Continuous variables

WiSe23/24

Daniela Palleschi

2023-10-10

Table of contents

Learning Objectives

Today we will learn…

- what multiple regression is
- how to include multiple predictor variables
- how to interpret slopes in multiple regression

- how to interpret interaction effects
- about the assumption of the absence of collinearity

Set-up environment

```
# suppress scientific notation
options(scipen=999)
```

```
# load packages
pacman::p_load(
               tidyverse,
               here,
               broom,
               lme4,
               janitor,
               languageR)
```

Load data

- we'll use the full dataset of the frequency data.

```
df_freq_full <-
  read_csv(here("data", "ELP_full_length_frequency.csv")) |>
  clean_names() |>
  mutate(freq = 10^(log10freq), # inverse log10
         freq_log = log(freq)) |>  # use natural logarithm
  relocate(word, rt, length, freq, freq_log)
```

- the variable `log10freq` is a remnant from Winter (2019); we'll use the natural logarithm

Multiple regression

- simple linear models do not differ so greatly from a one-sample *t*-test (intercept-only model) or two-sample *t*-test (for a categorical predictor), or Pearson's *r* (for a standardised continuous predictor)
- why then should we bother with linear regression?
  - it allows us to include *multiple* predictors in our models simultaneously

    * still boils down to modeling the mean, but while conditioning the mean on multiple variables at once

Extending the equation of a line

- the equation of a line (Equation 1):
  - value of y = the intercept (b_0) + the corresponding value of x multiplied by the slope ($b_1 x$) + the error (residuals (e))
- In multiple regression, we can include more than one slope (Equation 2)

$$y = b_0 + b_1 x + e \tag{1}$$

$$y = b_0 + b_1 x + b_2 x + ... + e \tag{2}$$

One predictor

Let's re-run our simple model with this dataset. Let's keep reaction times in the raw milliseconds for now for interpretability.

```
fit_freq_full <-
  lm(rt ~ log(freq), data = df_freq_full)
```

```
tidy(fit_freq_full)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic p.value
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)    907.      1.09       828.        0
2 log(freq)      -37.5     0.262     -143.        0
```

We see there is a decrease in reaction times (-37.5 milliseconds) for a 1-unit increase in log frequency. Let's look at the model fit using `glance()`.

```
glance(fit_freq_full)$r.squared
```

```
[1] 0.3834186
```

- $R^2 = 0.383$

  - our model describes 38% of the variance in response times

- is this described variance due solely to frequency?
  - Other effects that are correlated with frequency might be conflating the frequency effect
  - Let's expand our model to include word length Equation 3

$$y = b_0 + b_1 * log\ frequency + b_2 * word\ length \tag{3}$$

Adding a predictor

- add `length` as a predictor

```
fit_freq_mult <-
  lm(rt ~ log(freq) + length, data = df_freq_full)
```

```
tidy(fit_freq_mult) |> select(term, estimate)
```

```
# A tibble: 3 x 2
  term        estimate
  <chr>          <dbl>
1 (Intercept)    748.
2 log(freq)      -29.5
3 length          19.5
```

- length is also a significant predictor of reaction times
  - an increase in word length (+1 letter) corresponds to a 20ms increase in reaction times
  - our intercept is now 748ms, instead of 907ms
  - 907ms corresponds to the prediction for reaction times to a word with 0 log frequency and 0 word length, but this is not very interpretable. If we were to center both predictors, the intercept would be the reaction time for a wrd with average frequency and average length.

The slope for log frequency has also changed: from -37.5 to -29.5. This change tells us that some of the effect in our first model was confounded with length, as controlling for length weakens the effect of frequency.

```
glance(fit_freq_mult)$r.squared
```

```
[1] 0.4872977
```

We also see that including `length` increases the variance described by our model, reflected in the R-squared values (0.4872977 instead of 0.3834186.

Standardising our predictors

Recall that, when we have multiple continuous predictors, standardising them can help their interpretation, as their slopes are comparable. We could achieve this by centering each variable and then dividing by the standard deviation, or we could use the `scale()` function, which does just this.

```
# centre and then standardize
df_freq_full |>
  mutate(
         freq_z1 = (freq-mean(freq))/sd(freq),
         freq_z2 = scale(freq)) |>
  select(freq_z1, freq_z2) |>
  head()
```

```
# A tibble: 6 x 2
  freq_z1 freq_z2[,1]
    <dbl>       <dbl>
1 -0.0902     -0.0902
2 -0.0864     -0.0864
3 -0.0905     -0.0905
4 -0.0864     -0.0864
5 -0.0885     -0.0885
6 -0.0901     -0.0901
```

Let's use `scale()` for `freq` and `length`.

```
df_freq_full <-
  df_freq_full |>
  mutate(freq_z = scale(freq_log),
         length_z = scale(length))
```

```
fit_freq_z <-
  lm(rt ~ freq_z + length_z, data = df_freq_full)
```

First, let's check the R^2:

```
glance(fit_freq_z)$r.squared
```

```
[1] 0.4872977
```

We see that our R^2 value is 0.4872977, just like above. This serves as a reminder that the predictors still represent the same variance in the underlying model, their units and scales have simply changed. What about our coefficients:

```
tidy(fit_freq_z) |> select(term, estimate)
```

```
# A tibble: 3 x 2
  term        estimate
  <chr>          <dbl>
1 (Intercept)    770.
2 freq_z         -60.6
3 length_z        43.3
```

Here, a 1-unit change always corresponds to a change of 1 standard deviation. Now we see that frequency has a larger magnitude than the effect of length. So, for each instease in frequency by 1 standard deviation (holiding length constant), reaction times decrease by 29.5 ms.

Adding an interaction term

We won't spent much time talking about interactions, but please check out Ch. 8 (Interations and nonlinear effects) in Winter (2019) for a more in-depth treatment. For now, what's important to know is that interactions describe how effects of one predictor may be influenced by changes in another predictor. We can add interactin terms of two predictors by connecting them with a colon (`:`).

```
lm(rt ~ freq_z + length_z + freq_z:length_z,
   data = df_freq_full) |>
  tidy() |> select(term, estimate)
```

```
# A tibble: 4 x 2
  term            estimate
  <chr>              <dbl>
1 (Intercept)        766.
```

```
2 freq_z             -63.9
3 length_z            41.8
4 freq_z:length_z    -11.4
```

Or, we can simply connect the two predictors with an asterisk (*) to indicate that we want to look at both predictors and their interaction.

```
lm(rt ~ freq_z*length_z,
   data = df_freq_full) |>
  tidy() |> select(term, estimate)
```

```
# A tibble: 4 x 2
  term            estimate
  <chr>              <dbl>
1 (Intercept)        766.
2 freq_z             -63.9
3 length_z            41.8
4 freq_z:length_z    -11.4
```

The model estimates are the same for both models. The intercept is the predicted reaction time for a word with the mean length and mean frequency. Notice that the interaction slope is negative, meaning when both `freq` and `length` increase, reaction times will decrease.

Model assumptions

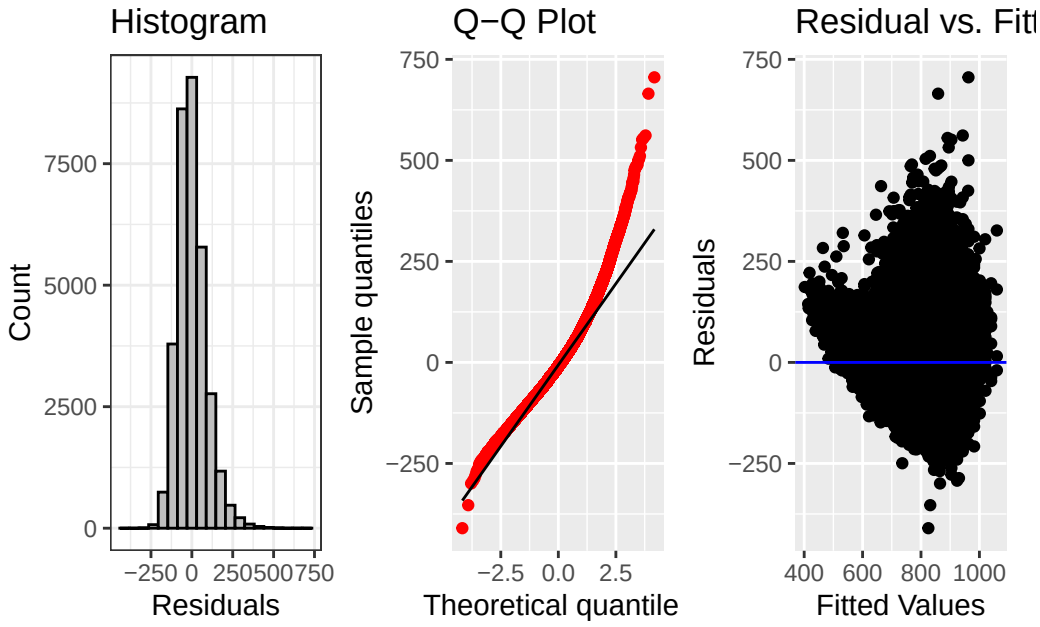
We've already discussed the assumptions of normality and homoscedasticity (constant variance), which both refer to the residuals of a model. We typically assess these assumptions visually, with histogram and Q-Q plots.

Normality and Homoscedasticity

For our model

```
fig_hist <-
fit_freq_z |>
  ggplot() +
  aes(x = .resid) +
  geom_histogram(bins = 20, fill = "grey", colour = "black") +
  theme_bw() +
```

```
  labs(title='Histogram', x='Residuals', y='Count')

fig_qq <-
fit_freq_z |>
  ggplot() +
  aes(sample = .resid) +
  geom_qq(colour = "red") +
  geom_qq_line() +
  labs(title='Q-Q Plot', x='Theoretical quantiles', y='Sample quantiles')

fig_res <-
  fit_freq_z |>
  ggplot() +
  aes(x = .fitted, y = .resid) +
  geom_point() +
  geom_hline(yintercept = 0, colour = "blue") +
  labs(title='Residual vs. Fitted Values Plot', x='Fitted Values', y='Residuals')

fig_hist + fig_qq + fig_res
```



The histogram looks approximately normally distributed, with a bit of a positive skew. The Q-Q plot suggests a less-normal distribution, with the model estimates fitting larger reaction times more poorly. The residual plot also shows that the variance of the residuals is not

constant, with much larger residual variance for larger fitted values. This tells us we should probably log reaction times. Let's try it all again, with log-transformed reaction times.

Log-transformed response variable

```
fit_freq_log_z <-
  lm(log(rt) ~ freq_z*length_z,
     data = df_freq_full)
```

```
glance(fit_freq_log_z)$r.squared
```

```
[1] 0.5176913
```

```
tidy(fit_freq_log_z) |> select(term, estimate)
```

```
# A tibble: 4 x 2
  term            estimate
  <chr>              <dbl>
1 (Intercept)      6.63
2 freq_z          -0.0826
3 length_z         0.0524
4 freq_z:length_z -0.00779
```

We see now that our values are much smaller, because they're on the log-scale.

```
exp(6.63 + -0.0826*5 + 0.0524*2)
```

```
[1] 556.5739
```

```
exp(6.63 + -0.0826*4 + 0.0524*2)
```

```
[1] 604.499
```

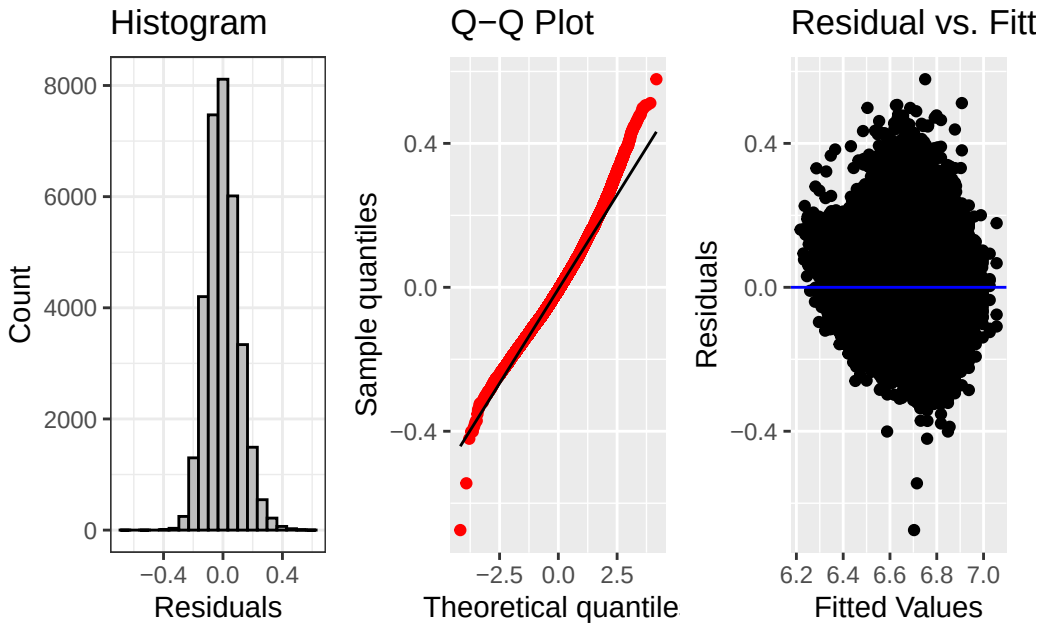
```
exp(6.63 + -0.0826*1 + 0.0524*6)
```

```
[1] 955.0847
```

```r
tidy(fit_freq_log_z)
```

```
# A tibble: 4 x 5
  term            estimate std.error statistic  p.value
  <chr>              <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)      6.63     0.000636   10428.   0
2 freq_z          -0.0826   0.000666    -124.   0
3 length_z         0.0524   0.000649      80.7  0
4 freq_z:length_z -0.00779  0.000581     -13.4  8.51e-41
```

```r
fig_hist <-
fit_freq_log_z |>
  ggplot() +
  aes(x = .resid) +
  geom_histogram(bins = 20, fill = "grey", colour = "black") +
  theme_bw() +
  labs(title='Histogram', x='Residuals', y='Count')

fig_qq <-
fit_freq_log_z |>
  ggplot() +
  aes(sample = .resid) +
  geom_qq(colour = "red") +
  geom_qq_line() +
  labs(title='Q-Q Plot', x='Theoretical quantiles', y='Sample quantiles')

fig_res <-
  fit_freq_log_z |>
  ggplot() +
  aes(x = .fitted, y = .resid) +
  geom_point() +
  geom_hline(yintercept = 0, colour = "blue") +
  labs(title='Residual vs. Fitted Values Plot', x='Fitted Values', y='Residuals')

fig_hist + fig_qq + fig_res
```



Looks much better.

Collinearity

Collinearity refers to when continuous predictor variables are correlated, which can make the interpretation of their coefficients difficult, and the results spurious. Regression assumes there is an *absence* of collinearity, i.e., our predictor variables are not correlatded.

To assess collinearity, you can use the `vif()` function from the `car` package to compare *variance inflation factors*. VIF values close to 1 indicates there is not a high degree of collinearity between your variables.

```
car::vif(fit_freq_log_z)
```

```
    freq_z       length_z freq_z:length_z
1.246509       1.184641       1.068283
```

Collinearity is a conceptual problem, and is something that you need to consider in the planning stage. Typically, we want to include predictors that we have specific predictions or research questions about. Shoving a bunch of predictors in a model to see what comes out significant is bad practice. Rather, we should have a principled approach to model building and variable selection. This is not to say that exploratory analyses should be avoided, but that this comes with caveats.

Adjusted R^2

Although we should avoid throwing any old predictor into our model, adjusted R^2 is a more conservative version of R^2 that takes into account the number of predictors in a model. For each additional predictor, adjusted R^2 includes the number of predictors (k) in its denominator (bottom half of a division), which means that the more predictors there are, the smaller R^2 will be, unless each additional predictor explains sufficient variance to counteract this penalisation.

```
glance(fit_freq_log_z)$adj.r.squared
```

```
[1] 0.5176475
```

If we were to look at the (adjusted) R^2 of our simple linear regression model, where log reaction times are predicted by standardised log frequency, we see that there is a large increase in our model which includes length and its interaction. This suggests that our model is not overfit, and that length contributes to the variance explained by the model.

```
glance(lm(log(rt) ~ freq_z, data = df_freq_full))$adj.r.squared
```

```
[1] 0.4148675
```

If we likewise compare to the same model without an interaction term (log reaction times ~ frequency * length), we see that the adjusted R^2 is not very different. If the adjusted R^2 were much lower, this would indicate that including the interaction term leads to overfitting.

```
glance(lm(log(rt) ~ freq_z + length_z, data = df_freq_full))$adj.r.squared
```

```
[1] 0.5150461
```

Important terms

Term	Definition	Equation/Code
NA	NA	NA

Learning Objectives

Today we learned...

Today we will learn...

- what multiple regression is
- how to include multiple predictor variables
- how to interpret slopes in multiple regression
- how to interpret interaction effects
- about the assumption of the absence of collinearity

Task

Load in the `english` dataset from the `languageR` package (Baayen & Shafaei-Bajestan, 2019) (code below). You don't need to load in any CSV file, because this dataset is available if you have the package loaded. From the manual:

> This data set gives mean visual lexical decision latencies and word naming latencies to 2284 monomorphemic English nouns and verbs, averaged for old and young subjects, with various predictor variables. (languageR manual, p. 29)

```
# load in 'english' dataset from languageR
df_freq_eng <-
  as.data.frame(english) |>
  dplyr::select(RTlexdec, RTnaming, Word, LengthInLetters, AgeSubject, WrittenFrequency) |
  rename(rt_lexdec = RTlexdec,
         rt_naming = RTnaming,
         freq_written = WrittenFrequency) |>
  clean_names() |>
  relocate(word)
```

We're keeping five variables:

- `word`: a factor with 2284 words
- `rt_lexdec`: numeric vector of log RT in visual lexical decision
- `rt_naming`: numeric vector of log RT in word naming
- `length_in_letters`: numeric vector with length of the word in letters
- `AgeSubject`: a factor with as levels the age group of the subject: young versus old.
- `freq_written`: numeric vector with log frequency in the CELEX lexical database

Take the following steps:

1. Perform an exploratory data analysis to understand the data.

Exploratory data analysis

Let's start with a summary of the data.

```
      word        rt_lexdec       rt_naming     length_in_letters age_subject
 arm    :   4   Min.   :6.205   Min.   :6.022   Min.   :2.000     old  :2284
 barge  :   4   1st Qu.:6.426   1st Qu.:6.149   1st Qu.:4.000     young:2284
 bark   :   4   Median :6.550   Median :6.342   Median :4.000
 bear   :   4   Mean   :6.550   Mean   :6.323   Mean   :4.342
 beef   :   4   3rd Qu.:6.653   3rd Qu.:6.490   3rd Qu.:5.000
 bind   :   4   Max.   :7.188   Max.   :6.696   Max.   :7.000
 (Other):4544
  freq_written
 Min.   : 0.000
 1st Qu.: 3.761
 Median : 4.832
 Mean   : 5.021
 3rd Qu.: 6.247
 Max.   :11.357
```

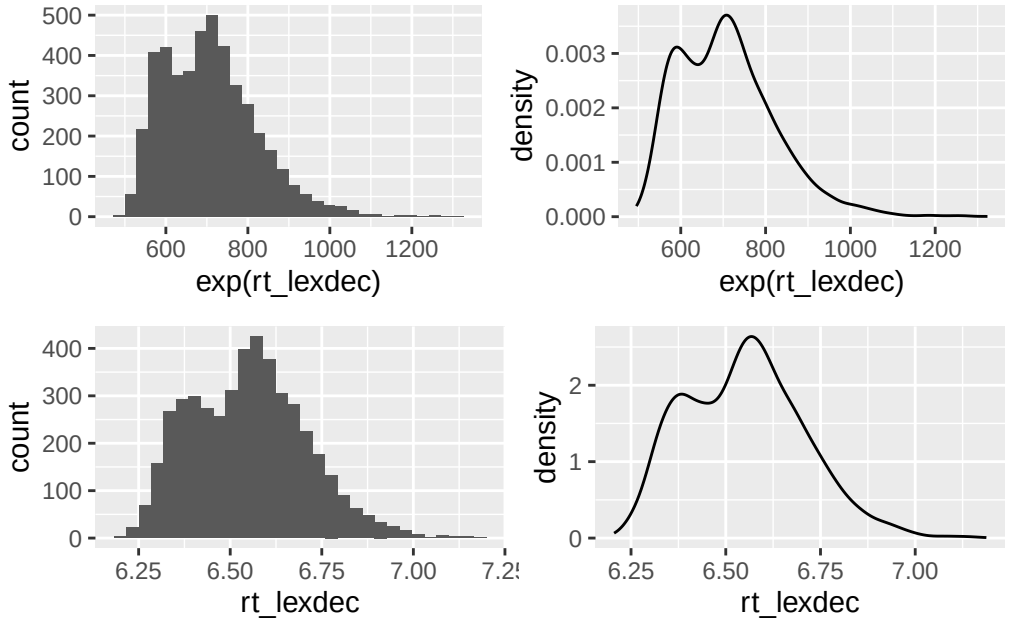
Remember that `word` is a factor, i.e., categorical variable. The numbers beside the words indicate how many observations (i.e., rows) there are per word. To see how many words have how many observations, we can use the `count()` function from the `dplyr` package, and then pipe to `count(n)`.

```
df_freq_eng |>
  count(word) |>
  count(n)
```

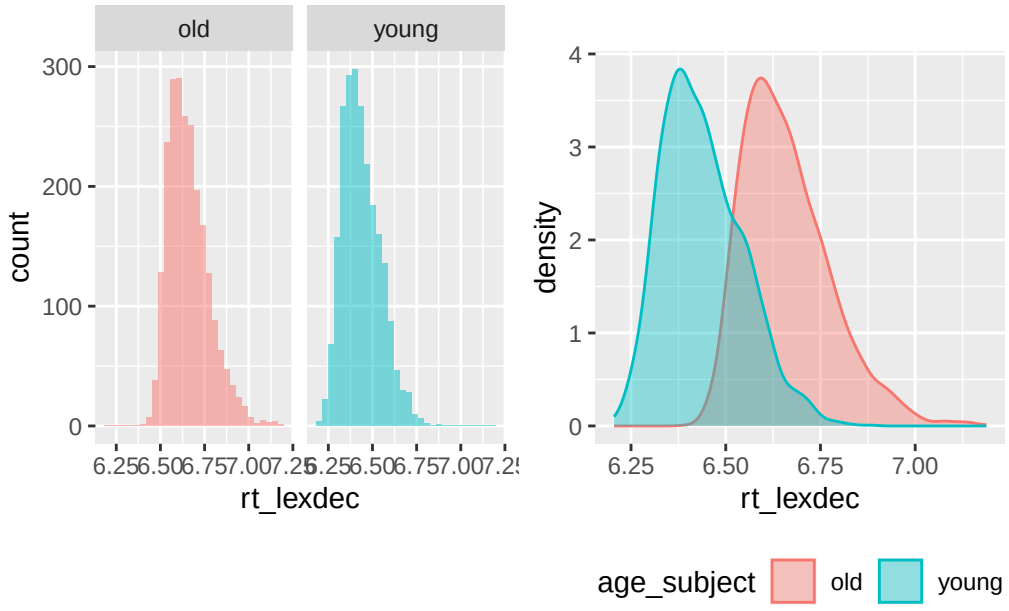
```
  n   nn
1 2 2110
2 4   87
```

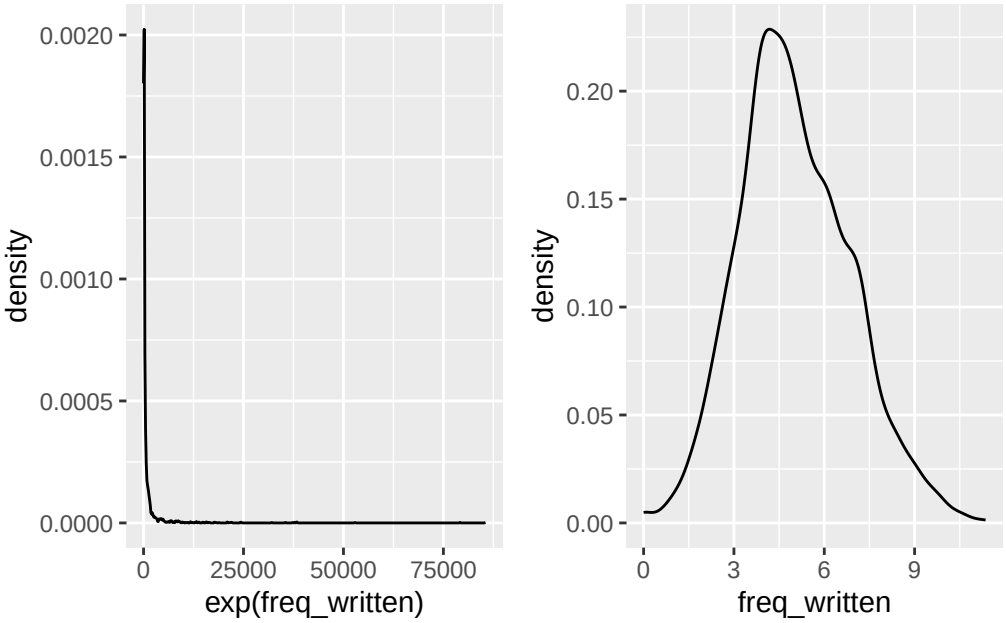
This tells us that there are 2110 words that have 2 observations, and 87 that have 4 observations.

Now let's look at the distribution of our raw reaction time data (which is already logged, so we feed it into `exp()`). We'll produce histograms and density plots of the raw and log reaction times.

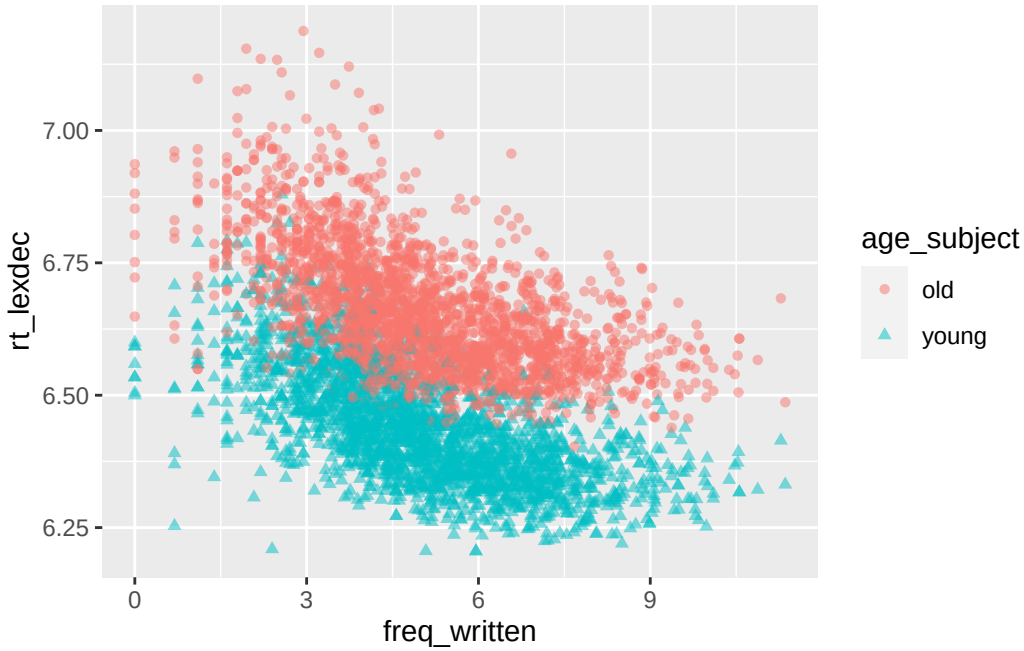


This looks like a *bimodal* distribution, i.e., there are two *modes* (most frequent value, i.e., peak in a histogram). What might be driving this? We know that there were two subject groups: old and young. How does the distribution of these two groups look?






```
df_freq_eng |>
  ggplot() +
  aes(x = freq_written, y = rt_lexdec, colour = age_subject, shape = age_subject) +
  geom_point(alpha = .5)
```



Model

Now let's run our model, ignoring `age_subject` for now.

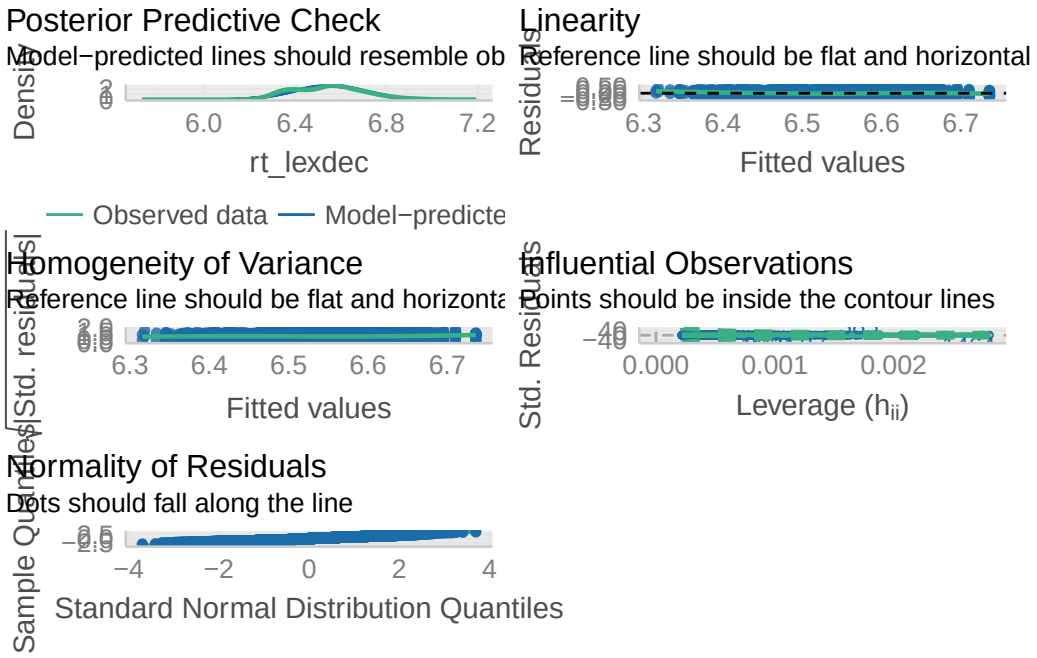
```
fit_freq_eng <-
  lm(rt_lexdec ~ freq_written, data = df_freq_eng)
```

```
summary(fit_freq_eng)
```

```
Call:
lm(formula = rt_lexdec ~ freq_written, data = df_freq_eng)

Residuals:
     Min       1Q   Median       3Q      Max
-0.45708 -0.11657 -0.00109  0.10403  0.56085

Coefficients:
             Estimate Std. Error t value            Pr(>|t|)
(Intercept)   6.735931   0.006067 1110.19 <0.0000000000000002 ***
freq_written -0.037010   0.001134  -32.63 <0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1413 on 4566 degrees of freedom
Multiple R-squared:  0.1891,    Adjusted R-squared:  0.1889
F-statistic:  1065 on 1 and 4566 DF,  p-value: < 0.00000000000000022
```

Let's check our model fit.

```
performance::check_model(fit_freq_eng)
```




```
tidy(fit_freq_eng)
```

```
# A tibble: 2 x 5
  term         estimate std.error statistic   p.value
  <chr>           <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)    6.74     0.00607    1110.   0
2 freq_written  -0.0370   0.00113     -32.6 4.46e-210
```

```
glance(fit_freq_eng)$r.squared
```

```
[1] 0.1890641
```

Literaturverzeichnis

Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing linguistic data: A practical introduction to statistics*. https://CRAN.R-project.org/package=languageR

Winter, B. (2019). Statistics for Linguists: An Introduction Using R. In *Statistics for Linguists: An Introduction Using R*. Routledge. https://doi.org/10.4324/9781315165547