EPsy 8252 Notes

Andrew Zieffler

2019-02-05

Contents

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")

options(max.print = "75")
options(scipen = 5)
options(digits = 4)
options("yaml.eval.expr" = TRUE)

opts_chunk$set(prompt = FALSE, comment = NA, message = FALSE, warning = FALSE, tidy = FALSE, fig.align =
opts_knit$set(width = 85)
```

Introduction

This website includes the notes for *EPsy 8252: Methods in Data Analysis for Educational Research II.* To download a ZIP file of the materials that make up this website (e.g., RMD files, HTML files) click here or visit the github repository.

The course website, which includes the syllabus, assignments, etc. can be found at http://zief0002.github.io/epsy-8252/

Chapter 1

R Markdown

In this set of notes, you will learn how to integrate R syntax directly into your word-processed documents to create more reproducible reports.

Preparation

Before class you will need to do the following:

- Download the sample BibTeX file
- Download the CSL style file for the American Psychological Association 6th edition (single-spaced bibliography) from Zotero's repository.
- Install the R package **tinytex**. See the documentation here.

Read the following:

- Rmarkdown (and friends) Tutorial

1.1 Notes

The notes and files you will need can be found at:

- Unit 01: R Markdown [Class Notes]

1.2 Other Resources

In addition to the notes and what we cover in class, there many other resources for learning about R Markdown. Here are some resources that may be helpful in that endeavor:

- R Markdown documentation: Official R Markdown documentation from RStudio

- R Markdown cheatsheet: What it sounds like; a cheatsheet for R Markdown
- knitr: Document and code chunk options for R Markdown
- R Markdown Gallery: - Gallery of some R Markdown outputs
- Pimp my Rmd: Blog post providing a few tips to improve the appearance of output documents.

For **typesetting equations** using R Markdown, check out:

- Using LaTeX to write mathematical content

For integrating **references** into R Markdown, here are a few resources:

- Zotero CSL style repository
- Export a BibTeX file from Mendeley
- Export a BibTeX file from Zotero

Here are some references for using reveal.js and remark.js to create sweet-looking **presentations**:

- Reveal.js presentations
- Customizing Reveal.js presentations
- xaringan

Finally here are some other tools for using Markdown in academia:

- Create a poster with the posterdown package

Pretty-Printing Tables in Markdown

Often it is useful to format table output to make it look good or to adhere a particular style (e.g., APA). There are several packages that help in this endeavor when working in an Rmarkdown document. Below the primary tools used are:

- The `kable()` function from the **knitr** package; and
- Functions from the **kableExtra** package.

Other packages for formatting tables, among others, include the gt package, the huxtable package, and the expss package. For complete APA formatting check out the papaja package.

The primary input to the `kable()` function is a data frame. This data frame will include the primary contents for you table. A data frame can either be created as the ouput of a function or directly using the `data.frame()` function. Below I will create and format a couple different common tables produced for statistical reports. To do so, I will use data from *ed-schools-2018.csv* file (see the data codebook here). These data include institutional-level attributes for several graduate education schools/programs rated by *U.S. News and World Report* in 2018.

```r
# Load libraries
library(AICcmodavg)
library(broom)
library(corrr)
library(dplyr)
library(knitr)
library(kableExtra)
library(readr)
library(tidyr)

# Read in data
ed = read_csv(file = "~/Documents/github/epsy-8252/data/ed-schools-2018.csv")

# Drop rows with missing data
educ = ed %>%
  drop_na()

# Create log-transformed variables
educ = educ %>%
  mutate(
    Lpeer = log(peer),
    Ldoc_accept = log(doc_accept),
    Lenroll = log(enroll)
    )
```

Summary Statistics Table

Say we wanted to produce a table of means and standard deviations for the variables: `peer`, `doc_accept`, and `enroll`. Furthermore, we want these for both the raw (untransformed) and log-transformed versions of these variables. Here is a sketch of what the table should look like:

[INSERT SKETCH]

To begin we will set up a data frame that includes the information from the table. We do this manually to illustrate use of the `data.frame()` function to set up a data frame.

```r
tab_01 = data.frame(
  Measure = c("Peer rating", "Ph.D. acceptance rate", "Enrollment"),
  M_1  = c(mean(educ$peer), mean(educ$doc_accept), mean(educ$enroll)),
  SD_1 = c(sd(educ$peer), sd(educ$doc_accept), sd(educ$enroll)),
  M_2  = c(mean(educ$Lpeer), mean(educ$Ldoc_accept), mean(educ$Lenroll)),
  SD_2 = c(sd(educ$Lpeer), sd(educ$Ldoc_accept), sd(educ$Lenroll))
)

tab_01
```

```
                Measure        M_1        SD_1      M_2      SD_2
1           Peer rating   3.312295   0.4893203 1.187198 0.1439847
2 Ph.D. acceptance rate  40.113115  20.2276300 3.525419 0.6461164
3            Enrollment 969.762295 664.9454219 6.657939 0.7228670
```

We can now use the `kable()` function to rename the columns, round the numeric values, and set a caption.

```r
kable(
  tab_01,
  col.names = c("Measure", "*M*", "*SD*", "*M*", "*SD*"),
  digits = 2,
  caption = "Means and Standard Deviations of Three Measures of Graduate Programs of Education ($n=122$
  )
```

Means and Standard Deviations of Three Measures of Graduate Programs of Education ($n = 122$)

Measure

M

SD

M

SD

Peer rating

3.31

0.49

1.19

0.14

Ph.D. acceptance rate

40.11

20.23

3.53

0.65

Enrollment

969.76

664.95

6.66

0.72

Finally, we use functions from the **kableExtra** package to add our top header row.

```r
kable(
  tab_01,
  col.names = c("Measure", "*M*", "*SD*", "*M*", "*SD*"),
  align = c("l", "c", "c", "c", "c"),
  digits = 2,
  caption = "Means and Standard Deviations of Three Measures of Graduate Programs of Education ($n=122$
  ) %>%
  add_header_above(
    header = c(" " = 1, "Untransformed" = 2, "Log-transformed" = 2)
    ) %>%
  footnote(
    general = "Variables were log-transformed using the natural logarithm.",
    general_title = "Note.",
    footnote_as_chunk = TRUE
    )
```

Means and Standard Deviations of Three Measures of Graduate Programs of Education ($n = 122$)

Untransformed

Log-transformed

Measure

M

SD

M

SD

Peer rating

3.31

0.49

1.19

0.14

Ph.D. acceptance rate

40.11

20.23

3.53

0.65

Enrollment

969.76

664.95

6.66

0.72

Note. Variables were log-transformed using the natural logarithm.

Correlation Table

For our second example, say we wanted to produce a table of pairwise correlations for the variables: `peer`, `doc_accept`, and `enroll`. To begin we will again set up a data frame, but this time we will generate it using functions from the **corrr** package.

```
tab_02 = educ %>%
  select(peer, doc_accept, enroll) %>%
  correlate() %>%
  shave(upper = TRUE) %>%
  fashion(decimals = 2, na_print = "-")

tab_02
```

```
     rowname peer doc_accept enroll
1       peer    -          -      -
2 doc_accept -.54          -      -
3     enroll  .10       -.03      -
```

Now we change the values in the `rownames` column by mutating in new values, and pipe this into the `kable()` function, where we will change the column name and add a caption. Keeping the default alignment will align the decimal points within columns.

```
tab_02 %>%
  mutate(
    rowname = c("1. Peer rating", "2. Ph.D. acceptance rate", "3. Enrollment")
  ) %>%
  kable(
    caption = "Correlations between Three Measures of Graduate Programs of Education",
    col.names = c("Measure", "1", "2", "3")
  )
```

Correlations between Three Measures of Graduate Programs of Education

Measure

1

2

3

1. Peer rating
   —
   —
   —

   2. Ph.D. acceptance rate
      -.54
      —
      —

      3. Enrollment
         .10
         -.03
         —

Regression Table: Single Model

It is common to report the coefficient-level information from a fitted regression model in a table. The nice thing about using the `tidy()` function to obtain coefficient-level information from a fitted model is that the output is formatted as a data frame. Thus, we can use the output from `tidy()` directly in the `kable()` function. Below I fit a regression model and then use piping to obtain the coefficient-level information and create the table.

```
lm(peer ~ 1 + doc_accept + gre_verbal, data = educ) %>%
  tidy() %>%
  kable()
```

term

estimate

std.error

statistic

p.value

(Intercept)

-1.5268

1.6668

-0.916

0.3615

doc_accept

-0.0107

0.0019

-5.519

0.0000

gre_verbal

0.0340

0.0106

3.221

0.0016

To format this, we might want to change the column names and round the numerical information to a better number of digits; typically *p*-values are rounded to three decimal places and coefficients, standard errors and *t*-values are rounded to two digits.

```r
lm(peer ~ 1 + doc_accept + gre_verbal, data = educ) %>%
  tidy() %>%
  kable(
    caption = "Coefficient-Level Estimates for a Model Fitted to Estimate Variation in Peer Ratings.",
    col.names = c("Predictor", "B", "SE", "t", "p"),
    digits = c(0, 2, 3, 2, 3)
  )
```

Coefficient-Level Estimates for a Model Fitted to Estimate Variation in Peer Ratings.

Predictor

B

SE

t

p

(Intercept)

-1.53

1.667

-0.92

0.362

doc_accept

-0.01

0.002

-5.52

0.000

gre_verbal

0.03

0.011

3.22

0.002

Last things to fix are the predictor names and the *p*-values. The rounding of the *p*-values has rendered them as zero. We can use the `pvalue()` function from the **scales** package to better format the column of *p*-values. This is carried out prior to piping the output into the `kable()` function by changing the values in the `p.value` column. (Note that rather than load a package for a single function we can specify the package directly prior to the function name using two colons; `scales::pvalue()`.) Similarly, we can change the names in the `term` column at the same time. Lastly, we note that the SEs were truncated when we rounded, so we fix that by increasing the number of digits displayed in that column.

```r
lm(peer ~ 1 + doc_accept + gre_verbal, data = educ) %>%
  tidy() %>%
  mutate(
    p.value = scales::pvalue(p.value),
    term = c("Intercept", "Ph.D. acceptance rate", "Verbal GRE score")
  ) %>%
  kable(
    caption = "Coefficient-Level Estimates for a Model Fitted to Estimate Variation in Peer Ratings.",
    col.names = c("Predictor", "B", "SE", "t", "p"),
    digits = c(0, 2, 3, 2, 3)
  )
```

Coefficient-Level Estimates for a Model Fitted to Estimate Variation in Peer Ratings.

Predictor

B

SE

t

p

Intercept

-1.53

1.667

-0.92

0.362

Ph.D. acceptance rate

-0.01

0.002

-5.52

<0.001

Verbal GRE score

0.03

0.011

3.22

0.002

One last tweak is that now in the column of *p*-values, the alignment of the decimal place is off (default alignment for text is left-aligned). We can fix this by changing the alignment to be right-aligned. This is useful for numeric values so that the decimal points within a column line up.

```r
lm(peer ~ 1 + doc_accept + gre_verbal, data = educ) %>%
  tidy() %>%
  mutate(
    p.value = scales::pvalue(p.value),
    term = c("Intercept", "Ph.D. acceptance rate", "Verbal GRE score")
  ) %>%
  kable(
    caption = "Coefficient-Level Estimates for a Model Fitted to Estimate Variation in Peer Ratings.",
    col.names = c("Predictor", "B", "SE", "t", "p"),
    digits = c(0, 2, 3, 2, 3),
    align = c("l", "r", "r", "r", "r")
  )
```

Coefficient-Level Estimates for a Model Fitted to Estimate Variation in Peer Ratings.

Predictor

B

SE

t

p

Intercept

-1.53

1.667

-0.92

0.362

Ph.D. acceptance rate

-0.01

0.002

-5.52

<0.001

Verbal GRE score

0.03

0.011

3.22

0.002

Regression Table: Multiple Models

There are several specific packages that help us create tables of regression results. The Stargazer package, the texreg package and the finalfit package are but a few of these.

I tend to use both the **texreg** package (more customizable) and the **stargazer** package (easier). Below I illustrate how to create a table of regression results using the **stargazer** package. First we fit a few models.

```
# Fit candidate models
lm.1 = lm(peer ~ 1 + doc_accept, data = educ)
lm.2 = lm(peer ~ 1 + enroll, data = educ)
lm.3 = lm(peer ~ 1 + doc_accept + enroll, data = educ)
```

After loading the **stagazer** package, the `stargazer()` function can be used to create a basic table of regression results. The `type=` argument defaults to `latex`, so if you are rendering to an HTML document, you need to change this to `type="html"`.

```r
library(stargazer)

stargazer(lm.1, lm.2, lm.3, type = "html")
```

```
<table style="text-align:center"><tr><td colspan="4" style="border-bottom: 1px solid black"></td></tr><t
<tr><td></td><td colspan="3" style="border-bottom: 1px solid black"></td></tr>
<tr><td style="text-align:left"></td><td colspan="3">peer</td></tr>
<tr><td style="text-align:left"></td><td>(1)</td><td>(2)</td><td>(3)</td></tr>
<tr><td colspan="4" style="border-bottom: 1px solid black"></td></tr><tr><td style="text-align:left">doc
<tr><td style="text-align:left"></td><td>(0.002)</td><td></td><td>(0.002)</td></tr>
<tr><td style="text-align:left"></td><td></td><td></td><td></td></tr>
<tr><td style="text-align:left">enroll</td><td></td><td>0.0001</td><td>0.0001</td></tr>
<tr><td style="text-align:left"></td><td></td><td>(0.0001)</td><td>(0.0001)</td></tr>
<tr><td style="text-align:left"></td><td></td><td></td><td></td></tr>
<tr><td style="text-align:left">Constant</td><td>3.836<sup>***</sup></td><td>3.238<sup>***</sup></td><td
<tr><td style="text-align:left"></td><td>(0.083)</td><td>(0.078)</td><td>(0.101)</td></tr>
<tr><td style="text-align:left"></td><td></td><td></td><td></td></tr>
<tr><td colspan="4" style="border-bottom: 1px solid black"></td></tr><tr><td style="text-align:left">Obs
<tr><td style="text-align:left">R<sup>2</sup></td><td>0.292</td><td>0.011</td><td>0.300</td></tr>
<tr><td style="text-align:left">Adjusted R<sup>2</sup></td><td>0.286</td><td>0.003</td><td>0.288</td></t
<tr><td style="text-align:left">Residual Std. Error</td><td>0.414 (df = 120)</td><td>0.489 (df = 120)</t
<tr><td style="text-align:left">F Statistic</td><td>49.400<sup>***</sup> (df = 1; 120)</td><td>1.328 (d
<tr><td colspan="4" style="border-bottom: 1px solid black"></td></tr><tr><td style="text-align:left"><em
</table>
```

The function outputs raw HTML (or LaTeX), so to get it to form into a table you need to include `results='asis'` in your Rmarkdown chunk.

~~~
```{r message=FALSE, results='asis'}
library(stargazer)

stargazer(lm.1, lm.2, lm.3, type = "html")
```
~~~

Dependent variable:

peer

(1)

(2)

(3)

doc_accept

-0.013***

-0.013***

(0.002)

(0.002)

enroll

0.0001

0.0001

(0.0001)

(0.0001)

Constant

3.836***

3.238***

3.769***

(0.083)

(0.078)

(0.101)

Observations

122

122

122

R2

0.292

0.011

0.300

Adjusted R2

0.286

0.003

0.288

Residual Std. Error

0.414 (df = 120)

0.489 (df = 120)

0.413 (df = 119)

F Statistic

49.400*** (df = 1; 120)

1.328 (df = 1; 120)

25.490*** (df = 2; 119)

Note:

p<0.1; ***p<0.05;*** p<0.01

There are several arguments in the `stargazer()` function to customize the table.

```r
stargazer(
  lm.1, lm.2, lm.3,
  type = "html",
  title = "Three Regression Models Predicting Variation in Peer Ratings",
  column.labels = c("Model A", "Model B", "Model C"),
  colnames = FALSE,
  model.numbers = FALSE,
  dep.var.caption = " ",
  dep.var.labels = "Peer rating (1-5 scale)",
  covariate.labels = c("Ph.D. acceptance rate", "Enrollment"),
  keep.stat = c("rsq", "f"),
  notes.align = "l",
  add.lines = list(c("Corrected AIC", round(AICc(lm.1), 1), round(AICc(lm.2), 1), round(AICc(lm.3), 1))
  out = "images/table1.html"
  )
```

Three Regression Models Predicting Variation in Peer Ratings

Peer rating (1-5 scale)

Model A

Model B

Model C

Ph.D. acceptance rate

-0.013***

-0.013***

(0.002)

(0.002)

Enrollment

0.0001

0.0001

(0.0001)

(0.0001)

Constant

3.836***

3.238***

3.769***

(0.083)

(0.078)

(0.101)

Corrected AIC

R2

0.292

0.011

0.300

F Statistic

49.400*** (df = 1; 120)

1.328 (df = 1; 120)

25.490*** (df = 2; 119)

Note:

p<0.1; ***p<0.05;*** p<0.01

There is a known bug with the stargazer table not printing the asterisks next to the significance values in the notes section of the table when outputting to HTML. The solution as documented here is to output the html code to an external file using the `out=` argument in `stargazer()` and then inserting that html code in a new code chunk via the `includeHTML()` function from the **htmltools** package.

The `add.lines=` argument adds a line to the bottom of the output. This argument takes a list that includes the name you want to output in the regression table and then the value to output for each of the models. Here we computed the corrected AIC value using the `AICc()` function from the **AICmodavg** package for each of the models. (Note: We will learn about this in the More Information Criteria for Model Selection unit.)

Chapter 2

Nonlinearity: Log-Transforming the Predictor

In this set of notes, you will learn about log-transforming the predictor in a regression model to account for nonlinearity.

Preparation

Before class you will need to do the following:

- Refresh your knowledge about logarithms by going though the Khan Academy Intro to Logarithms tutorial.

2.1 Dataset and Research Question

The data we will use in this set of notes, *mn-schools.csv* (see the data codebook here), contains 2011 institutional data for $n = 33$ Minnesota colleges and universities.

```r
# Load libraries
library(broom)
library(dplyr)
library(ggplot2)
library(readr)
library(sm)
library(tidyr)

# Import data
mn = read_csv(file = "~/Documents/github/epsy-8252/data/mn-schools.csv")
head(mn)
```

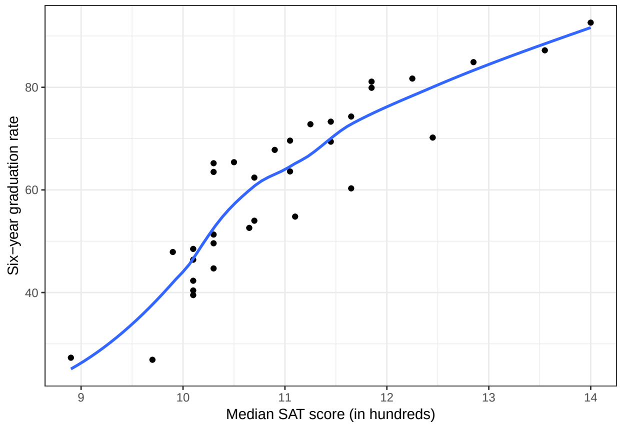


Figure 2.1: Scatterplot of the relationship between median SAT score and six-year graduation rate. The loess smoother is also displayed.

```
# A tibble: 6 x 6
     id name                            grad public   sat tuition
  <dbl> <chr>                          <dbl>  <dbl> <dbl>   <dbl>
1     1 Augsburg College                65.2      0  10.3    39.3
2     3 Bethany Lutheran College        52.6      0  10.6    30.5
3     4 Bethel University, Saint Paul, MN  73.3      0  11.4    39.4
4     5 Carleton College                92.6      0  14      54.3
5     6 College of Saint Benedict       81.1      0  11.8    43.2
6     7 Concordia College at Moorhead   69.4      0  11.4    36.6
```

Using these data, we will examine if (and how) academic "quality" of the student-body (measured by median composite SAT score) is related to institutional graduation rates.

2.2 Log-Transformation of a Variable

Recall that the scatterplot of SAT scores and graduation rates suggested that the relationship between these variables was curvilinear.

One way to model this nonlinearity was to fit a model that included a polynomial effect (quadratic). Another method of modeling nonlinearity is to transform the predictor (or outcome) using a nonlinear transformation. One commonly used nonlinear transformation is the logarithm. Below is a comparison of the quadratic function to the logarithmic function.

The quadratic function shows continuous and diminishing growth followed by continuous and increasing loss (parabola; the function changes direction), while the logarithmic function models continuous, albeit diminishing, growth (the function does not change direction).

2.2.1 Quick Refresher on Logarithms

The logarithm is an inverse function of an exponent. Consider this example,

$$\log_2(32)$$

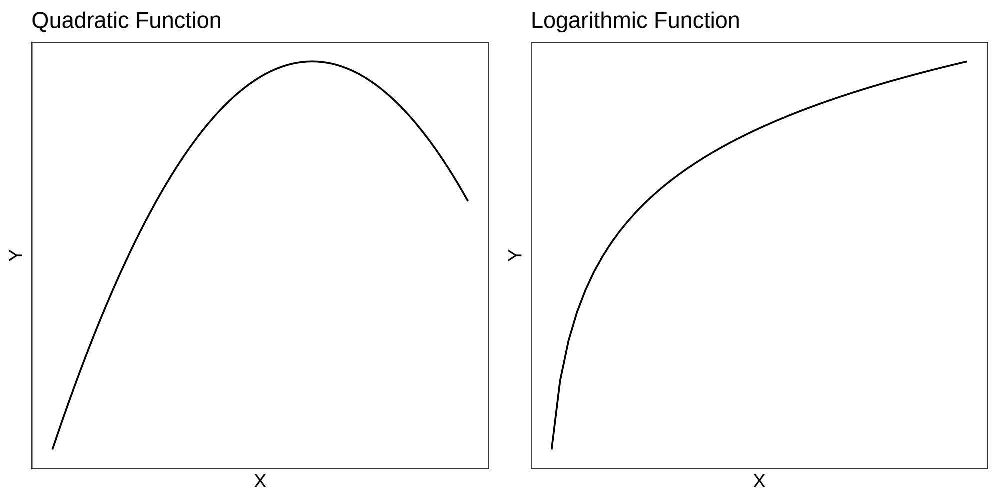


Figure 2.2: Quadratic and logarithmic functions.

The logarithm of 32 is the exponent to which the base, 2 in our example, must be raised to produce that number. In other words,

$$\log_2(32) \longrightarrow 2^x = 32 \longrightarrow x = 5$$

Thus,

$$\log_2(32) = 5$$

To compute a logarithm using R, we use the `log()` function. We also specify the argument `base=`, since logarithms are unique to a particular base. For example, to compute the mathematical expression $\log_2(32)$, we use

```
log(32, base = 2)
```

```
[1] 5
```

There is also a shortcut function to use base-2.

```
log2(32)
```

```
[1] 5
```

2.2.2 Log-Transforming Variables

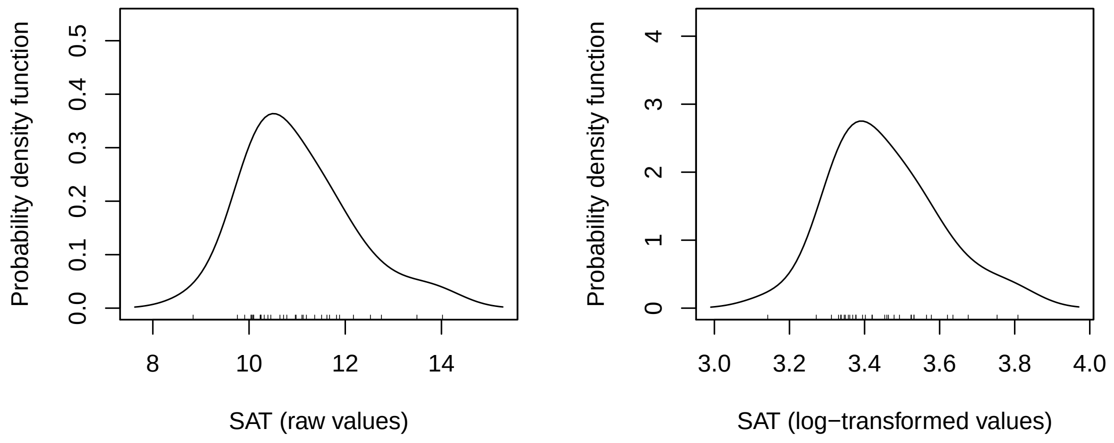
For our purposes, we need to log-transform each value in a particular variable. Here, we will log-transform the SAT variable (using base-2).

```r
mn = mn %>%
  mutate(
    L2sat = log(sat, base = 2)
    )

head(mn)
```

```
# A tibble: 6 x 7
     id name                             grad public   sat tuition L2sat
  <dbl> <chr>                           <dbl>  <dbl> <dbl>   <dbl> <dbl>
1     1 Augsburg College                 65.2      0  10.3    39.3  3.36
2     3 Bethany Lutheran College         52.6      0  10.6    30.5  3.41
3     4 Bethel University, Saint Paul, MN  73.3      0  11.4    39.4  3.52
4     5 Carleton College                 92.6      0  14      54.3  3.81
5     6 College of Saint Benedict        81.1      0  11.8    43.2  3.57
6     7 Concordia College at Moorhead    69.4      0  11.4    36.6  3.52
```

How does this log-transformed variable compare to the original SAT predictor. We can examine the density plot of both the original and log-transformed variables to answer this.



- Comparing the shapes of the two variables, we see that the original variable was right-skewed. The log-transformed variable is also right-skewed, although it is LESS right-skewed than the original.
- The scale is quite different between the two variables (one is, after all, log-transformed). This has greatly affected the variation. After log-transforming, the variation is much smaller.

What happens when we use the log-transformed variable in a scatterplot with graduation rates?

```r
ggplot(data = mn, aes(x = L2sat, y = grad)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  theme_bw() +
  xlab("Log-transformed SAT score") +
  ylab("Six-year graduation rate")
```

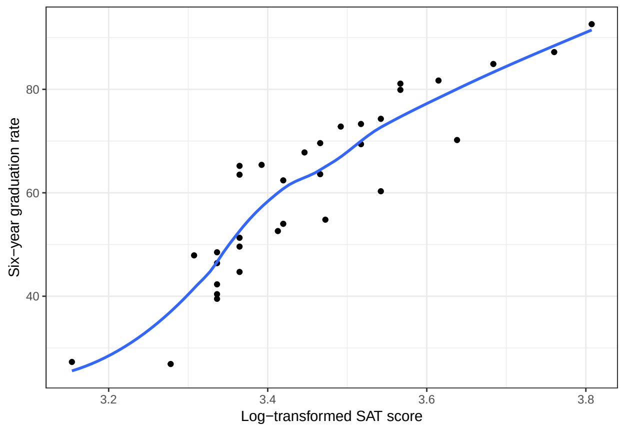


Figure 2.3: Scatterplot of the relationship between log-transformed median SAT score (base-2) and six-year graduation rate. The loess smoother is also displayed.

The relationship between graduation rate and the log-transformed SAT scores is MORE linear than the relationship between graduation rates and the untransformed SAT scores. By transforming the variable using a nonlinear transformation (log) we have "linearized" the relationship with graduation rates. As such, we can fit a linear model to predict graduation rates using the Log-transformed SAT scores as a predictor.

2.3 Fitting the Regression Model

To fit the model, we use the `lm()` function and input the log-transformed SAT scores as the predictor.

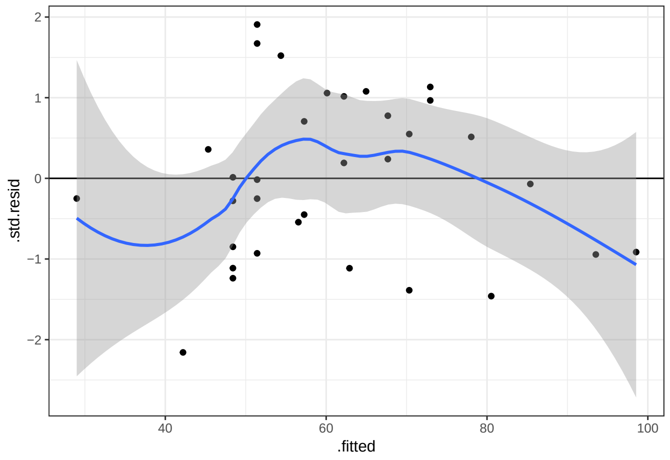
```r
lm.1 = lm(grad ~ 1 + L2sat, data = mn)
```

2.3.1 Examine the Assumption of Linearity

Before examining the coefficients, we can scrutinize the residuals to see whether the log-transformation helped us meet the assumption of linearity.

```r
# Obtain residuals
out = augment(lm.1)

# Check linearity assumptions
ggplot(data = out, aes(x = .fitted, y = .std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_smooth() +
  theme_bw()
```



The assumption looks reasonably met as the horizontal line of $y = 0$ is encompassed in the confidence envelope of the loess smoother.

2.3.2 Interpret the Regression Results

We can now look at the regression output and interpret the results.

```r
# Model-level output
glance(lm.1)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1     0.811         0.805  7.39      133. 9.30e-13     2  -112.  230.  234.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

Examining the model-level output, we see that differences in $\log_2(\text{SAT})$ explain 81.13% of the variation in graduation rates. This is statistically significant, $F(1,\ 31) = 133.3$, $p < .001$. Since differences in $\log_2(\text{SAT})$ imply that there are differences in the raw SAT scores, we would typically just say that "differences in SAT scores explain 81.13% of the variation in graduation rates."

Moving to the coefficient-level output,

```r
# Coefficient-level output
tidy(lm.1)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    -307.     31.9      -9.62 7.94e-11
2 L2sat           106.      9.22     11.5  9.30e-13
```

We can write the fitted equation as,

$$\hat{\text{Graduation Rate}} = -306.7 + 106.4\left[\log_2(\text{SAT})\right]$$

We can interpret the coefficients as we always do, recognizing that these interpretation are based on the log-transformed predictor.

- The intercept value of -306.7 is the predicted average graduation rate for all colleges/universities with a $\log_2(\text{SAT})$ value of 0.
- The slope value of 106.4 indicates that each one-unit difference in $\log_2(\text{SAT})$ is associated with a 106.4-unit difference in graduation rate, on average.

2.3.3 Better Interpretations: Back-transforming

While these interpretations are technically correct, it is more helpful to your readers (and more conventional) to interpret any regression results in the metric of SAT scores rather than log-transformed SAT scores. This means we have to *back-transform the interpretations*. To back-transform a logarithm, we use its inverse function; exponentiation.

We interpreted the intercept as, "the predicted average graduation rate for all colleges/universities with a $\log_2(\text{SAT})$ value of 0". To interpret this using the metric of our SAT attribute, we have to understand what $\log_2(\text{SAT}) = 0$ is.

$$\log_2(\text{SAT}) = 0 \longrightarrow 2^0 = \text{SAT}$$

In this computation, SAT = 1. Thus, rather than using the log-transformed interpretation, we can, instead, interpret the intercept as,

> The predicted average graduation rate for all colleges/universities with a SAT measurement of 1 (median SAT = 100) is -306.7. Since there are no colleges/universities in our data that have a SAT value of 1, this is extrapolation.

What about the slope? Our interpretation was that "each one-unit difference in $\log_2(\text{SAT})$ is associated with a 106.4-unit difference in graduation rate, on average." Working with the same ideas of back-transformation, we need to understand what a one-unit difference in $\log_2(\text{SAT})$ means. Consider four values of $\log_2(\text{SAT})$ that are each one-unit apart:

$$\log_2(\text{SAT}) = 1 \log_2(\text{SAT}) = 2 \log_2(\text{SAT}) = 3 \log_2(\text{SAT}) = 4$$

If we back-transform each of these, then we can see how the four values of the raw SAT variable would differ.

$$\begin{aligned}
\text{SAT} &= 2^1 = 2 \\
\text{SAT} &= 2^2 = 4 \\
\text{SAT} &= 2^3 = 8 \\
\text{SAT} &= 2^4 = 16
\end{aligned}$$

When $\log_2(\text{SAT})$ is increased by one-unit, the raw SAT value is doubled. We can use this in our interpretation of slope:

> A doubling of the SAT value is associated with a 106.4-unit difference in graduation rate, on average.

The technical language for doubling is a "two-fold difference". So we would conventionally interpret this as:

> Each two-fold difference in SAT value is associated with a 106.4-unit difference in graduation rate, on average.

To understand this further, consider a specific school, say Augsburg. Their measurement on the SAT variable is 10.3, and their log-transformed SAT score is 3.36. Using the fitted regression equation (which employs the log-transformed SAT),

```
-306.7 + 106.4 * 3.36
```

```
[1] 50.8
```

Augsburg's predicted graduation rate would be 50.8. If we increase the `L2sat` score by 1 to 4.36 (which is equivalent to a raw SAT measurement of 20.6; double 10.3), their predicted graduation rate is,

```
-306.7 + 106.4 * 4.36
```

```
[1] 157.2
```

This is an increase of 106.4.

2.4 Alternative Method of Fitting the Model

Rather that create the log-transformed SAT score in the data, we can use the `log()` function on SAT directly in the `lm()` computation.

```
lm.1 = lm(grad ~ 1 + log(sat, base = 2), data = mn)

# Model-level output
glance(lm.1)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1     0.811         0.805  7.39      133. 9.30e-13     2  -112.  230.  234.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
# Coefficient-level output
tidy(lm.1)
```

```
# A tibble: 2 x 5
  term               estimate std.error statistic  p.value
  <chr>                 <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)           -307.     31.9      -9.62 7.94e-11
2 log(sat, base = 2)     106.      9.22     11.5  9.30e-13
```

Using this method of fitting the model will be useful as we plot the fitted model.

2.5 Plotting the Fitted Model

To aid interpretation of the effect of SAT on graduation rate, we can plot the fitted model. If we used the method of fitting in which we used `log()` directly in the `lm()` function, we only need to set up a sequence of SAT values, predict graduation rates using the fitted model, and finally connect these values using a line.

```
# Set up data
plot_data = crossing(
    sat = seq(from = 8.9, to = 14.0, by = 0.1)
    ) %>%
  mutate(
    # Predict
    yhat = predict(lm.1, newdata = .)
  )

# Examine data
head(plot_data)
```

```
# A tibble: 6 x 2
    sat  yhat
  <dbl> <dbl>
1   8.9  29.0
2   9    30.7
3   9.1  32.4
4   9.2  34.1
5   9.3  35.7
6   9.4  37.4
```

```
# Plot
ggplot(data = plot_data, aes(x = sat, y = yhat)) +
    geom_line() +
    theme_bw() +
  xlab("Median SAT score (in hundreds)") +
  ylab("Predicted graduation rate")
```

2.6 Different Base Values in the Logarithm

The base value we used in the `log()` function in the previous example was base-2. Using a base value of 2 was an arbitrary choice. We can use any base value we want. For example, what happens if we use base-10.

```
mn = mn %>%
  mutate(
    L10sat = log(mn$sat, base = 10)
  )

# Examine data
head(mn)
```

```
# A tibble: 6 x 8
     id name                         grad public   sat tuition L2sat L10sat
```

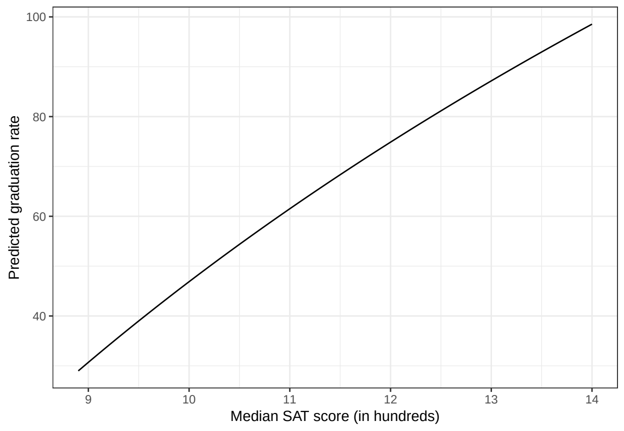


Figure 2.4: Plot of the predicted graduation rates as a function of median SAT score (in hundreds). The non-linearity in the plot indicates that there is a diminishing positive effect of SAT on graduation rates.

```
  <dbl> <chr>                        <dbl>  <dbl> <dbl>   <dbl> <dbl>  <dbl>
1     1 Augsburg College              65.2      0  10.3    39.3  3.36   1.01
2     3 Bethany Lutheran College      52.6      0  10.6    30.5  3.41   1.03
3     4 Bethel University, Saint P~  73.3      0  11.4    39.4  3.52   1.06
4     5 Carleton College              92.6      0  14      54.3  3.81   1.15
5     6 College of Saint Benedict     81.1      0  11.8    43.2  3.57   1.07
6     7 Concordia College at Moorh~  69.4      0  11.4    36.6  3.52   1.06
```

Comparing the logarithms of the SAT attribute using base-10 to those using base-2 we see that the base-10 logarithms are smaller. This is because now we are using the base of 10 in our exponent (rather than 2). For example, for Augsburg,

$$10^{1.013} = 10.3$$

If we fit a model using the base-10 logarithm,

```
lm.2 = lm(grad ~ 1 + log(sat, base = 10), data = mn)

# Model-level output
glance(lm.2)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1     0.811         0.805  7.39      133. 9.30e-13     2  -112.  230.  234.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

Examining the model-level output, we see that differences in $\log_{10}(\text{SAT})$ explain 81.13% of the variation in graduation rates. Or simply, that differences in SAT scores explain 81.13% of the variation in graduation rates. This is statistically significant, $F(1,\ 31) = 133.3$, $p < .001$. These model-level results are the same as when we used the base-2 logarithm.

```
# Coefficient-level output
tidy(lm.2)
```

```
# A tibble: 2 x 5
  term                 estimate std.error statistic  p.value
  <chr>                   <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)             -307.      31.9     -9.62 7.94e-11
2 log(sat, base = 10)      354.      30.6     11.5  9.30e-13
```

The fitted equation is,

$$\hat{\text{Graduation Rate}} = -306.7 + 353.6\bigg[\log_{10}(\text{SAT})\bigg]$$

We can interpret the coefficients using the base-10 logarithm of SAT scores as:

- The intercept value of -306.7 is the predicted average graduation rate for all colleges/universities with a $\log_{10}(\text{SAT})$ value of 0.
- The slope value of 353.6 indicates that each one-unit difference in $\log_{10}(\text{SAT})$ is associated with a 353.6-unit difference in graduation rate, on average.

Better yet, we can *back-transform the interpretations* so that we are using SAT scores rather than $\log_{10}(\text{SAT})$ scores.

- The predicted average graduation rate for all colleges/universities with a SAT value of 1 (median SAT score = 100) is -306.7.
- Each *ten-fold* difference in SAT is associated with a 353.6-unit difference in graduation rate, on average.

To further think about the effect of SAT, if Augsburg improved its median SAT score ten-fold (i.e., going from a SAT value of 10.3 to a value of 103) we would predict its graduation rate to go up by 353.6.

2.6.1 Comparing the Output from the Two Bases

The model-level information is all the same. Furthermore, the intercepts (and SE and p-value) was the same across both models. The slope coefficients and SEs were different in the two models, but the t-value and p-value for the effect of SAT was identical for both base-2 and base-10. The only real difference in using base-10 vs. base-2 in the logarithm is in the interpretation of the SAT effect.

What if we look at the residual fit?

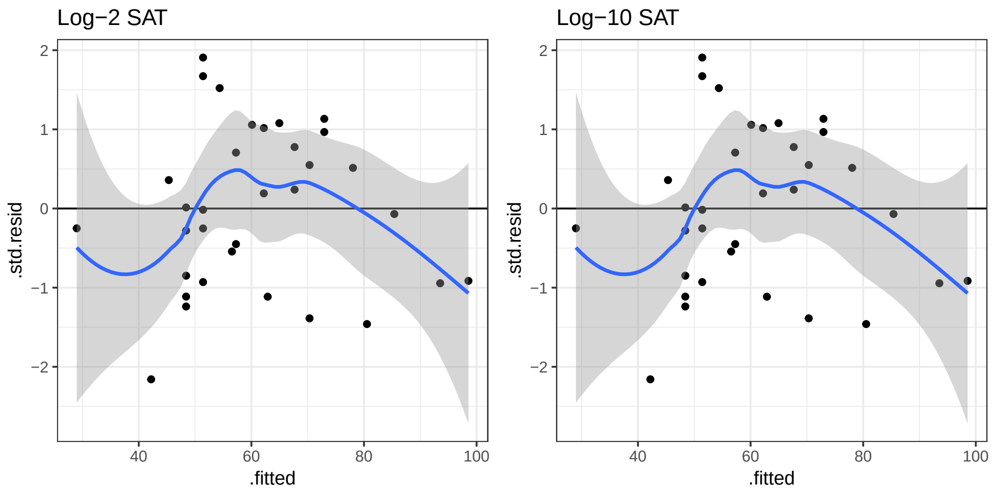


Figure 2.5: Standardized residuals versus the fitted values for the models fitted with the log-2 predictor (left) and the log-10 predictor (right).

The residuals fit EXACTLY the same. Why is this? Let's again use Augsburg as an example. Using the fitted model that employed the base-2 logarithm, we found that Augsburg's predicted graduation rate was,

$$\begin{aligned}\hat{\text{Graduation Rate}} &= -306.7 + 106.4\Big[\log_2(10.3)\Big] \\ &= -306.7 + 106.4\Big[3.36\Big] \\ &= 50.8\end{aligned}$$

Using the model that employed the base-10 logarithm, Augsburg's predicted graduation rate would be

$$\begin{aligned}\hat{\text{Graduation Rate}} &= -306.7 + 353.6\Big[\log_{10}(10.3)\Big] \\ &= -306.7 + 353.6\Big[1.01\Big] \\ &= 50.8\end{aligned}$$

Augsburg's predicted graduation rate is *exactly the same* in the two models. This implies that Augsburg's residual would also be the same in the two models. This is true for every college. Because of this, increasing (or decreasing) the base used in the logarithm does not help improve the fit of the model. The fit is exactly the same no matter which base you choose.

The only thing that changes when you choose a different base is the interpretation of the slope. You should choose the base to facilitate interpretation. For example, does it make more sense to talk about a *two-fold* difference in the predictor? A *five-fold* difference in the predictor? A *ten-fold* difference in the predictor?

2.7 Base-e Logarithm: The Natural Logarithm

In our example, neither of the bases we examined is satisfactory in terms of talking about the effect of SAT. Two-fold differences in SAT are very unlikely, to say anything of ten-fold differences. One base that is commonly used for log-transformations is base-e. e is a mathematical constant (Euler's number) that is approximately equal to 2.71828. We can obtain this by using the `exp()` function in R. This function takes e to some exponent that is given as the argument. So to obtain the approximation of e we use

```
exp(1)
```

```
[1] 2.718
```

The logarithm (base-e) for a number, referred to as the *natural logarithm*, can be obtained using the `log()` function with the argument `base=exp(1)`. However, this base is so commonly used that it is the default value for the `base=` argument. So, if we use the `log()` function without defining the `base=` argument, it will automatically use base-e. For example, the natural logarithm of Augsburg's SAT score of 1030 can be computed as

```
log(10.3)
```

```
[1] 2.332
```

If we took $e^{2.332}$ we would obtain 10.3. The natural logarithm even has its own mathematical notation; ln. For example, we would mathematically express the natural logarithm of 10.3 as

$$\ln(10.3) = 2.332.$$

2.7.1 Using the Natural Logarithm in a Regression Model

Below we regress graduation rates on the log-transformed SAT scores, using the natural logarithm.

```
# Fit model
lm.3 = lm(grad ~ 1 + log(sat), data = mn)

# Model-level output
glance(lm.3)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1     0.811         0.805  7.39      133. 9.30e-13     2  -112.  230.  234.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

As with any base, using base-e results in the same model-level information ($R^2 = .811$, $F(1,\ 31) = 133.3$, $p < .001$).

```
# Coefficient-level output
tidy(lm.3)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    -307.      31.9     -9.62 7.94e-11
2 log(sat)        154.      13.3     11.5  9.30e-13
```

The intercept has the same coefficient ($\hat{\beta}_0 = -306.7$), SE, t-value, and p-value as the intercept from the models using base-2 and base-10 log-transformations of SAT. (This is, again, because $2^0 = 10^0 = e^0 = 1$.) And, although the coefficient and SE for the effect of SAT is again different (a one-unit change in the three different log-scales does not correspond to the same amount of change in raw SAT for the three models), the t-value and level of statistical significance ($t(31) = 11.55$, $p < .001$) for this effect, are the same as when we used base-2 and base-10.

So how can we interpret the model's coefficients?

- The intercept can be interpreted exactly the same as in the previous models in which we used base-2 or base-10; namely that the predicted average graduation rate for colleges/universities with a SAT value of one is -306.7.
- Interpreting the slope, we could say that an e-fold difference in SAT value is associated with a 153.6-unit difference in graduation rates, on average.

2.7.1.1 Interpretation Using Percentage Change

Consider three schools, each having a SAT score that differs by 1%; say these schools have SAT values of 10, 10.1, 10.2. Using the fitted equation, we can compute the predicted graduation rate for each of these hypothetical schools:

$$\hat{\text{Graduation Rate}} = -306.7 + 153.6\bigg[\ln(\text{SAT})\bigg]$$

The SAT values and predicted graduation rates for these schools are given below:

SAT values and Graduation Rates for Three Hypothetical Schools that have SAT Values that Differ by One Percent.

SAT

Predicted Graduation Rate

10.0

46.88

10.1

48.41

10.2

49.93

The difference between each subsequent predicted graduation rate is 1.53.

```
48.4058 - 46.8778
```

```
[1] 1.528
```

```
49.9338 - 48.4058
```

```
[1] 1.528
```

In other words, schools that have a SAT value that differ by 1%, have predicted graduation rates that differ by 1.53, on average.

2.7.1.2 Mathematical Explanation

To understand how we can directly compute this difference, consider the predicted values for two x-values that differ by one-percent, if we use symbolic notation:

$$\begin{aligned}\hat{y}_1 &= \hat{\beta}_0 + \hat{\beta}_1 \left[\ln(x)\right] \\ \hat{y}_2 &= \hat{\beta}_0 + \hat{\beta}_1 \left[\ln(1.01x)\right]\end{aligned}$$

The difference in their predicted values is:

$$\begin{aligned}\hat{y}_2 - \hat{y}_1 &= \hat{\beta}_0 + \hat{\beta}_1 \left[\ln(1.01x)\right] - \Big(\hat{\beta}_0 + \hat{\beta}_1 \left[\ln(x)\right]\Big) \\ &= \hat{\beta}_0 + \hat{\beta}_1 \left[\ln(1.01x)\right] - \hat{\beta}_0 - \hat{\beta}_1 \left[\ln(x)\right] \\ &= \hat{\beta}_1 \left[\ln(1.01x)\right] - \hat{\beta}_1 \left[\ln(x)\right] \\ &= \hat{\beta}_1 \left[\ln(1.01x) - \ln(x)\right] \\ &= \hat{\beta}_1 \left[\ln(\frac{1.01x}{1x})\right]\end{aligned}$$

If we substitute in any value for x, we can now directly compute this constant difference. Note that a convenient value for x is 1. Then this reduces to:

$$\hat{\beta}_1 \left[\ln(1.01)\right]$$

So now, we can interpret this as: a one-percent difference in x is associated with a $\hat{\beta}_1 \left[\ln(1.01)\right]$-unit difference in Y, on average.

In our model, we can compute this difference using the fitted coefficient $\hat{\beta}_1 = 153.6$ as

$$153.6 \left[\ln(1.01)\right] = 1.528371$$

The same computation using R is

```r
153.6 * log(1.01)
```

```
[1] 1.528
```

This gives you the constant difference exactly. So you can interpret the effect of SAT as, each 1% difference in SAT score is associated with a difference in graduation rates of 1.53, on average.

2.7.1.3 Approximate Interpretation

We can get an approximate estimate for the size of the effect by using the mathematical shortcut of

$$\text{Effect} \approx \frac{\hat{\beta}_1}{100}$$

Using our fitted results, we could approximate the size of the effect as,

$$\frac{153.6}{100} = 1.536$$

We could then interpret the effect of SAT by saying a 1% difference in median SAT score is associated with a 1.53-unit difference in predicted graduation rate, on average.

2.8 Including Covariates

We can also include covariates in the model. Below we examine the nonlinear effect of SAT on graduation controlling for differences in sector.

```
# Fit model
lm.4 = lm(grad ~ 1 + public + log(sat), data = mn)

# Model-level output
glance(lm.4)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1     0.866         0.857  6.34      96.6 8.47e-14     3  -106.  220.  226.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

The model explains 86.5% of the variation in graduation rates, $F(2,\ 30) = 96.58$, $p < .001$.

```
# Coefficient-level output
tidy(lm.4)
```

```
# A tibble: 3 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  -286.        28.0     -10.2  2.73e-11
2 public         -8.50       2.44     -3.48 1.56e- 3
3 log(sat)      146.        11.6      12.6  1.74e-13
```

Interpreting each of the coefficients using the raw SAT scores:

- The intercept value of -286.1 is the predicted average graduation rate for all public colleges/universities with a SAT value of 1 (extrapolation).
- There is a statistically significant effect of sector after controlling for differences in SAT score ($p = .002$). Public schools have a predicted graduation rate that is 8.5-units lower, on average, than private schools controlling for differences in median SAT scores.
- There is a statistically significant effect of SAT on graduation rates, controlling for differences in sector ($p < .001$). A 1% difference in median SAT value is associated with a 1.46-unit difference in predicted graduation rate, on average, after controlling for differences in sector.

2.8.1 Plot of the Model Results

To further help interpret these effects, we can plot the fitted model.

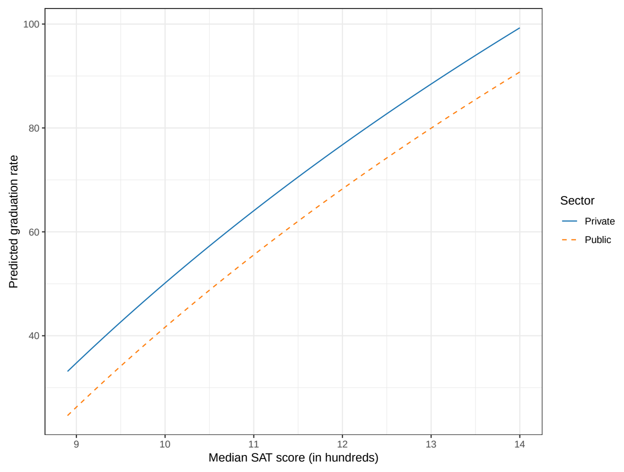


Figure 2.6: Predicted graduation rate as a function of median SAT score (in hundreds) and sector. The effect of SAT is log-linear.

```
# Set up data
plot_data = crossing(
  sat = seq(from = 8.9, to = 14.0, by = .1),
  public = c(0, 1)
  ) %>%
  mutate(
    yhat = predict(lm.4, newdata = .),
    public = factor(public, levels = c(0, 1), labels = c("Private", "Public"))
  )

#Examine data
head(plot_data)
```

```
# A tibble: 6 x 3
    sat public   yhat
  <dbl> <fct>   <dbl>
1   8.9 Private  33.1
2   8.9 Public   24.6
3   9   Private  34.8
4   9   Public   26.3
5   9.1 Private  36.4
6   9.1 Public   27.9
```

```
# Plot
ggplot(data = plot_data, aes(x = sat, y = yhat, color = public, linetype = public)) +
  geom_line() +
  theme_bw() +
  xlab("Median SAT score (in hundreds)") +
  ylab("Predicted graduation rate") +
  ggsci::scale_color_d3(name = "Sector") +
  scale_linetype_manual(name = "Sector", values = c("solid", "dashed"))
```

The plot shows the nonlinear, diminishing positive effect of SAT on graduation rate for both public and private schools. For schools with lower median SAT scores, there is a larger effect on graduation rates

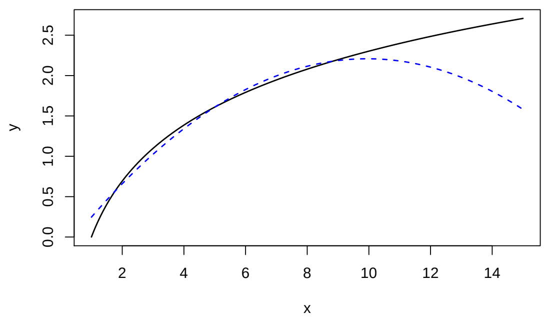


Figure 2.7: Comparison of quadratic (blue, dashed) and logarithmic (black, solid) functions of X.

than for schools with higher median SAT scores (for both private and public schools). The plot also shows the controlled effect of sector. For schools with the same median SAT score, private schools have a higher predicted graduation rate than public schools, on average.

2.9 Polynomial Effects vs. Log-Transformations

The inclusion of polynomial effects and the use of a log-transformation was to model the nonlinearity observed in the relationship between SAT scores and graduation rates. Both methods were successful in this endeavor. While either method could be used in practice to model nonlinearity, there are some considerations when making the choice of which may be more appropriate for a given modeling situation.

The first consideration is one of theory. The plot below shows the mathematical function for a log-transformed X-value (solid, black line) and for a quadratic polynomial of X (dashed, red line).

Both functions are nonlinear, however the polynomial function changes direction. For low values of X, the function has a large positive effect. This effect diminishes as X gets bigger, and around $X = 9$ the effect is zero. For larger values of X, the effect is actually negative. For the logarithmic function, the effect is always positive, but it diminishes as X gets larger. (Functions that constantly increase, or constantly decrease, are referred to as *monotonic functions.*) Theoretically, these are very different ideas, and if substantive literature suggests one or the other, you should probably acknowledge that in the underlying statistical model that is fitted.

Empirically, the two functions are very similar especially within certain ranges of X. For example, although the predictions from these models would be quite different for really high values of X, if we only had data from the range of 2 to 8 ($2 \leq X \leq 8$) both functions would produce similar residuals. In this case, the residuals would likely not suggest better fit for either of the two models. In this case, it might be prudent to think about Occam's Razor—if two competing models produce similar predictions, adopt the simpler model. Between these two functions, the **log-transformed model is simpler**; it has one predictor compared to the two predictors in the quadratic model. The mathematical models make this clear:

$$\begin{aligned}\textbf{Polynomial}: &\quad Y_i = \beta_0 + \beta_1(X_i) + \beta_2(X_i^2) + \epsilon_i \\ \textbf{Log-Transform}: &\quad Y_i = \beta_0 + \beta_1\bigg[\ln(X_i)\bigg] + \epsilon_i\end{aligned}$$

The quadratic polynomial model has two effects: a linear effect of X and a quadratic effect of X (remember it is an interaction model), while the model using the log-transformed predictor only has a single effect. If there is no theory to guide your model's functional form, and the residuals from the polynomial and log-transformed models seem to fit equally well, then the log-transformed model saves you a degree of freedom, and probably should be adopted.

Other Resources

In addition to the notes and what we cover in class, there are many other resources for learning about log-transformations. Here are some resources that may be helpful in that endeavor:

- Interpreting Coefficients in Regression with Log-Transformed Variables
- Interpret Regression Coefficient Estimates

Chapter 3

Nonlinearity: Log-Transforming the Outcome

In this set of notes, you will learn about log-transforming the outcome variable in a regression model to account for nonlinearity and heterogeneity of variance.

Preparation

Before class you will need to read the following:

- Osborne, Jason (2002). Notes on the use of data transformations. *Practical Assessment, Research & Evaluation, 8*(6).

3.1 Dataset and Research Question

The data we will use in this set of notes, *movies.csv* (see the data codebook here), includes attributes for $n = 1,806$ movies.

```
# Load libraries
library(broom)
library(dplyr)
library(ggplot2)
library(readr)
library(sm)
library(tidyr)

# Import data
movies = read_csv(file = "~/Documents/github/epsy-8252/data/movies.csv")
head(movies)
```

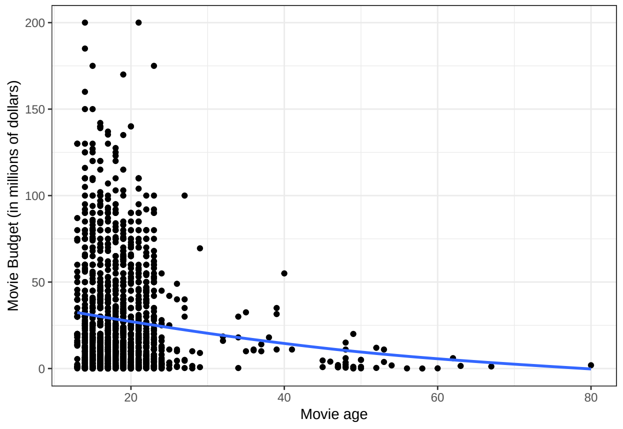


Figure 3.1: Scatterplot between age and budget. The loess smoother is also displayed.

```
# A tibble: 6 x 4
  title                        budget   age mpaa
  <chr>                         <dbl> <dbl> <chr>
1 'Til There Was You             23      21 PG-13
2 10 Things I Hate About You     16      19 PG-13
3 100 Mile Rule                   1.1    16 R
4 13 Going On 30                 37      14 PG-13
5 13th Warrior, The              85      19 R
6 15 Minutes                     42      17 R
```

Using these data, we will examine the relationship between age of a movie and budget.

3.2 Examine Relationship between Age and Budget

To being the analysis, we will examine the scatterplot between age and budget of our sample data.

```
ggplot(data = movies, aes(x = age, y = budget)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  theme_bw() +
  xlab("Movie age") +
  ylab("Movie Budget (in millions of dollars)")
```

The scatterplot suggests two potential problems with fitting a linear model to the data:

- The relationship is slightly curvilinear.
- The variation in budget for more recent movies is much greater than the variation in budget for older movies (heteroskedasticity).

We can see this much more clearly in the scatterplot of residuals versus fitted values from a fitted linear model.

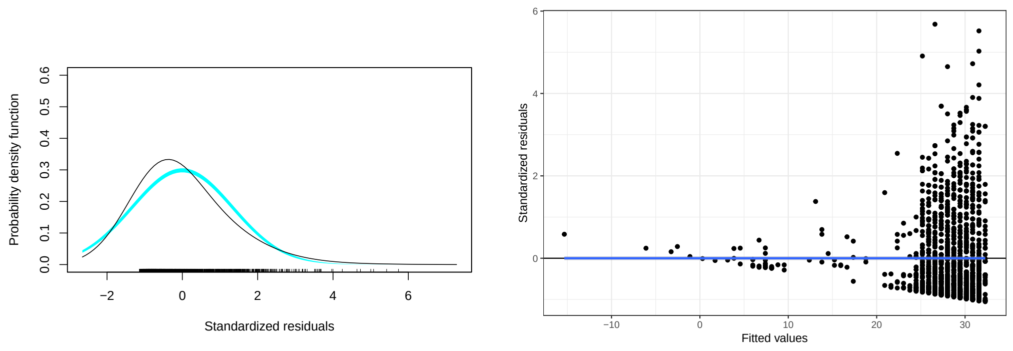


Figure 3.2: Residual plots from regressing budget on age.

```r
# Fit model
lm.1 = lm(budget ~ 1 + age, data = movies)

# Obtain residuals and fitted values
out_1 = augment(lm.1)

# Density plot of the residuals
sm.density(out_1$.std.resid, model = "normal", xlab = "Standardized residuals")

# Residuals versus fitted values
ggplot(data = out_1, aes(x = .fitted, y = .std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_smooth() +
  theme_bw() +
  xlab("Fitted values") +
  ylab("Standardized residuals")
```

These plots suggest violations of the normality assumption (the marginal distribution of the residuals is right-skewed) and of the assumption of homoskedasticity. Because of the large sample size, violation the linearity assumption is more difficult to see in this plot.

3.3 Transform the Outcome Using the Natural Logarithm (Base-e)

To alleviate problems of non-normality when the conditional distributions are right-skewed (or have high-end outliers) OR to alleviate heteroskedasticity, we can mathematically transform the outcome using a logarithm. Any base can be used for the logarithm, but we will transform the outcome using the natural logarithm because of the interpretive value.

First, we will create the log-transformed variable as a new column in the data, and then we will use the log-transformed budget (rather than raw budget) in any analyses.

```r
# Create log-transformed budget
movies = movies %>%
  mutate(
    Lbudget = log(budget)
  )

# Examine data
head(movies)
```

```
# A tibble: 6 x 5
  title                      budget   age mpaa  Lbudget
  <chr>                       <dbl> <dbl> <chr>   <dbl>
1 'Til There Was You           23      21 PG-13  3.14
2 10 Things I Hate About You   16      19 PG-13  2.77
3 100 Mile Rule                 1.1    16 R      0.0953
4 13 Going On 30               37      14 PG-13  3.61
5 13th Warrior, The            85      19 R      4.44
6 15 Minutes                   42      17 R      3.74
```

Recall that the logarithm is the inverse function of an exponent. As an example, consider the budget and log-transformed budget for *'Til There Was You*.

$$\begin{aligned}\ln(\text{Budget}) &= 3.135 \\ \ln(23.0) &= 3.135\end{aligned}$$

Or,

$$e^{3.135} = 23.0$$

Remember, the logarithm answers the mathematical question:

> e to what power is equal to 23.0?

3.4 Re-analyze using the Log-Transformed Budget

Now we will re-examine the scatterplot using the log-transformed outcome to see how this transformation affects the relationship.

```
# Scatterplot
ggplot(data = movies, aes(x = age, y = Lbudget)) +
  geom_point() +
  geom_smooth(se = FALSE) +
  theme_bw() +
  xlab("Movie age") +
  ylab("ln(Movie Budget)")
```

Log-transforming the outcome has drastically affected the scale for the outcome. Has this helped us better meet the assumptions? Again, we should examine the residual plots.

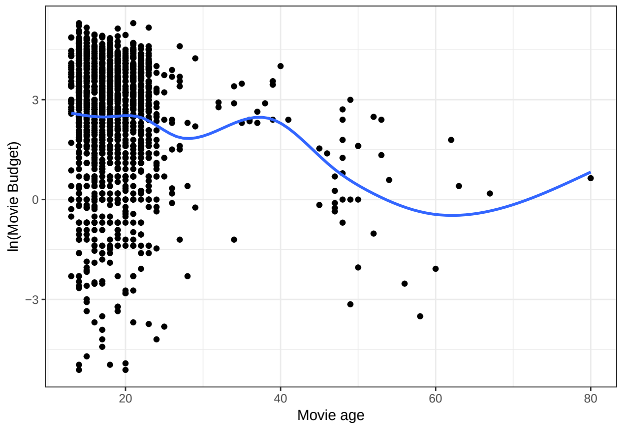


Figure 3.3: Scatterplot between age and log-transformed budget. The loess smoother is also displayed.

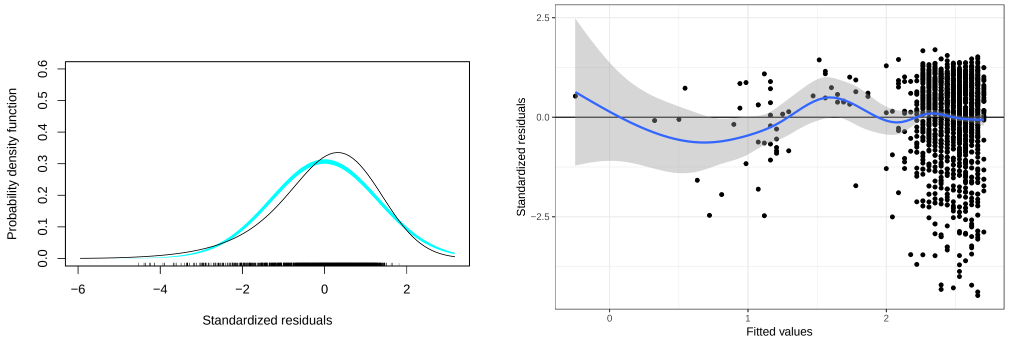


Figure 3.4: Residual plots from regressing the natural logarithm of budget on age.

```
# Fit model
lm.2 = lm(Lbudget ~ 1 + age, data = movies)

# Obtain residuals and fitted values
out_2 = augment(lm.2)

# Density plot of the residuals
sm.density(out_2$.std.resid, model = "normal", xlab = "Standardized residuals")

# Residuals versus fitted values
ggplot(data = out_2, aes(x = .fitted, y = .std.resid)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  geom_smooth() +
  theme_bw() +
  xlab("Fitted values") +
  ylab("Standardized residuals")
```

These plots still suggest violations of the normality assumption (the marginal distribution of the residuals is now left-skewed). The assumption of homoskedasticity also seems still violated, although much less. Most importantly, however, is the assumption of linearity now seems satisfied.

3.5 Interpreting the Regression Output

Let's examine the output from the model in which we regressed the log-transformed budget on age.

```
# Model-level output
glance(lm.2)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1    0.0213        0.0208  1.74      39.3 4.60e-10     2 -3560. 7125. 7142.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

The model-level summary information suggests that differences in movies' ages explains 2.1% of the variation in budget. (Remember, explaining variation in log-budget is the same as explaining variation in budget). Although this is a small amount of variation, it is statistically significant, $F(1, 1804) = 39.27$, $p < 0.001$.

```
# Coefficient-level output
tidy(lm.2)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)   3.28     0.139       23.6  1.69e-107
2 age          -0.0441   0.00703     -6.27 4.60e- 10
```

From the coefficient-level output the fitted equation is:

$$\ln\left(\hat{\text{Budget}_i}\right) = 3.28 - 0.04(\text{Age}_i)$$

With log-transformations, there are two possible interpretations we can offer. The first is to interpret the coefficients using the log-transformed values. These we interpret in the exact same way we do any other regression coefficients (except we use log-outcome instead of outcome):

- The intercept, $\hat{\beta}_0 = 3.28$, is the average predicted log-budget for movies made in 2019 (Age = 0).
- The slope, $\hat{\beta}_1 = -0.04$, indicates that each one-year difference in age is associated with a log-budget that differ by -0.04, on average.

3.5.1 Back-Transforming: A More Useful Interpretation

A second, probably more useful, interpretation is to back-transform log-budget to budget. To think about how to do this, we first consider a more general expression of the fitted linear model:

$$\ln\left(\hat{Y}_i\right) = \hat{\beta}_0 + \hat{\beta}_1(X_i)$$

The left-hand side of the equation is in the log-transformed metric, which drives our interpretations. If we want to instead, interpret using the raw metric of Y, we need to back-transform from $\ln(Y)$ to Y. To back-transform, we use the inverse function, which is to exponentiate using the base of the logarithm, in our case, base-e.

$$e^{\ln(Y_i)} = Y_i$$

If we exponentiate the left-hand side of the equation, to maintain the equality, we also need to exponentiate the right-hand side of the equation.

$$e^{\ln(Y_i)} = e^{\hat{\beta}_0 + \hat{\beta}_1(X_i)}$$

Then we use rules of exponents to simplify this.

$$Y_i = e^{\hat{\beta}_0} \times e^{\hat{\beta}_1(X_i)}$$

For our example, when we exponentiate both sides of the fitted equation:

$$\hat{\text{Budget}}_i = e^{3.28} \times e^{-0.04(\text{Age}_i)}$$

3.5.2 Substituting in Values for Age to Interpret Effects

To interpret the effects (which are now interpreted using budget—not log-budget) we can substitute in the different values for age and solve. For example when Age = 0:

$$\begin{aligned}\hat{\text{Budget}}_i &= e^{3.28} \times e^{-0.04(0)}\\ &= 26.58 \times 1\\ &= 26.58\end{aligned}$$

The predicted budget for a movie made in 2019 is 26.58 million dollars. How about a movie that was made in 2018 (a one-year difference)?

$$\begin{aligned}\hat{\text{Budget}}_i &= e^{3.28} \times e^{-0.04(1)}\\ &= 26.58 \times 0.96\\ &= 25.54\end{aligned}$$

The predicted budget for a movie made in 2019 is 25.54 million dollars. This is 0.96 TIMES the budget of a movie made in 2018.

Rather than using the language of **TIMES difference** you could also use the language of **fold difference**. In this case the slope coefficient would be interpreted as,

> Each one-year difference in age is associated with a 0.95-fold difference in budget, on average.

Simply put, when we back-transform from interpretations of log-Y to Y the interpretations are multiplicatively related to the intercept rather than additive. We can obtain these multiplicative values (and the back-transformed intercept) by using the `exp()` function to exponentiate the coefficients from the fitted model, which we obtain using the `coef()` function.

```r
exp(coef(lm.2))
```

```
(Intercept)         age
    26.5261      0.9569
```

3.5.3 Approximate Interpretation of the Slope

Remember that by using the natural logarithm we can interpret the effects as percent change. Rather than saying that a movie made in 2018 is predicted to have a budget that is 0.96 TIMES that of a movie made in 2019, we can directly interpret the slope as the percent change. Thus $\hat{\beta}_1 = -0.04$ can be interpreted as:

> Each one-year difference in age is associated with a four percent decrease in budget, on average.

If you want the specific mathematical change in budget, find $1 - e^{\hat{\beta}_1}$.

```
1 - exp(-0.04)
```

```
[1] 0.03921
```

If you use the language of percent decrease/increase, be very careful. *Percent change* and *percentage change* are sometimes interpreted differently! It is generally more clear to use the *X-fold difference* language.

3.6 Plotting the Fitted Model

As always, we can plot the fitted model to aid in interpretation. To do this we will create a sequence of ages, predict the log-budget using the fitted model, and then back-transform the log-budgets to raw budget.

```
# Set up data
plot_data = crossing(
    age = seq(from = 13, to = 80, by = 1)
    ) %>%
  mutate(
    # Predict
    yhat = predict(lm.2, newdata = .)
  )

# Examine data
head(plot_data)
```

```
# A tibble: 6 x 2
    age  yhat
  <dbl> <dbl>
1    13  2.71
2    14  2.66
3    15  2.62
4    16  2.57
5    17  2.53
6    18  2.48
```

```
# Back-transform the log-budgets
plot_data = plot_data %>%
  mutate(
    budget = exp(yhat)
  )

# Examine data
head(plot_data)
```

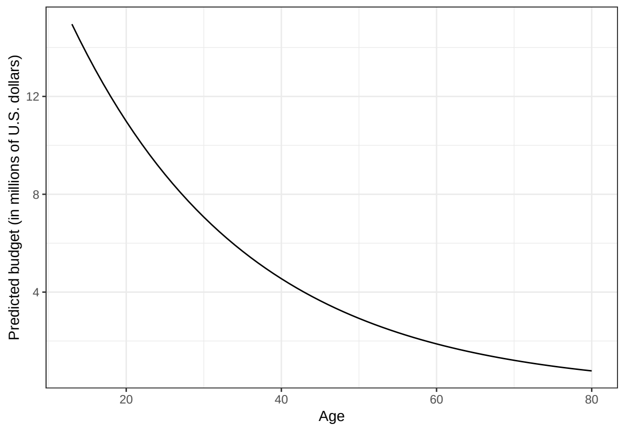


Figure 3.5: Plot of the predicted movie budget as a function of its age. The non-linearity in the plot indicates that there is a diminishing negative effect of age on budget.

```
# A tibble: 6 x 3
    age  yhat budget
  <dbl> <dbl>  <dbl>
1    13  2.71   15.0
2    14  2.66   14.3
3    15  2.62   13.7
4    16  2.57   13.1
5    17  2.53   12.5
6    18  2.48   12.0
```

```
# Plot
ggplot(data = plot_data, aes(x = age, y = budget)) +
    geom_line() +
    theme_bw() +
  xlab("Age") +
  ylab("Predicted budget (in millions of U.S. dollars)")
```

Based on this plot, we see the non-linear, negative effect of age on budget. In other words, older movies tend to have a smaller budget, on average, but this decrease is not constant. This pattern of non-linear decline is referred to as *exponential decay*.

Although this function has a different look than the function we saw in the previous unit (it is negative rather than positive), it is also a *monotonic* function (no change in direction).

3.7 Relationship between MPAA Rating and Budget

We also may want to control for differences in MPAA rating. Before we fit the multiple regression model, however, we will first explore whether MPAA rating is a useful covariate by seeing whether there are differences in budget between PG, PG-13m and R rated movies. Since we log-transformed budget (the outcome) in the previous analysis we will need to use the log-transformed outcome in this exploration as well.

```
# Plot the observed data
ggplot(data = movies, aes(x = mpaa, y = Lbudget)) +
  geom_jitter(alpha = 0.2) +
```

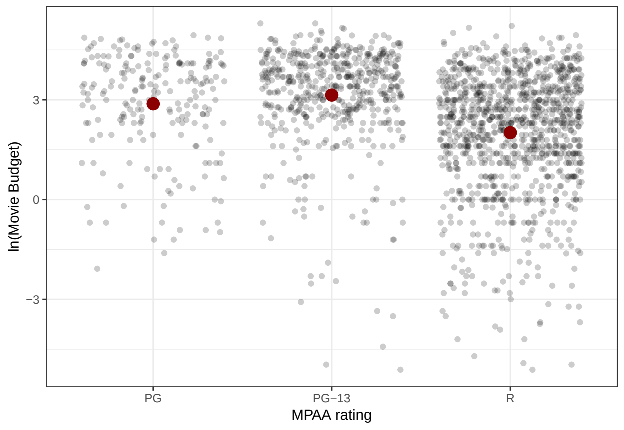


Figure 3.6: Jittered scatterplot of log-budget versus MPAA rating.

```r
  stat_summary(fun.y = 'mean', geom = "point", size = 4, color = "darkred") +
    theme_bw() +
    xlab("MPAA rating") +
    ylab("ln(Movie Budget)")
```

```r
# Compute summary statistics
movies %>%
  group_by(mpaa) %>%
  summarize(
    M = mean(Lbudget),
    SD = sd(Lbudget)
    )
```

```
# A tibble: 3 x 3
  mpaa      M    SD
  <chr> <dbl> <dbl>
1 PG     2.88  1.53
2 PG-13  3.14  1.51
3 R      2.01  1.78
```

The scatterplot and summary statistics indicate there are sample differences in the mean log-budgets for the three MPAA ratings. The variation in log-budgets seems roughly the same for the three ratings.

3.7.1 Regression Model

Let's regress log-transformed budget on MPAA rating and examine the output from the model. To do so, we will need to first create three dummy variables for the different ratings.

```r
# Create dummy variables
movies = movies %>%
  mutate(
    pg   = if_else(mpaa == "PG", 1, 0),
    pg13 = if_else(mpaa == "PG-13", 1, 0),
    r    = if_else(mpaa == "R", 1, 0)
  )

# Fit the model (pg is reference group)
lm.3 = lm(Lbudget ~ 1 + pg13 + r, data = movies)

# Model-level output
glance(lm.3)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1    0.0892        0.0882  1.68      88.3 2.66e-37     3 -3495. 6997. 7019.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

The model-level summary information suggests that differences in MPAA rating explains 8.9% of the variation in budget. (Remember, explaining variation in log-budget is the same as explaining variation in budget). Although this is a small amount of variation, it is statistically significant, $F(2, 1803) = 88.28$, $p < 0.001$.

```
# Coefficient-level output
tidy(lm.3)
```

```
# A tibble: 3 x 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)    2.88      0.115     25.0  9.98e-119
2 pg13           0.262     0.136      1.92 5.45e-  2
3 r             -0.867     0.126     -6.88 8.49e- 12
```

From the coefficient-level output we see that the fitted equation is:

$$\ln\left(\hat{\text{Budget}}_i\right) = 2.88 + 0.26(\text{PG-13}_i) - 0.87(\text{R}_i)$$

With log-transformations, there are two possible interpretations we can offer. The first is to interpret the coefficients using the log-transformed values. These we interpret in the exact same way we do any other regression coefficients (except we use log-outcome instead of outcome):

- The intercept, $\hat{\beta}_0 = 2.88$, is the average predicted log-budget for PG rated movies.
- The slope associated with PG-13, $\hat{\beta}_1 = 0.26$, indicates that PG-13 rated movies have a log-budget that is 0.26 higher than PG rated movies, on average.
- The slope associated with R, $\hat{\beta}_2 = -0.87$, indicates that R rated movies have a log-budget that is 0.87 lower than PG rated movies, on average.

We can also interpret these by back-transforming to raw budget. To do that we exponentiate the coefficients.

```
exp(coef(lm.3))
```

```
(Intercept)        pg13           r
    17.8134      1.2998      0.4201
```

- PG rated movies have a budget of 17.81 million dollars, on average.
- PG-13 rated movies have a budget that is 1.30 TIMES the estimated budget for PG rated movies, on average.
- R rated movies have a budget that is 0.42 TIMES the estimated budget for PG rated movies, on average.

3.7.2 Mathematical Explanation

Remember, if we want to interpret using the raw metric of Y, we need to back-transform from $\ln(Y)$ to Y. To back-transform, we use the inverse function, which is to exponentiate using the base of the logarithm, in our case, base-e. For our example, when we exponentiate both sides of the fitted equation:

$$
\begin{aligned}
e^{\ln\left(\hat{\text{Budget}}_i\right)} &= e^{2.88+0.26(\text{PG-13}_i)-0.87(\text{R}_i)} \\
\hat{\text{Budget}}_i &= e^{2.88} \times e^{0.26(\text{PG-13}_i)} \times e^{-0.87(\text{R}_i)}
\end{aligned}
$$

To interpret the effects (which are now interpreted using budget—not log-budget) we can substitute in the different dummy variable patterns and solve.

$$
\begin{aligned}
\textbf{PG Movie:}\ \ \hat{\text{Budget}}_i &= e^{2.88} \times e^{0.26(0)} \times e^{-0.87(0)} \\
&= 17.81 \times 1 \times 1 \\
&= 17.81
\end{aligned}
$$

$$
\begin{aligned}
\textbf{PG-13 Movie:}\ \ \hat{\text{Budget}}_i &= e^{2.88} \times e^{0.26(1)} \times e^{-0.87(0)} \\
&= 17.81 \times 1.30 \times 1 \\
&= 23.15
\end{aligned}
$$

$$
\begin{aligned}
\textbf{R Movie:}\ \ \hat{\text{Budget}}_i &= e^{2.88} \times e^{0.26(0)} \times e^{-0.87(1)} \\
&= 17.81 \times 1 \times 0.42 \\
&= 7.48
\end{aligned}
$$

3.7.3 Approximate Interpretations

Unfortunately, the approximate interpretations of the slopes by directly interpreting the coefficients using the language of percent change are not completely trustworthy. If we did interpret them, the interpretations for the two slopes would be:

- PG-13 rated movies have budget that is 26.2 percent higher than PG rated movies, on average.
- R rated movies have budget that is 87 percent lower than PG rated movies, on average.

This interpretation is roughly true for the PG-13 effect, but not for the R effect. This approximate interpretation starts to become untrustworthy when the slope value is higher than about 0.20 or so.

3.8 Multiple Regression: Main Effects Model

Now we can fit a model that includes our focal predictor of age and our covariate of MPAA rating.

```r
# Fit model (PG is reference group)
lm.4 = lm(Lbudget ~ 1 + age + pg13 + r, data = movies)

# Model-level output
glance(lm.4)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1     0.109         0.108  1.66      73.8 4.82e-45     4 -3474. 6959. 6986.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

The model-level summary information suggests that differences in age and MPAA rating of a movie explains 11.0% of the variation in budget. (Remember, explaining variation in log-budget is the same as explaining variation in budget); $F(3, 1802) = 73.84$, $p < 0.001$.

```
# Coefficient-level output
tidy(lm.4)
```

```
# A tibble: 4 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   3.73     0.175       21.3  3.76e-90
2 age          -0.0431   0.00673     -6.41 1.89e-10
3 pg13          0.208    0.135        1.54 1.23e- 1
4 r            -0.904    0.125       -7.24 6.79e-13
```

The coefficient-level output suggest that there is still a statistically significant effect of age on budget, after controlling for differences in MPAA rating; $t(1802) = -6.41$, $p < .001$.

To determine if there is an effect of MPAA rating, after accounting for differences in age, at least one of the effects of MPAA rating need to be statistically significant. Here we see that the coefficient associated with rated R movies is statistically significant.

Remember that when we have more than two categories (more than one dummy variable) there can be many ways for the effect to play out, and not all of these are represented in the model we fitted. One way we can simultaneously examine ALL the ways this effect can play out is to use a nested F-test.

3.8.1 Nested F-Test

If we want to examine if there is a controlled effect of MPAA rating (controlling for age), we want to see whether by including MPAA rating in a model THAT ALREADY INCLUDES age we explain additional variation in the outcome. To do this we can compare a model that only includes the effect of age to a model that includes both the effects of age and MPAA rating. If the latter model explains a statistically significant amount of additional variation we can say that there is an effect of MPAA rating after controlling for differences in age.

In statistical hypothesis testing we are examining the following null hypothesis:

$$H_0 : \rho^2_{\text{Age,MPAA rating}} - \rho^2_{\text{Age}} = 0$$

If we fail to reject this hypothesis, then the two models explain the SAME amount of variation and we should adopt the simpler model; MPAA rating does not explain additional variation in budget. If we reject this hypothesis, MPAA rating does explain additional variation in budget, above and beyond age; and we should adopt the model that includes both effects.

To test this hypothesis we fit both models and then give both models to the `anova()` function.

```r
# Fit models
lm.2 = lm(Lbudget ~ 1 + age,            data = movies)
lm.4 = lm(Lbudget ~ 1 + age + pg13 + r, data = movies)

# Nested F-test
anova(lm.2, lm.4)
```

```
Analysis of Variance Table

Model 1: Lbudget ~ 1 + age
Model 2: Lbudget ~ 1 + age + pg13 + r
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1   1804 5448
2   1802 4957  2       491 89.2 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test suggests that there is a statistically significant effect of MPAA rating even after accounting for differences in age; $F(2, 1802) = 89.21$, $p < .001$.

3.8.2 Coefficient-Level Interpretation

To interpret the coefficients, we will again exponentiate the fitted coefficients so we can interpret them using the raw-metric of budget.

```r
exp(coef(lm.4))
```

```
(Intercept)         age        pg13           r
    41.7120      0.9578      1.2318      0.4051
```

- The model estimated budget for a PG movie (reference group) that was made in 2019 (age = 0) is 41.71 million dollars.
- Each one-year difference in age is associated with a 0.96-fold difference (4.3% decrease) in budget, on average, controlling for differences in MPAA rating.
- PG-13 rated movies have a budget that is 1.23 times that for PG movies, on average, controlling for differences in age.
- R rated movies have a budget that is 0.41 times that for PG movies, on average, controlling for differences in age.

3.8.3 Plot of the Fitted Model

To plot the fitted model that includes a categorical predictor with more than two levels, it is best to re-fit the `lm()` using the categorical variable.

```r
# Re-fit the model
lm.5 = lm(Lbudget ~ 1 + age + mpaa, data = movies)

# Model-level output
glance(lm.5)
```

```
# A tibble: 1 x 11
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <int>  <dbl> <dbl> <dbl>
1     0.109         0.108  1.66      73.8 4.82e-45     4 -3474. 6959. 6986.
# ... with 2 more variables: deviance <dbl>, df.residual <int>
```

```
# Coefficient-level output
tidy(lm.5)
```

```
# A tibble: 4 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   3.73     0.175       21.3  3.76e-90
2 age          -0.0431   0.00673     -6.41 1.89e-10
3 mpaaPG-13     0.208    0.135        1.54 1.23e- 1
4 mpaaR        -0.904    0.125       -7.24 6.79e-13
```

Note this is the exact same model we fitted using the dummy variables, but R will choose the reference group for us (alphabetically). We can now set up our plotting data, predict, back-transform the outcome, and plot.

```
# Set up data
plot_data = crossing(
    age = seq(from = 13, to = 80, by = 1),
    mpaa = c("PG", "PG-13", "R")
    ) %>%
  mutate(
    yhat = predict(lm.5, newdata = .),
    budget = exp(yhat)
  )

# Examine data
head(plot_data)
```

```
# A tibble: 6 x 4
    age mpaa   yhat budget
  <dbl> <chr> <dbl>  <dbl>
1    13 PG     3.17  23.8
2    13 PG-13  3.38  29.3
3    13 R      2.27   9.65
4    14 PG     3.13  22.8
5    14 PG-13  3.34  28.1
6    14 R      2.22   9.24
```

```
# Plot
ggplot(data = plot_data, aes(x = age, y = budget, color = mpaa, linetype = mpaa)) +
    geom_line() +
    theme_bw() +
  xlab("Age") +
  ylab("Predicted budget (in millions of U.S. dollars)") +
  ggsci::scale_color_d3(name = "MPAA rating") +
  scale_linetype_manual(name = "MPAA rating", values = 1:3)
```

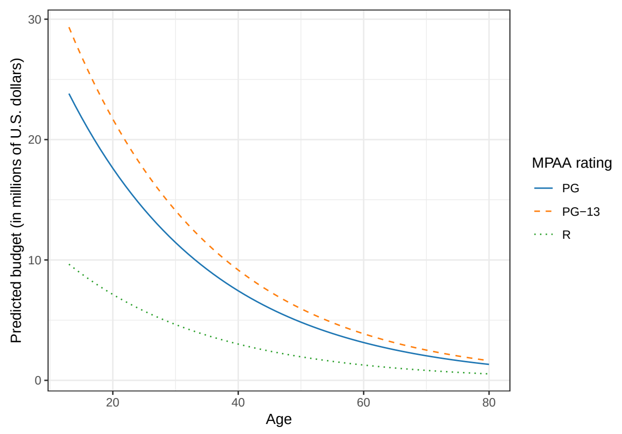


Figure 3.7: Plot of the predicted movie budget as a function of its age and MPAA rating. The non-linearity in the plot indicates that there is a diminishing negative effect of age on budget.

The plot displays the negative, nonlinear effect of age on budget for all three types of movies (main effect of age). It also shows that PG-13 rated movies have a higher predicted budget than PG and R rated movies, and that PG rated movies have a higher predicted budget than R rated movies at EVERY age. This is the main effect of MPAA rating.

Notice that in the plot, the three lines are not parallel. This is a mathematical artifact of back-transforming log-budget to raw budget. It does not indicate that an interaction model was fitted. How non-parallel the lines are depends on the size of the coefficients associated with the MPAA effects (in this example). This is why, especially with transformed data, it is essential to plot the model to make sure you are understanding the interpretations from your coefficients.

3.9 Multiple Regression: Interaction Model

To study whether there is an interaction effect between MPAA rating and age, we will fit the interaction model and compare it to the main-effects model using the nested F-test.

```r
# Fit the models
lm.5 = lm(Lbudget ~ 1 + age + mpaa,            data = movies)
lm.6 = lm(Lbudget ~ 1 + age + mpaa + age:mpaa, data = movies)

# Nested F-test
anova(lm.5, lm.6)
```

```
Analysis of Variance Table

Model 1: Lbudget ~ 1 + age + mpaa
Model 2: Lbudget ~ 1 + age + mpaa + age:mpaa
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1   1802 4957
2   1800 4949  2      7.76 1.41   0.24
```

The test suggests that we should adopt the main-effects model. The interaction-effect was not statistically significant; $F(2, 1800) = 1.41$, $p = .244$.

If the model that included the interaction effect was adopted, it would suggest that: (1) the effect of age on budget depends on MPAA rating, or (2) the effect of MPAA rating on budget depends on age of the movie. To further interpret these effects, you should plot the results of the fitted interaction model.

Log Transformations: Some Final Thoughts

There are four general curvilinear, monotonic functions (shown below).

In the previous two units you learned how to transform the data to "linearize" two of the four monotonic functions:

- For positive, decelerating functions we log-transformed X; and
- For positive, accelerating functions we log-transformed Y.

To better understand how we can use transformations to straighten-out relationships, we will examine a set of transformations known as **power transformations**.

Power Transformations

Consider a set of powers, p that can be used in the exponent of a variable X (the variable is irrelevant; it could also be Y) so that a transformation of the variable X is:

$$\text{Transformed Variable} = X^p$$

Consider the following values for p:

$$p = \{-3, -2, -1, 0, 1, 2, 3\}$$

These all represent particular transformations of X. Note that when $p = 1$, the variable X is left untransformed ($X^1 = X$). Consider the following transformations:

$$\begin{array}{ll} X^3 & \\ X^2 & \\ X^1 & \text{Untransformed} \end{array}$$

These are shown in the figure below.

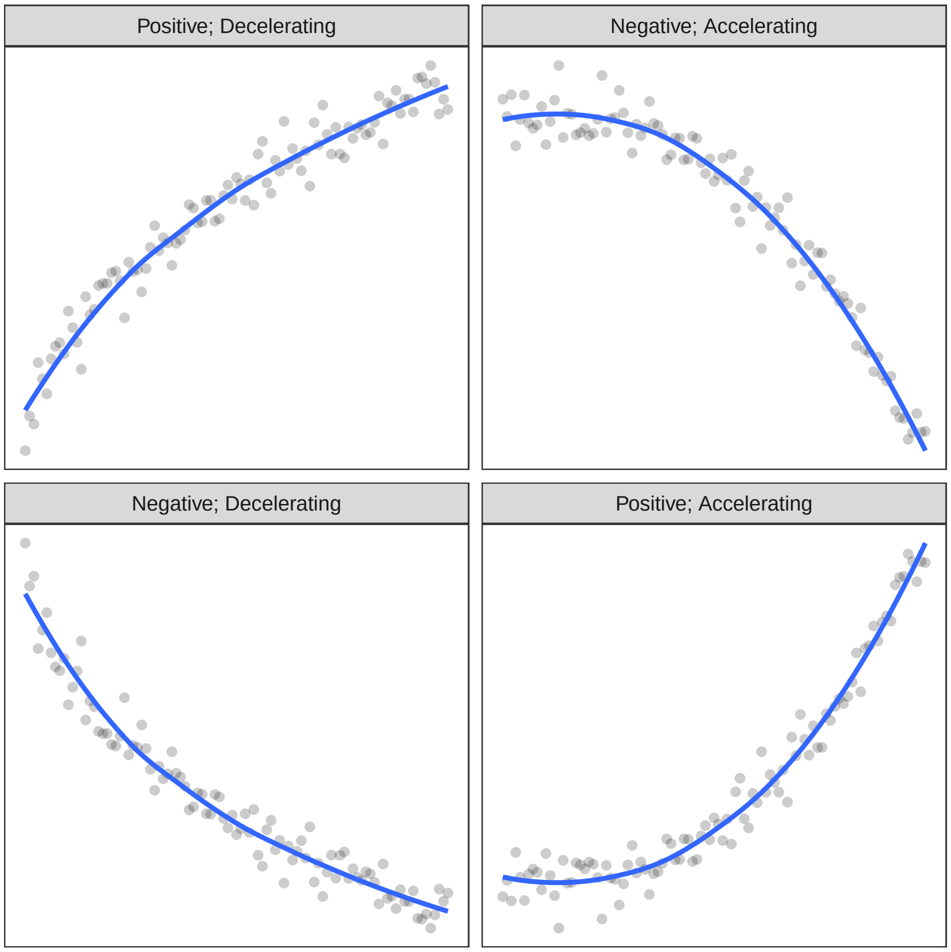


Figure 3.8: Four general monotonic, curvilinear shapes.