IS5 in R: Regression Wisdom (Chapter 8)

Margaret Chien and Nicholas Horton (nhorton@amherst.edu)

June 16, 2018

Introduction and background

This document is intended to help describe how to undertake analyses introduced as examples in the Fifth Edition of *Intro Stats* (2018) by De Veaux, Velleman, and Bock. More information about the book can be found at http://wps.aw.com/aw_deveaux_stats_series. This file as well as the associated R Markdown reproducible analysis source file used to create it can be found at http://nhorton.people.amherst.edu/is5.

This work leverages initiatives undertaken by Project MOSAIC (http://www.mosaic-web.org), an NSF-funded effort to improve the teaching of statistics, calculus, science and computing in the undergraduate curriculum. In particular, we utilize the `mosaic` package, which was written to simplify the use of R for introductory statistics courses. A short summary of the R needed to teach introductory statistics can be found in the mosaic package vignettes (http://cran.r-project.org/web/packages/mosaic). A paper describing the mosaic approach was published in the *R Journal*: https://journal.r-project.org/archive/2017/RJ-2017-024.

Chapter 8: Regression Wisdom

```
library(mosaic)
library(readr)
library(janitor)
```

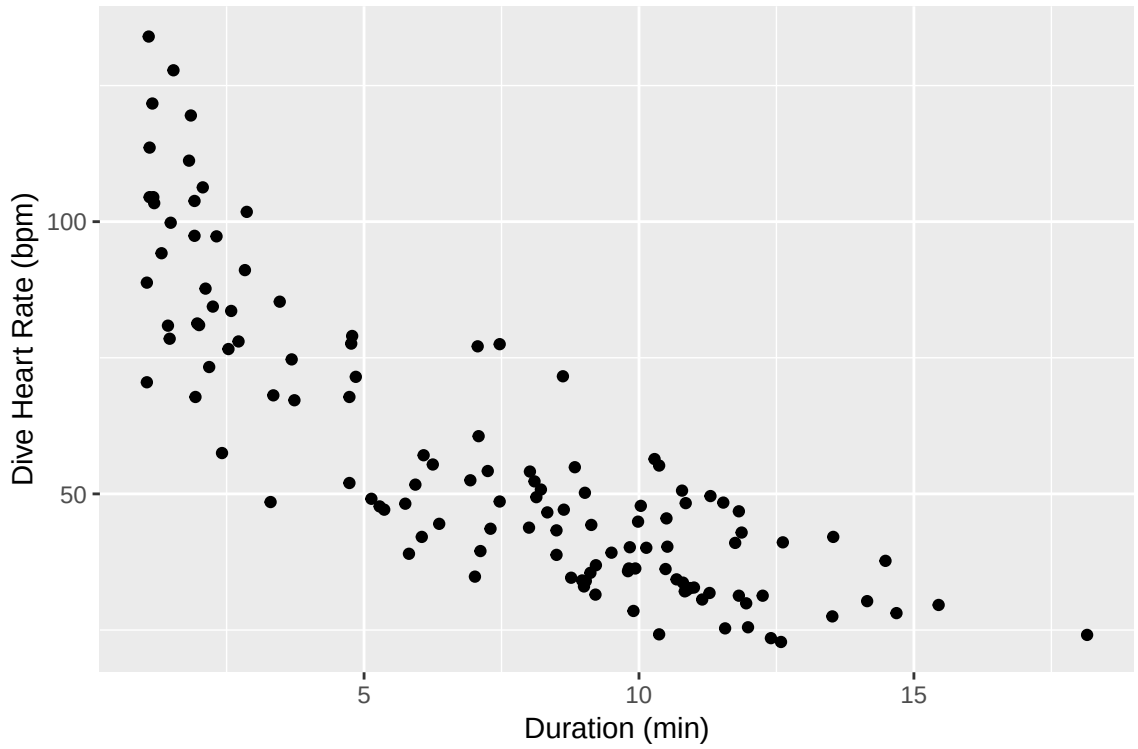
Section 8.1: Examining Residuals

Getting the "Bends": When the Residuals Aren't Straight

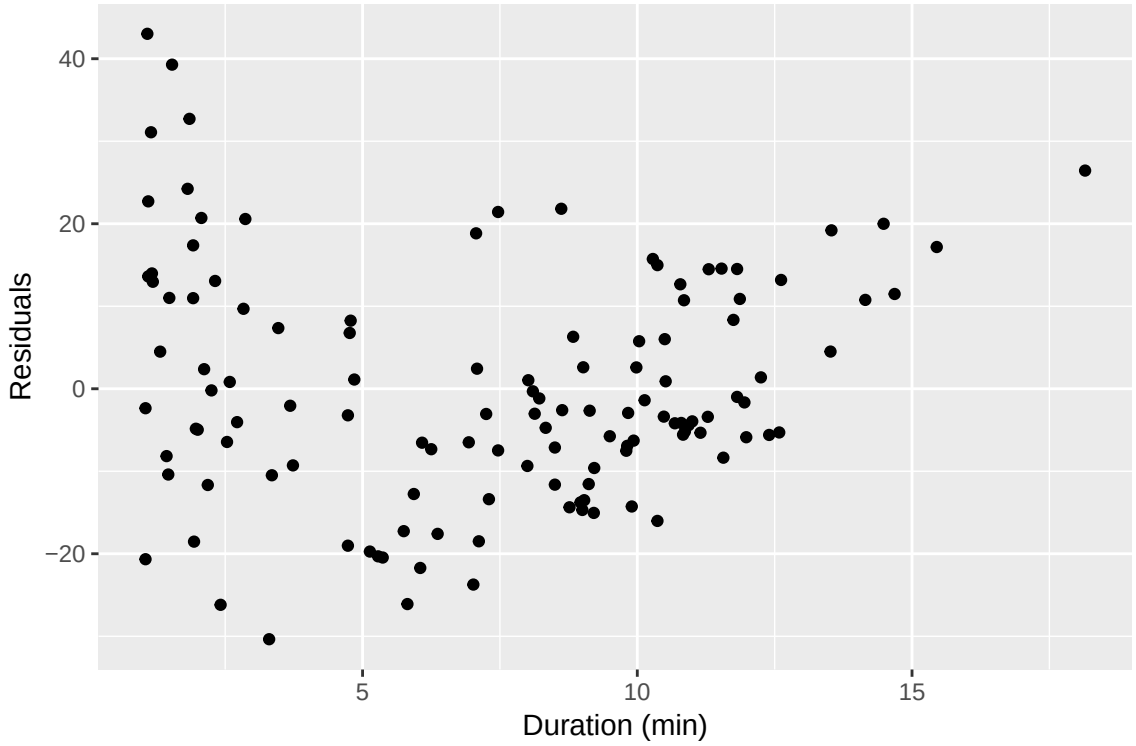
```
Penguins <- read_csv("http://nhorton.people.amherst.edu/is5/data/Penguins.csv") %>%
  clean_names()
```

```
## Parsed with column specification:
## cols(
##   `Dive Heart Rate` = col_double(),
##   `Depth(m)` = col_double(),
##   `Duration(min)` = col_double(),
##   `Bird#` = col_character()
## )
```

```
gf_point(dive_heart_rate ~ duration_min, data = Penguins, xlab = "Duration (min)", ylab = "Dive Heart Ra
```




```
penguinlm <- lm(dive_heart_rate ~ duration_min, data = Penguins)
gf_point(resid(penguinlm) ~ duration_min, data = Penguins, xlab = "Duration (min)", ylab = "Residuals")
```



By default, `read_csv()` prints the variable names. These messages can be suppressed using the `message = FALSE` code chunk option to save space and improve readability.

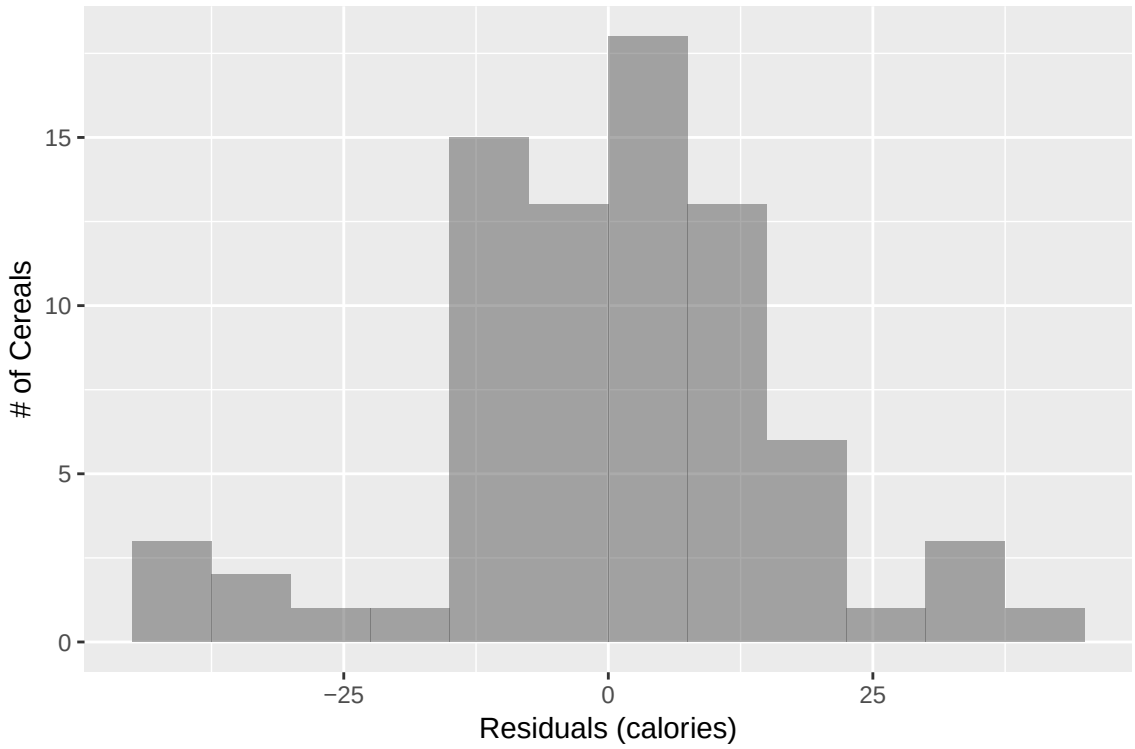
Here we use the `clean_names()` function from the `janitor` package to sanitize the names of the columns (which would otherwise contain special characters or whitespace).

Sifting Residuals for Groups

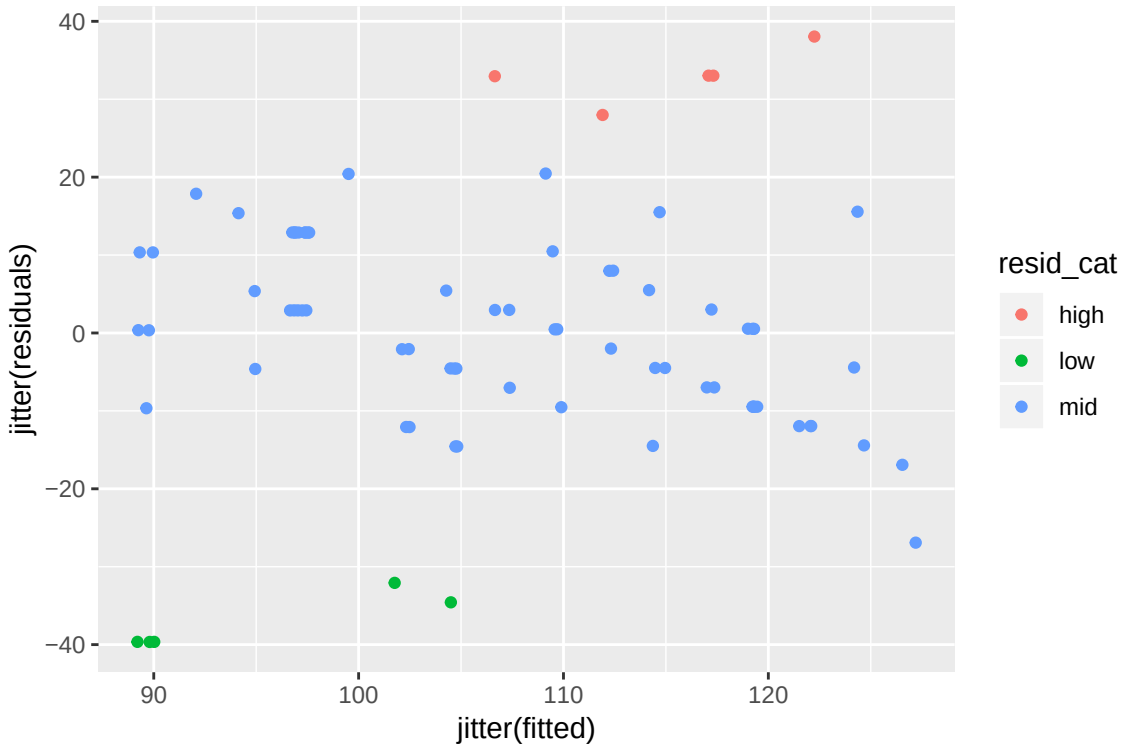
```
Cereal <- read_csv("http://nhorton.people.amherst.edu/is5/data/Cereals.csv")
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   mfr = col_character(),
##   calories = col_integer(),
##   sugars = col_integer(),
##   carbo = col_double(),
##   protein = col_integer(),
##   fat = col_integer(),
##   sodium = col_integer(),
##   fiber = col_double(),
##   potass = col_integer(),
##   shelf = col_integer(),
##   Middle = col_character(),
##   shelf_1 = col_integer(),
##   shelf_2 = col_integer(),
##   shelf_3 = col_integer()
## )
```

```
cereallm <- lm(calories ~ sugars, data = Cereal)
# Figure 8.3, page 235
gf_histogram(~ resid(cereallm), binwidth = 7.5, center = 7.5/2) %>%
  gf_labs(x = "Residuals (calories)", y = "# of Cereals")
```




```
Cereal <- Cereal %>%
  mutate(residuals = resid(cereallm),
         fitted = fitted(cereallm)) %>%
  mutate(resid_cat = ifelse(residuals >= -30 & residuals <= 25, "mid", ifelse(residuals > 25, "high", "
# Figure 8.4
gf_point(jitter(residuals) ~ jitter(fitted), color = ~ resid_cat, data = Cereal)
```



Jitter adds some random noise to allow easier observation of values that are shared by more than one cereal.

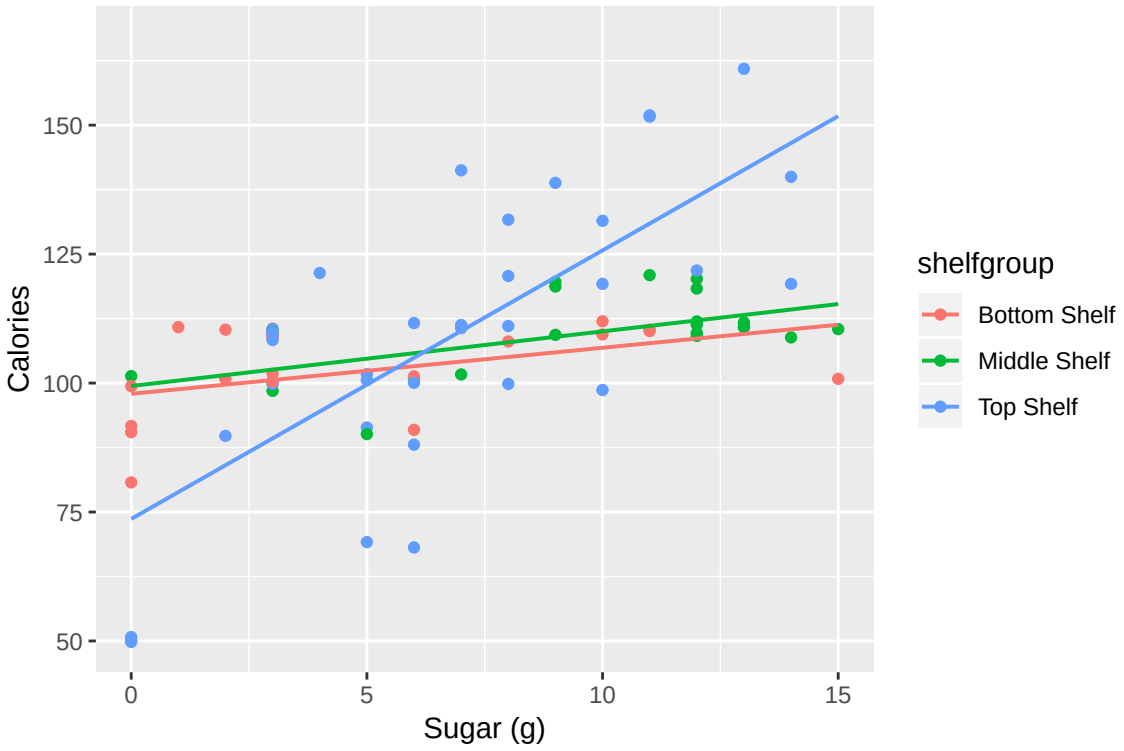
```
tally(~ shelf, data = Cereal)
```

```
## shelf
##  1  2  3
## 20 21 36
```

```
Cereal <- mutate(Cereal, shelfgroup =
                   recode(shelf,
                          `1` = "Bottom Shelf",
                          `2` = "Middle Shelf",
                          `3` = "Top Shelf")
                   )
tally(~ shelfgroup, data = Cereal)
```

```
## shelfgroup
## Bottom Shelf Middle Shelf    Top Shelf
##           20           21           36
```

```
# Figure 8.5
gf_point(jitter(calories) ~ sugars, color = ~ shelfgroup, data = Cereal) %>%
  gf_lm() %>%
  gf_labs(x = "Sugar (g)", y = "Calories")
```



The `recode()` function allows for efficient mutation of levels.

Section 8.2: Extrapolation: Reaching Beyond the Data

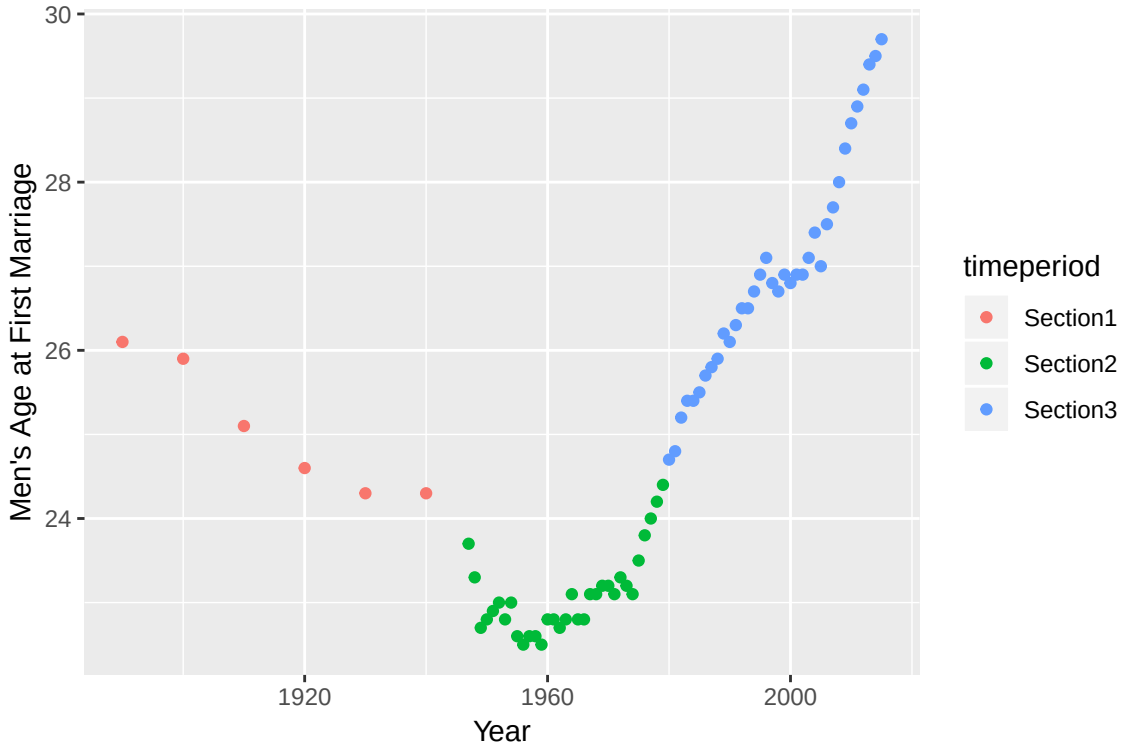
See displays on page 237 and 238.

Example 8.1: Extrapolation: Reaching Beyond the Data

```
MarriageAge <- read_csv("http://nhorton.people.amherst.edu/is5/data/Marriage_age_2015.csv")
```

```
## Parsed with column specification:
## cols(
##   Year = col_integer(),
##   Men = col_double(),
##   Women = col_double()
## )
```

```
MarriageAge <- MarriageAge %>%
  mutate(timeperiod = ifelse(Year <= 1940, "Section1", ifelse(Year >= 1980, "Section3", "Section2")))
gf_point(Men ~ Year, color = ~ timeperiod, data = MarriageAge, ylab = "Men's Age at First Marriage")
```



`ifelse()` works similarly to `recode()`.

Section 8.3: Outliers, Leverage, and Influence

```
Election2000 <- read_csv("http://nhorton.people.amherst.edu/is5/data/Election_2000.csv")
```

```
## Parsed with column specification:
## cols(
##   County = col_character(),
##   Gore = col_integer(),
##   Bush = col_integer(),
##   Buchanan = col_integer(),
##   Nader = col_integer()
## )
```

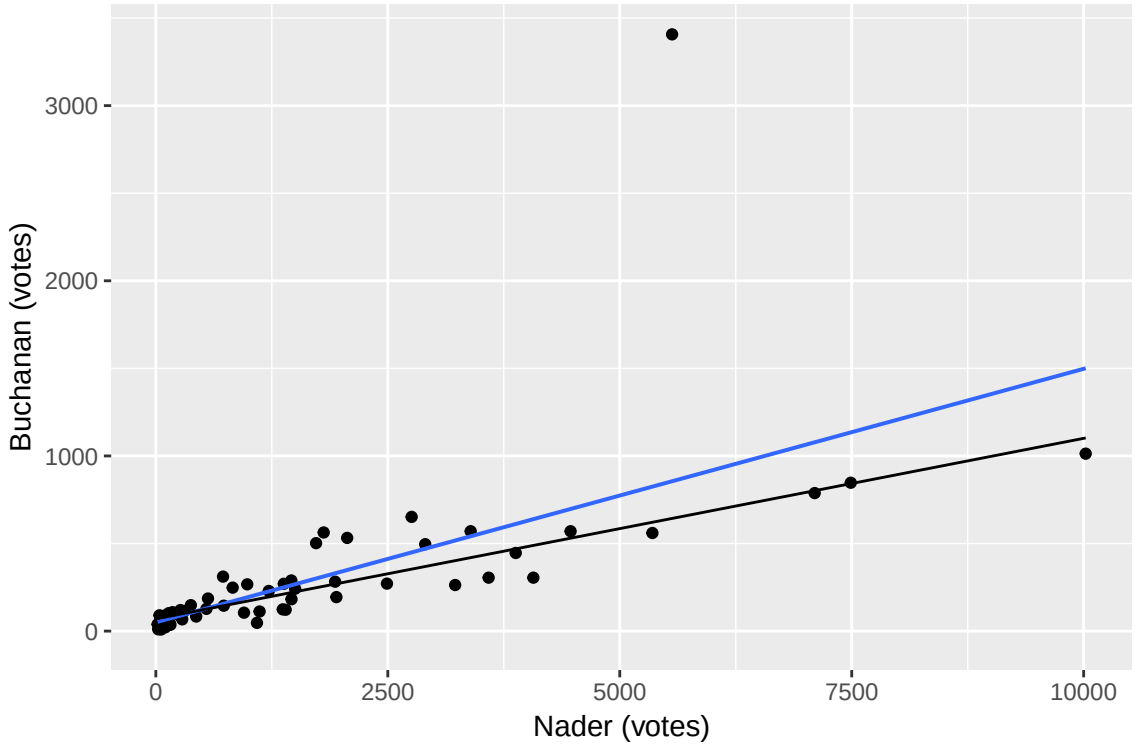
```
withlm <- lm(Buchanan ~ Nader, data = Election2000)
msummary(withlm)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 50.25627   51.63965   0.973    0.334
## Nader        0.14472    0.02076   6.971 1.95e-09 ***
##
## Residual standard error: 343 on 65 degrees of freedom
## Multiple R-squared:  0.4278, Adjusted R-squared:  0.419
## F-statistic: 48.59 on 1 and 65 DF,  p-value: 1.954e-09
```

```
withoutlm <- lm(Buchanan ~ Nader, data = filter(Election2000, Buchanan <= 3000))
msummary(withoutlm)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 69.52787   14.51176   4.791 1.02e-05 ***
## Nader        0.10309    0.00602  17.126  < 2e-16 ***
##
## Residual standard error: 96.27 on 64 degrees of freedom
## Multiple R-squared:  0.8209, Adjusted R-squared:  0.8181
## F-statistic: 293.3 on 1 and 64 DF,  p-value: < 2.2e-16
```

```
# Figure 8.10, page 241
gf_point(Buchanan ~ Nader, data = Election2000) %>%
  gf_lm() %>%
  gf_labs(x = "Nader (votes)", y = "Buchanan (votes)") %>%
  gf_fun(withoutlm)
```



See page 242 for example of high-leverage point.

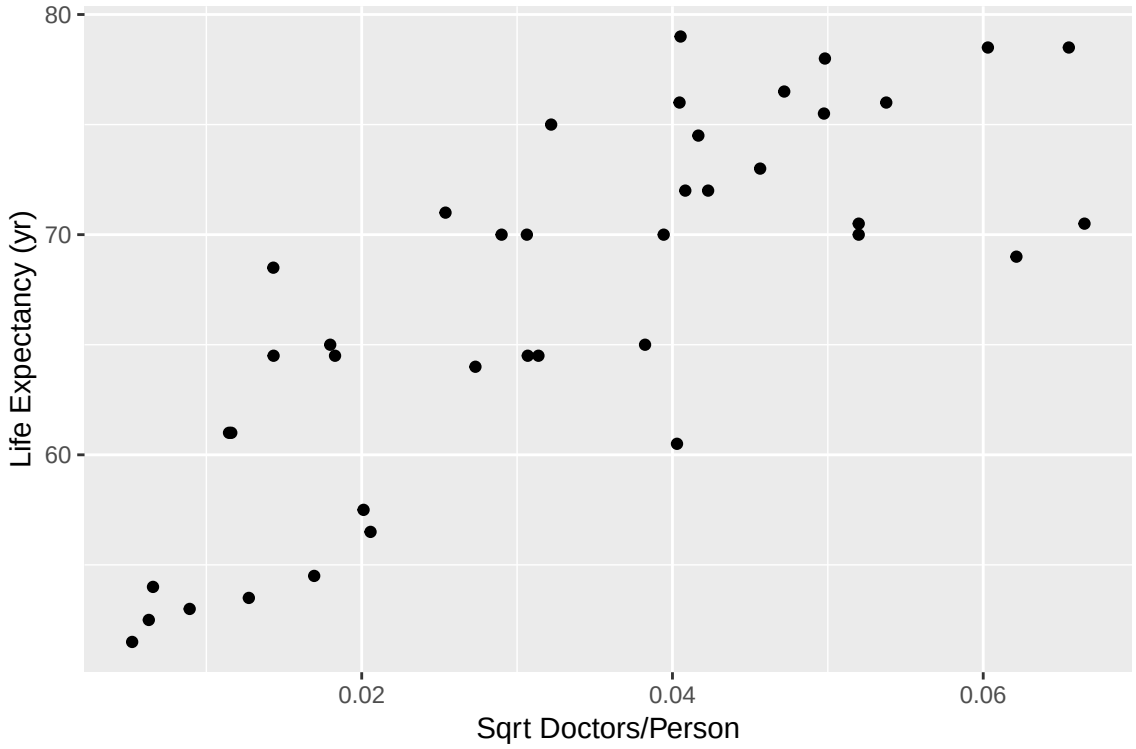
Section 8.4: Lurking Variables with Causation

```
Doctors <- read_csv("http://nhorton.people.amherst.edu/is5/data/Doctors_and_life_expectancy.csv") %>%
  clean_names()
```

```
## Parsed with column specification:
## cols(
```

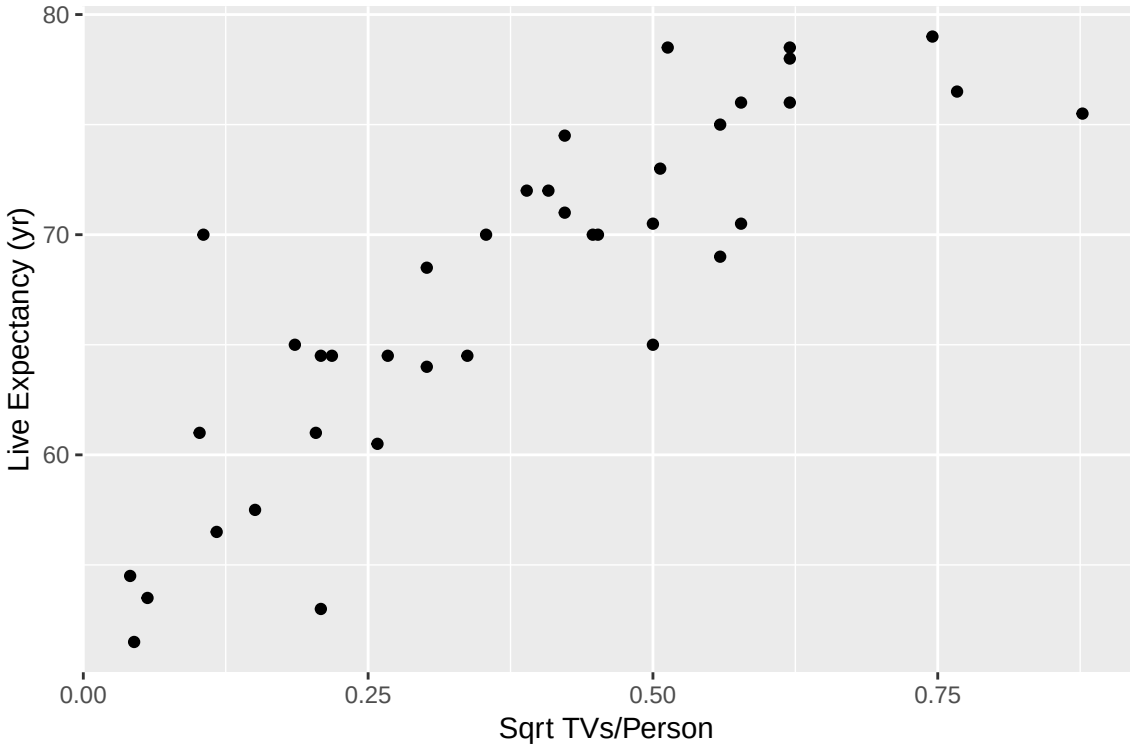
```
##   country = col_character(),
##   `life exp` = col_double(),
##   `sqrtTV/person` = col_double(),
##   `sqrtDoctors/person` = col_double()
## )
```

```
# Figure 8.13, page 243
gf_point(life_exp ~ sqrt_doctors_person, data = Doctors) %>%
  gf_labs(x = "Sqrt Doctors/Person", y = "Life Expectancy (yr)")
```




```
# Figure 8.14
gf_point(life_exp ~ sqrt_tv_person, data = Doctors) %>%
  gf_labs(x = "Sqrt TVs/Person", y = "Live Expectancy (yr)")
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



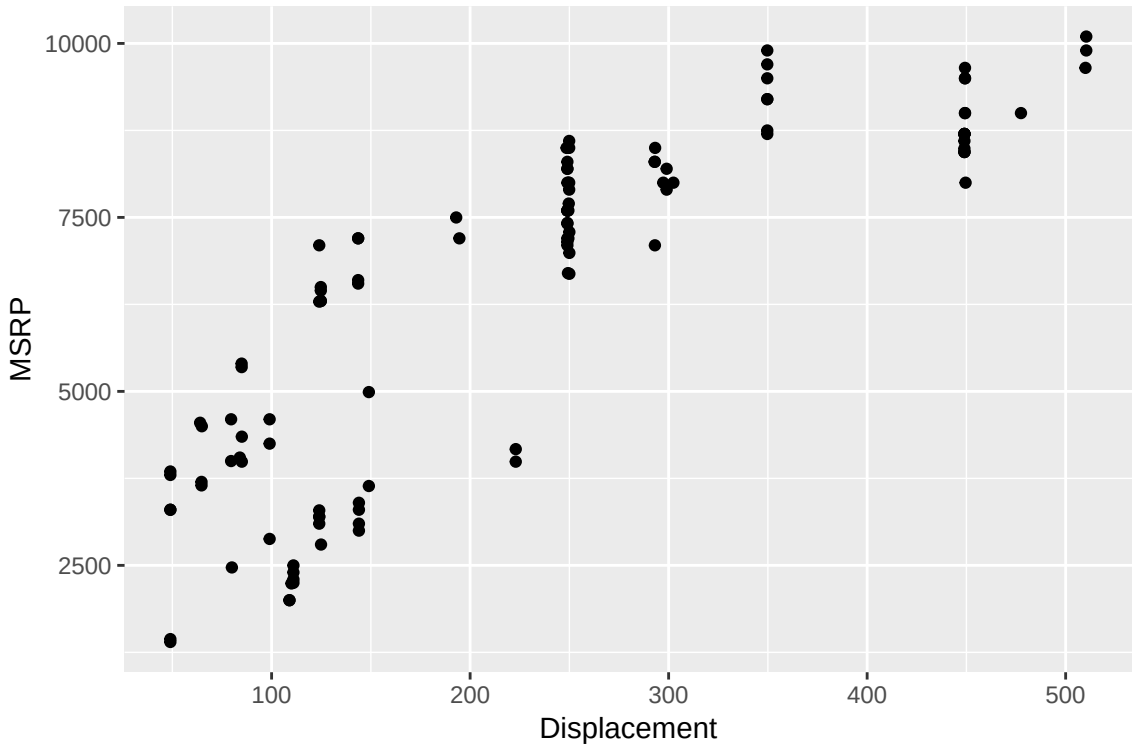
Example 8.2: Using Several of These Methods Together

```
DirtBikes <- read_csv("http://nhorton.people.amherst.edu/is5/data/Dirt_bikes_2014.csv")
```

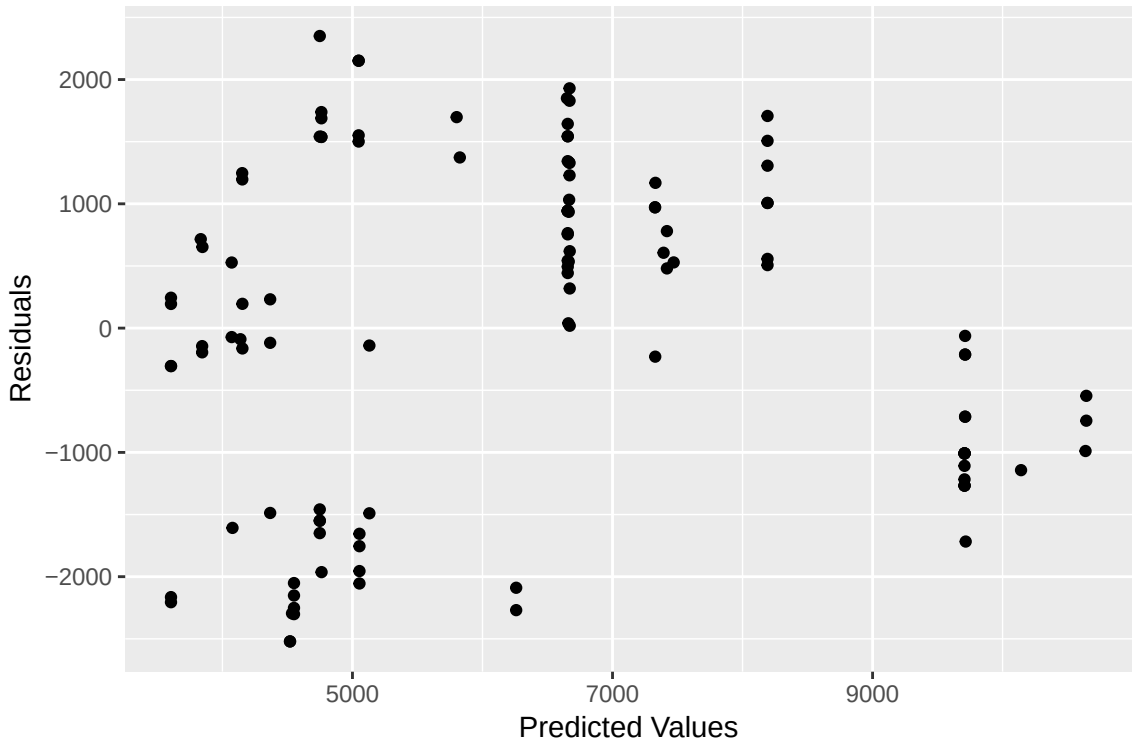
```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Year = col_integer(),
##   MSRP = col_integer(),
##   Displacement = col_double(),
##   `Wheel Base` = col_double(),
##   Bore = col_double(),
##   Stroke = col_double(),
##   Ratio = col_double(),
##   Weight = col_double(),
##   Rake = col_double(),
##   Trail = col_double(),
##   Tank = col_double(),
##   `Engine cooling` = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

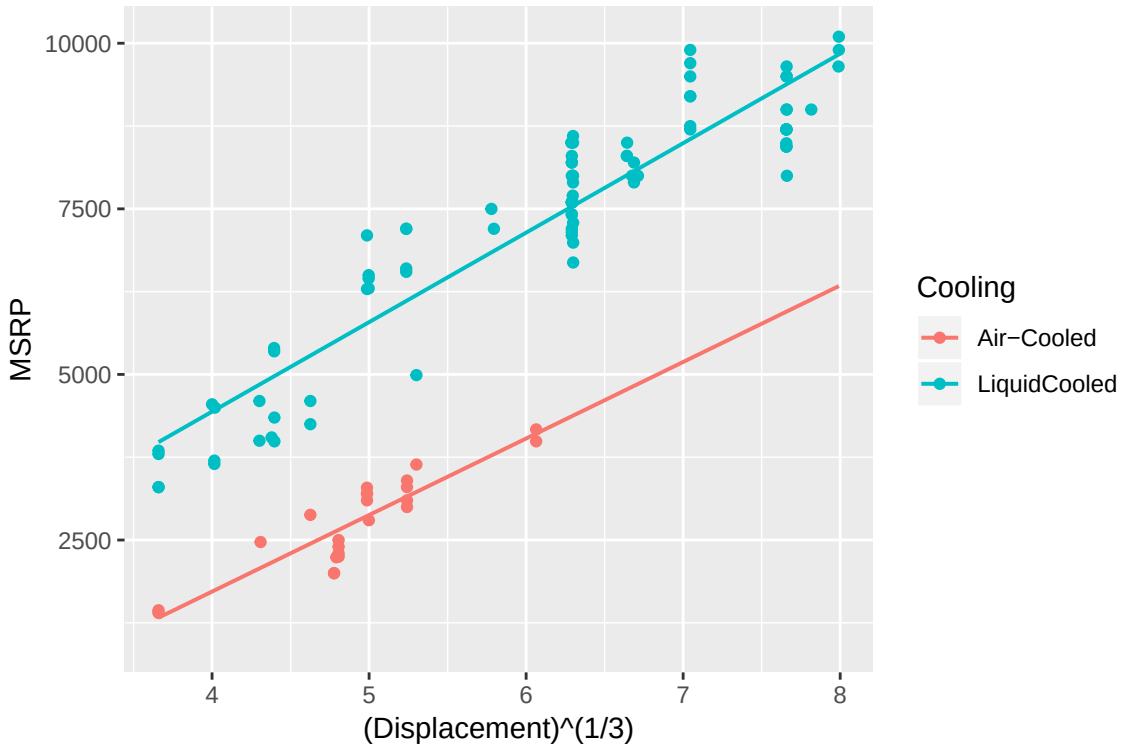
```
gf_point(MSRP ~ Displacement, data = DirtBikes)
```




```
bikeslm <- lm(MSRP ~ Displacement, data = DirtBikes)
gf_point(resid(bikeslm) ~ fitted(bikeslm)) %>%
  gf_labs(x = "Predicted Values", y = "Residuals")
```



```
DirtBikes %>%
  filter(Cooling != "NA") %>%
  mutate(Cooling = ifelse(Cooling == "Air-Cooled", "Air-Cooled", "LiquidCooled")) %>%
  gf_point(MSRP ~ (Displacement)^(1/3), color = ~ Cooling) %>%
  gf_lm()
```

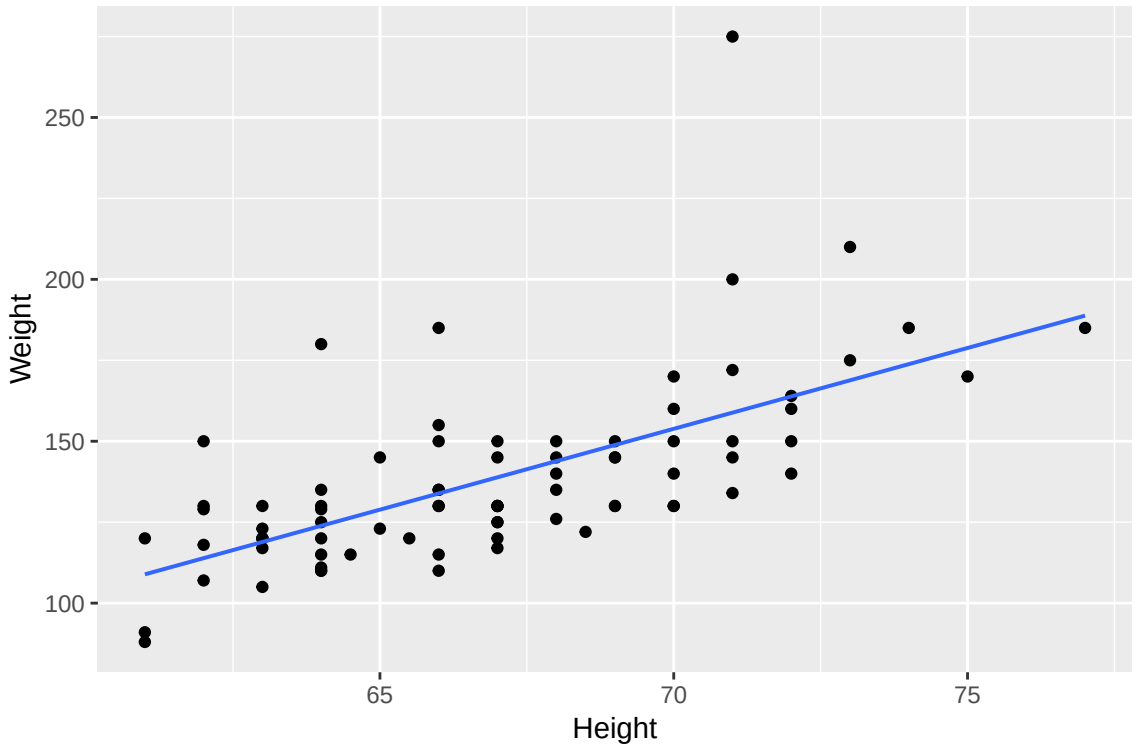


Section 8.5: Working with Summary Values

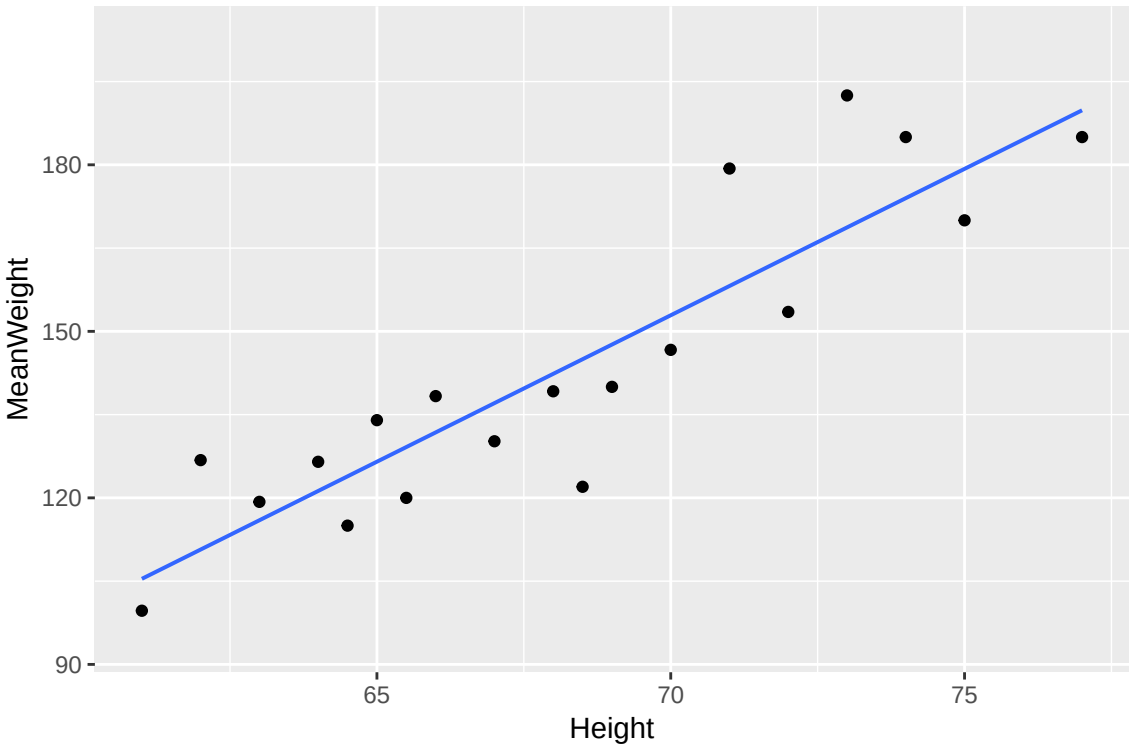
```
HeightsWeights <- read_csv("http://nhorton.people.amherst.edu/is5/data/Heights_and_weights.csv")
```

```
## Parsed with column specification:
## cols(
##   Weight = col_integer(),
##   Height = col_double()
## )
```

```
# Figure 8.15, page 246
gf_point(Weight ~ Height, data = HeightsWeights) %>%
  gf_lm()
```




```
# Figure 8.16
HeightsWeights %>%
  group_by(Height) %>%
  summarise(MeanWeight = mean(Weight)) %>%
  gf_point(MeanWeight ~ Height) %>%
  gf_lm()
```



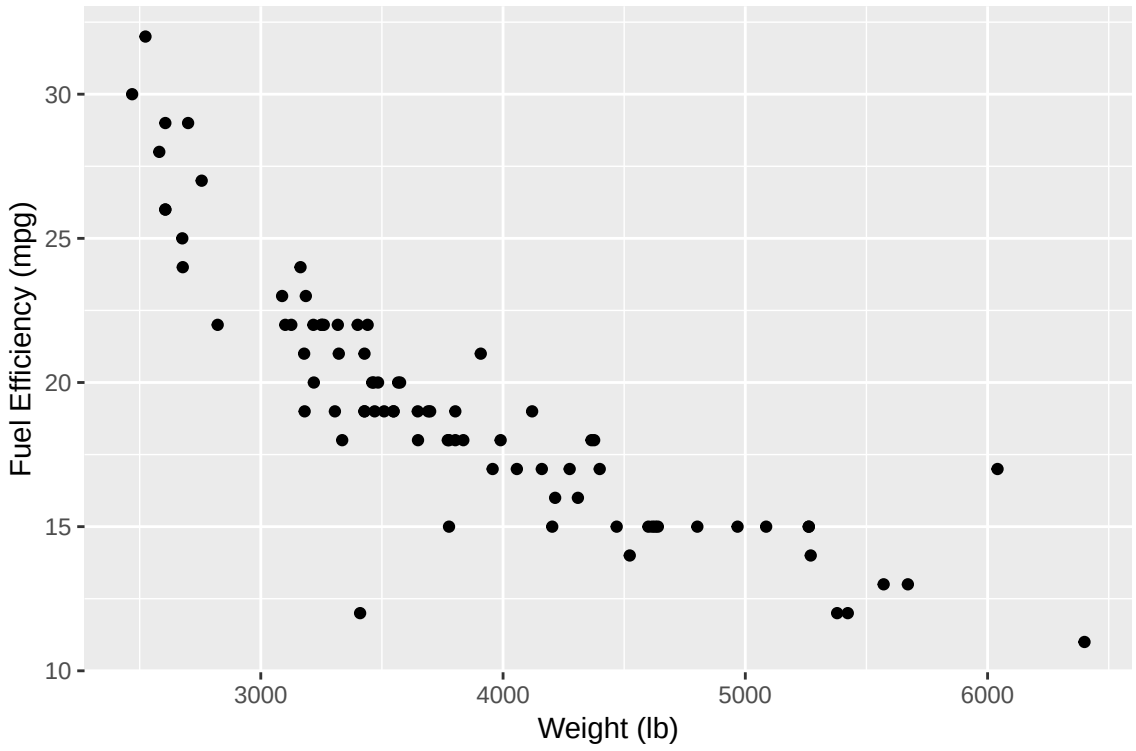
We can use `group_by()` and `summarise()` together to find summary values.

Section 8.6: Straightening Scatterplots–The Three Goals

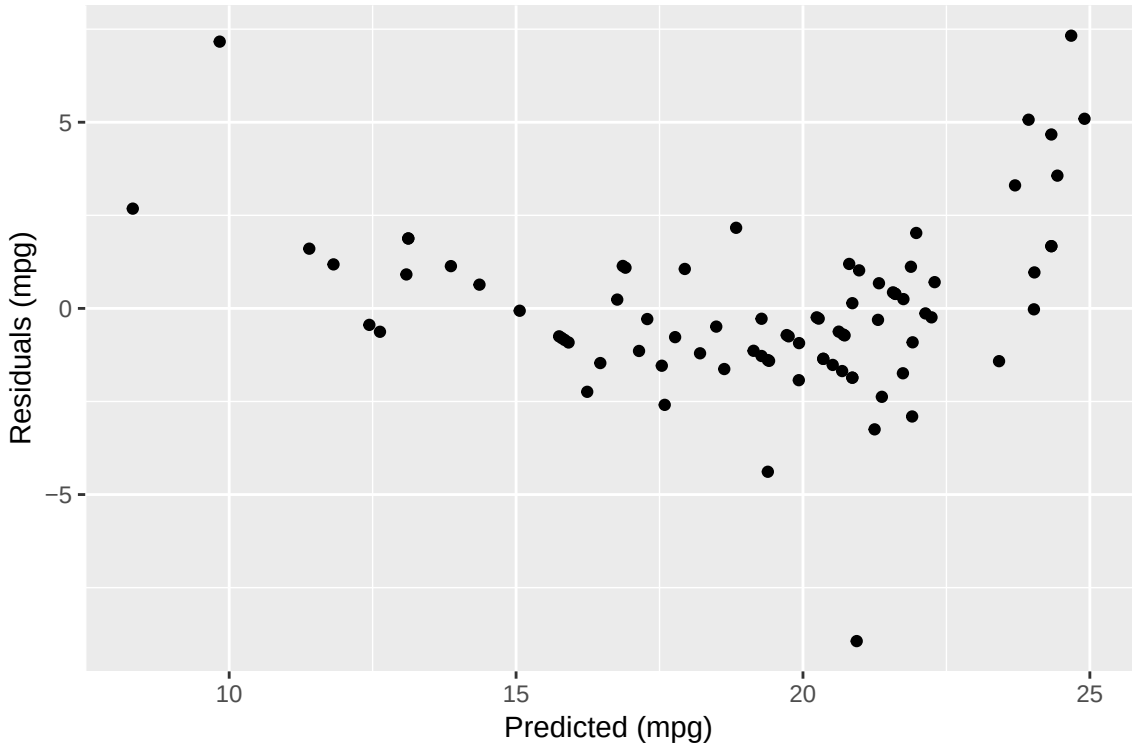
```
FuelEfficiency <- read_csv("http://nhorton.people.amherst.edu/is5/data/Fuel_efficiency.csv") %>%
  clean_names()
```

```
## Parsed with column specification:
## cols(
##   Model = col_character(),
##   `Eng Size` = col_double(),
##   Cylinders = col_integer(),
##   MSRP = col_integer(),
##   `City Mpg` = col_integer(),
##   `Highway Mpg` = col_integer(),
##   Weight = col_integer(),
##   Type = col_character(),
##   Country = col_character()
## )
```

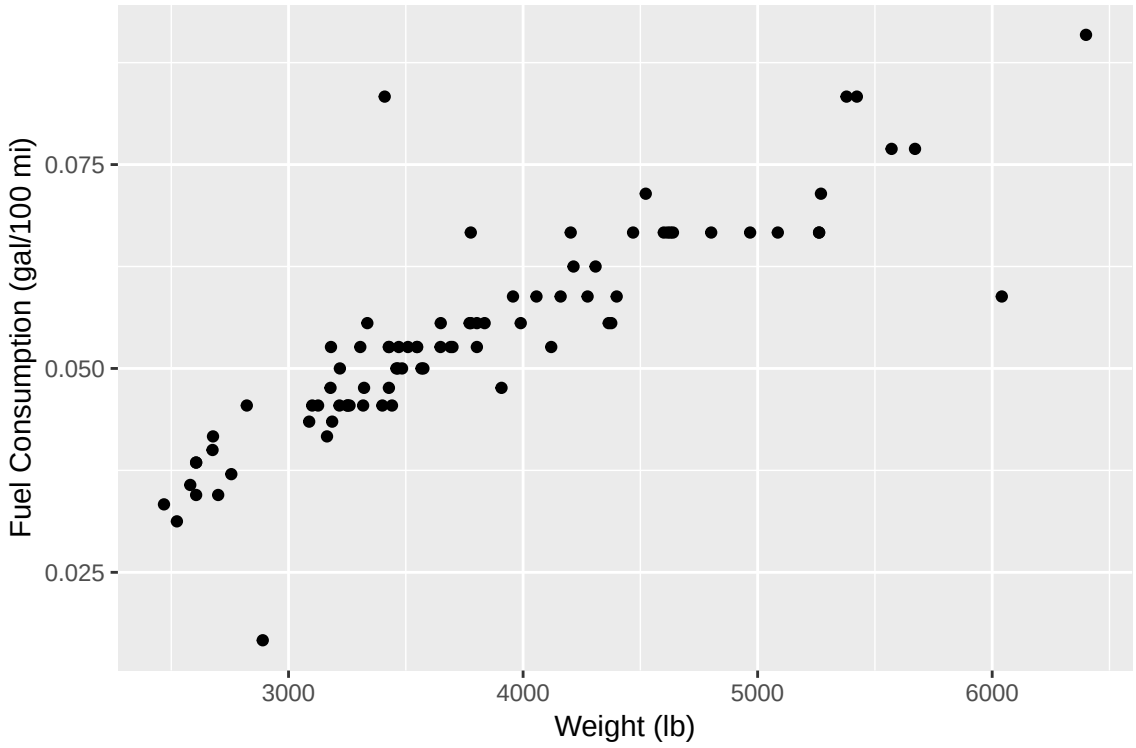
```
# Figure 8.17
gf_point(city_mpg ~ weight, data = filter(FuelEfficiency, city_mpg <= 40)) %>%
  gf_labs(x = "Weight (lb)", y = "Fuel Efficiency (mpg)")
```



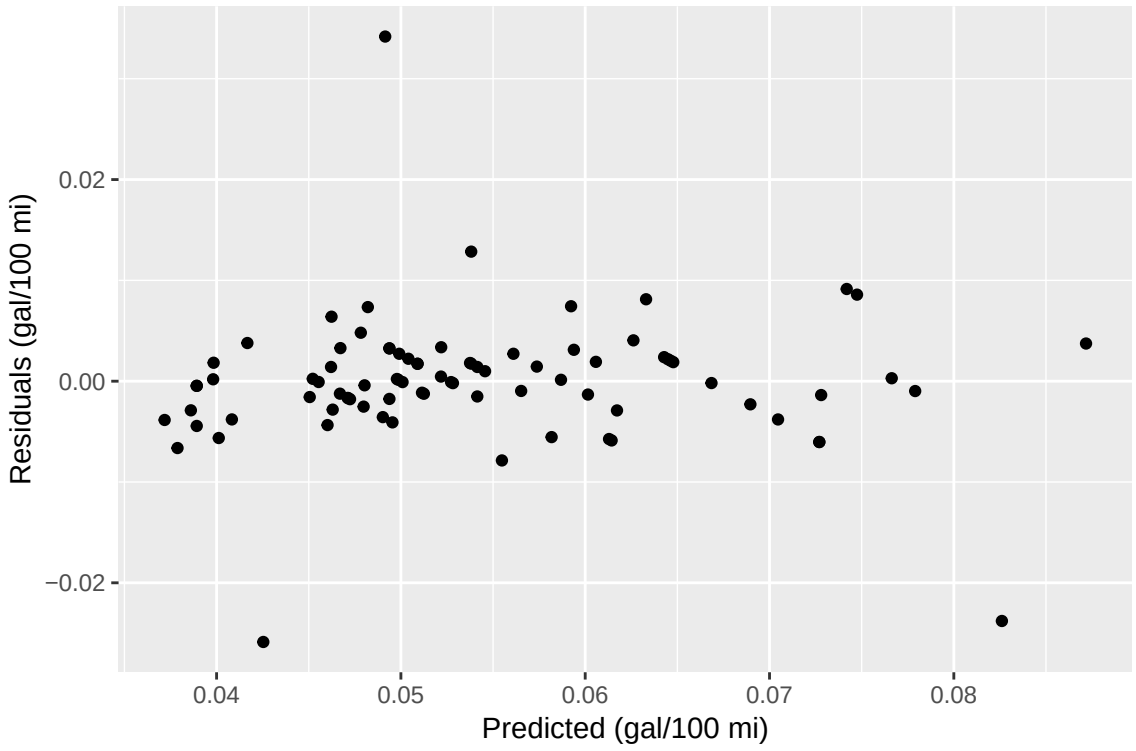

```
fuellm <- lm(city_mpg ~ weight, data = filter(FuelEfficiency, city_mpg <= 40))
gf_point(resid(fuellm) ~ fitted(fuellm)) %>%
  gf_labs(x = "Predicted (mpg)", y = "Residuals (mpg)")
```



```
FuelEfficiency <- FuelEfficiency %>%
  mutate(fuel_consumption = 1/city_mpg)
# Figure 8.19
gf_point(fuel_consumption ~ weight, data = FuelEfficiency) %>%
  gf_labs(x = "Weight (lb)", y = "Fuel Consumption (gal/100 mi)")
```




```
fuellm2 <- lm(fuel_consumption ~ weight, data = FuelEfficiency)
gf_point(resid(fuellm2) ~ fitted(fuellm2)) %>%
  gf_labs(x = "Predicted (gal/100 mi)", y = "Residuals (gal/100 mi)")
```



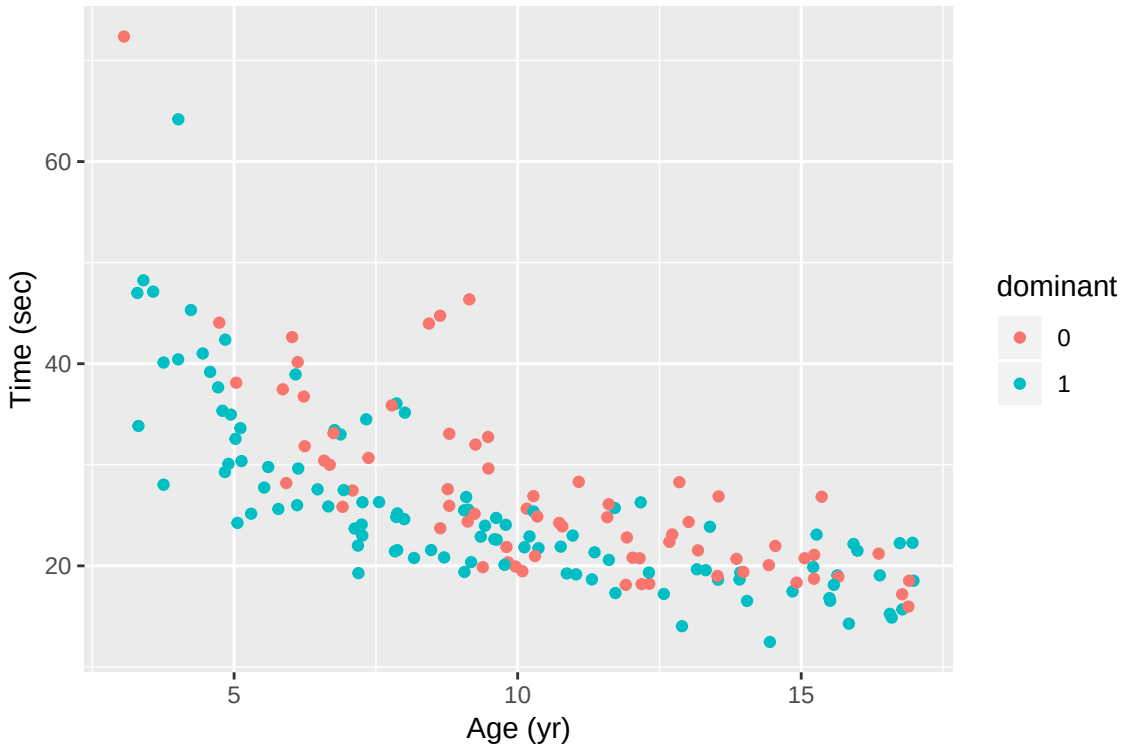
Goals of Re-Expression for Regression

```
HandDexterity <- read_csv("http://nhorton.people.amherst.edu/is5/data/Hand_dexterity.csv") %>%
  clean_names() %>%
  mutate(dominant = as.factor(dominant))
```

```
## Parsed with column specification:
## cols(
##   `Time(sec)` = col_double(),
##   Speed = col_double(),
##   `Age(yr)` = col_double(),
##   Dominant = col_integer(),
##   `Gender#` = col_integer(),
##   HD = col_character(),
##   `hand used` = col_character()
## )
```

```
# Figure 8.20, page 248
gf_point(time_sec ~ age_yr, color = ~ dominant, data = HandDexterity) %>%
  gf_labs(x = "Age (yr)", y = "Time (sec)")
```

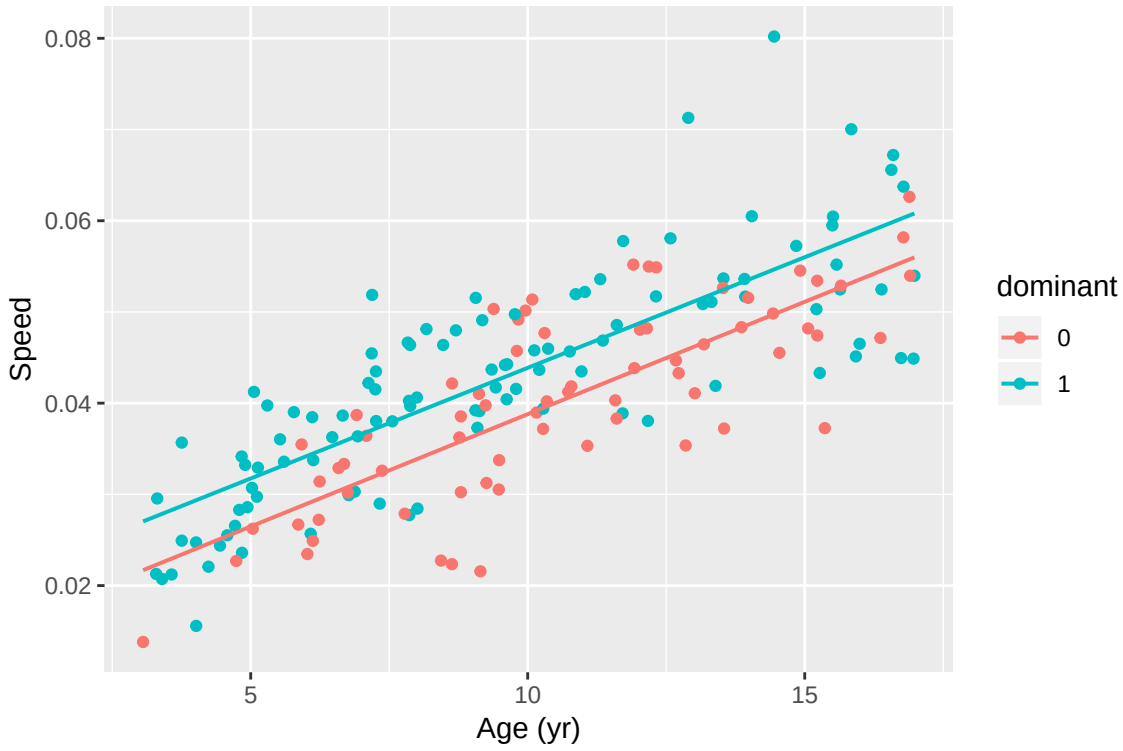
```
## Warning: Removed 1 rows containing missing values (geom_point).
```



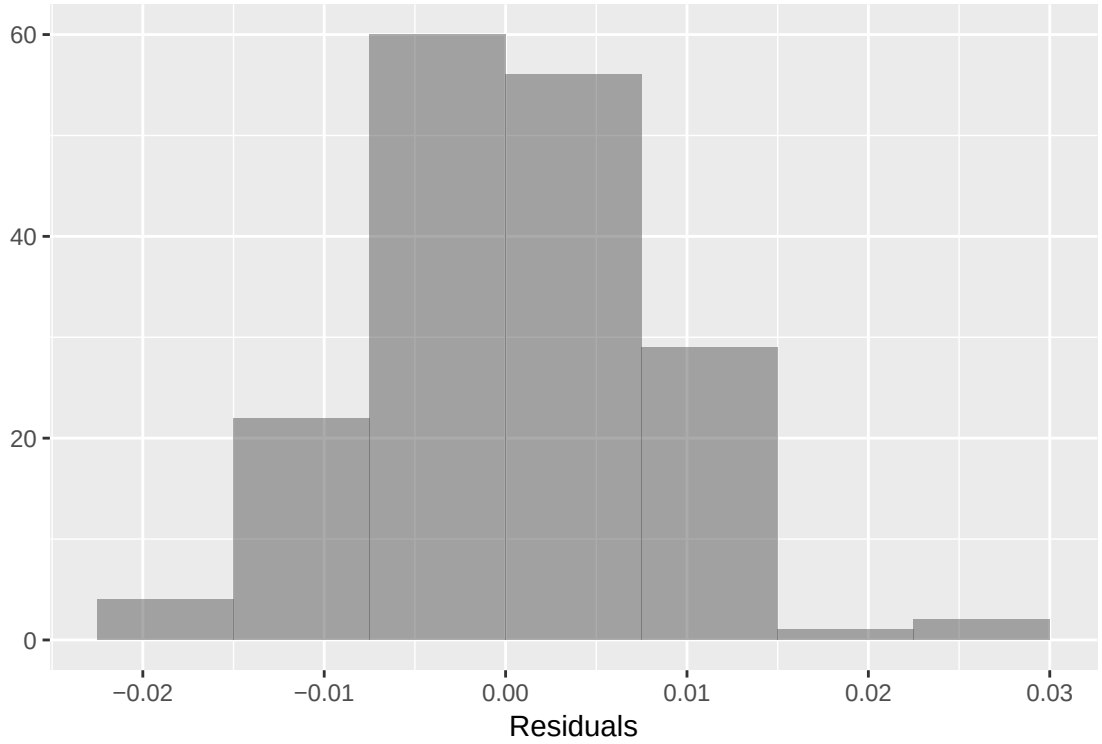

```
HandDexterity <- HandDexterity %>%
  mutate(speed = 1/time_sec)
# Figure 8.21
gf_point(speed ~ age_yr, color = ~ dominant, data = HandDexterity) %>%
  gf_lm() %>%
  gf_labs(x = "Age (yr)", y = "Speed")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_lm).

## Warning: Removed 1 rows containing missing values (geom_point).
```




```
handlm <- lm(speed ~ age_yr, data = HandDexterity)
# Figure 8.22
gf_histogram(~ resid(handlm), binwidth = .0075, center = .0075/2) %>%
  gf_labs(x = "Residuals", y = "")
```

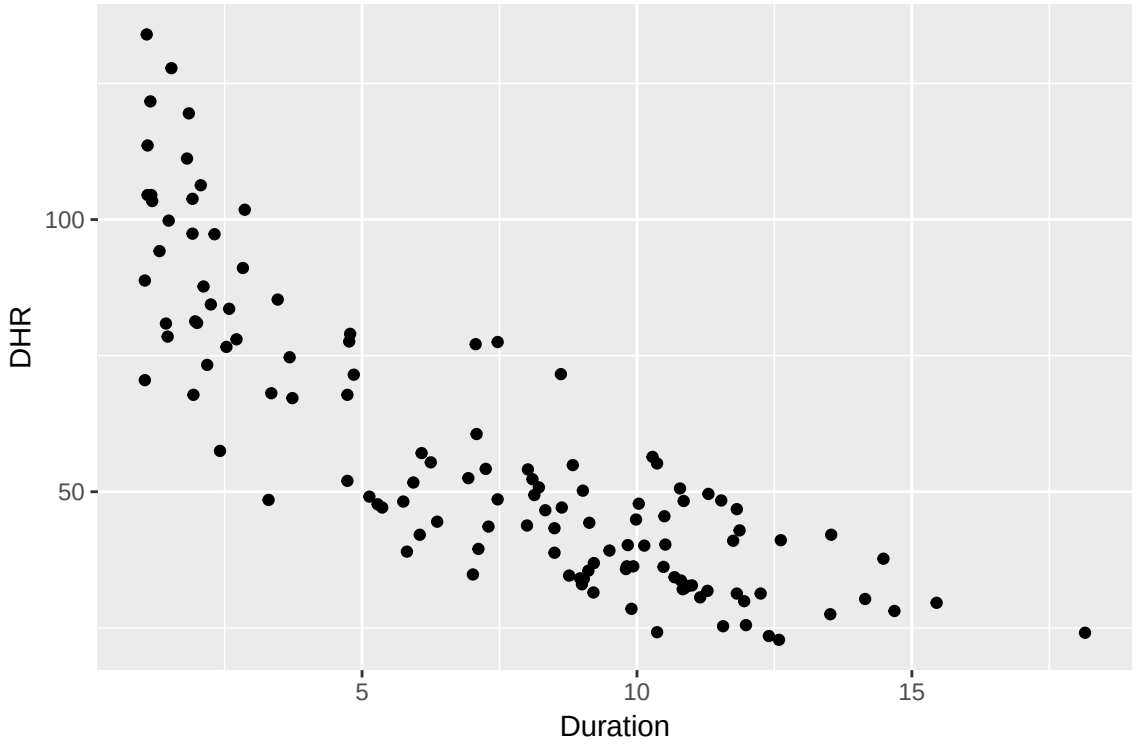


Section 8.7: Finding a Good Re-expression

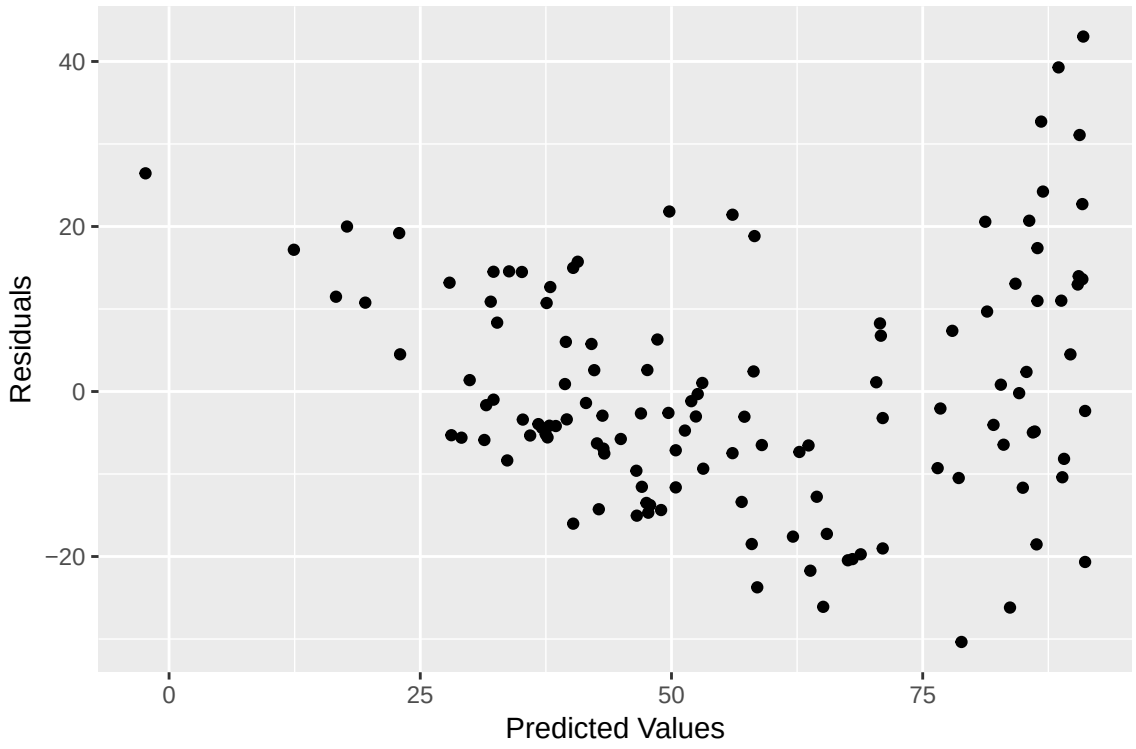
See table and Figure 8.23 on page 250.

Step-By-Step Example: Re-Expressing to Straighten a Scatterplot

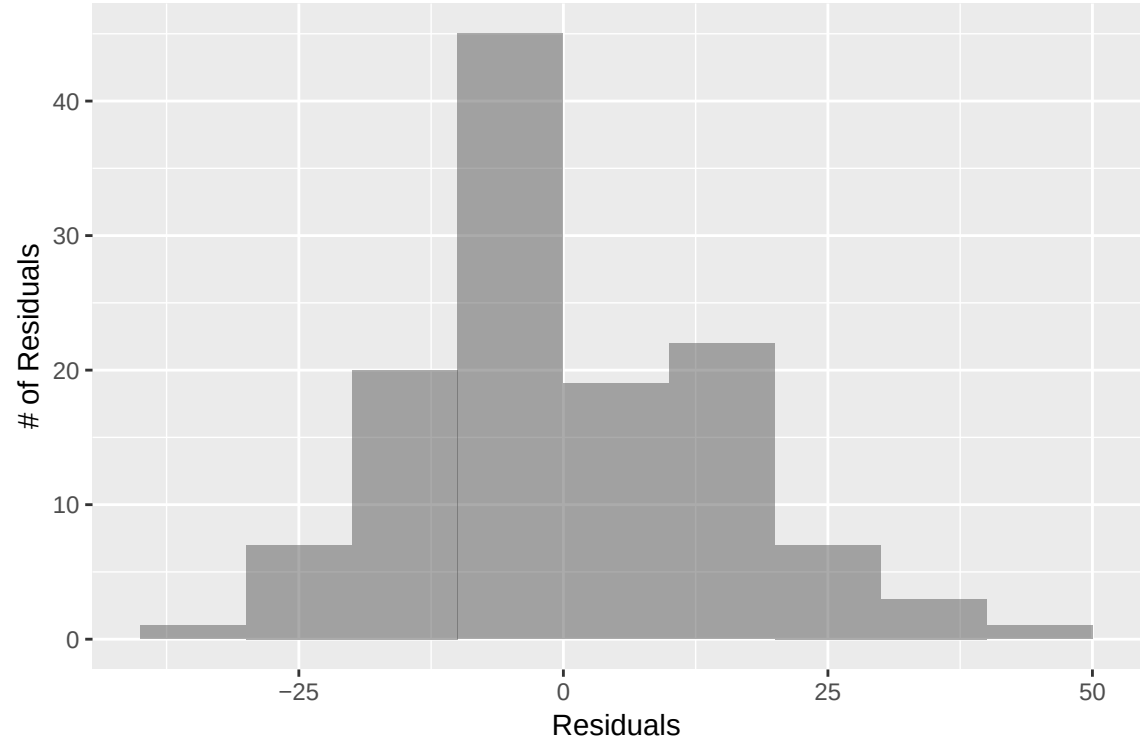
```
gf_point(dive_heart_rate ~ duration_min, data = Penguins, xlab = "Duration", ylab = "DHR")
```




```
gf_point(resid(penguinlm) ~ fitted(penguinlm), data = Penguins, xlab = "Predicted Values", ylab = "Resi
```

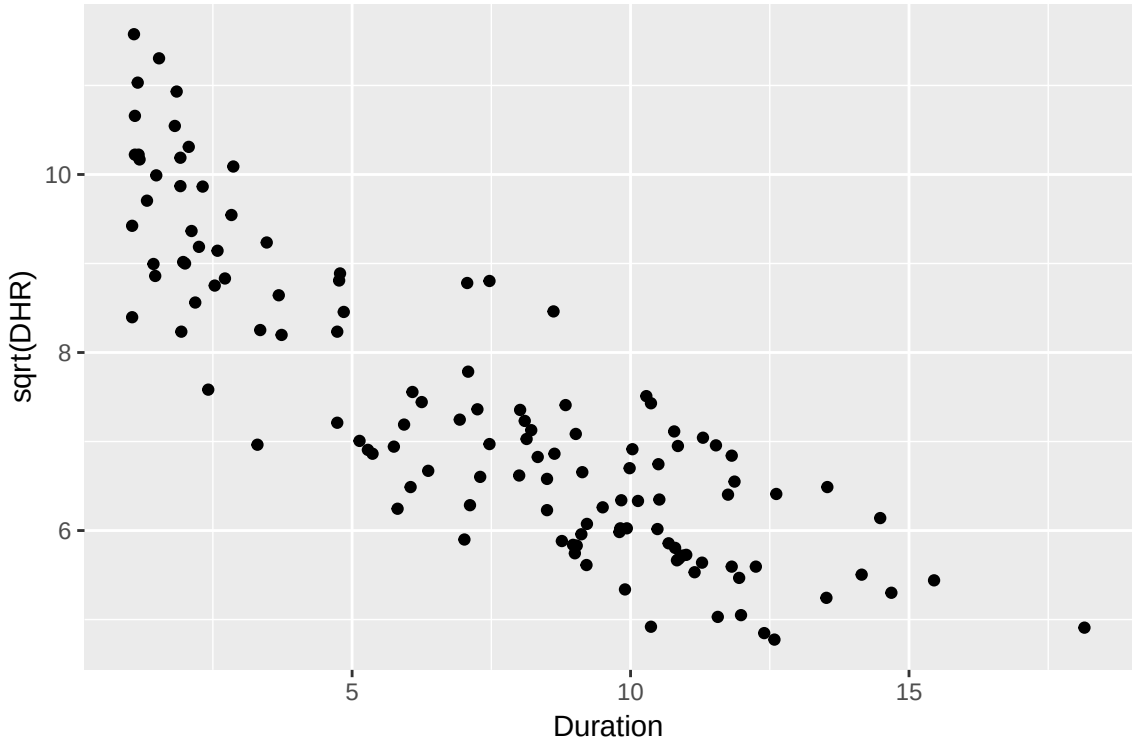



```
gf_histogram(~ resid(penguinlm), binwidth = 10, center = 5) %>%
  gf_labs(x = "Residuals", y = "# of Residuals")
```



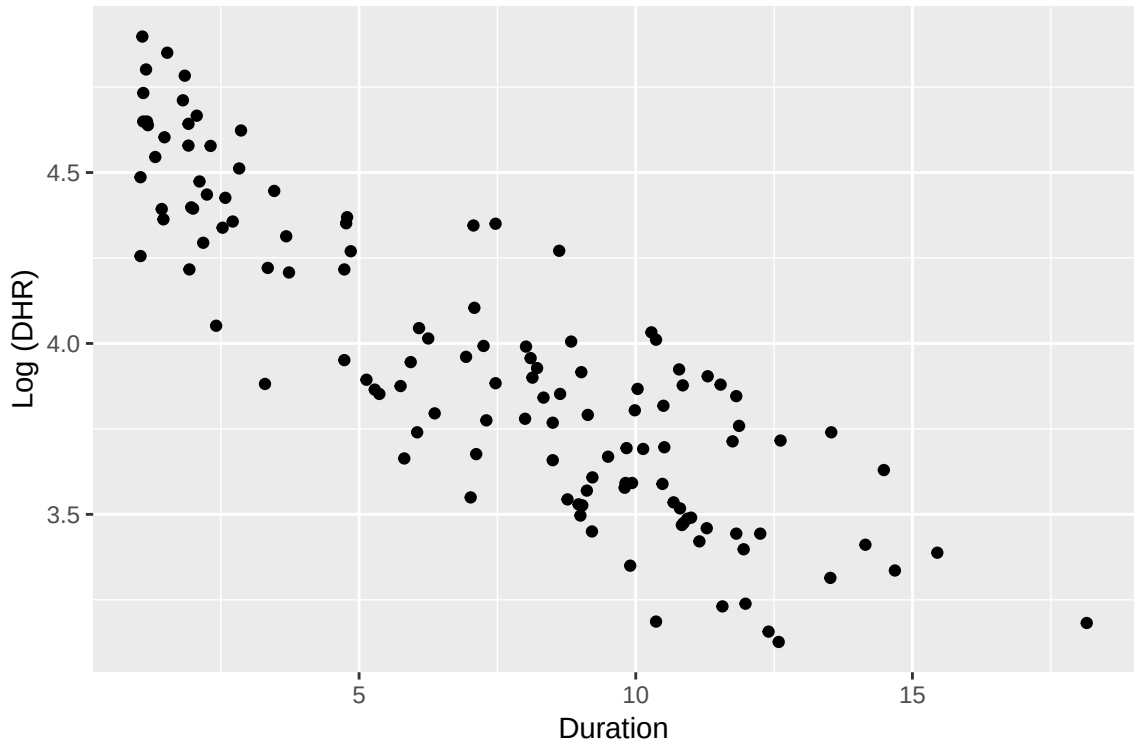
Mutating with the square root:

```
gf_point((dive_heart_rate)^(1/2) ~ duration_min, data = Penguins) %>%
  gf_labs(x = "Duration", y = "sqrt(DHR)")
```

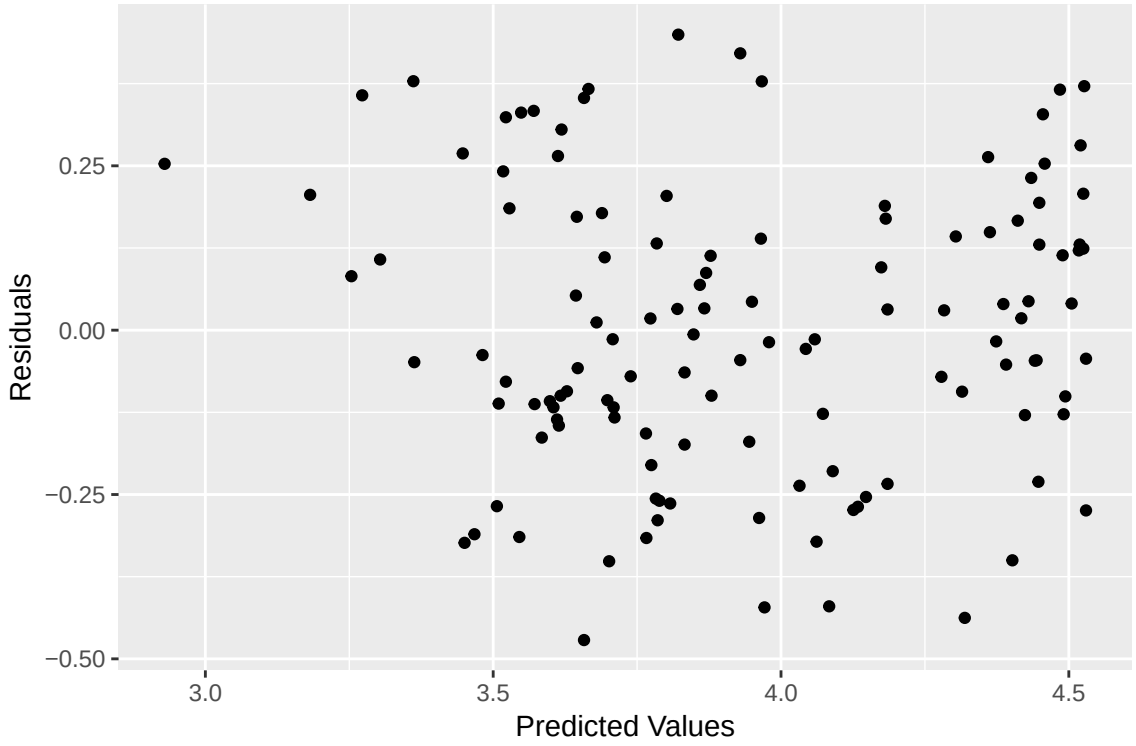


Mutating with a log:

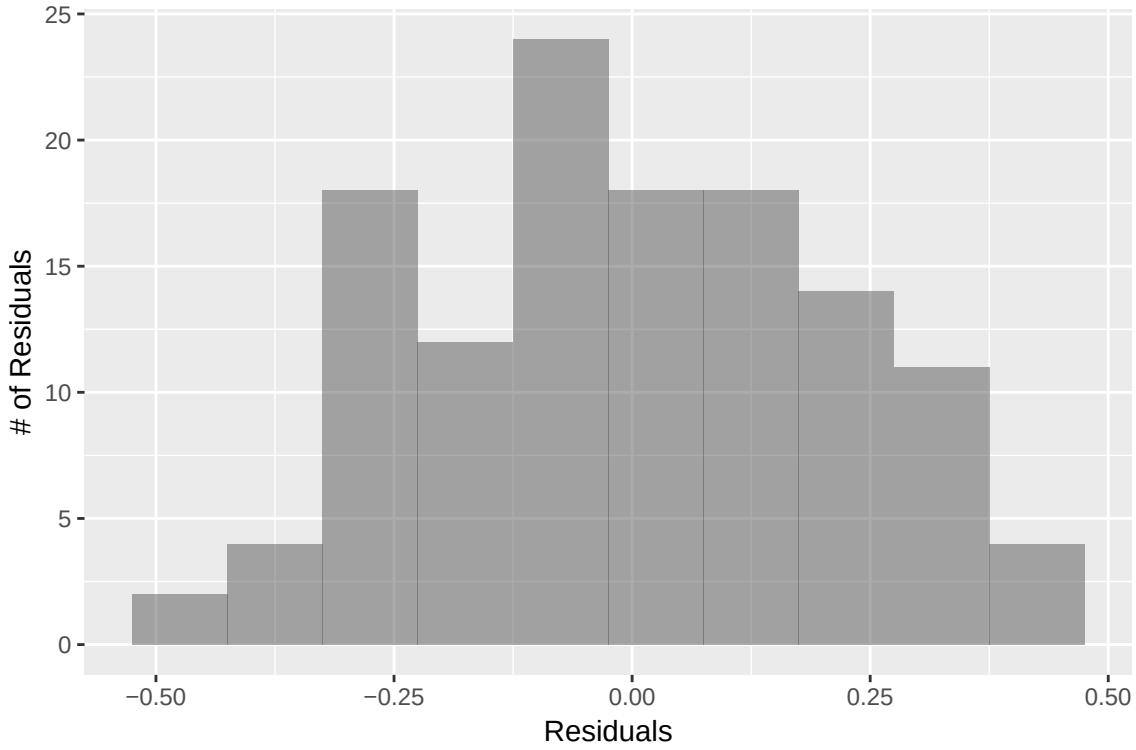
```
gf_point(log(dive_heart_rate) ~ duration_min, data = Penguins) %>%
  gf_labs(x = "Duration", y = "Log (DHR)")
```




```
penguinlm2 <- lm(log(dive_heart_rate) ~ duration_min, data = Penguins)
gf_point(resid(penguinlm2) ~ fitted (penguinlm2)) %>%
  gf_labs(x = "Predicted Values", y = "Residuals")
```

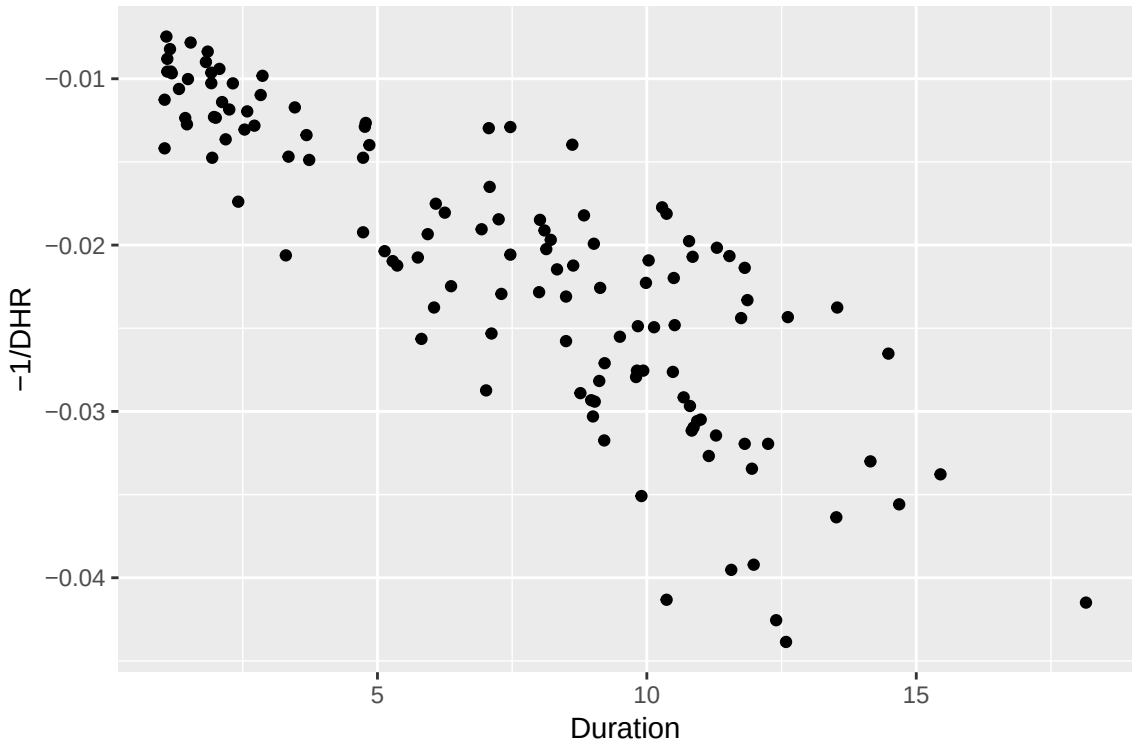


```
gf_histogram(~ resid(penguinlm2), binwidth = 0.1, xlab = "Residuals", ylab = "# of Residuals",
             center = .025)
```

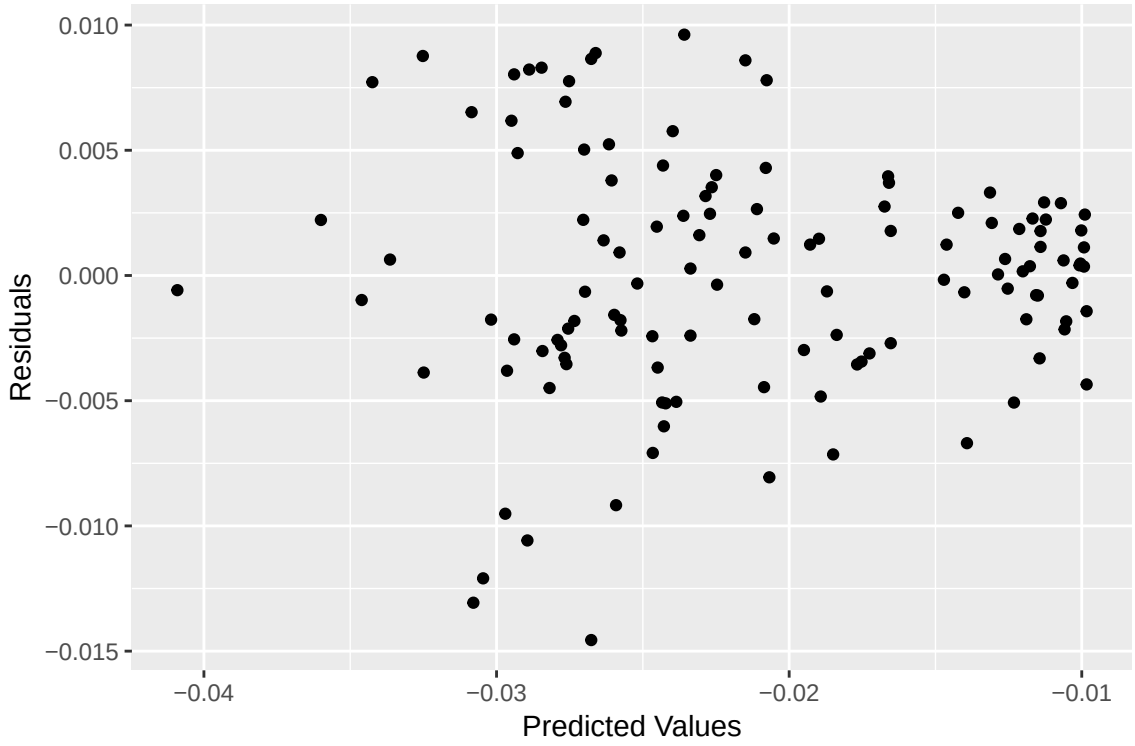


Mutating with a (negative) reciprocal:

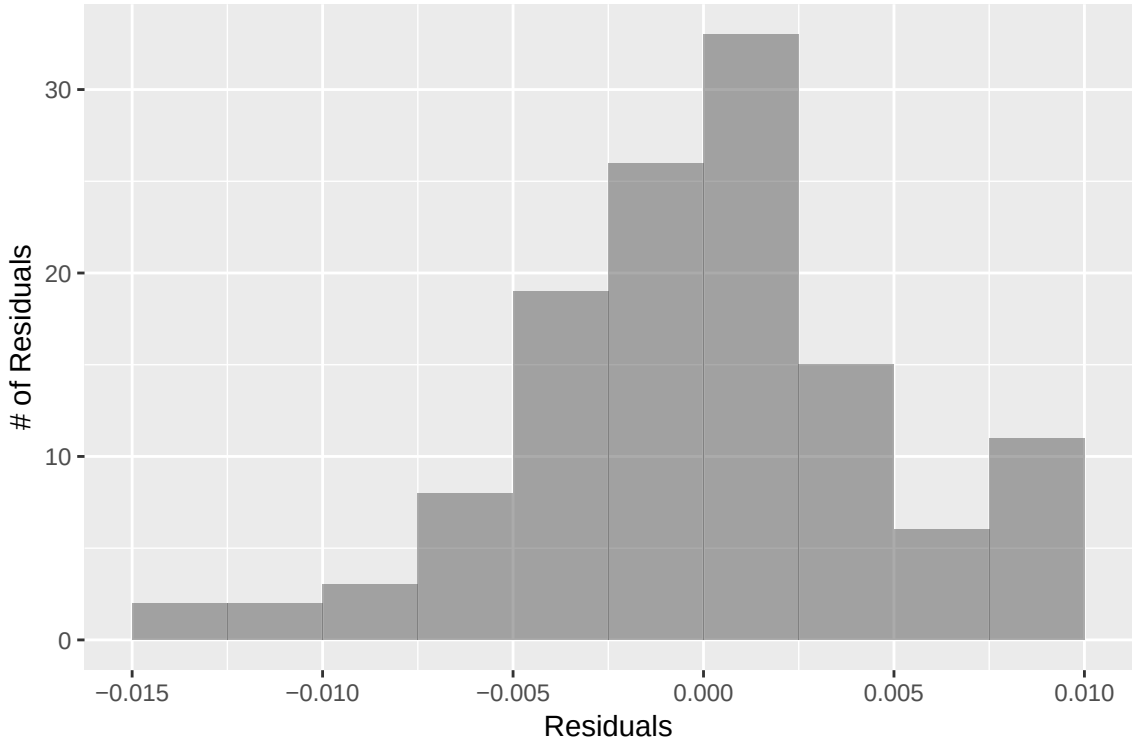
```
gf_point(-1/(dive_heart_rate) ~ duration_min, data = Penguins) %>%
  gf_labs(x = "Duration", y = "-1/DHR")
```




```
penguinlm3 <- lm(-1/(dive_heart_rate) ~ duration_min, data = Penguins)
gf_point(resid(penguinlm3) ~ fitted(penguinlm3)) %>%
  gf_labs(x = "Predicted Values", y = "Residuals")
```



```
gf_histogram(~ resid(penguinlm3), binwidth = 0.0025, xlab = "Residuals", ylab = "# of Residuals",
             center = 0.00125)
```



$-1/\sqrt{DHR}$ follows the same process on pages 253-254.