PSY 503: Foundations of Statistical Methods in Psychological Science

More LM: Transformations

Jason Geller, Ph.D. (he/him/his)

Princeton University

Updated:2022-11-06

Outline

- Check-In Q&A
- Transformations
  - Linear Transformations
    - Centering
    - Standardization
  - Nonlinear Transformations
    - Logarithms
- Polynomial regression

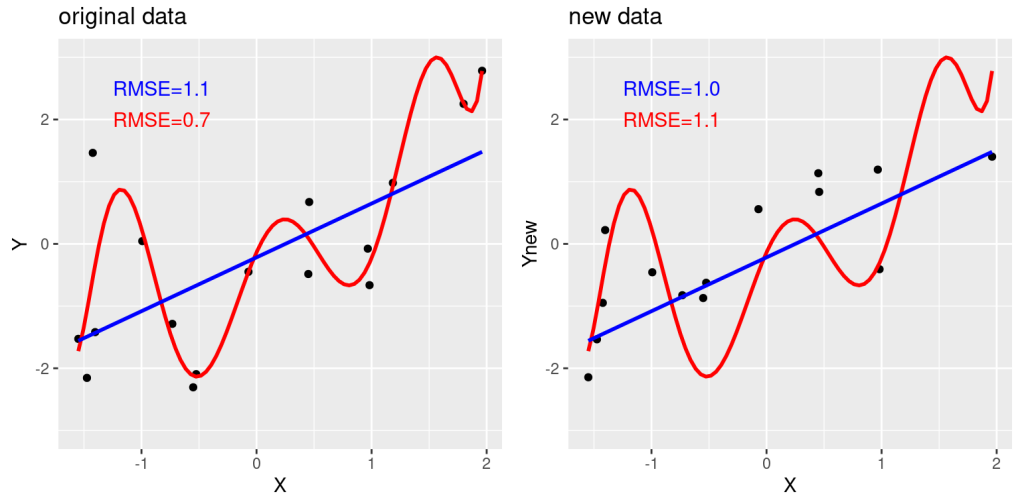
Check-In Q&A

1. Can a model be too good?
2. What is a grand mean?
3. Intercept of the linear model
4. In effects coding/sum coding, why is centering necessary?
5. η^2 vs η_p^2
6. Would you ever use dummy coding with multiple levels?

Check-in Questions

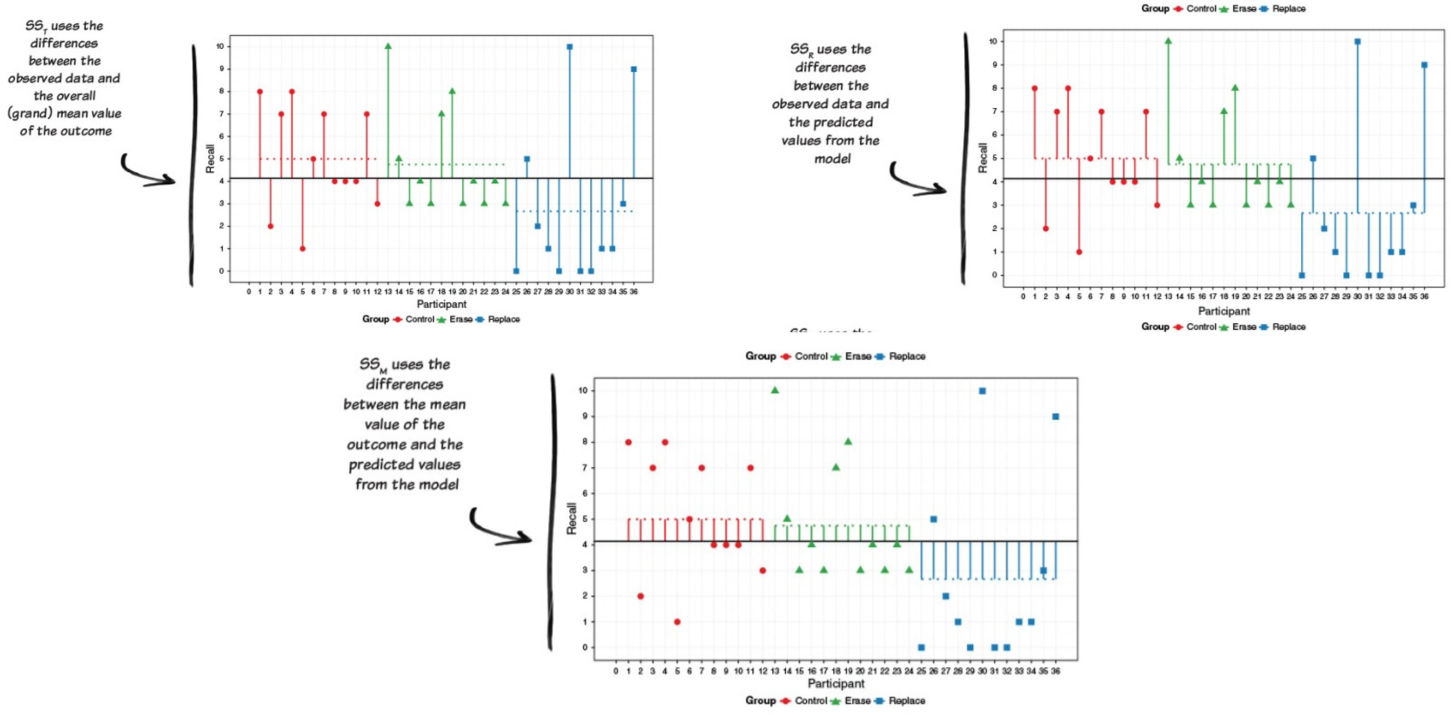
- How many predictors is too many?

- General rule of thumb is 10-20 participants for every predictor variable in your model
- Use other methods to overfiting like lasso or ridge regression
- Only include predictors that are theoretically meaningful



- Sum of Squares

Sum of squares	Degrees of Freedom
$SS_T = \sum(y_i - \bar{y})^2$	N - 1
$SS_R = \sum(\hat{y}_i - \bar{y})^2$	K
$SS_E = \sum(y_i - \hat{y}_i)^2$	N-k+1



Restricted and Full Model

$$F = \frac{SS_R - SS_F/df_R - df_F(p-1)}{SS_F/df_F(N-p)} = \frac{MS_{model}}{MS_{error}}$$

```
restricted <- lm(Val~ 1, data=senses) # intercept-only model

full <- lm(Val~Modality, data=senses)  # full model
```

Weekly Check-In Questions

- Significance of the intercept value
- Generally speaking the *p*-value for the intercept tells us if the intercept is different from 0. Most often this is not of importance.

$$H_0 : \beta_0 = 0$$

Weekly Check-In Questions

- In effects coding/sum coding, why is centering necessary/when would we want to change intercept?
  - Centering is what occurs as a function of taking the mean of dummy codes
  - When dealing with one factor with 2 levels dummy coding vs. sum coding does not matter
  - The real advantage of sum coding is when dealing with main effects and two way interactions
  - Sum coding is what ANOVA is doing to test main effects and interactions (more on this later)

Weekly Check-In Questions

- What is grand mean?
- It is the mean of the means

$$\frac{Group1_{mean} + Group2_{mean}}{2}$$

```
senses %>%
  group_by(Modality) %>% # get each group
  dplyr::summarise(meanval=mean(Val))%>% # get mean
 dplyr::summarise(meanofmeans=mean(meanval)) # mean of means
```

meanofmeans
5.56

Weekly Check-In Questions

- Eta-squared vs partial eta-squared

$$\eta^2$$

: % of total variance explained by IV

$$\eta_p^2$$

: % of variance explained by IV partailing out other variables in the model

Weekly Check-In Questions

- Would you ever use dummy coding with multiple levels?

```
senses <- read_csv("https://raw.githubusercontent.com/jgeller112/psy503-psych_stats/master/static/slides/11-Ca
lm(Val~Modality, data=senses) %>%
  model_parameters()
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	
(Intercept)	5.58	0.0189	0.95	5.54	5.62	295	400	0
ModalitySmell	-0.109	0.0564	0.95	-0.22	0.00229	-1.93	400	0.05
ModalitySound	-0.174	0.0376	0.95	-0.248	-0.101	-4.64	400	4.
ModalityTaste	0.228	0.0431	0.95	0.144	0.313	5.3	400	1.

Dummy Codes (Seidman, Wade, & Geller, 2022)

- Waitlist (do not complete a group, just measures)
- Group-only (attend a group + measures)
- Self-affirmation + group (complete an individual intervention, then attend group, + measures)

- GOvsSA = Waitlist (0), Group only (0), Self-affirmation (1)

- WLvsGO = Waitlist (0), Group only (1) Self-affirmation (1)

GroupvsNo = Waitlist (0), Group (1), Self-Affirmation (1)

emmeans

- Get means and pairwise comparisons

```
# get pairwise tests between all groups

sen <- lm(Val~Modality, data=senses)

as.data.frame(emmeans::emmeans(sen, specs = "Modality")) %>%
flextable()
```

Modality	emmean	SE	df	lower.CL	upper.CL
Sight	5.579663	0.01889440	400	5.542518	5.616808
Smell	5.471012	0.05317357	400	5.366477	5.575546
Sound	5.405193	0.03248092	400	5.341338	5.469047
Taste	5.808124	0.03878081	400	5.731884	5.884364
Touch	5.534435	0.03224121	400	5.471052	5.597818

Pairwise Tests

```
library(flextable)
# get pairwise tests between all groups
means1 = emmeans::emmeans(sen, specs = "Modality")
# use pairs
flextable(as.data.frame(pairs(means1)))
```

contrast	estimate	SE	df	t.ratio	p.value
Sight - Smell	0.10865148	0.05643072	400	1.925396	0.3055011645361
Sight - Sound	0.17447036	0.03757671	400	4.643046	0.0000456942086
Sight - Taste	-0.22846083	0.04313872	400	-5.295957	0.0000019440745
Sight - Touch	0.04522812	0.03736969	400	1.210289	0.7454337311622

Today's Datasets

- Pitch and Age

```
data = read_csv("https://raw.githubusercontent.com/jgeller112/psy503-psych_stats/master/static/slides/10-linear
```

Today's Datasets

- Memory and Time
  - 13 subjects were asked to memorize a list of disconnected items
  - The subjects were then asked to recall the items at various times up to a week later

```
log_df <- tibble::tribble(
   ~time, ~prop,
      1L,  0.84,
      5L,  0.71,
     15L,  0.61,
     30L,  0.56,
     60L,  0.54,
    120L,  0.47,
    240L,  0.45,
    480L,  0.38,
    720L,  0.36,
   1440L,  0.26,
   2880L,   0.2,
   5760L,  0.16,
  10080L,  0.08
  )
```

Transformations

- When should we transform our data?
  - When our data is *uninterpretable or nonsensical*
  - When our data is *non-linear*
  - When our data is *skewed*

Linear Transformations

- Adding, subtracting, dividing by, or multiplying a variable with a constant
- Does not change the relationships in a genuine way (“same model”)

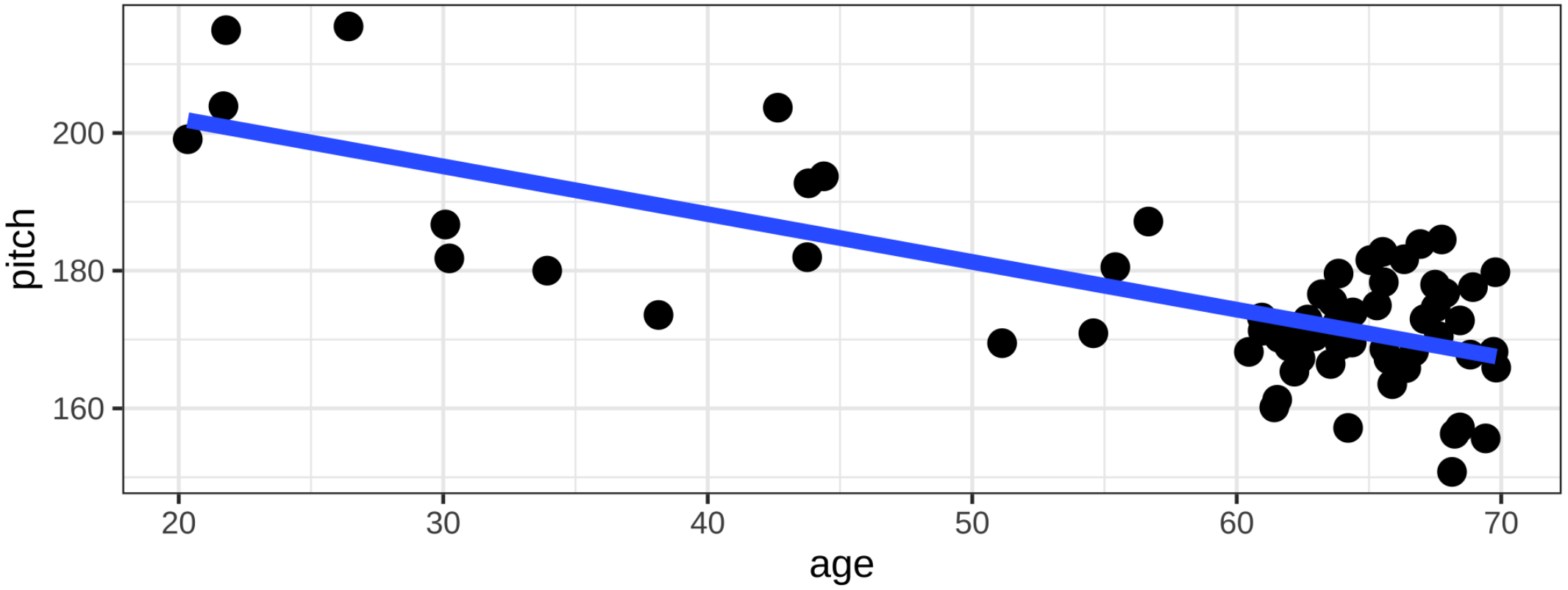
Linear Transformations

- Linear transformations
  - Adding, subtracting, dividing by, or multiplying a variable with a constant
  - Does not change the relationships in a genuine way (“same model”)
  - Common: centering and standardizing

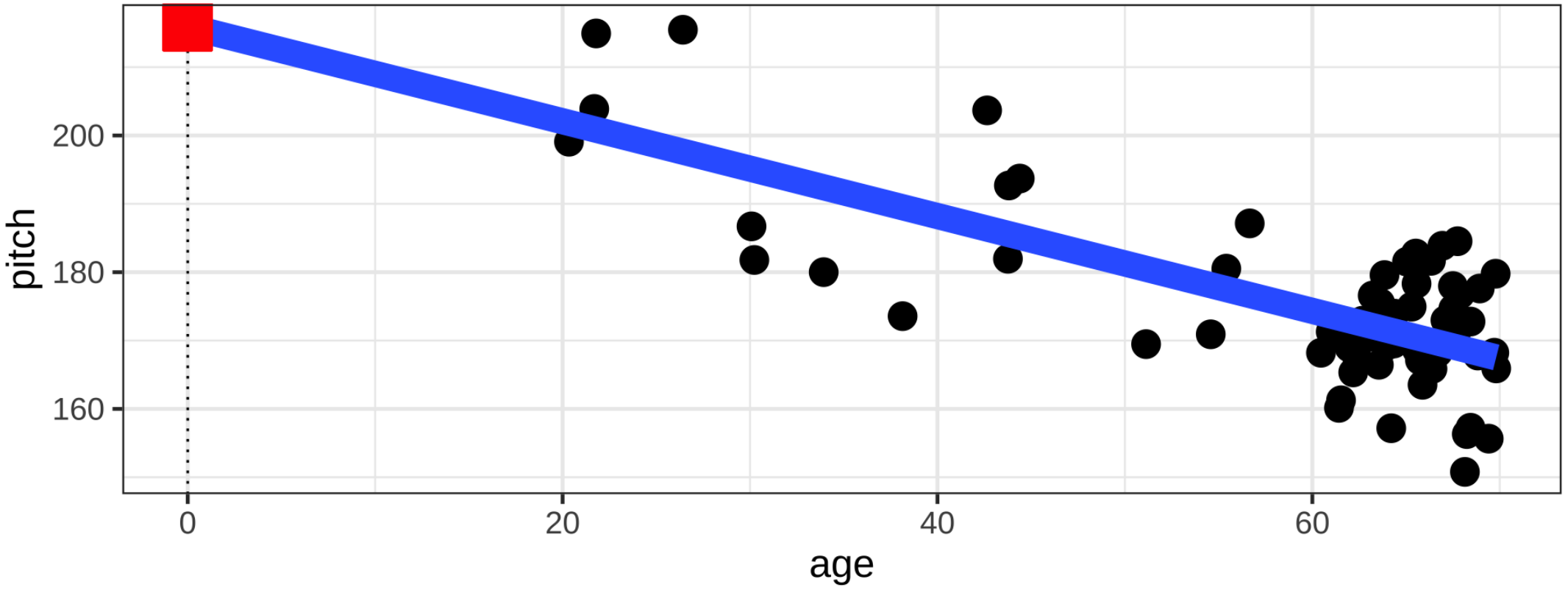
Transformations

- Nonlinear transformations
  - Transformation that affects different data points differently
  - Changes the relationships (“different model”)
  - Common: Logarthims
  - Makes models more in line with assumptions

Linear Transformations



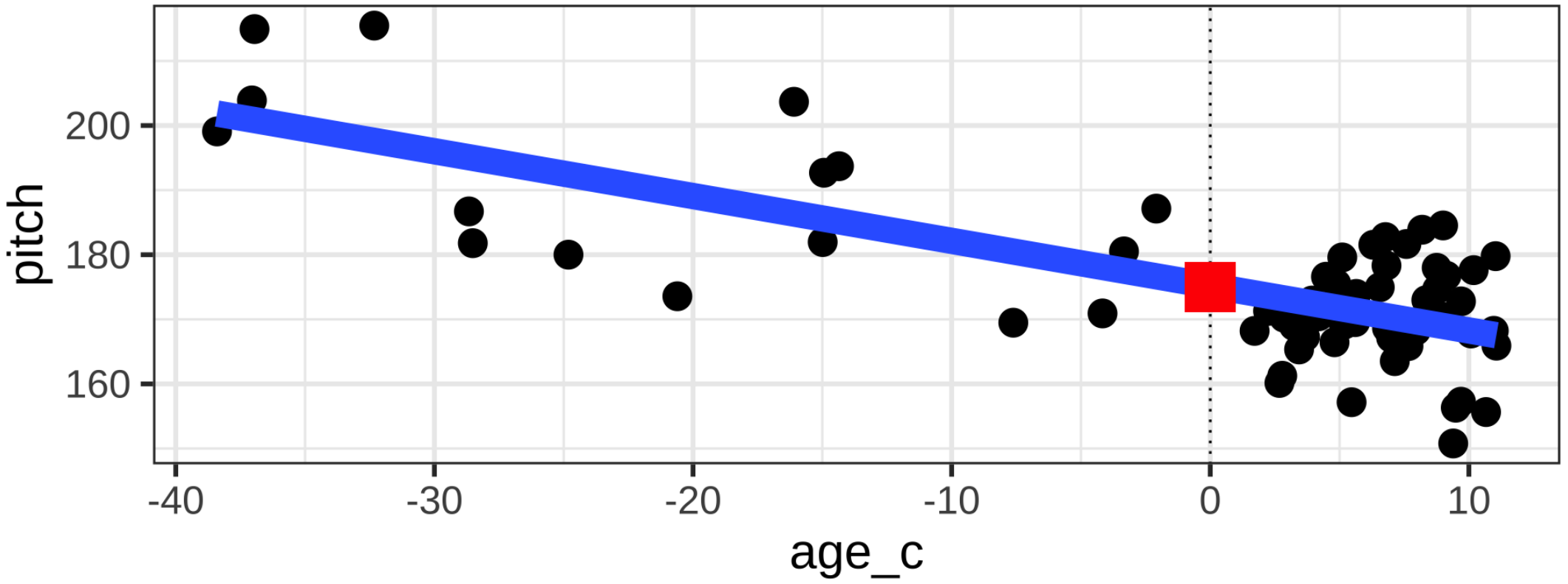
Linear Transformations



Centering

> Sometimes our data is *uninterpretable or nonsensical* (X = 0)

- Centering changes the model so that 0 is mean/average of X



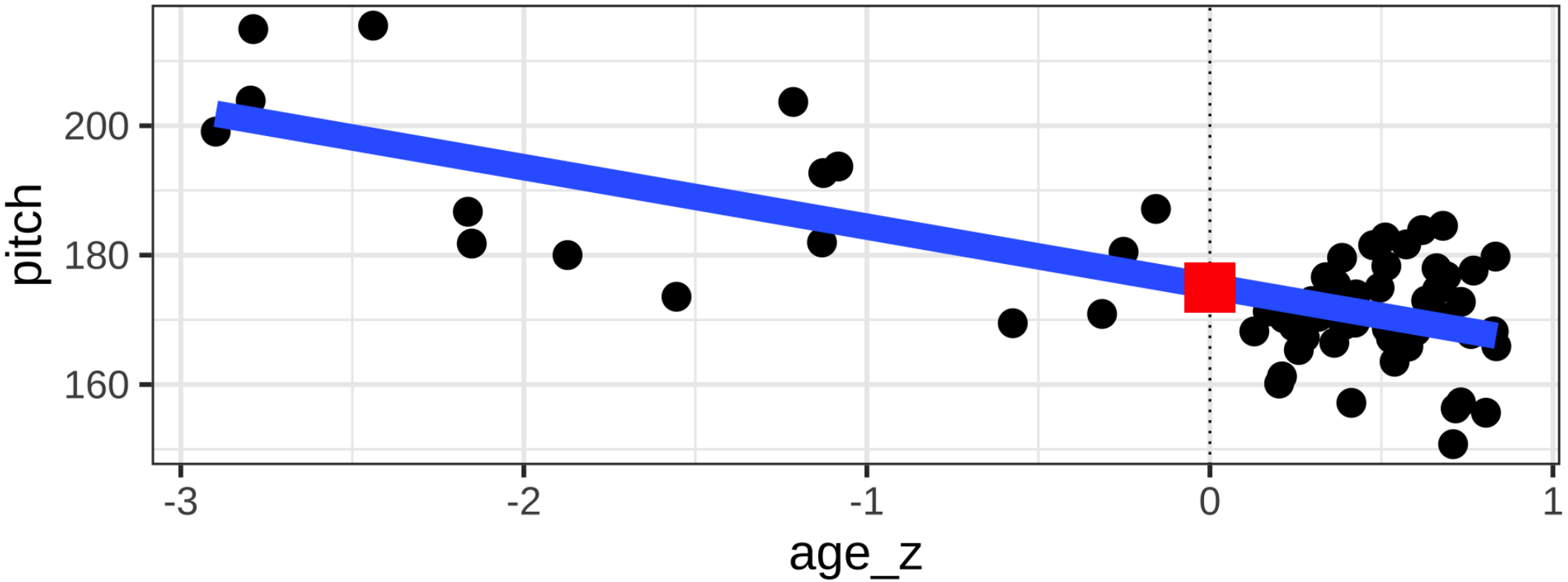
Centering - How?

```
df <- mutate(data,
             age_c = age - mean(age, na.rm = TRUE)) # center
```

```
library(datawizard) # package to center and standardize

df_wiz <- data %>%
  mutate(age_c = datawizard::center(age))
```

Standardize Predictors

- Dividing centered mean by σ
- Puts variables on same metrics



Standardizing - How?

```
df_stand <- data %>%
  mutate(age_z = age_c / sd(age, na.rm = TRUE))# standardize z score
```

age	pitch	age_c	age_z
66.4	166	7.66	0.578
64.3	170	5.6	0.423
61.9	172	3.14	0.237
54.6	171	-4.17	-0.314
69.4	156	10.7	0.805
62	169	3.23	0.244

Output

```
lm(pitch~age, data=data)
```

```
##
## Call:
## lm(formula = pitch ~ age, data = data)
##
## Coefficients:
## (Intercept)          age
##    215.9581      -0.6936
```

```
lm(pitch~age_c, data=df)
```

```
##
## Call:
## lm(formula = pitch ~ age_c, data = df)
##
## Coefficients:
## (Intercept)        age_c
##    175.2105      -0.6936
```

```
lm(pitch~age_z, data=df_wiz)
```

```
##
```

Centering Around Other Values

- We could also make 0 correspond to some other sensible/useful value
    - Smallest logically possible value?

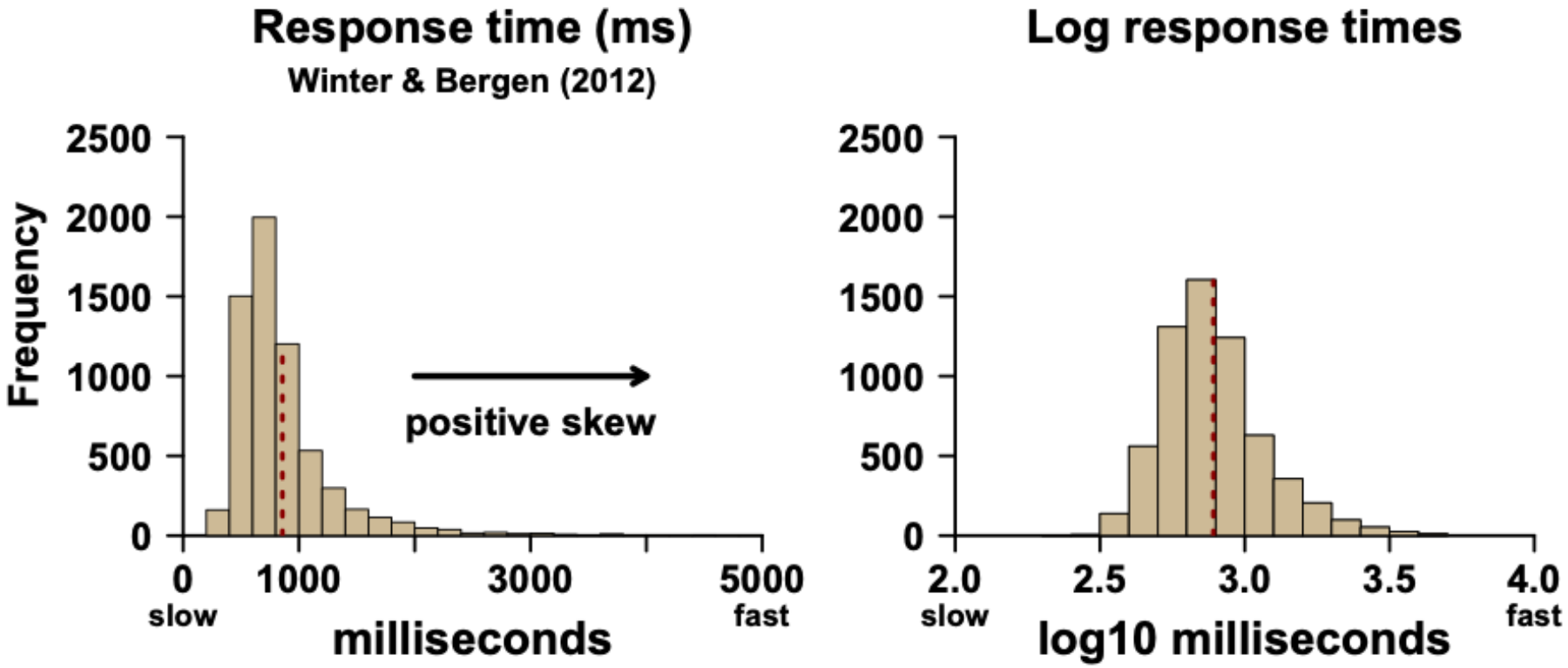
Centering

- Good if zero is *meaningful*
- Do not center if zero is meaningful

When in doubt, center!

Nonlinear Transformations

Logathimthic Transformations



Logathimthic Transformations

```
knitr::include_graphics("base.PNG")
```

Logathimthic Transformations

- Exponentiation
  - Takes small numbers and grows them

$$10^1 = 10$$

$$10^2 = 100$$

$$10^3 = 1000$$

$$10^4 = 10000$$

$$10^5 = 100000$$

Log Transformations

- Logarithmic
  - How many times must one “base” number be multiplied by itself to get some other particular number?
  - Takes large numbers and shrinks them

$$1 = log_{10}(10)$$

$$2 = log_{10}(100)$$

$$3 = log_{10}(1000)$$

$$4 = log_{10}(10000)$$

$$5 = log_{10}(100000)$$

Log Transfomrations

- Tracks the order of magnitude
- Large numbers shrink more than smaller numbers
    - Compression effect (larger values are closer to the other)

```
# time in ms
RTs <- c(600, 650, 700, 1000, 4000)

logRTs <- log(RTs) # base = 2.718282

logRTs
```

```
## [1] 6.396930 6.476972 6.551080 6.907755 8.294050
```

```
exp(logRTs)
```

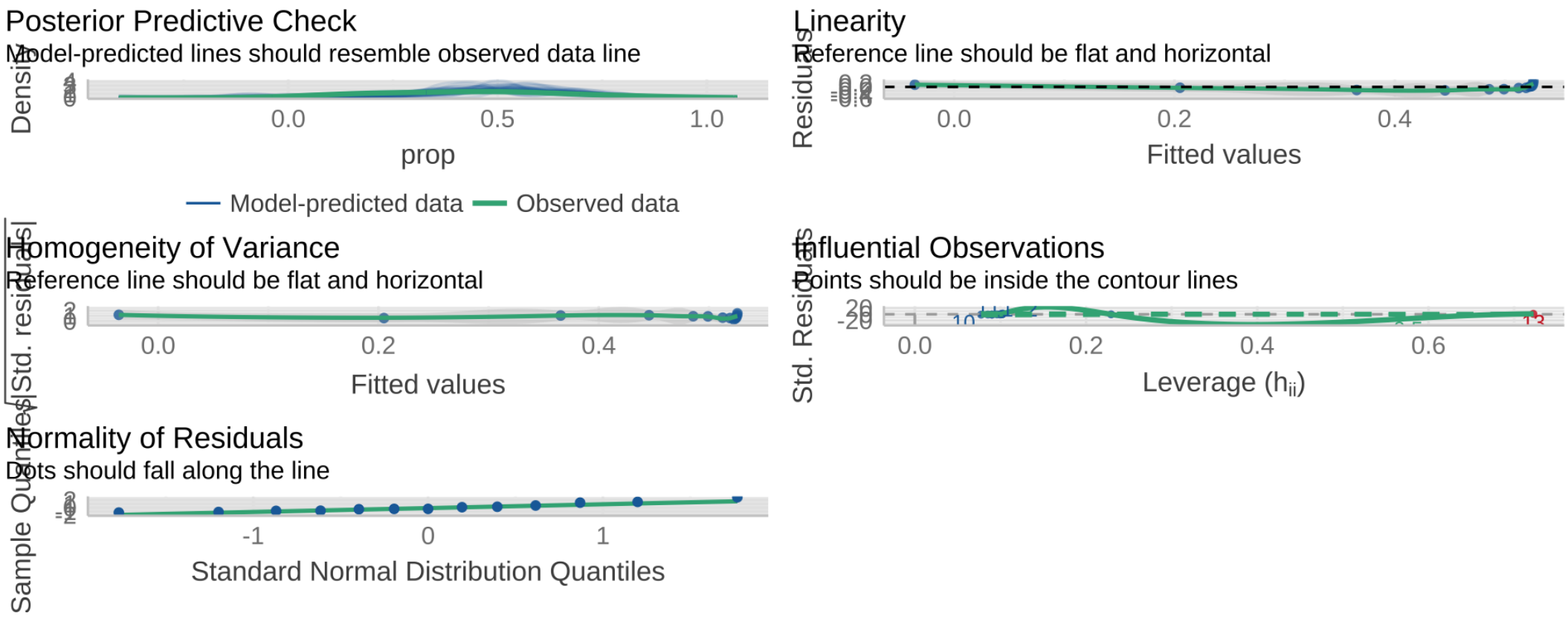
```
## [1]  600  650  700 1000 4000
```

Predictor X Transformations

- Make more linear
- Normality
- Dont do much to fix heteroscadacity
- Lets plot our memory data

Other Checks

```
lm(prop~time, data=log_df)%>%
check_model()
```

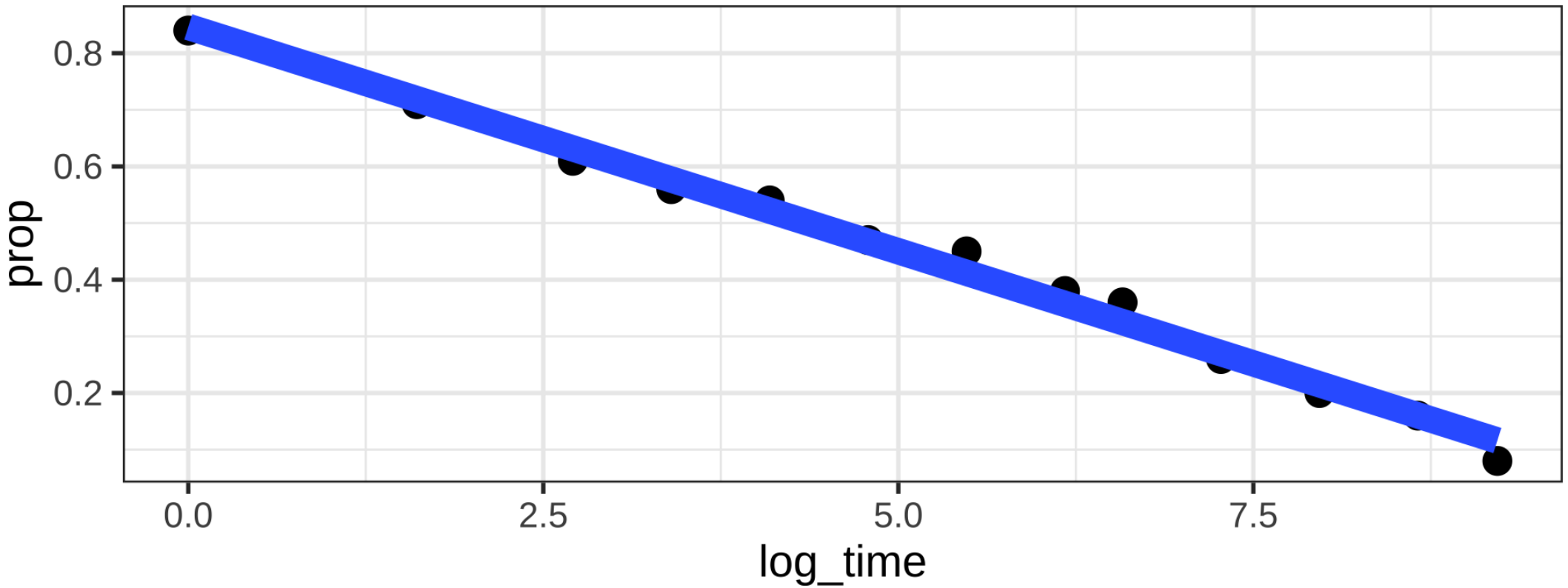


Better?

```
log_df <- log_df %>%
  mutate(log_time=log(time))
```

Check Again

- Much better!



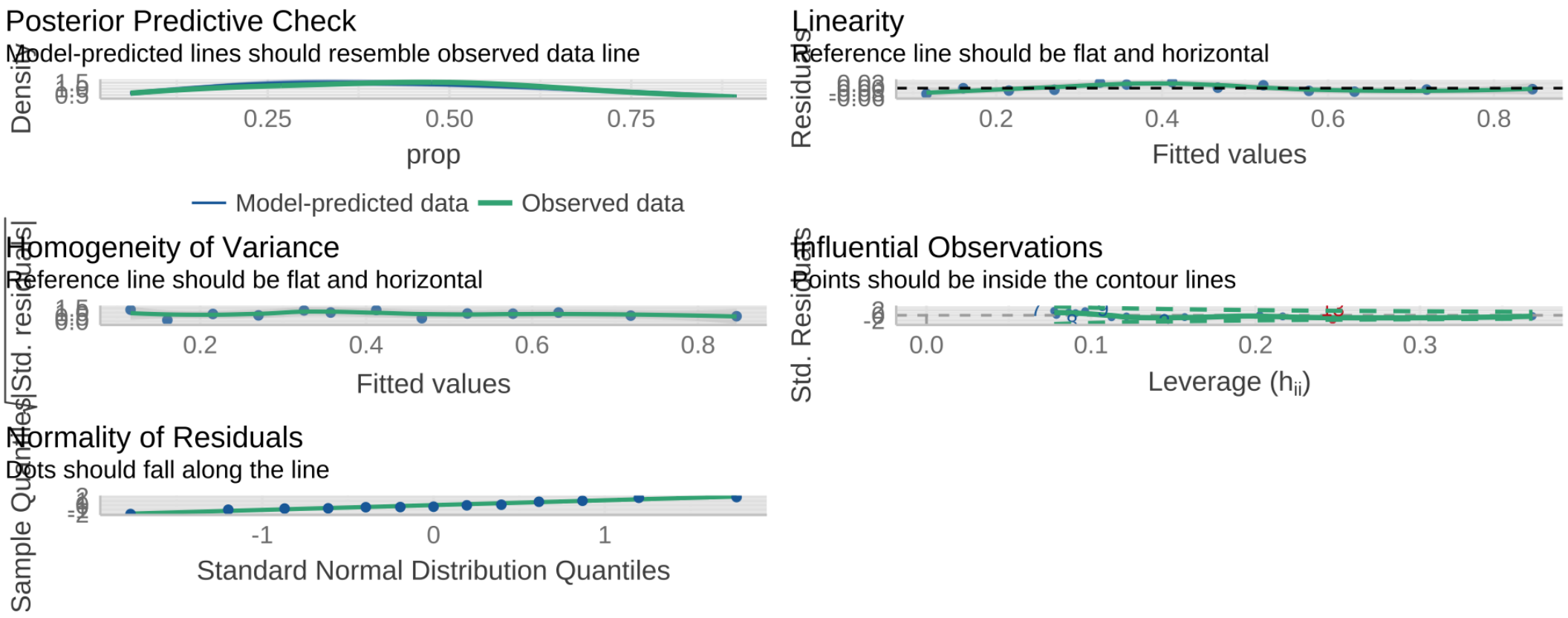
Let's Analyze

```
lm(prop~log_time, data=log_df)
```

```
##
## Call:
## lm(formula = prop ~ log_time, data = log_df)
##
## Coefficients:
## (Intercept)     log_time
##     0.84642     -0.07923
```

Check Assumptions Again

```
lm(prop~log_time, data=log_df)%>%
check_model()
```



Interpretation of Log transformation

- When using logarithms, you model percentage increase or decrease instead of absolute differences
- Not: RTs are 40 ms slower in the incongruent trials
- But: RTs are 7% slower in incongruent trials

Outcome Y Transformations

- Make Y more linear
- Make Y normal
- Help correct heteroscadacity

```
knitr::include_graphics("log.bmp")
```

Outcome Y Transformations

- Square root ($\sqrt(y)$)
- Inverse transformations (1/y)
    - Makes interpretation hard!

Rules for Interpretation

- Only the dependent/response variable?
    - Exponentiate the coefficient, subtract one from this number, and multiply by 100. This gives the percent increase (or decrease) in the response for every one-unit increase in the independent variable. Example: the coefficient is 0.198. (exp(0.198) – 1) * 100 = 21.9. For every one-unit increase in the independent variable, our dependent variable increases by about 22%
- Only independent/predictor variable(s) is log-transformed?
    - Divide the coefficient by 100. This tells us that a 1% increase in the independent variable increases (or decreases) the dependent variable by (coefficient/100) units. Example: the coefficient is 0.198. 0.198/100 = 0.00198. For every 1% increase in the independent variable, our dependent variable increases by about 0.002. For x

Rules for Interpretation

- When dependent/response variable and independent/predictor variable(s) are log-transformed
    - Interpret the coefficient as the percent increase in the dependent variable for every 1% increase in the independent variable
    - Example: the coefficient is 0.198. For every 1% increase in the independent variable, our dependent variable increases by about 0.20%. For x percent increase, calculate 1.x to the power of the coefficient, subtract 1, and multiply by 100. Example: For every 20% increase in the independent variable, our dependent variable increases by about (1.20 0.198 – 1) * 100 = 3.7 percent.

When Should you Log Transform?

- Ideally: When it's theoretically motivated
- After you look at the relationship to DV
- Common in linguistics and psychology:
    - Frequency
    - Response times
    - Perceptual magnitudes
- *If you want/need to center, apply log transform after!*

Data Transformations

1. If the primary problem with your model is non-linearity, look at a scatter plot of the data to suggest transformations that might help

2. If the variances are unequal and/or error terms are not normal, try a "power transformation"

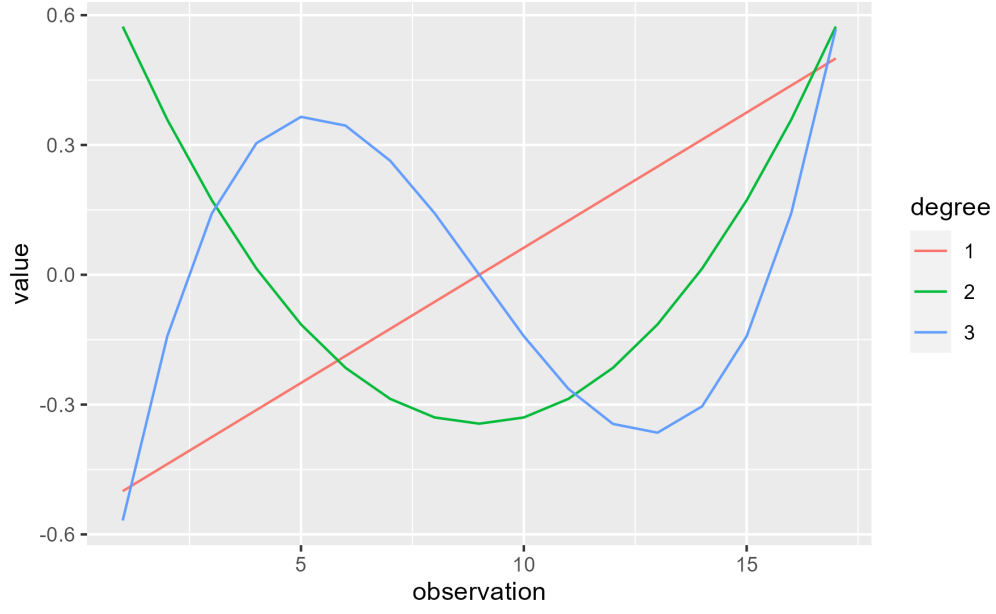
    - Box-Cox Transformation

3. Be transparent!

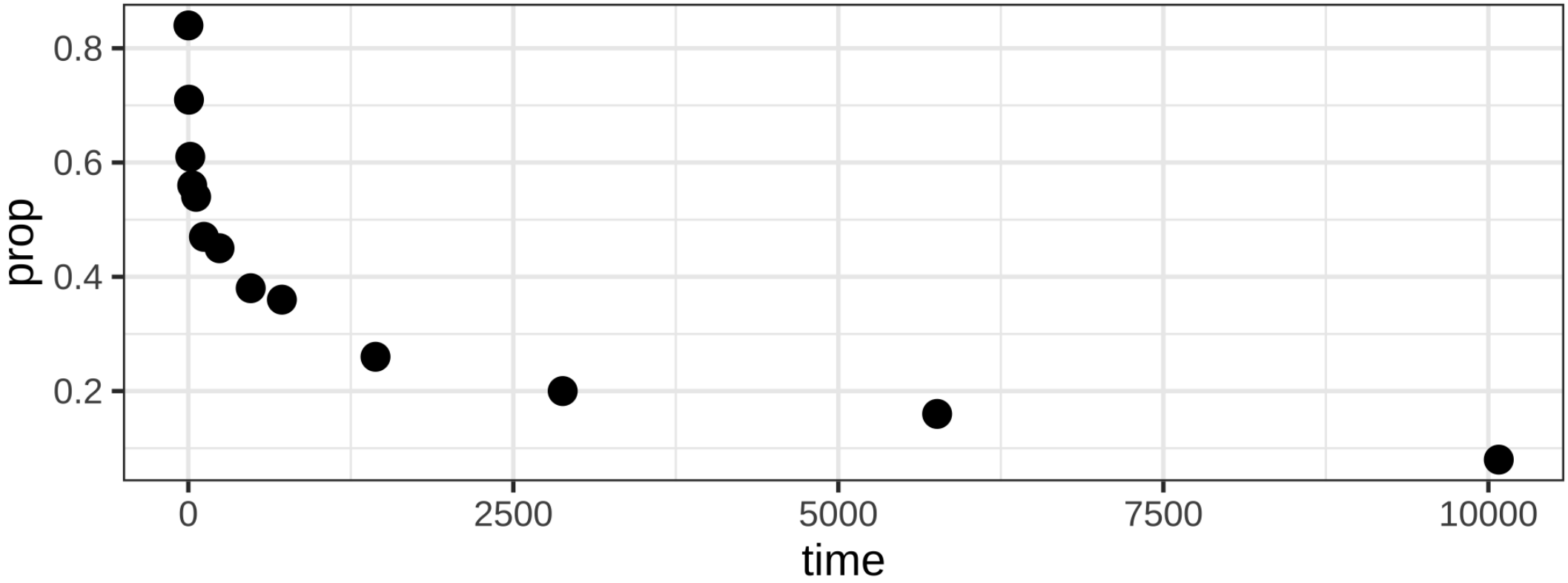
Polynomial Models

Polynomial Models

- A nonlinear regression method that models the relationship between X and Y using polynomials
- Polynomial is mathematical expression of operators and non-negative powers



Memory Example



Testing Polynomial Models

- Analyze all the terms in model
- Forward selection
  - Testing some of the X terms, starting with the lowest order term and including the next higher-order terms from there
  - Include all lower terms of that variable

Polynomial Regression

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i.$$

```
log_df_quad <- log_df%>%
  mutate(time2=time^-2) # add in quadratic

lm(prop ~ time + time2,  data=log_df_quad) %>%
  model_parameters()
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	0.487	0.04	0.95	0.398	0.576	12.2	10	2.57e-07
time	-4.99e-05	1.14e-05	0.95	-7.54e-05	-2.44e-05	-4.36	10	0.00141
time2	0.362	0.125	0.95	0.084	0.64	2.9	10	0.0158

Model Fit Quadratic

```
anim.1<- ggplot(log_df, aes(time,prop))+
   geom_point(size=5)+
  theme_bw(base_size=18)+
 stat_smooth(method = "lm", formula = y ~ x + I(x^2), size = 1)

anim.1
```

Model Fit Comparison

```
# all
lm_quad <- lm(prop ~ time + time2 + I(time^3),  data=log_df_quad)
# forward
lm_lin <- lm(prop ~ time,  data=log_df_quad)
lm_quad <- lm(prop ~ time + time2,  data=log_df_quad)
anova(lm_lin, lm_quad)
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
11	0.255				
10	0.139	1	0.117	8.42	0.0158

In-Class Activity

- English Lexicon Project
    - High frequency words responded to faster than low frequency words

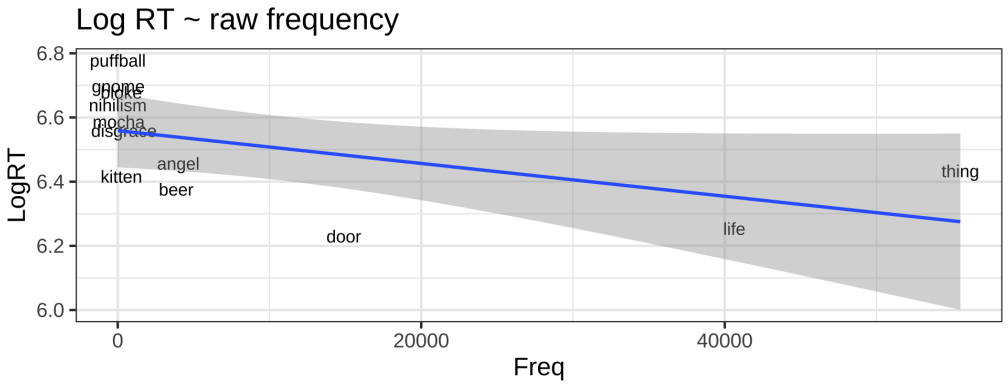
```
# Load the frequency data

ELP <- read_csv("https://raw.githubusercontent.com/jgeller112/psy503-psych_stats/master/static/slides/12-Transf
```

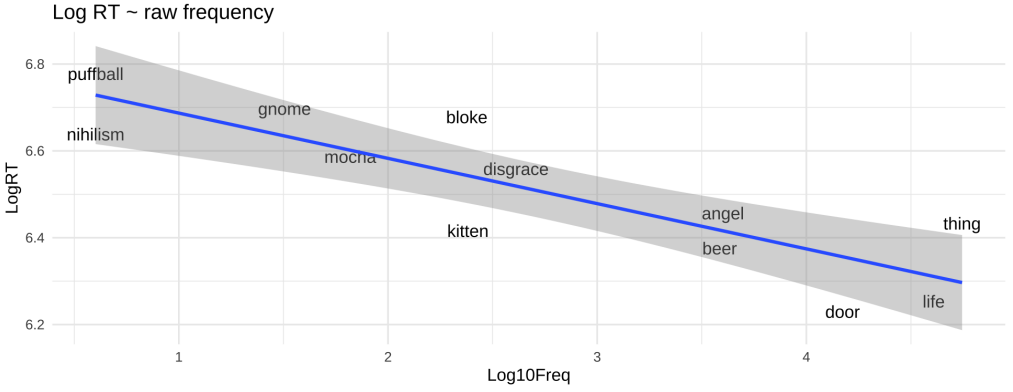
Log Transformation

```
# Log10-transform frequency, log-transform RTs:

ELP <- mutate(ELP,
              Log10Freq = log10(Freq),
              LogRT = log(RT))

head(ELP) %>%
  flextable()
```

Word	Freq	RT	Log10Freq	LogRT
thing	55,522	621.77	4.744465	6.432570
life	40,629	519.56	4.608836	6.252982
door	14,895	507.38	4.173041	6.229260
angel	3,992	636.56	3.601191	6.456079
beer	3,850	587.18	3.585461	6.375331
disgrace	409	705.00	2.611723	6.558198

Plot


Log RT ~ raw frequency
LogRT
Freq
6.8
6.6
6.4
6.2
6.0
0
20000
40000
puffball
gnome
bloke
nihilism
mocha
disgrace
kitten
angel
beer
door
life
thing


Log RT ~ raw frequency
LogRT
Log10Freq
6.8
6.6
6.4
6.2
1
2
3
4
puffball
nihilism
gnome
mocha
bloke
disgrace
kitten
angel
beer
door
life
thing

Regression

```
# Fit a regression model:
ELP_mdl <- lm(LogRT ~ Log10Freq, data = ELP) %>%
model_parameters()
```

Centering & Standardizing

```
# Center and standardize in one go:

ELP <- mutate(ELP,
              Log10Freq_c = Log10Freq - mean(Log10Freq),
              Log10Freq_z = Log10Freq_c / sd(Log10Freq_c))

# Select the frequency columns to compare:

ELP %>%
  dplyr::select(Freq, Log10Freq, Log10Freq_c, Log10Freq_z)
```

Freq	Log10Freq	Log10Freq_c	Log10Freq_z
5.55e+04	4.74	2.03	1.41
4.06e+04	4.61	1.89	1.31
1.49e+04	4.17	1.46	1.01
3.99e+03	3.6	0.884	0.614

Centering & Standardizing

```
# Same as before, but this time using datawizard:

ELP <- mutate(ELP,
              Log10Freq_c = datawizard::center(Log10Freq),
              Log10Freq_z =datawizard::standardise(Log10Freq))
```

Regression

```
# Fit raw, centered and unc
ELP_mdl_c <- lm(LogRT ~ Log10Freq_c, ELP) %>%
  model_parameters()

ELP_mdl_z <- lm(LogRT ~ Log10Freq_z, ELP) %>%
  model_parameters()
```