EPL 2021 - Assignment 1

Yanming Li 2033070
Xinhao Lan 1082620
Sili Wang 4621514
Koert Mollers 5942446
Bram Pramono 7978359

Q1:

Report descriptive summaries for reaction times (RT) and for frequencies (use FreqCOCAspok, which is a spoken corpus of American English).

	max_RT	min_RT	avg_RT	sd_RT	max_fre	min_fre	avg_fre	sd_fre
result	2998	101	1024.31	331.0548	2694449	0	916.1701	20265.68

Table 1.1 Summaries of RT and Frequencies.

Report descriptive summaries of RT for words and pseudowords (i.e., depending on whether IsWord is TRUE or FALSE) for the whole dataset and also for Subjects 15351, 16854 and 170373.

name	IsWord	count	max_RT	min_RT	avg_RT	sd_RT
Pseudowords	FALSE	86190	2998	110	1090.8393	353.1309
Words	TRUE	86190	2986	101	957.6232	292.5903

Table 1.2 Summaries of RT for words and pseudowords for the whole dataset.

name	IsWord	count	max_RT	min_RT	avg_RT	sd_RT
Pseudowords	FALSE	390	2955	284	1525.2026	469.7161
Words	TRUE	390	2745	559	966.4256	313.4160

Table 1.3 Summaries of RT for words and pseudowords for Subject 15351.

name	IsWord	count	max_RT	min_RT	avg_RT	sd_RT
Pseudowords	FALSE	390	2802	691	1222.177	337.5057
Words	TRUE	390	2972	694	1162.992	354.4487

Table 1.4 Summaries of RT for words and pseudowords for Subject 16854.

name	IsWord	count	max_RT	min_RT	avg_RT	sd_RT
Pseudowords	FALSE	390	1781	585	906.8000	172.1085
Words	TRUE	390	2073	557	847.2385	175.0316

Table 1.5 Summaries of RT for words and pseudowords for Subject 170373.

—

Q2:

```
cohend <- function(x1, x2) {
   n1 <- length(x1)
   n2 <- length(x2)
   if (n1 == n2) {
      s <- sqrt( (var(x1) + var(x2)) / 2 )
   } else {
      numerator <- (n1 - 1) * var(x1) + (n2 - 1) * var(x2)
      denominator <- n1 + n2 - 2
      s <- sqrt(numerator / denominator)
   }
   d <- (mean(x1) - mean(x2)) / s
   return(d)
}
```

	Subject 15351	Subject 16854	Subject 170373	All subjects	Selected subject 15292
Cohen's d	-1.399431117 30686	-0.171013812 364667	-0.343143399 184897	-0.410810089 564973	0.463331740 936854
Effect size	large	small	small	small	small

Table 2.1 Cohen's d and effect size for different Subjects.

—

Q3:

```
tcalculation <- function(x1, x2) {
   n1 <- length(x1)
   n2 <- length(x2)
   if (n1 == n2) {
     se <- sqrt( (var(x1) + var(x2)) / n1 )
   }else{
     se <- sqrt( var(x1) / n1 + var(x2) / n2 )
   }
   t <- (mean(x1)-mean(x2)) / se
   return(t)
```

}

	Subject 15351	All subjects	Selected subject 15292
t-value	-19.5419920467928	-85.2814437314518	1.50136643517682

Table 3.1 t-value for different Subjects.

Say briefly why RTs for all words and pseudowords (the second question) have the highest t-value compared to pseudoword/word 15292 or even the responses of Subject numbered 15351. A brief description of the crucial intuition suffices.

The whole dataset of words and pseudowords has a large n1 and n2 (the number of observations of x1 and x2) while having var(x1) var(x2) on the same scale as the others. This results in a small Standard Error (SE) for the whole dataset. Also, the difference between the mean(x1) and mean(x2) of all samples are on the same scale. With a large SE as the denominator, the t-values for all words and pseudowords are therefore the highest. This result is aligned with the components in the formula to calculate t-value as mentioned by Winter (2019, p. 166). The difference between the 15351 and 15292 datasets can be reasoned using the same arguments.

—

Q4:

Based on what we said so far, you should be able to tie t-values that you provided in Q3 to p-values under the null hypothesis that population means between RTs of words and RTs for pseudowords do not differ, i.e., mean(wordRT)=mean(pseudowordRT). Use the t-value from Q3 for the data set word 15292 and pseudoword 15292 and use the function pt (with degrees of freedom = 40) to provide the answer. You can also check Baayen, section 4.1, for some relevant code on how to use pt.

```
# For NHST right-tail test
p_value_1 <- pt(tcalculation_15292, df=40, lower.tail = FALSE)
p_value_1 = 0.0705571359389232
# For Two-tailed test
p_value_2 <- 2*pt(tcalculation_15292, df=40, lower.tail = FALSE)
p_value_2 = 0.141114271877846
```

This is close to the p-value for t-test function which is 0.1417 (using df=37.008)

When you are done, come back to one of the assumptions of t-probability distributions: t-values are collected from samples of independent and normally distributed data. We focus on the latter condition. Check if RTs in words and pseudowords are normally distributed. If not, try a transformation to get closer to normal distribution. Among transformations, it is common to consider squaring, cubing, taking an inverse, taking square root, or

log-transforming data. It is fine if you find only a roughly normal distribution (no testing needed, just checking by observing a histogram is sufficient for this exercise). Once you find the best case of transformation, report t values and p values for this transformed distribution. You can decide whether you want to use one-tailed or two-tailed tests but whatever you decide, report that

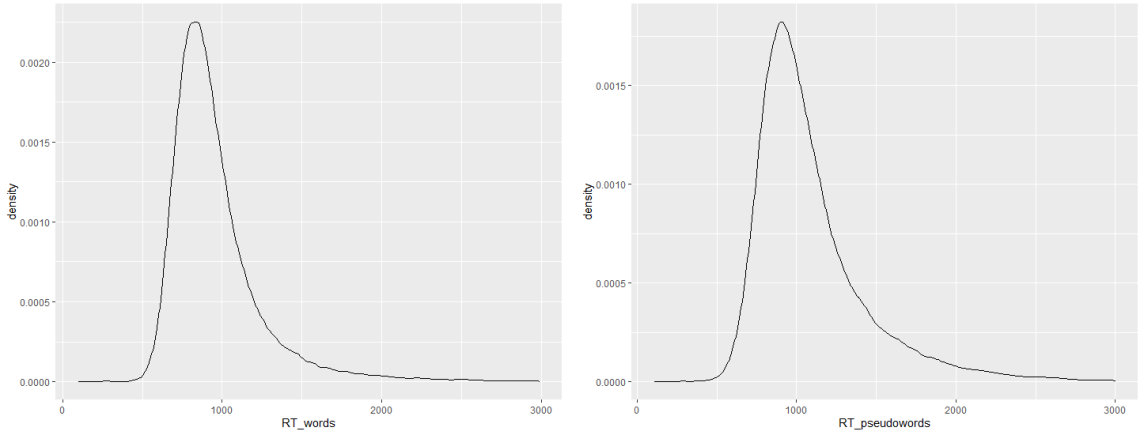

Fig 4.1 Density of the original RT distribution.

As given in Figure 4.1, it is clear that RTs in words and pseudowords are not normally distributed. From the graph above, we can see that the skewness is bigger than zero and the dataset is in the right-skewed distribution. We should use square root or log function to transform the data. Here, we simply use the sqrt function. The plot of the transformed data can be seen in Figure 4.2.

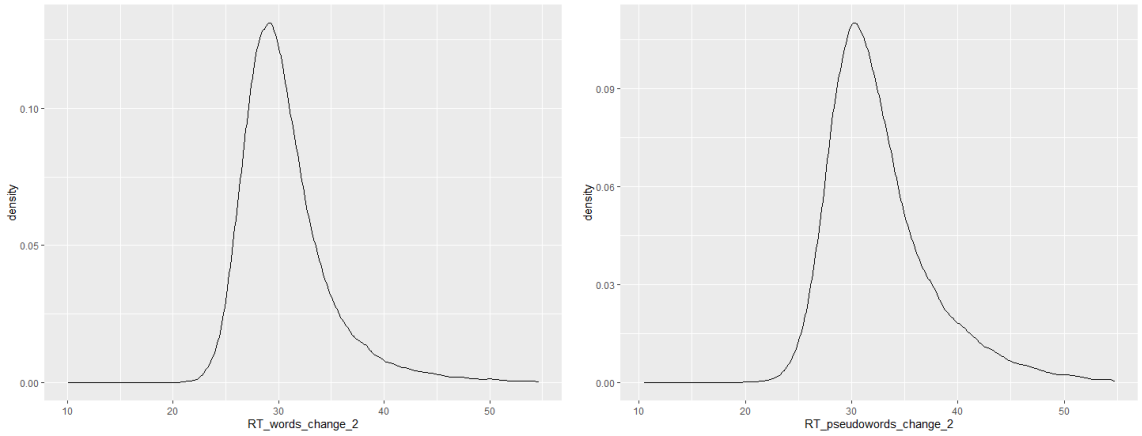

Fig 4.2 Density of the transformed RT distribution.

We use the two-tailed tests and get the following results. The t-value is -90.265, while the p-value is smaller than 2.2e-16. For this analysis, we used the two tailed tests because we were not sure which direction the difference between the groups would lean towards.

—

Q5:

Even if we get a normal distribution of underlying data, we still did not address the issue of independence. Are all RTs in our data set independent? Clearly not. Participants tend to differ in reaction times and so there will be dependence in reaction times of a participant. The way to avoid it is to not work with raw data but aggregations. Commonly when running a t-test on experimental data, we aggregate the dependent variable, e.g., RTs for words, per participant (that is, we get just one measure per participant, its mean RT over words). We do the same for pseudowords.

```
aggBySubjectAndIsWord <- MergeData %>%
  group_by(Subject, IsWord) %>%
  summarise(mean_RT = mean(RT, na.rm = TRUE))
```

Then, we calculate the t-value over these aggregated measures and then we calculate p-values. Do that for the dataset and report the results. Be careful in thinking about the type of t-test. Is this paired or unpaired?

When using paired t.test, the results are: t = -20.483, df = 220, p-value < 2.2e-16. To motivate our choice, paired sample t-tests are often used to compare scores within subjects, whereas unpaired compares the scores between subjects (independently from each other). Since each participant has completed RTs (at least it looks that way) for both pseudowords and words, the conditions in the experiment are not independent from each other. Therefore, a paired t-test is required to compare the data.

—

Q6:

This is an individual question. The answer is not made available here.

—

Q7:

Right above, we calculated p values based on the assumption that there are 40 degrees of freedom, corresponding to the collection of 42 data points (21 for word 15292 and 21 for pseudoword 15292; for each group the degrees of freedom are 21-1, which makes 40 degrees of freedom in total). In this way, we are behaving as if the number of data points was fixed and it was only open what the values of the data points was. However, it is quite common that experimentalists do not know in advance how many participants they want to collect. Imagine the following situation: we decided we would be collecting data for the whole day and then we will stop and check the results. It happens so that on that day, there was a 50% probability that we would collect 12 responses (6 for words, 6 for pseudowords) and a 50% probability that we would collect 42 data points (21 for words and 21 for

pseudowords). In our actual sample, we happened to collect the latter amount (i.e., 42 data points) and we got results as shown in word 15292 and pseudoword 15292. What would then be the p value? This is a slightly mind-boggling question so do not worry if you cannot answer it. If you cannot calculate the value, try to at least reason about this: do you think that the p-value will be smaller than in Q4 or bigger? In any case, note one very unintuitive aspect of p-values: they are dependent on experimenters' intentions and hypothetical situations (which might often not be explicitly stated, and might not even be implicitly considered!).

In theory, if the relative difference stays the same between the groups. With a higher DF, the critical T-value you need to achieve for a significant result will be lower than with a lower DF, which means it will more easily be breached. Therefore, a higher DF would most likely lead to a lower p-value. In this question, we can confirm that the lower degree of freedom leads to a higher p-value as given by Table 7.1.

```
t_15292 <- tcalculation(word_15292, pseudoword_15292)
[1] "t:"              "1.50136643517682"

dfValues <- c(10, 40)
for (dfVal in dfValues) {
  print(c("df:", dfVal, ";p:", 2*pt(t_15292, df=dfVal, lower.tail = FALSE)))
}
```

df	10	20	30	40
p	0.164159356725083	0.148884589081023	0.14371384083878	0.141114271877846

Table 7.1 The relation between the df and p-value given the same t-value.

With the given uncertainty of the participants in this experiment, we can come with the following expected p-value: 0.5 * 0.164159356725083 + 0.5 * 0.141114271877846 = 0.152636814301464

—

Q8:

Provide a graphical summary in which we can clearly see that IsWord affects RTs, ACC affects RTs, and the interaction of the two factors affects RTs.

In Figure 8.1, we see the difference of the mean between IsWord = True and IsWord = False. However, this difference is barely visible when IsWord = True and ACC = False as given in Figure 8.3. Meanwhile, for ACC the mean difference is still visible even when IsWord = False and ACC = True. Therefore, we can say that ACC has more effect on RTs than IsWord. Also, the

combination of IsWord = True and ACC = True gives more difference to the mean, which suggests that the two variables are correlated. Next to the mean difference, ACC seems to have an effect on the spread of the RTs.

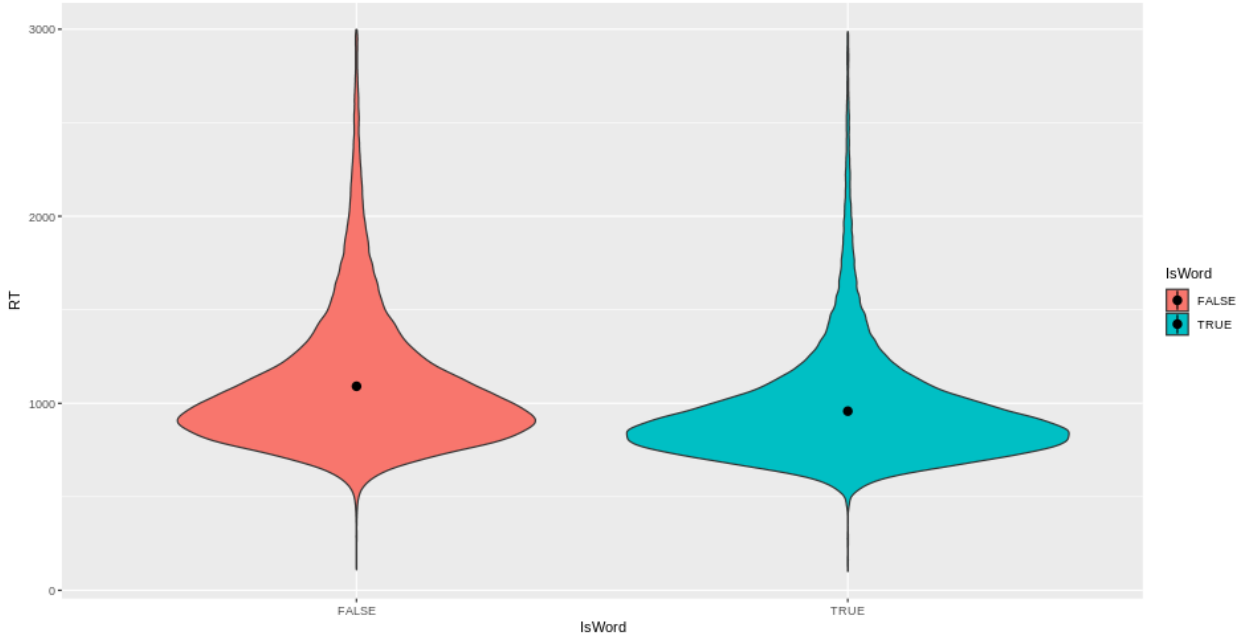

Fig 8.1 Graphical summary of the effects of IsWord on RTs.[1]

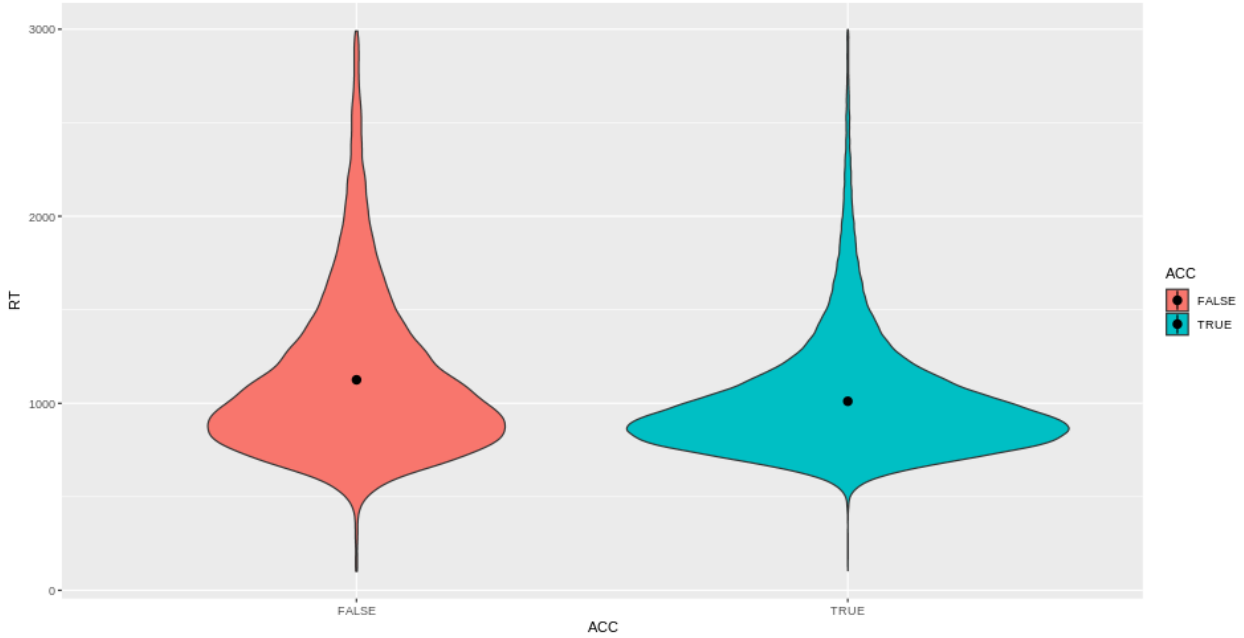

Fig 8.2 Graphical summary of the effects of ACC on RTs.

[1] The dot in the middle of each violin indicates the mean of the sample.

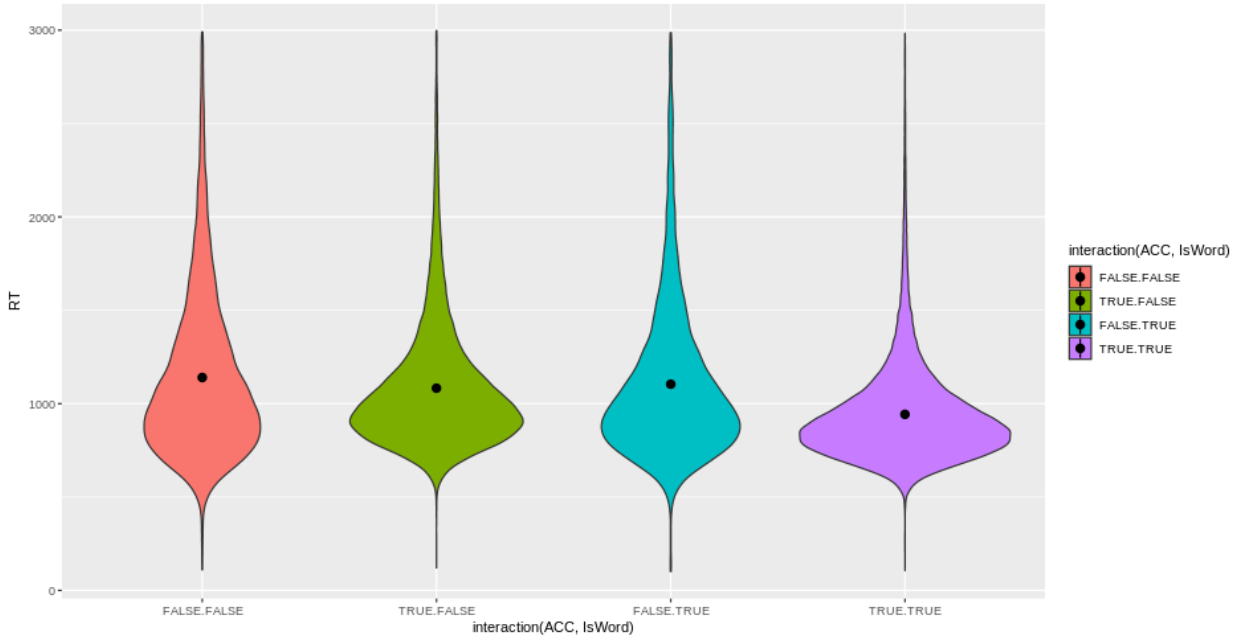

Fig 8.3 The violin plot showing the effects of ACC x IsWord on RTs.

—

Q9:

There are various sources of frequency in the corpus: FreCOCA, FreqGoogle, FreqSUBTLEX and FreqCOCAspok. Find out which of these provides the best fit of the model to the dependent variable log-transformed reaction times. You can do so by comparing models in which different frequency sources are added, or by comparing how big a proportion of the variance is explained by the model (see Baayen, chapter 6, if you do not know how).

To find the best fitting model, we started with analyzing the different frequencies as the single predictors followed by several combinations of these variables. The model with the highest R^2 should give us the best prediction. The formula to calculate this value is the reverse calculation of the model misfit or 'sum of squared errors' (SSE). Therefore, the lower the SSE is, the higher the R^2 value becomes (Winter, 2019, p. 76). Since the pseudowords are not real words, these data points will add noise to our models. Therefore, we analyzed only using the data out of the words dataset.

```
wordsBySubject <- MergeData %>%
  filter(IsWord == TRUE) %>%
  group_by(Subject)

summary(lm(log10(RT) ~ FreqCOCA, wordsBySubject))$r.squared
[1] 0.0006674872
```

summary(lm(log10(RT) ~ FreqGoogle, wordsBySubject))$r.squared
[1] 0.0008979794
summary(lm(log10(RT) ~ FreqSUBTLEX, wordsBySubject))$r.squared
[1] 0.000638437
summary(lm(log10(RT) ~ FreqCOCAspok, wordsBySubject))$r.squared
[1] 0.0007265849
summary(lm(log10(RT) ~ FreqCOCA + FreqGoogle, wordsBySubject))$r.squared
[1] 0.001096241
summary(lm(log10(RT) ~ FreqCOCA + FreqSUBTLEX, wordsBySubject))$r.squared
[1] 0.0007584633
summary(lm(log10(RT) ~ FreqCOCA + FreqCOCAspok, wordsBySubject))$r.squared
[1] 0.0007269443

Predictor(s)	R^2 based on log10(RT)
FreqCOCA	0.0006674872
FreqGoogle	0.0008979794
FreqSUBTLEX	0.000638437
FreqCOCAspok	0.0007265849
FreqCOCA + FreqGoogle	0.001096241
FreqCOCA + FreqSUBTLEX	0.0007584633
FreqCOCA + FreqCOCAspok	0.0007269443

Table 9.1 R^2 between log10(RT) and freq

For the individual predictors, the model with only FreqGoogle as the predictor gives the highest R^2 value. As for the multiple predictors, the combination of FreqCOCA and FreqGoogle gives the highest R^2 value. This might indicate that any combination of Freq variables with FreqGoogle would help improve the performance of the model.

Plot the relation between the dependent and the non-transformed independent variable to see whether any clear relation can be observed and whether the relation looks linear. Then, transform the frequency to log and plot again. Then, check the resulting model.

```
install.packages("cowplot")
library(cowplot)
ln_RT <- log(MergeData$RT,exp(1))
```

```
Freq_data <- data.frame(RT=ln_RT, FreqCOCA=MergeData$FreqCOCA,
FreqGoogle=MergeData$FreqGoogle, FreqSUBTLEX=MergeData$FreqSUBTLEX,
FreqCOCAspok=MergeData$FreqCOCAspok)

tempdata= select(filter(Freq_data,FreqCOCA != 0),c(FreqCOCA,RT))
g1 = ggplot(tempdata, aes( x = FreqCOCA, y = RT, )) + geom_point(size=1) +
geom_smooth(method = 'lm', formula = y ~ x, se = T) +
theme_minimal_grid(font_size = 12) + coord_fixed(ratio=10)
tempdata$FreqCOCA <- log(tempdata$FreqCOCA,exp(1))
g2 = ggplot(tempdata, aes( x = FreqCOCA, y = RT, )) + geom_point(size=1) +
geom_smooth(method = 'lm', formula = y ~ x, se = T) +
theme_minimal_grid(font_size = 12) + labs(x="ln(FreqCOCA)") +
coord_fixed(ratio=10)
m_logFreqCOCA = lm(RT ~ FreqCOCA, tempdata)
plot_grid(g1,g2,align = "h")
cat("R.Squared of ln(FreqCOCA):",summary(m_logFreqCOCA)$r.squared)
```

R.Squared of ln(FreqCOCA): 0.05448166

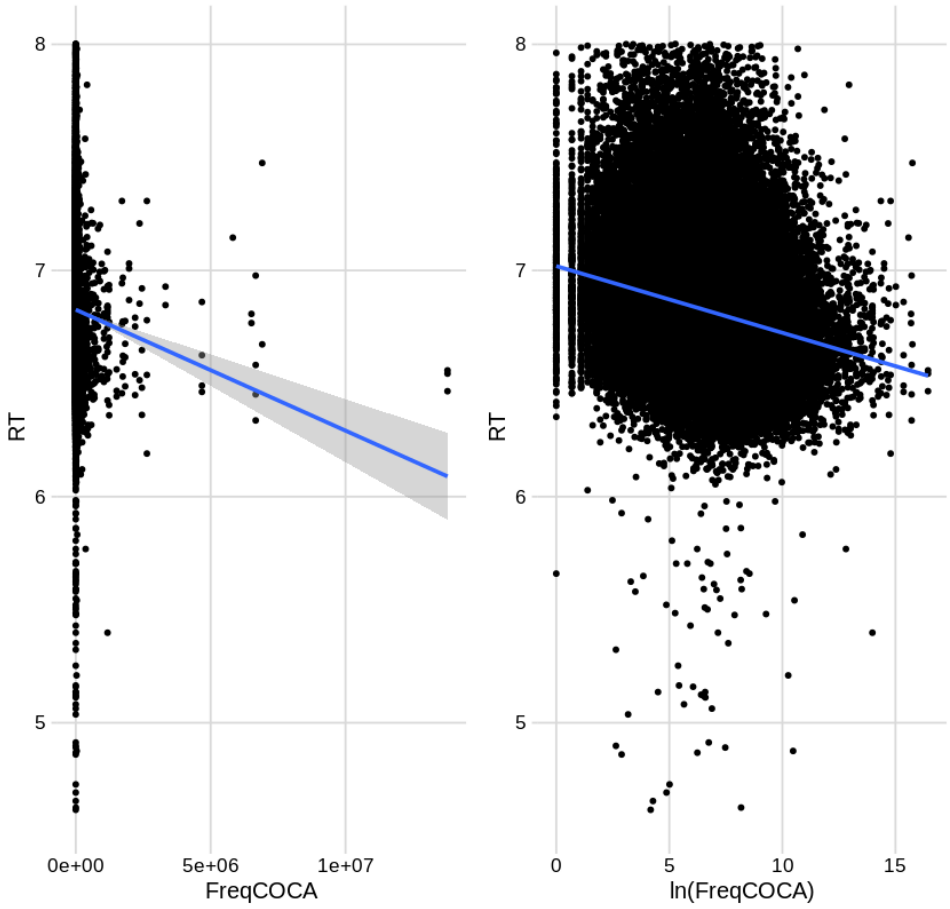

Fig 9.1 Linear model of FreqCOCA and ln(FreqCOCA)

```
tempdata= select(filter(Freq_data,FreqGoogle != 0),c(FreqGoogle,RT))
g1 = ggplot(tempdata, aes( x = FreqGoogle, y = RT, )) + geom_point(size=1)
+ geom_smooth(method = 'lm', formula = y ~ x, se = T) +
theme_minimal_grid(font_size = 12) + coord_fixed(ratio=10)
tempdata$FreqGoogle <- log(tempdata$FreqGoogle,exp(1))
g2 = ggplot(tempdata, aes( x = FreqGoogle, y = RT, )) + geom_point(size=1)
+ geom_smooth(method = 'lm', formula = y ~ x, se = T) +
theme_minimal_grid(font_size = 12) + labs(x="ln(FreqGoogle)") +
coord_fixed(ratio=10)
m_logFreqGoogle = lm(RT ~ FreqGoogle, tempdata)
plot_grid(g1,g2,align = "h")
cat("R.Squared of ln(FreqGoogle):",summary(m_logFreqGoogle)$r.squared)
```

R.Squared of ln(FreqGoogle): 0.05236818

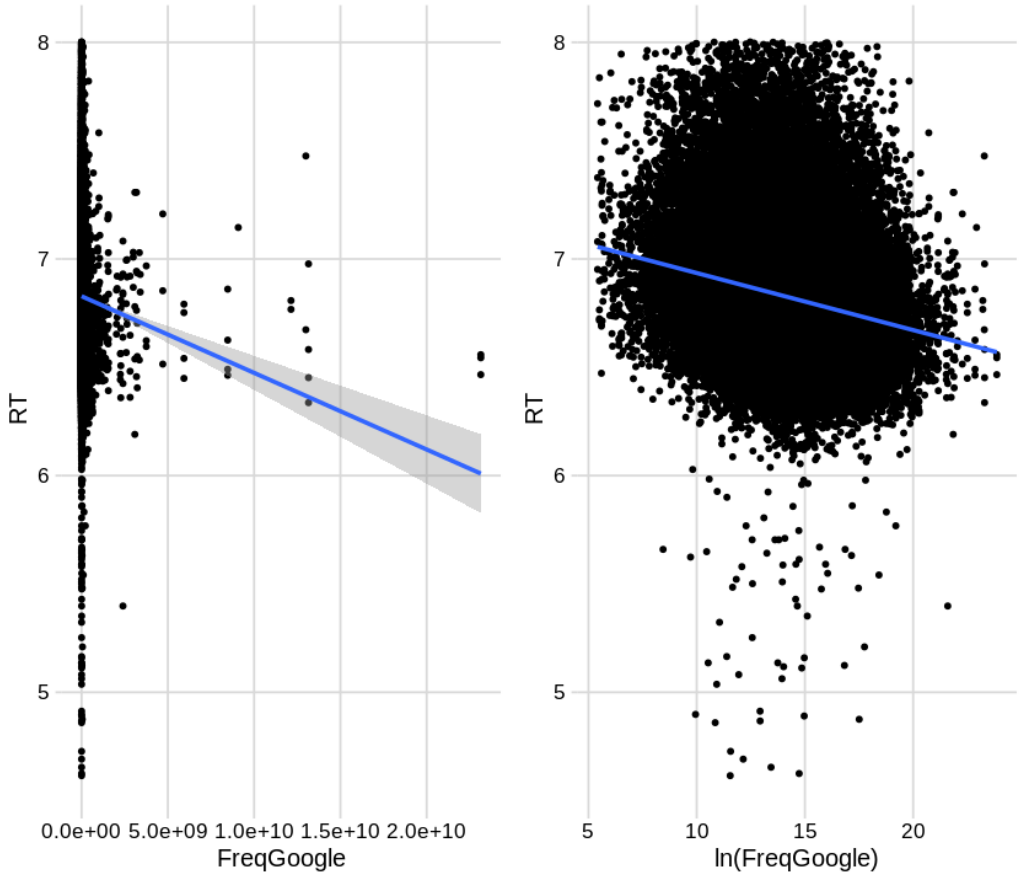

Fig 9.2 Linear model of FreqGoogle and ln(FreqGoogle)

```
tempdata= select(filter(Freq_data,FreqSUBTLEX != 0),c(FreqSUBTLEX,RT))
g1 = ggplot(tempdata, aes( x = FreqSUBTLEX, y = RT, )) +
geom_point(size=1) + geom_smooth(method = 'lm', formula = y ~ x, se = T) +
theme_minimal_grid(font_size = 12) + coord_fixed(ratio=10)
tempdata$FreqSUBTLEX <- log(tempdata$FreqSUBTLEX,exp(1))
g2 = ggplot(tempdata, aes( x = FreqSUBTLEX, y = RT, )) +
geom_point(size=1) + geom_smooth(method = 'lm', formula = y ~ x, se = T) +
theme_minimal_grid(font_size = 12) + labs(x="ln(FreqSUBTLEX)") +
coord_fixed(ratio=10)

m_logFreqSUBTLEX = lm(RT ~ FreqSUBTLEX, tempdata)
```

```
plot_grid(g1,g2,align = "h")
cat("R.Squared of ln(FreqSUBTLEX):",summary(m_logFreqSUBTLEX)$r.squared)
```

R.Squared of ln(FreqSUBTLEX): 0.05194333

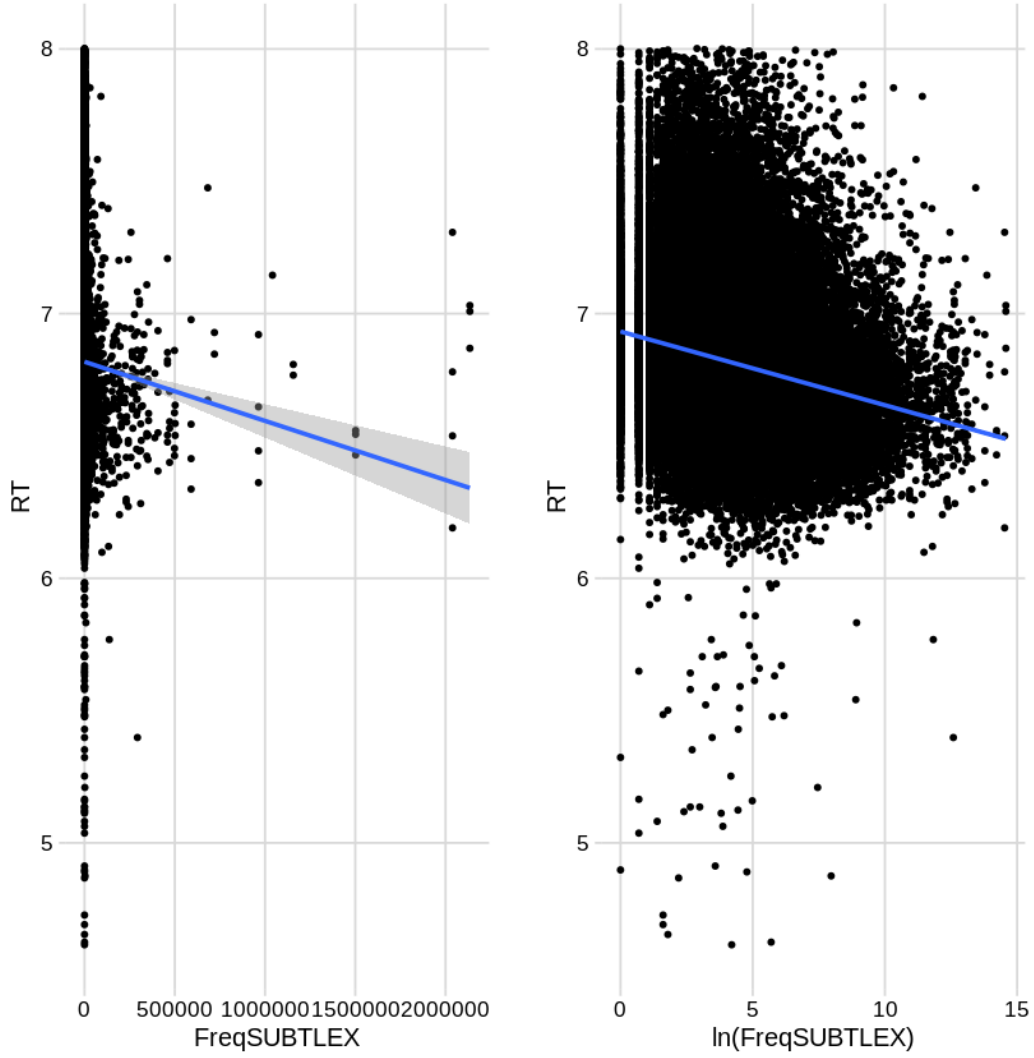

Fig 9.3 Linear model of FreqSUBTLEX and ln(FreqSUBTLEX)

```
tempdata= select(filter(Freq_data,FreqCOCAspok != 0),c(FreqCOCAspok,RT))
g1 = ggplot(tempdata, aes( x = FreqCOCAspok, y = RT, )) +
geom_point(size=1) + geom_smooth(method = 'lm', formula = y ~ x, se = T) +
theme_minimal_grid(font_size = 12) + coord_fixed(ratio=10)
tempdata$FreqCOCAspok <- log(tempdata$FreqCOCAspok,exp(1))
g2 = ggplot(tempdata, aes( x = FreqCOCAspok, y = RT, )) +
geom_point(size=1) + geom_smooth(method = 'lm', formula = y ~ x, se = T) +
```

```
theme_minimal_grid(font_size = 12) + labs(x="ln(FreqCOCAspok)") +
coord_fixed(ratio=10)
m_logFreqCOCAspok = lm(RT ~ FreqCOCAspok, tempdata)
plot_grid(g1,g2,align = "h")
cat("R.Squared of ln(FreqCOCAspok):",summary(m_logFreqCOCAspok)$r.squared)
```

R.Squared of ln(FreqCOCAspok): 0.04059883

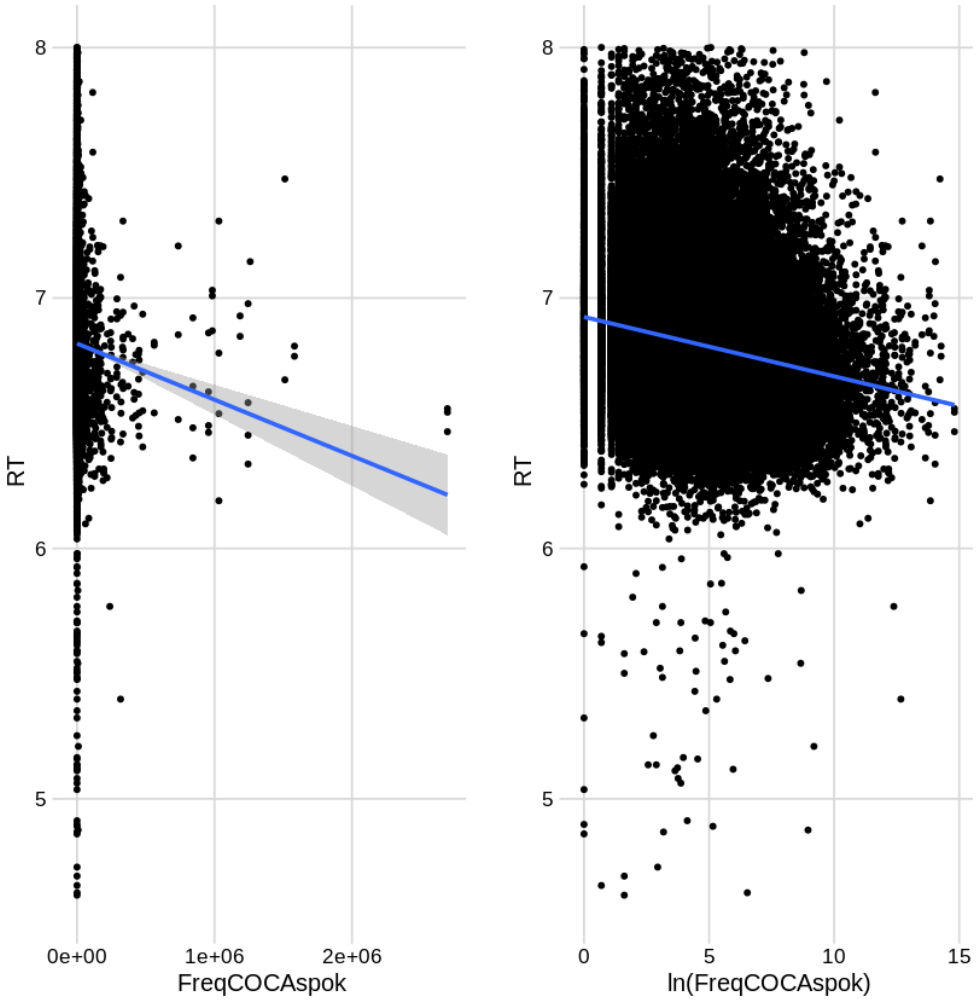

Fig 9.4 Linear model of FreqCOCAspok and ln(FreqCOCAspok)

Finally, check whether the log-transformation of all frequency data sets changes your previous answer. Which corpus of frequency is now the best predictor when we consider its log-transformation?

Predictor(s)	R^2 based on ln(RT)
ln(FreqCOCA)	0.05448166
ln(FreqGoogle)	0.05236818
ln(FreqSUBTLEX)	0.05194333
ln(FreqCOCAspok)	0.04059883

Table 9.2 R^2 between ln(RT) and ln(freq)

According to the graphs and the table above, the log-frequency of a word is a good predictor, better than a plain frequency, for log-transformed reaction times is correct. FreqCOCA now is the best predictor when we consider its log-transformation.

—

Q10:

Explore models with at least three predictors. All the predictors you use should be significantly different from 0 and only one of them should be a frequency (p values of each those should be smaller than .05 in the summary of the model). Explore more than one parameter and report the best fitting model that you found. Discuss briefly what variables you used in your model. Did it make an intuitive sense that such variables affect reaction times? Make sure to transform reaction times: you can use log-transformation. Furthermore, you might also see that some predictors are missing or their values are non-sensical in case of some observations. In that case, remove those cases from your consideration and the final model. You don't need to find the best possible model but you should be able to get a model whose adjusted R^2 is greater than 0.1 (i.e., a case in which at least 10% of the variance in the data has been explained by the model).

We explored a wide variety of predictor combinations. We attempted two strategies; One meant selecting predictors manually based on what seems to make sense and picking them manually, the other being a step-wise regression. The step-wise regression method produced suboptimal results and as a result, we selected variables manually.

In order to compare the models, we inspected each R-squared as an indicator of how well the model performed. Below are some of the results.

Parameters: NumPhones, NumSylls, FreqCOCA, Duration.

Intuitive sense: How a real word's complexity (NumPhones, NumSylls, Duration) affects reaction time.

```
tempdata = summarise(group_by(select(filter(MergeData,FreqCOCA != 0),c(Subject,RT,NumPhones,NumSylls,FreqCOCA,Duration)),NumPhones,NumSylls,Subject,FreqCOCA,Duration), RT = mean(RT))
tempdata$RT = log(tempdata$RT,exp(1))
```

```
tempdata$FreqCOCA = log(tempdata$FreqCOCA,exp(1))
m_temp <- lm(RT ~ NumPhones + NumSylls + FreqCOCA + Duration, tempdata)
print(summary(m_temp))
```

```
`summarise()` has grouped output by 'NumPhones', 'NumSylls', 'Subject',
'FreqCOCA'. You can override using the `.groups` argument.
Call:
lm(formula = RT ~ NumPhones + NumSylls + FreqCOCA + Duration,
    data = tempdata)

Residuals:
     Min       1Q   Median       3Q      Max
-2.28786 -0.15917 -0.03998  0.11220  1.34004

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  6.734e+00  5.312e-03 1267.716   <2e-16 ***
NumPhones   -9.076e-03  8.304e-04  -10.929   <2e-16 ***
NumSylls     3.155e-03  1.541e-03    2.048   0.0406 *
FreqCOCA    -2.277e-02  4.295e-04  -53.027   <2e-16 ***
Duration     5.088e-04  9.410e-06   54.068   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2468 on 85546 degrees of freedom
Multiple R-squared:  0.1001, Adjusted R-squared:    0.1
F-statistic:  2379 on 4 and 85546 DF,  p-value: < 2.2e-16
```

Parameters: NumPhones,PhonUP,PhonLev,PhonND,Duration,FreqCOCA.
Intuitive sense: How a real word's phonological features (Duration, NumPhones, PhonUP, PhonLev, PhonND) affects reaction time.

Duration: Duration of the item in milliseconds; Intuitively one can say that with a longer duration a participant has more time to recognize whether it is a word or pseudoword. While the original article (Tucker, 2019) noted that the word/pseudoword effect on durations is small (R2=0.00.17), it seems to significantly improve our model whenever we add it. It might be the case that instead a longer duration overall could prime an individual to react faster, regardless of whether it is a pseudoword or word.

Numphones: The number of phones in an item. One could say that an increased number of phones is related to the duration of the item (more phones, longer word) and as such would produce a similar effect as duration.

PhonUP: Phonological uniqueness point; intuitively one might say that the more unique a word is, the more attention one would pay attention to it, and as a result an individual may be reacting faster to it.

PhonLeve: Mean phone-level Levenshtein distance. While phone-level distance by itself may not be enough to really explain reaction times, particular distances with a longer duration might elevate one's attention and result in a faster RT.

PhonND: Number of phonological neighbors. Again, the number of phonological neighbors may not be very interesting in itself, but could add value when a particular number of neighbors are combined with for example the number of phones.

```
tempdata    =    summarise(group_by(select(filter(MergeData,FreqCOCA    !=
0),c(Subject,RT,Duration,NumPhones,PhonUP,PhonLev,PhonND,FreqCOCA)),Durati
on,NumPhones,PhonUP,PhonLev,PhonND,FreqCOCA), RT = mean(RT))
tempdata$RT = log(tempdata$RT,exp(1))
tempdata$FreqCOCA = log(tempdata$FreqCOCA,exp(1))
m_temp <- lm(RT ~ Duration + NumPhones + PhonUP + PhonLev + PhonND +
FreqCOCA, tempdata)
print(summary(m_temp))
```

```
`summarise()` has grouped output by 'Duration', 'NumPhones', 'PhonUP',
'PhonLev', 'PhonND'. You can override using the `.groups` argument.
Call:
lm(formula = RT ~ Duration + NumPhones + PhonUP + PhonLev + PhonND +
    FreqCOCA, data = tempdata)

Residuals:
     Min       1Q   Median       3Q      Max
-2.29273 -0.11471 -0.02182  0.08878  1.07565

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.740e+00  1.170e-02 575.906   <2e-16 ***
Duration     4.912e-04  1.185e-05  41.449   <2e-16 ***
NumPhones    3.303e-05  2.236e-03   0.015    0.988
PhonUP      -7.569e-03  8.912e-04  -8.493   <2e-16 ***
PhonLev      2.164e-03  3.317e-03   0.652    0.514
PhonND       5.748e-04  6.516e-05   8.822   <2e-16 ***
FreqCOCA    -2.341e-02  5.668e-04 -41.302   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.1723 on 26543 degrees of freedom
Multiple R-squared:  0.1887, Adjusted R-squared:  0.1885
F-statistic:  1029 on 6 and 26543 DF,  p-value: < 2.2e-16
```

Parameters: NumPhones,StressPattern,StressCat,Duration,FreqCOCA.
Intuitive sense: How a real word's stress pattern (NumPhones, StressPattern, StressCat, Duration) affects reaction time.

```
tempdata    =    summarise(group_by(select(filter(MergeData,FreqCOCA    !=
0),c(Subject,RT,NumPhones,StressPattern,StressCat,Duration,FreqCOCA)),NumP
hones,StressPattern,StressCat,Duration,Subject,FreqCOCA), RT = mean(RT))
tempdata$RT = log(tempdata$RT,exp(1))
tempdata$FreqCOCA = log(tempdata$FreqCOCA,exp(1))
m_temp <- lm(RT ~ NumPhones + StressPattern + StressCat + Duration +
FreqCOCA, tempdata)
print(summary(m_temp))
```

```
`summarise()` has grouped output by 'NumPhones', 'StressPattern',
'StressCat',
'Duration', 'Subject'. You can override using the `.groups` argument.
Call:
lm(formula = RT ~ NumPhones + StressPattern + StressCat + Duration +
    FreqCOCA, data = tempdata)

Residuals:
     Min       1Q   Median       3Q      Max
-2.28866 -0.15911 -0.04004  0.11221  1.33667

Coefficients:
                         Estimate Std. Error  t value Pr(>|t|)
(Intercept)             6.744e+00  6.583e-03 1024.513  < 2e-16 ***
NumPhones              -9.866e-03  6.240e-04  -15.811  < 2e-16 ***
StressPattern           1.813e-10  7.070e-10    0.257    0.798
StressCatInitial        1.348e-03  3.249e-03    0.415    0.678
StressCatMedial         1.900e-02  3.787e-03    5.016  5.3e-07 ***
StressCatMultiple      -8.474e-03  1.027e-02   -0.825    0.409
StressCatNone           6.060e-02  3.694e-02    1.641    0.101
StressCatSecondaryOnly  7.424e-02  6.607e-02    1.124    0.261
Duration                5.087e-04  9.489e-06   53.609  < 2e-16 ***
FreqCOCA               -2.313e-02  4.328e-04  -53.430  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 0.2469 on 85525 degrees of freedom
  (19 observations deleted due to missingness)
Multiple R-squared:  0.1005, Adjusted R-squared:  0.1004
F-statistic:  1061 on 9 and 85525 DF,  p-value: < 2.2e-16
```

References

Winter, B. (2019). *Statistics for linguists: An introduction using R*. Routledge.

Tucker, B. V., Brenner, D., Danielson, D. K., Kelley, M. C., Nenadić, F., & Sims, M. (2019). The massive auditory lexical decision (MALD) database. *Behavior research methods*, *51*(3), 1187-1204.