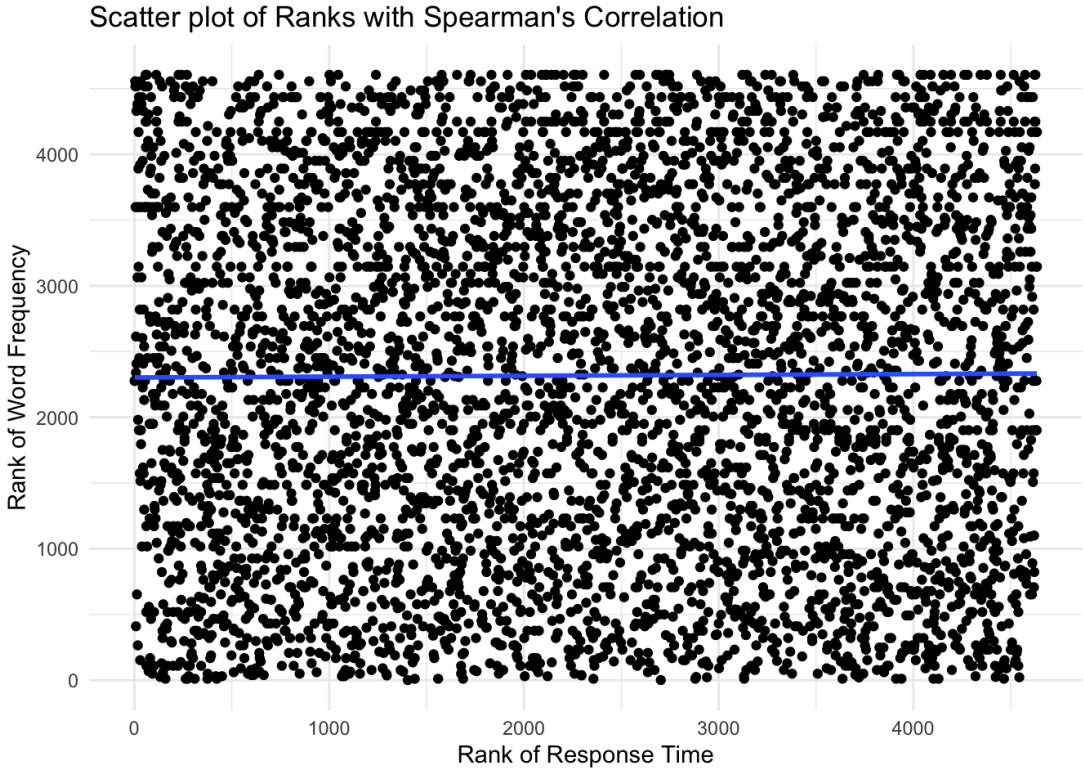
The Spearman's rank correlation test result shows a rho value of approximately 0.0066, suggesting a very weak positive correlation between response time and word frequency. However, the p-value of 0.6553 suggests that this correlation is not statistically significant, meaning there's no strong evidence of a monotonic relationship between the two variables in the sample data.

Visualizing the Spearman's rank correlation coefficient

```
# # We first need to calculate the ranks of the data
combined_data$rank_rt <- rank(combined_data$rt, ties.method = "average")
combined_data$rank_freq <- rank(combined_data$freq, ties.method = "average")

# Now create a scatter plot of these ranks
ggplot(combined_data, aes(x = rank_rt, y = rank_freq)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Rank of Response Time", y = "Rank of Word Frequency",
       title = "Scatter plot of Ranks with Spearman's Correlation") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



The scatter plot shows the ranks of response time on the x-axis and the ranks of word frequency on the y-axis. Each point represents a pair of ranks, and the plot is a visual representation of the Spearman's rank correlation.

The blue line across the plot appears to be a best-fit line through the data points, which should be flat because the ranks of response time and word frequency are plotted against each other. The flatness of the line suggests there is very little to no monotonic relationship between the two ranks, which aligns with the previously mentioned Spearman correlation coefficient of approximately 0.0066 and the high p-value.

The dense clustering of points along the entire range of ranks without a clear upward or downward trend further supports the conclusion of a very weak correlation. This means that knowing the rank of a word's frequency does not provide much information about the rank of the response time in this dataset.

DISCUSSION

We have found that there appears to be no correlation between response time and word frequency. But also exploring the database, we have seen certain limitations that may be interfering in the validity of our analyses, such as that since it is an online experiment there is no type of supervision by any researcher, response times are very high. In general, there was no time limit set, and there are many variables that can interfere with this reaction time such as internet problems, typing speed, distractors around while taking the test, and multiple variables that have not already been controlled. which was not the original goal of the experiments. Therefore, it is important to interpret the results taking these limitations into account. From here it is invited to carry out more controlled tests in future research and to carry out the analysis of the semantic distance between words, to see if relationships are found between the semantic distances and the response time, which would be more in line with the foraging effect that is studied in the original paper of this data set.

The results of this analysis are of utmost importance because they will help us design better experiments where we can control many more variables, generate better databases and study the phenomenon of semantic memory recovery with more detail and scientific rigor.

REFERENCES

Graf, P., & Mandler, G. (1984). Activation makes words more accessible, but not necessarily more retrievable. Journal of Verbal Learning and Verbal Behavior, 23(5), 553–568. https://doi.org/10.1016/S0022-5371(84)90346-3 (https://doi.org/10.1016/S0022-5371(84)90346-3)

Hills, T. T., Jones, M. N., & Todd, P. M. (2012). Optimal foraging in semantic memory. Psychological Review, 119(2), 431. http://dx.doi.org/10.1037/a0027373 (http://dx.doi.org/10.1037/a0027373)

Malaie, S., Spivey, M., & Marghetis, T. (2023). Divergent and Convergent Creativity Are Different Kinds of Foraging.

Peng, D., & Elizabeth, M. (2015). “The Art of Data Science.” A Guide for Anyone Who Works with Data. Skybrude Consulting, LLC.

Sánchez, A., Carreiras, M., & Paz-Alonso, P. M. (2023). Word frequency and reading demands modulate brain activation in the inferior frontal gyrus. Scientific Reports, 13(1), 17217.

Warrington, E. K., & Weiskrantz, L. (1970). Amnesic syndrome: Consolidation or retrieval? Nature, 228(5272), 628–630. https://doi.org/10.1038/228628a0 (https://doi.org/10.1038/228628a0)