Continuous variables

WiSe23/24

Daniela Palleschi

2023-10-10

Table of contents

This lecture is based on Ch. 5 (Correlation, Linear, and Nonlinear transformations) from Winter (2019).

Learning Objectives

Today we will learn…

- why and how to centre continuous predictors
- when and how to standardize continuous predictors
- why and how to log-transform continuous variables

Set-up environment

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

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

Load data

```
df_freq <- read_csv(here("data", "ELP_frequency.csv")) |>
  clean_names()
```

- Reminder of our variables:

```
summary(df_freq)
```

```
     word                freq               rt       
 Length:12          Min.   :    4.0   Min.   :507.4  
 Class :character   1st Qu.:   57.5   1st Qu.:605.2  
 Mode  :character   Median :  325.0   Median :670.8  
                    Mean   : 9990.2   Mean   :679.9  
                    3rd Qu.: 6717.8   3rd Qu.:771.2  
                    Max.   :55522.0   Max.   :877.5  
```

Linear transformations

- refer to constant changes across values that do not alter the relationship between these values
  - adding, subtracting, or multiplying by a constant value
- let's look at some common ways of linearly transforming our data, and the reasons behind doing so

Centering

- Centering is typically applied to predictor variables
  - subtracting the mean of a variable from each value
  - results in each centered value representing the original value's deviance from the mean (i.e., a mean-deviation score)
- What would a centered value of 0 represent in terms of the original values?

- let's centre our frequency values using the tidyverse

```r
# add centered variable with the tidyverse
df_freq <-
  df_freq |>
  mutate(freq_c = freq-mean(freq))
```

- and with base R (more verbose…)

```r
# add centered variable with base R
df_freq$freq_c <- df_freq$freq-mean(df_freq$freq)
```

- both code chunks produce the same result

```r
head(df_freq)
```

```
# A tibble: 6 x 4
  word       freq    rt freq_c
  <chr>     <dbl> <dbl>  <dbl>
1 thing     55522  622. 45532.
```

```
2 life      40629  520. 30639.
3 door      14895  507.  4905.
4 angel      3992  637. -5998.
5 beer       3850  587. -6140.
6 disgrace    409  705  -9581.
```

Centred predictor

- re-fit our model with and without centred predictor

```
# run our model with the original predictor
fit_rt_freq <-
  lm(rt ~ freq, data = df_freq)
```

```
# run our model with the centered predictor
fit_rt_freq_c <-
  lm(rt ~ freq_c, data = df_freq)
```

- what is the difference between the two models?

```
tidy(fit_rt_freq)
```

```
# A tibble: 2 x 5
  term         estimate std.error statistic       p.value
  <chr>           <dbl>     <dbl>     <dbl>         <dbl>
1 (Intercept) 714.       34.6          20.6   0.00000000160
2 freq         -0.00338   0.00170     -1.99 0.0746
```

```
tidy(fit_rt_freq_c)
```

```
# A tibble: 2 x 5
  term         estimate std.error statistic  p.value
  <chr>           <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept) 680.       30.2          22.5   6.71e-10
2 freq_c       -0.00338   0.00170     -1.99 7.46e- 2
```

- the intercept values: 713.706298 (uncentered) and 679.9166667 (centered)

- what does this correspond to?

```r
mean(df_freq$rt)
```

```
[1] 679.9167
```

- intercept with a single centered continuous predictor variable = the mean of a continuous response variable
- this is crucial in interpreting interaction effects, which we will discuss briefly tomorrow (for more: Chapter 8 in Winter (2019))

Standardizing (*z*-scoring)

- standardize continuous predictors
  - dividing centered values by the standard deviation of the sample
  - when to do this? if you have multiple continuous predictors (which we don't at present)

- what are our mean and standard deviation?

```r
mean(df_freq$freq)
```

```
[1] 9990.167
```

```r
sd(df_freq$freq)
```

```
[1] 18558.69
```

- what are the first six values of `freq` in the original scale and centred scale?

```r
df_freq$freq[1:6]
```

```
[1] 55522 40629 14895  3992  3850   409
```

```r
df_freq$freq_c[1:6]
```

```
[1] 45531.833 30638.833  4904.833 -5998.167 -6140.167 -9581.167
```

- standardise z-scores for frequency by dividing these centered values by the standard deviation of `freq`
  - Again, this can be done with `mutate()` from `dplyr`, or by using base R syntax.

```r
# standardise using the tidyverse
df_freq <-
  df_freq |>
  mutate(freq_z = freq_c/sd(freq))
```

```r
# standardize with base R
df_freq$freq_z <- df_freq$freq_c/sd(df_freq$freq)
```

```r
df_freq |>
  select(freq, freq_c, freq_z) |>
  head()
```

```
# A tibble: 6 x 3
   freq freq_c freq_z
  <dbl>  <dbl>  <dbl>
1 55522 45532.  2.45
2 40629 30639.  1.65
3 14895  4905.  0.264
4  3992 -5998. -0.323
5  3850 -6140. -0.331
6   409 -9581. -0.516
```

Non-linear transformations

- the meat and potates of dealing with continuous variables (depending on your subfield)
- in linguistic research, and especially experimental research, we often deal with continuous variables truncated/bound at 0

  - Reaction times, reading times and formant frequencies: there is (typically) no such thing as a negative reading time or fundamental frequency

- these types of data are almost never normally distributed, typically having a 'positive skew' (long tail to the right)
  - this has implications for the normality of residuals fit to a straight line
  - these very large, exceptional values will have a stronger influence on the line of best fit, leading to the coefficient estimates that are "suboptimal for the majority of data points" [@Baayen (2008); p. 92]
- How do we deal with this nonnormality?
  - We use non-linear transformations, the most common of which is the log-transformation

Log-transformation

- luckily, we can easily log-transform continuous values by passing them through the `log()` function
  - this uses the *natural* logarithm, which finds the power to which Euler's number (e = `2.718281828459`) is raised to equal x (don't worry about the math)
- importantly, log-transforming a continuous variable makes numbers smaller, with a larger shrinkage for larger numbers

---

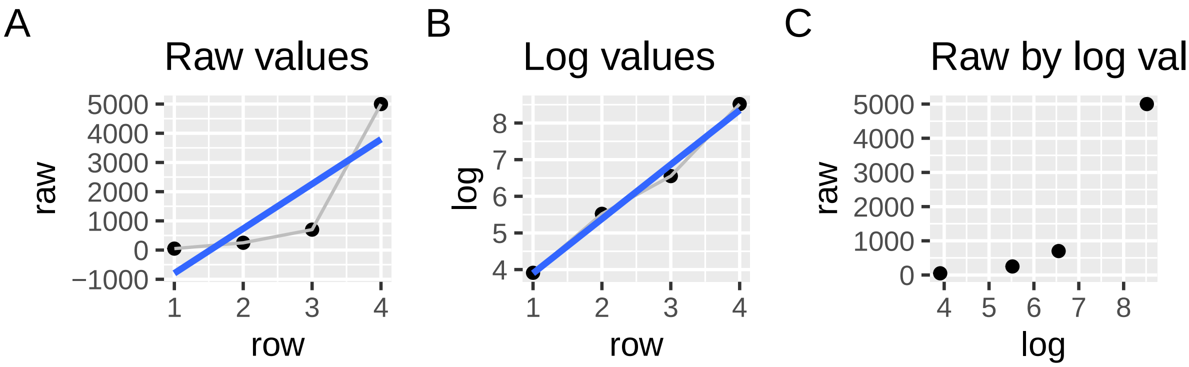
- let's see how log-transformation changes values

```
raw_values <-
  tibble(
    row = 1:4,
    raw = c(50, 250, 700, 5000),
    log = log(raw))

raw_values
```

```
# A tibble: 4 x 3
    row   raw   log
  <int> <dbl> <dbl>
1     1    50  3.91
2     2   250  5.52
```

```
3     3   700  6.55
4     4  5000  8.52
```

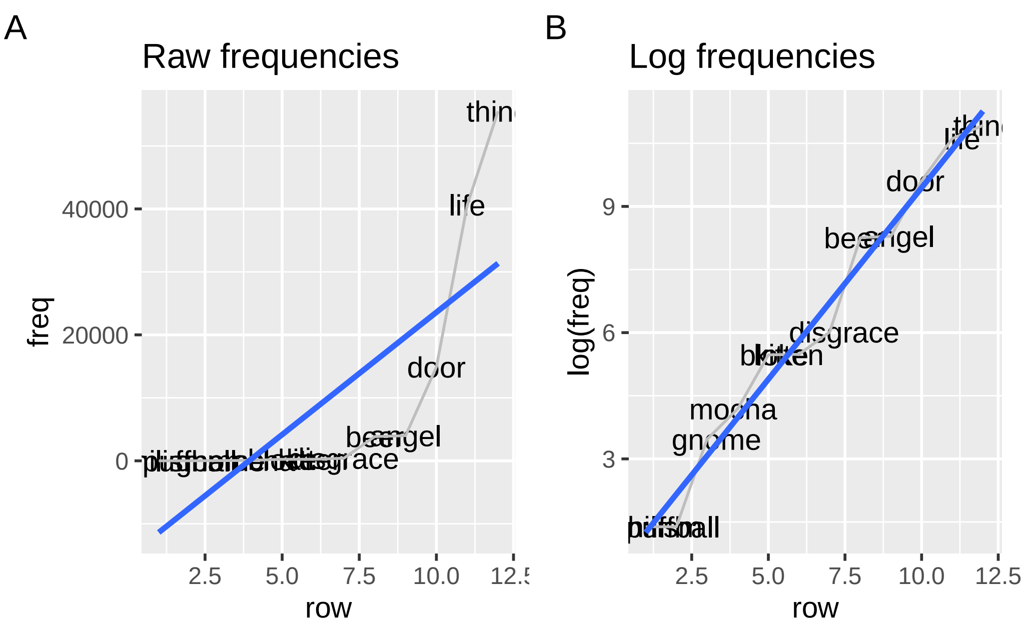
```
fig_raw <-
  raw_values |>
  ggplot() +
  aes(x = row, y = raw) +
  labs(title = "Raw values") +
  geom_point() +
  geom_line(colour = "grey") +
  geom_smooth(method = "lm", se = F)


fig_log <-
raw_values |>
  ggplot() +
  aes(x = row, y = log) +
  labs(title = "Log values") +
  geom_point() +
  geom_line(colour = "grey") +
  geom_smooth(method = "lm", se = F)


fig_log_raw <-
  raw_values |>
  ggplot() +
  aes(x = log, y = raw) +
  labs(title = "Raw by log values") +
  geom_point()

library(patchwork)

fig_raw + fig_log + fig_log_raw + plot_annotation(tag_levels = "A")
```



Log word frequency

```
df_freq$freq
```

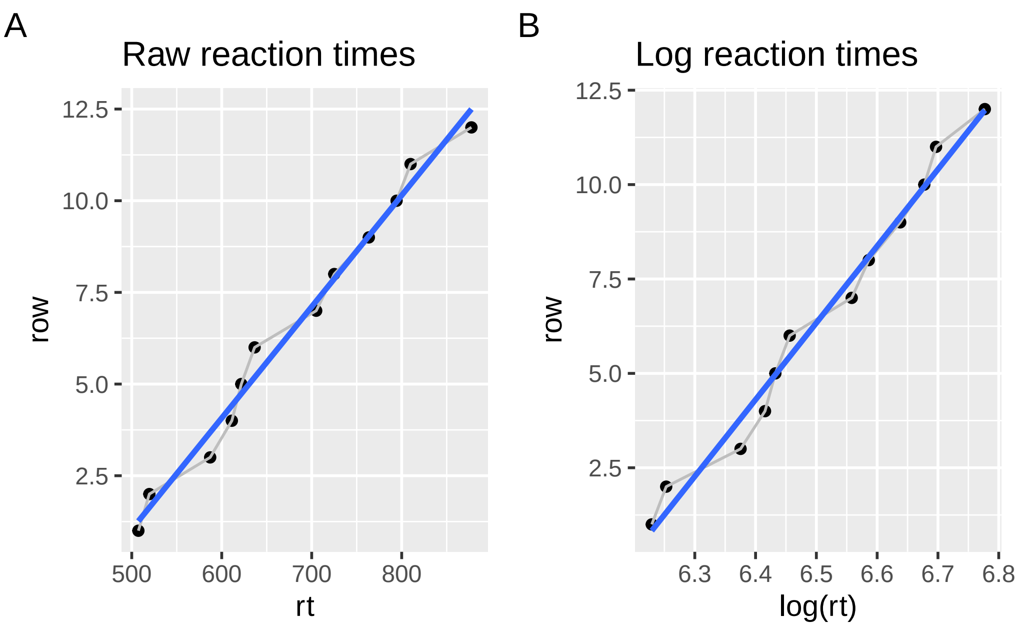
```
[1] 55522 40629 14895  3992  3850   409   241   238    66    32     4     4
```



Log reaction times

```
df_freq |>
  mutate(log_rt = log(rt)) |>
  arrange(rt) |>
  select(rt, log_rt)
```

```
# A tibble: 12 x 2
      rt log_rt
   <dbl>  <dbl>
 1  507.   6.23
 2  520.   6.25
 3  587.   6.38
 4  611.   6.42
 5  622.   6.43
 6  637.   6.46
 7  705    6.56
 8  725.   6.59
 9  764.   6.64
10  794.   6.68
11  810.   6.70
12  878.   6.78
```



Model with log-transformed variables

```
df_freq <-
  df_freq |>
    mutate(rt_log = log(rt),
           freq_log = log(freq),
           freq_log_c = freq_log - mean(freq_log))
```

```
fit_log <- lm(rt_log ~ freq_log_c, data = df_freq)
```

```
# or, log-transform directly in the model syntax
fit_log <- lm(log(rt) ~ freq_log_c, data = df_freq)
```

```
tidy(fit_rt_freq_c)
```

```
# A tibble: 2 x 5
  term          estimate std.error statistic  p.value
  <chr>            <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept) 680.        30.2         22.5  6.71e-10
2 freq_c        -0.00338   0.00170     -1.99 7.46e- 2
```

```
tidy(fit_log)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   6.51     0.0277     235.   4.72e-20
2 freq_log_c   -0.0453   0.00871     -5.20 4.03e- 4
```

- what has changed?

Extracting predictions

- the inverse of the log is the exponential

```
log(2)
```

```
[1] 0.6931472
```

```r
exp(0.6931472)
```

```
[1] 2
```

- we can plus our equation of a line into the `exp()` function to extract predictions
  - what's our predicted reaction time for the word *door*?

$$

y_i & = b_0 + b_1x_i y_i & = b_0 + b_1*freq(door)

$$

```r
b0 <- coef(fit_log)['(Intercept)']
b1 <- coef(fit_log)['freq_log_c']
freq_door <-
  df_freq |> filter(word == "door") |> select(freq_log_c)
```

```r
b0 + b1*freq_door
```

```
  freq_log_c
1   6.356246
```

```r
exp(b0 + b1*freq_door)
```

```
  freq_log_c
1   576.0795
```

```r
predict(fit_log)
```

```
       1        2        3        4        5        6        7        8 
6.296675 6.310814 6.356246 6.415861 6.417500 6.519012 6.542958 6.543525 
       9       10       11       12 
6.601596 6.634371 6.728517 6.728517 
```

`augment()`

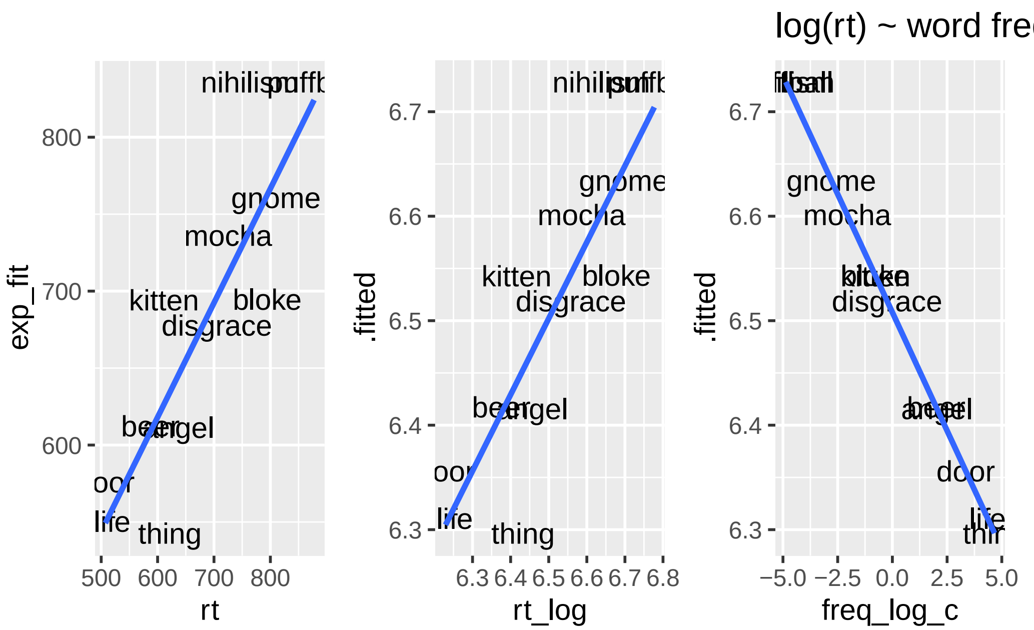
- remember the `augment()` function appends model output to the data frame

```r
df_freq <-
augment(fit_log, data = df_freq) |>
  arrange(freq) |>
  mutate(exp_fit = exp(.fitted)) |>
  relocate(exp_fit, .after = rt)
df_freq |>
  select(word, rt_log, .fitted, rt, exp_fit) |>
  head()
```

```
# A tibble: 6 x 5
  word     rt_log .fitted    rt exp_fit
  <chr>     <dbl>   <dbl> <dbl>   <dbl>
1 nihilism   6.64    6.73  764.    836.
2 puffball   6.78    6.73  878.    836.
3 gnome      6.70    6.63  810.    761.
4 mocha      6.59    6.60  725.    736.
5 bloke      6.68    6.54  794.    695.
6 kitten     6.42    6.54  611.    694.
```

```r
fig_fit_raw <-
  df_freq |>
  ggplot() +
  aes(x = rt, y = exp_fit, label = word) +
  geom_text() +
  geom_smooth(method = "lm", se = F)


fig_fit_log <-
df_freq |>
  ggplot() +
  aes(x = rt_log, y = .fitted, label = word) +
  geom_text() +
  geom_smooth(method = "lm", se = F)

fig_fit_freq <-
df_freq |>
  ggplot() +
  aes(x = freq_log_c, y = .fitted, label = word) +
  labs(title = "log(rt) ~ word frequency") +
```

```
  geom_text() +
  geom_smooth(method = "lm", se = F)

fig_fit_raw + fig_fit_log + fig_fit_freq
```



Log for positive values

- notice that we logged `freq` *before* centering it
  - this is because centering makes half the values negative (because half are below the mean)
  - it's not possible to log-transform zero or negative numbers (because Eulen's *e* cannot be risen to the power of 0 or a negative number!)
  - so, always log before centering predictors!

Reporting transformations

- data transformations are typically reported in the data analysis section

  Reaction times and word frequencies had a non-normal distribution with a positive skew. Both variables were log-transformed to achieve normality. Log word frequencies were then standardized by subtracting the variable's mean from each value, and dividing by the standard deviation of the variable's standard deviation. A linear regression model was fit to log-transformed reaction times with standardized log-transformed frequency values as fixed effect.

- it's always good practice to look at papers in a relevant field to get an idea of what to report, especially those that you think were well written and whose methodology you trust

Learning Objectives

Today we learned...

- why and how to centre continuous predictors
- when and how to standardize continuous predictors
- why and how to log-transform continuous variables

Take-home messages

- linear transformations are cosmetic changes
  - do not alter the relationship between variables or values
  - changes the representation of values
  - centring should be performed on continuous predictors (and interval response variables, like ratings scales; but subtract median possible response from observed responses)
  - standardizing should be performed when there are multiple continuous predictors
- non-linear transformations attempt to normalise skewed variables
  - compress the data, squeezing larger/more extreme values to the rest of the data
  - reduces the spread in the distribution

Task

Assessing assumptions

1. Re-run the models `fit_rt_freq`, `fit_rt_freq_c`, and `fit_log`
2. Produce diagnostic plots for each of them (histograms, Q-Q plots, residual plots)
3. Interpret the plots

Model comparison

1. Use the `glance()` function to inspect the R^2, AIC, and BIC of each model.
2. Which is the best fit? Why?

Literaturverzeichnis

Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics using R*.

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