IS5 in R: Regression Wisdom (Chapter 8)

Nicholas Horton (nhorton@amherst.edu)

2025-01-14

Table of contents

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. This file as well as the associated Quarto 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 (https://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.

We begin by loading packages that will be required for our analyses.

```
library(mosaic)
library(tidyverse)
```

Chapter 8: Regression Wisdom

Section 8.1: Examining Residuals

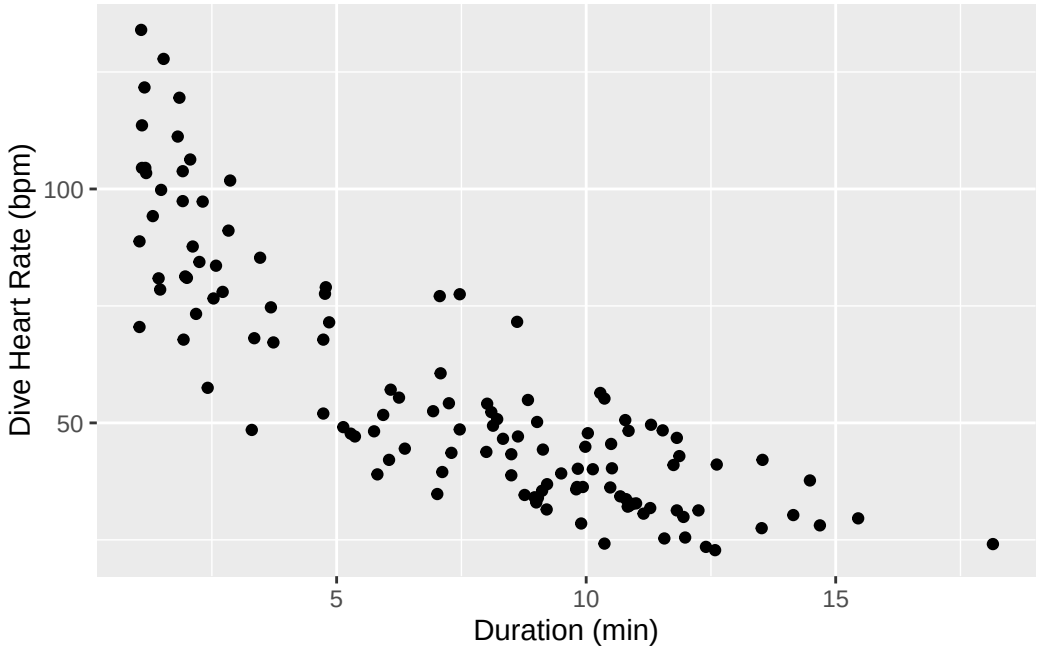
Getting the "Bends": When the Residuals Aren't Straight

We begin by reading in the data.

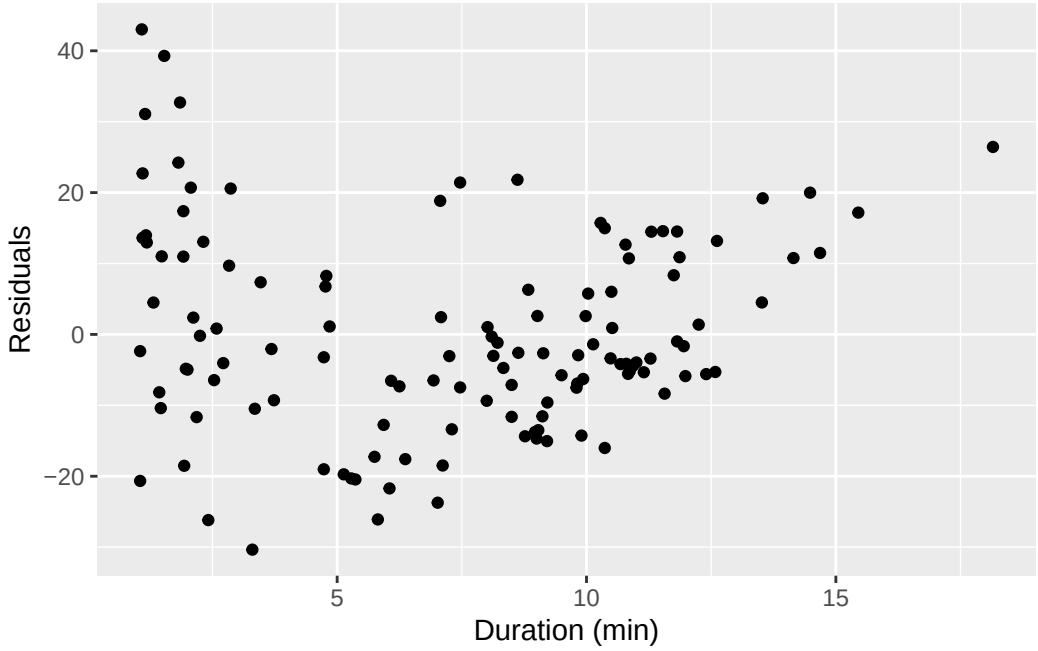
```
Penguins <- read_csv("http://nhorton.people.amherst.edu/is5/data/Penguins.csv") |>
  janitor::clean_names()
```

By default, the `read_csv()` function prints the variable names. These messages have been 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).

```
# Figure 8.1, page 234
gf_point(dive_heart_rate ~ duration_min, data = Penguins) |>
  gf_labs(x = "Duration (min)", y = "Dive Heart Rate (bpm)")
```



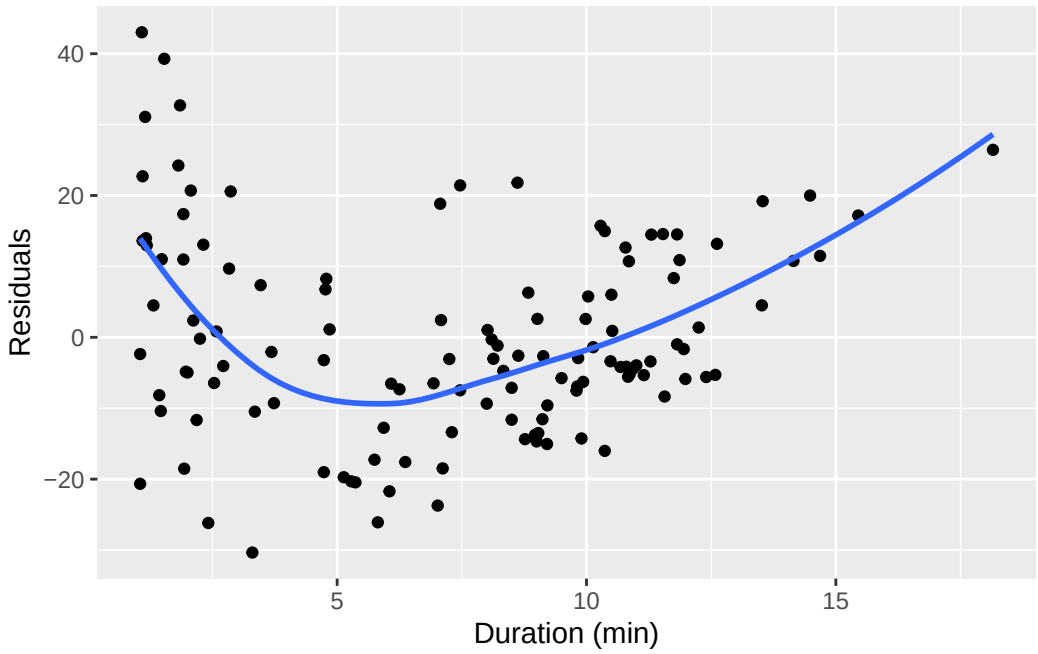
```
penguinlm <- lm(dive_heart_rate ~ duration_min, data = Penguins)
# Figure 8.2
gf_point(resid(penguinlm) ~ duration_min, data = Penguins) |>
  gf_labs(x = "Duration (min)", y = "Residuals")
```



This figure calls out for a smoother (to help better understand the relationship between duration and heart rate). It would be appropriate to add the `message: false` option to suppress the message about which method is being used since it is a distraction. Here we've left it.

```
# improved Figure 8.2
gf_point(resid(penguinlm) ~ duration_min, data = Penguins) |>
  gf_labs(x = "Duration (min)", y = "Residuals") |>
  gf_smooth(se = FALSE)
```

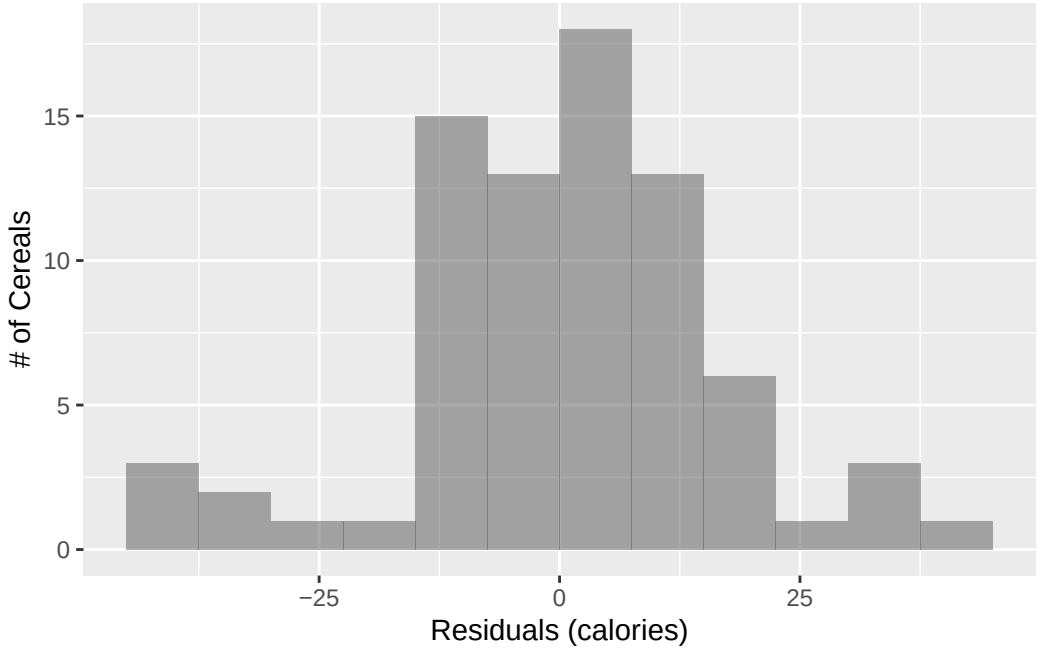
```
`geom_smooth()` using method = 'loess'
```



Sifting Residuals for Groups

We begin by reading in the data.

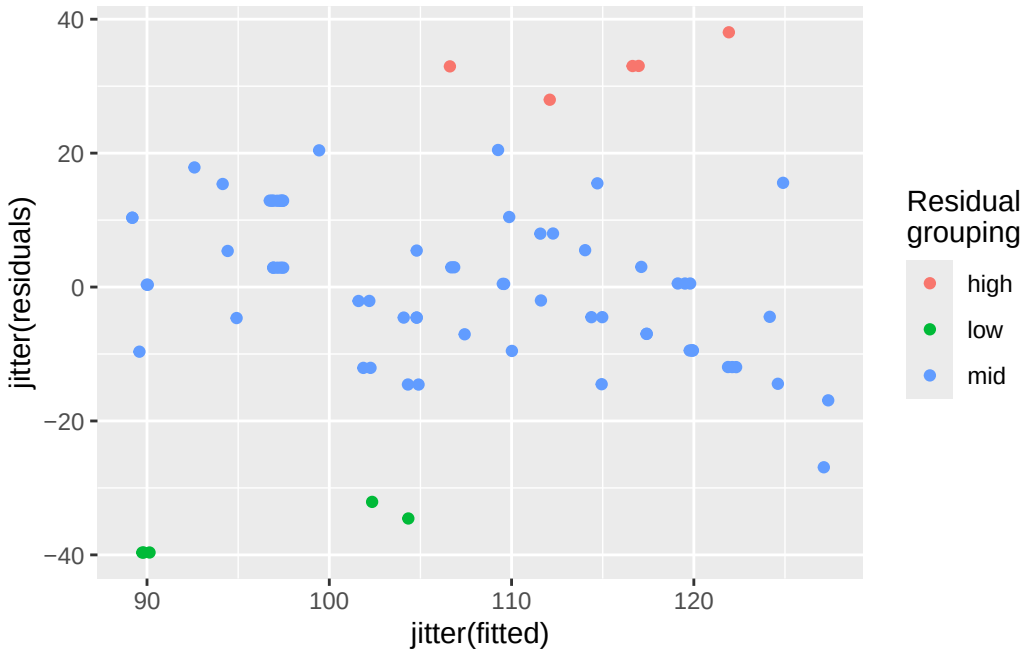
```
Cereal <- read_csv("http://nhorton.people.amherst.edu/is5/data/Cereals.csv")
cereallm <- lm(calories ~ sugars, data = Cereal)
# Figure 8.3, page 235
gf_histogram(~ resid(cereallm), binwidth = 7.5, center = 7.5 / 2) |>
  gf_labs(x = "Residuals (calories)", y = "# of Cereals")
```




```
Cereal <- Cereal |>
  mutate(
    residuals = resid(cereallm),
    fitted = fitted(cereallm)
  ) |>
  mutate(
   resid_cat = case_when(
     residuals >= -30 & residuals <= 25 ~ "mid",
     residuals > 25 ~ "high",
     TRUE ~ "low" # or residuals < -30!
  )) # Needed for color categories
```

The `case_when()` function allows us to use logic to create groupings based on the values of the residual. This approach is far more straightforward than making nested `ifelse()` calls.

```
# Figure 8.4
gf_point(jitter(residuals) ~ jitter(fitted), color = ~ resid_cat, data = Cereal) |>
  gf_labs(color = "Residual\ngrouping")
```



The `jitter()` function adds some random noise to allow easier observation of values that are shared by more than one type of breakfast cereal.

Here we relabeled the legend title to improve readability.

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

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

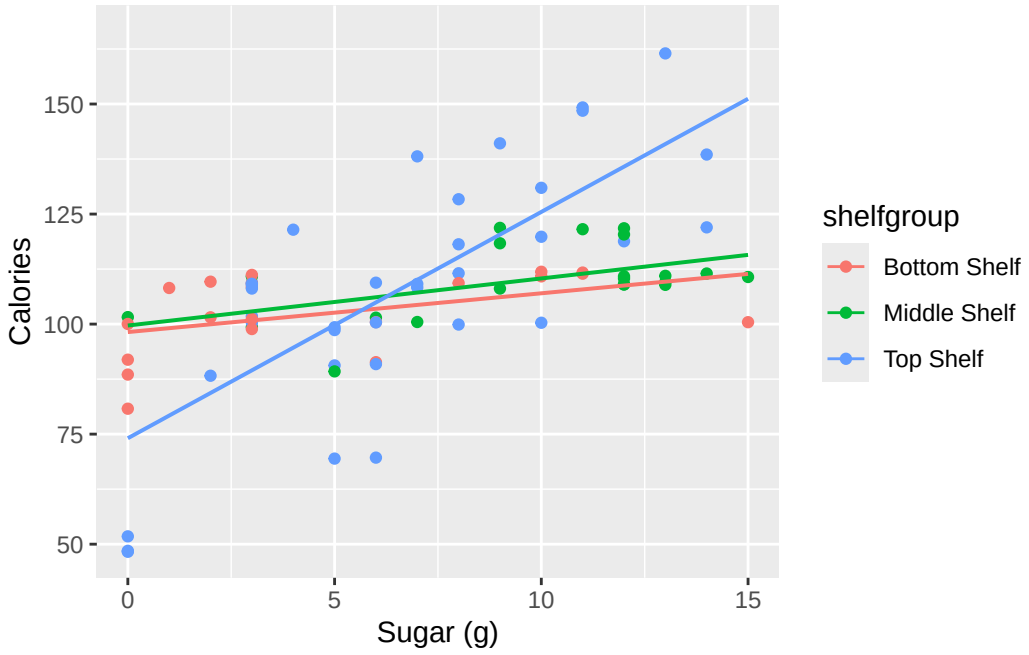
```
Cereal <- Cereal |>
  mutate(,
  shelfgroup =
    recode(shelf,
      `1` = "Bottom Shelf",
      `2` = "Middle Shelf",
      `3` = "Top Shelf"
    )
)
```

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

```
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")
```



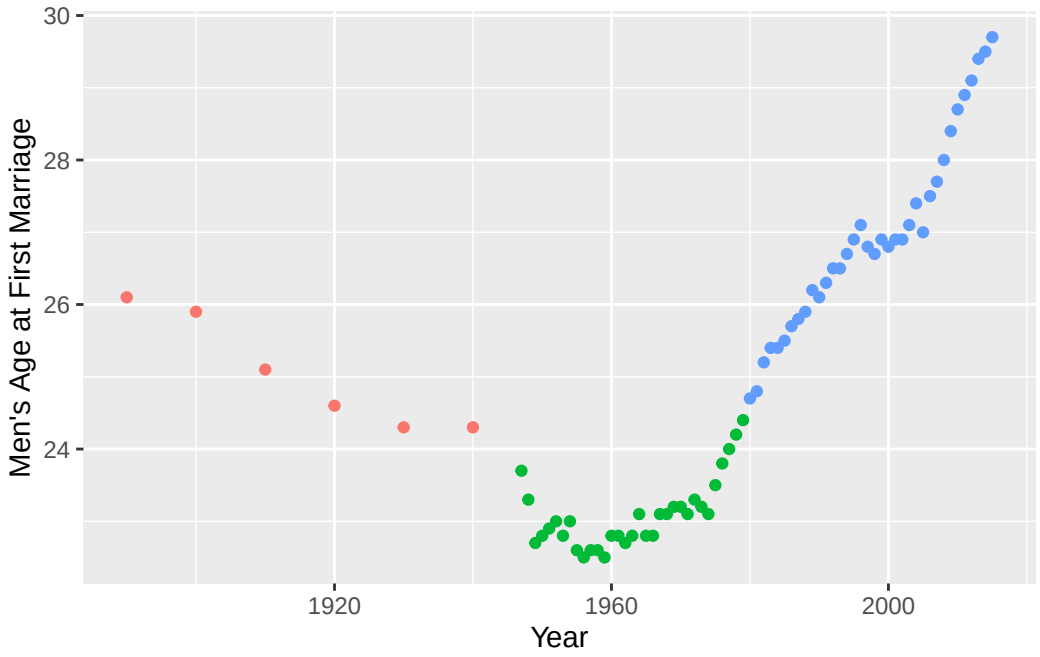
Section 8.2: Extrapolation: Reaching Beyond the Data

See displays on page 237 and 238.

Example 8.1: Extrapolation: Reaching Beyond the Data

We begin by reading in the data.

```
MarriageAge <- read_csv("http://nhorton.people.amherst.edu/is5/data/Marriage_age_2015.csv")
MarriageAge <- MarriageAge |>
  mutate(
    timeperiod = case_when(
      Year <= 1940 ~ "Section1",
      Year >= 1980 ~ "Section3",
      TRUE ~  "Section2"
  ))
gf_point(
  Men ~ Year,
  color = ~ timeperiod,
  data = MarriageAge,
  ylab = "Men's Age at First Marriage"
) +
  guides(color = "none")
```



The `case_when()` function works similarly to `recode()` for more general logic. Here we use a `ggplot2` function (`guides()`) which requires special syntax (+) to connect to a ggformula graphic.

Section 8.3: Outliers, Leverage, and Influence

We begin by reading in the data.

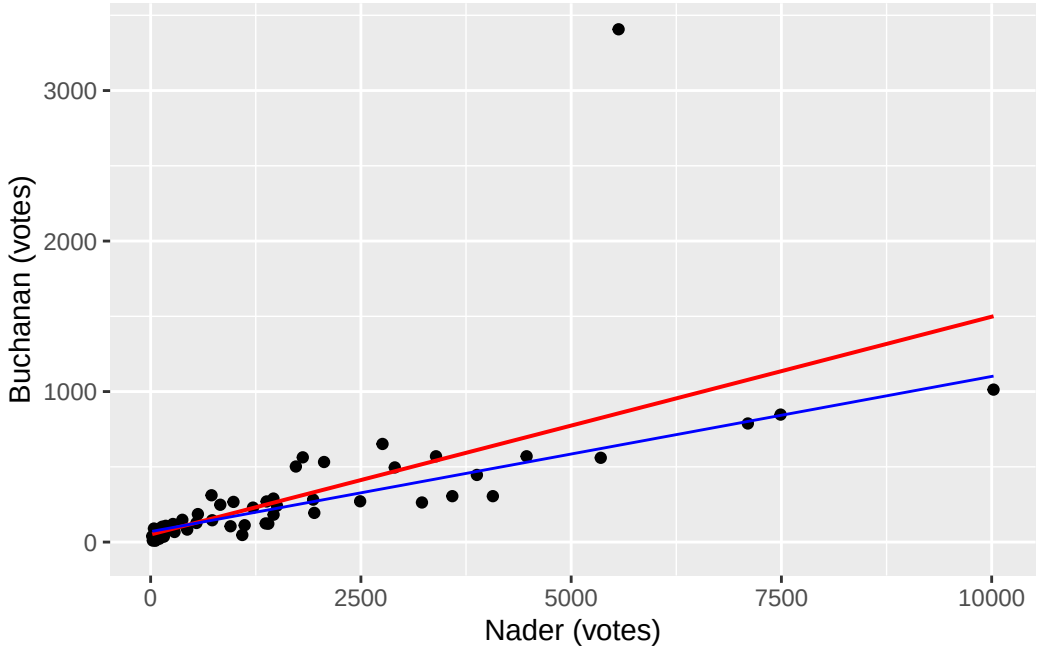
```
Election2000 <- read_csv("http://nhorton.people.amherst.edu/is5/data/Election_2000.csv")
withlm <- lm(Buchanan ~ Nader, data = Election2000)
msummary(withlm)
```

```
            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(color = "red") |>
  gf_labs(x = "Nader (votes)", y = "Buchanan (votes)") |>
  gf_fun(withoutlm, color = "blue") # adds line for model without outlier
```

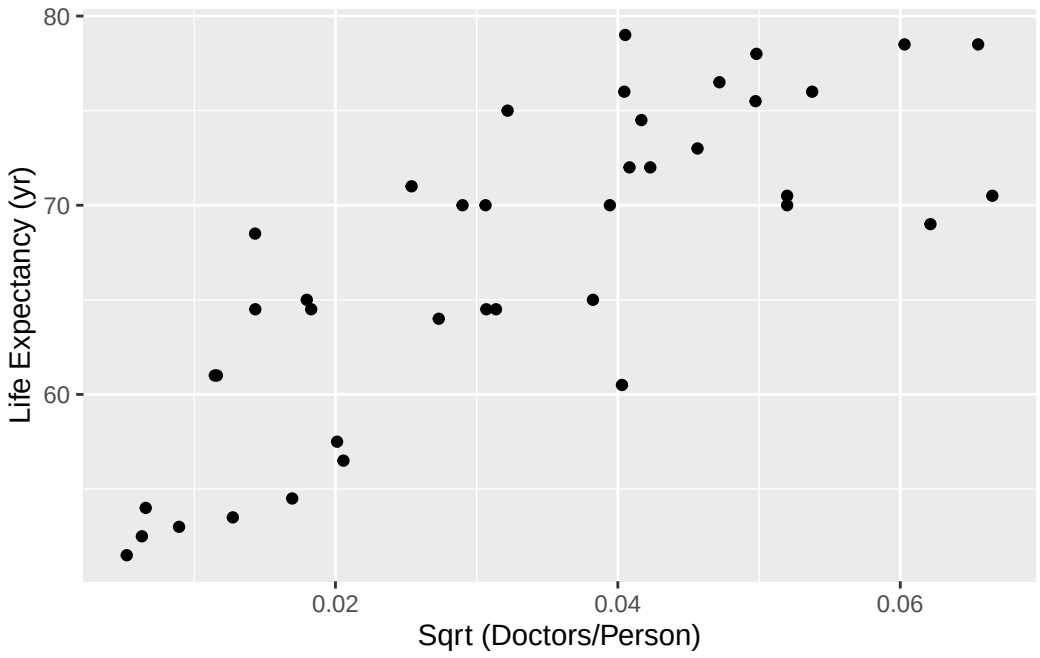


See page 242 for example of high-leverage point.

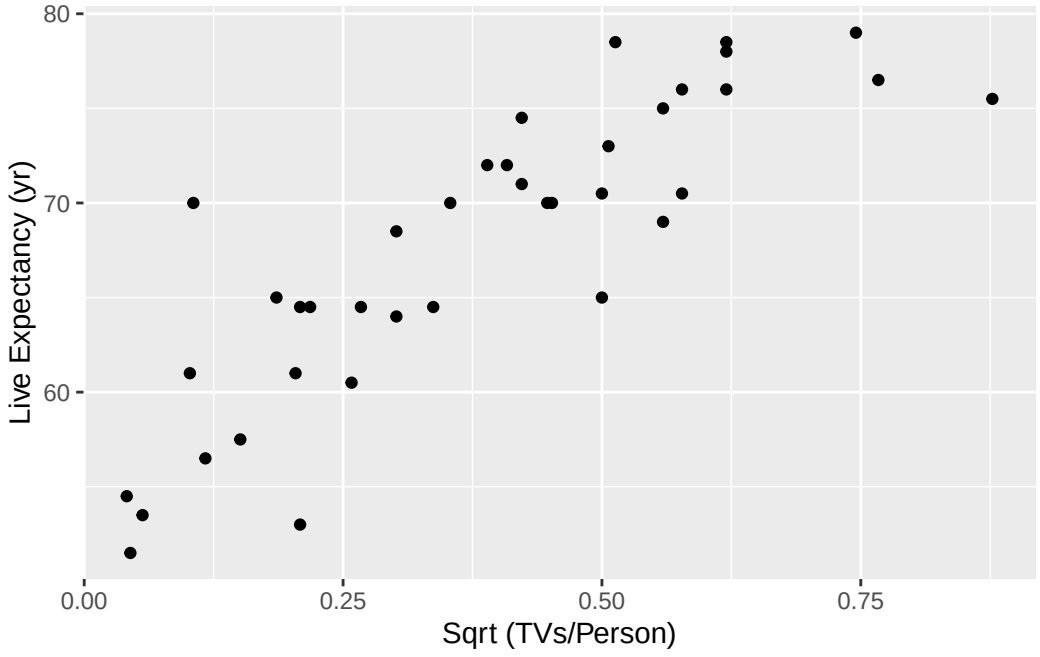
Section 8.4: Lurking Variables with Causation

The Doctors data can help with thinking in a multivariate way.

```
Doctors <-
  read_csv("http://nhorton.people.amherst.edu/is5/data/Doctors_and_life_expectancy.csv") |>
  janitor::clean_names()
# Figure 8.13, page 243
gf_point(life_exp ~ sqrt_doctors_person, data = Doctors) |>
  gf_labs(x = "Sqrt (Doctors/Person)", y = "Life Expectancy (yr)")
```



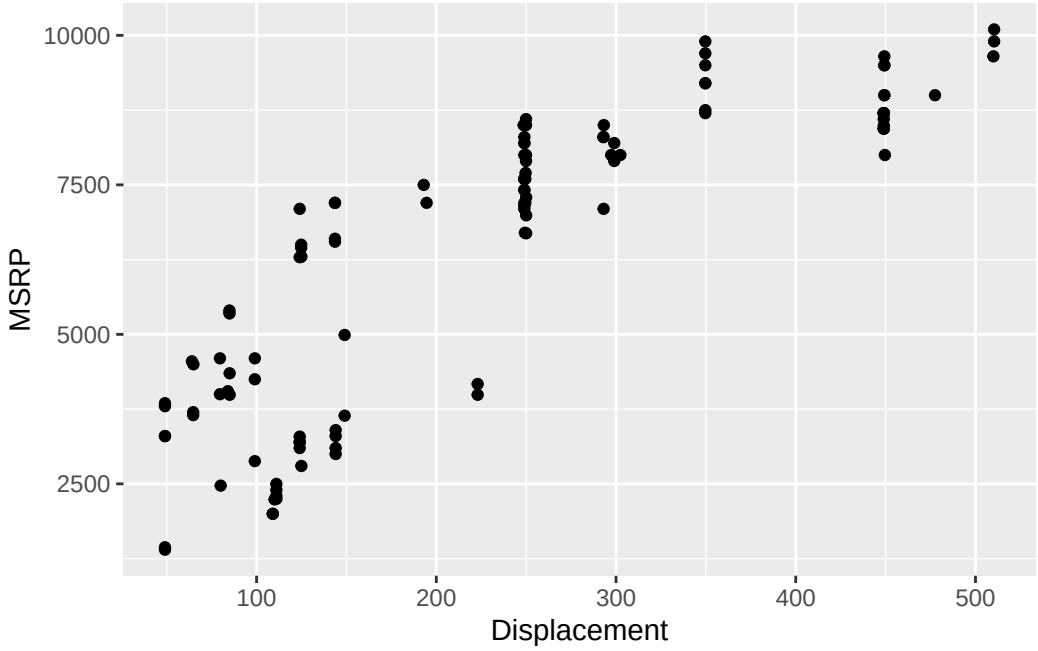

```
# Figure 8.14
gf_point(life_exp ~ sqrt_tv_person, data = Doctors) |>
  gf_labs(x = "Sqrt (TVs/Person)", y = "Live Expectancy (yr)")
```



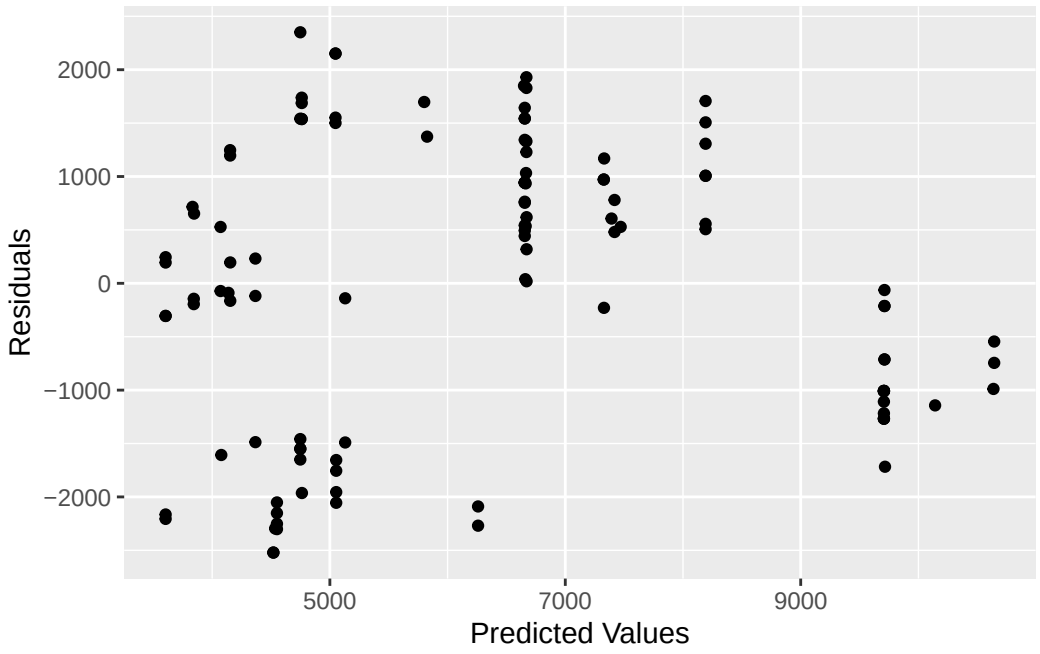
Example 8.2: Using Several of These Methods Together

We can use these approaches together, as seen in the following example.

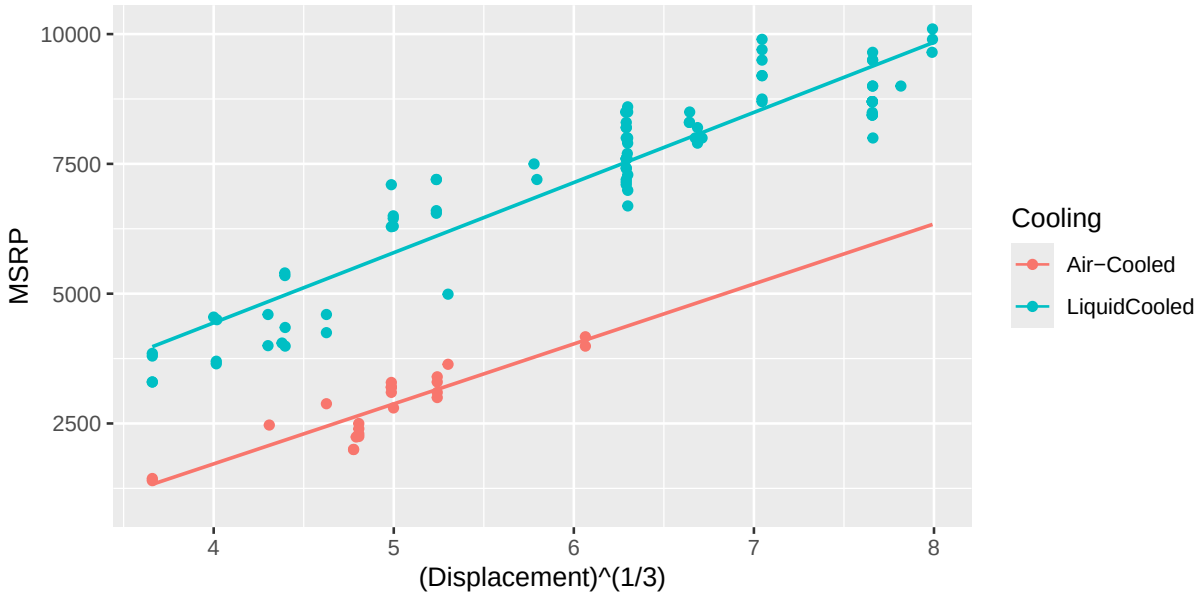
```
# page 244
DirtBikes <- read_csv("http://nhorton.people.amherst.edu/is5/data/Dirt_bikes_2014.csv")
gf_point(MSRP ~ Displacement, data = DirtBikes)
```




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



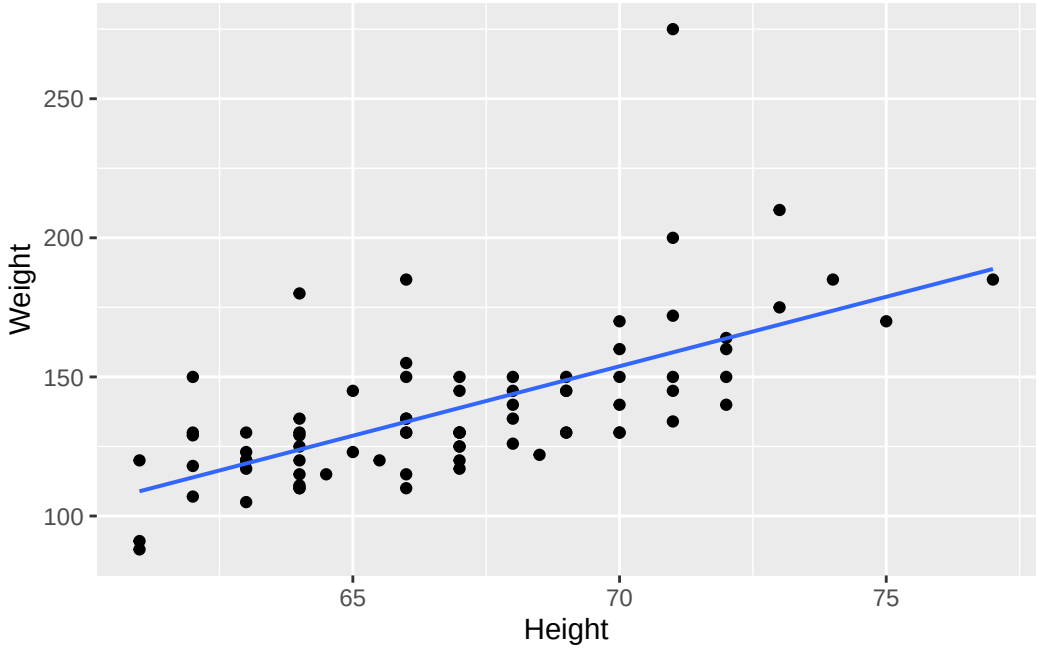

```
DirtBikes |>
  filter(Cooling != "NA") |>
  mutate(Cooling = ifelse(Cooling == "Air-Cooled", "Air-Cooled", "LiquidCooled")) |>
  gf_point(MSRP ~ (Displacement)^(1 / 3), color = ~Cooling) |>
  gf_lm()
```



Section 8.5: Working with Summary Values

We can also work with summary values.

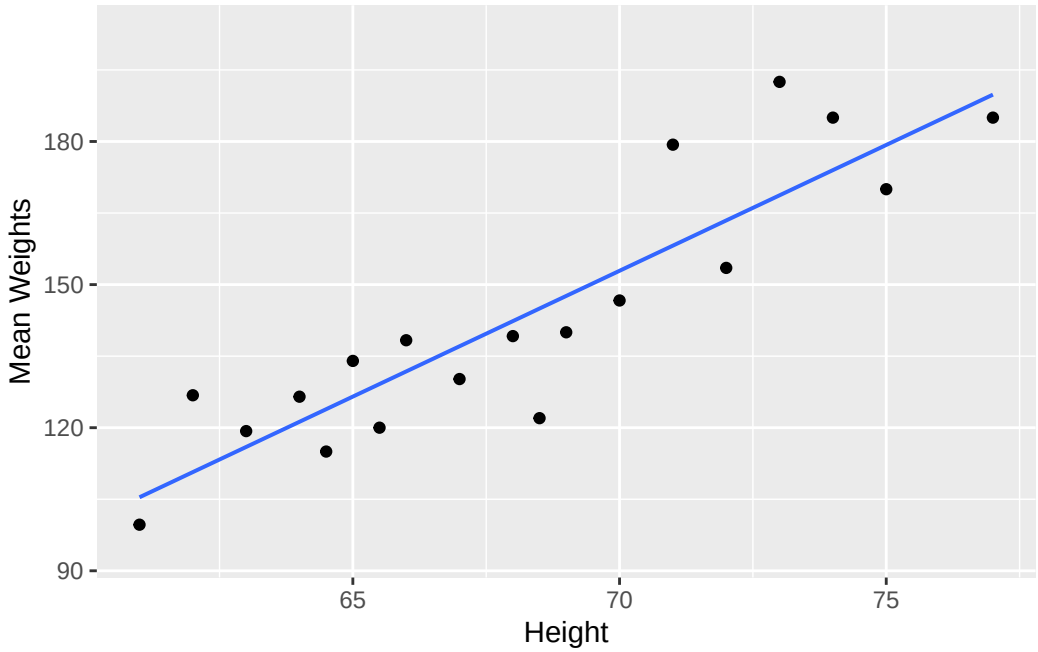
```
HeightsWeights <-
  read_csv("http://nhorton.people.amherst.edu/is5/data/Heights_and_weights.csv")
# Figure 8.15, page 246
gf_point(Weight ~ Height, data = HeightsWeights) |>
  gf_lm()
```




```
# Figure 8.16
HeightWeightSum <- df_stats(Weight ~ Height, data = HeightsWeights)
head(HeightWeightSum)
```

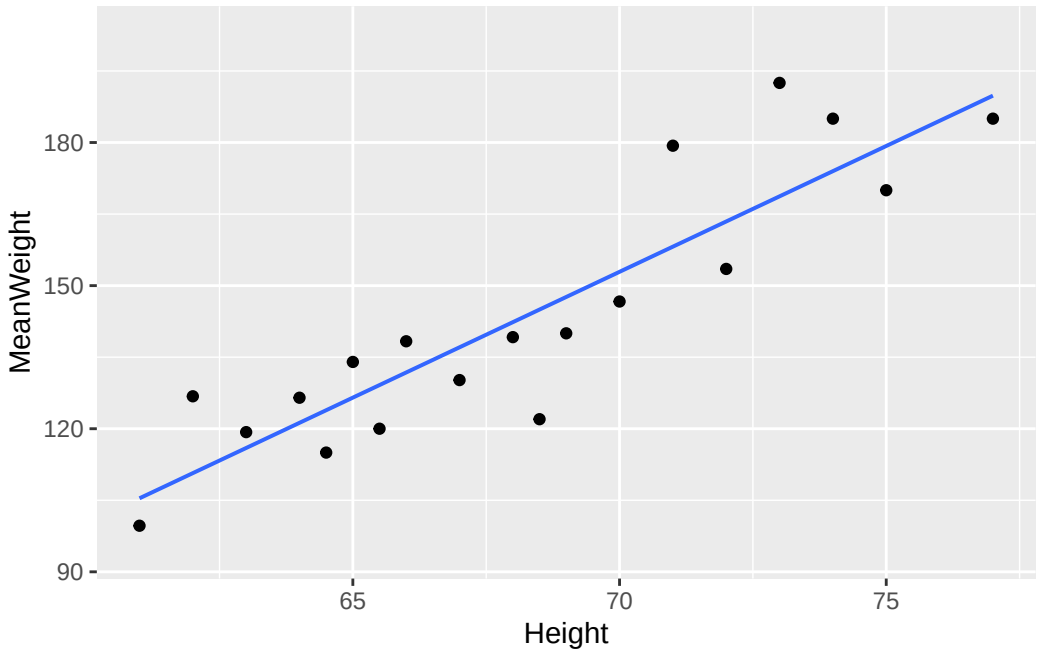
```
  response Height min    Q1 median     Q3 max      mean        sd  n missing
1   Weight   61.0  88  89.5   91.0 105.50 120  99.66667 17.672955  3       0
2   Weight   62.0 107 118.0  129.0 130.00 150 126.80000 15.990622  5       0
3   Weight   63.0 105 118.5  120.0 121.50 130 119.28571  7.521398  7       0
4   Weight   64.0 110 112.0  122.5 129.75 180 126.50000 20.855322 10       0
5   Weight   64.5 115 115.0  115.0 115.00 115 115.00000        NA  1       0
6   Weight   65.0 123 128.5  134.0 139.50 145 134.00000 15.556349  2       0
```

```
gf_point(mean ~ Height, data = HeightWeightSum) |>
  gf_lm() |>
  gf_labs(x = "Height", y = "Mean Weights")
```



Alternately, we can use `group_by()` and `summarise()` together to find summary values of data:

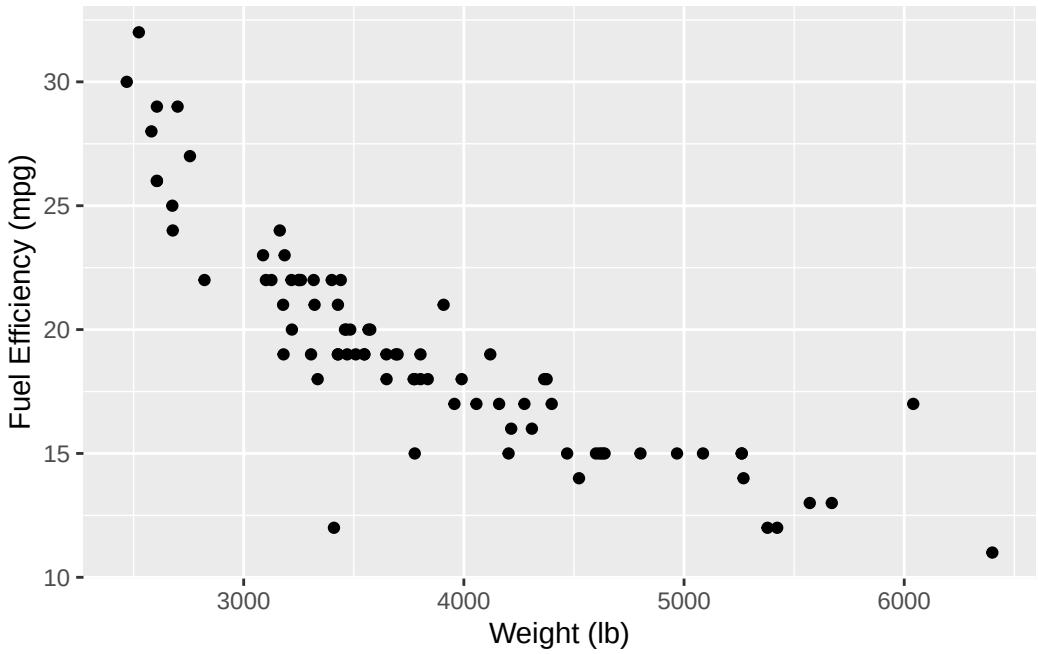
```
HeightsWeights |>
  group_by(Height) |>
  summarise(MeanWeight = mean(Weight), .groups = "drop") |>
  gf_point(MeanWeight ~ Height) |>
  gf_lm()
```



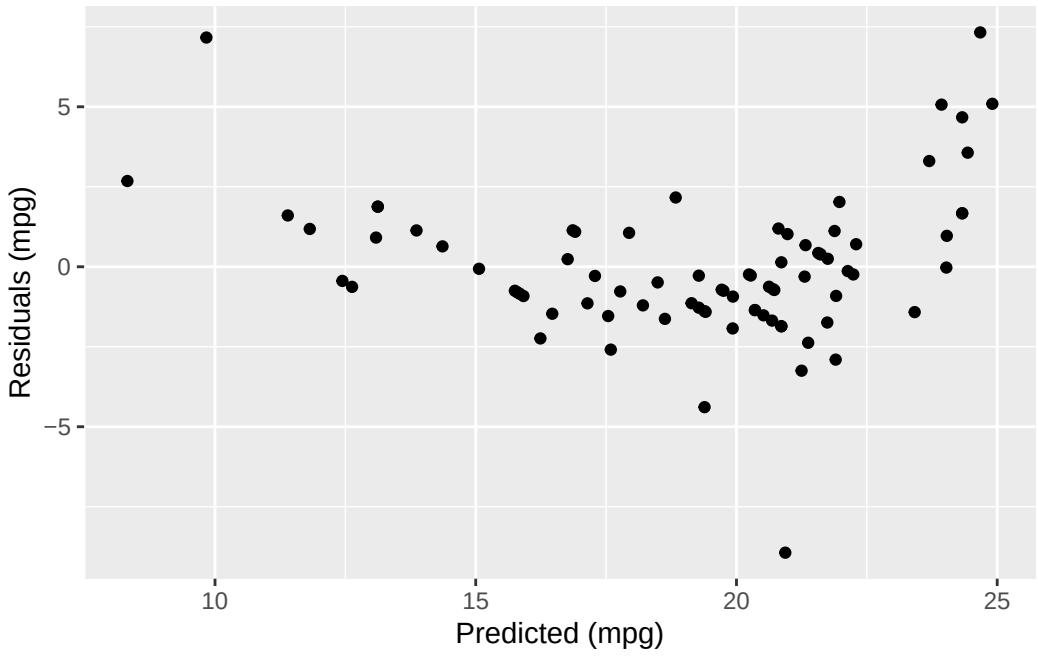
Section 8.6: Straightening Scatterplots–The Three Goals

We begin by reading in the data.

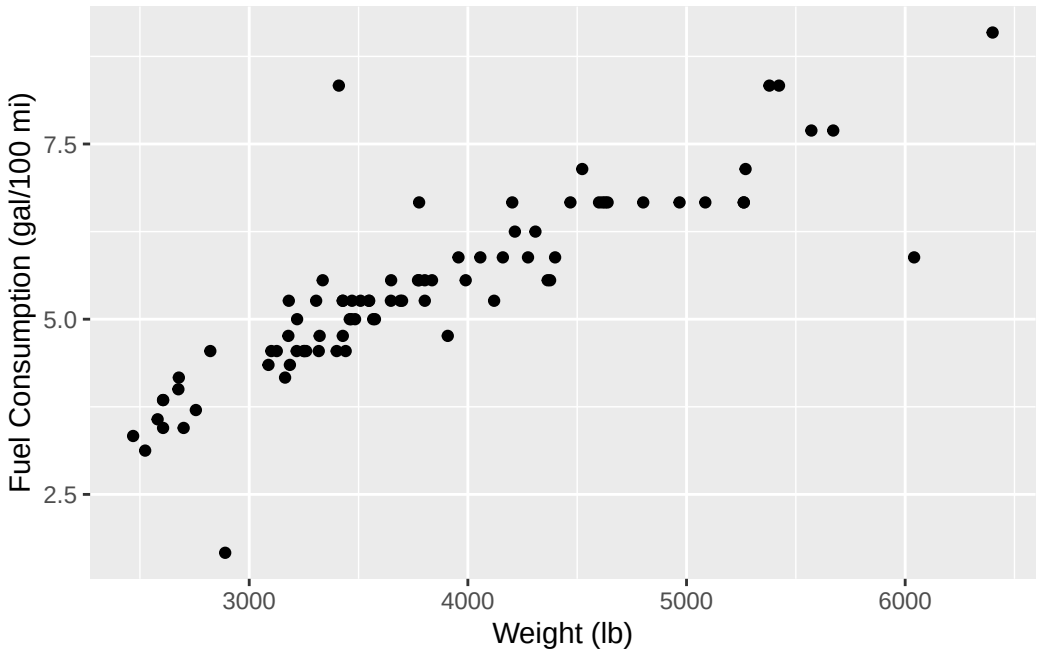
```
FuelEfficiency <-
  read_csv("http://nhorton.people.amherst.edu/is5/data/Fuel_efficiency.csv") |>
  janitor::clean_names()
# Figure 8.17
gf_point(city_mpg ~ weight, data = filter(FuelEfficiency, city_mpg <= 40)) |>
  gf_labs(x = "Weight (lb)", y = "Fuel Efficiency (mpg)")
```



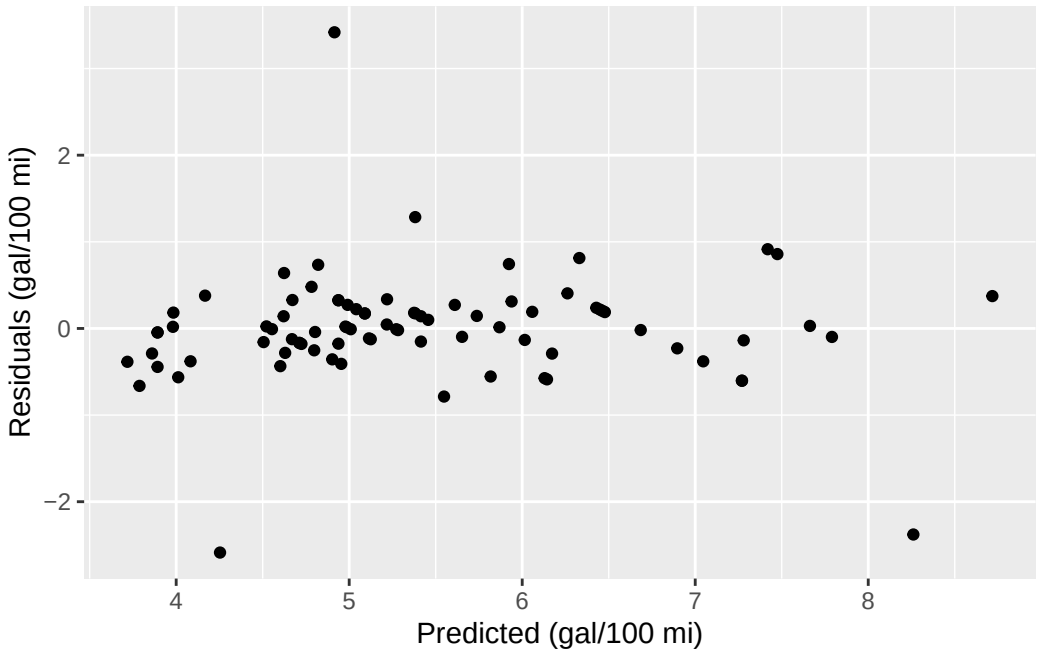

```
fuellm <- lm(city_mpg ~ weight, data = filter(FuelEfficiency, city_mpg <= 40))
gf_point(resid(fuellm) ~ fitted(fuellm)) |>
  gf_labs(x = "Predicted (mpg)", y = "Residuals (mpg)")
```



```
FuelEfficiency <- FuelEfficiency |>
  mutate(fuel_consumption = (1 / city_mpg) * 100)
# Figure 8.19, page 247
gf_point(fuel_consumption ~ weight, data = FuelEfficiency) |>
  gf_labs(x = "Weight (lb)", y = "Fuel Consumption (gal/100 mi)")
```



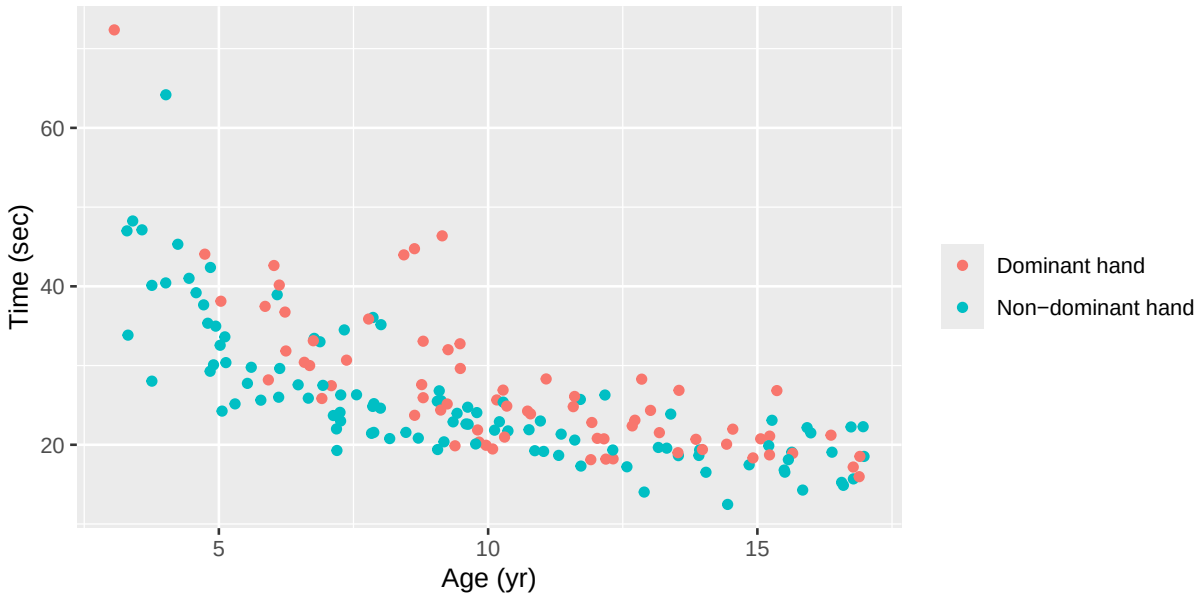

```
fuellm2 <- lm(fuel_consumption ~ weight, data = FuelEfficiency)
gf_point(resid(fuellm2) ~ fitted(fuellm2)) |>
  gf_labs(x = "Predicted (gal/100 mi)", y = "Residuals (gal/100 mi)")
```



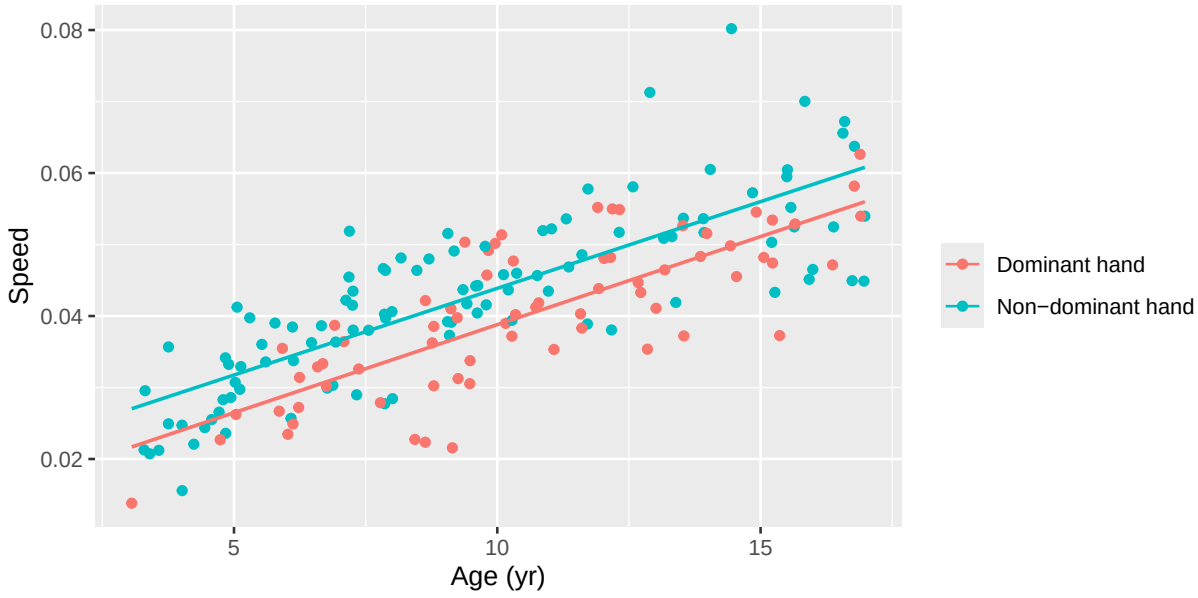
Goals of Re-Expression for Regression

We use the hand dexterity data to illustrate re-expressions.

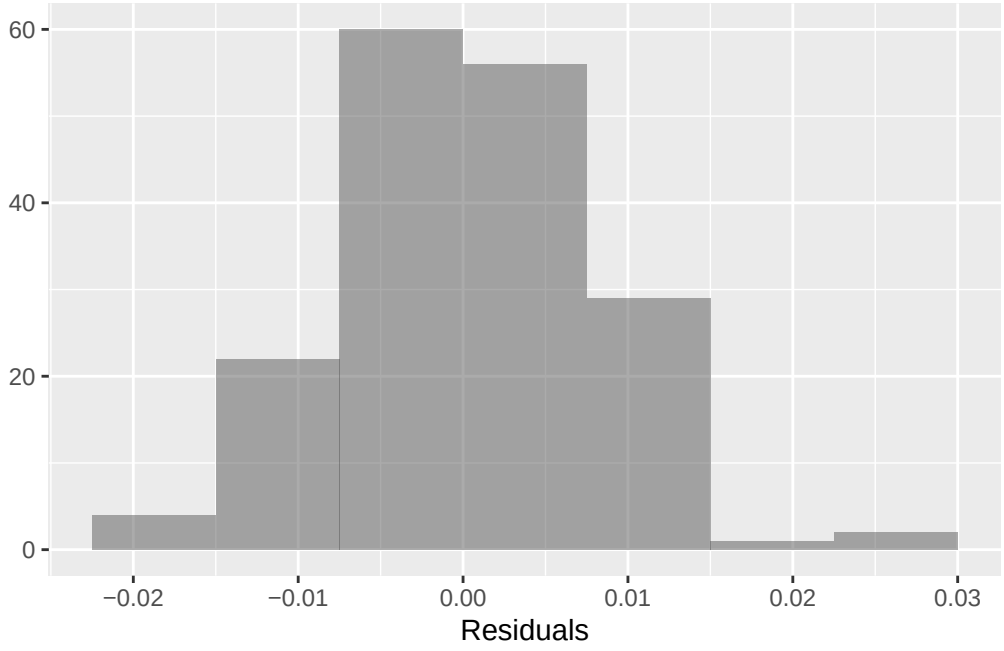
```
HandDexterity <-
  read_csv("http://nhorton.people.amherst.edu/is5/data/Hand_dexterity.csv") |>
  janitor::clean_names() |>
  mutate(dominant = ifelse(dominant == 0, "Dominant hand", "Non-dominant hand")) |>
  mutate(dominant = as.factor(dominant))
# Figure 8.20, page 248
gf_point(time_sec ~ age_yr, color = ~ dominant, data = HandDexterity) |>
  gf_labs(x = "Age (yr)", y = "Time (sec)", color = "")
```




```
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", color = "")
```



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



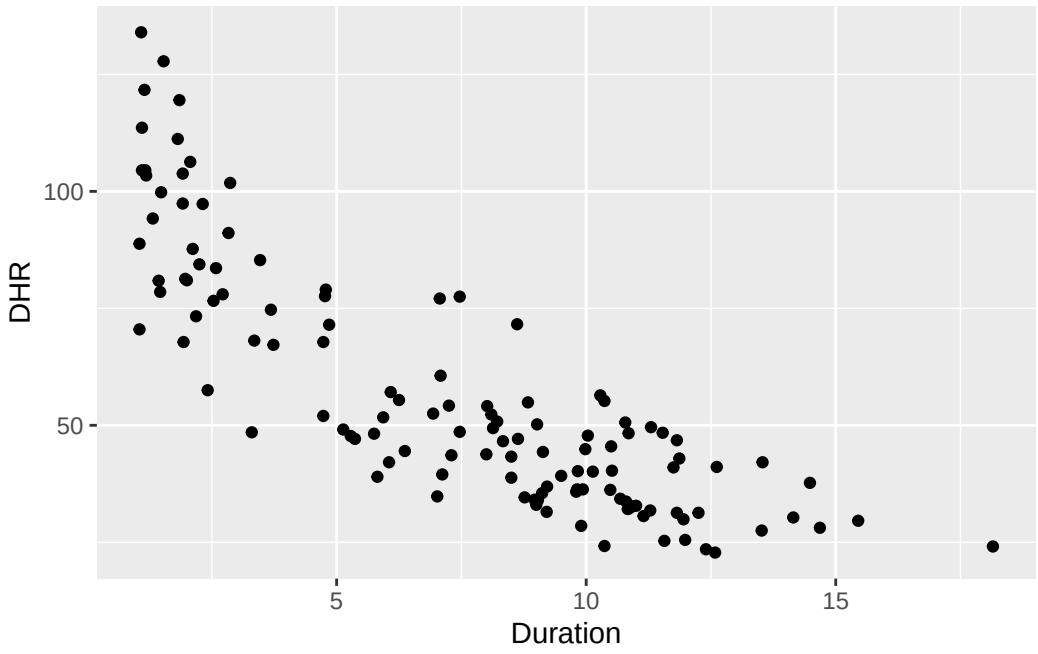
Section 8.7: Finding a Good Re-expression

See the table and Figure 8.23 on page 250.

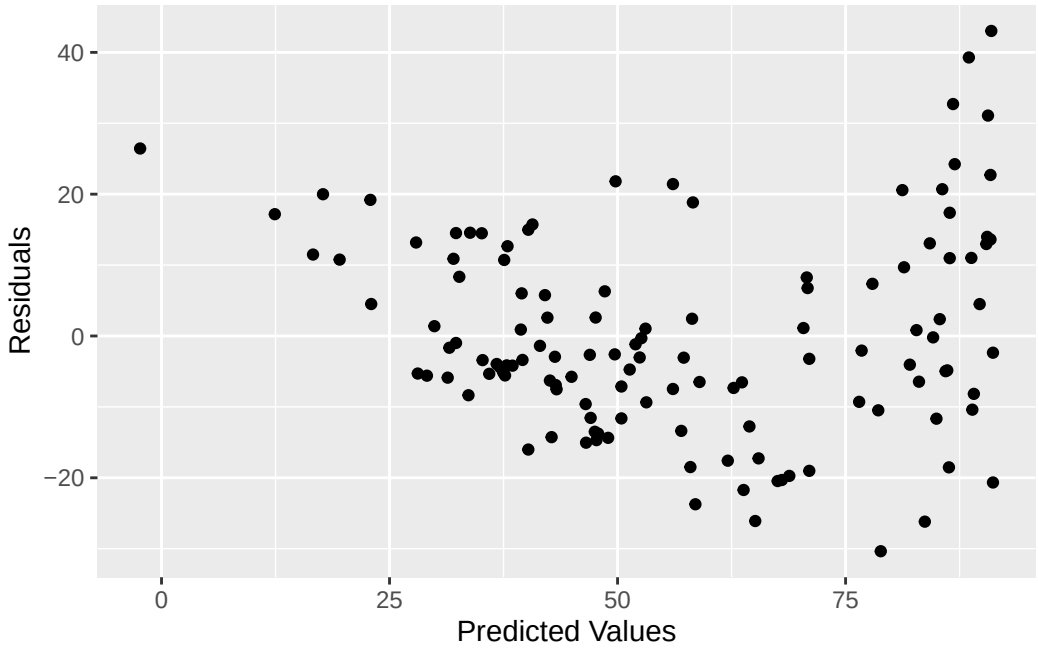
Step-By-Step Example: Re-Expressing to Straighten a Scatterplot

We can explore different re-expressions.

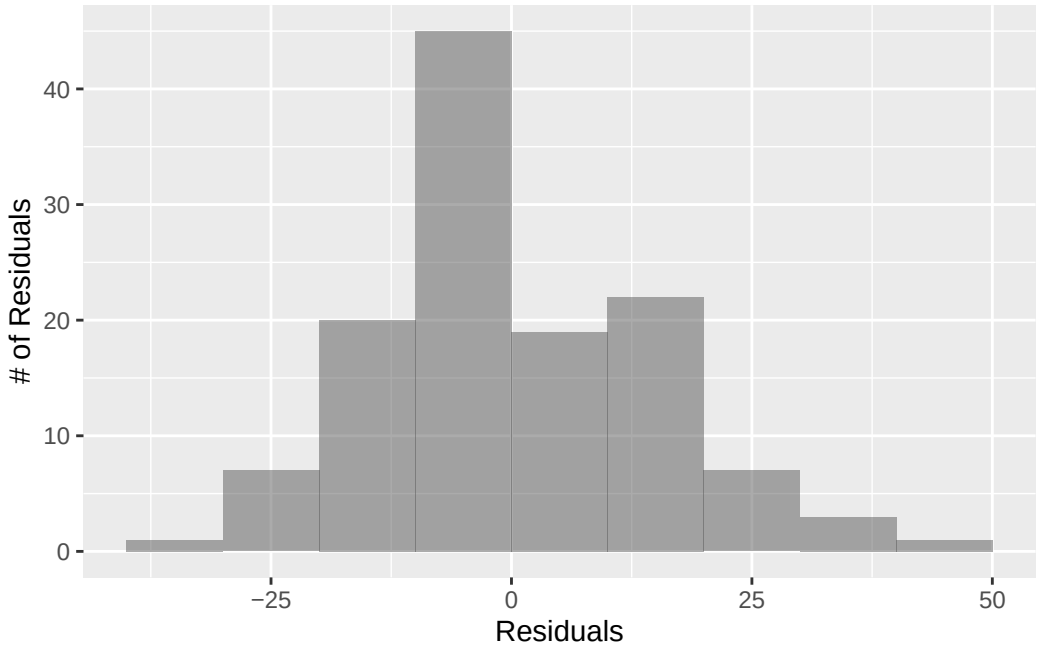
```
gf_point(dive_heart_rate ~ duration_min, data = Penguins, xlab = "Duration", ylab = "DHR")
```




```
gf_point(resid(penguinlm) ~ fitted(penguinlm), data = Penguins) |>
  gf_labs(x = "Predicted Values", y = "Residuals")
```

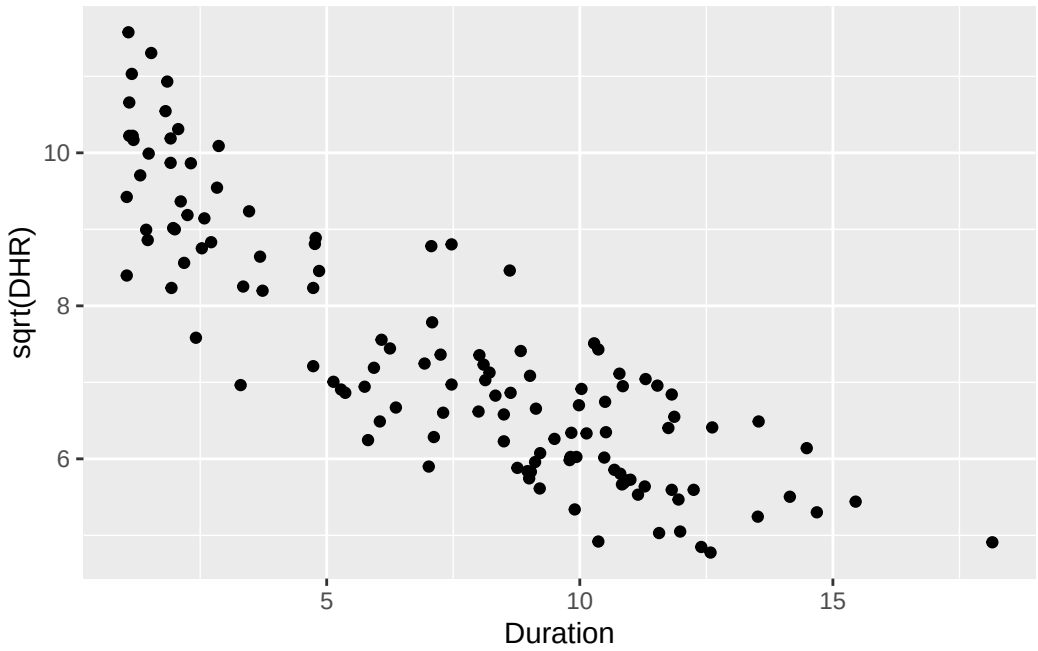


```
gf_histogram(~ resid(penguinlm), binwidth = 10, center = 5) |>
  gf_labs(x = "Residuals", y = "# of Residuals")
```



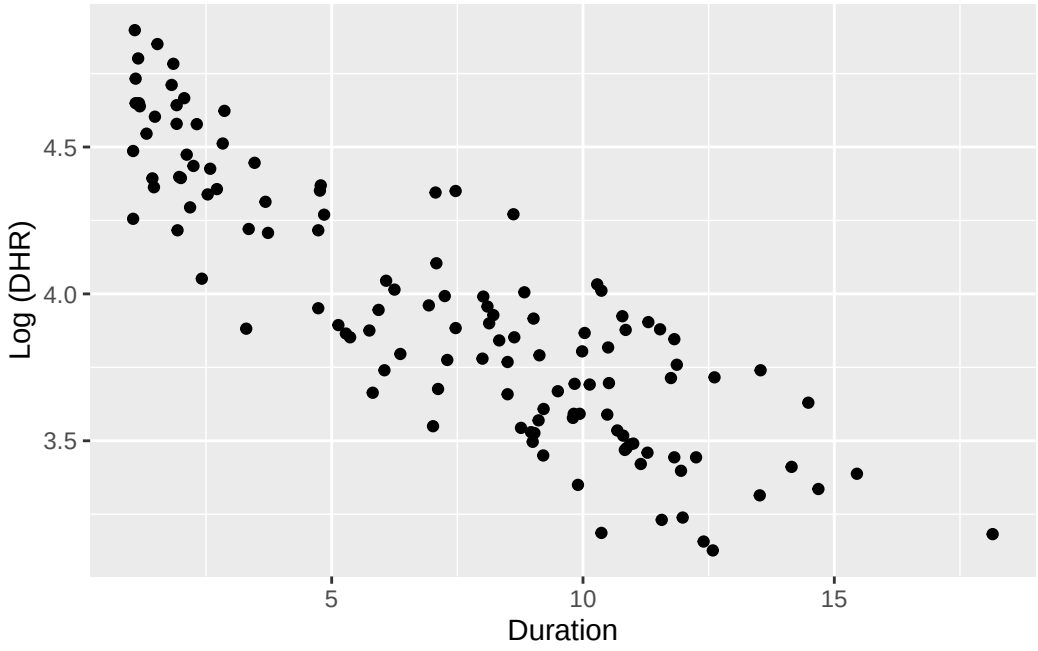
Mutating with the square root:

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

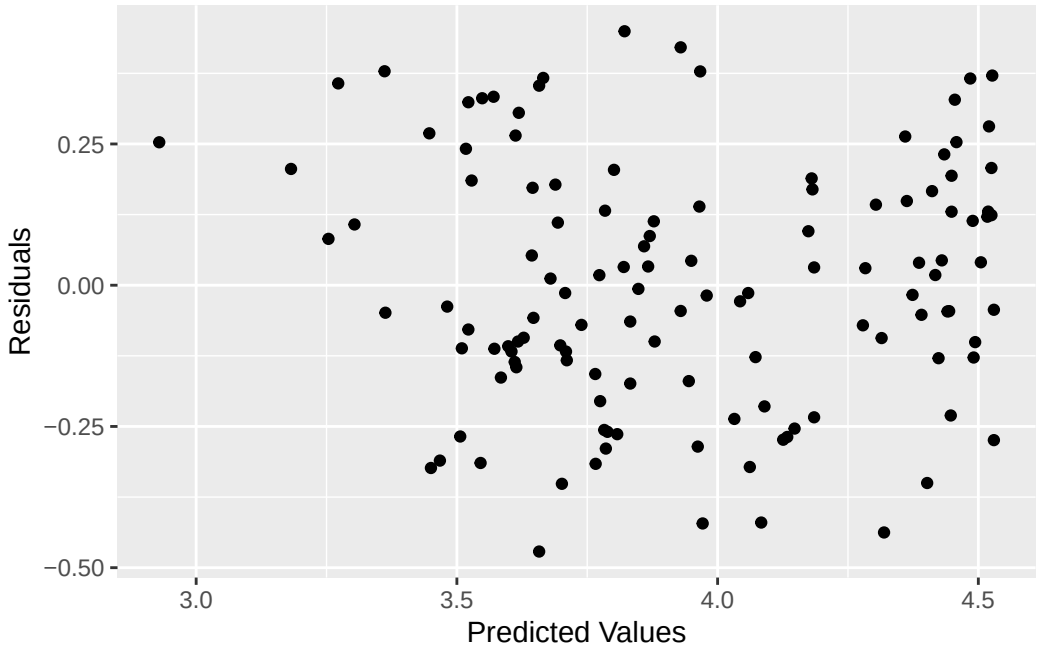


Mutating with a log:

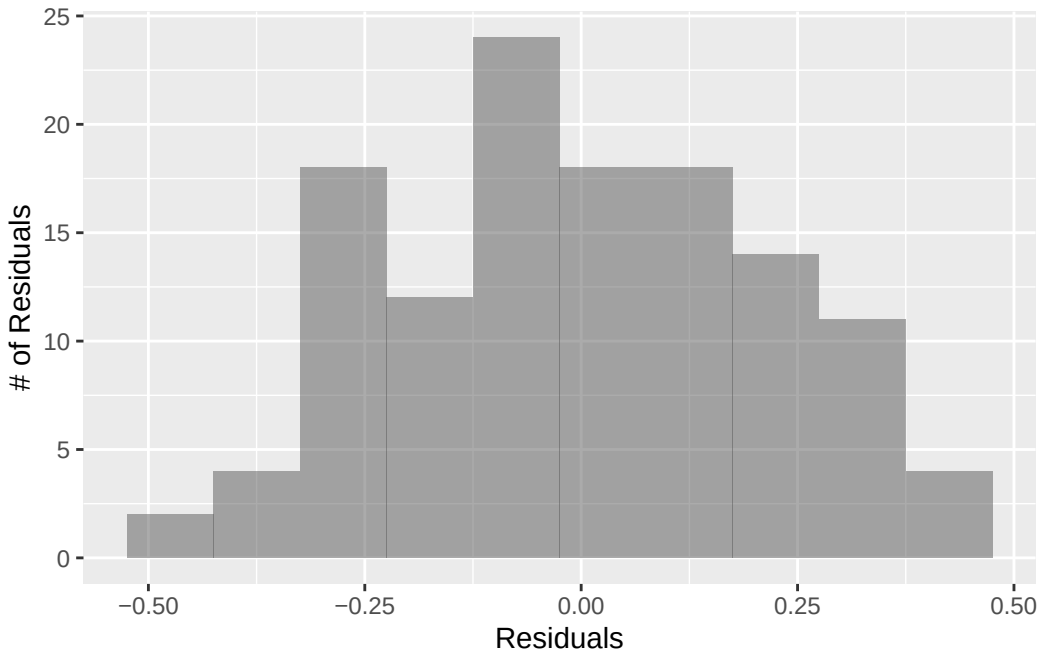
```
gf_point(log(dive_heart_rate) ~ duration_min, data = Penguins) |>
  gf_labs(x = "Duration", y = "Log (DHR)")
```



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

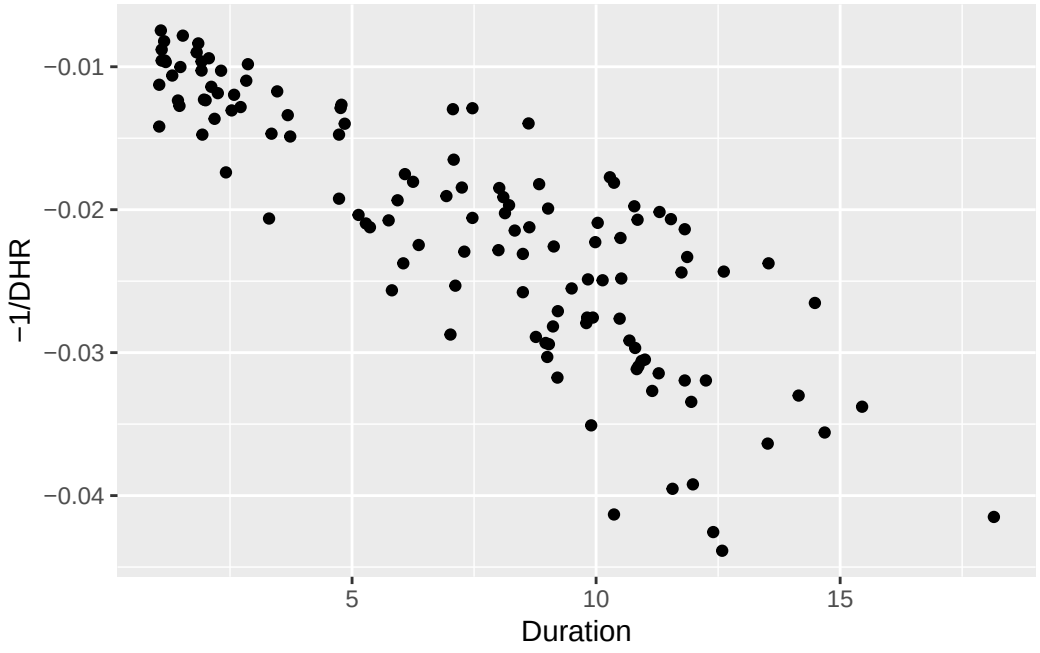



```r
gf_histogram(~ resid(penguinlm2), binwidth = 0.1, center = .025) |>
  gf_labs(x = "Residuals", y = "# of Residuals")
```

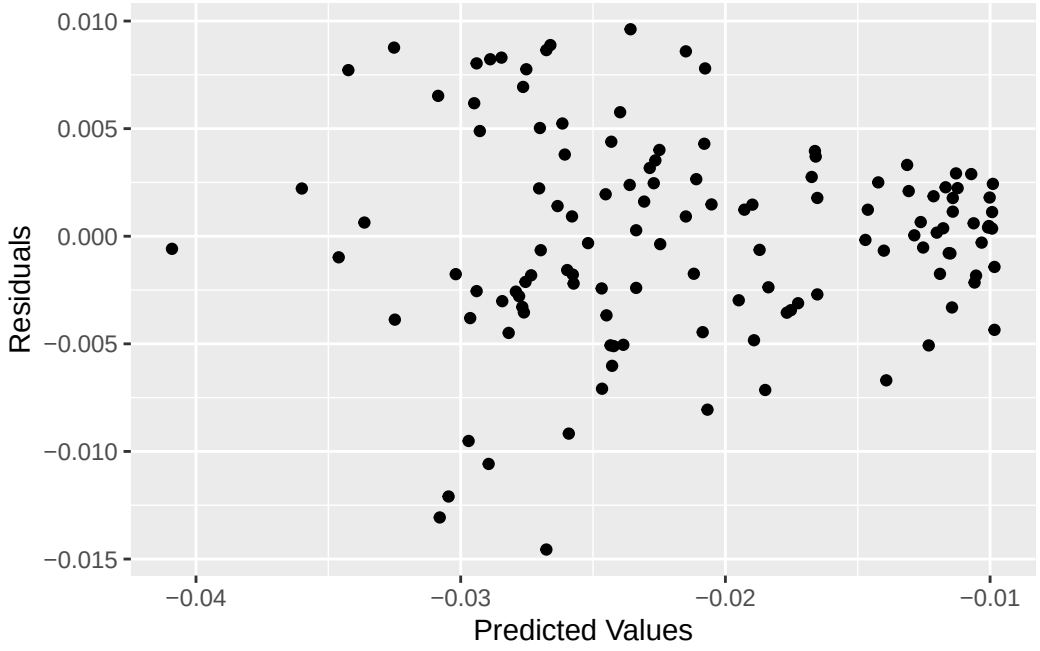


Mutating with a (negative) reciprocal:

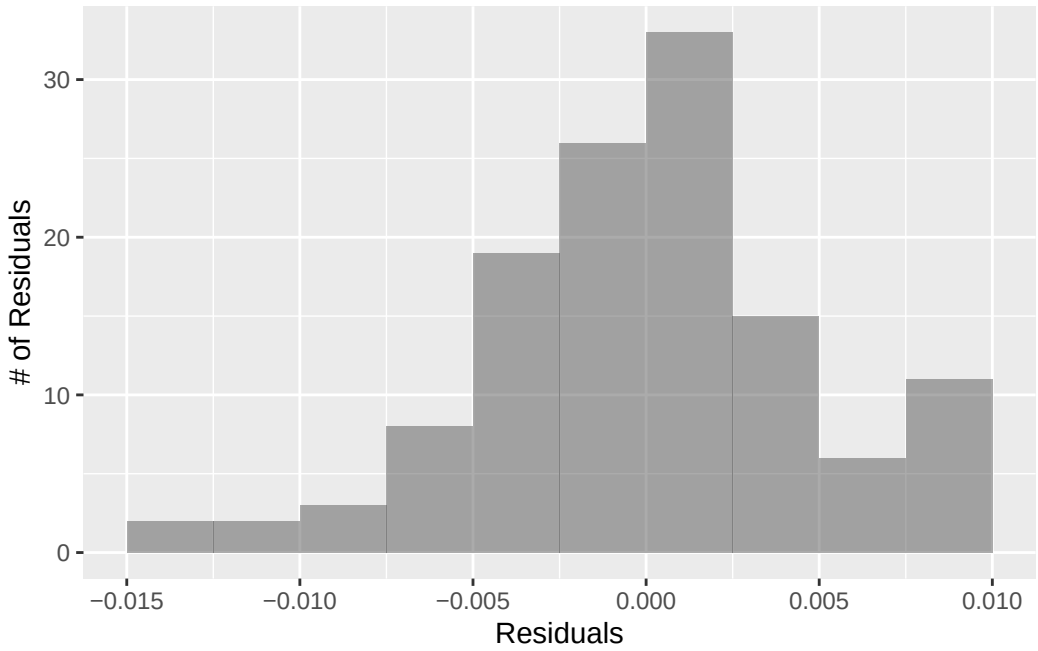
```
gf_point(-1 / (dive_heart_rate) ~ duration_min, data = Penguins) |>
  gf_labs(x = "Duration", y = "-1/DHR")
```



```
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, center = 0.00125) |>
  gf_labs(x = "Residuals", y = "# of Residuals")
```



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