IS5 in R: Linear Regression (Chapter 7)

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

July 24, 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 7: Linear Regression

```r
library(mosaic)
library(readr)
library(janitor)
# Figure 7.1
BurgerKing <- read_csv("http://nhorton.people.amherst.edu/is5/data/Burger_King_items.csv") %>%
  clean_names()
```

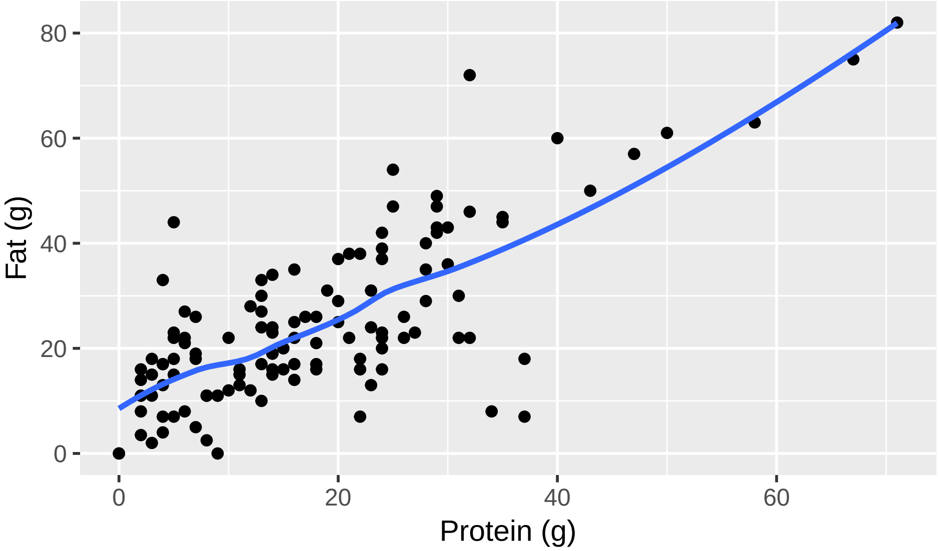
```
## Parsed with column specification:
## cols(
##   Item = col_character(),
##   Serving.size = col_integer(),
##   Calories = col_integer(),
##   Fat.Cal = col_integer(),
##   `Protein(g)` = col_integer(),
##   `Fat(g)` = col_double(),
##   `Sat.Fat(g)` = col_double(),
##   `Trans.fat(g)` = col_double(),
##   `Chol(mg)` = col_integer(),
##   `Sodium(mg)` = col_integer(),
##   `Carbs(g)` = col_integer(),
##   `Fiber(g)` = col_integer(),
##   `Sugar(g)` = col_integer(),
##   Meat = col_integer(),
##   Breakfast = col_integer(),
##   `Not Breakfast` = col_integer(),
##   CarbsxMeat = col_integer()
## )
```

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).

```
gf_point(fat_g ~ protein_g, data = BurgerKing) %>%
  gf_smooth() %>%
  gf_labs(x = "Protein (g)", y = "Fat (g)")
```

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



Here we add a smoother to show a clearer picture of the relationship.

Section 7.1: Least Squares: The Line of "Best Fit"

See display on page 197.

We can calculate the residual for a particular value with 31 grams of protein by creating an function called `burgerfun` using the `makeFun()` function.

```
burgerlm <- lm(fat_g ~ protein_g, data = BurgerKing)
burgerfun <- makeFun(burgerlm)
burgerfun(protein_g = 31)
```

```
##        1
## 36.70931
```

Section 7.2: The Linear Model

```
coef(burgerlm)
```

```
## (Intercept)   protein_g
##   8.4021494   0.9131343
```

```
burgerfun(protein_g = 0)
```

```
##        1
## 8.402149
```

```
burgerfun(32) - burgerfun(31)
```

```
##         1
## 0.9131343
```

```
msummary(burgerlm)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.40215    1.60400   5.238 7.02e-07 ***
## protein_g    0.91313    0.07177  12.723  < 2e-16 ***
##
## Residual standard error: 10.57 on 120 degrees of freedom
## Multiple R-squared:  0.5743, Adjusted R-squared:  0.5707
## F-statistic: 161.9 on 1 and 120 DF,  p-value: < 2.2e-16
```

Example 7.1: A Linear Model for Hurricanes

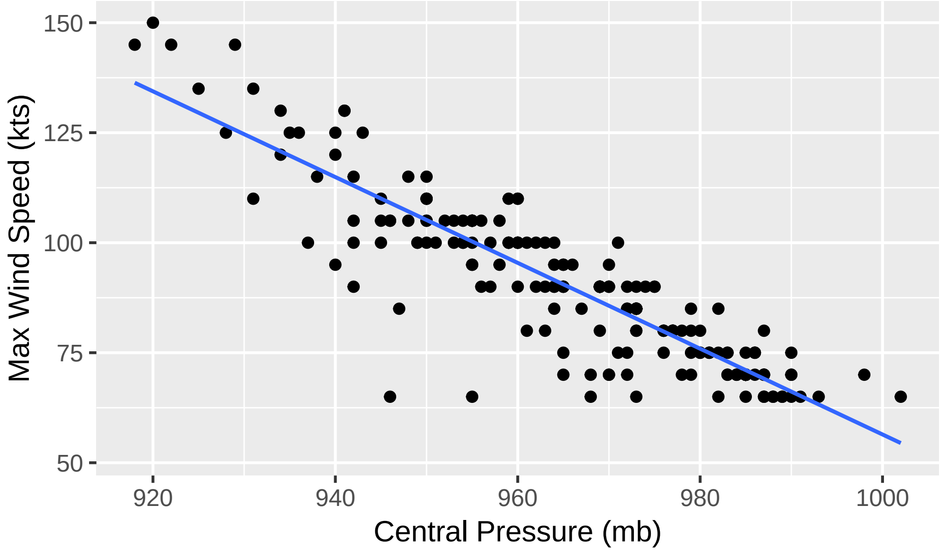
```
Hurricanes <- read_csv("http://nhorton.people.amherst.edu/is5/data/Hurricanes_2015.csv") %>%
  clean_names()
```

```
## Parsed with column specification:
## cols(
##   Name = col_character(),
##   Year = col_integer(),
##   `Max.Wind.Speed(kts)` = col_integer(),
##   `Central.Pressure(mb)` = col_integer(),
##   Category = col_integer()
## )
```

```
gf_point(max_wind_speed_kts ~ central_pressure_mb, data = Hurricanes) %>%
  gf_lm() %>%
  gf_labs(x = "Central Pressure (mb)", y = "Max Wind Speed (kts)")
```

```
## Warning: Removed 7 rows containing non-finite values (stat_lm).
```

```
## Warning: Removed 7 rows containing missing values (geom_point).
```



The function generates a warning because some of the data are missing: this output can be suppressed by adding `warning=FALSE` as an option in this code chunk.

Section 7.3: Finding the Least Squares Line

Example 7.2: Finding the Regression Equation

```
favstats(~ protein_g, data = BurgerKing)
```

```
##  min Q1 median    Q3 max     mean       sd   n missing
##    0  7   15.5 24.75  71 17.93443 13.38911 122       0
```

```
favstats(~ fat_g, data = BurgerKing)
```

```
##  min    Q1 median Q3 max     mean       sd   n missing
##    0 14.25     22 33  82 24.77869 16.13362 122       0
```

```
sx <- sd(~ protein_g, data = BurgerKing)
sx
```

```
## [1] 13.38911
```

```
sy <- sd(~ fat_g, data = BurgerKing)
sy
```

```
## [1] 16.13362
```

```
r <- cor(protein_g ~ fat_g, data = BurgerKing)
r   # same as cor(fat_g ~ protein_g)!
```

```
## [1] 0.7578003
```

```
r*sy/sx
```

```
## [1] 0.9131343
```

```
coef(burgerlm)[2]
```

```
## protein_g
## 0.9131343
```

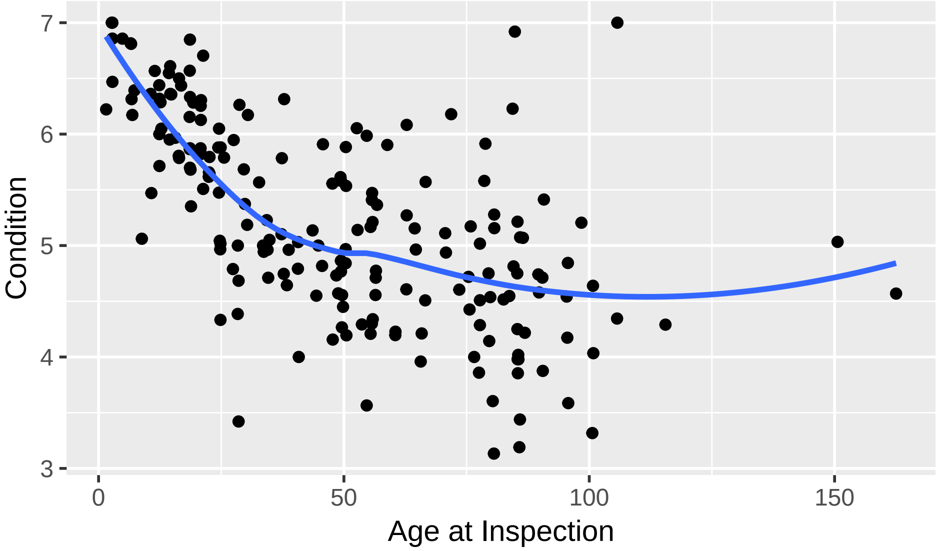
Step-by-Step Example: Calculating a Regression Equation

```
TompkinsBridges <-
  read_csv("http://nhorton.people.amherst.edu/is5/data/Tompkins_county_bridges_2016.csv") %>%
  clean_names()
```

```
## Parsed with column specification:
## cols(
##   Municipality = col_character(),
##   Location = col_character(),
##   Route = col_character(),
##   Owner = col_character(),
##   Built = col_integer(),
##   Date.Inspected = col_character(),
##   SD.FO.Status = col_character(),
##   Condition = col_double(),
##   YearInspected = col_double(),
##   AgeAtInspection = col_double()
## )
```

```
gf_point(condition ~ age_at_inspection, data = TompkinsBridges) %>%
  gf_smooth() %>% # To show relationship
  gf_labs(x = "Age at Inspection", y = "Condition")
```

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



See calculations on page 203.

Section 7.4: Regression to the Mean

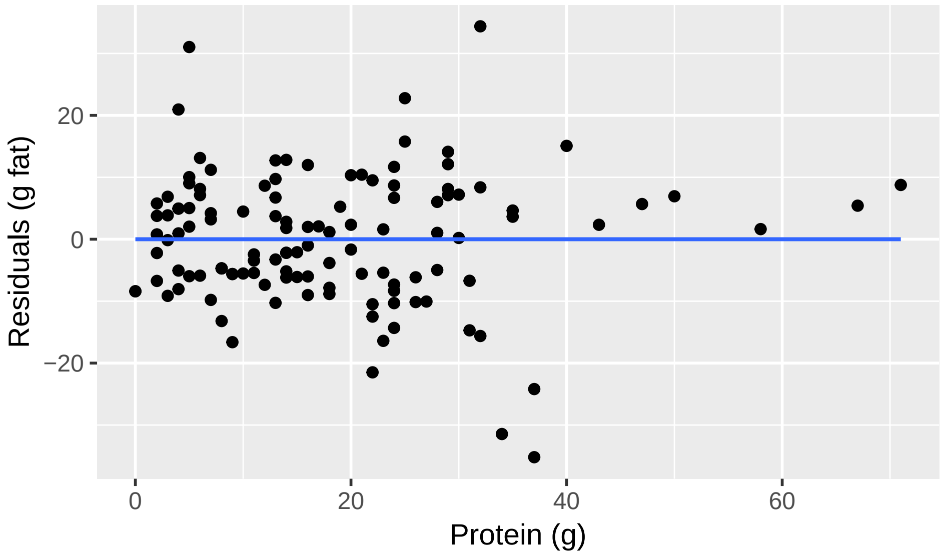
See Figure 7.4 on page 205 to visualize standard deviations.

Section 7.5: Examining the Residuals

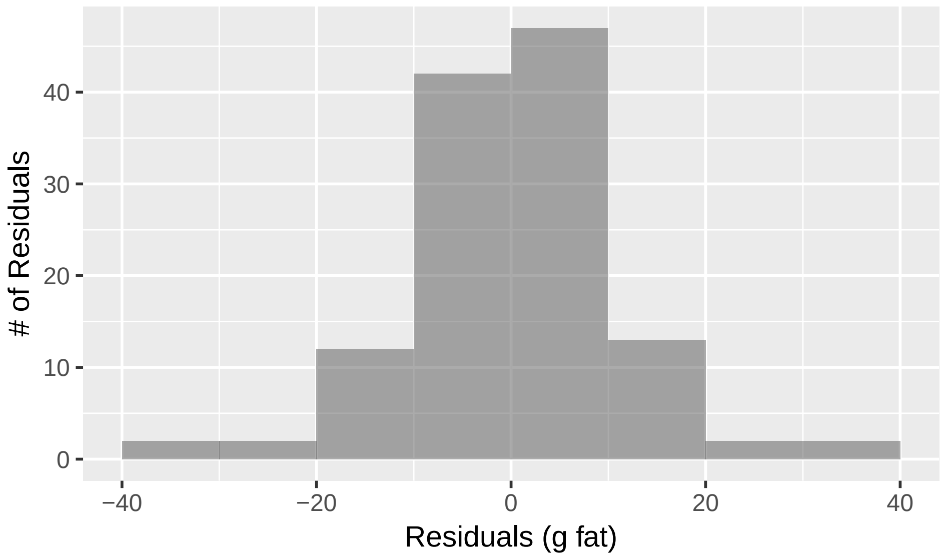
```
msummary(burgerlm)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.40215    1.60400   5.238 7.02e-07 ***
## protein_g    0.91313    0.07177  12.723  < 2e-16 ***
##
## Residual standard error: 10.57 on 120 degrees of freedom
## Multiple R-squared:  0.5743, Adjusted R-squared:  0.5707
## F-statistic: 161.9 on 1 and 120 DF,  p-value: < 2.2e-16
```

```
#Figure 7.5 , page 207
gf_point(resid(burgerlm) ~ protein_g, data = BurgerKing) %>%
  gf_lm() %>%
  gf_labs(x = "Protein (g)", y = "Residuals (g fat)")
```



```r
# Figure 7.6
gf_histogram(~ resid(burgerlm), binwidth = 10, center = 5) %>%
  gf_labs(x = "Residuals (g fat)", y = "# of Residuals")
```



Section 7.6: R^2–The Proportion of Variation Accounted for by the Model

```r
rsquared(burgerlm)
```

```
## [1] 0.5742613
```

Section 7.7: Regression Assumptions and Conditions