IS5 in R: Multiple Regression (Chapter 9)

Nicholas Horton (nhorton@amherst.edu)

December 17, 2020

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 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 (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.

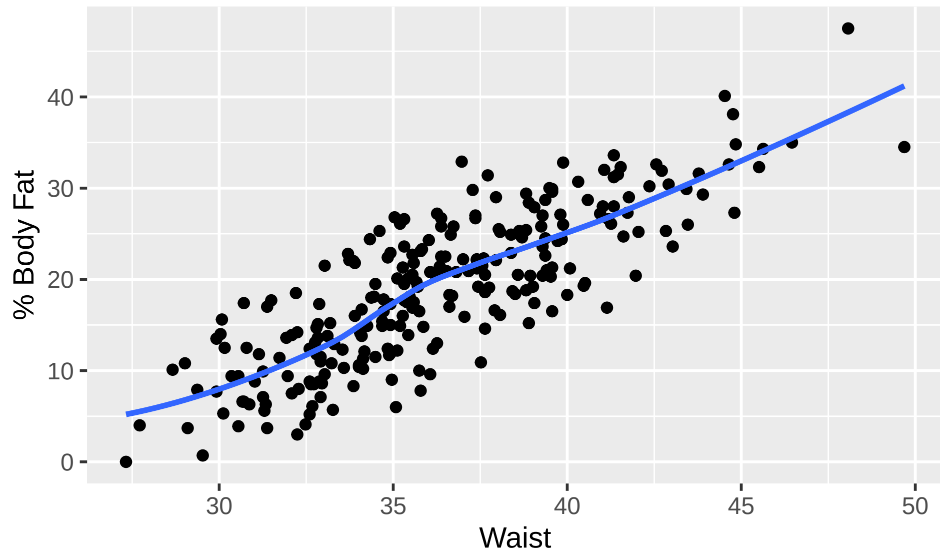
Chapter 9: Multiple Regression

```
library(mosaic)
library(readr)
library(janitor)
library(broom) # We'll use this for augment() later
BodyFat <- read_csv("http://nhorton.people.amherst.edu/is5/data/Bodyfat.csv") %>%
  janitor::clean_names()
```

By default, `read_csv()` 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 9.1, page 276
gf_point(pct_bf ~ waist, data = BodyFat) %>%
  gf_labs(x = "Waist", y = "% Body Fat") %>%
  gf_smooth()
```

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



We've added `gf_smooth()` to demonstrate how to add a smoother.

Section 9.1: What is Multiple Regression?

```
# Table 9.1, page 277
multiplereg <- lm(pct_bf ~ waist + height, data = BodyFat)
summary(multiplereg)
```

```
##
## Call:
## lm(formula = pct_bf ~ waist + height, data = BodyFat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1692  -3.4133  -0.0977   3.0995   9.9082
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.10088    7.68611  -0.403    0.687
## waist        1.77309    0.07158  24.770  < 2e-16 ***
## height      -0.60154    0.10994  -5.472 1.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.46 on 247 degrees of freedom
## Multiple R-squared:  0.7132, Adjusted R-squared:  0.7109
## F-statistic: 307.1 on 2 and 247 DF,  p-value: < 2.2e-16
```

The `summary()` function provides the multiple R-squared along with the regression coefficients.

```
RealEstate <- read_csv("http://nhorton.people.amherst.edu/is5/data/Real_Estate.csv") %>%
  janitor::clean_names()
realestatelm <- lm(price ~ living_area + bedrooms, data = RealEstate)
summary(realestatelm)
```

Example 9.1: Modeling Home Prices

```
##
## Call:
## lm(formula = price ~ living_area + bedrooms, data = RealEstate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -433211 -198136  -63249  137183 1054177
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 308100.44   41147.84   7.488 1.69e-13 ***
## living_area    135.09      11.48  11.771  < 2e-16 ***
## bedrooms    -43346.81   12844.14  -3.375 0.000771 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 266900 on 891 degrees of freedom
## Multiple R-squared:  0.1463, Adjusted R-squared:  0.1444
## F-statistic: 76.34 on 2 and 891 DF,  p-value: < 2.2e-16
```

```r
# Predicted Values
realestatefn <- makeFun(realestatelm) # Making a function to find predicted values
# Predicted price for a home with 2800 sq ft living area and 5 bedrooms
realestatefn(living_area = 2800, bedrooms = 5)
```

```
##        1
## 469614.9
```

```r
# Predicted price for a home with 2801 sq ft living area and 5 bedrooms
realestatefn(living_area = 2801, bedrooms = 5)
```

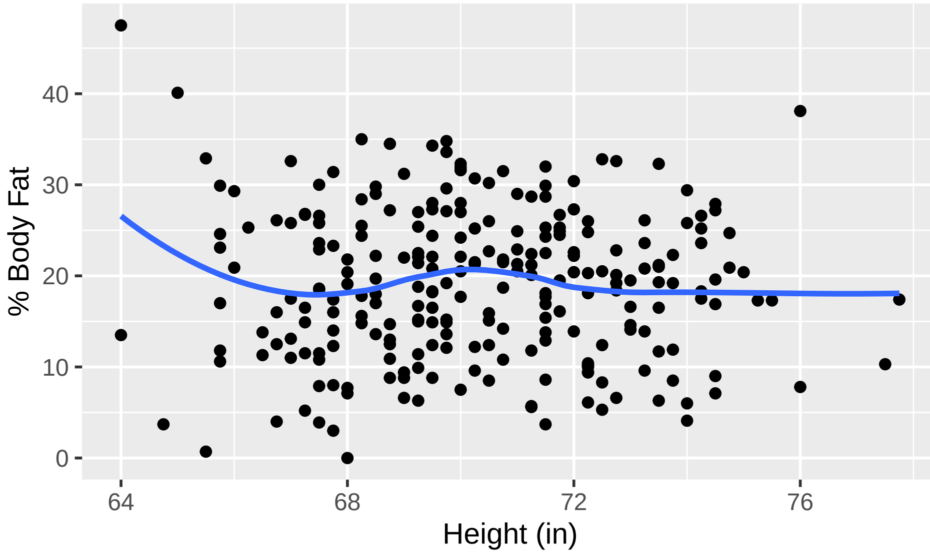
```
##      1
## 469750
```

```r
# If we subtract predicted values one value apart, we get the slope
realestatefn(living_area = 2801, bedrooms = 5) - realestatefn(living_area = 2800, bedrooms = 5)
```

```
##        1
## 135.0887
```

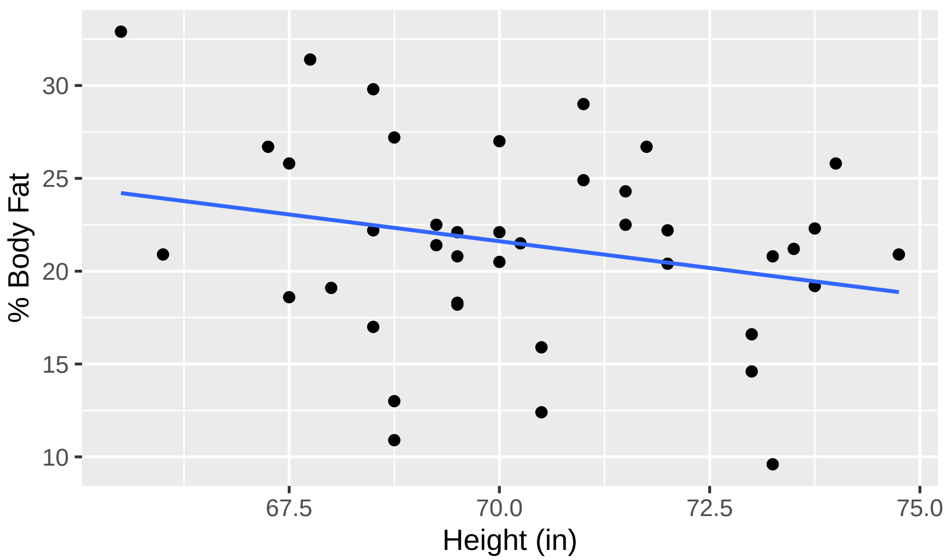
Section 9.2: Interpreting Multiple Regression Coefficients

```r
# Figure 9.2, page 279
gf_point(pct_bf ~ height, data = BodyFat) %>%
  gf_smooth() %>% # Added a smoother to assess linearity
  gf_labs(x = "Height (in)", y = "% Body Fat")
```

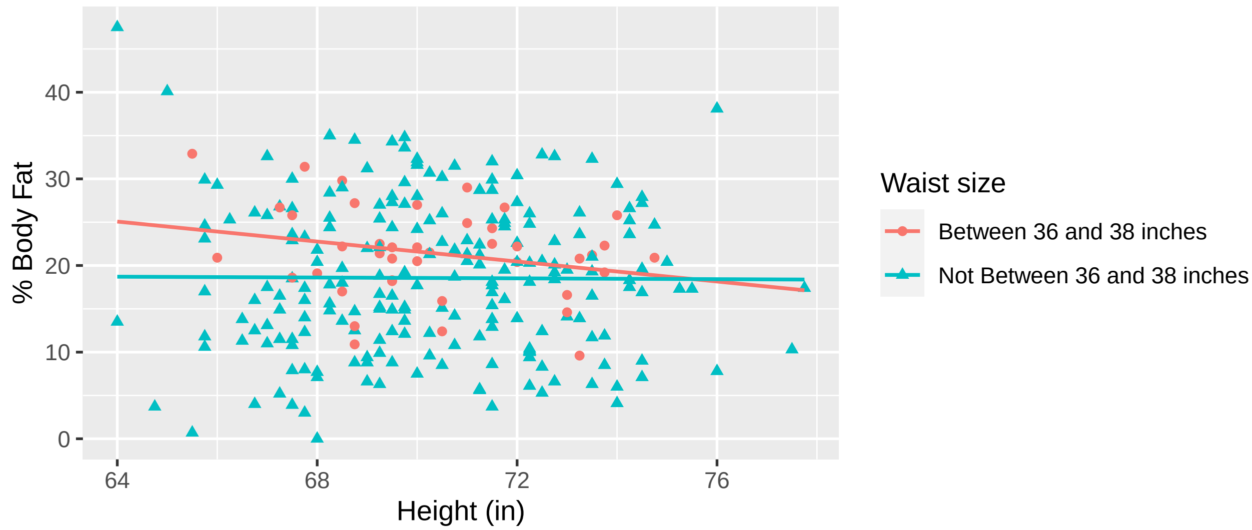


A message about the default smoother option was suppressed by adding `message = FALSE` as a code chunk option.

```
# Figure 9.3
BodyFat %>%
  filter(waist >= 36 & waist <= 38) %>% # Just plotting waist sizes between 36 and 38 inches
  gf_point(pct_bf ~ height) %>%
  gf_labs(x = "Height (in)", y = "% Body Fat") %>%
  gf_lm()
```




```
# Plotting all points
BodyFat %>%
  mutate(waistsize = ifelse(waist >= 36 & waist <= 38, "Between 36 and 38 inches",
    "Not Between 36 and 38 inches"
  )) %>% # Subsetting
  gf_point(pct_bf ~ height, shape = ~waistsize, color = ~waistsize) %>%
  gf_labs(x = "Height (in)", y = "% Body Fat", shape = "Waist size", color = "Waist size") %>%
  gf_lm()
```

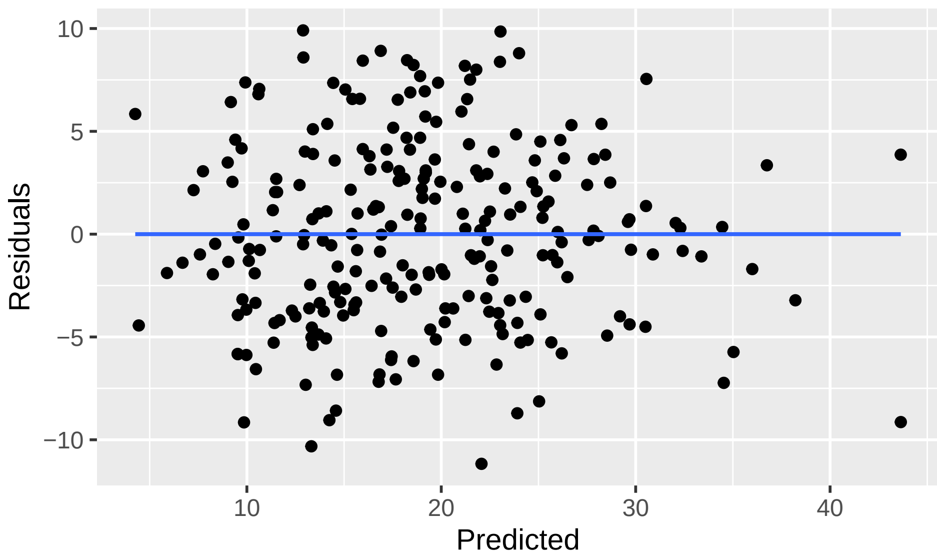


Section 9.3: The Multiple Regression Model–Assumptions and Conditions

Linearity Assumption

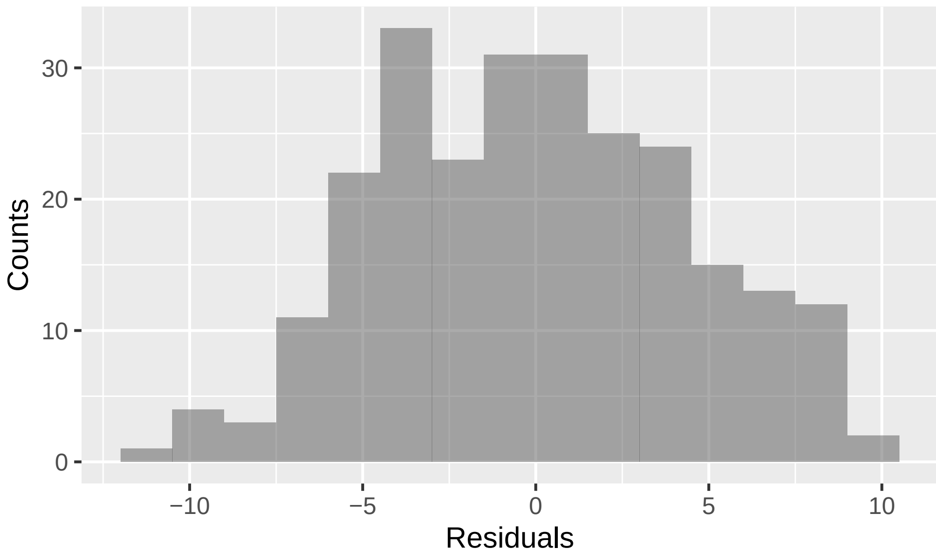
Equal Variance Assumption We can assess the equal variance assumption in several ways. The simplest is through a scatterplot of residuals vs. fitted values.

```
bodyfatlm <- lm(pct_bf ~ waist + height, data = BodyFat)
# Figure 9.4, page 282
gf_point(resid(bodyfatlm) ~ fitted(bodyfatlm)) %>%
  gf_lm() %>%
  gf_labs(x = "Predicted", y = "Residuals")
```

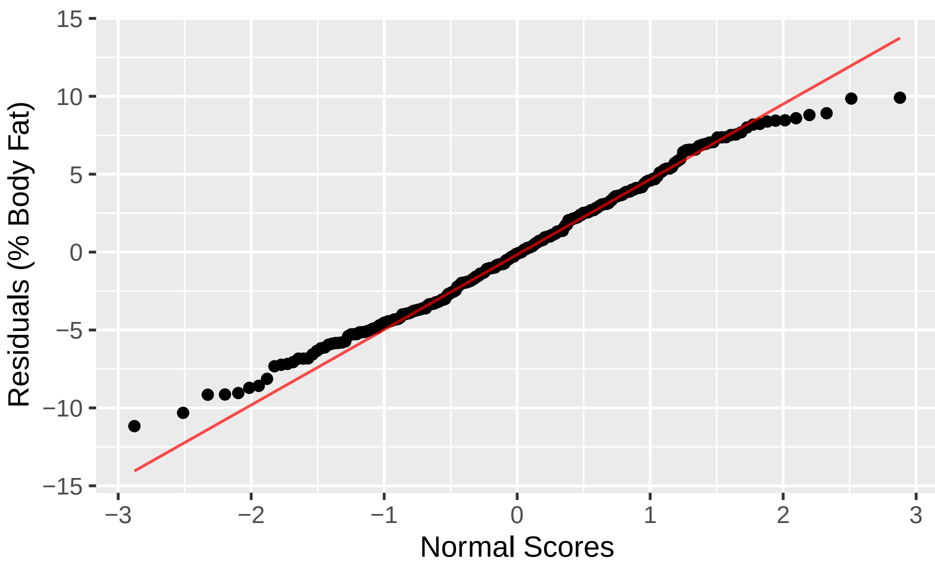


Check the Residuals It's important to look at the residuals to see if the "Nearly Normal" condition is reasonable to assume.

```
# Figure 9.5
gf_histogram(~ resid(bodyfatlm), binwidth = 1.5, center = 0.75) %>%
  gf_labs(x = "Residuals", y = "Counts")
```

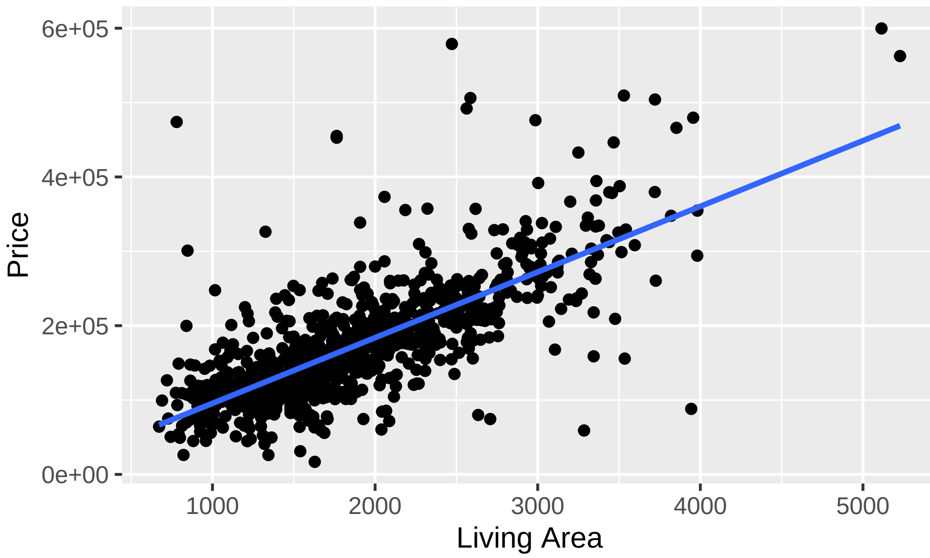



```
gf_qq(~ resid(bodyfatlm)) %>%
  gf_qqline(linetype = "solid", color = "red") %>%
  gf_labs(x = "Normal Scores", y = "Residuals (% Body Fat)")
```

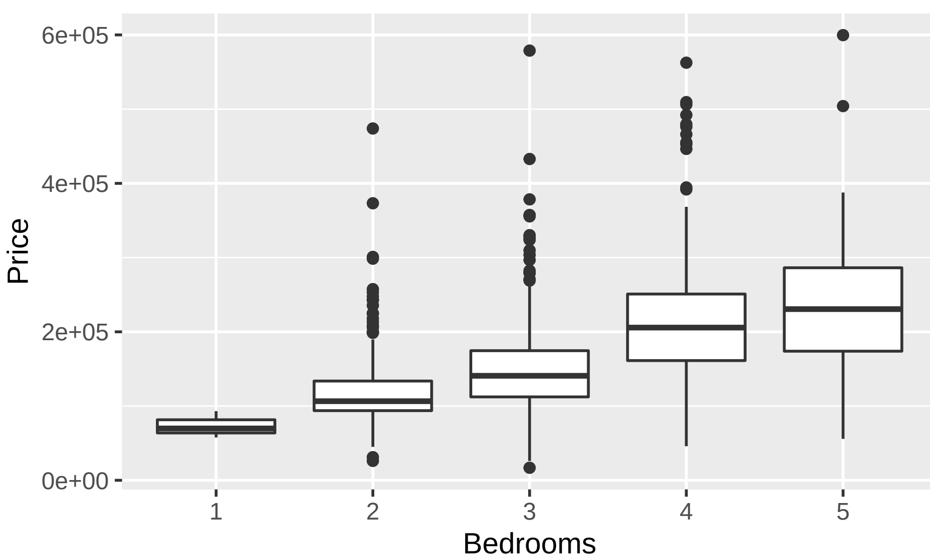


Step-By-Step Example: Multiple Regression We begin by reading in the data for the step-by-step example.

```
HousingPrices <- read_csv("http://nhorton.people.amherst.edu/is5/data/Housing_prices.csv") %>%
  janitor::clean_names()
gf_point(price ~ living_area, data = HousingPrices) %>%
  gf_smooth() %>%
  gf_labs(x = "Living Area", y = "Price")
```

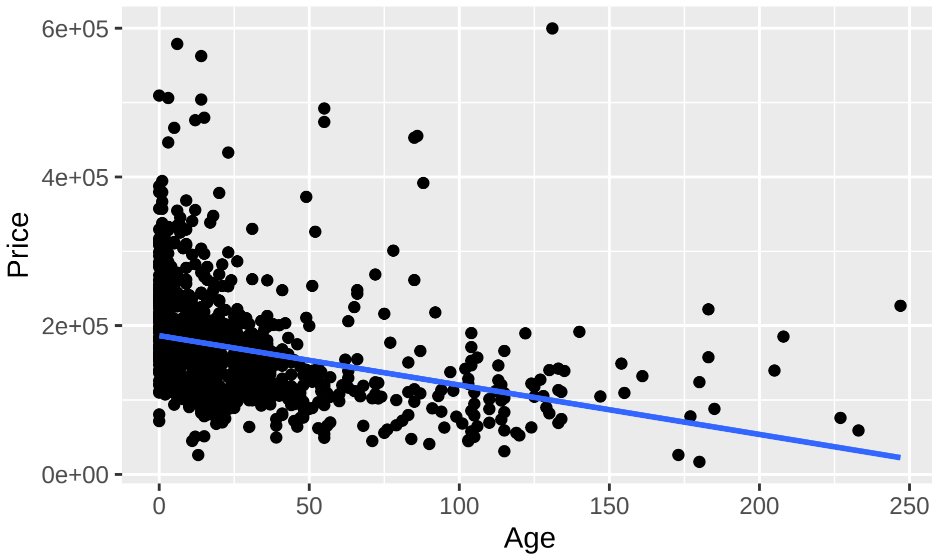



```
gf_boxplot(price ~ as.factor(bedrooms), data = HousingPrices) %>%
  gf_labs(x = "Bedrooms", y = "Price")
```



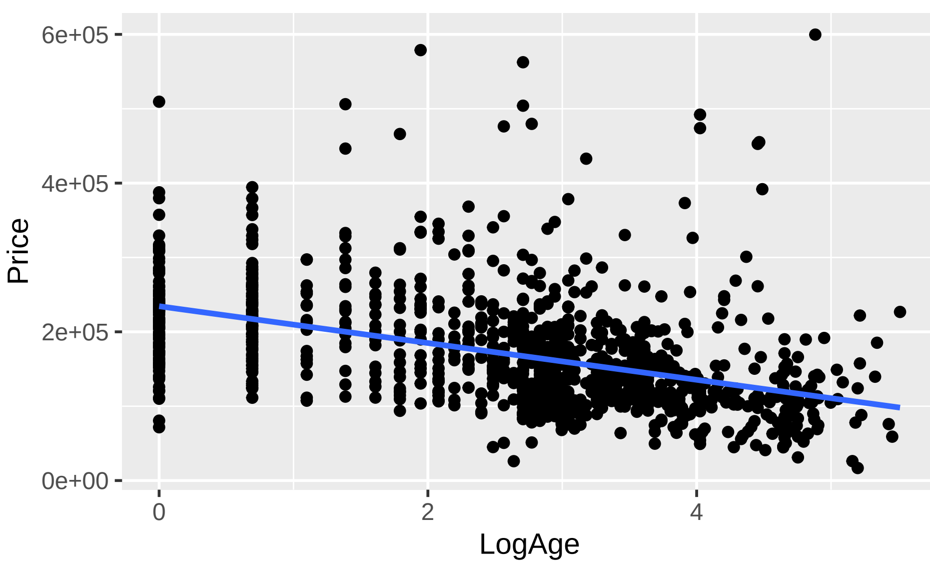

```
gf_point(price ~ age, data = HousingPrices) %>%
  gf_smooth() %>%
  gf_labs(x = "Age", y = "Price")
```

```
## `geom_smooth()` using method = 'gam'
```



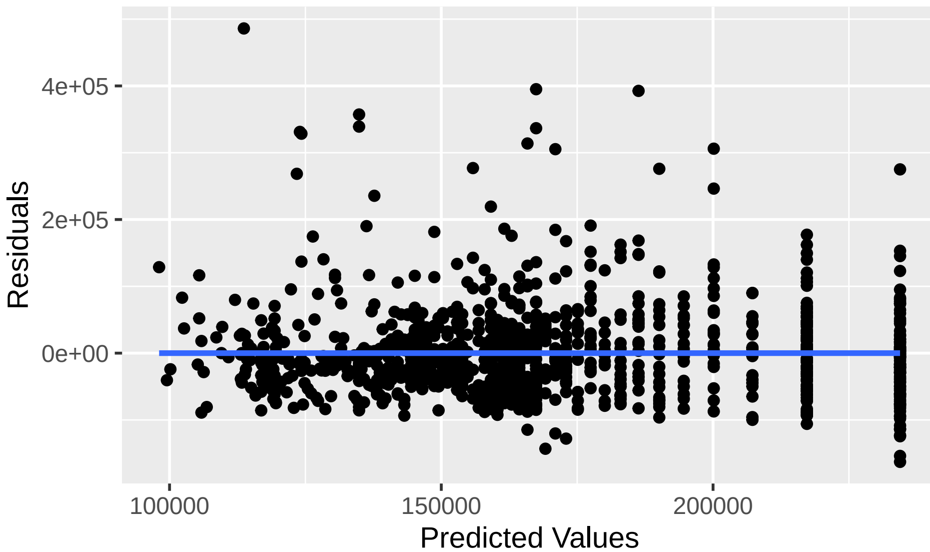

```
gf_point(price ~ log(age + 1), data = HousingPrices) %>%
  gf_smooth() %>%
  gf_labs(x = "LogAge", y = "Price")
```

```
## `geom_smooth()` using method = 'gam'
```




```
housinglm <- lm(price ~ log(age + 1), data = HousingPrices)
gf_point(resid(housinglm) ~ fitted(housinglm)) %>%
  gf_smooth() %>%
  gf_labs(x = "Predicted Values", y = "Residuals")
```

```
## `geom_smooth()` using method = 'gam'
```

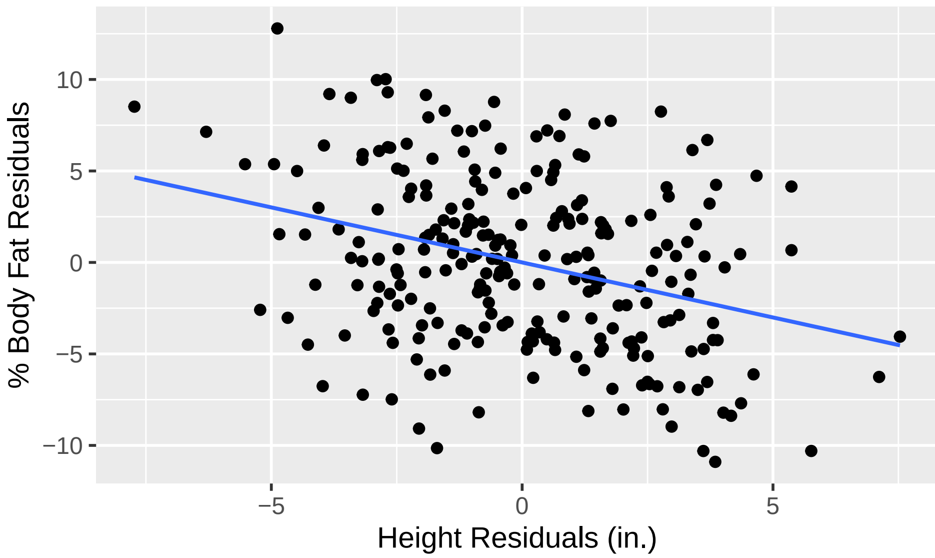



```r
housinglm2 <- lm(price ~ living_area + log(age + 1) + bedrooms, data = HousingPrices)
msummary(housinglm2)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  44797.165   8356.609   5.361 1.02e-07 ***
## living_area     87.260      3.365  25.928  < 2e-16 ***
## log(age + 1) -6270.813   1299.133  -4.827 1.59e-06 ***
## bedrooms     -5902.756   2773.934  -2.128   0.0336 *
##
## Residual standard error: 49620 on 1053 degrees of freedom
## Multiple R-squared:  0.5876, Adjusted R-squared:  0.5864
## F-statistic: 500.1 on 3 and 1053 DF,  p-value: < 2.2e-16
```

Section 9.4: Partial Regression Plots

```r
# Figure 9.6 (instructions on 287)
# Step 1
otherthanheightlm <- lm(pct_bf ~ waist, data = BodyFat)
# Step 2
residualsoflm <- resid(otherthanheightlm)
# Step 3
yheightlm <- lm(height ~ waist, data = BodyFat)
# Step 4
residualsoflm2 <- resid(yheightlm)
# Step 5
gf_point(residualsoflm ~ residualsoflm2) %>%
  gf_lm() %>%
  gf_labs(x = "Height Residuals (in.)", y = "% Body Fat Residuals")
```




```
Hurricanes <- read_csv("http://nhorton.people.amherst.edu/is5/data/Hurricanes_2015.csv") %>%
  janitor::clean_names()
hurricanelm <- lm(max_wind_speed_kts ~ year + central_pressure_mb, data = Hurricanes)
msummary(hurricanelm)
```

Just Checking

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.032e+03  3.852e+01  26.789   <2e-16 ***
## year                  -3.132e-04  9.075e-03  -0.035    0.973
## central_pressure_mb -9.750e-01  3.287e-02 -29.666   <2e-16 ***
##
## Residual standard error: 8.199 on 217 degrees of freedom
##   (7 observations deleted due to missingness)
## Multiple R-squared:  0.8056, Adjusted R-squared:  0.8038
## F-statistic: 449.6 on 2 and 217 DF,  p-value: < 2.2e-16
```

Section 9.5: Indicator Variables

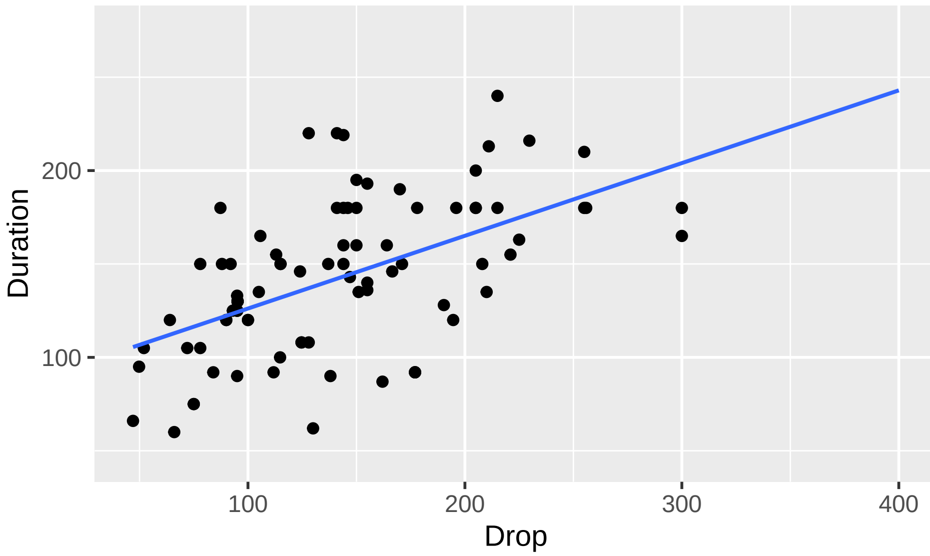
```
Coasters <- read_csv("http://nhorton.people.amherst.edu/is5/data/Coasters_2015.csv")
# Table 9.2, page 288
head(Coasters)
```

```
## # A tibble: 6 x 9
##   Name          Park          Track Speed Height  Drop Length Duration Inversions
##   <chr>         <chr>         <chr> <dbl>  <dbl> <dbl>  <dbl>    <dbl>      <dbl>
## 1 Top Thrill ~ Cedar Point   Steel   120    420  400     2800       NA          0
## 2 Superman Th~ Six Flags Ma~ Steel   100    415  328.    1235       NA          0
## 3 Millennium ~ Cedar Point   Steel    93    310  300     6595      165          0
## 4 Goliath      Six Flags Ma~ Steel    85    235  255     4500      180          0
## 5 Titan        Six Flags Ov~ Steel    85    245  255     5312      210          0
## 6 Phantom's R~ Kennywood Pa~ Steel    82    160  228     3200       NA          0
```

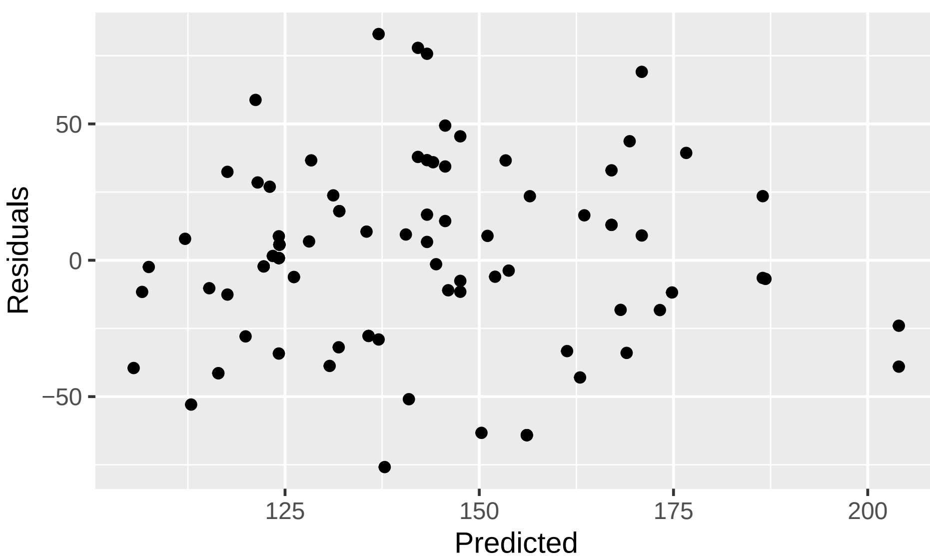
```
# Figure 9.7
# Tower of Terror isn't included by the book
Coasters <- Coasters %>%
```

```
  filter(Name != "Tower of Terror") %>%
  mutate(Inversions = as.factor(Inversions))
```

```
gf_point(Duration ~ Drop, data = Coasters) %>%
  gf_lm()
```




```
coasterlm <- lm(Duration ~ Drop, data = Coasters)
gf_point(resid(coasterlm) ~ fitted(coasterlm)) %>%
  gf_labs(x = "Predicted", y = "Residuals")
```



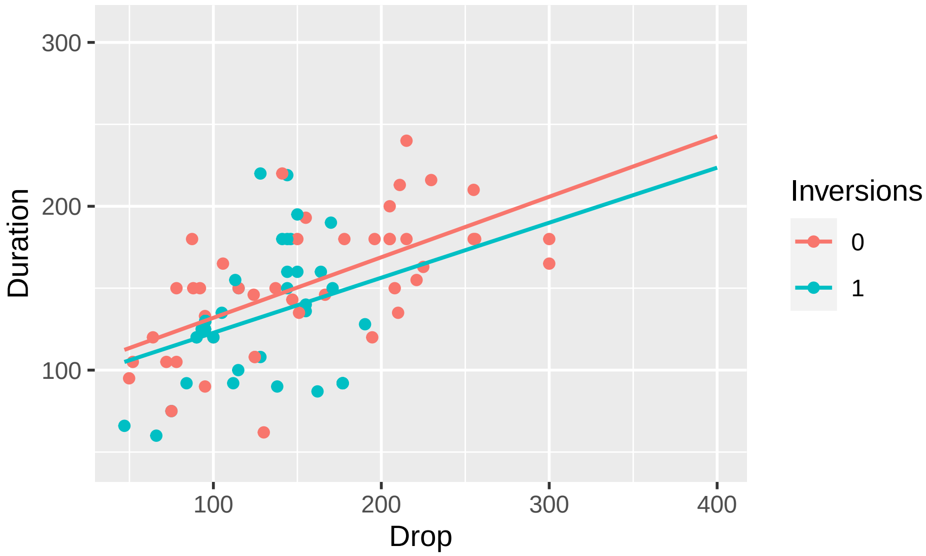

```
msummary(coasterlm)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 87.22005    9.73524   8.959 4.98e-14 ***
## Drop         0.38928    0.06428   6.056 3.36e-08 ***
##
## Residual standard error: 34.06 on 88 degrees of freedom
##   (150 observations deleted due to missingness)
## Multiple R-squared:  0.2942, Adjusted R-squared:  0.2862
```

```
## F-statistic: 36.68 on 1 and 88 DF,  p-value: 3.356e-08
```

```
# Figure 9.8
gf_point(Duration ~ Drop, color = ~Inversions, data = Coasters) %>%
  gf_lm() %>%
  gf_labs(color = "Inversions")
```

```
## Warning: Removed 150 rows containing non-finite values (stat_lm).
## Warning: Removed 150 rows containing missing values (geom_point).
```



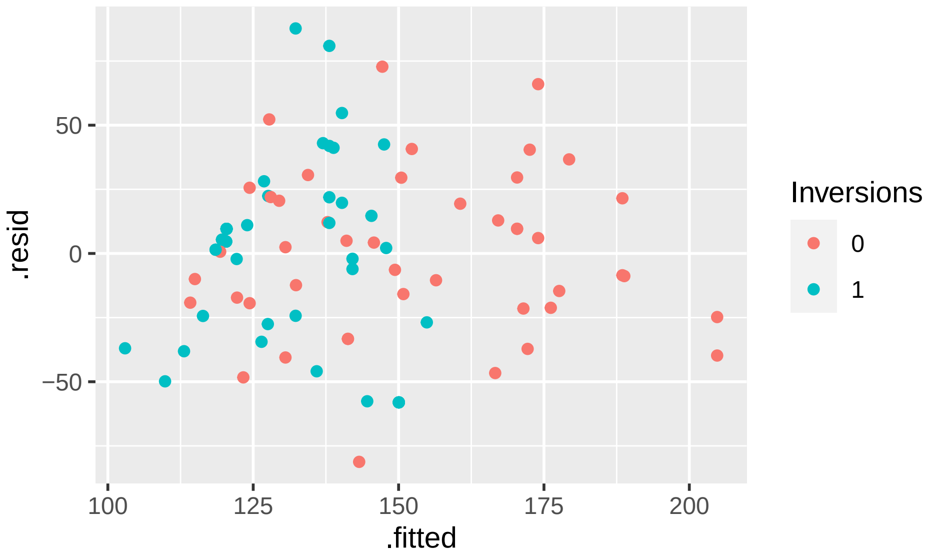

```
coasterlm2 <- lm(Duration ~ Drop + Inversions, data = Coasters)
msummary(coasterlm2)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  96.14026   11.69140   8.223 1.74e-12 ***
## Drop          0.36215    0.06699   5.406 5.58e-07 ***
## Inversions1 -10.20093    7.48401  -1.363    0.176
##
## Residual standard error: 33.9 on 87 degrees of freedom
##   (150 observations deleted due to missingness)
## Multiple R-squared:  0.3089, Adjusted R-squared:  0.293
## F-statistic: 19.45 on 2 and 87 DF,  p-value: 1.045e-07
```

```
coasterlm2asdata <- augment(coasterlm2)
glance(coasterlm2) %>% data.frame()
```

```
##   r.squared adj.r.squared    sigma statistic     p.value df    logLik      AIC
## 1 0.3089346      0.293048 33.89636  19.44628 1.04492e-07  2 -443.2766 894.5532
##        BIC deviance df.residual nobs
## 1 904.5524 99959.82          87   90
```

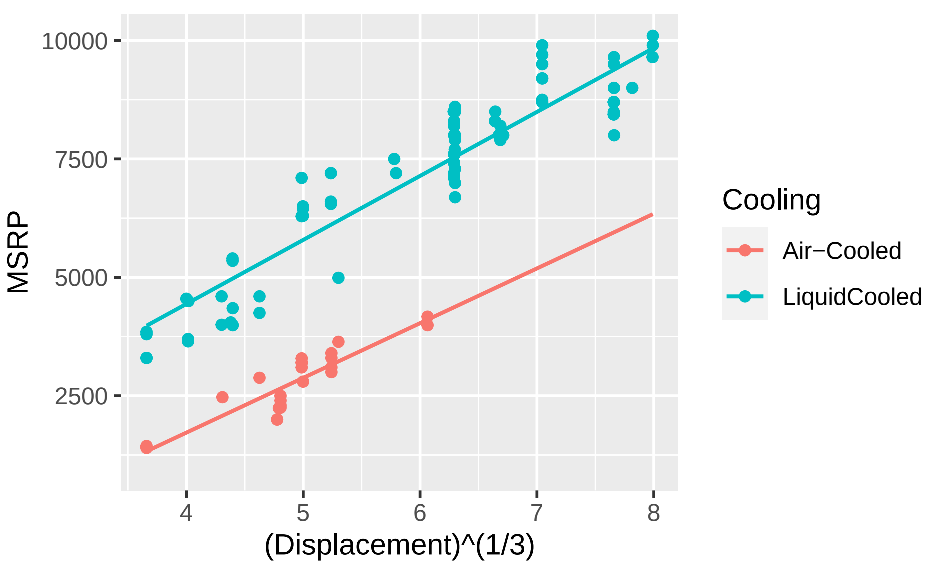
```
gf_point(.resid ~ .fitted, color = ~Inversions, data = coasterlm2asdata)
```



The augment() function creates a data frame from a linear model that includes a column for residuals, fitted values, etc. Here we use names() to check out the column names and glance() to view the structure of the data set.

Example 9.3: Using Indicator Variables We can explore the use of indicator variables to model categorical variables.

```
DirtBikes <- read_csv("http://nhorton.people.amherst.edu/is5/data/Dirt_bikes_2014.csv")
DirtBikes <- DirtBikes %>%
  filter(Cooling != "NA") %>%
  mutate(Cooling = ifelse(Cooling == "Air-Cooled", "Air-Cooled", "LiquidCooled"))
gf_point(MSRP ~ (Displacement)^(1 / 3), color = ~Cooling, data = DirtBikes) %>%
  gf_lm()
```




```
bikeslm <- lm(MSRP ~ I(Displacement^(1 / 3)) + Cooling, data = DirtBikes)
msummary(bikeslm)
```

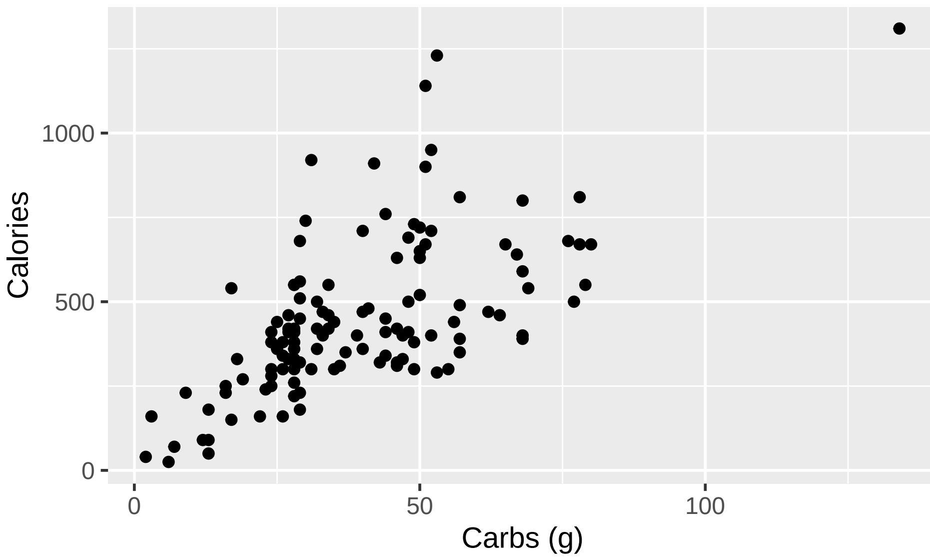
```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -3814.9      278.0  -13.72   <2e-16 ***
```

```
## I(Displacement^(1/3))   1341.4       50.4   26.61   <2e-16 ***
## CoolingLiquidCooled     2908.1      154.0   18.88   <2e-16 ***
##
## Residual standard error: 602.7 on 106 degrees of freedom
## Multiple R-squared:  0.9423, Adjusted R-squared:  0.9413
## F-statistic: 866.3 on 2 and 106 DF,  p-value: < 2.2e-16
```

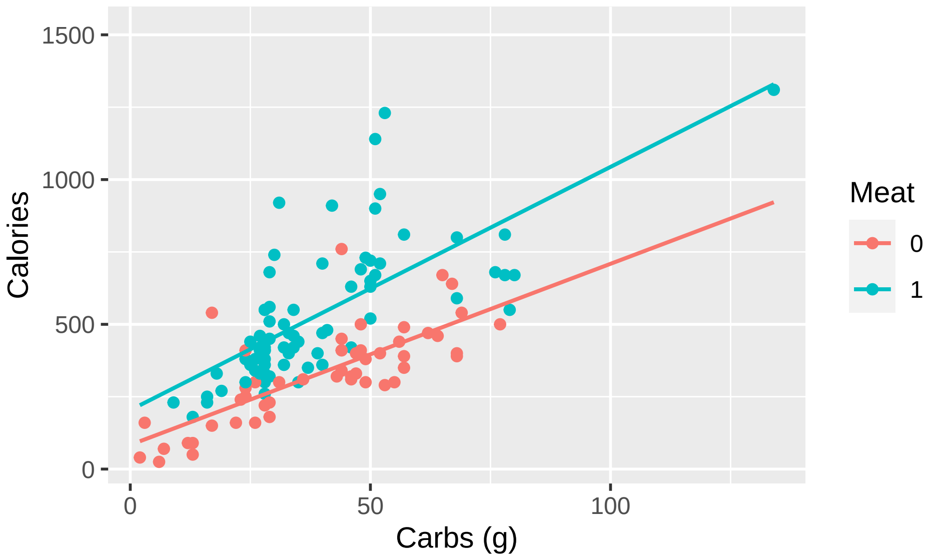
The `I()` function is used to keep the class of an object the same. Here we use it to keep the variable `Displacement` "as is" to prevent an error.

Adjusting for Different Slopes We can fit a model with different slopes.

```
BurgerKing <- read_csv("http://nhorton.people.amherst.edu/is5/data/Burger_King_items.csv") %>%
  clean_names()
# Figure 9.9, page 292
gf_point(calories ~ carbs_g, data = BurgerKing) %>%
  gf_labs(x = "Carbs (g)", y = "Calories")
```




```
# Figure 9.10
gf_point(calories ~ carbs_g, color = ~ as.factor(meat), data = BurgerKing) %>%
  gf_labs(x = "Carbs (g)", y = "Calories", color = "Meat") %>%
  gf_lm()
```

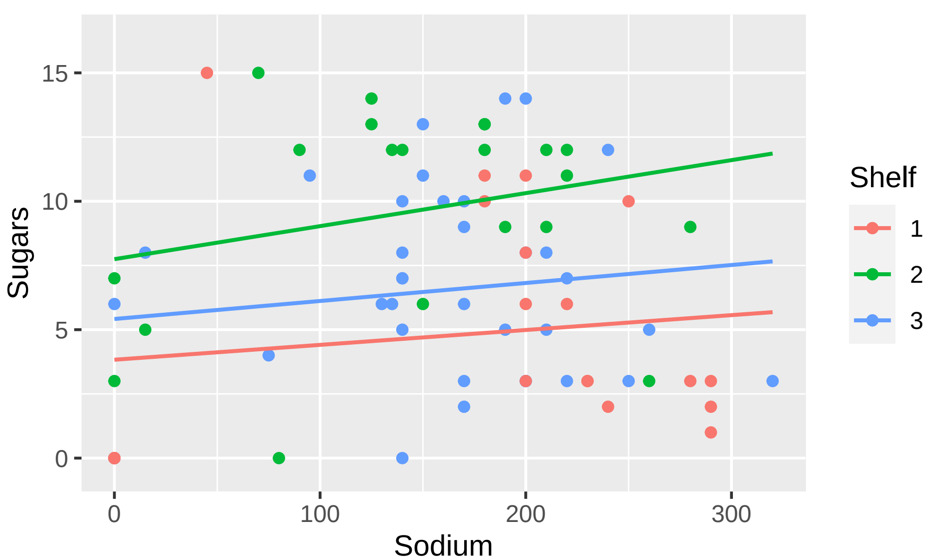



```r
msummary(lm(calories ~ carbs_g * as.factor(meat), data = BurgerKing))
```

```
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  83.533     46.955   1.779   0.0778 .
## carbs_g                       6.255      1.063   5.885 3.81e-08 ***
## as.factor(meat)1            120.220     60.694   1.981   0.0499 *
## carbs_g:as.factor(meat)1      2.145      1.378   1.557   0.1222
##
## Residual standard error: 146.5 on 118 degrees of freedom
## Multiple R-squared:  0.6072, Adjusted R-squared:  0.5972
## F-statistic:  60.8 on 3 and 118 DF,  p-value: < 2.2e-16
```

One, Two, Many We can also consider three level variables.

```r
Cereal <- read_csv("http://nhorton.people.amherst.edu/is5/data/Cereals.csv")
cereallm <- lm(sugars ~ sodium + as.factor(shelf), data = Cereal)
gf_point(sugars ~ sodium, color = ~ as.factor(shelf), data = Cereal) %>%
  gf_lm() %>%
  gf_labs(x = "Sodium", y = "Sugars", color = "Shelf")
```



```r
msummary(cereallm)
```

```
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.446740   1.345111   2.562 0.012457 *
## sodium             0.007962   0.005620   1.417 0.160818
## as.factor(shelf)2 5.012166   1.283154   3.906 0.000207 ***
## as.factor(shelf)3 1.818214   1.139384   1.596 0.114857
##
## Residual standard error: 4.07 on 73 degrees of freedom
## Multiple R-squared:  0.1866, Adjusted R-squared:  0.1532
## F-statistic: 5.583 on 3 and 73 DF,  p-value: 0.001669
```

Example 9.4: Indicators for Variables with Several Levels We will read in the diamonds data.

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

# Parallel Slopes
diamondlm <- lm(sqrt(price) ~ carat_size + color, data = Diamonds)
msummary(diamondlm)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   13.1946     0.5488  24.043  < 2e-16 ***
## carat_size    61.2491     0.5032 121.722  < 2e-16 ***
## colorE        -2.1027     0.5399  -3.895 0.000101 ***
## colorF        -2.8640     0.5576  -5.136 3.00e-07 ***
## colorG        -3.6320     0.5769  -6.296 3.57e-10 ***
## colorH        -7.8948     0.5858 -13.477  < 2e-16 ***
## colorI       -11.8542     0.6261 -18.932  < 2e-16 ***
## colorJ       -16.6404     0.6637 -25.071  < 2e-16 ***
## colorK       -21.3577     0.8282 -25.787  < 2e-16 ***
##
## Residual standard error: 7.218 on 2681 degrees of freedom
## Multiple R-squared:  0.8583, Adjusted R-squared:  0.8579
## F-statistic:  2030 on 8 and 2681 DF,  p-value: < 2.2e-16
```

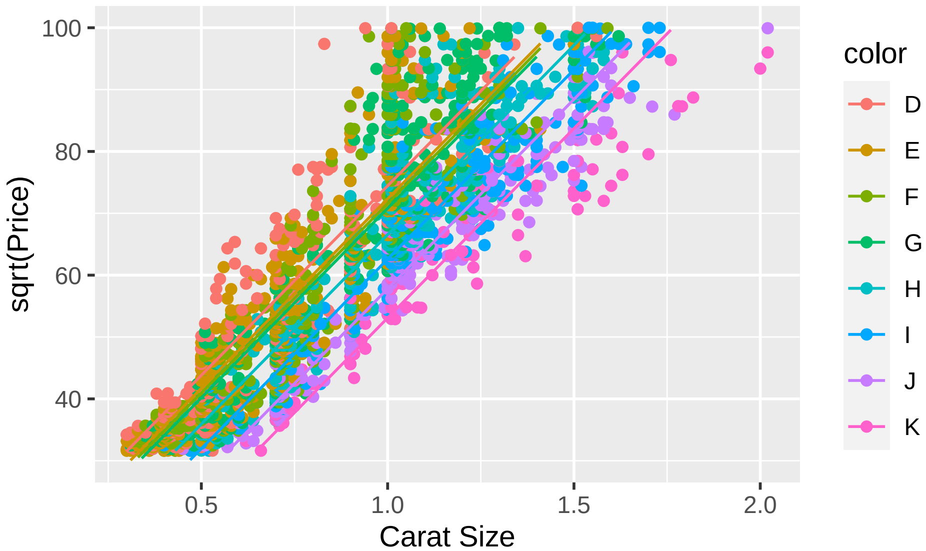
```r
diamondpredict <- makeFun(diamondlm)

diamonddata <- augment(diamondlm) %>% # To get fitted values
  clean_names()
str(diamonddata)
```

```
## tibble [2,690 x 8] (S3: tbl_df/tbl/data.frame)
##  $ sqrt_price: num [1:2690] 31.6 31.6 31.6 31.6 31.6 ...
##  $ carat_size: num [1:2690] 0.3 0.44 0.31 0.66 0.47 0.4 0.36 0.52 0.53 0.43 ...
##  $ color     : chr [1:2690] "E" "E" "E" "K" ...
##  $ fitted    : num [1:2690] 29.5 38 30.1 32.3 34.1 ...
##  $ std_resid : num [1:2690] 0.2991 -0.8902 0.2141 -0.0889 -0.342 ...
##  $ hat       : num [1:2690] 0.00269 0.00226 0.00265 0.00959 0.00375 ...
##  $ sigma     : num [1:2690] 7.22 7.22 7.22 7.22 7.22 ...
##  $ cooksd    : num [1:2690] 2.68e-05 2.00e-04 1.35e-05 8.51e-06 4.89e-05 ...
##  - attr(*, "terms")=Classes 'terms', 'formula'  language sqrt(price) ~ carat_size + color
##   .. ..- attr(*, "variables")= language list(sqrt(price), carat_size, color)
##   .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
```

```
##   .. .. .. ..$ : chr [1:3] "sqrt(price)" "carat_size" "color"
##   .. .. .. ..$ : chr [1:2] "carat_size" "color"
##   .. ..- attr(*, "term.labels")= chr [1:2] "carat_size" "color"
##   .. ..- attr(*, "order")= int [1:2] 1 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(sqrt(price), carat_size, color)
##   .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "character"
##   .. .. ..- attr(*, "names")= chr [1:3] "sqrt(price)" "carat_size" "color"
```

```
gf_point(sqrt_price ~ carat_size, color = ~color, data = diamonddata) %>%
  gf_line(fitted ~ carat_size) %>%
  gf_labs(x = "Carat Size", y = "sqrt(Price)") +
  ylim(30, 100)
```




```
# With interaction
diamondlm2 <- lm(sqrt(price) ~ carat_size * color, data = Diamonds)
msummary(diamondlm2)
```

```
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.3239     1.2142   7.679 2.23e-14 ***
## carat_size            67.0408     1.7025  39.379  < 2e-16 ***
## colorE                -0.5392     1.5075  -0.358  0.72063
## colorF                -2.3716     1.5627  -1.518  0.12922
## colorG                -2.6709     1.6643  -1.605  0.10867
## colorH                -3.9177     1.8248  -2.147  0.03189 *
## colorI                -2.5481     1.9301  -1.320  0.18689
## colorJ                -5.4176     2.0716  -2.615  0.00897 **
## colorK                 0.5976     2.7815   0.215  0.82991
## carat_size:colorE     -2.4007     2.0999  -1.143  0.25305
## carat_size:colorF     -1.3211     2.0954  -0.630  0.52843
## carat_size:colorG     -2.5457     2.0868  -1.220  0.22260
## carat_size:colorH     -5.9017     2.1774  -2.710  0.00676 **
## carat_size:colorI    -10.9139     2.1812  -5.004 5.99e-07 ***
## carat_size:colorJ    -12.4948     2.2531  -5.546 3.22e-08 ***
## carat_size:colorK    -21.4477     2.6978  -7.950 2.72e-15 ***
```

```
##
## Residual standard error: 7.058 on 2674 degrees of freedom
## Multiple R-squared:  0.8649, Adjusted R-squared:  0.8641
## F-statistic:  1141 on 15 and 2674 DF,  p-value: < 2.2e-16
```

```
gf_point(sqrt(price) ~ carat_size, color = ~color, data = Diamonds) %>%
  gf_lm() %>%
  gf_labs(x = "Carat Size", y = "sqrt(Price)") +
  ylim(30, 100)
```