IS5 in R: Scatterplots, Association, and Correlation (Chapter 6)

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

July 7, 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.

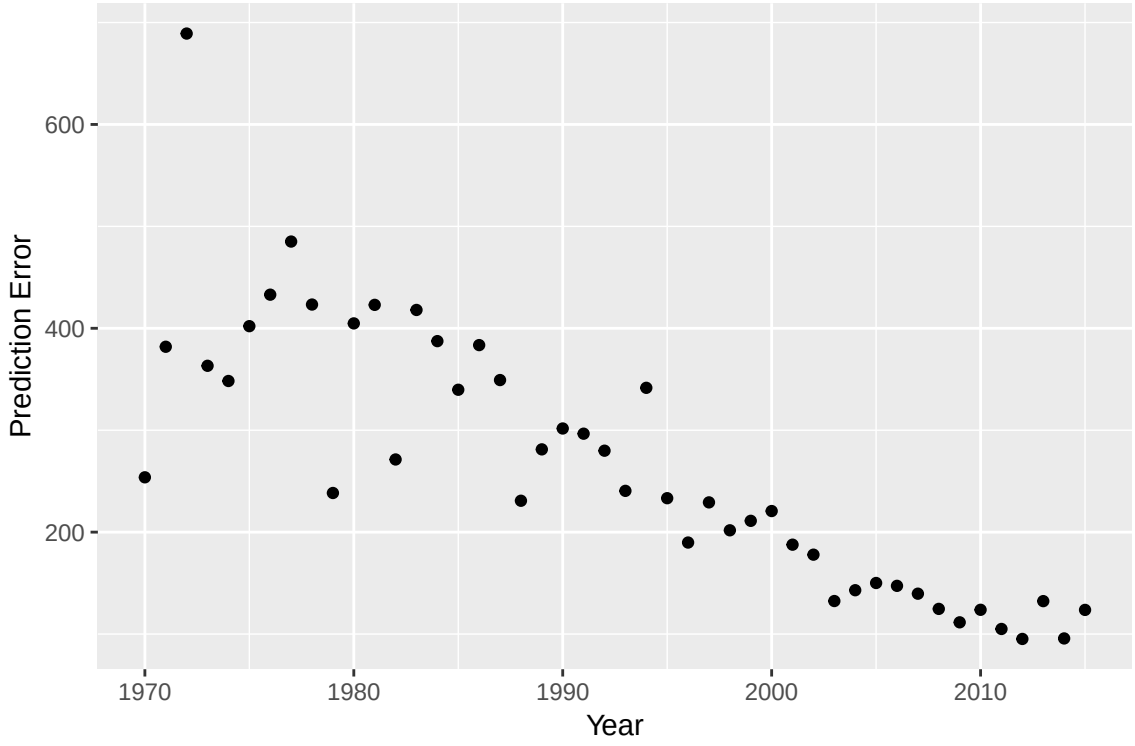
Chapter 6: Scatterplots, Association, and Correlation

```
library(mosaic)
library(readr)
library(janitor)
Hurricanes <- read_csv("http://nhorton.people.amherst.edu/is5/data/Tracking_hurricanes_2015.csv")
```

```
## Parsed with column specification:
## cols(
##   Year = col_integer(),
##   Error_24h = col_double(),
##   Error_48h = col_double(),
##   Error_72h = col_double()
## )
```

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.

```
# Figure 6.1, page 164
gf_point(Error_72h ~ Year, data = Hurricanes, ylab = "Prediction Error")
```



Section 6.1: Scatterplots

See dots on pages 164-165.

Example 6.1: Comparing Prices Worldwide

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

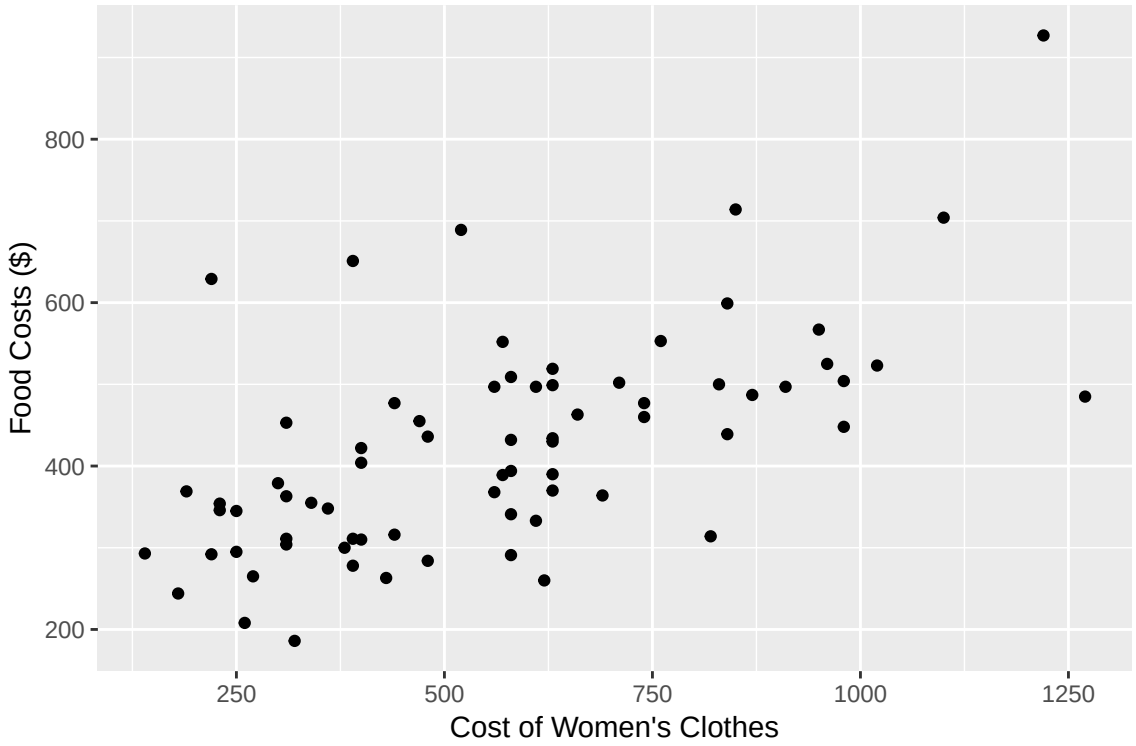
```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   City = col_character(),
##   `Food Costs($)` = col_integer(),
##   `Womens Clothing($)` = col_integer(),
##   `Mens Clothing($)` = col_integer(),
##   `Hours Worked` = col_integer(),
##   `Vacation Days` = col_integer(),
##   `Big Mac(min)` = col_integer(),
##   `Bread(kg in min)` = col_integer(),
##   `Rice(kg in min)` = col_integer(),
##   `Goods and Services($)` = col_integer()
## )

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

```
names(Prices)
```

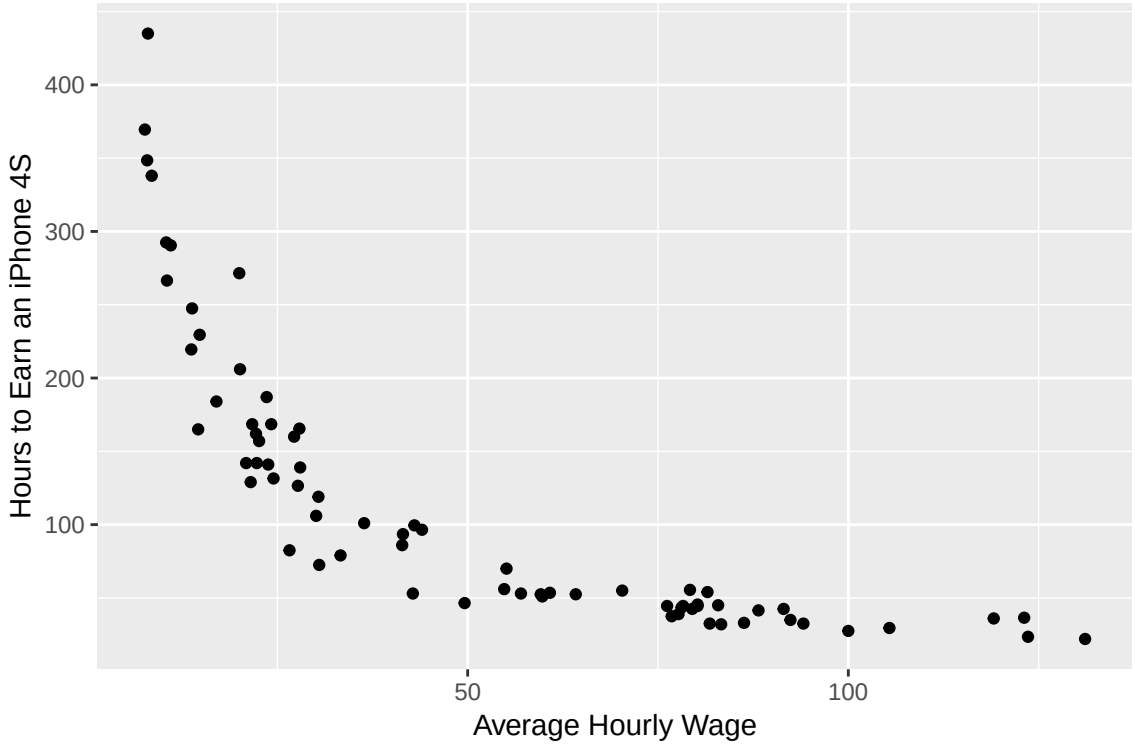
```
##  [1] "city"                    "food_costs"
##  [3] "womens_clothing"         "mens_clothing"
##  [5] "i_phone_4s_hr"           "clothing_index"
##  [7] "hours_worked"            "wage_gross"
##  [9] "wage_net"                "vacation_days"
## [11] "col_excl_rent"           "col_incl_rent"
## [13] "pur_power_gross"         "pur_power_net"
## [15] "pur_power_annual"        "big_mac_min"
## [17] "bread_kg_in_min"         "rice_kg_in_min"
## [19] "goods_and_services"      "good_and_services_index"
## [21] "food_index"
```

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).

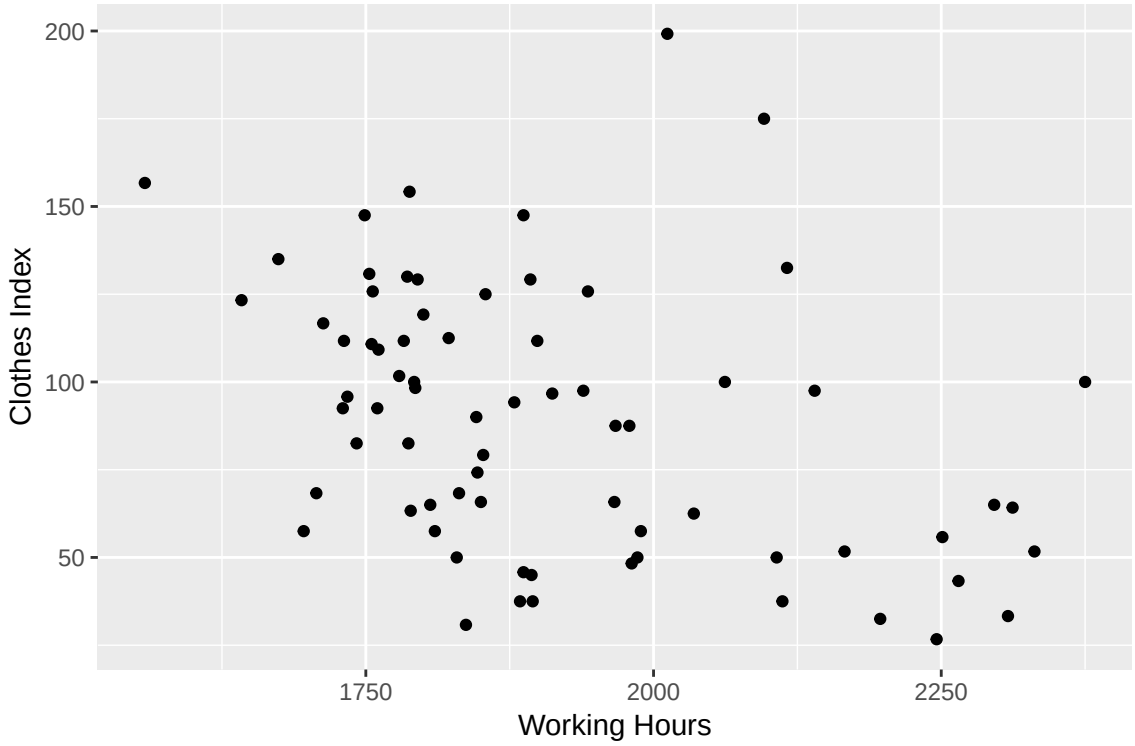
```
gf_point(food_costs ~ womens_clothing, data = Prices, xlab = "Cost of Women's Clothes",
         ylab = "Food Costs ($)")
```



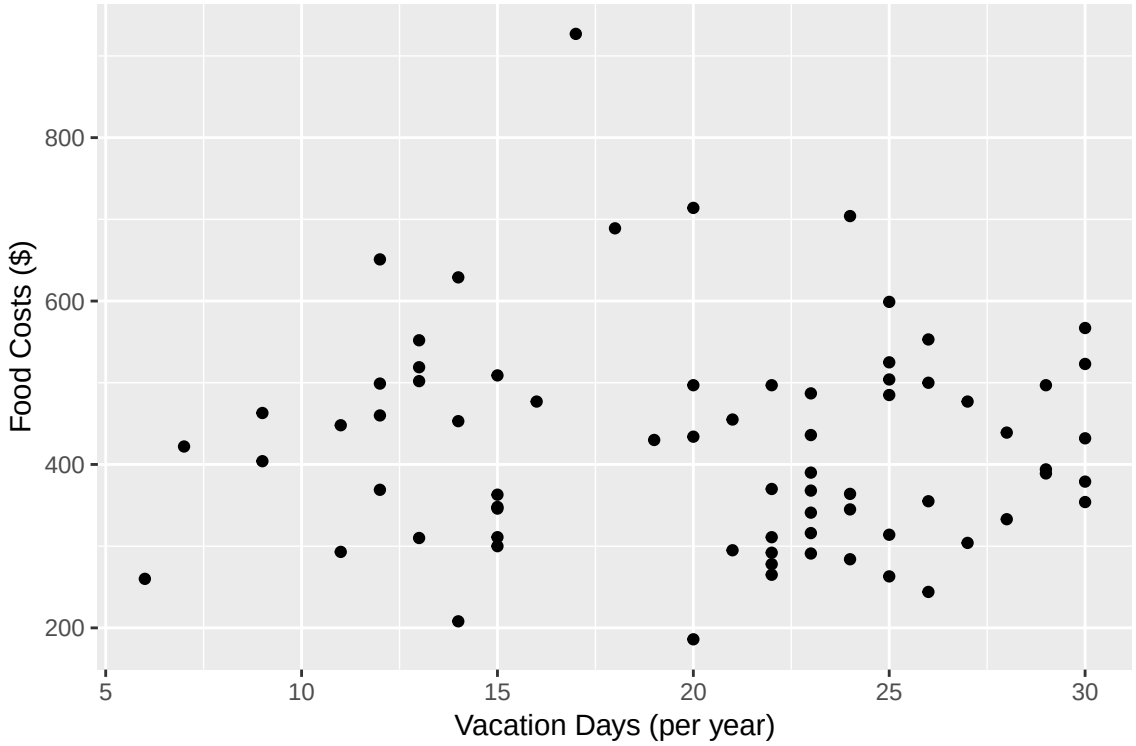

```
gf_point(i_phone_4s_hr ~ wage_gross, data = Prices, xlab = "Average Hourly Wage",
         ylab = "Hours to Earn an iPhone 4S")
```




```
gf_point(clothing_index ~ hours_worked, data = Prices, xlab = "Working Hours",
         ylab = "Clothes Index")
```



```
gf_point(food_costs ~ vacation_days, data = Prices, xlab = "Vacation Days (per year)",
         ylab = "Food Costs ($)")
```



Roles for Variables

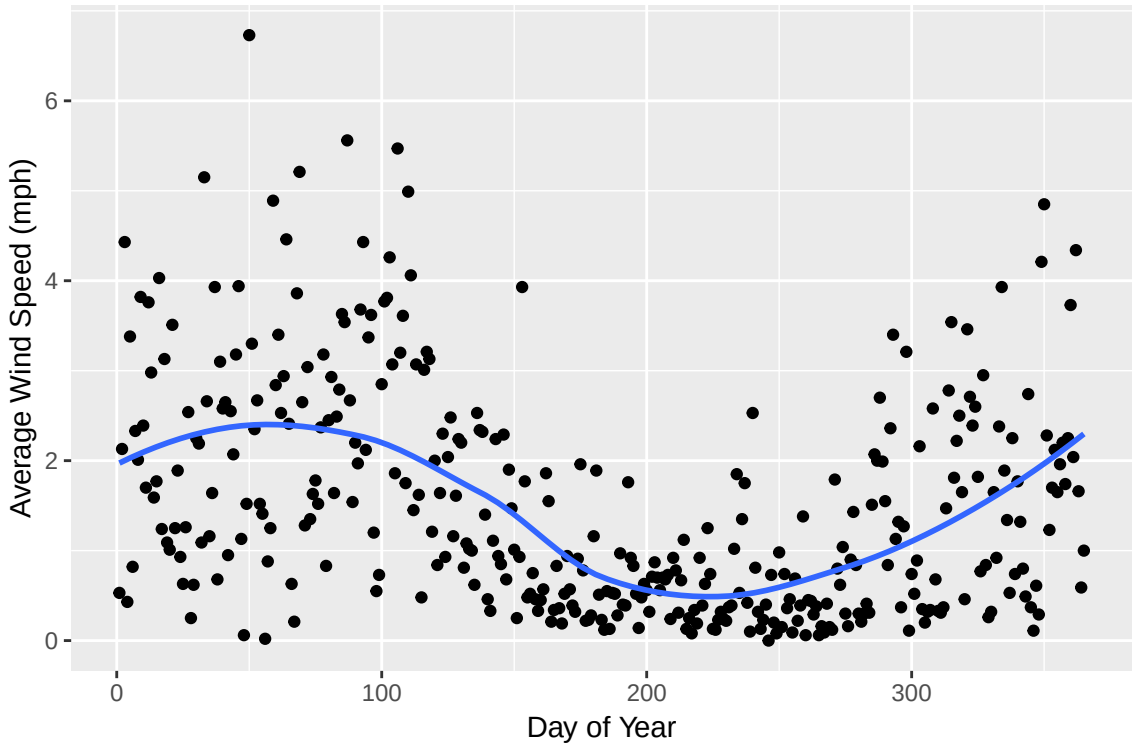
Smoothing Scatterplots

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

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Date = col_character(),
##   Year = col_integer(),
##   Month = col_integer(),
##   Day = col_integer(),
##   `Day of Year` = col_integer(),
##   `Max Sol Rad (w/m^2)` = col_integer(),
##   `Min Sol Rad (w/m^2)` = col_integer(),
##   `Total Sol Rad (w/m^2)` = col_integer(),
##   `Min Wind (mph)` = col_integer(),
##   `Max Barom (mb)` = col_integer(),
##   `Min Barom (mb)` = col_integer()
## )

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

```
# Figure 6.2, page 168
gf_point(avg_wind_mph ~ day_of_year, data = HopkinsForest) %>%
  gf_smooth(se = FALSE, xlab = "Day of Year", ylab = "Average Wind Speed (mph)")
```

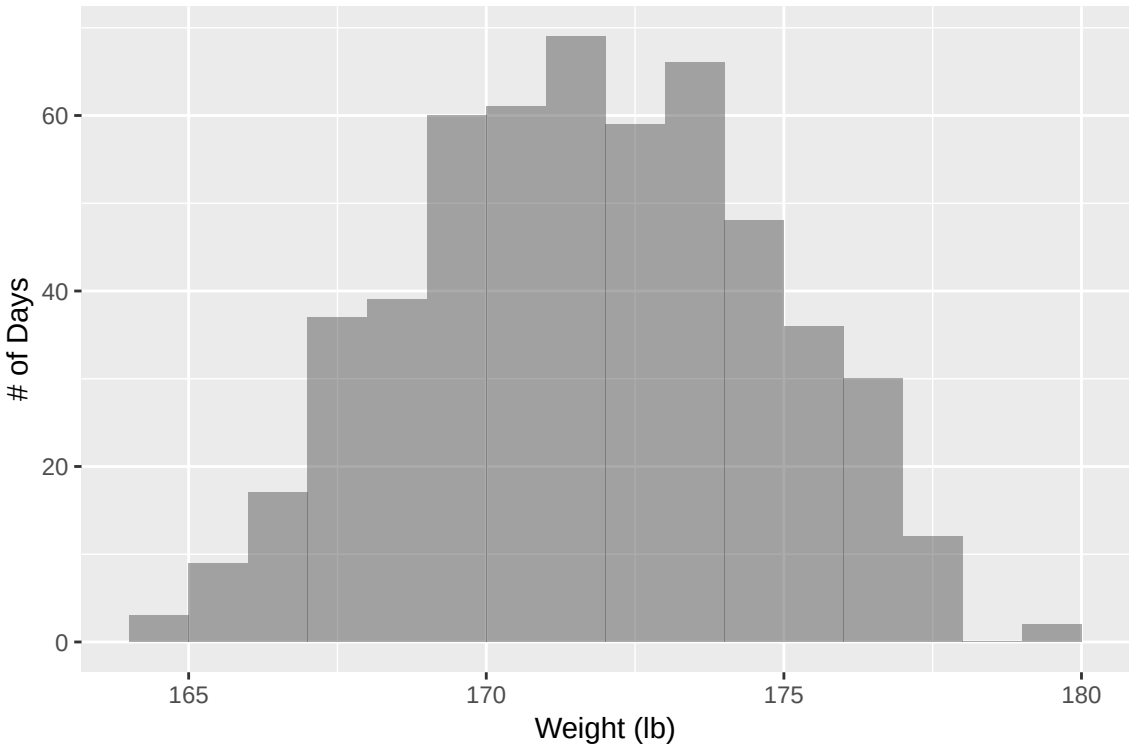
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Example 6.2: Smoothing Timeplots

```
Fitness <- read_csv("http://nhorton.people.amherst.edu/is5/data/Fitness_data.csv") %>%
  clean_names()
gf_histogram(~ weight, data = Fitness, binwidth = 1, ylab = "# of Days", xlab = "Weight (lb)",
             center = .5)
```

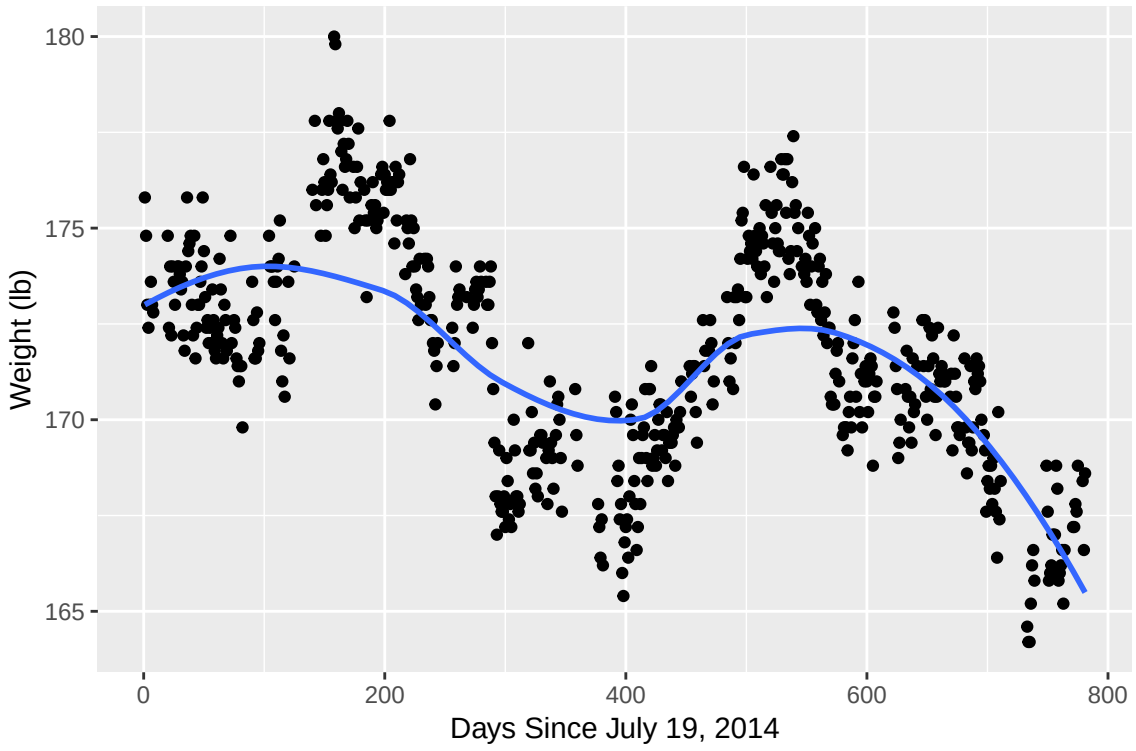
```
## Warning: Removed 70 rows containing non-finite values (stat_bin).
```




```
gf_point(weight ~ days_since_july_19_2014, data = Fitness) %>%
  gf_smooth(se = FALSE, xlab = "Days Since July 19, 2014", ylab = "Weight (lb)")
```

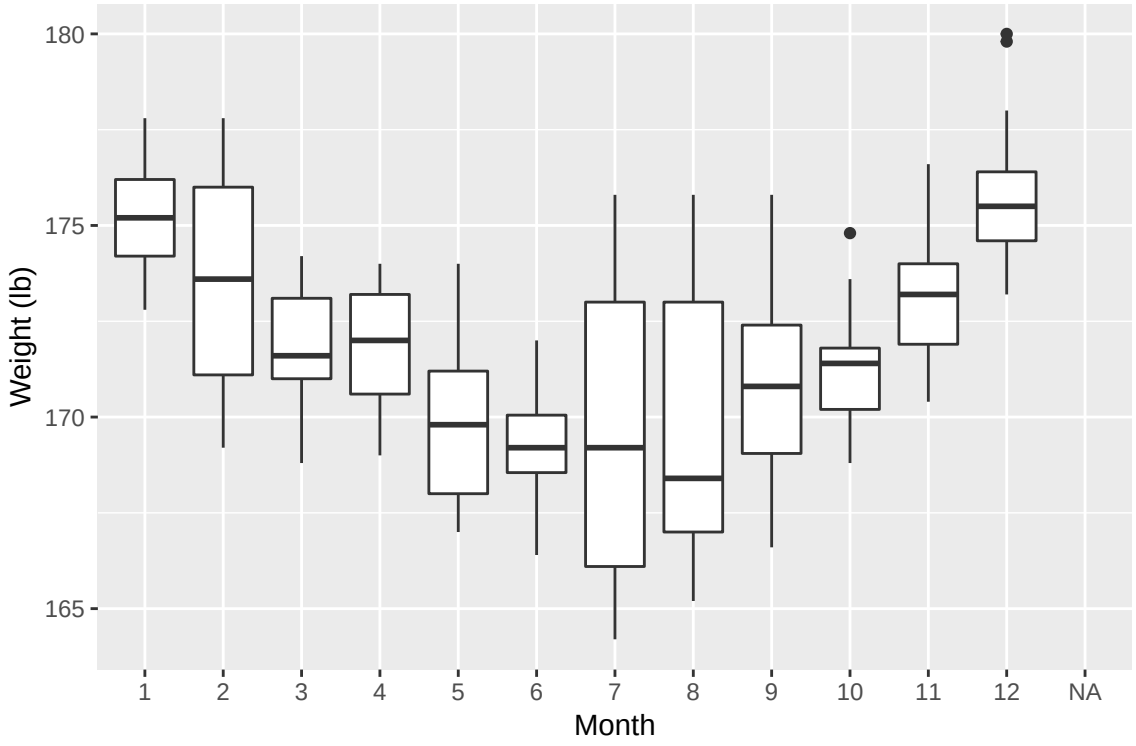
```
## Warning: Removed 70 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 70 rows containing missing values (geom_point).
```




```
gf_boxplot(weight ~ as.factor(month), data = Fitness, xlab = "Month", ylab = "Weight (lb)")
```

```
## Warning: Removed 70 rows containing non-finite values (stat_boxplot).
```



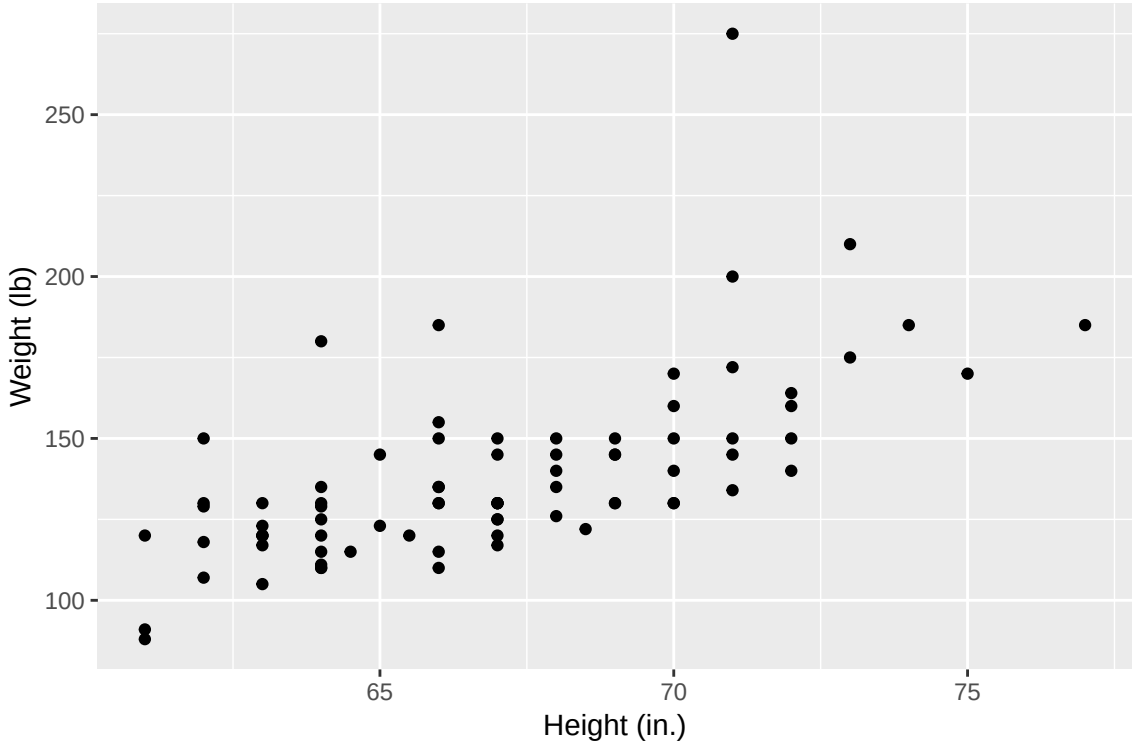
Warnings can be suppressed with the `warnings=FALSE` chunk option.

Section 6.2: Correlation

```
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 6.3, page 170
gf_point(Weight ~ Height, data = HeightsWeights, xlab = "Height (in.)", ylab = "Weight (lb)")
```




```
cor(Weight ~ Height, data = HeightsWeights)
```

```
## [1] 0.6440311
```

See displays on pages 170 - 171.

Step-by-Step Example: Looking at Association

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

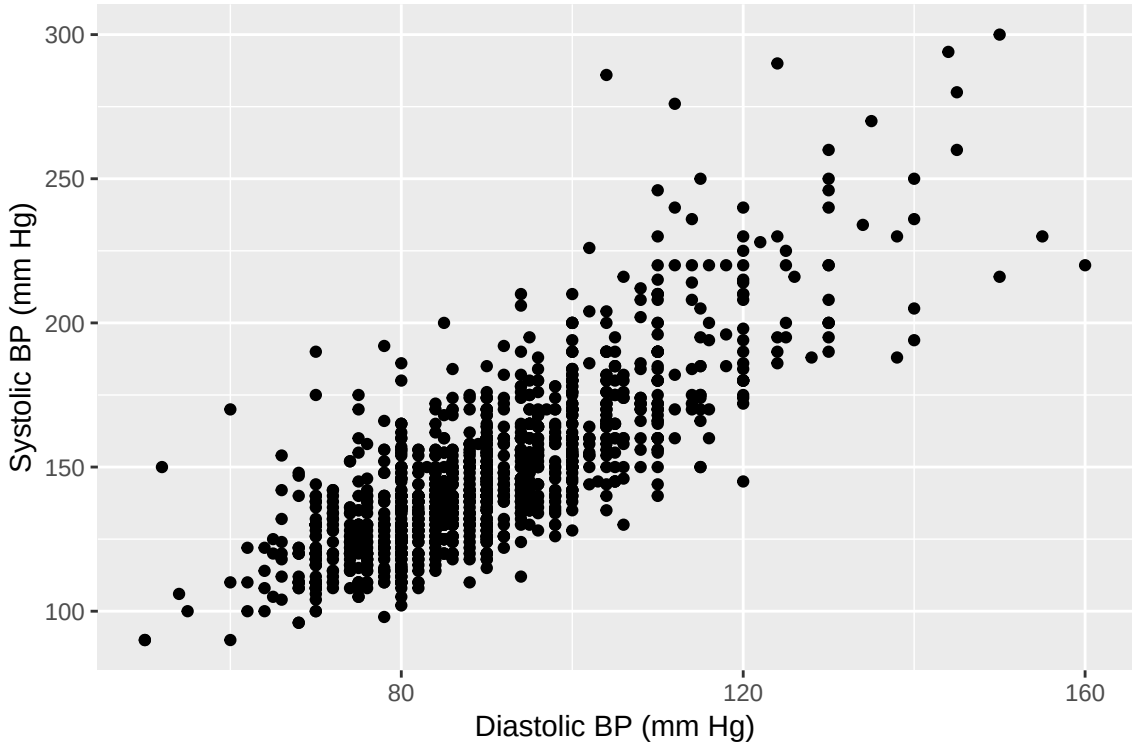
```
## Parsed with column specification:
## cols(
```

```
##   Cholesterol = col_integer(),
##   Age = col_integer(),
##   Sex = col_character(),
##   SBP = col_integer(),
##   DBP = col_integer(),
##   CIG = col_integer()
## )
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 2)
```

```
## Warning: 1 parsing failure.
## row # A tibble: 1 x 5 col     row col   expected   actual file
```

```
gf_point(SBP ~ DBP, data = Framingham, xlab = "Diastolic BP (mm Hg)", ylab = "Systolic BP (mm Hg)")
```




```
cor(SBP ~ DBP, data = Framingham)
```

```
## [1] 0.7924792
```

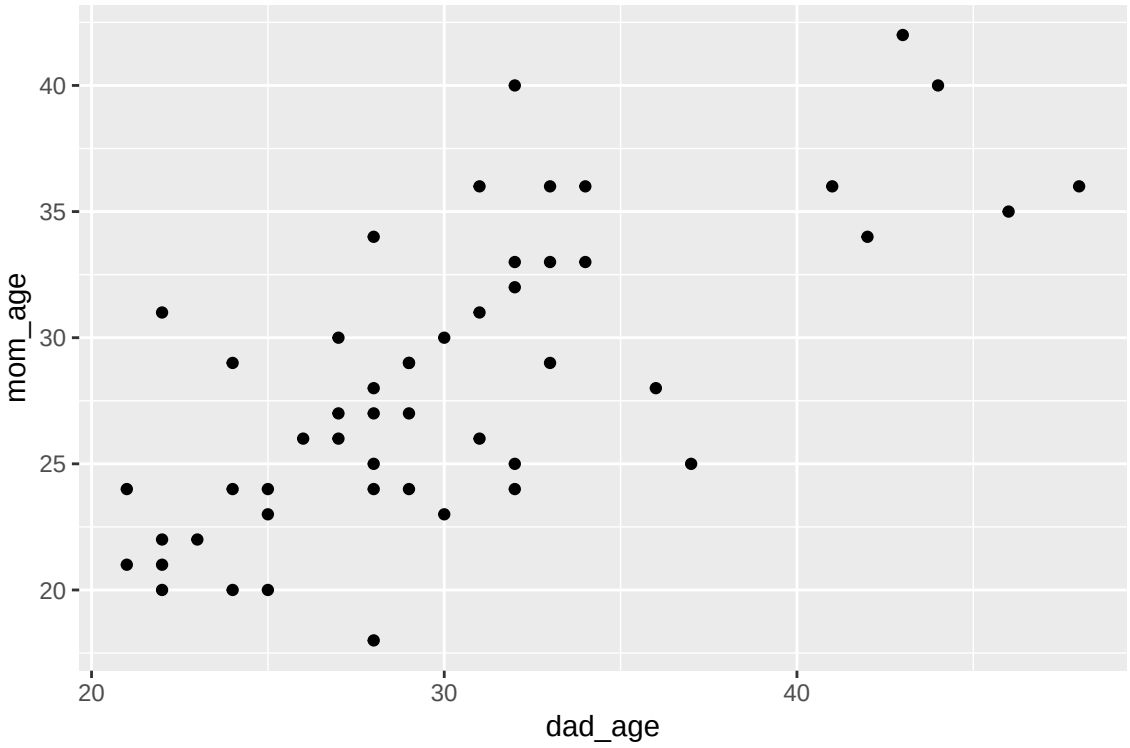
Random Matters: Correlations Vary

```
LiveBirths <- read_csv("http://nhorton.people.amherst.edu/is5/data/Babysamp_98.csv") %>%
  clean_names()
```

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

```
##   MomAge = col_integer(),
##   DadAge = col_integer(),
##   MomEduc = col_integer(),
##   MomMarital = col_integer(),
##   numlive = col_integer(),
##   dobmm = col_integer(),
##   gestation = col_integer(),
##   sex = col_character(),
##   weight = col_integer(),
##   prenatalstart = col_integer(),
##   orig.id = col_integer(),
##   preemie = col_logical()
## )
```

```
LiveBirths <- LiveBirths %>%
  filter(dad_age != "NA")
set.seed(14513) # For reproducibility
numsim <- 10000 # Number of samples
gf_point(mom_age ~ dad_age, data = sample(LiveBirths, 50))
```




```
# Graph will look different for different samples
cor(mom_age ~ dad_age, data = LiveBirths)
```

```
## [1] 0.7516507
```

```
# What does do() do?
cor(mom_age ~ dad_age, data = sample(LiveBirths, 50)) # Correlation of one random sample
```

```
## [1] 0.7619002
```

```
cor(mom_age ~ dad_age, data = sample(LiveBirths, 50)) # Correlation of another random sample
```

```
## [1] 0.7767026
```

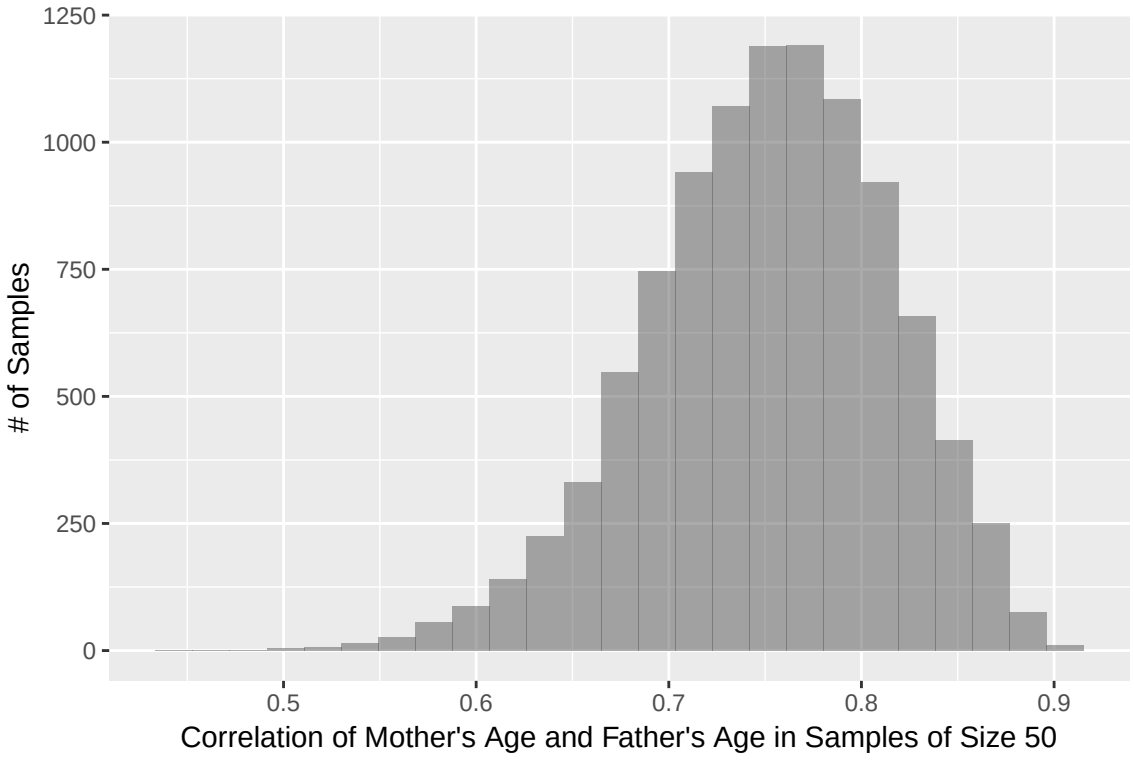
```
do(2) * cor(mom_age ~ dad_age, data = sample(LiveBirths, 50)) # Finds the correlation twice
```

```
##         cor
## 1 0.7067583
## 2 0.7401397
```

```
# For the visualization, we need 10,000 correlations
LiveCorr <- do(numsim) * cor(mom_age ~ dad_age, data = sample(LiveBirths, 50))
```

The `do()` function runs, 10,000 times, the correlation and sampling functions on a random sample of 50.

```
gf_histogram(~ cor, data = LiveCorr,
  xlab = "Correlation of Mother's Age and Father's Age in Samples of Size 50",
  ylab = "# of Samples")
```

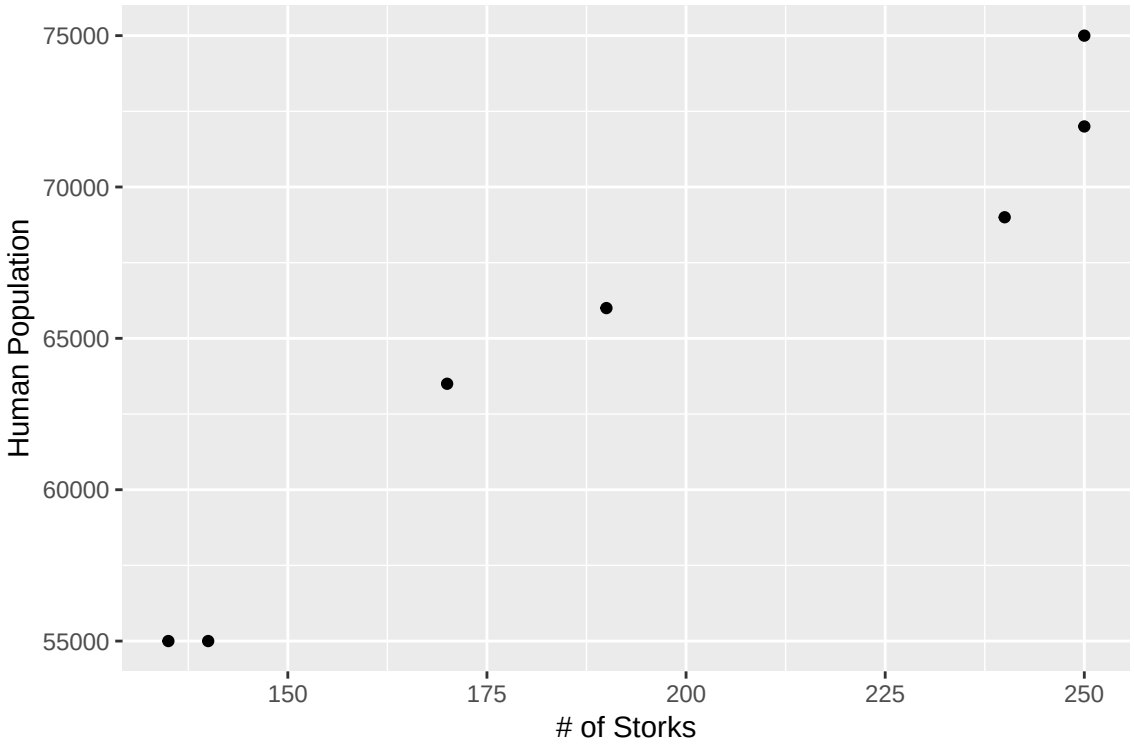


Section 6.3: Warning: Correlation $\neq$ Causation

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

```
## Parsed with column specification:
## cols(
##   Storks = col_integer(),
##   Population = col_integer()
## )
```

```
# Figure 6.9
gf_point(Population ~ Storks, data = Storks, xlab = "# of Storks", ylab = "Human Population")
```



Correlation Tables

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

```
## Parsed with column specification:
## cols(
##   Company = col_character(),
##   Assets = col_integer(),
##   Sales = col_integer(),
##   `Market Value` = col_integer(),
##   Profits = col_double(),
##   `Cash Flow` = col_double(),
##   Employees = col_double(),
##   sector = col_character(),
##   Banks = col_integer()
## )
```

```
Companies %>%
  select(assets, sales, market_value, profits, cash_flow, employees) %>%
  cor()
```

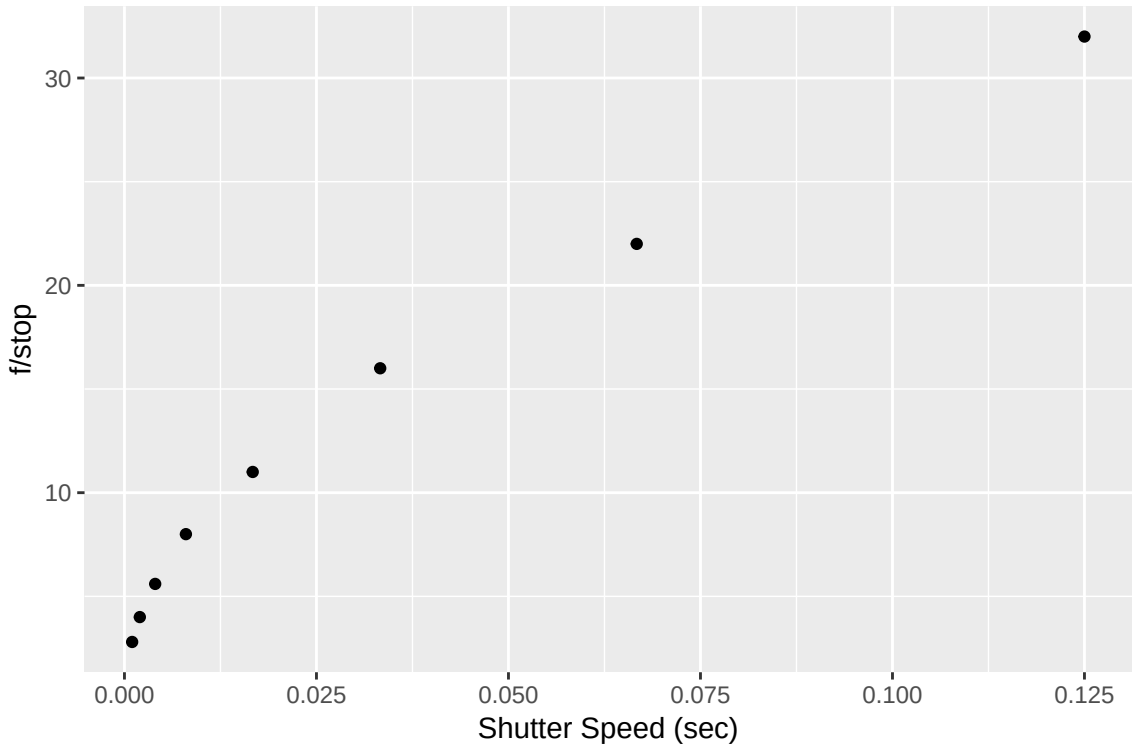
```
##                  assets     sales market_value   profits cash_flow
## assets       1.0000000 0.7464649    0.6822122 0.6016986 0.6409018
## sales        0.7464649 1.0000000    0.8788920 0.8137758 0.8549172
## market_value 0.6822122 0.8788920    1.0000000 0.9681987 0.9702851
## profits      0.6016986 0.8137758    0.9681987 1.0000000 0.9887795
## cash_flow    0.6409018 0.8549172    0.9702851 0.9887795 1.0000000
## employees    0.5943581 0.9240429    0.8182161 0.7621057 0.7866148
##              employees
## assets       0.5943581
## sales        0.9240429
## market_value 0.8182161
## profits      0.7621057
## cash_flow    0.7866148
## employees    1.0000000
```

Section 6.4: Straightening Scatterplots

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

```
## Parsed with column specification:
## cols(
##   `F-stop` = col_double(),
##   ShutterSpeed = col_double()
## )
```

```
# Figure 6.10, page 179
gf_point(f_stop ~ shutter_speed, data = FStops, xlab = "Shutter Speed (sec)", ylab = "f/stop")
```



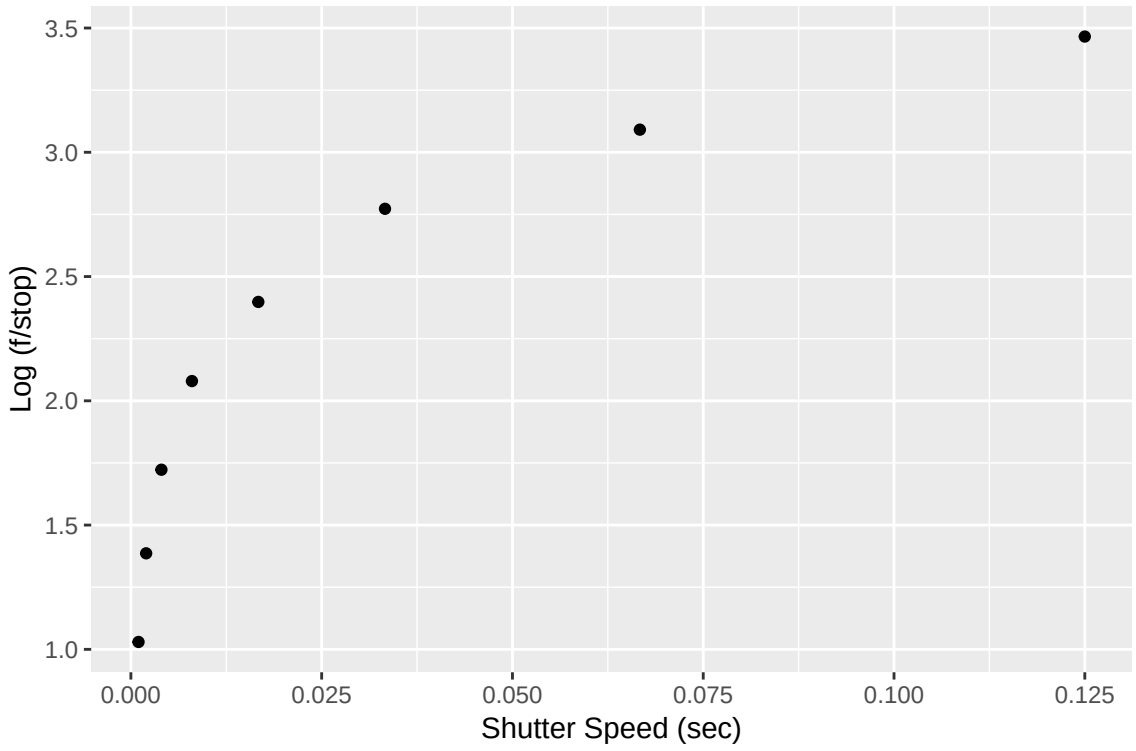

```
cor(f_stop ~ shutter_speed, data = FStops)
```

```
## [1] 0.9786716
```

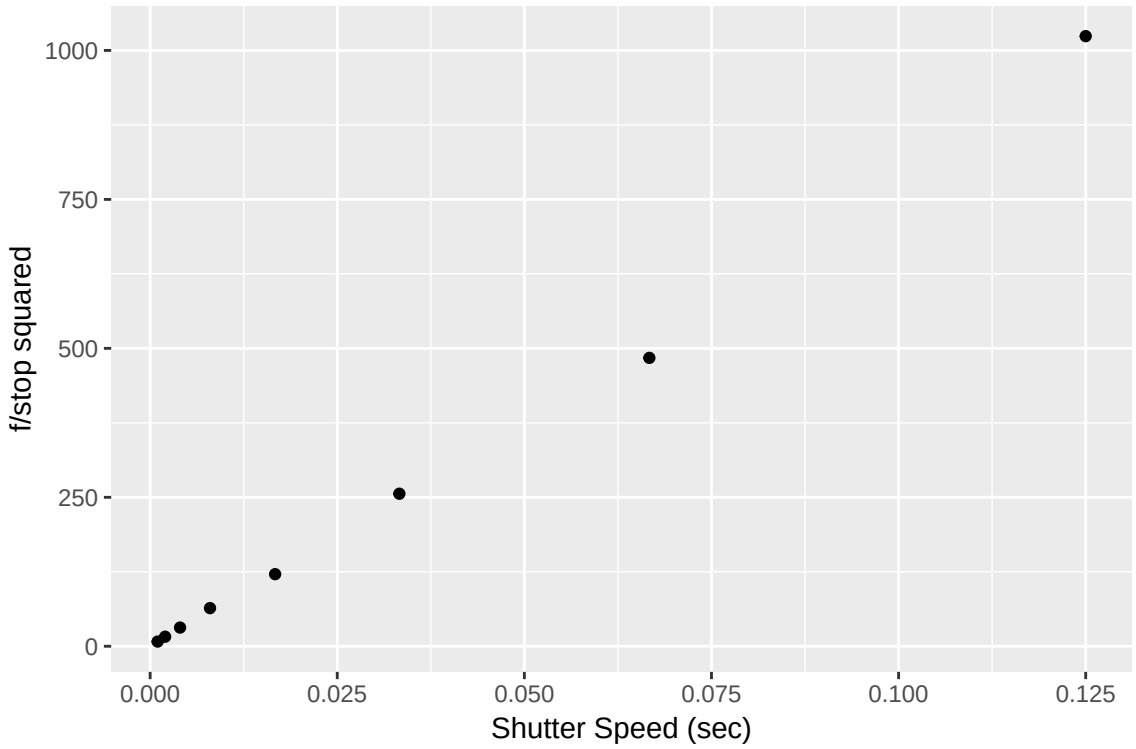
The Ladder of Powers

f/Stops Again

```
# Figure 6.11, page 181
gf_point(log(f_stop) ~ shutter_speed, data = FStops, xlab = "Shutter Speed (sec)",
         ylab = "Log (f/stop)")
```




```
# Figure 6.12
gf_point((f_stop)^2 ~ shutter_speed, data = FStops, xlab = "Shutter Speed (sec)",
         ylab = "f/stop squared")
```



See displays in "What Can Go Wrong?" on pages 181-183.