Linear Regression 1

Simple Linear Regression

Daniela Palleschi

2023-04-13

Table of contents

```
## play sound if error encountered
### from: https://sejohnston.com/2015/02/24/make-r-beep-when-r-markdown-finishes-or-when-i
options(error = function(){    # Beep on error
  beepr::beep(sound = "wilhelm")
  Sys.sleep(2) #
  }
```

```r
 )
## and when knitting is complete
.Last <- function() {          # Beep on exiting session
  beepr::beep(sound = "ping")
  Sys.sleep(6) # allow to play for 6 seconds
  }
```

```r
# Create references.json file based on the citations in this script
# make sure you have 'bibliography: references.json' in the YAML
rbbt::bbt_update_bib("_lin_reg1.qmd")
```

```
Wrote 5 references to './references/references.json'
```

```r
knitr::opts_chunk$set(eval = T, # change this to 'eval = T' to reproduce the analyses; mak
                      echo = T, # 'print code chunk?'
                      message = F, # 'print messages (e.g., warnings)?'
                      error = F,
                      warning = F)
```

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

# load libraries
library(tidyverse)
library(ggplot2)
```

Resources

- these slides are based on a mix of the following resources

DeBruine and Barr (2021); Winter (n.d.); Winter (2014); Winter (2019)

- and on slides that were originally based on Field, Miles, and Field (2013), but if you're looking for a textbook I'd recommend Winter (2019)

(Linear) Regression

- our data exploration has given us an idea about what our data look like

- we fit a model to our data, and use it to *predict* values of our DV based on one (or more) IV(s)
    - i.e., *predicting* an outcome variable (DIV) from one or more predictors (IVs)
- because we're making predictions, we need to take into account the variability (i.e., *error*) in our data

$$outcome_i = (model) + error_i$$

Types of regression

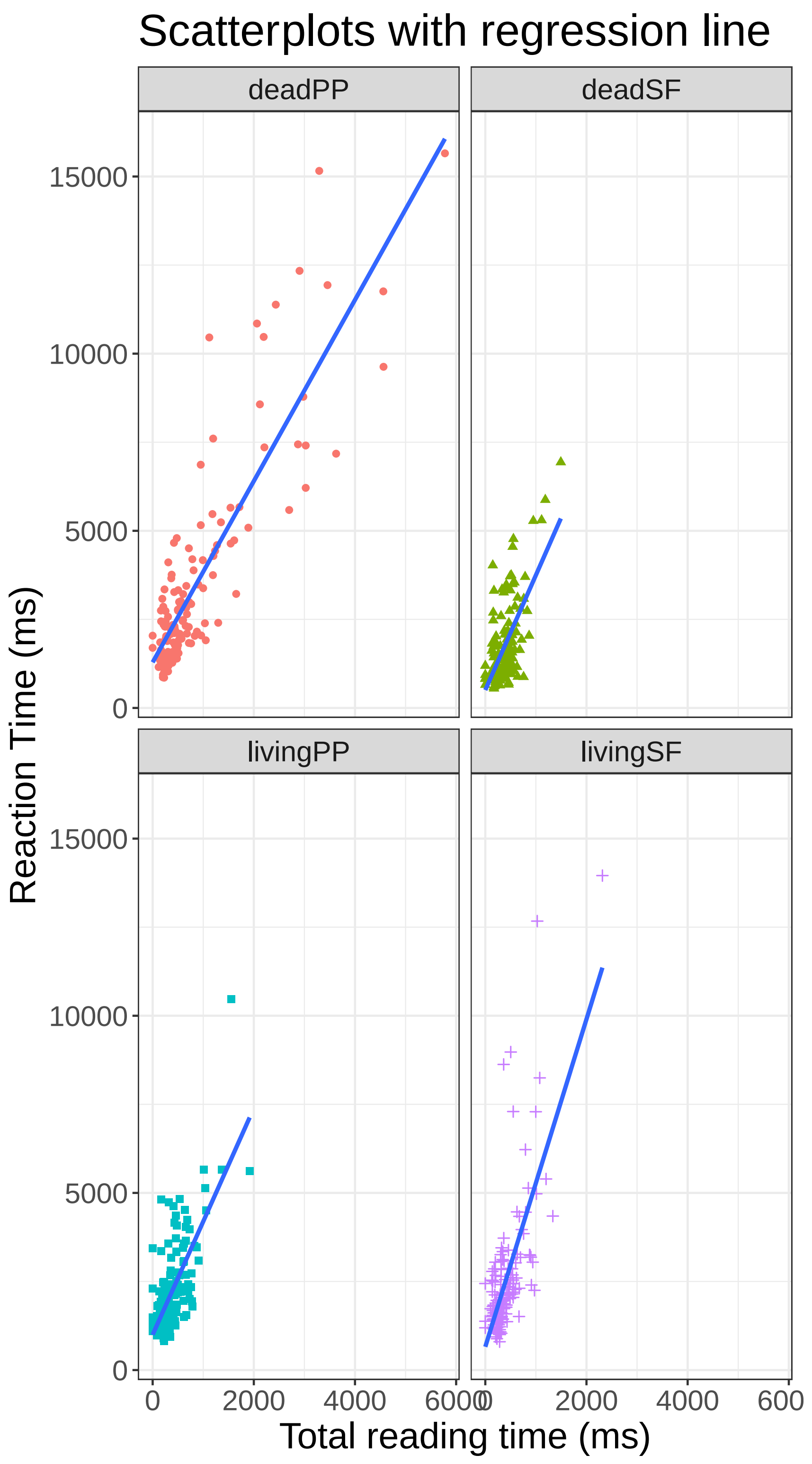
- Simple regression
    - single continuous predictor
- Multiple regression
    - multiple predictors
- Logistic regression
    - binary predictor
- Hierarchical/mixed models
    - include random effects

Straight lines

- regression: summarise the data with a straight line
- straight lines can be defined by
    - Slope (b_1)
        * regression coefficient for the predictor
        * gradient (slope) f the regression line
        * direction/strenth of relationship
    - Intercept (b_0)
        * value of Y when $X = 0$

```
df_crit_verb <- readr::read_csv(here::here("data/tidy_data_lifetime_pilot.csv"),
                                # for special characters
                                locale = readr::locale(encoding = "latin1")
                                ) |>
```

```
  mutate_if(is.character,as.factor) |> # all character variables as factor
  filter(type == "critical", # only critical trials
         px != "px7",
         region == "verb")

df_crit_verb |>
ggplot(aes(x = tt, y = rt)) +
  facet_wrap(.~condition) +
  labs(title = "Scatterplots with regression line",
       x = "Total reading time (ms)",
       y = "Reaction Time (ms)") +
  geom_point(aes(colour = condition, shape = condition)) +
  geom_smooth(method="lm", se=F, fullrange=FALSE, level=0.95) +
  theme_bw() +
  theme(legend.position = "none",
        text = element_text(size=18))
```



Scatterplots with regression line
deadPP
deadSF
livingPP
livingSF
Reaction Time (ms)
0
5000
10000
15000
0
2000
4000
6000
600
Total reading time (ms)

Slopes

- slopes describe a change in x (Δx) over a change in y (Δy)
  - positive slope: as x increases, y increases
  - negative slope: as x increases, y decreases
  - if the slope is 0, there is no change in y as a function of x

$$slope = \frac{\Delta x}{\Delta y}$$

- or: the change in y when x increase by 1 unit
  - sometimes referred to as "rise over run": how do you 'rise' in y for a given 'run' in x?

Slopes

Intercepts

Varying slopes and intercepts

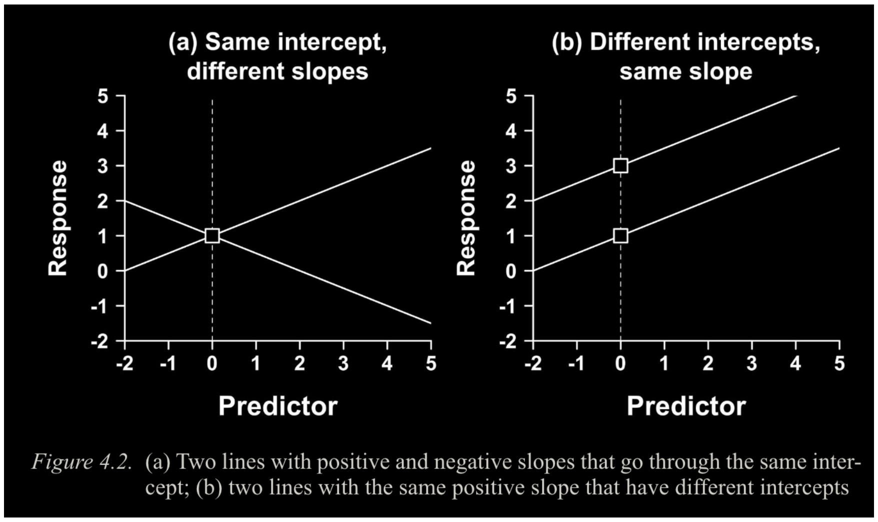


Figure 1: Image source: Winter (2019) (all rights reserved)

A simple linear model

- first, order our predictor
  - we predict longer reading times for dead versus living, so order living dead

```
# order factor levels
df_crit_verb$lifetime <- factor(df_crit_verb$lifetime, levels = c("living","dead"))

# set contrasts
contrasts(df_crit_verb$lifetime) <- c(-0.5,+0.5); contrasts(df_crit_verb$lifetime)
```

```
       [,1]
living -0.5
dead    0.5
```

fit model

- let's exclude missing observations (0)

```
# fit simple linear model
fit_1 <- df_crit_verb %>%
  filter(ff > 0) %>%
  lm(ff ~ lifetime, data = .)
```

```
# alternatively
fit_1 <- lm(ff ~ lifetime,
            data = df_crit_verb, subset = ff > 0)
```

Coefficients table with summary()

```
> summary(fit_1)

Call:
lm(formula = ff ~ lifetime, data = df_crit_verb, subset = ff > 0) #<1>

Residuals:                                                        #<2>
    Min      1Q  Median      3Q     Max
-118.78  -37.78  -11.39   25.22  264.61
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)                #<3>
(Intercept)   199.089      2.466  80.743   <2e-16 ***            #<4>
lifetimedead    5.388      4.931   1.093    0.275                #<5>
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 57.46 on 541 degrees of freedom
Multiple R-squared:  0.002202,  Adjusted R-squared:  0.0003575   #<6>
F-statistic: 1.194 on 1 and 541 DF,  p-value: 0.275
```

① formula
② Residuals: differences between observed values and those predicted by the model
③ Names for columns Estimates, SE, t-value, p-value
④ Intercept (b_0), i.e., value of y (first fix.) with a move of one unit of x (lifetime)
⑤ Slope (b_1), i.e., change in first fixation going from `dead` to `living`
⑥ (Adjusted) R^2, a measure of model fit (squared residuals)

- what is the **intercept**?
- is the **slope** positive or negative?
  - what is it's value?
- this is what the slope would look like:

Understanding the summary

- let's compute summary statistics based on *lifetime*
  - then compare this to the model output

Exercises

1. Subtract the mean first-fixation reading time of `dead` from that of `living`
   - what does this correspond to in the model summary?
2. Compute the mean of `dead`+`living`
   - what does this correspond to in the model summary?
3. Divide the slope in 2. Subtract this from the mean of `dead`.
   - what does this correspond to?

Summary statistics

Table 1: Summmary statistics for first-fixation duration at the verb region

lifetime	N	mean	sd	se	ci	lower.ci	up
living	271	196.395	59.639	3.623	7.133	189.262	20
dead	272	201.783	55.197	3.347	6.589	195.194	20

```
# compute summary
summary_ff_life <- df_crit_verb |>
  filter(region=="verb",
         ff > 0) |>
  group_by(lifetime) %>%
  summarise(N = n(),
            mean = mean(ff, na.rm = T),
            sd = sd(ff, na.rm = T)) %>%
  # compute standard error, confidence intervals, and lower/upper ci bounds
  mutate(se = sd / sqrt(N),
         ci = qt(1 - (0.05 / 2), N - 1) * se,
         lower.ci = mean - qt(1 - (0.05 / 2), N - 1) * se,
         upper.ci = mean + qt(1 - (0.05 / 2), N - 1) * se)

knitr::kable(summary_ff_life, digits=3,
             caption = "Summmary statistics for first-fixation duration at the verb region
  kableExtra::kable_styling(font_size = 24,
                            position = "left")
```

Model summary

```
summary(fit_1)
```

```

Call:
lm(formula = ff ~ lifetime, data = df_crit_verb, subset = ff >
    0)

Residuals:
    Min      1Q  Median      3Q     Max
-118.78  -37.78  -11.39   25.22  264.61

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)
```

```
(Intercept)  199.089      2.466  80.743 <0.0000000000000002 ***
lifetime1      5.388      4.931   1.093              0.275
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 57.46 on 541 degrees of freedom
Multiple R-squared:  0.002202,  Adjusted R-squared:  0.0003575
F-statistic: 1.194 on 1 and 541 DF,  p-value: 0.275
```

broom package for tidy model summaries

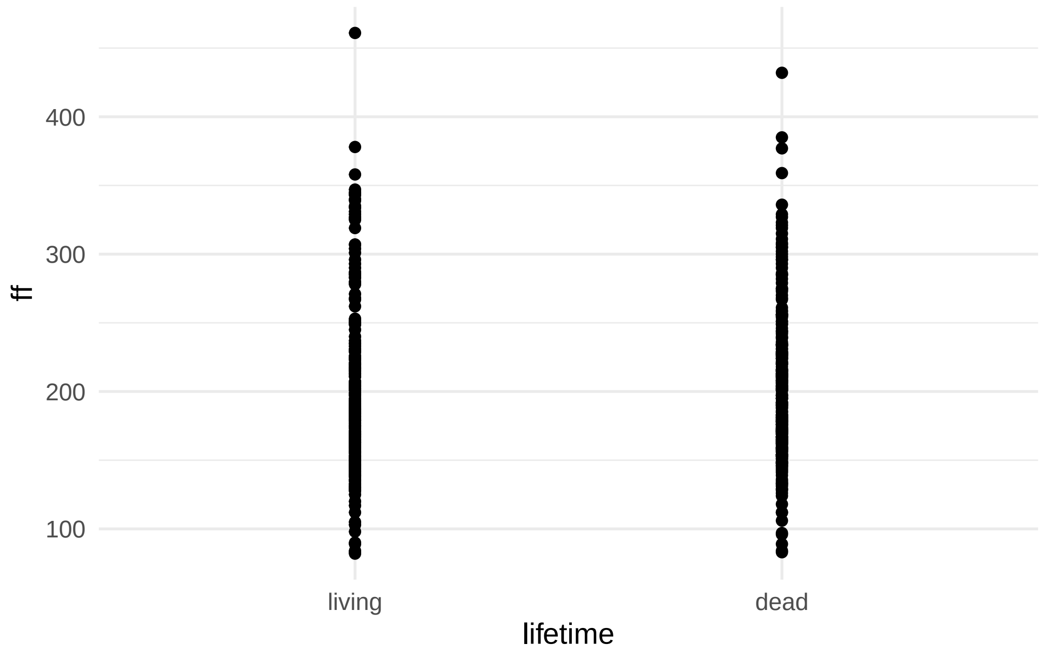
```
# install.packages("broom")
library(broom)
tidy(fit_1)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)   199.        2.47     80.7  5.95e-304
2 lifetime1       5.39      4.93      1.09 2.75e-  1
```

```
glance(fit_1)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1   0.00220      0.000358  57.5      1.19   0.275     1 -2969. 5944. 5957.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
df_crit_verb %>%
  filter(ff > 0) |>
  ggplot(aes(x = lifetime, y = ff)) +
  geom_point(position = position_dodge(.6)) +
  geom_smooth(method = 'lm') + theme_minimal()
```



```
df_crit_verb %>%
  # mutate(life_c = if_else(lifetime=="living",-0.5,+0.5)) %>%
  select(ff,fp,rpd,tt) %>%
  cor()
```

```
            ff        fp       rpd        tt
ff  1.0000000 0.4530152 0.4146744 0.1273391
fp  0.4530152 1.0000000 0.8956934 0.3620559
rpd 0.4146744 0.8956934 1.0000000 0.3325920
tt  0.1273391 0.3620559 0.3325920 1.0000000
```

```
# fit simple linear model with log
fit_2 <- df_crit_verb %>%
  filter(ff > 0) %>% # because you can't log transform 0
  mutate(ff_c = ff-mean(ff)) %>%
  lm(ff_c ~ lifetime, data = .)
summary(fit_2)
```

```
Call:
lm(formula = ff_c ~ lifetime, data = .)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-118.78  -37.78  -11.39   25.22  264.61 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.004962   2.465724  -0.002    0.998
lifetime1    5.388254   4.931448   1.093    0.275

Residual standard error: 57.46 on 541 degrees of freedom
Multiple R-squared:  0.002202,	Adjusted R-squared:  0.0003575 
F-statistic: 1.194 on 1 and 541 DF,  p-value: 0.275
```

Exploring the model

- the linear model contains *fitted* values corresponding to our *observed* values
  - these *fitted* values are fit to a straight line
  - our *observed* values are not fit to a straight line
  - the *residuals* are the differences along the y axis from the fitted to the observed values

```
# how many observed values did we enter into the model?
df_crit_verb |>
  filter(ff > 0) |>
  nrow()
```

```
[1] 543
```

```
# how many observed values did we enter into the model?
length(fitted(fit_1))
```

```
[1] 543
```

```
# how many observed values did we enter into the model?
head(fitted(fit_1))
```

```
       1        2        3        4        5        6 
196.3948 196.3948 201.7831 196.3948 201.7831 196.3948 
```

```r
# look at the residuals
head(residuals(fit_1))
```

```
        1         2         3         4         5         6 
-21.39483  10.60517  26.21691  34.60517  10.21691 -28.39483 
```

```r
coef(fit_1)
```

```
(Intercept)   lifetime1 
 199.088961    5.388254 
```

```r
coef(fit_1)['(Intercept)'] + coef(fit_1)['lifetime1'] * -0.5
```

```
(Intercept) 
   196.3948 
```

```r
coef(fit_1)['(Intercept)'] + coef(fit_1)['lifetime1'] * 0.5
```

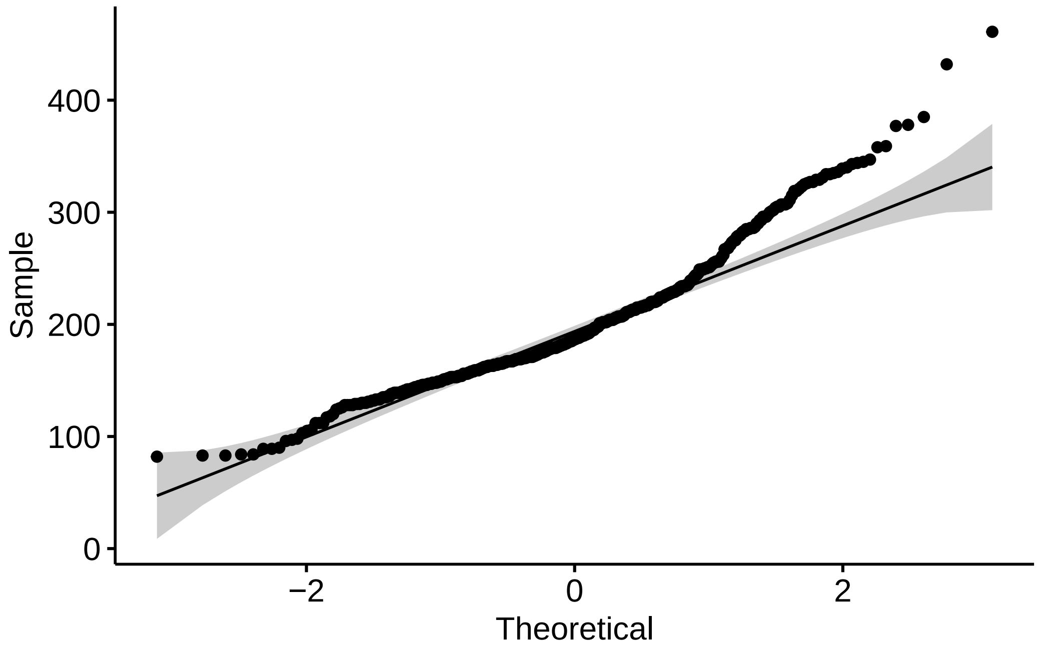
```
(Intercept) 
   201.7831 
```

Assumptions

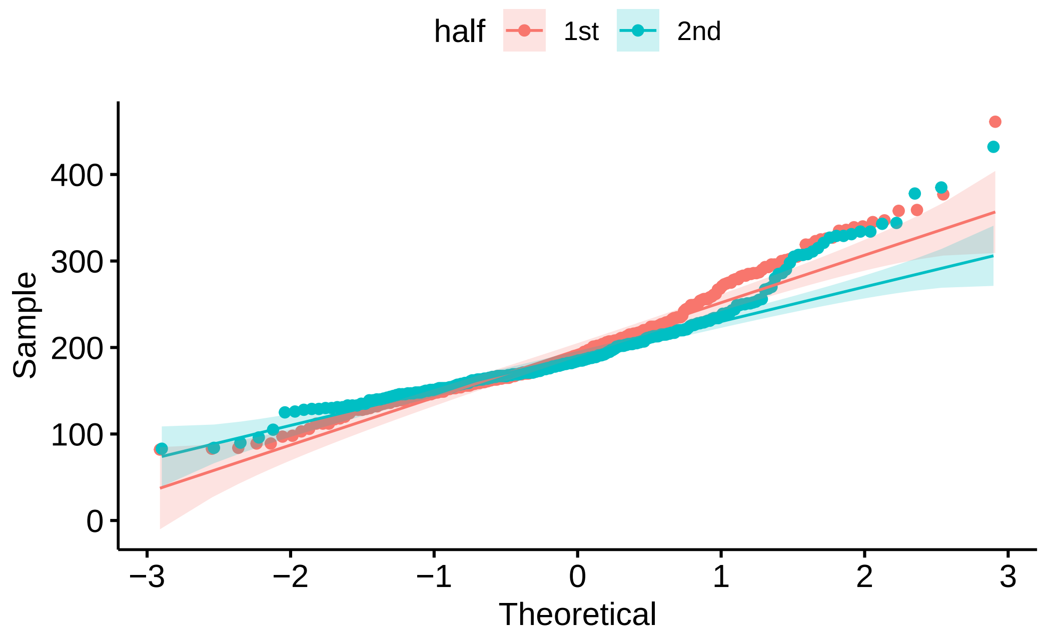
- normality assumption
  - residuals of the model are (approximately) normally distributed
- constant variance assumption (homoscedasticity)
  - spread of residuals should be (approximately) equal along the regression line

Normality assumption

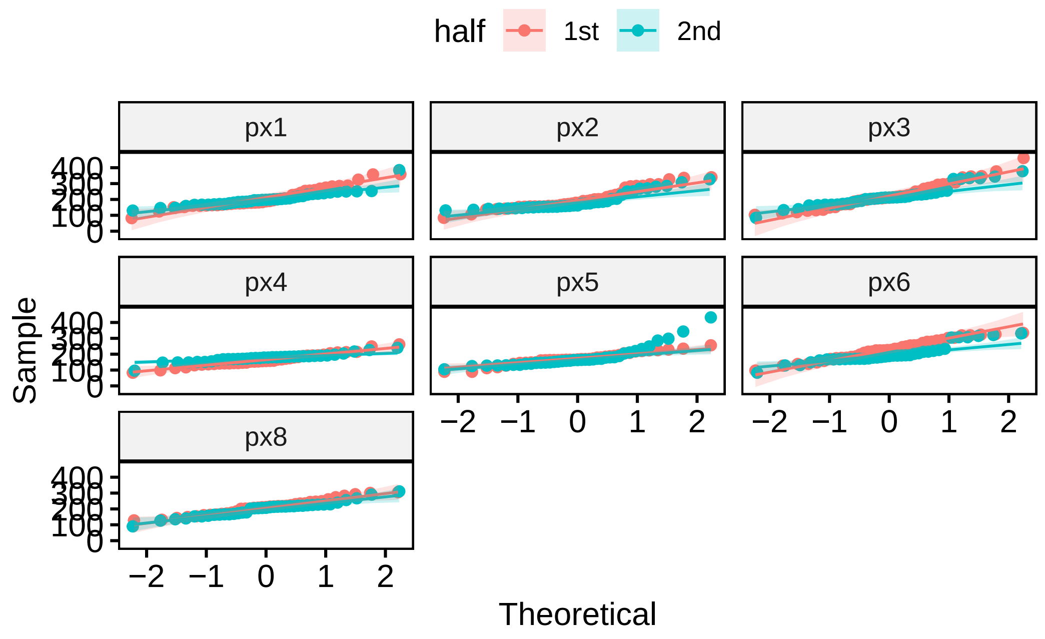
```
df_crit_verb |>
  filter(ff > 0) |>
  mutate(half = if_else(trial >= 104, "1st","2nd")) |>
  ggpubr::ggqqplot(x = "ff")
```



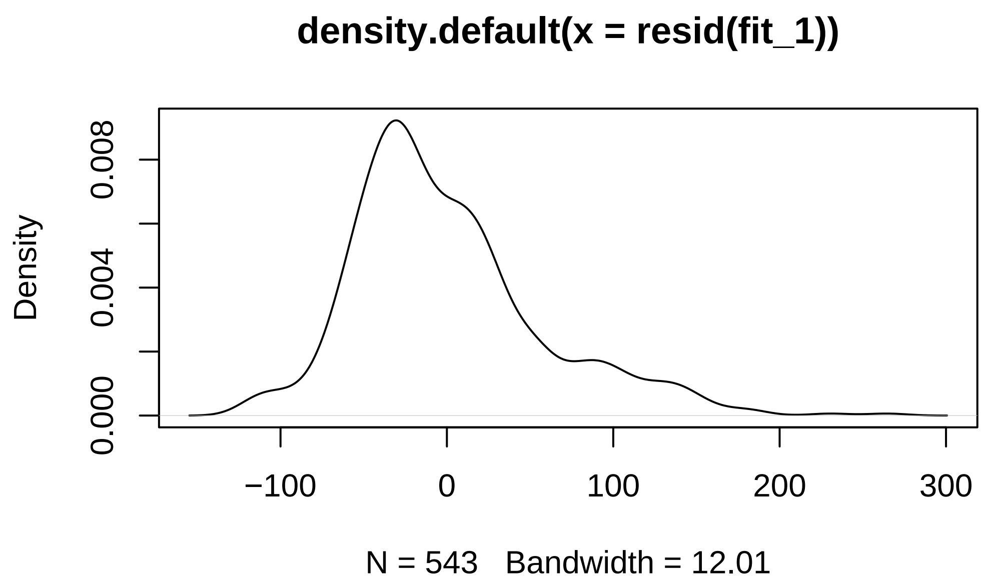
```
df_crit_verb |>
  filter(ff > 0) |>
  mutate(half = if_else(trial >= 104, "1st","2nd")) |>
  ggpubr::ggqqplot( x = "ff",
                    color = "half")
```



```
df_crit_verb |>
  filter(ff > 0) |>
  mutate(half = if_else(trial >= 104, "1st","2nd")) |>
  ggpubr::ggqqplot( x = "ff",
                    color = "half",
                    facet.by = "px")
```



```
plot(density(resid(fit_1)))
```



Log transformation

- for more see Section 5.4 in Winter (2019)
- the R funtion `log()` computes the 'natural logarithm' (and is the inverse of the exponential `exp()`)
  - `log()` makes large numbers smaller
  - `exp()` makes small numbers larger

```
log(0)
```

```
[1] -Inf
```

```
log(1:10)
```

```
 [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101
 [8] 2.0794415 2.1972246 2.3025851
```

```
log(c(10,20,30,40,100))
```

```
[1] 2.302585 2.995732 3.401197 3.688879 4.605170
```

```
exp(1:10)
```

```
 [1]     2.718282     7.389056    20.085537    54.598150   148.413159
 [6]   403.428793  1096.633158  2980.957987  8103.083928 22026.465795
```

```
exp(log(1:10))
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

- continuous variables truncated at 0 typically have a *positive skew*
  - a lot of small values (e.g., `tt` $<$ 500ms), with some larger values ($>$ `tt` 1000)
  - this usually means our residuals are also positively skewed, i.e., not normally distributed
- so we typically log-transform raw reading/reaction times for our linear models

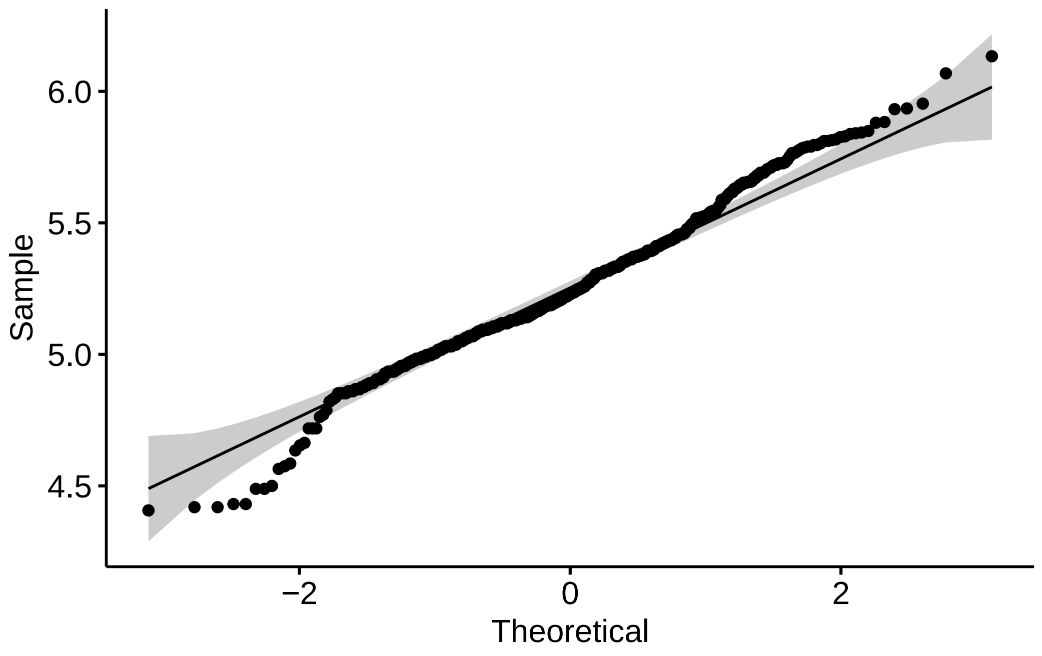
```r
# fit simple linear model with log
fit_2 <- df_crit_verb %>%
  filter(ff > 0) %>% # important! you can't log transform 0
  lm(log(ff) ~ lifetime, data = .)
summary(fit_2)
```

```

Call:
lm(formula = log(ff) ~ lifetime, data = .)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.85242 -0.17140 -0.01899  0.15691  0.89550 

Coefficients:
            Estimate Std. Error t value            Pr(>|t|)    
(Intercept)  5.25458    0.01197 439.064 <0.0000000000000002 ***
lifetime1    0.03336    0.02394   1.394               0.164    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2789 on 541 degrees of freedom
Multiple R-squared:  0.003579,	Adjusted R-squared:  0.001737 
F-statistic: 1.943 on 1 and 541 DF,  p-value: 0.1639
```

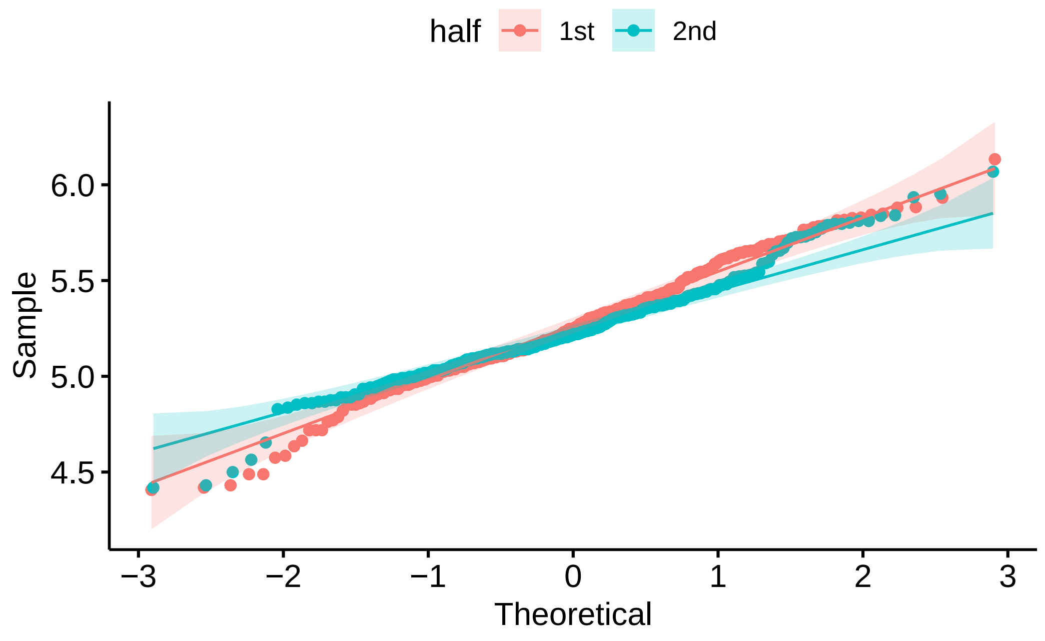
Communicating your results

- model summaries can be provided via tables and/or figures
  - you should always report the t-values and p-values of an effect

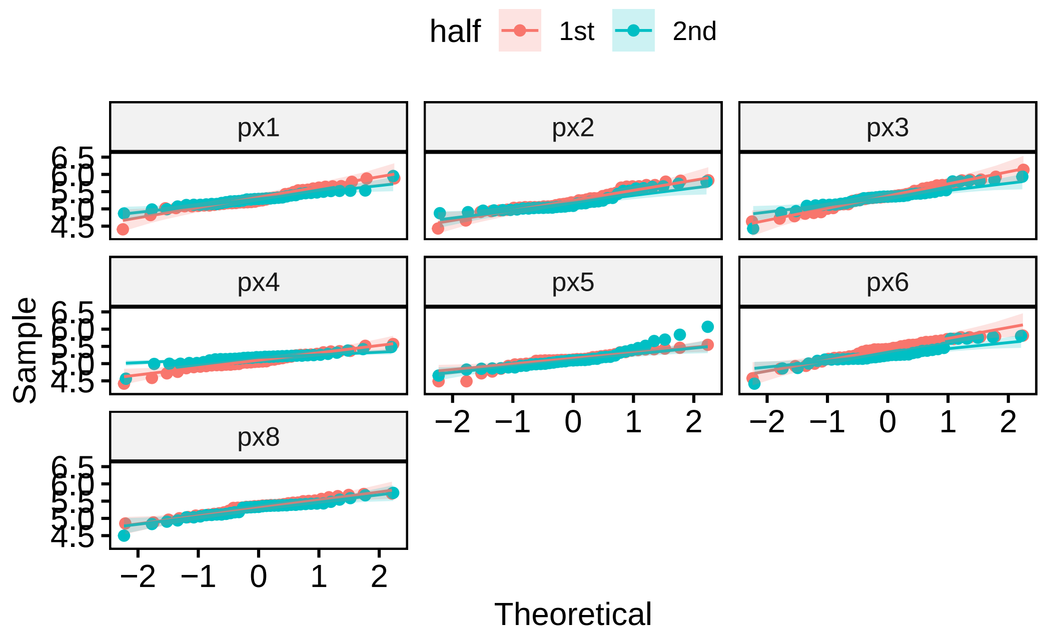
```r
df_crit_verb |>
  filter(ff > 0) |>
  mutate(log_ff = log(ff)) |>
  mutate(half = if_else(trial >= 104, "1st","2nd")) |>
  ggpubr::ggqqplot(x = "log_ff")
```



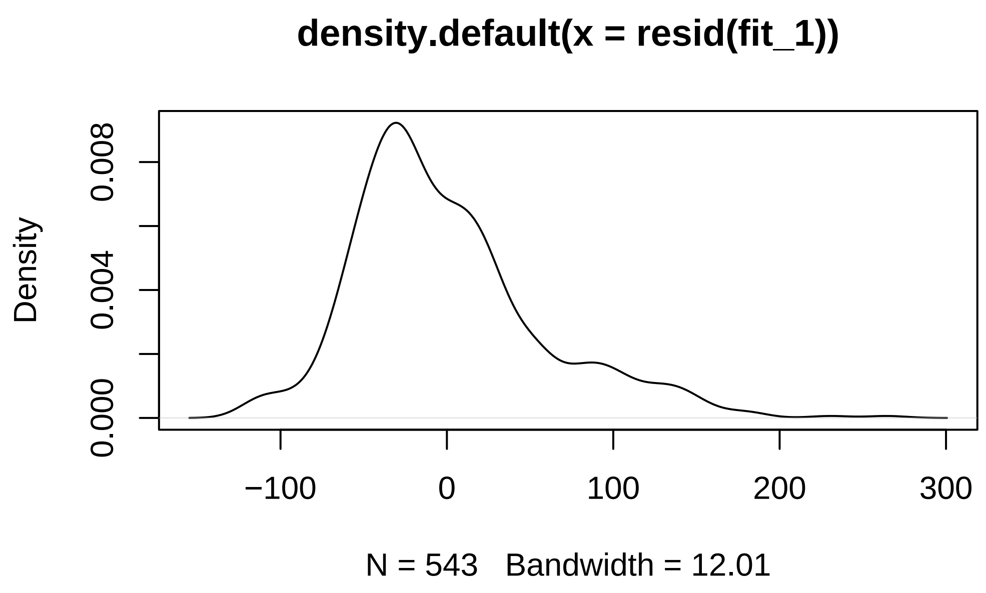
```
df_crit_verb |>
  filter(ff > 0) |>
  mutate(log_ff = log(ff)) |>
  mutate(half = if_else(trial >= 104, "1st","2nd")) |>
  ggpubr::ggqqplot( x = "log_ff",
                    color = "half")
```



```
df_crit_verb |>
  filter(ff > 0) |>
  mutate(log_ff = log(ff)) |>
  mutate(half = if_else(trial >= 104, "1st","2nd")) |>
  ggpubr::ggqqplot( x = "log_ff",
                    color = "half",
                    facet.by = "px")
```



```
plot(density(resid(fit_1)))
```



```
b0 <- coef(fit_2)["(Intercept)"]
b1 <- (coef(fit_2)["lifetime1"])

exp(b0+b1)
```

```
(Intercept)
   197.9364
```

DeBruine, Lisa M., and Dale J. Barr. 2021. "Understanding Mixed-Effects Models Through Data Simulation." *Advances in Methods and Practices in Psychological Science* 4 (1): 251524592096511. https://doi.org/10.1177/2515245920965119.

Field, Andy, Jeremy Miles, and Zoe Field. 2013. *Discovering Statistics Using R*. Vol. 50. https://doi.org/10.5860/choice.50-2114.

Winter, Bodo. 2014. "A Very Basic Tutorial for Performing Linear Mixed Effects Analyses (Tutorial 2)." 2014.

———. 2019. *Statistics for Linguists: An Introduction Using R*. Routledge. https://doi.org/10.4324/9781315165547.

———. n.d. "Linear Models and Linear Mixed Effects Models in R: Tutorial 1."