Intro Regression In R

Kate Freier

2025-09-20

Table of contents

Preface

TBD

1 Visualize the Relationship

1.1 Scatterplots

Ever wondered how to tell if two things are related—like hours spent studying and the number of snacks consumed? If you haven't worked with them before, it is now time for you to meet the scatterplot. A scatterplot is a graph that displays pairs of values for two quantitative variables. Each point represents one observation, with its position determined by the values of the two variables—one on the x-axis (horizontal), the other on the y-axis (vertical).

- **X-axis:** Usually the independent variable (the one you think is doing the influencing).
- **Y-axis:** Usually the dependent variable (the one you think is being influenced).

Consider Figure 1.1 below showing the relationship between vehicle weight and fuel efficiency for 32 cars from 1973-1974 as reported by Motor Trend magazine.

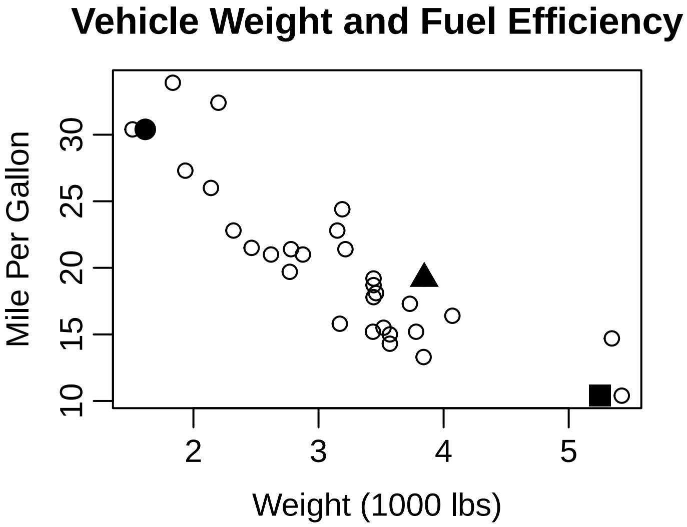


Figure 1.1: Vehicle weight and fuel efficiency

Three points have been highlighted with large solid plotting characters. The three points correspond to the data describing the Cadillac Fleetwood, Honda Civic, and Pontiac Firebird.

```
                      wt  mpg
Cadillac Fleetwood 5.250 10.4
Honda Civic        1.615 30.4
Pontiac Firebird   3.845 19.2
```

Starting with the Cadillac, where shoulld that car's data be noted in the scatterplot of Figure 1.1? Reading across the x-axis for weight, we want the point to be a bit past 5 at 5.25 since the axis is measured in terms of 1,000 lbs. From X=5.25 we then move upward to a Y corresponding to 10.4 mpg. Since 10.4 is one of the smallest mpgs in the dataset we don't move up much. X=5.25 and Y=10.4 intersect at the location of the solid square in the plot. The solid square marks the Cadillac Fleetwood. The solid circle belongs to the Honda Civic, and the Pontiac Firebird is marked by the solid triangle.

By showing every observed combination of your two quantitative variables, a scatter plot provides a quick view of the spread of each variable separately and a view of how they relate to each other. Looking at Figure 1.1 you can easily see the Cadillac Fleetwood had one of the lowest fuel efficiency levels and that the most efficient cars from the data achieved approximately 35 mpg. The figure also shows you that the weight of vehicles ranges from about 1,500 lbs up to about 5,500 lbs and that there seem to be a lot of cars right around 3,500 lbs, but hardly any in the 4,000 to 5,000 lb range.

The general downward slope of points going from the upper left corner down to the lower right corner shows the relationship between vehicle weight and fuel efficiency is exactly as we'd expect: heavier cars don't get as many miles on a gallon of gas as lighter cars. This direction of the relationship, assuming weight influences fuel efficiency, is why weight was plotted on the x-axis as the independent variable and miles per gallon was plotted on the y-axis as the dependent variable.

In R

The scatterplot shown in Figure 1.1 was completed using simple base R graphics and the mtcars data contained in the datasets library and included in the default install of R. Try the code for yourself:

```r
plot(mtcars$wt, mtcars$mpg,
     xlab="Weight (1000 lbs)", ylab="Mile Per Gallon",
     main="Vehicle Weight and Fuel Efficiency")

points(mtcars$wt[c(15,19,25)], mtcars$mpg[c(15,19,25)],
       pch=c(15,16,17), cex=1.5)

identify(mtcars$wt, mtcars$mpg, rownames(mtcars))
```

That last line of code, the `identify` function, allows you to generate labels for points in your scatterplot by clicking on points of interest. When you are done labeling points, hit the **stop** button or the **esc key** on your keyboard.

On Your Own

1. Consider the `cars` dataset built into R via the datasets library. This dataframe does not provide information about different models of car like the mtcars dataset seen in Figure 1, instead it shows 50 measurements from an experiment that wanted to explore the stopping distance of a car traveling at various speeds. There are two columns in the data frame: speed, measured in miles per hour, and stopping distance, measured in feet.

   a. Which one should be the X in your scatterplot and which should be the Y? Why?

   b. Create your scatterplot. Be sure to include relevant axis labels.

   c. What can you learn from your scatterplot about the speeds used in the experiment?

   d. What can you learn from your scatterplot about the stopping distances that were observed in the experiment?

   e. What relationship between stopping distance and speed do you see in the scatterplot? Explain.

   f. Based on your scatterplot, would a stopping distance of 60 ft be unusual for a car traveling at 10 mph? What about for a car traveling 20 mph? Explain.

2. Use the help menu to learn about the USArrests data frame in R.

   ```
   ?USArrests
   ```

   a. Create a scatterplot with the rate of arrest for assault as the independent variable and the rate of arrest for murder as the dependent variable.
   b. Using your scatterplot, what is your estimate for the highest murder arrest rate in the US? Does it pair with the highest arrest rate for assault?
   c. From your scatterplot, what is more common: a murder arrest rate higher than 10 per 100,000 or lower than 10 per 100,000?
   d. What would you guess is the arrest rate for murder in a state with an assault arrest rate of 100 per 100,000? Explain.
   e. What does your scatterplot indicate is the relationship between a state's arrest rates for assault and murder?
   f. Using the identify function, what state corresponds to the point at X of about 240 and Y of about 6? What two states have the highest arrest rates for assault? Which state has the highest rate of arrest for murder?

1.2 Advanced Scatterplotting

1.2.1 Jittering

Scatterplots are essential tools for visualizing relationships between two quantitative variables. However, when multiple data points share the same or very similar values, they can overlap on the plot, making it difficult to assess true distributions and where the density of the data lies. This phenomenon is known as overplotting.

Jittering is a technique used to address overplotting by adding a small amount of random noise to the position of each point. This noise is usually applied to one or both axes, causing points that would otherwise overlap to appear side by side.

Benefits of Jittering Include

- **Revealing Hidden Data Density**

  Without jittering, overlapping points may appear as a single point, obscuring the actual number of observations at that location. Jittering spreads these points apart, making clusters and concentrations more visible.

- **Improving Data Interpretation**

  By making individual points distinguishable, jittering allows viewers to better estimate the frequency of specific values and identify patterns or outliers that would otherwise be hidden.

- **Enhance Visual Clarity**

  Jittering reduces visual clutter caused by overplotting, resulting in a clearer and more informative plot. This is especially useful for categorical or discrete variables, where many data points may share identical values.

Though jittering includes adding noise to data, the amount of jitter is typically small relative to the scale of the data being jittered to ensure that the overall structure and trends of the data remain intact. This preserves the interpretability of the scatterplot while providing a more accurate representation of the data distributions.

Jittering is particularly helpful when one or both axes represent categorical or discrete variables, as these often lead to many overlapping points. For example, consider Figure 1.2 showing data from the same `mtcars` library seen in Figure 1.1. On the left you see the number of engine cylinders in each car plotted against the vehicle weight. As you might have suspected, the heavier cars tend to have more cylinders. Since cylinder count is discrete, the scatterplot of the raw data has a vertical stripe look to it as all values are equal to four, six, or eight. By adding small random noise to the cylinder count for the graph on the right of Figure 1.2, it becomes easier to see each vehicle in the plot. Particularly in the case of eight cylinders, several points in the 3,200 to 4,000 lb range are now visible where before the points were stacked on top of each other.

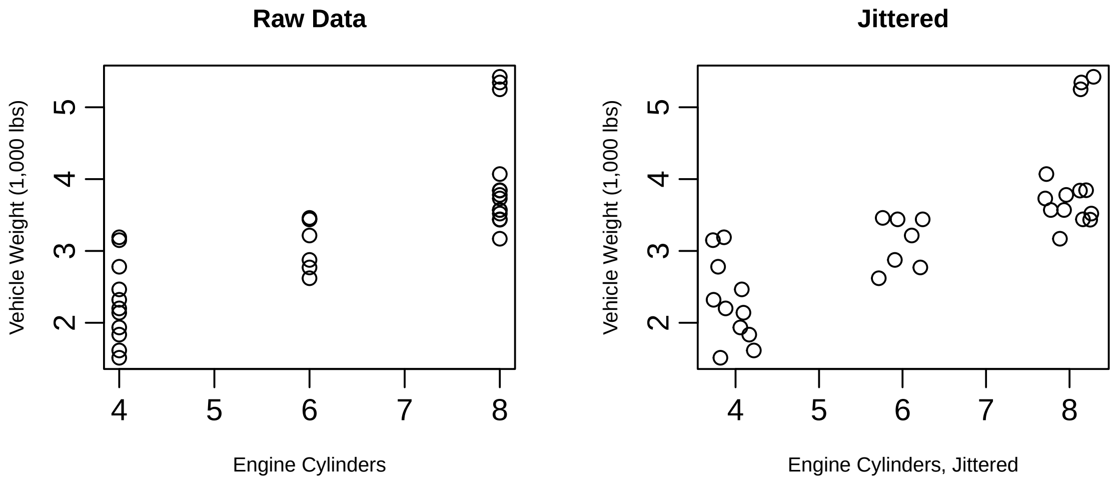


Figure 1.2: Vehicle Engine Size and Weight

1.2.2 Adding a Third Variable

As we've seen, scatterplots are great at showing two variables - but what about three? Three dimensional scatterplots are possible with today's technology and R will happily generate them. Below you'll find a screenshot of a 3D scatterplot showing the weight, mpg, and cylinder count from cars in the `mtcars` data frame. If you run the code below for yourself, you'll generate this plot in a form that allows you to click and drag the plotting box to rotate into different viewing angles.

```
library(rgl)
plot3d(mtcars$wt, mtcars$mpg, mtcars$cyl)
```

This 3D scatterplot functionality is certainly fun to play with, and it can be useful as you explore data relationships, but it has clear limitations. When working with stationary forms of communication like printed paper, a rotating scatterplot just doesn't work. Putting an animated 3D scatterplot into presentation slides can work, but it often just creates a distraction.

Incorporating information about your third variable into your plotting character is often a better choice especially if one of your variables is discrete or categorical. Figure 1.4 illustrates how color of plotting character can be used for a third variable that is either discrete or

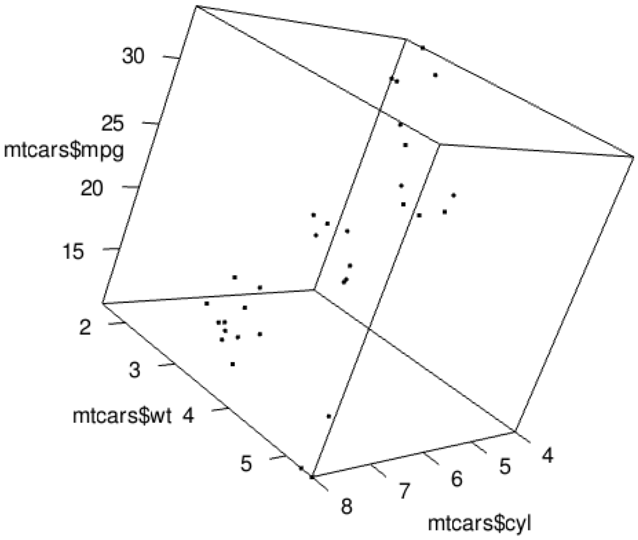


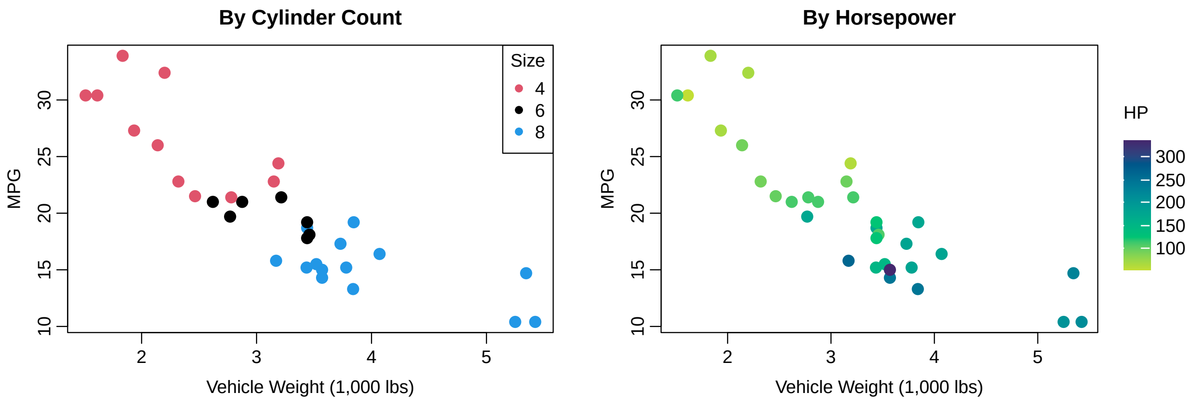
Figure 1.3: 3D Scatterplot Example

continuous. On the left of Figure 1.4 you see Figure 1.1 redone with color coded dots indicating the cylinder count of the engine. On the right you see color coding indicating the horsepower of the engine.

A few things to keep in mind when using color:

- A non-trivial segment of the population has some form of color blindness. The inability to clearly distinguish between red and green is the most common form of color handicap so avoid creating plots that rely on distinguishing red dots from green ones.
- If your work is going to be printed in black and white, you'll want to use colors that will still be distinguishable in grey scale. Using color combined with plotting character changes is a good strategy for keeping your work clear when printed without color.
- colorbrewer2.org is a good source for information on simple color schemes that are color-blind friendly and grey scale safe.
- If using a continuous color scale, consult resources on Viridis (the scale used on the right side of Figure 1.4) or other color scales designed with color blindness in mind. R provides good information to consider here: https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html.

Plotting character differences don't usually pop to our eyes quite as clearly as color, but sometimes they're the better choice. Consider the case of a plot that has points for both male and female patients - using "M" and "F" as your plotting characters quickly distinguishes between

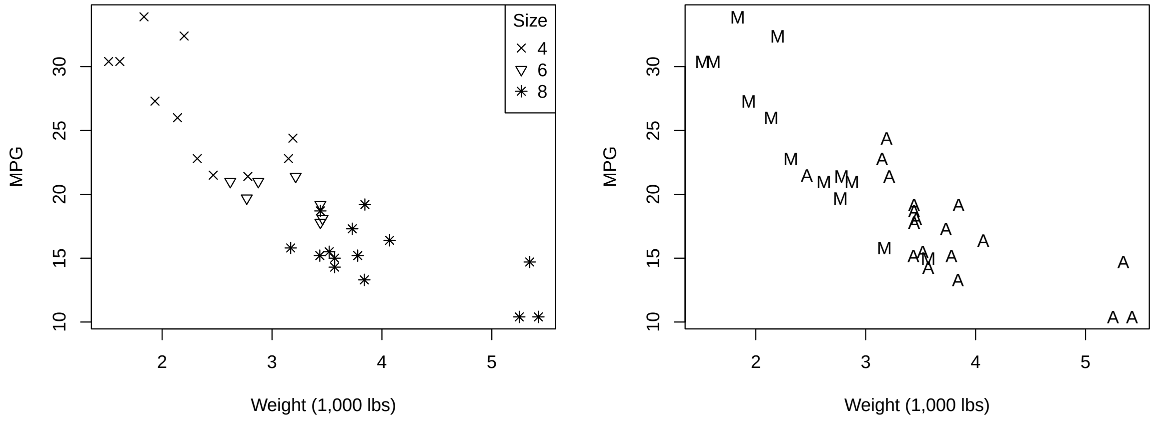


(a) Discrete Color

(b) Continuous Color

Figure 1.4: Color for third variable

the two groups and does so in a way that makes a legend less essential for understanding. Our brains are well trained to spot letter and numeric characters so you can use that to your advantage - think of "Y" and "N" for yes/no situations, "G" and "B" for good/bad, or even "1", "2", "3", "4" if you have groups that are readily identified by a number like 1st, 2nd, 3rd, or 4th year of school. Figure 1.5 shows you vehicle weight and fuel efficiency with plotting character indicating if the vehicle transmission is manual (M) or automatic (A). Avoid letter combos like "O" and "C" for open/closed though - some letters can look the same too easily especially if partially obscured by other points.



(a) By Cylinder Count

(b) By Transmission

Figure 1.5: Plotting character options

1.2.3 Scatterplot Matrix

When your data has more than two columns and you want to get a quick idea of all possible two-variable relationship, a scatterplot matrix is your answer. A **scatterplot matrix** (sometimes called a "pairs plot") is a grid of scatterplots that visualizes relationships between each pair of numerical variables in a dataset. Each cell in the matrix displays a scatterplot comparing a pair of variables, with one variable plotted along the x-axis and the other along the y-axis.

Figure 1.6 shows a scatterplot matrix of five quantitative variables from the `mtcars` data frame: MPG, engine displacement, horsepower, rear axle ratio, and weight. Because of the way it is formatted, every pair of variables is plotted twice. This allows you to see engine displacement on the X-axis with MPG on the Y-axis in the top row, second position and the reverse in the first position of the second row.

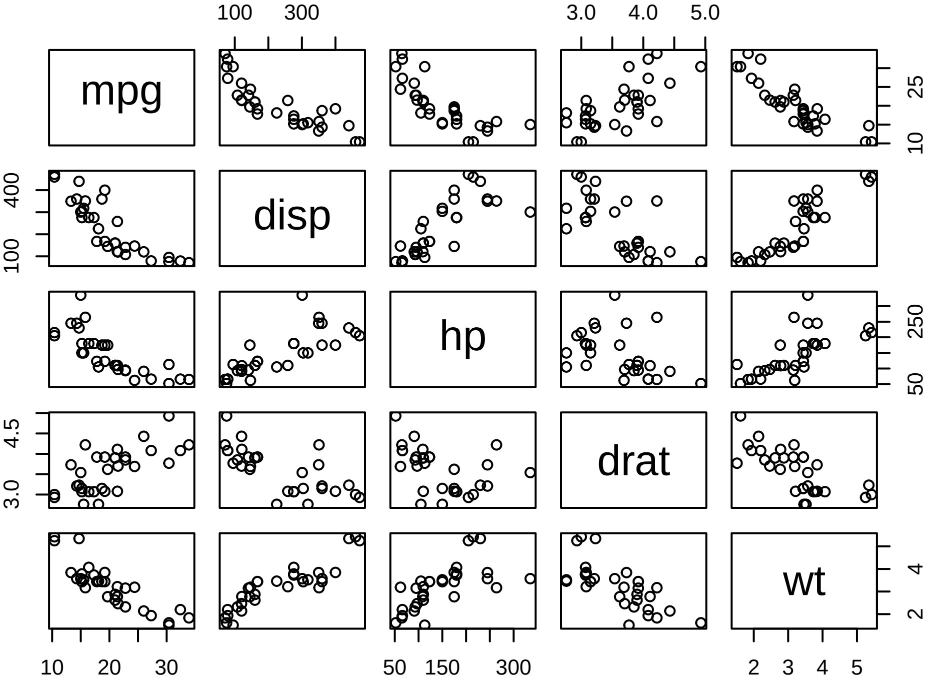


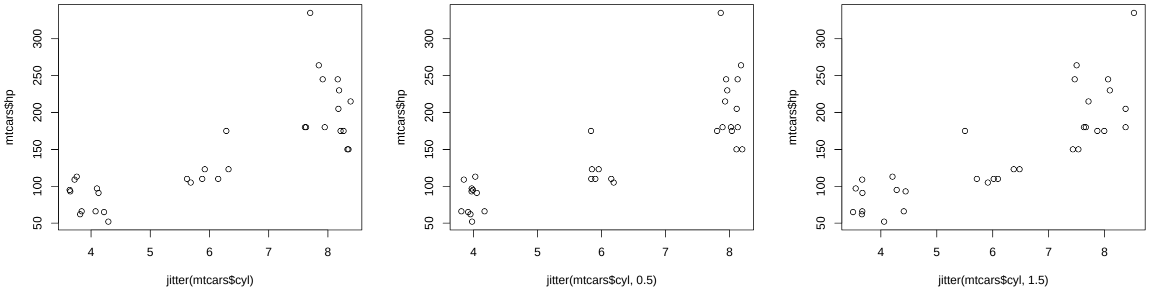
Figure 1.6: Scatterplot Matrix Example

A scatterplot matrix allows for a comprehensive, at-a-glance view of how all variables in your data frame relate to one another, making it a valuable tool in exploratory data analysis. You can quickly see from Figure 1.6 which pairs are most strongly related to each other, which have positive upward-sloping trends, and which are downward-sloping.

In R

To jitter a variable for your plot, you can use the `jitter` function. The jitter function can work with simply the variable you wish to jitter as the only input, or can take additional arguments to specify the amount of jitter you want to apply. Consider each of the jittered graphs below in Figure 1.7 showing a the number of cylinders in a car's engine plotted against the horsepower of the vehicle. The graphs were created using the shown code.

```r
plot(jitter(mtcars$cyl), mtcars$hp)
plot(jitter(mtcars$cyl, .5), mtcars$hp)
plot(jitter(mtcars$cyl, 1.5), mtcars$hp)
```



(a) Default jitter

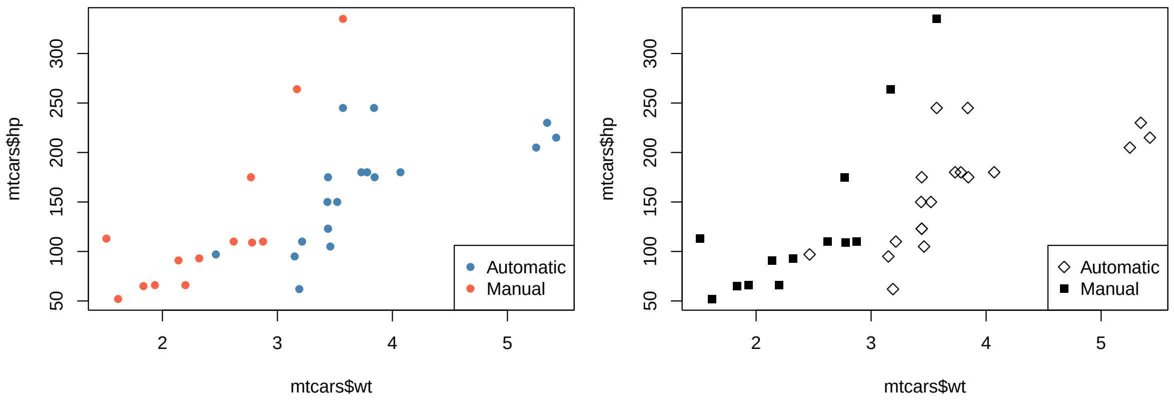
(b) Half Jitter

(c) 1.5x Jitter

Figure 1.7: Three jitter sizes

Adding color or changing your plotting characters can be done inside your plot function call using the `col` and `pch` inputs. Specifying `col` is for color, and `pch` is for plotting character. As you can see below, sometimes you might get a little creative with arithmetic to pull out the numbered plotting characters you're after. The numbers one through ten can pull out colors for you as well, or you can specify color names as done here, or even hex code color references.

```r
par(mar=c(4,4,1,1))
plot(mtcars$wt, mtcars$hp, col=c("steelblue", "tomato")[mtcars$am+1], pch=16)
legend("bottomright", c("Automatic", "Manual"),
       col=c("steelblue", "tomato"), pch=16)

plot(mtcars$wt, mtcars$hp, pch=(mtcars$am+5)+mtcars$am*9)
legend("bottomright", c("Automatic", "Manual"), pch=c(5, 15))
```



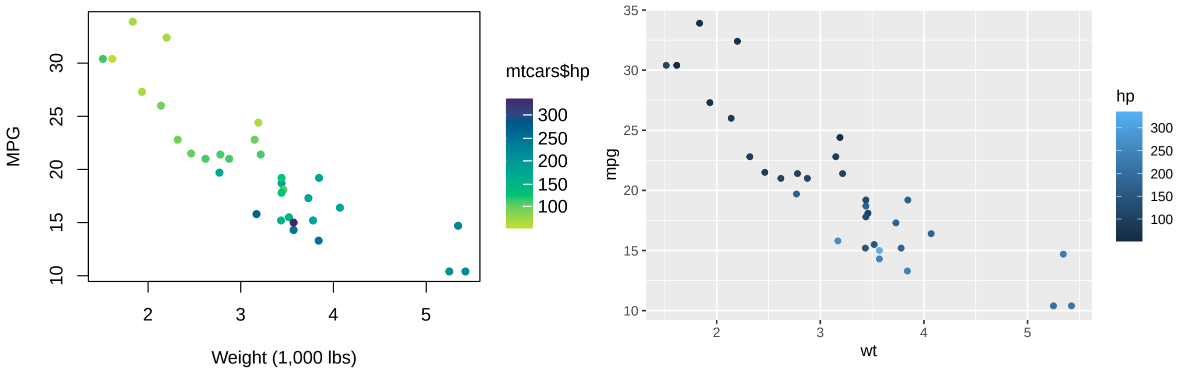
(a) Transmission type by color

(b) Transmission type by plotting character

Figure 1.8: Transmission

The continuous color approach is best done going beyond the base R plot command. Options include using the `plt` function in the `tinyplot` library or switching to ggplot graphics and the `ggplot2` library. Both of these options are shown in Figure 1.9.

```r
library(tinyplot)
plt(mtcars$wt, mtcars$mpg, by=mtcars$hp,
    xlab="Weight (1,000 lbs)", ylab="MPG", pch=16)

library(ggplot2)
ggplot(data=mtcars, aes(x=wt, y=mpg, color=hp))+
  geom_point()
```



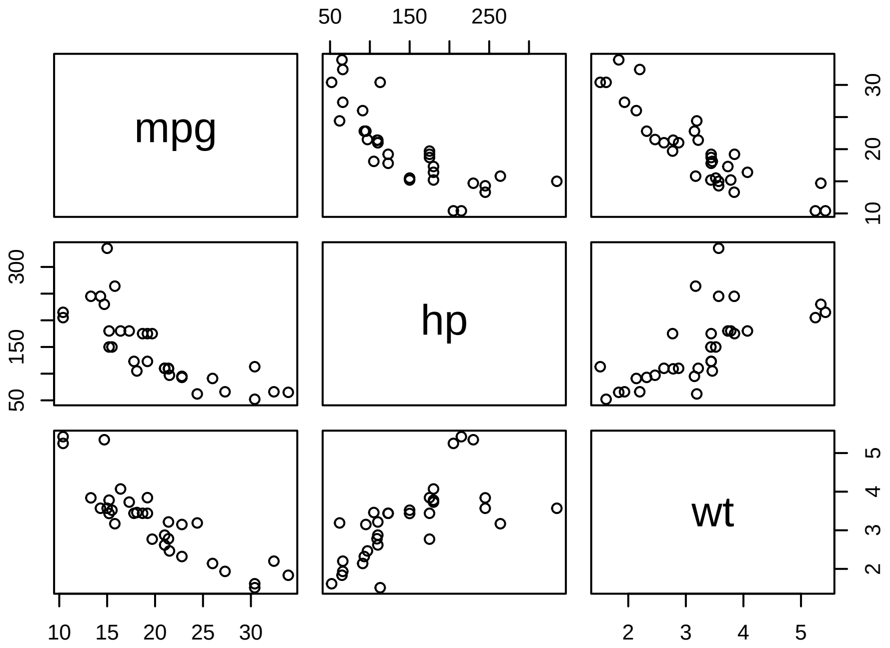
(a) tinyplot

(b) ggplot

Figure 1.9: Two R plotting packages options

Creating a scatterplot matrix is very straightforward provided your data is in a data frame. Using the `plot` command on a data frame will automatically create a scatterplot matrix. If you want only a subset of the columns, specify the columns you want in square brackets.

```
plot(mtcars[,c(1,4,6)])
```



On Your Own

1. Consider the `trees` dataset built into R via the datasets library. This dataframe gives the girth, height, and volume of Black Cherry trees.

    a. Create a scatter plot matrix of the tree data. Based on your plot, if you want to estimate a tree's volume, what is more helpful to know: the tree's girth or the tree's height? Explain your choice.

    b. Create a scatter plot of tree height and volume. Use plotting character 16, a solid circle for trees with a girth greater than 13 and an empty triangle for trees with a girth less than 13. Include a legend. What do you see of interest in this plot?

2. Consider the `swiss` dataset built into R via the datasets library. Use the help menu to learn about this data covering fertility and socioeconomic indicators over 47 provinces in 1888.

   a. Create a scatterplot matrix of all columns in the the swiss data frame. Which included socioeconomic factors, if any, are positively related to fertility rate?

   b. Using your scatterplot matrix from part a, which socioeconomic factors, if any, are negatively related to fertility rate?

   c. Create a scatterplot with the percent of males in agricultural occupations on the X-axis and fertility rate on the Y-axis. Use plotting character 16, the solid circle. For provinces with a population that is less than 50% Catholic, make the circles orchid. For provinces with a population that is over 50% Catholic, make the circles royalblue3. Do you see any interesting relationships in your graph? Explain.

   d. Create a scatterplot with the percent of males in agricultural occupations on the X-axis and the rate of education beyond primary school on the Y-axis. Use plotting character 16, the solid circle and color it using a continuous color scale for percent Catholic. Do you see any interesting relationships in your graph? Explain.

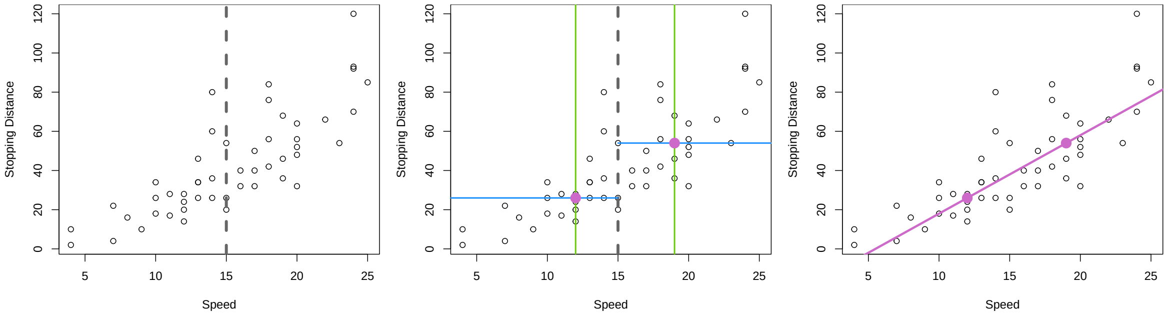
2 Add a Line

2.1 Describing the Relationship with a Line

With a scatterplot in front of you and a linear trend evident the obvious next step is to draw a line on your plot. We begin with a simple method for fitting a line to a scatterplot that is easy to compute in four steps:

1. Calculate the median, 1st quartile, and 3rd quartile, of your X variable.
2. Calculate the median Y value of points with a corresponding X value less than the median you found in step 1.
3. Calculate the median Y value of points with a corresponding X value greater than the median you found in step 1.
4. Add the points corresponding to the 1st quartile of X paired with the median calculated in step 2 and the 3rd quartile of X paired with the median calculated in step 3 to your scatterplot. Draw a line connecting those two points.

That's it. You're cutting the data in half from left to right along the X-axis, then finding the median X and Y on each side to create two points, then connecting the points with a line. What does that look like? Figure 2.1 below walks through the process as applied to the `cars` speed and stopping distance data.



(a) Find median X, divide (b) Median X & Y each side (c) Connect two points

Figure 2.1: Median-Median Line Steps

First, the data is split into a left and right half based on the median vehicle speed, shown with the vertical grey dashed line. In the second plot, the 25th and 75th percentiles of speed are marked with green vertical lines and the median stopping distances for the two halves are marked with horizontal blue lines. Orchid dots mark the intersection of the medians. These two key dots are then joined with a line for the third graph on the right.

In R

A big part of the beauty of this method for drawing a line on a scatterplot is that it is simple. The ideas involved are nothing beyond quartiles and the algebra of connecting two points with a line. To find quartiles in R you'll use the `quantile` function and base R arithmetic will get your through the rest of it.

Here's the code to fit the line to the `cars` data frame shown in Figure 2.1.

```r
# make initial scatterplot to begin
plot(cars$speed, cars$dist, xlab="Speed", ylab="Stopping Distance")

# step 1: calculate the median, 1st quartile and 3rd quartile of X, weight
quartx<-quantile(cars$speed, c(.5, .25, .75))

# step 2: calculate the median Y of points to the left of the median X
med_y_left<-median(cars$dist[cars$speed<=quartx[1]])

# step 3: calculate the median Y of points to the right of the median X
med_y_right<-median(cars$dist[cars$speed>=quartx[1]])

# step 4: Add points and line connecting them
points(quartx[2:3], c(med_y_left, med_y_right), pch=16, col=2)

slope<-(med_y_right-med_y_left)/(quartx[3]-quartx[2])
intercept<-med_y_left-(slope*quartx[2])
abline(intercept, slope, col="orchid3", lwd=3)
```

On Your Own

1. Create a scatterplot of vehicle weight and horsepower from the `mtcars` data frame.
   a. Overlay your plot with a median-median line in red.
   b. Interpret the median-median line in context.
2. Explore the variability in the median-median line approach through the following steps

a. Write a function that takes in an X and a Y, then generates the slope and intercept for the median-median line.
b. Use your function on the `cars` data frame to identify the median-median line describing the relationship between vehicle speed and stopping distance.
c. Sample the rows of the `cars` data frame with replacement to create a new data frame with speed and stopping distance that is the same size as the original. Square brackets and `sample(1:nrows(cars), replace=TRUE)` will help you out with this. Once you have your new data frame, apply your median-median line function. How did the intercept and slope change?
d. Repeat the steps of part c 200 times in a loop, saving the new intercept and slope each time. Then generate histograms of your 200 intercepts and 200 slopes. How variable is the median-median line from one data sample to the next?

2.2 Residuals

Let's consider a relatively small data set. The `bac` data from the `openintro` (Çetinkaya-Rundel et al. 2024) R library gives data from 16 student volunteers at Ohio State University who drank beer and then had their blood alcohol content measured.

Figure 2.2 shows the 16 data points from this study (one from each student) with the beer consumption level jittered for clarity. The red line is the result when using the Tukey median-median line method to describe the relationship between blood alcohol content (BAC) and beer consumption.

Obviously this red fit line does not exactly pass through every point. From the scatterplot it is clear that while BAC and beer consumption amounts are related, the relationship is not perfect and no straight line could perfectly pass through all 16 points. You'll also see that each point has a thin line drawn vertically connecting it to the red line. The lengths of these lines are considered the error of the red line fit. These errors have a special name: ***residuals***.

A residual is the difference between the actual observed value of the dependent variable (y) and the value predicted by the linear fit ($\widehat{y}$).

Mathematically, it is expressed as: $Residual = y - \widehat{y}$ or specific to the i^{th} observation, $r_i = y_i - \widehat{y}_i$, where y_i is the observed value and $\widehat{y}_i$ is the value predicted from the line.

Key points about residuals:

- A positive residual means the observed value is above the predicted value (the model underestimated).

- A negative residual means the observed value is below the predicted value (the model overestimated).

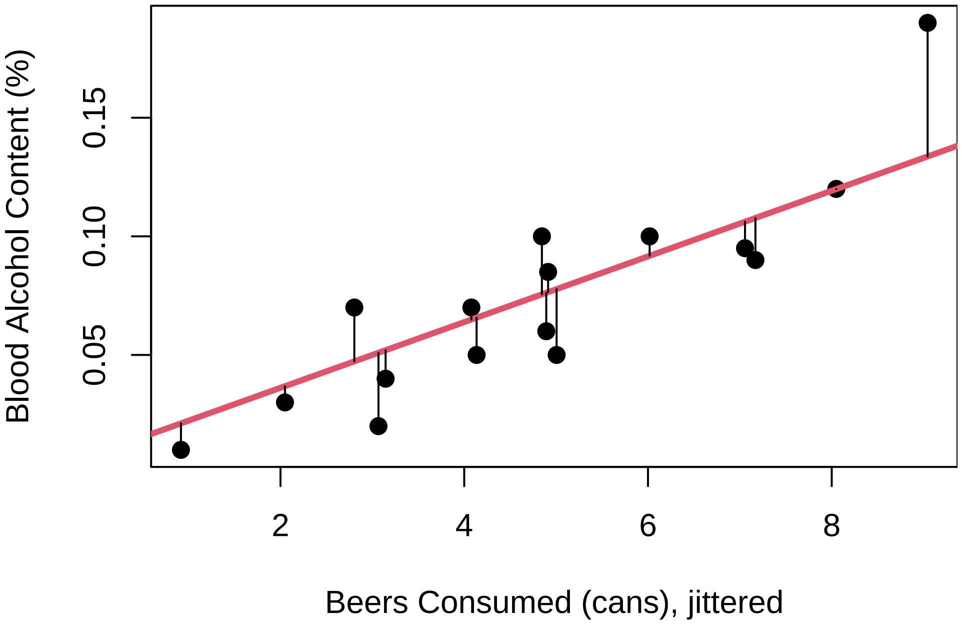


Figure 2.2: Residual diagram

- The residuals represent the vertical distances between the data points and the regression line on a scatterplot. The errors are parallel to the y-axis, not projections meeting the linear fit at a right-angle.

2.3 Least-Squares Regression Line

With the idea of a residual established, we can move beyond the simple median-median line and define a new line of fit: the least-squares regression line. With this approach our fit line is defined as the line that minimizes the sum of the squared residuals.

Why the sum of *squared* residuals and not simply the sum of residuals? Recall that residuals can be both positive and negative. Errors from points above the line and errors from points below the line would cancel each other out in a simple sum of residuals.

Why not the sum of the absolute value of residuals? Well… calculus. Think back to your calculus classes. What was more straight-forward: finding the minimum of a smooth continuous polynomial function or finding the minimum of a function with a kink in it caused by an absolute value?

You can also see the squaring as a nice feature as it puts an added penalty whenever your line is far off. Big residuals squared become even bigger errors in the sum you're minimizing.

Some notation for all this is needed now. Our goal is to model the relationship between a single independent variable x and a dependent variable y with a line in the form:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Where

- y = Dependent (response) variable
- x = Independent (predictor) variable
- β_0 = Intercept (value of y when $x = 0$)
- β_1 = Slope (change in y for a one-unit change in x)
- ε = Error term that captures deviations from the line (residuals)

Minimizing the sum of squared errors means minimizing:

$$\sum (y_i - (\beta_0 + \beta_1 x_i))^2$$

$$= \sum (y_i - \beta_0 - \beta_1 x_i)^2$$

By differentiating with respect to β_0 and then separately differentiating with respect to β_1 , setting each to 0 and solving for the minimums you can find that the least-squares estimate for β_0 and β_1 are:

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of x and y, respectively.

Figure 2.3 shows the least-squares regression line added to the plot from Figure 2.2. The red line is the Median-Median fit of the data, the dashed blue line shows the least-squares regression fit.

Now a question: If you swap your dependent and independent variables and fit a line to X~Y instead, will you get the same linear equation just solved for X instead of solved for Y? Nope. When BAC is fit as a function of beer consumption you get that $BAC = -0.0127 + 0.0180 \times beers$. If instead you fit beers as a function of BAC, then, $beers = 1.5288 + 44.5252 \times BAC$ which when solved for BAC comes to $BAC = -0.0343 - 0.0225 \times temperature$. Both of these are shown on Figure 2.4 to show how different the results are:

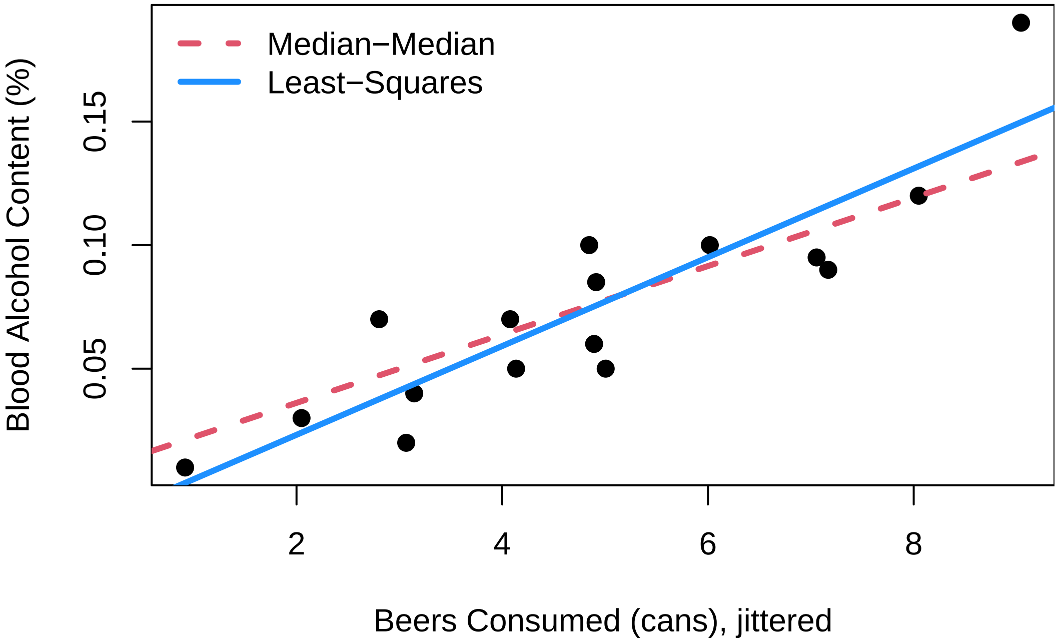


Figure 2.3: Median-Median and Least-Squares Lines

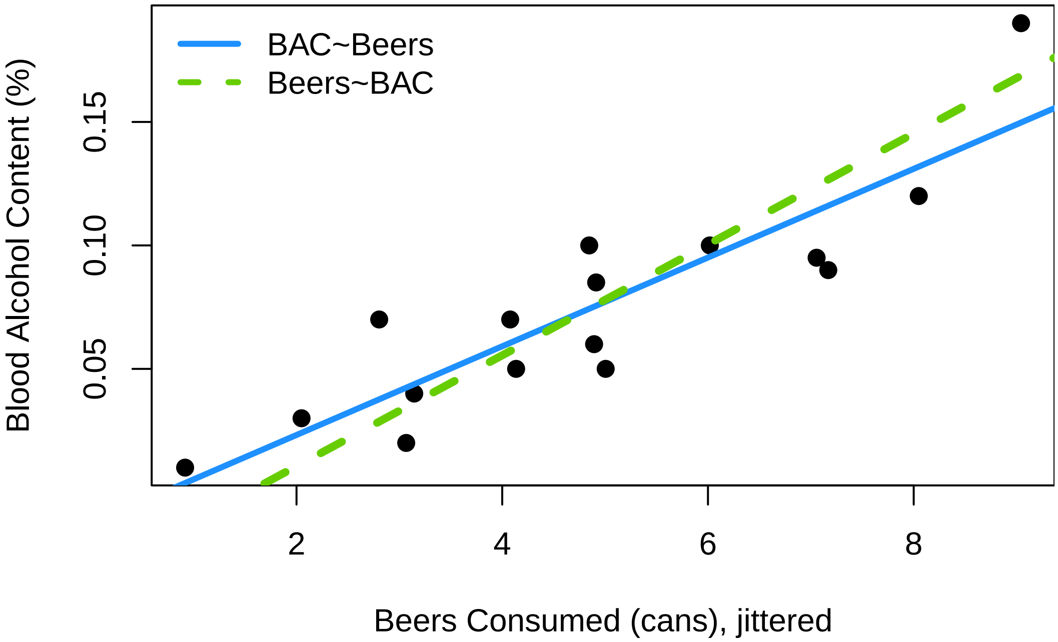


Figure 2.4

Why are they so different? It is all because of how residuals are defined. Remember our least-squares regression fit is designed to minimizes the sum of squared residuals and residuals are defined as the error measured parallel to the y-axis. When you change which term is y, you change the error measure that matters.

It is worth noting that every regression line will pass through the point $(\bar{x}, \bar{y})$ and because of that, the point at which the two lines intersect is, and always will be, $(\bar{x}, \bar{y})$. With this data, that means $(4.812, 0.07375)$.

In R

Least-squares linear regression is a foundational technique in statistics and data science. Some consider it the first step you'll take into the world of machine learning. As such, it is built into base R and those formulas above are not ones you'll need to memorize. R will do all the heavy lifting. Here's the code to fit the blue regression line in Figure 2.3 above and pull out the slope and intercept.

```
blue_model<-lm(bac~beers, data=bac)
blue_model$coef
```

```
(Intercept)       beers
-0.01270060  0.01796376
```

You can even draw the regression line on your scatterplot using the model object directly with the `abline` function. Here's the code to create the plot and insert the line in the color blue with a line width of three:

```
plot(bac$beers, bac$bac)
abline(blue_model, col=4, lwd=3)
```

On Your Own

1. Create a scatterplot of vehicle weight and horsepower from the `mtcars` data frame.

   a. Overlay your plot with a least-squares regression line in red.
   b. What is the equation of your fitted line?

2. Create a scatterplot of vehicle speed and stopping distance from the `cars` data frame.

   a. Overlay your plot with a least-squares regression line in the color of your choice.
   b. What is the equation of your fitted line?

3. Fit a least-squares model predicting a newborn's father's age as a function of the babies' mother's age using the `births14` data frame in the `openintro` library.

    a. What is the equation of your fitted line?
    b. Make a scatterplot of the data with your fitted line superimposed on top.
    c. Now fit the model predicting the mother's age as a function of father's age. What is the equation of the fitted line?
    d. Adjust your new model by solving for father's age and add the new model to your scatterplot.
    e. Confirm the point of intersection is at $(\bar{x}, \bar{y})$.

4. Explore the variability in the least-squares regression line approach through the following steps a. Identify the slope and intercept of the fitted regression line predicting car stopping distance as a function of speed from the `cars` data frame. (yes, this is the same as #2 above) b. Sample the rows of the `cars` data frame with replacement to create a new data frame with speed and stopping distance that is the same size as the original. Square brackets and `sample(1:nrows(cars), replace=TRUE)` will help you out with this. Once you have your new data frame, find the new least-squares slope and intercept. How did the line coefficients change? c. Repeat the steps of part b 200 times in a loop, saving the new intercept and slope each time. Then generate histograms of your 200 intercepts and 200 slopes. How variable is your regression line from one data sample to the next? d. Plot car speed and stopping distance in a scatterplot and overlay 10 of your fitted regression lines from part c. When you look at the 10 lines, how do they differ from eachother? Do they tend to be parallel? Is the variability in line placement the same across the range of x, or are there areas of x where the lines are more or less consistent?

2.4 SSTotal, SSRegression, and R^2

In Section 2.2 we discussed residuals as being the measure from each data point to the fitted line. There are two other important metrics in regression to be aware of.

Below in Figure 2.5 is the same student alcohol data with the blue least-squares line fit earlier. The horizontal grey dashed line is at 0.07375, the mean blood alcohol level across the 16 students.

The zoomed-in section of Figure 2.5 shows each of the three key components to be aware of in regression modeling drawn out for the point corresponding to our biggest drinker as an example.

- As you know from Section 2.2, the thin black line is the residual; the difference between the fitted regression model in blue, and the actual observed point. By design, our least-squares fit in blue minimizes the sum of the squared lengths of the black residual lines for all points. The sum of the squared errors is known as SSE, SSError, or SSResidual.

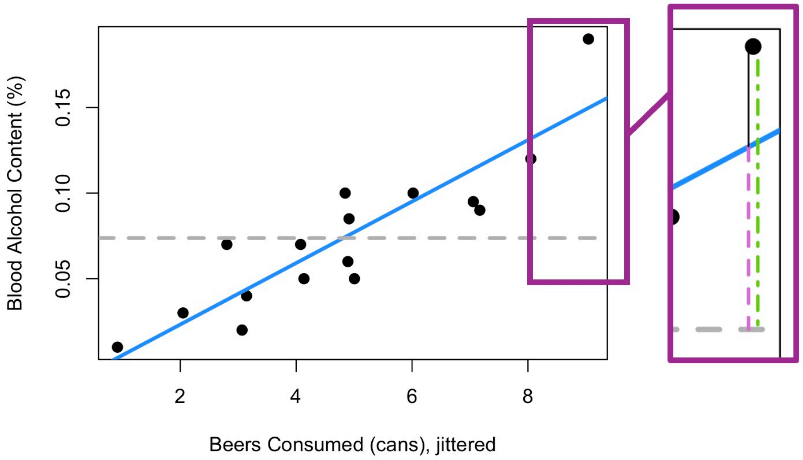


Figure 2.5: SST, SSE, and SSR

- The green dot-dashed line is the distance from the observed point to the observed mean Y. The sum of these distances squared is known as Sum of Squares Total, SSTotal, or SST. This is a measure of the total variability of Y in your data set. The idea of the sum of squared distances to the mean as a measure of variability is not new: recall standard deviation from your intro stats course. By definition, the standard deviation of Y (a sample) is:

  $$\sqrt{\frac{\sum(y-\bar{y})^2}{n-1}}$$

  with the sum of squared distances to the mean right there in the numerator.

- The third line shown dotted in orchid is the distance from the mean of Y to the regression line fit.

  If beer consumption were unknown, our best guess for y_1 would be 0.07375, the overall average that corresponds to the horizontal grey line. However, using beer consumption and our fitted model, the best guess for y_1 is .14897, the fitted value corresponding to nine consumed beers calculated using the intercept and slope found earlier: $-0.0127 + (0.01796 * 9) = 0.1489$.

  The orchid dotted line shows this difference between the 0.1489 and 0.07375, and can be thought of as the value added by the linear model. It is graphically how much closer we are to the actual value of y_i because we used the linear relationship to X in our estimation. The sum of squares of these differences attributed to the regression model is known as Sum Squares Regression, SSReg, or SSR.

In summary:

$$SSError = SSE = \sum (y_i - \hat{y}_i)^2$$

$$SSTotal = SST = \sum (y_i - \bar{y})^2$$

$$SSRegression = SSR = \sum (\bar{y}_i - \hat{y}_i)^2$$

Clearly from the visual in Figure 2.5 these three Sum of Squares measurements are closely related. In fact

$$SSTotal = SSError + SSRegression$$

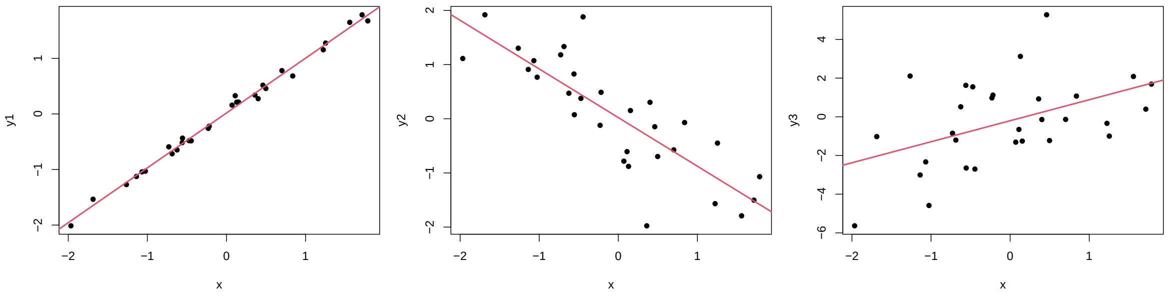
Typically this relationship is relied on when calculating SSRegression as SSTotal and SSError are easier to calculate directly.

To avoid potential confusion, throughout this book SSR will refer only to SSRegression and SSE will be used for the Sum of Squared Residuals. Be aware though that some sources may use SSR to refer to SSRegression or SSResiduals.

All of this leads us to R^2, the coefficient of determination.

$$R^2 = 1 - \frac{SSError}{SSTotal} = \frac{SSReg}{SSTotal}$$

Put into words, this means R^2 is the proportion of variability in Y that has been explained by your linear regression model. Note that R^2 must always be positive and between 0 and 1, inclusive. The R^2 value also has nothing to do with the direction of the relationship between X and Y, only the strength of that relationship. The slope may be upward or downward sloping, R^2 only cares how closely the points follow the line.



(a) $R^2 = .99$ (b) $R^2 = .68$ (c) $R^2 = .22$

Figure 2.6: R^2, Coefficient of Determination

Consider the three plots in Figure 2.6. On the left where the points closely follow the fitted red regression line, SSE is small, meaning SST and SSR are close to equal in value and R^2 is near 1 at 0.99. In the middle, the regression line does a fair job of describing the variability in Y so SSR is larger than SSE and the R^2 is 0.68. The downward slope of this middle graph is irrelevant. On the far right, the residuals are larger relative to the total variability of Y, so $\frac{SSE}{SST}$ is larger than in the other graphs making $1 - \frac{SSE}{SST}$ smaller and R^2 is only 0.22.

What constitutes a "good" R^2 varies by discipline. If your data is related to engineering and is the X-Y relationship is related to a law of physics for instance, R^2 should be very nearly 1. If however you are working in the social sciences and the variability in the X-Y relationship is about people being different from each other, then an R^2 of 0.4 might be super exciting. Context matters when evaluating R^2.

In R

Code below calculates the SST and SSE from the least-squares line predicting BAC using beer consumption. SSR is calculated two ways; by subtracting SSE from SST, and directly using the mean of crawling age and model fitted values.

```
blue_model<-lm(bac~beers, data=bac)

# calculate mean BAC
mean_bac<-mean(bac$bac)

# calculate fitted values for BAC
model_fit<-blue_model$coef[1]+(blue_model$coef[2]*bac$beers)

# renaming just to make later calculations clear
bac_obs<-bac$bac

# calculate SST
SSTotal<-sum((bac_obs-mean_bac)^2)
SSTotal
```

```
[1] 0.029225
```

```
# calculate SSE
SSError<-sum((bac_obs-model_fit)^2)
SSError
```

```
[1] 0.005849655
```

```
# calculate SSR, two ways
SSReg<-SSTotal-SSError
SSReg_long<-sum((model_fit-mean_bac)^2)

c(SSReg, SSReg_long)
```

```
[1] 0.02337535 0.02337535
```

Note the two approaches to SSR produced the same result. There was an easier way though. Just as `$coef` after your model name will retreive the fitted model coeffiecents, there are similarly intuitive ways to quickly pull the fitted values and the residuals:

```
summary(blue_model$fitted.values)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
0.005263 0.041191 0.077118 0.073750 0.099573 0.148973 
```

```
summary(blue_model$residuals)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
-0.027118 -0.017350  0.001773  0.000000  0.008623  0.041027 
```

And here are three ways to calculate R^2, the coefficient of determination.

```
opt1<-1-(SSError/SSTotal)

opt2<-SSReg/SSTotal

opt3<-summary(blue_model)$r.square

c(opt1, opt2, opt3)
```

```
[1] 0.7998407 0.7998407 0.7998407
```

On Your Own

1. Fit a least-squares model predicting vehicle hwy_mpg as a function of vehicle weight using the `cars04` data frame in the `openintro` library.

   a. What is the equation of your fitted line?
   b. Make a scatterplot of the data with your fitted line superimposed on top.
   c. Find SST, SSE, and SSR.
   d. Find R^2.

2. Fit a least-squares model predicting total SAT score as a function of high school GPA using the `satgpa` data frame in the `openintro` library.

   a. What is the equation of your fitted line?
   b. Make a scatterplot of the data with your fitted line superimposed on top.
   c. Find SST, SSE, and SSR.
   d. Find R^2.

3. Fit a least-squares model predicting a newborn's father's age as a function of the babies' mother's age using the `births14` data frame in the `openintro` library.

   a. What is the equation of your fitted line?
   b. Make a scatterplot of the data with your fitted line superimposed on top.
   c. Find SST, SSE, and SSR.
   d. Find R^2.

2.5 Using Regression Models

Great! We have a line! Now what? The two most common uses for a fitted regression line are:

- Using the slope as a descriptor of the relationship between X and Y
- Using the fitted line to predict future observations

Continuing our work with the student alcohol data, recall the fit:

$$bac = -0.0127 + (0.01796 \times beers)$$

The positive slope tells us that average BAC goes up as beer consumption goes up. Not surprising at all. Specifically, with a value of 0.01796, the model tells us the average BAC goes up by 0.01796 with every additional can of beer.

Using the full equation, we can also estimate that if someone drinks 4 beers, their blood alcohol level will likely be about 0.059 because $-0.0127 + (0.01796 \times 4) = 0.05914$. Similarly, a student who consumes seven beers likely has a BAC of about 0.113.

One important caveat for using your equation for estimation: extrapolation beyond the bounds of your observed data is not a good idea. What does that mean? Well think of a student with remarkably bad judgement who drinks 12 cans of beer. Our data doesn't actually include any students drinking more than nine beers so by using our model to estimate the BAC at 12 beers we are making the assumption that the linear trend continues beyond the scope of our plot. Maybe it does and this student has a BAC just a bit over 0.2, but maybe it doesn't and beers 10, 11, and 12 have a different impact than beers 1, 2, and 3, on increasing BAC. Estimations near the middle of our observed X range are where we feel the most confident; estimations beyond the range of observed X should be made with caution.

In R

You already know that `model_name$coef` will get you your model coefficients. From there you can get predicted values using your line with a little algebra. There's another way though; one that is particularly useful when you have more than one prediction to make. Here is how you can use the `predict.lm` function.

```r
# create a data frame with the x-values you want a prediction for, making
#   sure the column name matches the x name used when you created your model

new_data<-data.frame(beers=c(2,3,6.5))

# get your predicted values
predict.lm(blue_model, new_data)
```

```
         1          2          3
0.02322692 0.04119068 0.10406385
```

On Your Own

1. Fit a least-squares model predicting vehicle hwy_mpg as a function of vehicle weight using the `cars04` data frame in the `openintro` library.

   a. What is the equation of your fitted line?
   b. How do you expect hwy_mpg to change with a 100 lb increase in vehicle weight?
   c. What highway gas mileage would you expect for a car that weighs 3,802 lbs?
   d. What highway gas mileage would you expect for a car that weighs 1,700 lbs?
   e. Which estimate, d or e, do you feel better about? Why?

2. Fit a least-squares model predicting total SAT score as a function of high school GPA using the `satgpa` data frame in the `openintro` library.

    a. What is the equation of your fitted line?
    b. If Anne has a high school GPA of 3.3 and and Clark has a GPA of 3.8, how do you think Clark's and Anne's SAT scores compare? Explain with estimates of each SAT score using your model.

3. Fit a least-squares model predicting a newborn's father's age as a function of the babies' mother's age using the `births14` data frame in the `openintro` library.

    a. What is the equation of your fitted line?
    b. For each additional year of age for the mother, how much older do you anticipate the father to be?
    c. Grace's Mom was 29 when she was born, how old would you guess her father was?

2.6 Matrix Notation

Section 2.3 expressed the form of our fitted model in the notation of a simple linear function ($y = \beta_0 + \beta_1 x + \varepsilon$) but that isn't the only option. Our model could also be expressed using matrix notation as:

$$Y = X\beta + \varepsilon$$

Where

- Y is an $n \times 1$ column matrix containing your n observed outcomes
- X is an $n \times 2$ matrix with the first column full of 1s to correspond to the intercept term of the linear model and the second column containing the x values for your observed outcomes
- β is 2×1 containing β_0 and β_1 in separate rows
- ε is an $n \times 1$ column matrix of the error terms

So with the student alcohol data data:

$$Y = \begin{bmatrix} 0.1 \\ 0.03 \\ 0.19 \\ \vdots \\ 0.05 \end{bmatrix} andX = \begin{bmatrix} 1 & 5 \\ 1 & 2 \\ 1 & 9 \\ \vdots & \vdots \\ 1 & 4 \end{bmatrix}$$

To estimate β, the $\hat{\beta}$ is calculated by:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Which means your estimate for β is dependent of $X^T X$ having an inverse. No inverse, no $\hat{\beta}$. Putting this $\hat{\beta}$ back in our model equation, this means

$$\hat{Y} = X\hat{\beta}$$
$$\hat{Y} = X(X^T X)^{-1} X^T Y$$

We then define $X(X^T X)^{-1} X^T$ as H, the "hat matrix" because $HY = \hat{Y}$. H "puts a hat" on Y. The hat matrix will play a role later on in chapter XXXXXXX.

In R

You won't do it regularly, but it's not a bad exercise to work through fitting a linear model with matrix algebra rather than the `lm` shortcut offered by R.

```r
Y_mat<-matrix(bac$bac, ncol=1)
X_mat<-matrix(c(rep(1, 16), bac$beers), ncol=2)

XTX<-t(X_mat)%*%X_mat
XTX_inv<-solve(XTX)
XTX_invXT<-XTX_inv%*%t(X_mat)
beta_hat<-XTX_invXT%*%Y_mat

beta_hat
```

```
            [,1]
[1,] -0.01270060
[2,]  0.01796376
```

```r
lm_mod<-lm(bac~beers, data=bac)
lm_mod$coef
```

```
(Intercept)       beers
-0.01270060  0.01796376
```

Yay! They match, just as expected.

On Your Own

1. Fit a least-squares model predicting total SAT score as a function of high school GPA using the `satgpa` data frame in the `openintro` library.

   a. What is the $(X^TX)^{-1}$ matrix in the interim calculation of $\hat{\beta}$?
   b. Confirm the model fit using `lm` matches the fit using matrix algebra.
   c. What is the value in the 1,1 position of the hat matrix?

2. No data now, just linear algebra:

   a. What are the dimensions of the hat matrix?
   b. What does HH equal?

3 How sure are you?

Every confidence interval you learned in intro statistics has the same basic format:

$$\textit{point estimate} \pm \textit{multiplier} \times \textit{standard error}$$

Similarly, the hypothesis tests you've learned for means and proportions have test statistics in the general format of:

$$\frac{\textit{observed value} - \textit{hypothesized value}}{\textit{standard error}}$$

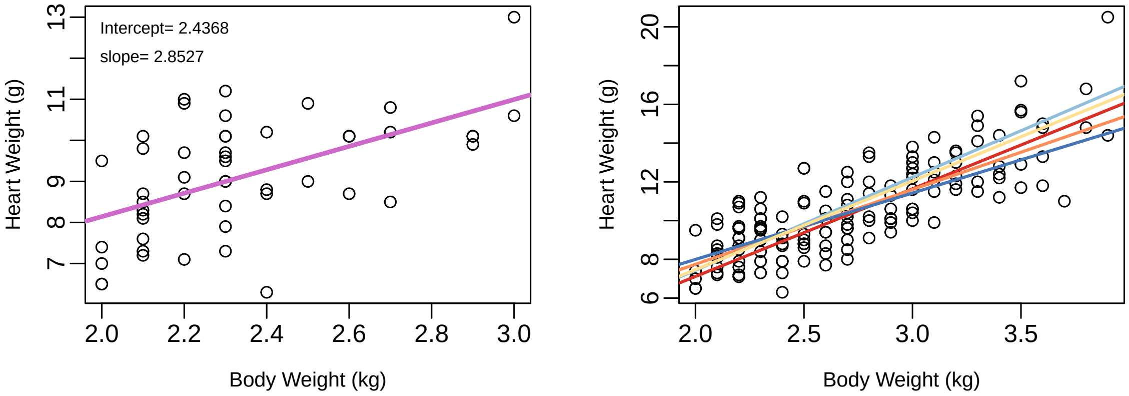
Both of these basic structures will continue in the realm of regression analysis.

3.1 Inference for Model Coefficients

You can now fit a regression model, but in many ways that's equivalent to being able to calculate a sample mean. The least-squares fit gives you a way to describe your sample data well, but the line alone doesn't tell you all you need to make statements about the population your sample came from. Just as you learned how to put a confidence interval on a sample mean in intro stat, you'll now put confidence intervals on your fitted model coefficients. Confidence intervals allow you to quantify how much uncertainty surrounds the estimates of the intercept (β_0) and slope (β_1) coefficients.

Consider the `cats` data frame available in the `MASS` library for R. This data frame includes the body weight (in kg) and heart weight (in grams) for 144 cats. The least-squares fit for the first 50 cats in Figure 3.1.

But what if a different sample of 50 cats had been selected? How different might the $\hat{\beta_0}$ and $\hat{\beta_1}$ be? Figure 3.1 (b) shows five different regression lines fit on five different samples of size 50 from the cat data superimposed on a scatterplot of all 144 cats in the data frame.



(a) Fit of first 50 cats

(b) 5 fits from 5 samples of 50

Figure 3.1: Cat Body and Heart Weight

As you see, each sample produces a slightly different line. The five intercepts range from -2.164 up to 1.141, and the five slopes range from 3.4268 up to 4.8005. Clearly which random sample you look at matters, and when this variability exists we need a confidence intervals to communicate this uncertainty, and hypothesis testing is needed to evaluate hypotheses.

3.1.1 Confidence Intervals

Intercept β_0

- The point estimate for the intercept in your linear model describing the relationship between X and Y is simply your $\hat{\beta_0}$.
- The multiplier used in your confidence interval will come from a t-distribution with $n-2$ degrees of freedom. Why $n-2$? You started with n data points (and n degrees of freedom), and then you lose one degree of freedom for each parameter you estimate. You estimated two parameters: β_0 and β_1, hence $n-2$.
- The standard error for $\hat{\beta_0}$ is $SE_{\hat{\beta_0}} = S_R\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum(x-\bar{x})^2}}$ where $S_R = \sqrt{\frac{\sum(y_i-\hat{y_i})^2}{n-2}}$, the standard deviation of the residuals.

Example with cat heart weight model from Figure 3.1 (a)

- Point estimate is simply $\hat{\beta}_0 = 2.4368$.
- With $n = 50$ there are $50 - 2 = 48$ degrees of freedom, so for a 95% confidence interval the multiplier will be $t_{.025,48} = 2.010635$.
- Using the components $\bar{x} = 2.34$, $\sum(x - \bar{x})^2 = 3.76$, and $S_R = 1.2007$, we can calculate $SE_{\hat{\beta}_0} = 1.2007\sqrt{\frac{1}{50} + \frac{2.34^2}{3.76}} = 1.4588$.
- Interval: $2.4368 \pm (2.010635 \times 1.4588) = (-0.4963, 5.3699)$
- With this, we are 95% confident that the interval going from -0.4963 to 5.3699 captures the intercept for the linear model describing heart weight as a function of body weight in the full population of cats.

Slope β_1

- The point estimate for the intercept in your linear model describing the relationship between X and Y is simply your $\hat{\beta}_0$.
- The multiplier used in your confidence interval will come from a t-distribution with $n-2$ degrees of freedom. Why $n - 2$? You started with n data points (and n degrees of freedom), and then you lose one degree of freedom for each parameter you estimate. You estimated two parameters: β_0 and β_1, hence $n - 2$.
- The standard error for $\hat{\beta}_1$ is $SE_{\hat{\beta}_1} = \frac{S_R}{\sqrt{\sum(x-\bar{x})^2}}$ with S_R the same as before.

Example with cat heart weight model from Figure 3.1 (a)

- Point estimate is simply $\hat{\beta}_1 = 2.8527$.
- With $n = 50$ there are $50 - 2 = 48$ degrees of freedom, so for a 95% confidence interval the multiplier will be $t_{.025,48} = 2.010635$.
- Using the components $\sum(x - \bar{x})^2 = 3.76$, and $S_R = 1.2007$, we can calculate $SE_{\hat{\beta}_1} = \frac{1.2007}{\sqrt{3.76}} = 0.6192$
- Interval: $2.8527 \pm (2.010635 \times 0.6192) = (1.6077, 4.0977)$
- With this, we are 95% confident that the interval going from 1.6077 to 4.0977 captures the slope for the linear model describing heart weight as a function of body weight in the full population of cats.

3.1.2 Hypothesis tests

Hypothesis tests for regression coefficients most often test the null hypothesis that $\beta_0 = 0$ or that $\beta_1 = 0$, but there is nothing stopping you from testing any value of interest for β_0 or β_1.

Intercept β_0

To test $H_0 : \beta_0 = b_0$ the test statistic is $t = \frac{\hat{\beta_0} - b_0}{SE_{\beta_0}}$ with $SE_{\hat{\beta_0}}$ as defined earlier. The test statistic follows a t distribution with $n - 2$ degrees of freedom.

Example with cat heart weight model from Figure 3.1 (a):

- $H_0 : \beta_0 = 0$ corresponding to testing if the line passes through the origin.
- Test statistic $t = \frac{\hat{\beta_0} - 0}{SE_{\beta_0}} = \frac{2.4368 - 0}{1.4588} = 1.670$.
- Being a two-tailed test, the p-value is the area under the $t_{df=48}$ distribution curve to the right of 1.670 and to the left of -1.670. This comes out to $p = 0.1014$, indicating there is not evidence the intercept differs from 0.

Slope β_1

As with intercept, to test $H_1 : \beta_1 = b_1$ the test statistic is $t = \frac{\hat{\beta_1} - b_1}{SE_{\beta_1}}$ with $SE_{\hat{\beta_1}}$ as defined earlier and the test statistic follows a t distribution with $n - 2$ degrees of freedom.

Example with cat heart weight model from Figure 3.1 (a):

- $H_0 : \beta_1 = 0$ corresponding to testing if the line is horizontal meaning knowing a cat's body weight gives no meaningful insight into it's heart weight.
- Test statistic $t = \frac{\hat{\beta_1} - 0}{SE_{\beta_1}} = \frac{2.8527 - 0}{0.6192} = 4.607$.
- Being a two-tailed test, the p-value is the area under the $t_{df=48}$ distribution curve to the right of 4.607 and to the left of -4.607. This comes out to $p < 0.0001$, indicating there is strong evidence the slope differs from 0.

In R

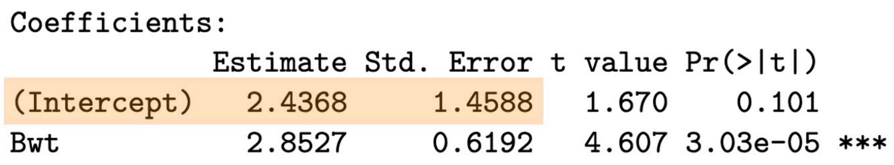
Confidence Intervals

There are a couple ways you can get your confidence intervals using R. Option one is to use the summary command to get the various elements you need for the intervals:

```r
mod_cat<-lm(Hwt~Bwt, data=cats[1:50,])

summary(mod_cat)
```

```
Call:
lm(formula = Hwt ~ Bwt, data = cats[1:50, ])

Residuals:
     Min       1Q   Median       3Q      Max
-2.98316 -0.79264 -0.00526  0.86316  2.28737

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.4368     1.4588   1.670    0.101
Bwt           2.8527     0.6192   4.607 3.03e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.201 on 48 degrees of freedom
Multiple R-squared:  0.3066,    Adjusted R-squared:  0.2922
F-statistic: 21.23 on 1 and 48 DF,  p-value: 3.029e-05
```

There is lots of great information in this summary output. It begins with the *Call* section that shows you the function call you made that created the model object in the first place. Next is the *Residuals* section that gives you the five number summary for the model residuals providing a snapshot of the model errors. The *Coefficients* section has the output you care about most. This table gives you the $\hat{\beta}_0$ and $\hat{\beta}_1$ estimates along with their standard errors. Highlighted below as an example is the detail for the intercept. The estimate for $\hat{\beta}_0$ of 2.4368 is in the Estimate column, and the standard error of $\hat{\beta}_0$, 1.4588 appears along side it in the Std. Error column.



```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.4368     1.4588   1.670    0.101
Bwt           2.8527     0.6192   4.607 3.03e-05 ***
```

Getting your t multiplier then is as simple as using the qt function:

```
qt(.025, df=48)
```

```
[1] -2.010635
```

Also helpful: the "Residual standard error" given in the summary output below the *Coefficients* table is the S_R used in the standard error formulas.

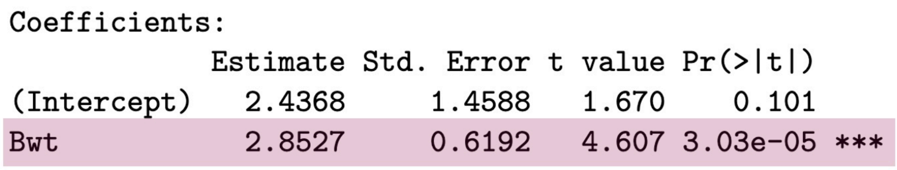
If you want to have R do even more of the work for you, there's the `confint` function. This provides you confidence intervals on both the intercept and slope as easily as:

```
confint(mod_cat, level=0.95)
```

```
                 2.5 %   97.5 %
(Intercept) -0.4963738 5.369927
Bwt          1.6076963 4.097623
```

Hypothesis Tests

The summary of your regression model Coefficients table not only gives the point estimate and standard error, but also the test statistic and p-value for the test of whether $\beta_i = 0$. Below is the cat model summary table highlighted to show the details for the test of $\beta_1 = 0$.



```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.4368     1.4588   1.670    0.101
Bwt           2.8527     0.6192   4.607 3.03e-05 ***
```

The third column gives the t test statistic, 4.607, and the fourth column provides the p-value for the test. To the right of the p-value column R provides a code indicating the outcome of test. If the p-value is below 0.001 as is the case with slope here, three astrisk marks are shown. A p-value between 0.01 and 0.001 gets two asterisks, between 0.01 and 0.05, a single asterisk, a single "." appears if the p-value is between 0.05 and 0.1, and the area is blank if the p-value is greater than 0.1. This is why no symbol is seen in the first row corresponding to the intercept.

On Your Own

1. Consider the `bac` data frame from the `openintro` library. Fit a regression model predicting blood alcohol content as a linear function of beer consumption.

   a. Interpret your model.
   b. Complete and interpret a 95% confidence interval for the intercept.
   c. Complete and interpret a 95% confidence interval for the slope. What does the intercept mean in the context of this data?
   d. Is there significant evidence at the $\alpha = 0.05$ level that the intercept is different from zero?
   e. Is there significant evidence at the $\alpha = 0.05$ level that the slope is greater than 0.015?

2. Consider the `mtcars` data frame from the `datasets` library. Fit a linear regression model predicting mpg as a function of vehicle weight in 1,000s of lbs.

   a. Interpret your model.
   b. Complete and interpret a 95% confidence interval for the intercept. What would the intercept mean in this context?
   c. Complete and interpret a 95% confidence interval for the slope.
   d. Is there significant evidence at the $\alpha = 0.05$ level that a car's fuel efficiency drops by an average of more than 5 mpg for every extra 1,000 lbs weight?

3. Consider the record race times for Scottish hill races provided in the `hills` data in the `MASS` library. Fit a linear model predicting the record race time in minutes as a function of the race distance length in miles.

   a. Interpret your model.
   b. What does the intercept mean in this context? Complete and interpret a 90% confidence interval for intercept.
   c. Complete and interpret a 90% confidence interval for the slope.
   d. Is there significant evidence at the $\alpha = 0.01$ level that an additional mile of race distances increases the record winning time by more than 7 minutes?

4. Consider our `cats` heart and body weight data from the `MASS` library.

   a. Fit a model predicting cat heart weight as a function of body weight using only the Male cats. Interpret your result.
   b. Fit a model predicting cat heart weight as a function of body weight using only the Female cats. Interpret your result.
   c. Complete 90% confidence intervals for the slopes of each model. Discuss your results. Is there evidence the relationship between heart weight and body weight varies by cat sex?

3.2 Confidence Intervals for $\bar{Y}|X$

In addition to confidence intervals for regression coefficients, you can also create confidence intervals for the mean Y at a given value of X. The point estimate that serves as the center of your confidence interval is simply the predicted value of Y given X based on your model fit and the multiplier is again a t distribution multiplier with $n-2$ degrees of freedom.

The standard error of $\bar{Y}$ when $X = x_*$ is

$$SE_{\bar{Y}|x_*} = S_R\sqrt{\frac{1}{n} + \frac{(x_* - \bar{x})^2}{\sum(x_i - \bar{x})^2}}$$

The S_R is the same one seen in section 3.1 and reflects the variability of residual error observed with the model fit. The $(x_* - \bar{x})^2$ component is entirely new though. If you look back at Figure 2, you'll see the five lines are closest together near the center of our collection of X values. On the outer edges, the lines are farther apart. This means we are much more confident in our prediction of Y given X in the middle of the X range than we are on the edges of the observed X range. That is what the $(x_* - \bar{x})^2$ component is capturing because x_* is the value of X for which you are predicting Y. When x_* is close to $\bar{x}$, this term will quite small and only the $\frac{1}{n}$ in the square-root will really matter. When x_* is far from $\bar{x}$, this term will be large and the full $SE_{(\bar{Y}|X)}$ will increase.

So using our cat body and heart weight data model on the first 50 cats, if we are interested in the mean heart weight of cats with a body weight of 2.3 lbs, the steps to complete a 98% confidence interval would look like:

$$\hat{Y}|(X = 2.3) \pm t_{\frac{0.02}{2},48} \times SE_{(\bar{Y}|X=2.3)}$$

$$\hat{\beta_0} + (\hat{\beta_1} \times 2.3) \pm 2.4066 \times 1.2007\sqrt{\frac{1}{50} + \frac{(2.3 - 2.34)^2}{3.76}}$$

$$8.9979 \pm 2.4066 \times 0.1716$$

$$(8.5849, 9.4109)$$

The interval for the mean heart weight of cats with a weight of 2.9 lbs however would be:

$$\hat{\beta_0} + (\hat{\beta_1} \times 2.9) \pm 2.4066 \times 1.2007\sqrt{\frac{1}{50} + \frac{(2.9 - 2.34)^2}{3.76}}$$

$$10.7095 \pm 2.4066 \times 0.3861$$

$$(9.7803, 11.6387)$$

Because 2.9 lbs is farther from the mean X of 2.34, the $SE_{\bar{Y}|X=2.9} > SE_{\bar{Y}|X=2.3}$ resulting in a wider interval.

In R

Confidence intervals for $\bar{Y}|X = x_*$ are available through the same predict.lm function seen in Chapter 2 for calculating predicted values with your model. The usage begins the same as with finding predicted values: you input your fitted model object and then a data frame with the x_* values of interest. All that's needed beyond that is specifying `interval="confidence"`. By default the confidence level is .95, but any value can be specified with a `level=` input as shown in the example below.

```
mod_cat<-lm(Hwt~Bwt, data=cats[1:50,])

new_cats<-data.frame(Bwt=c(2.3, 2.9))
predict.lm(mod_cat, newdata=new_cats, interval="confidence", level=.98)
```

```
        fit      lwr      upr
1  8.997894 8.584937  9.41085
2 10.709489 9.780337 11.63864
```

These result match what was found in the step-by-step calculations above.

On Your Own

1. Consider the `bac` data frame from the `openintro` library.

    a. Fit and interpret a regression model predicting blood alcohol content as a linear function of beer consumption.
    b. What do you estimate for the mean blood alcohol level when 1, 3, 5, or 7 beers are consumed?
    c. What is the t-multiplier you would use for a 98% confidence interval for the mean blood alcohol level after 1 beer?

d. Would a 95% confidence interval for the mean blood alcohol level after 1 beer be wider, narrower, or the same width as a 98% confidence interval for blood alcohol after 4 beers? Explain.

e. Complete 98% confidence intervals for the mean blood alcohol levels after 1, 3, 5, and 7 cans of beer. Write out an interpretation for the 3-beer interval.

2. Consider the `mtcars` data frame from the `datasets` library.

   a. Fit and interpret a linear regression model predicting mpg as a function of vehicle weight in 1,000s of lbs.
   b. Complete and interpret a 95% confidence interval for the mean mpg of cars weight 2,000 lbs, 2,500 lbs, 3,000 lbs, and 3,500 lbs.
   c. How wide are each of your intervals from part b? Why is the 2,000 lb interval so much wider than the 3,000 lb interval? Explain.
   d. Complete and interpret a 98% confidence interval for the mean mpg of cars weight 3,000 lbs. How does this interval compare to the 3,000 lb interval from part b.

3. Consider the speed and stopping distance data in the `cars` data frame.

   a. Fit and interpret a linear regression model predicting stopping distance as a function of speed.
   b. Complete and interpret 90% confidence intervals for the mean stopping distance of cars traveling at 20 and 40 miles per hour.
   c. How wide are each of your intervals from part b? Are you equally confident in each of them? Why or why not?

3.3 Prediction Intervals for Y

There is a new type of interval in regression modeling that is a little different than those you've learned up until now. Your intro stats class likely considered confidence intervals for a mean, a proportion, the difference between two means, or the difference between two proportions. In section 3.1 we looked at confidence intervals for intercept and slope for our regression fit. Section 3.2 was confidence intervals for the mean Y at a given $X = x_*$. What all of these have in common is that the interval is all about the estimate of an unknown population parameter.

There exists a true population mean, and you're using your sample data to create an interval likely to capture that population parameter because measuring the entire population to calculate it directly is not feasible. Just as there exists a true slope for the relationship between X and Y for the entire population - you can't measure the full population so you use your sample to develop an interval likely to include the population slope parameter.

This new type of interval, a ***prediction interval***, isn't about estimating an unknown population parameter. The goal with a prediction interval is to predict the next Y observation at a set $X = x_*$. As such, prediction intervals are quite a bit wider than the corresponding confidence interval at $X = x*$ because single observations have quite a bit more variability than means. How much more variability? Well... look back at Figure 1. What measure do we have that describes the variability of individual points around the fitted regression line? The S_R term. That's why while the standard error of $\bar{Y}$ at a given X looks like this:

$$SE_{\bar{Y}|x_*} = S_R\sqrt{\frac{1}{n} + \frac{(x_* - \bar{x})^2}{\sum(x_i - \bar{x})^2}}$$

The standard error of the next Y at a given X takes on this form:

$$SE_{y|x_*} = S_R\sqrt{1 + \frac{1}{n} + \frac{(x_* - \bar{x})^2}{\sum(x_i - \bar{x})^2}}$$

The two are nearly identical, but with the addition of a simple 1+ we've added in the S_R one more time to account for the variability of individual points.

The general form of your prediction interval is exactly as with the confidence interval:

$$pointestimate \pm multiplier \times standarderror$$

so if we want a 90% prediction interval for the heart weight of a cat with a body weight of 2.3, that is achieved through

$$E(Y|(X = 2.3)) \pm t_{\frac{0.1}{2},48} \times SE_{(y|X=2.3)}$$

$$\hat{\beta_0} + (\hat{\beta_1} \times 2.3) \pm 1.6772 \times 1.2007\sqrt{1 + \frac{1}{50} + \frac{(2.3 - 2.34)^2}{3.76}}$$

$$8.9979 \pm 1.6772 \times 1.2129$$

$$(6.9636, 11.0322)$$

Note that little 1+ addition has a big impact. Even though this is a 90% level interval, it is much much wider than the 98% confidence interval calculated in section 3.2 for the same $X = 2.3$. That's because while that one was about $\bar{Y}$ at $X = 2.3$, this is about a single observation. So we can now say we are 90% confident the interval ranging from 6.9636 to 11.0322 grams includes the heart weight of Fluffy, a cat with a body weight of 2.3lbs.

In R

For prediction intervals you're using the `predict.lm` function again only now specifying `interval="prediction"`.

```
mod_cat<-lm(Hwt~Bwt, data=cats[1:50,])

new_cats<-data.frame(Bwt=c(2.3, 2.9))
predict.lm(mod_cat, newdata=new_cats, interval="prediction", level=.9)
```

```
        fit      lwr      upr
1  8.997894 6.963668 11.03212
2 10.709489 8.594171 12.82481
```

On Your Own

1. Consider the `bac` data frame from the `openintro` library.

    a. Fit and interpret a regression model blood alcohol content as a linear function of the number of cans of beer consumed.
    b. What do you estimate for the blood alcohol content when 2, 4, 6, or 8 beers are consumed?
    c. What is the t-multiplier you would use for a 90% prediction interval for blood alcohol when 4 beers are consumed?
    d. Complete 90% confidence intervals for blood alcohol level with 2, 4, 6, or 8 beers are consumed. Write out an interpretation for the 4-beer interval.
    e. Would a 98% confidence interval for blood alcohol content when 2 beers are consumed be wider, narrower, or the same width as a 98% confidence interval for when 4 beers are consumed? Explain.

2. Consider the `mtcars` data frame from the `datasets` library.

    a. Fit and interpret a linear regression model predicting mpg as a function of vehicle weight in 1,000s of lbs.
    b. Complete and interpret a 95% prediction interval for the mpg of three new cars coming out soon with weights of 2,200 lbs, 2,700 lbs, and 3,300 lbs.
    c. How wide are each of your intervals from part b? Why is the 2,200 lb interval so much wider than the 3,300 lb interval? Explain.
    d. Complete and interpret a 95% confidence interval for the mean mpg of cars weighing 3,300 lbs. How does this interval compare to the 3,300 lb interval from part b? Explain the difference.

3. Consider the speed and stopping distance data in the `cars` data frame.

   a. Fit and interpret a linear regression model predicting stopping distance as a function of speed.
   b. Complete and interpret 90% prediction intervals for the stopping distance of cars traveling at 15, 25, 35, and 45 miles per hour.
   c. How wide are each of your intervals from part b? How would the width of an interval for a car traveling at 10mph compare to the widths seen in b?
   d. Are you equally confident in each of your intervals from part b? Why or why not?

4 Assumptions Were Made

4.1 Linearity

Perhaps the most fundamental assumption you are making when you fit a linear regression model is that a line is a good model. This is why before even fitting your model you should start with a scatterplot. If your X and Y don't look linearly related in a scatterplot it should be obvious that fitting a line is a bad idea.

But sometimes departure from linearity is subtle and your initial scatterplot can be misleading. Consider the case of the `forbes` data in the `MASS` library plotted below in Figure 4.1. This dataset contains 17 observations exploring the relationship between barometric pressure and the boiling point of water.

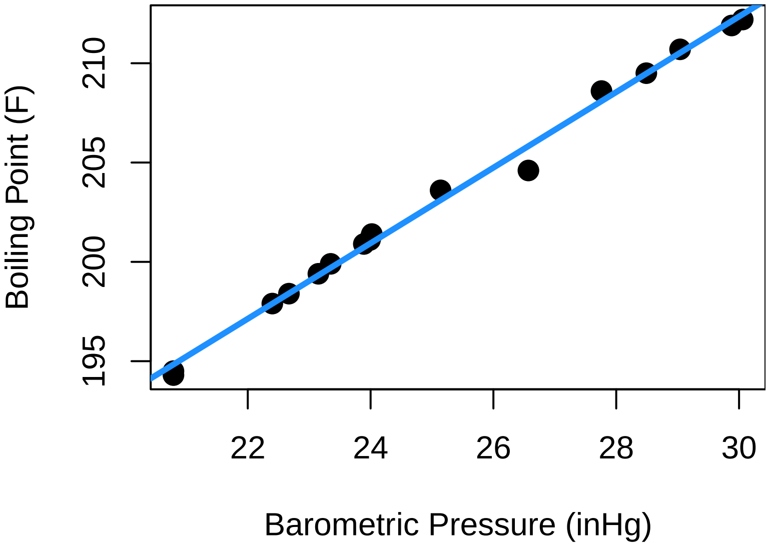
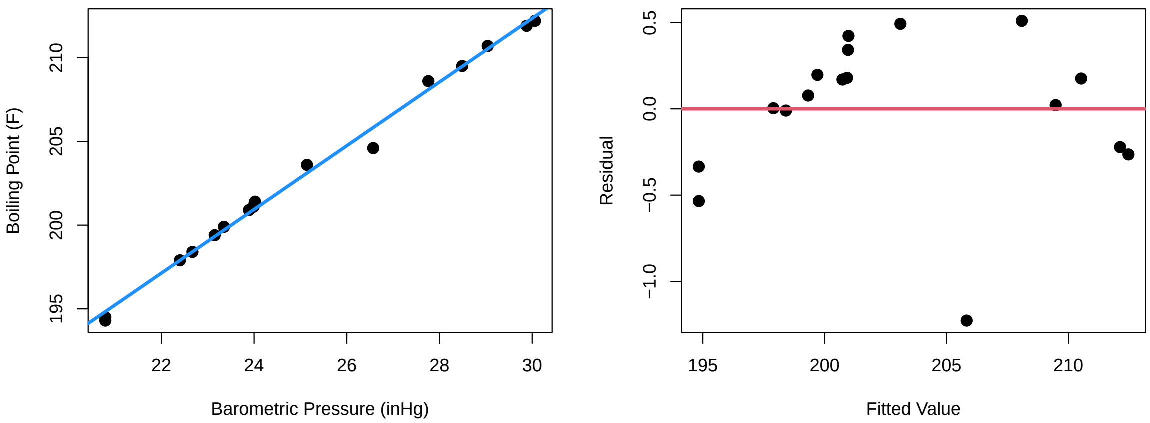


Figure 4.1: Barometric pressure and boiling point, fitted

The fitted regression line looks pretty good at first glance, but let's consider a new plot: the fitted vs. residual diagnostic scatter plot. In Figure 4.2 you can see the original data with the same fitted line as before along with a scatterplot of the model's fitted values plotted against residuals. A red reference line corresponding to a residual equal to 0 is included.



(a) Fitted Model

(b) Fitted vs. Residuals

Figure 4.2: Boiling model with residuals

Hmm... the fitted vs. residual plot is a giant arc (with one outlier). From the fitted vs. residual plot it becomes much more clear that as we scan from left to right along the blue fit line our data points begin below the fit line, then there's a string of points above it (with the exception of an oddity at pressure=26.6), and then the last points are below the line again. This means that there is subtle curvature to the relationship between pressure and boiling point.

Ideally your fitted vs. residual plot should be random scatter centered around zero. Without linearity, your line tends to over-estimate Y in some parts of X and under-estimate Y in others. If your linear model error isn't consistent throughout the range of X, the inference methods of Chapter 3 fall apart and aren't valid.

In R

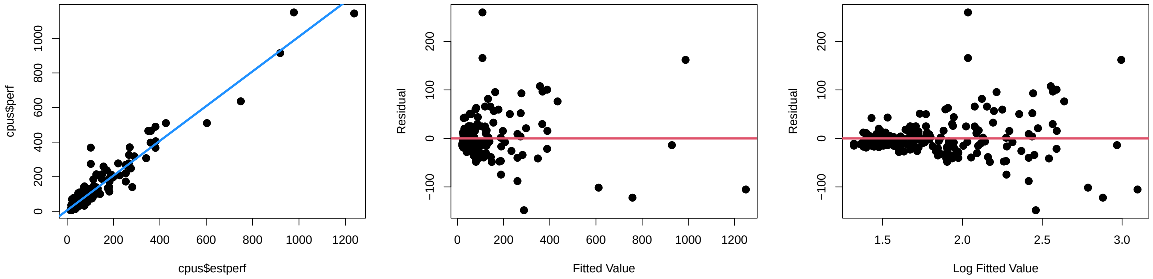
Creating the fitted vs. residual diagnostic plot is straight-forward in R and makes use of elements stored in the `lm` output object:

```r
# fit the model
mod_pressure<-lm(bp~pres, data=forbes)

# plot fitted values and residuals
plot(mod_pressure$fitted, mod_pressure$residual,
     xlab="Fitted Value", ylab="Residual", pch=16)

# add horizontal reference line
abline(h=0, col=2, lwd=3)
```

4.2 Homoskedasticity

Not only do you want your model to consistently have an average residual of zero, you want the spread around zero to be the same across all values of X. The term for this idea of same spread is ***homoskedasticity***.

Consider the cpus data frame in the MASS library of R. This data includes a variety of performance measures and characteristics of CPUs (computer Central Processing Units). The left side of Figure 4.3 shows a plot of the estimated CPU performance based on a prediction model by Ein-Dor and Feldmesser and the published performance based on an industry benchmark. Part b of Figure 4.3 shows the fitted vs. residual plot for the fitted model predicting published performance based on the Ein-Dor/Feldmesser estimate.



(a) Data with fit

(b) Fitted vs. Residuals

(c) Log(Fitted) vs. Residuals

Figure 4.3: CPU Model and Residuals

This fitted vs. residual plot doesn't show curvature like we saw with the boiling point data, but it doesn't exactly look like a random cloud either. Figure 4.3 c stretches out the x-axis using a log scale so you can better picture what is going on with the residuals as we move from low performing to high performing CPUs. The CPUs on the lower end of our scale with log fitted performance below 2.0 have residuals with a noticeably tighter clump around zero than the more powerful CPUs with log fitted performance above 2.0.

Recall from the inference equations in Chapter 3, a single S_R is used to represent the variability of residuals throughout the model. If S_R is not constant, those inference equations are no longer valid.

Beyond evaluating constant variance with a fitted vs. residual scatterplot, there is also a formal hypothesis test you can perform to check if the homoskedasticity assumption is reasonable. The F-test considers the null hypothesis that the ratio of two variances is equal to 1, vs the alternative that the ratio is different from 1. The test statistic is simply:

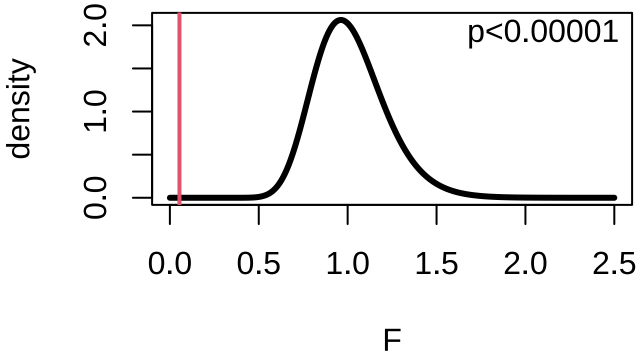
$$F = \frac{\sigma_1^2}{\sigma_2^2}$$

where σ_1^2 is the variance from group 1, and σ_2^2 is the variance of the second group. The degrees of freedom for the test are $n_1 - 1$ and $n_2 - 1$, where n_1 and n_2 are the two group sample sizes.

Performing this test on the CPU performance data and splitting the data so group 1 includes fitted values <50 and group 2 has fitted values <=50 yields:

$$F = \frac{166.859}{3098.414} = 0.05385$$

with $100-1 = 99$ and $109-1 = 108$ degrees of freedom. As shown below this means the p-value for the test is very very small meaning there is strong evidence the ratio of variances is not 1 and homoskedasticity is not present. The opposite of homoskedasticity is heteroskedasticity.



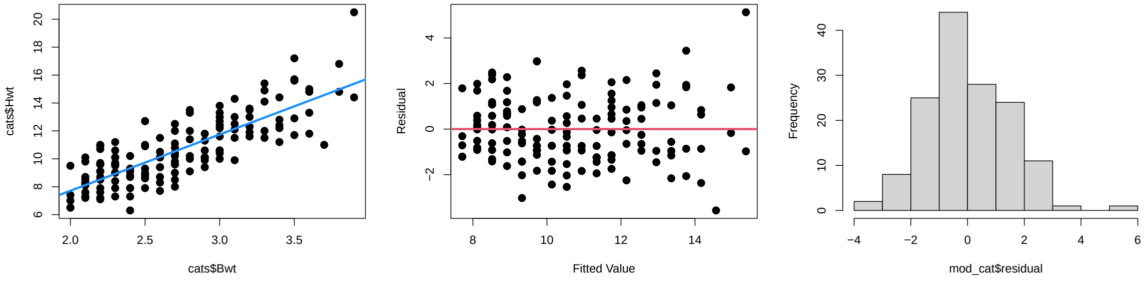
In R

The new element here is the variance test. R makes this easy to do with the `var.test` function.

```
mod_cpu<-lm(perf~estperf, data=cpus)

group<-mod_cpu$fitted<50

var.test(mod_cpu$residuals[group==TRUE], mod_cpu$residuals[group==FALSE])
```

4.3 Normality

Thinking back to the inference procedures of Chapter 3, recall that the t-distribution plays a role. That is because it is assumed that the residuals are approximately Normally distributed. Looking again at the cat heart weight and body weight data and model, Figure 4.4 shows the fitted model, the fitted vs. residual plot, and a histogram of the model residuals.

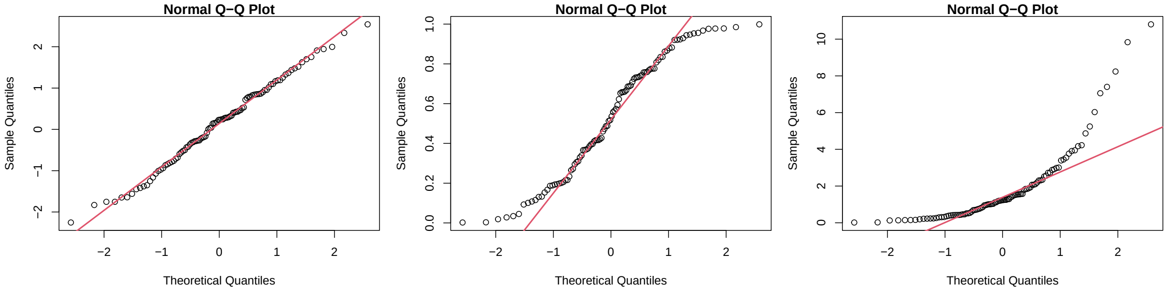


(a) Data with fit (b) Fitted vs. Residuals (c) Residual Hisogram

Figure 4.4: Cat model and residuals

A line looks like a reasonable approach looking at the random scatter of the fitted vs. residual plot, and there isn't cause to suspect heteroskedasticiy either (an F-test comparing the variance of the first 50% of fitted values to the second 50% of values has a p-value of 0.1012). The histogram of residuals looks fairly Normal, but there's more to do to check.

First is a residual Quantile-Quantile plot, usually simply referred to as a QQ plot. This plot is a scatterplot looking at the quantiles of your residuals as observed vs the quantiles you would have if your data was perfectly normally distributed. Figure 4.5 shows three example QQ plots for three different distributions shapes.



(a) Normal (b) Uniform (c) Skewed

Figure 4.5: Three QQ-Plot examples

In a), you see the ideal case where the data and the theoretical Normal align and follow the red reference line. Figure 4.5 b) shows what your QQ plot will look like if your residuals are distributed in a way that has more weight on both tails than is expected with a Normal. Along the middle of 6B the reference line is fairly closely followed, but left side is above the line and the right side is below the line. On the far right, Figure 4.5 c shows what a QQ plot looks like with right-skewed data. The right-skew makes for points in the QQ plot that stretch high above the reference line on the right side of the plot. If left-skewed, the plot would show a trail of points far below the line on the left side.

In the case of the cat data, the QQ plot of the residuals is as shown in Figure 7. Though not perfect, the points follow the reference line fairly well.

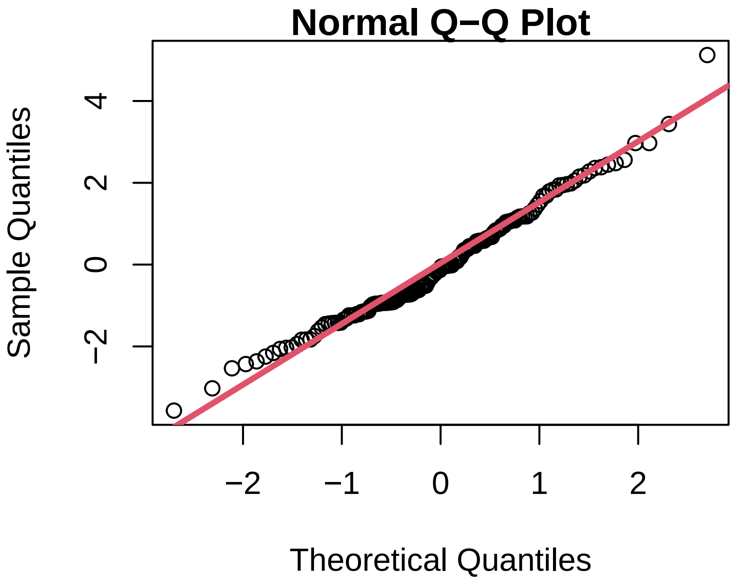


Figure 4.6: Cat Model Residuals

A formal test of Normality can be performed using the Shapiro-Wilk test. Details of how the test works are beyond our scope here, but the execution of the test is easy with software such as R. The null hypothesis in the Shapiro-Wilk test is that the sample data comes from a Normal distribution. Therefore, if the Shapiro-Wilk p-value is small, that indicates the Null is likely false, and therefore your assumption about Normal residuals is likely false. In our cat model, the Shapiro-Wilk test on the residuals yields a p-value of 0.1046, supporting our conclusion that assuming Normality is reasonable based on the histogram and QQ plots.

In R

QQ plots require two lines of code to create the visualization. The first line makes the scatterplot and the second overlays the reference line. The `col` and `lwd` options in the `qqline` function are optional, but can be helpful to make your reference line stand out a bit.

```r
qqnorm(mod_cat$residuals)
qqline(mod_cat$residuals, col=2, lwd=3)
```

The Shapiro-Wilk test for Normality is even easier to execute in R than the F test for homoskedasticity.

```r
shapiro.test(mod_cat$residuals)
```

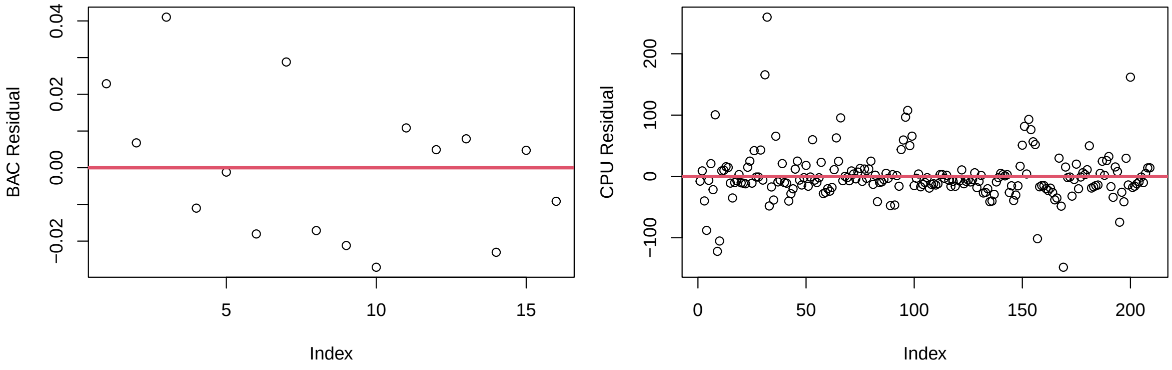
4.4 Independence

The final big assumption that was made in fitter the regression model is related to independence. Does each point in your scatterplot represent an observation that is independent from all the others?

Often this is an assumption that requires little more than pausing to think about how the data was collected. Does each point come from a different cat/cpu/person/car… whatever your study subject is? That's a good indication you are working with independent data. One plot that is often worthwhile when considering independence is simply a look at your residuals in order from first collected to last collected.

Like a good fitted vs. residual plot, a plot of residuals ordered by collection time should be a random scatter centered at zero. If, however, your data collector got better at their job over time, or equipment changed as it was used, or experimental conditions changed in a way you weren't aware of, or… some other weirdness… you might see a pattern or drift in the plot indicating a lack of independence.

Figure 4.7 shows the residuals from a)the CPU performance model b) the model predicting blood alcohol level based on beer consumption from Chapter 2.



(a) BAC Residuals

(b) CPU Residuals

Figure 4.7: Residuals in order

The plot of BAC residuals might look like it has a hint of a downward slope, but much of that is caused by students 1 and 3 having two of three largest residuals. If point 3 was removed, your eye wouldn't question this as a random jumble and we're looking for patterns stronger than a single point's influence.

The CPU residuals look random at first glance, but on a closer inspection you'll notice that there seem to be some clumping behavior. For example right around the index of 100, there are six points all together with unusually large residuals. Later on around 155, there's a clump of more than 10 points largely on top of each other forming a downward sloping tightly knit cluster.

Thankfully you need not rely on your eyes catching deviations from random scatter to properly assess independence; a formal test for correlated residuals exists. The Durbin-Watson test has the null hypothesis that the correlation of successive residuals (known as the autocorrelation of residuals) is zero against the alternative that it is non-zero. This means that small p-values from the Durbin-Watson test indicate the assumption of independence may not be valid.

The p-value for the Durbin-Watson test applied to the BAC model is 0.6152 and for the CPU model the Durbin-Watson p-value is 10^{-4}. This indicates that while our independence assumption for the student blood alcohol level data is likely sound, the same assumption for the CPU data is in doubt.

Why might that be? It seems unlikely that one published CPU performance compared to estimated performance would influence another one. The problem lies in our data's ordering. The CPU data frame is not listed in the order of measurement, but rather is listed by alphabetical order of the CPU name. What's happening here is that the performance estimate does a better job on CPUs from some manufacturers than others and so the errors turn up in clusters based on maker. Just because your Durbin-Watson test indicates significant autocorrelation doesn't mean trouble - it just means there is structure to your data order you maybe weren't aware of.

In R

The plot of residuals from first to last is a simple `plot` command. The Durbin-Watson test is available in the `lmtest` library in R and takes in a lm model object as the key input. Note that putting your residuals into the `dwtest` function won't work - you need to input the model object.

```r
# fit the model
mod_bac<-lm(bac~beers, data=bac)

# plot the residuals, add in a reference line if you like
plot(mod_bac$residuals)
abline(h=0, col=2, lwd=3)
```

```
# load the lmtest library
library(lmtest)

# run the Durbin-Watson test
dwtest(mod_bac)
```

All Assumptions, On Your Own

Nothing says your model can only break one assumption at a time. So you check them all…

1. Using the `bac` dataframe in the `openintro` library, fit a model predicting blood alcohol level as a linear function of beer consumption. Evaluate:

   a. Is the linearity assumption valid? Provide evidence and explain.
   b. Is the homoskedasticity assumption valid? Provide evidence and explain.
   c. Is the Normality assumption valid? Provide evidence and explain.

2. Create a model estimating an orange tree's age using it's circumference using the `Orange` data in the `datasets` library.

   a. Is the linearity assumption valid? Provide evidence and explain.
   b. Is the homoskedasticity assumption valid? Provide evidence and explain.
   c. Is the Normality assumption valid? Provide evidence and explain.
   d. Is the independence assumption valid? Provide evidence and explain.

3. Create a model estimating a Gentoo penguin's body mass as a linear function of flipper length using data from the `penguins` data frame from the `datasets` library. Be sure to limit your model to only include the Gentoo species.

   a. Is the linearity assumption valid? Provide evidence and explain.
   b. Is the homoskedasticity assumption valid? Provide evidence and explain.
   c. Is the Normality assumption valid? Provide evidence and explain.
   d. Is the independence assumption valid? Provide evidence and explain.

4. The `MASS` library contains the `anorexia` data frame that includes the pre and post treatment weights of anorexia patients. Fit a model predicting a patient's post-CBT treatment weight based on their pre-CBT treatment weight.

   a. Is the linearity assumption valid? Provide evidence and explain.
   b. Is the homoskedasticity assumption valid? Provide evidence and explain.
   c. Is the Normality assumption valid? Provide evidence and explain.
   d. Is the independence assumption valid? Provide evidence and explain.

5. Consider the hills data frame in the MASS library. The data frame contains the record winning times for 35 Scottish hill races along with their distances and elevation climbs. Fit two models predicting a race's record time 1) as a function of the race's distance and 2) as a function of the race's elevation climb.

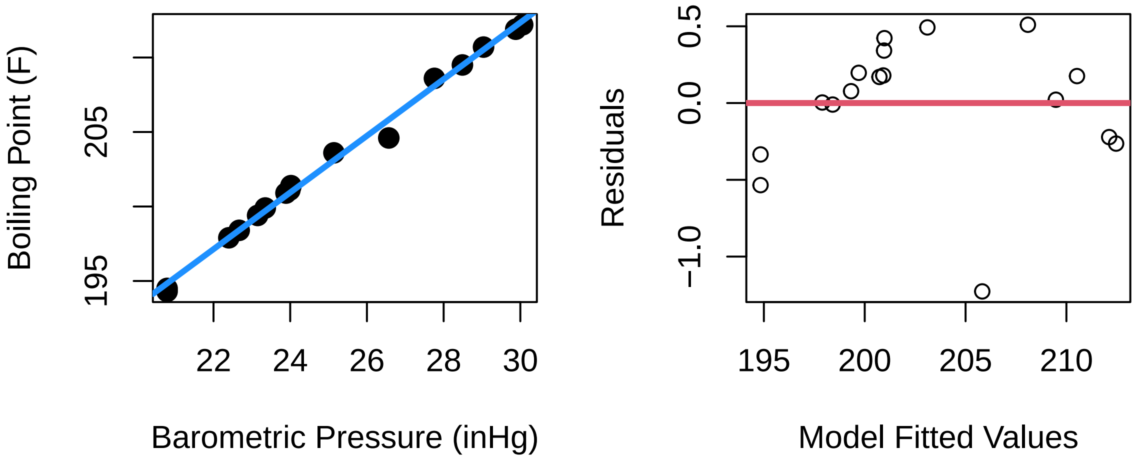
   a. Is the linearity assumption valid for either model? Provide evidence and explain.
   b. Is the homoskedasticity assumption valid for either model? Provide evidence and explain.
   c. Is the Normality assumption valid for either model? Provide evidence and explain.
   d. Is the independence assumption valid for either model? Provide evidence and explain.

5 Make it Work

Just because an assumption or two isn't met, that doesn't mean you can't use regression modeling. It just means you have to work a little harder at it.

5.1 Simple Transformations

We know from the work in **?@sec-chap4** that the relationship between barometric pressure and boiling point is not quite linear. Figure 5.1 shows the fitted regression line and the resulting fitted vs. residual plot with clear curvature. So what do we do about plots that aren't linear? We make them linear.



(a) Data and model fit

(b) Fitted vs residual plot

Figure 5.1: Boiling point model and residuals

The idea is this: rather than model boiling point as a function of pressure directly, could we model boiling point as a function of the square-root of pressure? Or log(pressure)? There are four main transformations to consider when trying to make non-linear data more linear: Squares, square-roots, inverses, and logs. Trial and error plus a little exploration might lead you beyond those four, but those are always good places to start.

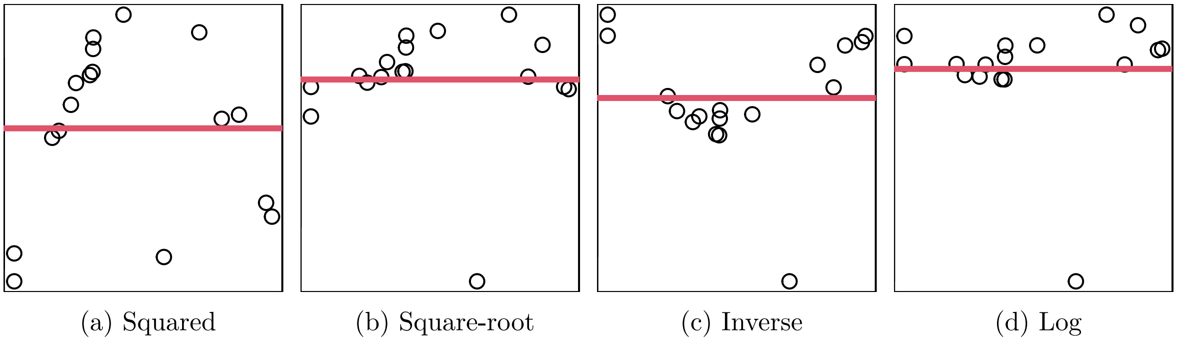

(a) Squared (b) Square-root (c) Inverse (d) Log

Figure 5.2: Four Tranformations

Figure 5.2 shows the resulting fitted vs. residual plot for each of these transformations applied to pressure. Axis labels are removed because we're focused only on the shape of the scattered points here and how closely they follow the red line.

Figure 5.2 (a) shows that results from modeling boiling point as a function of $pressure^2$ is an even more pronounced parabola than what was originally seen in Figure 5.1; and Figure 5.2 (c) shows that when boiling point is modeled as a function of $\frac{1}{pressure}$ the parabola flips to become concave up, but does not flatten considerably. From part (b) it is clear that modeling using $\sqrt{pressure}$ is a step in the right direction, but it is the model using $log(pressure)$ in (d) that is the clear winner of these four.

This means that rather than

$$boilingpoint = \beta_0 + \beta_1 pressure + \varepsilon$$

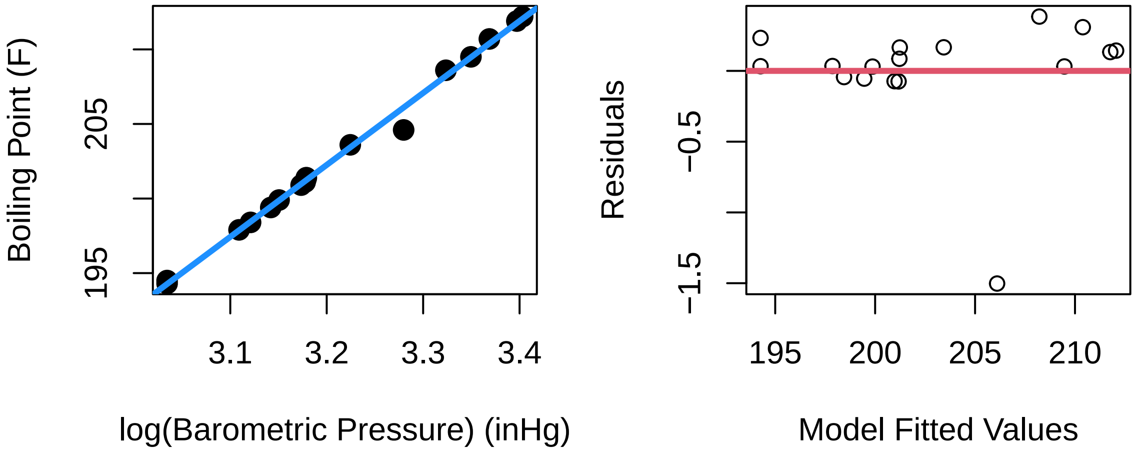
our model will take the form of

$$boilingpoint = \beta_0 + \beta_1 log(pressure) + \varepsilon$$

so a plot of our linear fit becomes what is shown in Figure 5.3. The estimated coefficients of the transformed model come out to be$\widehat{\beta}_0 = 47.8638$ and $\widehat{\beta}_1 = 48.2467$. So now to estimate the boiling point when $pressure = 28$ we'd have

$$47.8638 + 48.2467 \times log(28) = 208.6317$$

Finding a transformation that makes your linear model work can be a challenge. Maybe your data needs $\frac{1}{x^3}$ or some other less obvious approach - that's fine. Just start by looking at the shape of the curve you see in your data and consider what function will reverse that shape to create a line. At least one of squares, square-roots, inverses, and logs will usually show some



(a) Log pressure plot with model fit

(b) Fitted vs residual plot

Figure 5.3: Boiling point log pressure model

promise and then you can trial and error your way from there to find x^4 actually does a better job than x^2.

It is to your advantage though to use a simpler transformation when possible. Why? Interpretation and communication. It's pretty straight-forward to tell someone that boiling point is linearly related to the log of barometric pressure, but to try explaining that the square-root of boiling point is linearly related the inverse log of barometric pressure.

We've transformed X here, but you can also transform Y. Transforming X is generally preferable, because 1)every confidence interval or prediction interval would need to be either transformed back to original units, or interpreted in the revised transformed scale and 2) later we'll be working with models that include more than one X and transforming Y will impact the relationships for all X at once rather than allowing you to tweak each separately.

In R

When transforming data for linear modeling in R you have two options. First, you can create new variables for use in the `lm` command:

```r
log_pres<-log(forbes$pres)
model_forbes<-lm(forbes$bp~log_pres)
```

This takes an extra line of code but sometimes it's the cleanest approach if you're using the transformed x for uses beyond your linear model.

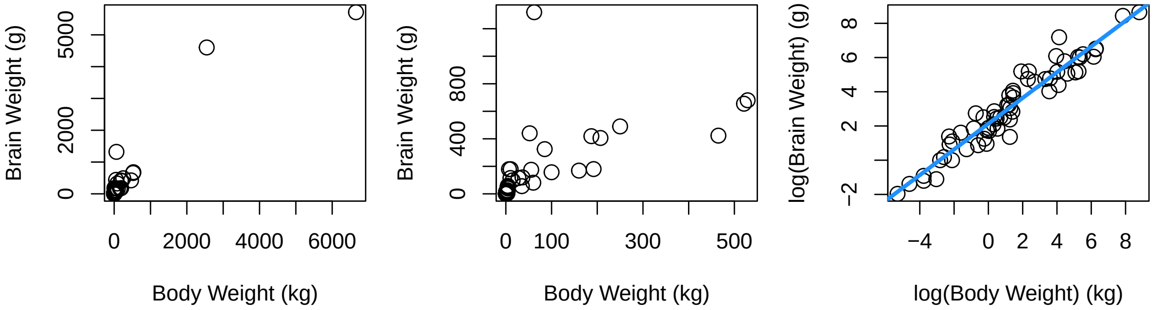
Second, you can use the I function inside the lm:

```
model_forbes<-lm(bp~I(pres^2), data=forbes)
```

This is a bit simpler as it is all in one line and when coupled with a data statment makes it 100% clear that both your X and Y live within the named data frame.

5.2 Log-Log Transformation

One transformation that is quite common for violations in linearity and Normality involves taking the log of both your X and your Y term. Consider the plots below in Figure 5.4 showing the body and brain weights of 62 mammals. The raw data in fig (a) shows a hard to distinguish clump of points in the lower left corner because African elephant and Asian elephant dwarf the other mammals, but the relationship does not appear to be linear. Part (b) of Figure 5.4 shows the raw data limited only to those mammals weighing less than 1,000kg. You'll see a hard to distinguish clump still remains since 39 of the 62 mammals are under 5kg, and the points beyond the clump provide further evidence a simple line isn't going to fit well. Part (c) shows the result when a log transformation is applied to both body and brain weight of all 62 mammals.



(a) All raw data (b) Only mammals <1000kg (c) Log-log with linear fit

Figure 5.4: Mammal body and brain weight

Below is the coefficent summary of the log-log model fit.

```
             Estimate Std. Error  t value     Pr(>|t|)
(Intercept) 2.1347887 0.09604339 22.22734 1.183207e-30
log_body    0.7516859 0.02846356 26.40871 9.835792e-35
```

The interpretation of coefficients in a log-log model is a bit special. Whereas in a non-transformed regression fit the slope is interpreted as the mean increase in Y associated with a one-unit increase in X, in the log-log case the slope is the mean percent increase in Y associated with a 1% increase in X. So in the case of mammal brain and body weight, we now see that a 1% increase in a mammal body weight is associated with a roughly 0.75% increase in brain weight.

As an example of how the transformation must be considered when making estimates, we turn to black-tailed deer. A 2023 study of dwarfed black-tailed deer on Blakely Island (Geiman and Long 2023) found an average body weight 40.2 kg. So using our model, what would be our best estimate of dwarf black-tailed deer brain weight? First we must take the log of 40.2, and after evaluating the linear fit, we must "un-log" the result to get back to the units of brain weight we care about.

```
#using predict.lm
new_wt<-data.frame(log_body=log(40.2))
pred_log_brain<-predict.lm(mod_mammal, new_wt)
exp(pred_log_brain)
```

```
       1 
135.8317 
```

With a 95% confidence interval of:

```
pred.ci<-predict.lm(mod_mammal, new_wt, interval="conf", level=.95)
exp(pred.ci)
```

```
       fit     lwr      upr
1 135.8317 108.833 169.5281
```

The Blakely Island researchers estimated a brain weight of 153.8g based on skull cavity measures of found skulls; not far from our estimate of 135.8g and included in our confidence interval for the mean brain weight that ranges from 108.8g to 169.5g.

5.3 Box-Cox Transformation

Real-world data often violate more than just the linearity assumption. Response variables might be skewed, or the variance might increase as the value of the response variable increases (heteroscedasticity). The **Box-Cox transformation** is a family of mathematical functions used to transform non-Normal dependent variables into a form that more closely follows a Normal distribution. The Box-Cox approach can also help in making variances more consistent to meet our homoskedasticity assmption for regression inference.

Box-Cox transformations are especially useful when:

- Your response variable is strictly positive ($y > 0$).
- You observe non-constant variance (heteroscedasticity) in residual plots.
- The distribution of your response variable is highly skewed.
- Residuals from a fitted model are not normally distributed.

The down side to Box-Cox is model interpretability - you'll definitely have some reverse transforming to do once your model is fit so that people can understand it.

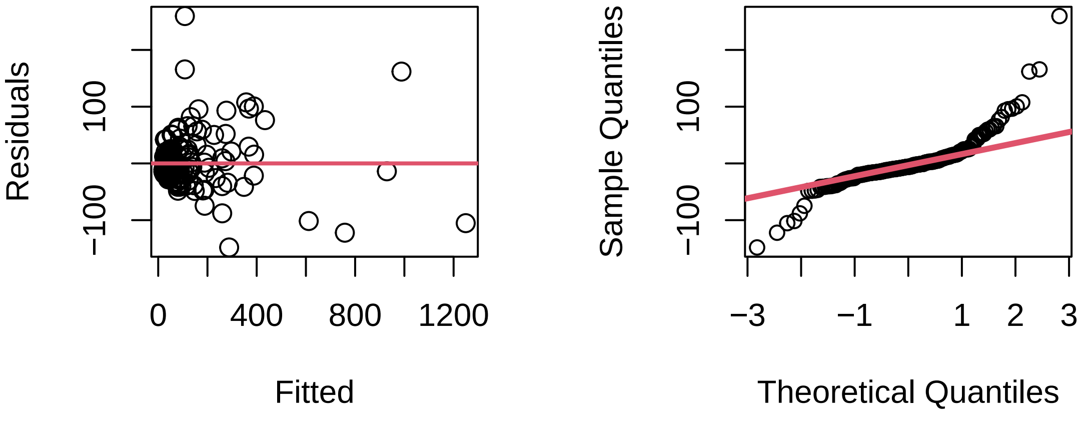
The Box-Cox transformation is defined as follows for a positive variable y:

$$y^{(\lambda)} = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln(y) & \text{if } \lambda = 0 \end{cases}$$

Here, λ (lambda) is a parameter that determines the form and strength of the transformation. The logarithm ($\ln y$) is a special case when $\lambda = 0$.

So how do you decide what value to use for λ? The algorithm to determine λ is an iterative one that includes repeated fitting a linear model on transformed data and selecting the λ with the maximum log-likelihood. Good news - R will do heavy repetitive lifting for you. Don't just take R's output and run with it though. You should always inspect diagnostic plots before and after applying Box-Cox to ensure it has improved model assumptions.

Figure 5.5 shows residuals from our CPU performance model first seen in Chapter 4. When CPU performance is modeled as a linear function of estimated performance the variance of residuals is seen to grow substantially from left to right in our fitted vs. residual plot and a QQ plot of the residuals reveals notable deviations from Normality.



(a) Fitted vs. Residuals

(b) Residual QQ Plot

Figure 5.5: CPU Performance Linear Model

The plot shown in Figure 5.6 is produced when R is called on to evaluate values of λ in a Box-Cox tranformation. Dotted lines are included to make reading off the ideal λ easier. Details from R tell us the maximum log-likelihood occurs when $\lambda = 0.6262$.

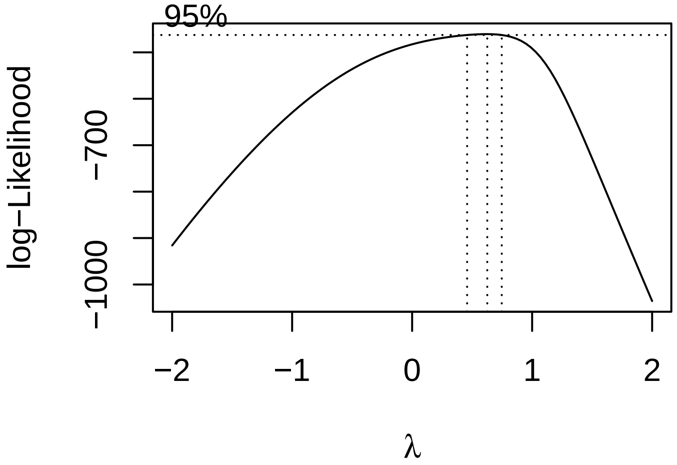
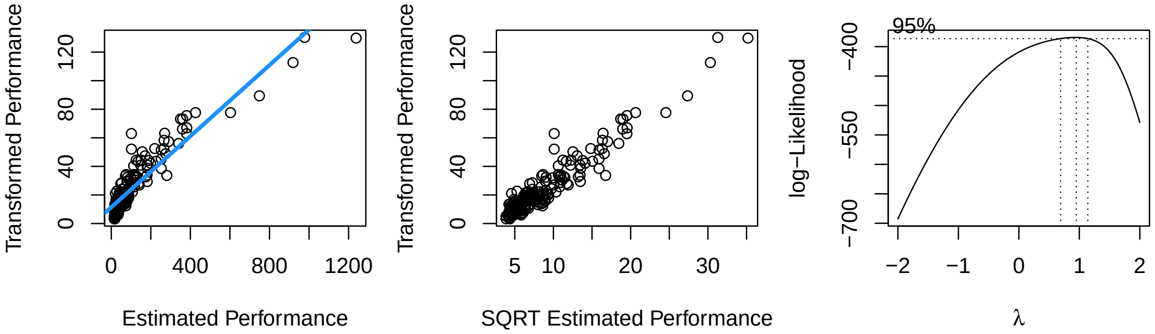


Figure 5.6: Box-Cox λ log-likelihoods

When the Box-Cox transformation with $\lambda = .6262$ is then applied to CPU performance something interesting happens though. Yes, the variance becomes much more stable, but we also see that there is significant curvature in the relationship between estimated CPU performance and our new transformed performance outcome. This necessitates a second step: transforming X, then re-running the Box Cox algorithm to find the optimal λ when the transformed X is used to predict Y.

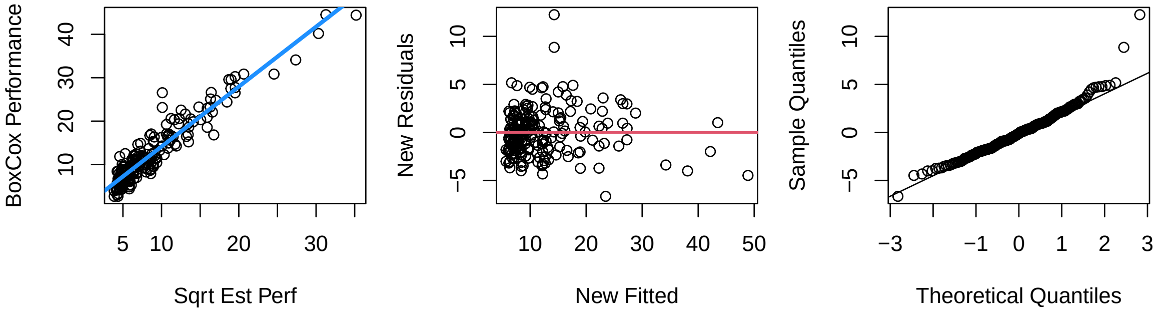


(a) Plot of transformed data (b) Add X Transformation (c) New Box-Cox λ plot

Figure 5.7: CPU Model with Box-Cox

The curvature after the first Box-Cox round is shown in Figure 5.7 (a), with the fitted line overlaid in blue to make clear just how much curvature is present. Figure 5.7 (b) shows that if the square-root of estimated CPU performance is used instead of the estimated CPU performance directly the relationship becomes much more linear. Part (c) then shows the updated λ estimate is slightly smaller at 0.4242.

The revised model model for CPU performance is now shown in Figure 5.8. The curvature is much improved, the variance appears fairly constant across the fitted vs. residual plot, and the QQ plot of residuals is much improved over the original QQ plot of Figure 5.5.

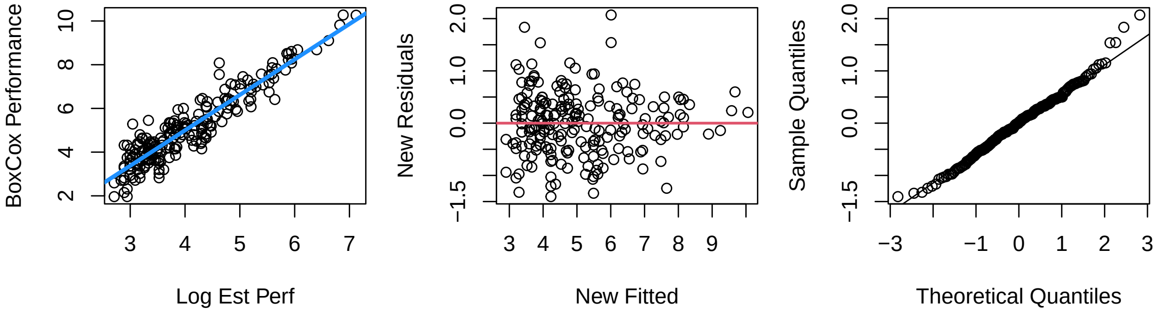


(a) Plot of transformed data (b) Fitted vs. Residuals (c) QQ Plot

Figure 5.8: Updated CPU Box-Cox, sqrt(x)

Work determining a model for CPU performance could certainly end here. Figure 5.8 looks quite reasonable. However, there is still slight curvature in our data shown in plot (a). Maybe we then try a log(estperf) approach to handle curvature. Then a new Box Cox λ search would be needed.

Figure 5.9 shows the result of using log(estperf) to predict a Box Cox with $\lambda = 0.10101$ applied to CPU performance. The curvature issue is improved over what was seen in Figure 5.8 (a), but the variance in Figure 5.9 (b) is not nearly as consistent as in Figure 5.8 (b) and creates homoskedasticity concerns. Normality is reasonably met with both models



(a) Plot of transformed data (b) Fitted vs. Residuals (c) QQ Plot

Figure 5.9: Updated CPU Box-Cox, log(x)

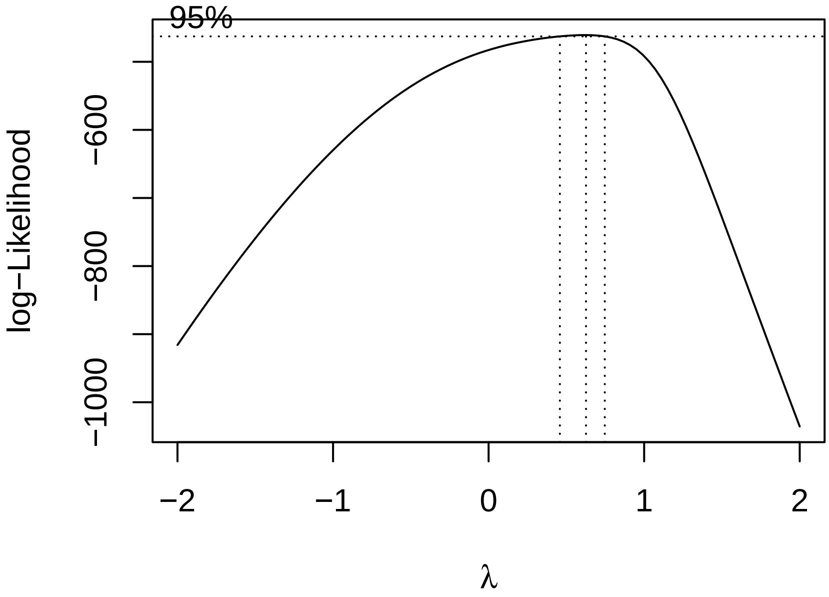
So which model is the right model? Box-Cox combined with square-root of estimated performance, or Box-Cox combined with log estimated performance? I prefer the square-root one, but really this is where you have to accept something not all students handle well. There is no one right model. As you gain experience with regression and other data science techniques you will learn this work has an element of art to it. Two different statisticians will quite reasonably come up with two slightly different models and that doesn't mean either one is wrong.

Statistician George Box (yup, the Box in Box-Cox) is credited with saying "All models are wrong, but some are useful." He's also credited with admonishing that "Statisticians, like artists, have a bad habit of falling in love with their models." Take these words to heart. If two competing models seem equally useful, pick one and move on with your life. But also, if you discover a flaw in your model, don't be so wedded to it that you don't revise it to fix the flaw.

In R

To work with Box-Cox transformations you'll want to load the `MASS` library. From there it is as simple as using the `boxcox` function. The only input the function needs is your basic fitted model object, and it will then output the graph below.

```
library(MASS)
simple_mod<-lm(perf~estperf, data=cpus)
bc<-boxcox(simple_mod)
```



By saving the boxcox result as bc (or whatever other name you'd like) you are able to pull the optimal λ parameter by:

```
bc$x[which.max(bc$y)]
```

```
[1] 0.6262626
```

And then apply the result:

```
bc_perf<-((cpus$perf^0.6262626)-1)/0.6262626
mod_bc<-lm(bc_perf~estperf, data=cpus)
```

5.4 Unit change

There is one last transformation to be aware of and it's the simplest of all: a change of units. Regardless of the units used to measure x and y the relationship between x and y should remain unchanged. This means if our car fuel efficiency data had km per liter instead of miles per gallon, heavier cars would still be less efficient than lighter cars. It also means if we measured boiling point in degrees Celsius instead of Fahrenheit, the relationship between barometric pressure and boiling point should be the same as we saw back in Figure 5.3 (a).

This doesn't mean the fitted parameters stay the same as units change though. Consider our beer consumption model from Section 2.3. The data is recorded as cans of beer, but could just as easily have been measured in fluid ounces (and that probably would have been better for clarity). Let's see how the model changes for the two measures of beer consumption:

```
mod1<-lm(bac~beers, data=bac)
summary(mod1)$coef
```

```
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) -0.01270060 0.012637502 -1.004993 3.319551e-01
beers        0.01796376 0.002401703  7.479592 2.969480e-06
```

```
beer_oz<-bac$beers*12
mod2<-lm(bac~beer_oz, data=bac)
summary(mod2)$coef
```

```
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) -0.01270060 0.012637502 -1.004993 3.319551e-01
beer_oz      0.00149698 0.000200142  7.479592 2.969480e-06
```

The intercepts of the two models are exactly the same, but the slopes are different. When beer is measure in cans, the slope is 0.01796 but when measured in fluid ounces, 12 times the can count, the slope is 0.0015. Notice that 12 times 0.0015 equals 0.01796. This makes sense because if $bac = -0.0127 + 0.01796 \times cans$ then the relationship can be equivalently expressed as $bac = -0.0127 + (0.01796/12) \times (cans \times 12)$ (the 12s cancel out to make it all simply equal to multiplying by 1).

Standard error for our slope estimate changes by the same multiplier so in the end the t-value and corresponding p-value for slope remain unchanged.

Remember this simple transformation should you find yourself with really tiny slopes you'd like to make bigger, or really large slopes you'd like to make smaller, just for ease of communication. A quick change of measuring in kg to measuring in g will shrink a slope by a factor of 1000 just as going from m to km will increase it by a factor of 1000. Likewise sometimes it makes sense to use measure of 1,000s of people or millions of dollars just because the coefficients are unwieldy when individual people or dollars are the units. Talking about an increase of 2.3 extra points on statewide tests per million dollars in education spending is just easier than an increase of 0.0000023 points per dollar spent.

On Your Own, All Transformations

1. The `Puromycin` dataset in the `datasets` package contains information regarding an experiment investigating instantaneous enzymatic reaction rates of cells as a function of substrate concentrations.

    a. Build a model for reaction rate as a function of concentration considering only those cells that are untreated. Communicate your model fit and provide evidence major assumptions on residuals are all met.
    b. Build a model for reaction rate as a function of concentration considering only those cells that are treated. Communicate your model fit and provide evidence major assumptions on residuals are all met.
    c. Create and explain a 90% prediction interval for the reaction rate of an untreated cell with a concentration of 0.4.
    d. Create and explain a 90% confidence interval for the mean reaction rate of treated cells with a concentration of 0.4.
    e. Is there sufficient evidence, at the $\alpha = 0.05$ level, that the reaction rate change associated with changes in concentration is significantly different for the treated and untreated cell groups? Explain.

2. The muscle dataset in the MASS library has information on an experiment on rat heart muscle. Using the data, fit a model predicting the change in muscle strip length using the calcium chloride solution concentration.

a. Explain what transformation you used and interpret your model. Show all assumptions are met.

b. Create a graph of your model, adding the fitted the curve to a plot of un-transformed concentration and length.

c. Is the expected length change with zero calcium chloride concentration greater than 0? Explain.

3. The `mammals` dataset in the `MASS` library contains the brain and body weights of 62 mammals.

   a. Apply the appropriate transformation to make a linear model work for predicting brain weight as a function of body weight. Explain your model and demonstrate all assumptions are met.

   b. An adult llama weighs approximately 180kg. Create and explain a 90% prediction interval for the weight of a llama brain. After you've created your interval, google it and see if your interval contains the answer you find.

   c. Create and interpret a 95% confidence interval for the slope of your model.

4. Using the `pressure` dataset in the `datasets` package, fit an appropriate model for vapor pressure as a function of temperature.

   a. Fitting the model requires a transformation. What transformation did you select and why?

   b. What is the equation of your fitted line? Explain how pressure changes for every one degree increase in temperature.

   c. Provide graphical evidence that all key assumptions are met in your model.

   d. Modify the data to use temperature in degrees Farhenheit. How does your model change? Does this match your expectation? Explain.

5. Using the `trees` dataset from the `datasets` package,

   a. Fit a model for tree volume as a function of tree height. Discuss your chosen model and demonstrate assumptions are all met.

   b. Fit a model for tree volume as a function of tree girth. Discuss your chosen model and demonstrate assumptions are all met.

   c. Which model does a better job of predicting tree volume? Explain your choice.

6. Use the `cabbages` data in the `MASS` library. Your goal is to fit a model for the weight of a head of cabbage as a function of the vitamin C content.

a. What assumption or assumptions are not met by a basic linear model? Explain how you know.

b. Apply a transformation and fit your model. Interpret the fit and give evidence assumptions are all met.

5.5 Weighted Least Squares

Sometimes your data points should not all be treated equally. Why not? Well consider the `babies_crawl` data frame in the `openintro` library. This data provides the average age at which babies began to crawl by birth month. Researchers speculated that babies who are bundled up for cold weather when they are six months old likely learn to crawl later than babies who are six months in warmer seasons and have less clothing inhibiting movement (Benson 1993).

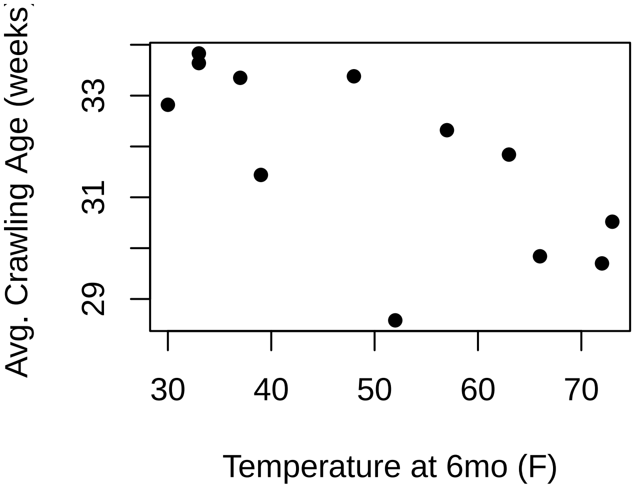


Figure 5.10: Crawling Age and Temperature

From Figure 5.10 it looks like the hypothesis has some merit. A best fit line would reach from the upper left to the lower right indicating babies with warmer temperatures are crawling earlier than babies learning in colder temperatures. However, this plot is hiding some key information about the data. We have just one point for every birth month, but each birth month includes anywhere from 21 to 49 babies. Some months the standard deviation of crawling age was less than 6 weeks, but others it was greater than 8 weeks. If we treat each point equally, those differences of samples sizes and in-month variability are lost.

Enter Weighted Least Squares (WLS) regression. By applying WLS, you give more weight to observations you trust more, and less weight to those you trust less. Generally in simple linear regression our goal is to fit the $Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$ model to minimize $\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$, the sum of squared residuals. With WLS we instead aim to minimize weighted residuals $\sum_{i=1}^{n} w_i(Y_i - \hat{Y}_i)^2$ where, w_i is the weight assigned to the i^{th} data point. Along with this, our SST and SSR change to incorporate weights as well:

$$SSError = SSE = \sum w_i(y_i - \hat{y}_i)^2$$

$$SSTotal = SST = \sum w_i(y_i - \bar{y})^2$$

$$SSRegression = SSR = \sum w_i(\bar{y}_i - \hat{y}_i)^2$$

It still holds that SST=SSE+SSR. Comparing R^2 values of models with different weights is not a fair comparison when evaluation options. Choosing suitable and appropriate weights is crucial. Common approaches include:

- **Inverse Variance Weighting (Most Common)**

  If each observation has a known variance σ_i^2 , assign $w_i = 1/\sigma_i^2$. This means observations with smaller variance (higher precision) get more weight.

- **Based on Measurement Error**

  If the measurement error for each Y_i is known, assign smaller weight to points with larger error.

- **Subjective or Utility Weights**

  Sometimes weights reflect how important or relevant each data point is for your particular context.

- **Iteratively Estimated Weights**

  When heteroskedasticity is present but true variances are unknown, estimate the relationship between variance and the predictor and then assign weights accordingly.

By understanding when and how to assign weights, you can improve the performance of your regression analysis and draw better conclusions from your data.

In the `babies_crawl` data, the number of babies for each month is given in the `n` column, and the standard deviation of crawling age observed within each month is given in `sd`. From the Central Limit Theorem, we know these two components can give us the standard error for the 12 average crawling ages: $\frac{sd}{sqrt(n)}$. Using the inverse of standard error as weights will give higher weights to low standard errors and low weights to months with high standard errors.

Figure 5.11 shows the model without weights in a red dashed line, and the model using weights in blue. Each point in the plot is sized relative to it's weight so we can see the unusually low crawling age at a temperature of 52 is weighted quite low in the weighted model.

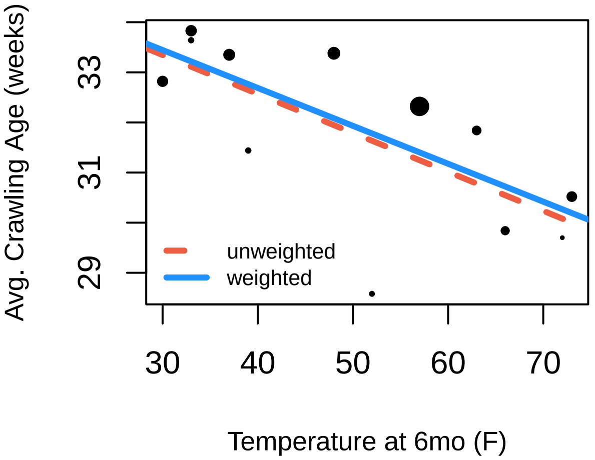
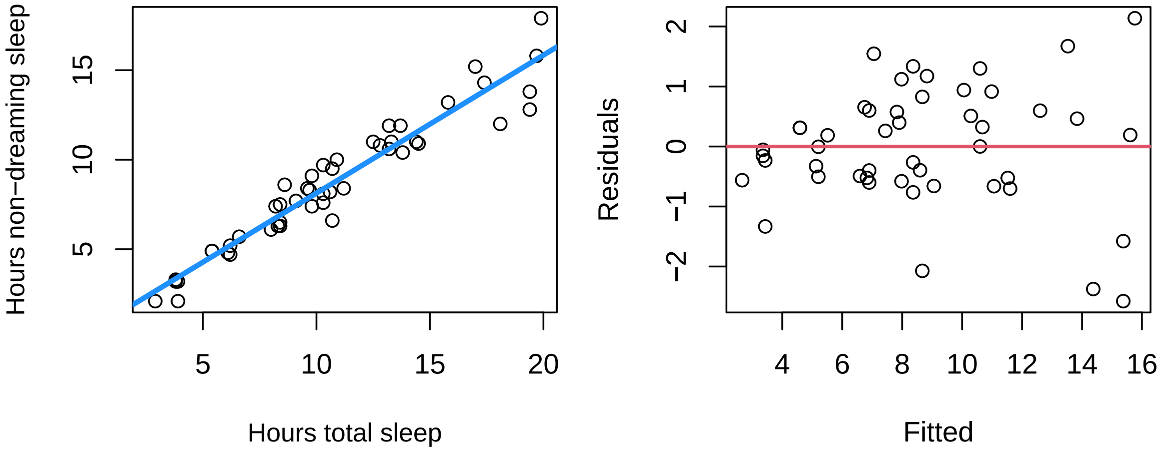


Figure 5.11: Crawling Age and Temperature, two models

Now for an example that doesn't have standard deviation provided directly. In the `openintro` library you'll find the `mammals` data frame that contains not only brain and body weight as we looked at earlier, but also information on sleep and dreaming.

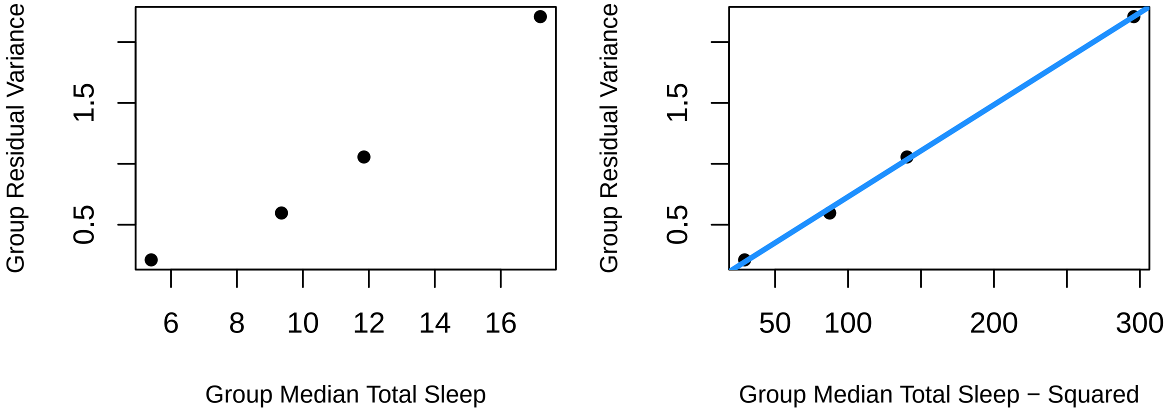


(a) Data with fitted line

(b) Fitted vs. Residuals

Figure 5.12: Mammal Sleeping and Non-Dreaming

If you split the data into four quartiles and calculate the variance for each group the result is as shown in Figure 5.13 (a). Though close, the relationship between the center of each grouping and the group's residual variance is not quite linear but a simple square transformation on the group center fixes that. This means the variance of the residuals can be modeled linearly by the square of X. Therefore, to use inverse variance as our weights then, we can use $\frac{1}{X^2}$.



(a) Four group variances

(b) Group centers squared vs group variances

Figure 5.13: Mammal Sleeping and Non-Dreaming

Without incorporating weights, the model is the dashed orchid line shown in Figure 5.14. With the inverse X^2 weights the model is the solid blue line. The models are quite close, but real differences emerge when inference methods are applied under the two models. That is because S_R now includes the weights:

$$S_{R,w} = \sqrt{\frac{\sum w_i(y_i - \hat{y}_i)^2}{n-2}}$$

and weights further play a role in the standard errors for the weighted model coefficients as:

$$SE_{\hat{\beta}_{0,w}} = S_{R,w}\sqrt{\frac{1}{n} + \frac{\bar{x}_w^2}{\sum w_i(x - \bar{x})^2}}$$

$$SE_{\hat{\beta}_{1,w}} = \frac{S_{R,w}}{\sqrt{\sum w_i(x - \bar{x}_w)^2}}$$

where $\bar{x}_w$ is the weighted mean:

$$\bar{x}_w = \frac{\sum(w_i \times x_i)}{\sum w_i}$$

This means in the unweighted model, $S_R = 1.0026$ but in the weighted version, $S_R = 0.0902$.

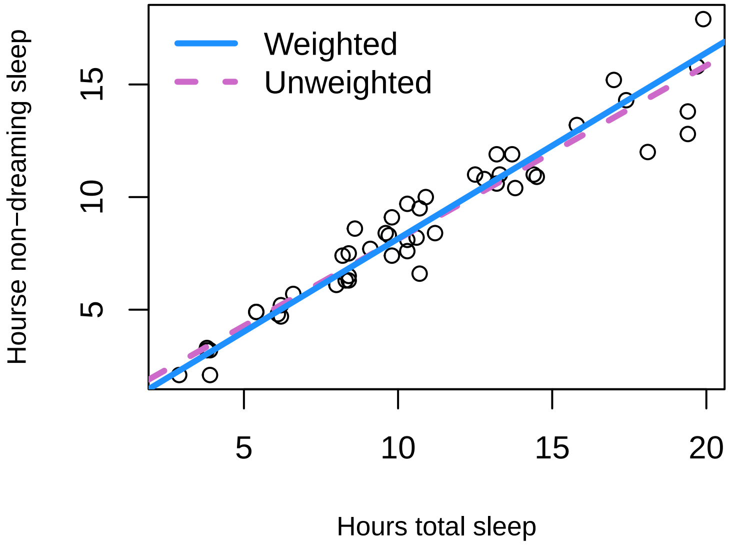


Figure 5.14: Mammal sleep model, Weighted & Unweighted

Further, for the unweighted model the 95% confidence interval for slope $\hat{\beta}_1$ ranges from 0.7064 up to 0.8349 but in the weighted model the interval for $\hat{\beta}_{1,w}$is from 0.7717 to 0.8796.

Similarly, inference for Y values corresponding to a particular $X = x_*$ changes with revised standard error equations:

$$SE_{\bar{Y}|x_*,w} = S_{R,w}\sqrt{\frac{1}{\sum w_i} + \frac{(x_* - \bar{x}_w)^2}{\sum (x_i - \bar{x}_w)^2}}$$

The standard error of the next Y at a given X takes on this form:

$$SE_{y|x_*,w} = S_{R,w}\sqrt{\frac{1}{w_*} + \frac{1}{\sum w_i} + \frac{(x_* - \bar{x}_w)^2}{\sum w_i(x_i - \bar{x}_w)^2}}$$

where w_* is the weight associated with x_*, the X location of the prediction.

This means that while the 95% prediction interval for the amount of non-dreaming sleep that is typical for a mammal that gets nine hours total sleep ranges from 5.32 to 9.4 with the unweighted model, our weighted model produces a narrower interval going from 5.68 to 8.98.

In R

The `lm` command accommodates weights with minimal change to the code you've already learned:

```r
mod_mam_sleep_wt<-lm(non_dreaming~total_sleep, data=mammals,
                     weights=1/mammals$total_sleep^2)

summary(mod_mam_sleep_wt)$coef
```

```
              Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -0.1039658 0.20031687 -0.5190066 6.062443e-01
total_sleep  0.8256810 0.02681064 30.7967632 2.383219e-32
```

The `confint` and `predict.lm` functions can still be used on weighted models just as you did with the simpler models without weights. With prediction intervals, you'll need to input the weight that should be used at the $X = x_*$ of interest.

```r
confint(mod_mam_sleep_wt)
```

```
                 2.5 %    97.5 %
(Intercept) -0.5071827 0.2992512
total_sleep  0.7717140 0.8796480
```

```r
predict.lm(mod_mam_sleep_wt, data.frame(total_sleep=c(9,12)),
           interval="prediction", weights=c(1/81, 1/144))
```

```
       fit      lwr       upr
1 7.327163 5.675607  8.978719
2 9.804206 7.596301 12.012112
```

On Your Own

1. Install and load the `fivethirtyeight` package. Using the `murder_2015_final` data file, estimate the 2015 murder rate in large US cities using the 2014 murder rates.

    a. Begin by fitting a basic model in the form of murders_2015~murders_2014 regression model. Check all inference assumptions. Which assumptions are met and which need attention?

b. Develop a model for the variance to be used in a weighted model. Explain your approach.

c. Fit and interpret your weighted least squares model.

2. Examine the Rabbit data frame in the MASS library.

   a. Find the variance for each dose level and use that in a weighted regression model predicting blood pressure change. Explain what your model shows.

   b. Complete and interpret a 95% confidence interval for the mean change in blood pressure when given a dose of 80 micrograms of Phenylbiguanide.

6 Zeros and Ones

6.1 Indicator variables

An indicator variable, also known as a dummy variable, is a variable used to represent membership in a specific category of a categorical variable. Using only the values 0 and 1, the indicator variable will indicate the presence of a particular attribute or membership in a category of interest with a 1, and use a 0 for everything else.

Indicator variables are essential when incorporating categorical variables—like treatment group, sex, or employment status—into linear regression models. For example, to represent whether each individual included in a survey is employed or unemployed, an indicator variable could be defined as 1 if the person is employed, and 0 if the person is unemployed.

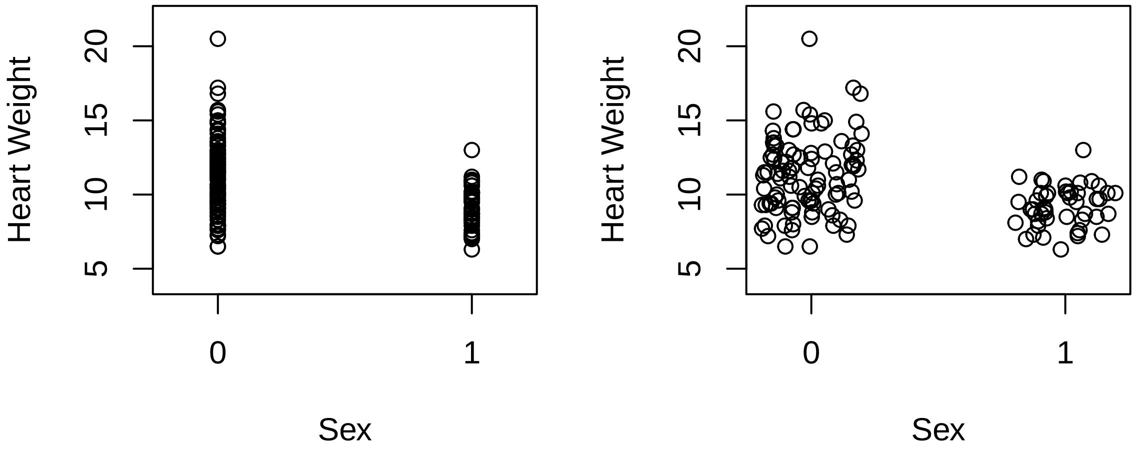
For categorical variables with more than two categories, a separate indicator variable is created for each category except one, which becomes the reference (or baseline) group. This encoding, often called one-hot encoding, ensures that the model can estimate effects relative to the reference group while avoiding perfect collinearity (remember we need $X^T X$ to have an inverse). For instance, if we instead treated employment as having three categories of full-time, part-time, or unemployed we would create two indicator variables: one for full-time and one for part-time. Unemployed would then serve as the reference group.

The choice of which group serves as the reference group is arbitrary, but it will affect the interpretation of model coefficients. For this reason, if your categorical variable is ordinal, it is customary to select either the lowest or the highest ordered category as your reference group; Interpretation of coefficients relative to max or min just often makes the most sense.

6.2 Indicator only Regression Model

While not commonly done, you can create a simple linear regression model using only an indicator variable as a predictor. Consider again the `cats` data frame in the `MASS` library. This data contains not only the heart weight and body weight of 144 cats as we've seen in earlier models, but also the sex of those 144 cats. From Figure 6.1 below it is clear that the heart weight of female cats is a bit less on average than the heart weight of male cats.

If we fit a model of the form $y = \beta_0 + \beta_1 x + \varepsilon$ and have $x = 0$ for female cats and $x = 1$ for male cats what values of β_0 and β_1 should we expect? Since there are only two possible values



(a) Straight data

(b) Jittered sex for clarity

Figure 6.1: Cat heart weight by sex, 1=Female

of x, there will be only two possible fitted values from our model: one corresponding to when $x = 0$ and one for when $x = 1$. To minimize the sum of squared error, we want the output when $x = 0$ to be equal to the mean heart weight for female cats and the output when $x = 1$ to be equal to the mean heart weight for male cats. This is achieved when β_0 equals the mean heart weight for females and when $\beta_0 + \beta_1$ mean heart rate for males. Therefore β_1 should be the difference between the two group means.

Here's the work in R:

```
mean(cats$Hwt[cats$Sex=="F"])
```

```
[1] 9.202128
```

```
mean(cats$Hwt[cats$Sex=="M"])-mean(cats$Hwt[cats$Sex=="F"])
```

```
[1] 2.120553
```

```
mod_cat_sex<-lm(Hwt~Sex, data=cats)
summary(mod_cat_sex)$coef
```

```
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 9.202128  0.3250734 28.307842 2.959034e-60
SexM        2.120553  0.3960745  5.353924 3.379786e-07
```

Just as suspected, the result gives us an intercept equal to the mean heart weight of female cats and a slope equal to the difference between male and female heart weights.

6.3 Adding in an Indicator

You can also add an indicator term to a model to create a linear equation of the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

Continuing with the cat heart weight example, we could fit this type of model with y = heart weight, x_1 = body weight, and x_2 = sex (0=female, 1 = male)

The resulting model fit is:

```
               Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -0.41495263  0.7273243 -0.5705194 5.692336e-01
Bwt          4.07576892  0.2947885 13.8260785 5.119676e-28
SexM        -0.08209684  0.3040474 -0.2700133 7.875448e-01
```

How can this be interpreted? First, note that the third row begins with `SexM` this is R's way of communicating that for the Sex variable level M is the one that is corresponding to the indicator equalling 1. This tells us that the heart weight of cats is, on average, equal to -0.415 plus (4.0758 × body weight) for female cats, and (-0.415 plus -0.0821) plus (4.0758 × body weight) for male cats. Same slope for all cats, but intercepts that differ by -0.0821.

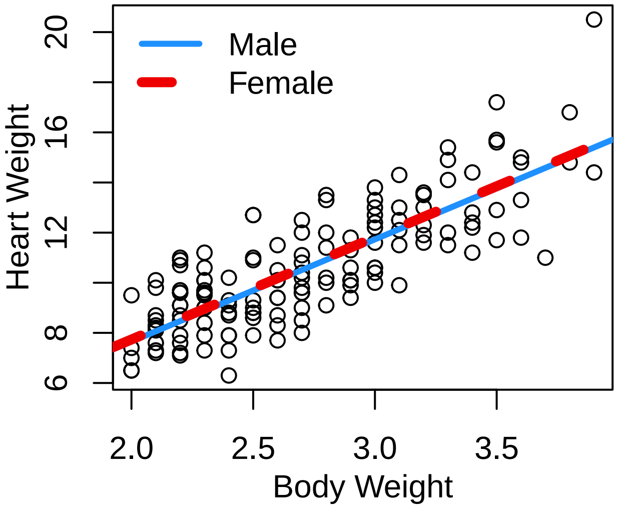


Figure 6.2: Cat heart weight as function of body weight

Hmmm... the two lines look pretty much the same. Why is that? Well take a look again at the model output above. We see that the difference in intercepts is only -0.0821, not a very big number. But scanning across this line in the output we also see that the p-value in the t-test of $H_o : \beta_2 = 0$ is 0.788 which means that with any reasonable α level, we would fail to reject the null. This means the coefficient associated with sex is not significantly different from zero, and the two lines aren't really needed - one will do just fine.

Other times the indicator will matter quite a bit. Consider the `mtcars` dataframe from the `datasets` library. In a model predicting vehicle fuel efficiency as a function of horse power, does it help if we add in an indicator that is 1 for a manual transmission and 0 for an automatic? Our model will again be in the form of:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

with y =mpg, x_1 =horsepower, and x_2 =manual transmission.

```
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 26.5849137 1.425094292 18.654845 1.073954e-17
hp          -0.0588878 0.007856745 -7.495191 2.920375e-08
am           5.2770853 1.079540576  4.888270 3.460318e-05
```

In this case, the test of $H_o : \beta_2 = 0$ has a p-value that is 3×10^{-5} meaning there is strong evidence this additional term related to transmission type matters. In a plot of the fitted model we see:

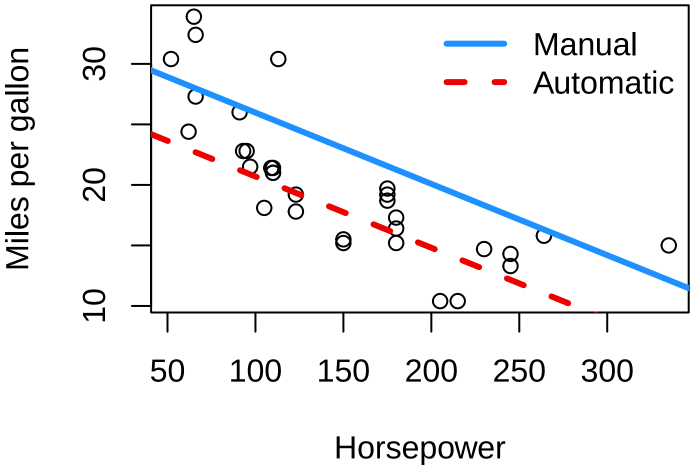


Figure 6.3: Car MPG as function of HP

Figure 6.3 shows us very clearly that cars of equal horsepower tend to get better gas mileage if they are a manual transmission. The difference between the two lines is β_2, 5.2771.

6.4 Interacting with an Indicator

Parallel lines are great and all, but there's no reason to believe that the best model fit for two levels of your indicator should always have the same slope. Ideally we want a model that takes a form more like:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \varepsilon$$

With this structure, for our baseline level of $x_2 = 0$, the β_2 and β_3 terms disappear since they are multiplied by zero, leaving you with the basic β_0 as intercept and β_1 as slope. When $x_2 = 1$ though, the β_2 and β_3 terms stick around and $\beta_0 + \beta_2$ becomes the intercept and $\beta_1 + \beta_3$ becomes the slope. Having a $x_1 x_2$ term is called having an interaction between x_1 and x_2. The x_2 indicator value is interacting with the slope associated with your continuous x_1.

Revisiting the cat heart and body weight data, might sex play a significant role in predicting heart weight if we allow slope to change as well as intercept? Here's the R model summary for the revised model that includes the $\beta_3 x_1 x_2$ term:

```
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  2.981312  1.8428394  1.617782 0.1079604636
Bwt          2.636414  0.7759022  3.397869 0.0008845733
SexM        -4.165400  2.0617552 -2.020318 0.0452578391
Bwt:SexM     1.676265  0.8373255  2.001927 0.0472246471
```

The last row labeled `Bwt:SexM` corresponds to our $\beta_3 x_1 x_2$ term. Reading across the output summary table shows us that this β_3 coefficient is estimated to be significantly different from 0 at the $\alpha = 0.05$ level. The β_2 term allowing for different intercepts that was not significant in the limited $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ model is now also showing a p-value below 0.05 at 0.0453. Plotted, the fits for male and female cats now look like the lines in Figure 6.4.

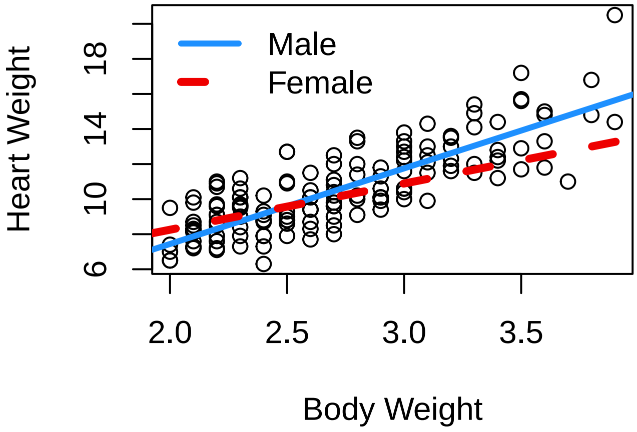


Figure 6.4: Cat heart weight as function of body weight

In R

In the above example using mtcars, you may have noticed that the reference level wasn't clear in the model summary output. That is because the transmission type variable, am, is pre-coded as a 0/1 indicator variable. The help menu for mtcars explains that am is a 0 for automatic, and 1 for manual. When read into R, it takes this 0/1 coding as a numeric 0/1 and doesn't think of it as a factor type needing a reference level at all. Mathematically that is fine - it just makes interpretation a little trickier because you need to remember what is coded as 0 and what is coded as a 1.

Here is the code for the model described earlier predicting mpg as a function of horsepower and transmission type (am) :

```
# adding on +am to mpg~hp creates an additive beta for transmission
mod_mpg<-lm(mpg~hp+am, data=mtcars)
summary(mod_mpg)$coef
```

```
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 26.5849137 1.425094292 18.654845 1.073954e-17
hp          -0.0588878 0.007856745 -7.495191 2.920375e-08
am           5.2770853 1.079540576  4.888270 3.460318e-05
```

A simple run of ?mtcars to pull up the help menu will show you that am=0 for automatics and therefore automatic is the baseline reference level for transmission type and the β_2 shown for am is the change in intercept associated with a manual transmission.

In the cats data set explored this chapter, Sex is coded with F and M levels and is recognized by R as a factor type variable. You can include factors in your lm function input formula as-is and R will take care of the 0/1 coding behind the scenes for you. The first level of your factor will be treated as the 0 reference level. Here is code that first looks at the levels of Sex, then creates and summarizes the cat heart weight model including indicator interaction:

```
# look at levels to know which will be reference group
levels(cats$Sex)
```

```
[1] "F" "M"
```

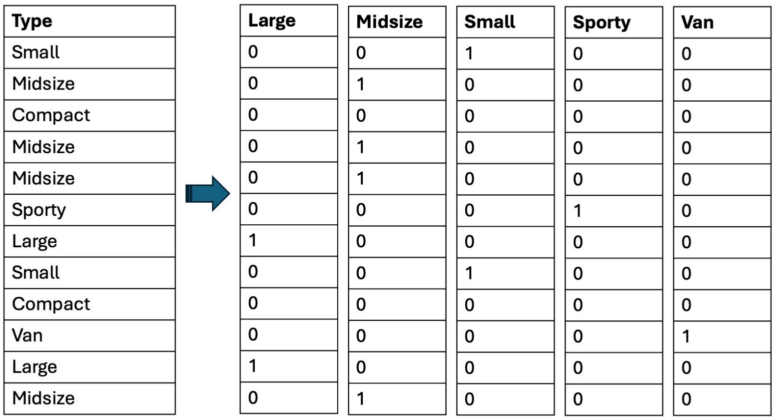
```
# multiplying the indicator factor creates both an additive beta for
#   different intercepts and the beta for the slope change
mod_cats_full<-lm(Hwt~Bwt*Sex, data=cats)
summary(mod_cats_full)$coef
```

```
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  2.981312  1.8428394  1.617782 0.1079604636
Bwt          2.636414  0.7759022  3.397869 0.0008845733
SexM        -4.165400  2.0617552 -2.020318 0.0452578391
Bwt:SexM     1.676265  0.8373255  2.001927 0.0472246471
```

As pointed out earlier, the row labels of `SexM` and `Bwt:SexM` also make clear level M is the one that is being treated as a 1, meaning level F must be our 0 reference group. If you want that switched, you can use `relevel` to change the ordering of the levels of the factor variable.

6.5 Multi-level factors in Regression

For categorical variables with more than two categories, we create a separate indicator variable for all categories except one, our baseline reference group. The below illustrates this process with an example from the `Cars93` data set from the `MASS` library. Vehicle type is a factor that takes on six different values: Compact, Large, Midsize, Small, Sporty, and Van. To incorporate this one factor with six levels, we create five new variables: Large, Midsize, Small, Sporty, and Van; each coded as a 0 or a 1. We do not need a variable for Compact - the Compact car level is signaled by not being any of the other types. This is often called one-hot encoding.



Type
Small
Midsize
Compact
Midsize
Midsize
Sporty
Large
Small
Compact
Van
Large
Midsize

Large	Midsize	Small	Sporty	Van
0	0	1	0	0
0	1	0	0	0
0	0	0	0	0
0	1	0	0	0
0	1	0	0	0
0	0	0	1	0
1	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	1
1	0	0	0	0
0	1	0	0	0

Figure 6.5: One-hot encoding example

This is how car type can be included in a regression model - through the addition of five indicators. The order of the levels of car `Type` is, by default, alphabetical:

```
levels(Cars93$Type)
```

```
[1] "Compact" "Large"   "Midsize" "Small"   "Sporty"  "Van"
```

To build a model of vehicle highway mpg as a function of weight, allowing for different intercepts for each type, the code looks much like what we've seen with the `lm` command before:

```
mod_car93<-lm(MPG.highway~Weight+Type, data=Cars93)
summary(mod_car93)$coef
```

```
                Estimate  Std. Error     t value     Pr(>|t|)
(Intercept) 49.532032100 3.308855496 14.96953619 1.126489e-25
Weight      -0.006736186 0.001104706 -6.09772023 2.969242e-08
TypeLarge    2.088508950 1.450279960  1.44007296 1.534773e-01
TypeMidsize  0.098272245 1.115603939  0.08808883 9.300109e-01
TypeSmall    1.523993750 1.194802165  1.27551974 2.055600e-01
TypeSporty  -1.213784862 1.092190088 -1.11133115 2.695231e-01
TypeVan     -1.839809387 1.600556866 -1.14948080 2.535445e-01
```

The main difference here is that withe the simple `+Type` we've now added five more estimates; one for each of the new indicator terms. So the given intercept of 49.532 is the intercept for the baseline Compact car type, then we add 2.0885 to that to get the intercept for Large cars, add 0.0983 to 49.532 to get the intercept for Midsize cars, and so on through to adding -1.8398 to 49.532 for Vans.

Note in these results none of the p-values in the last column for these Type-related s indicate the is significantly different from 0. This does NOT mean that no two vehicle types have significantly different intercepts, it just means that none of the vehicle types have an intercept significantly different from Compact cars. If a different car type were selected to be the reference level, these estimates and their corresponding p-values would change. Below are the results from the same model but using Van as the baseline reference rather than Compact.

```
Type2<-relevel(Cars93$Type, "Van")
mod_car93_Van<-lm(MPG.highway~Weight+Type2, data=Cars93)
summary(mod_car93_Van)$coef
```

```
                 Estimate  Std. Error    t value     Pr(>|t|)
(Intercept)  47.692222713 4.346958390 10.9714008 5.017827e-18
Weight       -0.006736186 0.001104706 -6.0977202 2.969242e-08
Type2Compact  1.839809387 1.600556866  1.1494808 2.535445e-01
Type2Large    3.928318337 1.349446429  2.9110591 4.586149e-03
Type2Midsize  1.938081632 1.272889330  1.5225846 1.315314e-01
Type2Small    3.363803137 2.055312537  1.6366383 1.053606e-01
Type2Sporty   0.626024525 1.637942803  0.3822017 7.032546e-01
```

The resulting linear equations are exactly the same. Slope is obviously the same as before, and the intercepts are just expressed with relation to a different baseline. Intercept for Midsize cars for example was originally 49.532 + 0.0983=49.6303 and in the newer model the Midsize intercept is 47.6922 + 1.9381=49.6303. So no real change… just a different comparitor in the baseline position. You'll also see that with Van as the baseline, the associated with Large cars now has a p-value below 0.05 indicating it is significantly different from zero.

For the full model with interactions between car type and weight the addition of *Type leads to the addition of 10 new estimates: five for unique intercepts plus five for unique slopes. And just as we saw with the additive indicator model, the values of the estimates will change depending on what factor level is used as the baseline reference level.

With the default Compact as the reference level of type:

```
mod_car93_full<-lm(MPG.highway~Weight*Type, data=Cars93)
summary(mod_car93_full)$coef
```

```
                        Estimate   Std. Error    t value     Pr(>|t|)
(Intercept)         4.781871e+01 10.199629443  4.6882791 1.099859e-05
Weight             -6.149055e-03  0.003486294 -1.7637797 8.153978e-02
TypeLarge          -7.310074e+00 16.890945386 -0.4327807 6.663243e-01
TypeMidsize        -2.810940e+00 12.244349583 -0.2295704 8.190043e-01
TypeSmall           1.774275e+01 11.678767728  1.5192314 1.325981e-01
TypeSporty         -3.170950e+00 11.741827087 -0.2700559 7.878041e-01
TypeVan            -1.878289e+01 27.646402307 -0.6793971 4.988232e-01
Weight:TypeLarge    2.419781e-03  0.005036981  0.4804029 6.322358e-01
Weight:TypeMidsize  7.724377e-04  0.004011302  0.1925653 8.477814e-01
Weight:TypeSmall   -6.858783e-03  0.004257682 -1.6109196 1.110875e-01
Weight:TypeSporty   6.787099e-04  0.004013261  0.1691168 8.661264e-01
Weight:TypeVan      4.283285e-03  0.007555762  0.5668898 5.723567e-01
```

And with Van as the reference level of type:

```
mod_car93_full2<-lm(MPG.highway~Weight*Type2, data=Cars93)
summary(mod_car93_full2)$coef
```

```
                        Estimate   Std. Error    t value  Pr(>|t|)
(Intercept)         29.035823951 25.696130443  1.1299687 0.2618244
Weight              -0.001865770  0.006703380 -0.2783327 0.7814662
Type2Compact        18.782885727 27.646402307  0.6793971 0.4988232
Type2Large          11.472811300 29.009700360  0.3954819 0.6935270
Type2Midsize        15.971945860 26.574099715  0.6010343 0.5494954
```

```
Type2Small          36.525636382 26.318288216  1.3878424 0.1689910
Type2Sporty         15.611936118 26.346331477  0.5925658 0.5551224
Weight:Type2Compact -0.004283285  0.007555762 -0.5668898 0.5723567
Weight:Type2Large   -0.001863504  0.007625761 -0.2443696 0.8075625
Weight:Type2Midsize -0.003510847  0.006990822 -0.5022080 0.6168838
Weight:Type2Small   -0.011142068  0.007135048 -1.5615968 0.1222818
Weight:Type2Sporty  -0.003604575  0.006991947 -0.5155324 0.6075853
```

On Your Own

1. Continuing with the examples above in Section 6.5:

   a. What is the linear function to predict the highway fuel efficiency of a large car that weighs 3800 lbs using the model with Compact cars as the reference level?

   b. Confirm the linear function from (a) still applies if you instead use the model fit with Vans as the reference level.

   c. If the reference level is changed to Sporty, what do you expect the `TypeSmall` and `Weight:TypeSmall` coefficient estimates to be? Explain the reasoning and answer without running the new model.

   d. Create a plot showing the six fit lines predicting highway mpg as a function of vehicle weight - one line per vehicle type. You'll need the `Cars93` data set contained in the `MASS` library.

   e. Are all assumptions necessary for inference met? Explain.

2. Consider all cats with a body weight 3.5kg or less in the `cats` data frame from the MASS library.

   a. Fit a simple linear regression model estimating a cat's heart weight as a function of body weight. What is the equation of your fitted model?

   b. Are all assumptions for inference met by your model in (a)? Explain.

   c. Is there sufficient evidence that the slope is greater than 3? Explain.

   d. Now fit a model estimating heart weight using body weight interacting with cat sex. What is the equation of the fitted model?

   e. Are all assumptions for inference met by your model in (d)? Explain.

   f. Is there sufficient evidence that the slope is greater than 3 for male cats? Is there evidence that the slope is less than 3 for female cats? Use an $\alpha = 0.05$ level and explain your findings thoroughly.

g. Create a plot showing both the male and female cat fit lines. Jitter your body weights and use either plotting characters or color to distinguish which sex applies to each data point.

h. Why is it perfectly reasonable to calculate a prediction interval for the heart weight of a male cat weighing 3.3 kg, but not a good idea to make a prediction interval for a female cat weighing 3.3 kg?

3. The crabs data set in the MASS library contains a variety of body measurements on two species of crabs.

   a. Estimate a crab's body depth as a function of carapace width, including crab species information in your model. What type of model makes the most sense: a model with an additive inclusion of a species indicator, or a model with a multiplicative interaction with a species indicator? Explain your answer.

   b. What is the fitted equation of your selected model?

   c. Do Orange crabs and Blue crabs have a significantly different intercept in a linear model using carapace width to explain body depth? Explain.

   d. Are all assumptions for inference met in your model? Explain, using an $\alpha = 0.01$ significance level in any tests you run. If your model needs fixing for inference procedures to be valid, fix it.

   e. Using a model that meets all assumptions at the $\alpha = 0.01$ level, fit and interpret a 90% confidence interval for the mean body depth of a Blue crab with a carapace width of 40mm. Then fit and interpret a 90% confidence interval for the mean body depth of an Orange crab with a carapace width of 40mm. How do they compare?

   f. Now complete a model predicting body depth as a function of carapace width using an indicator term for crab sex. What indicator term is more helpful in predicting body depth: sex or species? Explain.

4. The `openintro` library includes a data frame called fastfood that contains nutrition information for 515 fast food items.

   a. Fit a model for food item calories as a function of total fat content. Include restaurant as an interaction term in your model. Which two restaurants have the most similar fit lines? What are those two fit lines?

   b. Which restaurant has the steepest slope? Which has the least-steep slope? Are the two slopes significantly different from each other?

   c. Fit a new model that does not include restaurant. Which restaurant from your first model comes closest to matching this restaurant-blind overall fit for calories as a function of fat? Explain.

d. Create a plot showing the raw data with three lines: the fit for the restaurant with the steepest slope, the fit for the restaurant with the least-steep slope, and the fit for the restaurant that most closely matched the restaurant neutral fit. Include a legend that provides the name of the restaurant associated with each model fit.

7 Go Bigger

7.1 Multiple Regression

Multiple linear regression is just like simple linear regression, but instead of using only one independent variable (X) to predict the outcome (Y), it uses two or more independent variables at the same time. Each independent variable gets its own coefficient ($\beta_1, \beta_2, \beta_3$... etc.), representing its unique effect on the dependent variable while keeping the others constant. This allows you to see how each factor influences the outcome, controlling for the others.

So for a model with p independent predictor terms, the model now takes the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_p x_p + \varepsilon$$

As you can see, it gets to be a bit much to write out this long-form notation, hence the beauty and value of the alternative matrix approach. Recall that in addition to describing a linear regression model with a simple algebraic linear equation like $y = \beta_0 + \beta_1 x_1 + \varepsilon$ our model can also be expressed using matrix notation as:

$$Y = X\beta + \varepsilon$$

While up until now we've only worked in the situation where X is an $n \times 2$ matrix with the first column full of 1s to correspond to the intercept term, there is no reason X can't be $n \times (p + 1)$ where p is any number of parameters we'd like to include in our quest to model Y. All we need to do is expand β to be $(p + 1) \times 1$ to match.

The matrix approach to solve for $\hat{\beta}$ is still calculated by:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

This of course means that the solution for β still requires $X^T X$ have an inverse. No inverse, no $\hat{\beta}$, no matter the dimensions.

What does this look like in practice? Consider the `hills` data set in the `MASS` library that contains the winning times for 35 Scottish hill races in 1984. These races range from a greulling 16km length with a climb of 7500 meters that takes hours to simpler runs of 3km rising 300 meters that are done in under 20 minutes. Since both distance and elevation change obviously play a role in the challenge presented by a race, it would make sense to include both distance and climb in a model predicting the winning time. We want a model that looks like:$winningtime = \beta_0 + (\beta_1 \times distance) + (\beta_2 \times climb) + \varepsilon$.

The matrix form of our model data is:

$$Y = \begin{bmatrix} 16.083 \\ 48.350 \\ 33.650 \\ \vdots \\ 159.833 \end{bmatrix} and X = \begin{bmatrix} 1 & 2.5 & 650 \\ 1 & 6 & 2500 \\ 1 & 6 & 900 \\ \vdots & \vdots & \\ 1 & 20 & 5000 \end{bmatrix} \tag{7.1}$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y = \begin{bmatrix} -8.992 \\ 6.218 \\ 0.011 \end{bmatrix}$$

Through R we can obtain this same fit with:

```
library(MASS)
mod_hills<-lm(time~dist+climb, data=hills)
summary(mod_hills)$coef
```

```
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) -8.99203896 4.302734388 -2.089843 4.466516e-02
dist         6.21795571 0.601147884 10.343471 9.859214e-12
climb        0.01104791 0.002050892  5.386882 6.445183e-06
```

The approach to using and interpreting this model is similar to if only one predictor term was used. To use it for prediction, we just plug in values of x_1 and x_2 and note the resulting y. For example, to estimate the winning race time for a new race that's 10km with an elevation change of 3100m, our our model would suggest $-8.992 + (6.218 \times 10) + (0.011 \times 3100) = 87.436$ minutes. The intercept tells us the winning race time expected for a hypothetical race that is 0km long with 0m climb: an end almost 9 minutes before the race begins. Not a meaningful answer but not a possible real race either. Our slopes are now in two different dimensions making for a fit plane in 3-dimensional space rather than a fit line. It also makes a plot of the data and the fit hard to do. Interpreting the slopes one by one we see that each additional km of distance adds, on average, about 6.2 minutes to the finishing time and each additional meter of climb adds, on average, 0.011 minutes which is less than one second.

7.2 Variance and Inference for $\hat{\beta}$

For standard errors in multiple regression we'll continue with the matrix notation for our model and rely more on your linear algebra skills. Start with:

$$\hat{\beta} = (X^TX)^{-1}X^TY$$

Since our model is based on $Y = X\beta + \varepsilon$ we can then plug in $X\beta + \varepsilon$ for Y to get:

$$\hat{\beta} = (X^TX)^{-1}X^T(X\beta + \varepsilon)$$

which reduces to

$$\hat{\beta} = \beta + (X^TX)^{-1}X^T\varepsilon$$

thanks to $(X^TX)^{-1}X^TX = I$. Unlike our estimate $\hat{\beta}$, the parameter β is an unknown constant and therefore has zero variance. This means

$$var(\hat{\beta}) = var((X^TX)^{-1}X^T\varepsilon)$$

A property of random vectors says that if A is a matrix and v a random vector, then $var(Av) = Avar(v)A^T$. Applying that, we now can express $var(\hat{\beta})$ as:

$$var(\hat{\beta}) = (X^TX)^{-1}X^Tvar(\varepsilon)X(X^TX)^{-1} \tag{7.2}$$

Recall one of our big assumptions is that $Var(\varepsilon)$ is constant. As part of that, we denote $Var(\varepsilon)$ as simply σ^2I where σ is that constant variance. So now:

$$var(\hat{\beta}) = (X^TX)XT^{-1}\sigma^2IX(X^TX)^{-1}$$

$$var(\hat{\beta}) = \sigma^2I(X^TX)^{-1}X^TX(X^TX)^{-1}$$

$$var(\hat{\beta}) = \sigma^2I(X^TX)^{-1}$$

This is what R is doing under the hood when it produces your column of standard errors for the model summary output. This holds whether $\hat{\beta}$ is a simple 2×1 vector with an intercept and slope for a single predictor like we saw back in Chapter 3, or whether $\hat{\beta}$ is a much larger $(p+1) \times 1$ vector with an intercept and slopes for p predictors.

An example demonstrating this in R on the hill race model:

```
xmat<-matrix(c(rep(1, nrow(hills)), hills$dist, hills$climb), ncol=3)

sigma<-summary(mod_hills)$sigma

var_b<-(sigma^2)*solve(t(xmat)%*%xmat)

#diagonal is the variance of each beta, off-diagonals are covariances
var_b
```

```
             [,1]          [,2]          [,3]
[1,] 18.513523217 -1.2606623775 -1.580489e-03
[2,] -1.260662378  0.3613787787 -8.042704e-04
[3,] -0.001580489 -0.0008042704  4.206156e-06
```

```
#but standard errors are square-roots of variances
c(sqrt(var_b[1,1]), sqrt(var_b[2,2]), sqrt(var_b[3,3]))
```

```
[1] 4.302734388 0.601147884 0.002050892
```

```
#compare to output from model summary
summary(mod_hills)$coef
```

```
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) -8.99203896 4.302734388 -2.089843 4.466516e-02
dist         6.21795571 0.601147884 10.343471 9.859214e-12
climb        0.01104791 0.002050892  5.386882 6.445183e-06
```

Why does variance of $\hat{\beta}$ matter? Inference in multiple regression is just like inference in the single-predictor simple linear regression with confidence intervals that take the form of:

$$point\ estimate \pm multiplier \times standard\ error$$

and hypothesis test statistics have the general format of:

$$\frac{observed\ value - hypothesized\ value}{standard\ error}$$

so obviously standard error is a key element of both. Confidence intervals for each β coefficient can be estimated from $\hat{\beta}_i \pm t_{\alpha/2} \times SE_{\hat{\beta}_i}$ where t has $n-(p+1)$ degrees of freedom and tests of $\beta_i = 0$ have a test statistic of $t = \hat{\beta}_i/SE_{\hat{\beta}_i}$ with p-values calculated from a $t_{n-(p+1)}$ distribution.

The easiest way to do confidence intervals for β in R is still the `confint` function as seen earlier with single predictor models.

```r
mod_hills<-lm(time~dist+climb, data=hills)
confint(mod_hills, level=.9)
```

```
                      5 %        95 %
(Intercept) -16.280392325 -1.70368559
dist          5.199678069  7.23623334
climb         0.007573928  0.01452189
```

For hypothesis tests on β, all of the information you need for the most common test of $H_0 : \beta_i = 0$ is given in the model summary output. If instead you are interested in testing $H_0 : \beta_i = b$ for some value b, you can simply calculate your test statistic from the standard error as given in the model summary table, and the corresponding p-value using the `pt` function. For example, if we wanted to test the hypothesis that each additional km of distance added an average of more than 5 minutes to the race time, that would be testing $H_0 : \beta_{dist} \leq 5$ with $H_A : \beta_{dist} > 5$. The test statistic would be calculated as:

```r
summary(mod_hills)$coef
```

```
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) -8.99203896 4.302734388 -2.089843 4.466516e-02
dist         6.21795571 0.601147884 10.343471 9.859214e-12
climb        0.01104791 0.002050892  5.386882 6.445183e-06
```

```r
#using estimate and std. error for dist:
t_stat<-(6.217956-5)/0.601148

#find upper tail since alternative is upper tail
pt(t_stat, nrow(hills)-3, lower.tail = FALSE)
```

```
[1] 0.02558215
```

With an $\alpha = 0.05$, this is sufficient evidence that the mean increase in race time for every additional km of distance is indeed greater than 5 minutes.

7.3 Variance and Inference for fitted Y

Recall $\hat{y} = X\hat{\beta} = Hy$ where $H = X(X^TX)^{-1}X^T$ and is known as the “hat” matrix. Therefore, $var(\hat{y}) = Hvar(y)H^T$ using the same property applied in Equation 7.2. Fun fact about H: it is both symmetric and idempotent. This means $H = H^T$ and $HH = H$. Assuming Y is distributed $N(X\beta, \sigma^2)$ we can then plug in $\sigma^2 I$ for $var(y)$ yielding:

$$var(\hat{y}) = \sigma^2 H$$

which means the variance for any specific fitted $\hat{y}_i$ is:

$$var(\hat{y}) = \sigma^2 h_{ii} \tag{7.3}$$

the variance of y, multiplied by the i^{th} diagonal of the hat matrix.

This is great, but more often in regression we aren’t interested in just the variance of specific fitted values for the data set the model trained on. Usually un-observed x^* values are plugged in to our fitted model to estimate additional y outcomes. When that’s the case there is no h_{ii} that applies. This takes us back to $\hat{y} = X\hat{\beta}$ but now instead of the full X we only have the vector x_* so:

$$var(\hat{y}_*) = \sigma^2 x_*^T (X^TX)^{-1} x_*$$

This is the same as formula you saw for the simple linear case back in chapter 3:

$$SE_{\bar{Y}|x_*} = S_R \sqrt{\frac{1}{n} + \frac{(x_* - \bar{x})^2}{\sum (x_i - \bar{x})^2}} \tag{7.4}$$

The matrix algebra is what’s going on behind the scenes in R no matter how many predictors you have. Writing it out longhand as in Equation 7.4 is much harder to do for multiple predictors, but seeing it that way for the simple case makes it easier to see that when x_* values are close to the mean x values the standard error is smaller making the resulting intervals narrower.

The `predict` function used for confidence and prediction intervals for y_* in the simple linear case continues to do the job for multiple regression models.

Consider the 90% prediction intervals below for a race with a distance of 7km and climb of 1800m (values near the respective means) and a race with a distance 23km and a climb of 6000m (more unusual but within the range observed in the training set).

```
new_races<-data.frame(dist=c(7, 23), climb=c(1800, 6000))
predict(mod_hills, new_races, interval="prediction", level=.9)
```

```
        fit       lwr       upr
1  54.41989  29.20351  79.63627
2 200.30840 172.08899 228.52782
```

The race first race has a narrower interval because it involves x_* nearer the center of the training X values.

7.4 Transformations

7.4.1 Units change on one term

We begin with the simplest type of transformation on a single predictor: a unit change. Maybe meters of climb isn't the unit of measurement we want to use for climb in our hill racing model. Let's change it so that we can get the amount of time added per km of climb so that the units on climb will match the units on distance. That means instead of what we saw earlier in Equation 7.1 now:

$$Y = \begin{bmatrix} 16.083 \\ 48.350 \\ 33.650 \\ \vdots \\ 159.833 \end{bmatrix} and X = \begin{bmatrix} 1 & 2.5 & 0.650 \\ 1 & 6 & 2.5 \\ 1 & 6 & 0.9 \\ \vdots & \vdots & \\ 1 & 20 & 5.0 \end{bmatrix}$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y = \begin{bmatrix} -8.992 \\ 6.218 \\ 11.048 \end{bmatrix}$$

Or through R code:

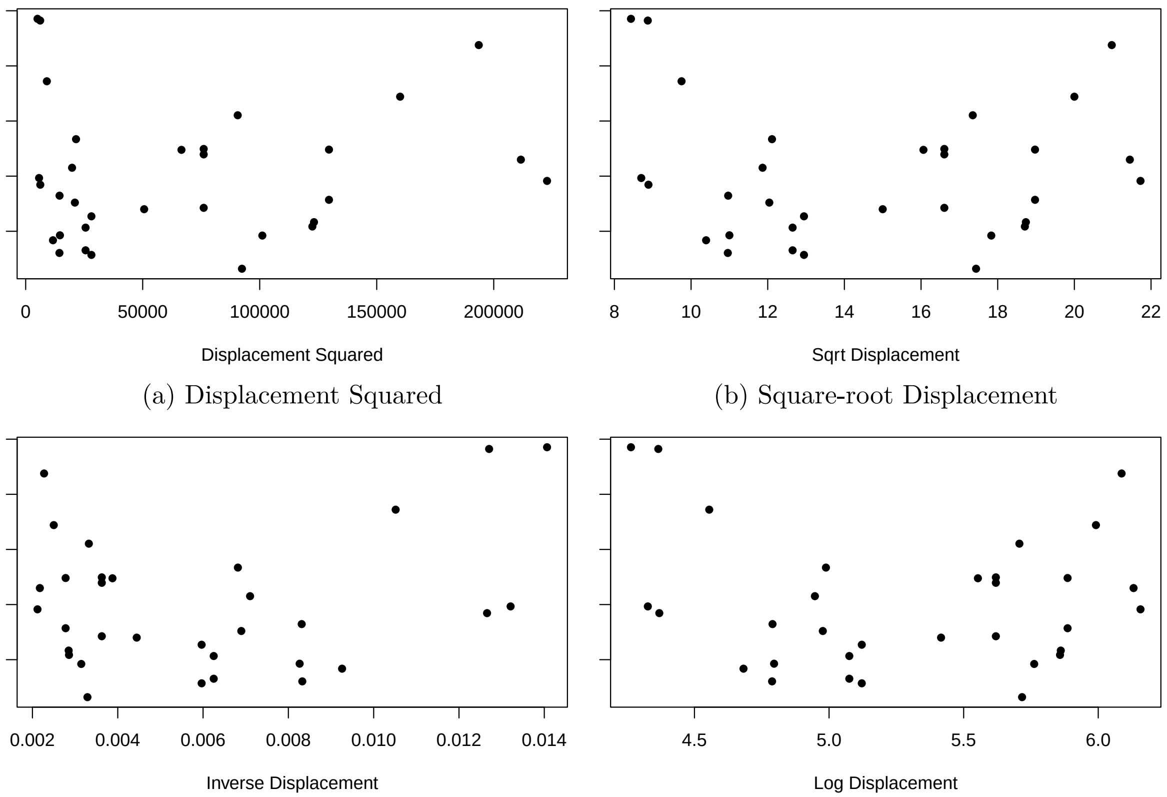
```
climb_km<-hills$climb/1000
mod_hills2<-lm(time~dist+climb_km, data=hills)
summary(mod_hills2)$coef
```

```
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) -8.992039  4.3027344 -2.089843 4.466516e-02
dist         6.217956  0.6011479 10.343471 9.859214e-12
climb_km    11.047910  2.0508916  5.386882 6.445183e-06
```

Which makes perfect sense and behaves just as we saw in the simple linear example from Chapter 5: we divided our x_2 by 1000, so to get the same result from $\beta_2 \times x_2$ the value of β_2 would need to increase by a factor of 1000. And now our interpretation includes that the winning time increase by about 11 minutes on average for every km of climb added.

Any time you want to make a units change on a predictor term the β estimate for the associated variable will simply change by the inverse and all other will be unaffected. Standard errors will change in scale as well which will leave you with parameter test statistics and p-values that are unchanged. The quality of your fit as measured by R^2 won't change at all either so use whatever scales you'd like predictor by predictor.

7.4.2 One or more X terms



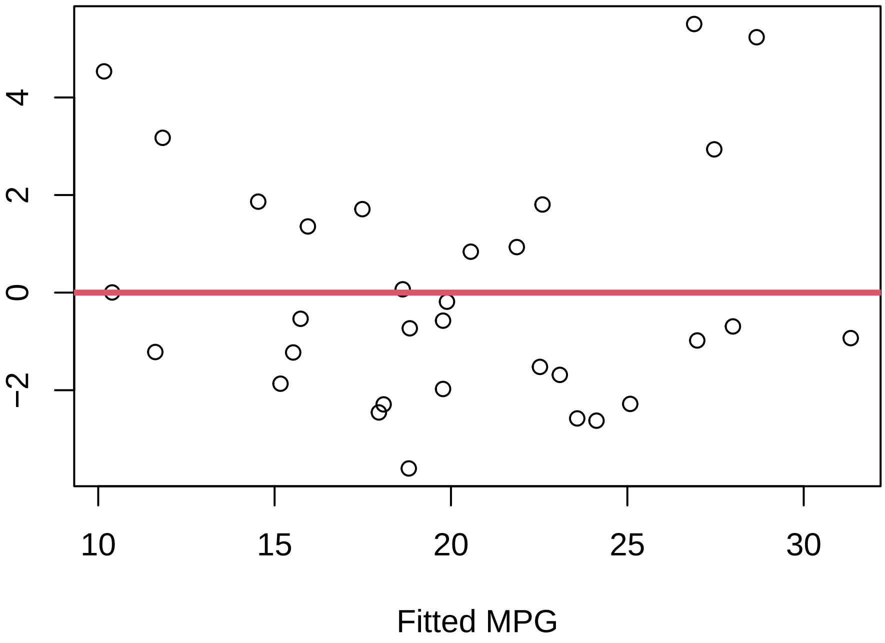
(a) Displacement Squared

(b) Square-root Displacement

(c) Inverse Displacement

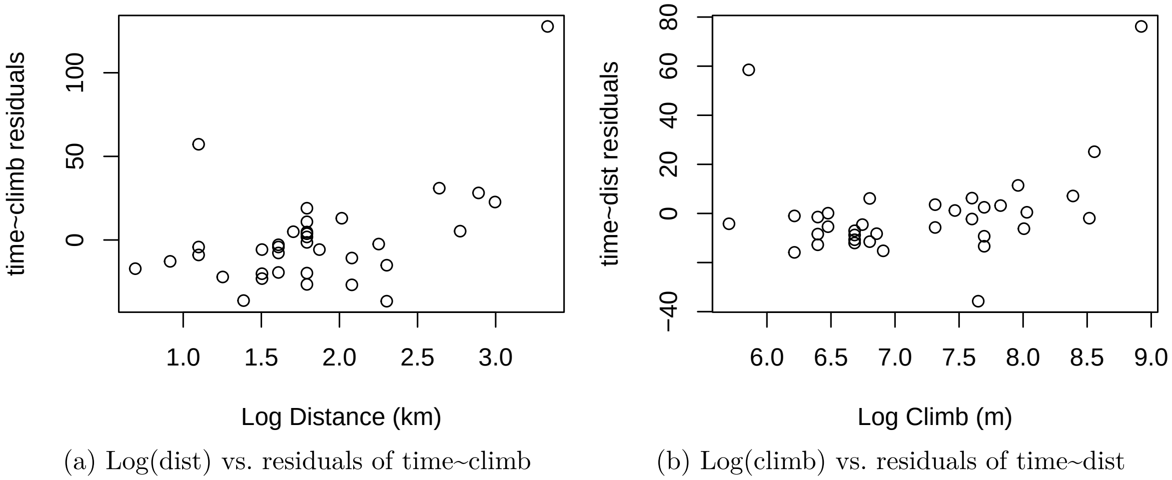
(d) Log Displacement

Figure 7.1: Displacement Transformations and Residuals



(a) Model with Displacement^2 plus HP, Axle Ratio, and Wt

Figure 7.2: Motor Trend Car Data



(a) Log(dist) vs. residuals of time~climb

(b) Log(climb) vs. residuals of time~dist

Figure 7.3: Scottish Hill Races, Log transformations

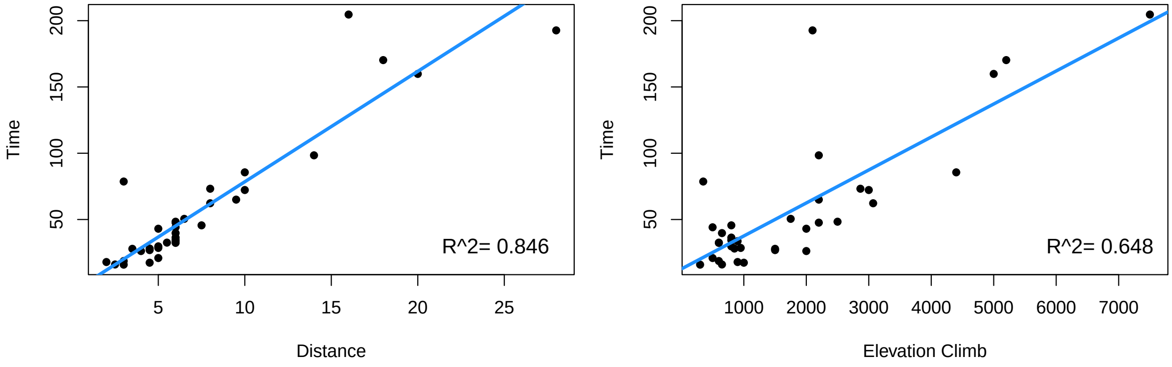
7.4.3 Transforming Y

Why would you want to transform Y instead of one or more of your X terms? Well, what if what you really want for the format of your model is $y = e^{\beta_0 + x_1\beta_1 + x_2\beta_2 + \epsilon}$? You aren't able to fit that type of relationship directly with the least-squares regression methods we've covered, but you could use a log transform on y and then fit $log(y) = \beta_0 + x_1\beta_1 + x_2\beta_2 + \epsilon$ without any trouble. Generally, any time you think the relationship at hand is of the form $y = f(\beta_0 + x_1\beta_1 + x_2\beta_2 + \epsilon)$ the solution is to transform your y with f' and fit the $f'(y) = \beta_0 + x_1\beta_1 + x_2\beta_2 + \epsilon$ model.

7.5 Plotting for Multiple Regression

While the basics of fitting a model and using it for inference are largely unchanged from our simple linear regression modeling, expanding beyond one independent predictor variable in our model makes plotting the data more complicated (but also more vital). It's more complicated because we can generally only plot two continuous dimensions at a time (in a way our brains can understand) simply because our screens (and paper) are two-dimensional, but more vital because there are now even more nuances to the relationships of the variables that we need to know about.

A good first step is to look at the simple pairwise relationships between each indpendent x predictor and the dependent outcome y. This means for the hill data we'd consider the plots shown in Figure 7.4.



(a) Distance and Time

(b) Climb and Time

Figure 7.4: Scottish Hill Race Data

From Figure 7.4 it is clearly seen that both the distance of the race and the elevation climb of the race are reasonably linearly related to our outcome of race record time. R^2 values for the two simple linear models tell us that distance alone explains 84.6% of the variability in race time and elevation climb explains 64.8% of race time variability. The next step now is to consider how your independent variables relate to each other.

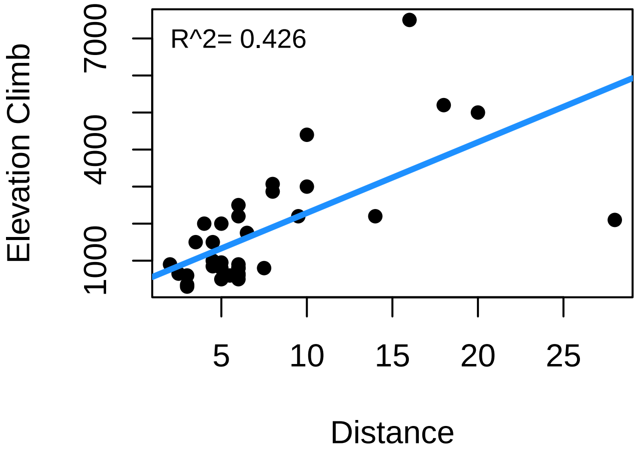
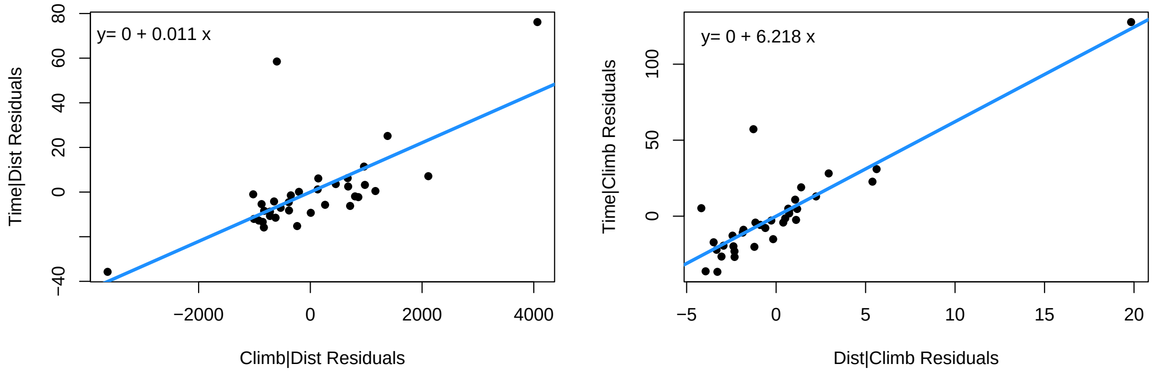


Figure 7.5: Scottish Hill Race Distance and Elevation Climb

This is the relationship shown in Figure 7.5. As one might expect, race distance and elevation change are related to each other. This makes intuitive sense since longer races have greater opportunity for changes in elevation. You aren't going to have a race that increases 3000 meters in elevation but is only 1 km long - that's just too steep to be at all safe (or enjoyable). The R^2 of a linear model predicting climb as a function of distance is 42.6%. Why is this important when working to predict time? Well if distance gets us 84.6% of the way to explaining time, and climb gets us 64.8% of the way to explaining time, clearly there must be some overlap in the value they offer since 0.846+0.648>1. The 42.6% is a measure of the overlap.

7.5.1 Added Variable Plot (aka Partial Regression Plot)

The ***Added Variable Plot***, also sometimes referred to as a ***Partial Regression Plot***, is a tool for capturing how much value an additional independent variable offers in predicting your dependent variable. The plot contains the residuals from two different linear models. First, the x-axis is the residuals from a model predicting x_2 using x_1. Then, the y-axis is the residuals of a model predicting y using only only x_1. To put this another way, the added variable plot shows the relationship between the variability of your possible new predictor term given the existing predictor compared to the variability that remains in your dependent variable after the first predictor is considered.



(a) Adding Climb to Distance Only

(b) Adding Distance to Climb Only

Figure 7.6: Added Variable Plots

Figure 7.6 shows two added variable plots. Figure 7.6 (a) shows the plot assessing the value climb offers in predicting time after distance, and Figure 7.6 (b) assesses the value of adding distance if climb was included first. Equations of the least-squares regression lines are given in the top left corners.

Note that the equations both have an intercept of zero. That's because, as you should recall, 1) fitted regression lines always pass through the mean of X and the mean of Y, and 2) our X and Y in these cases are both residuals and the mean residual from a linear regression model is always zero. The slopes from these fits are worth noting though.

When the multiple regression model is fit including distance and climb together, notice the slopes for the distance and climb terms exactly match the slopes from the added variable plots.

```
mhill<-lm(time~dist+climb, data=hills)
summary(mhill)$coef
```

```
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) -8.99203896 4.302734388 -2.089843 4.466516e-02
dist         6.21795571 0.601147884 10.343471 9.859214e-12
climb        0.01104791 0.002050892  5.386882 6.445183e-06
```

Now let's expand beyond our two-predictor hill race model and look at a larger data set with multiple predictors. Figure 7.7 shows a scatterplot matrix for the first 6 columns of the `mtcar` data frame we've worked with earlier.

Across the top row you can see that all of the available independent variables are strong potential predictors of fuel efficiency (MPG). Some terms may warrant some transformation

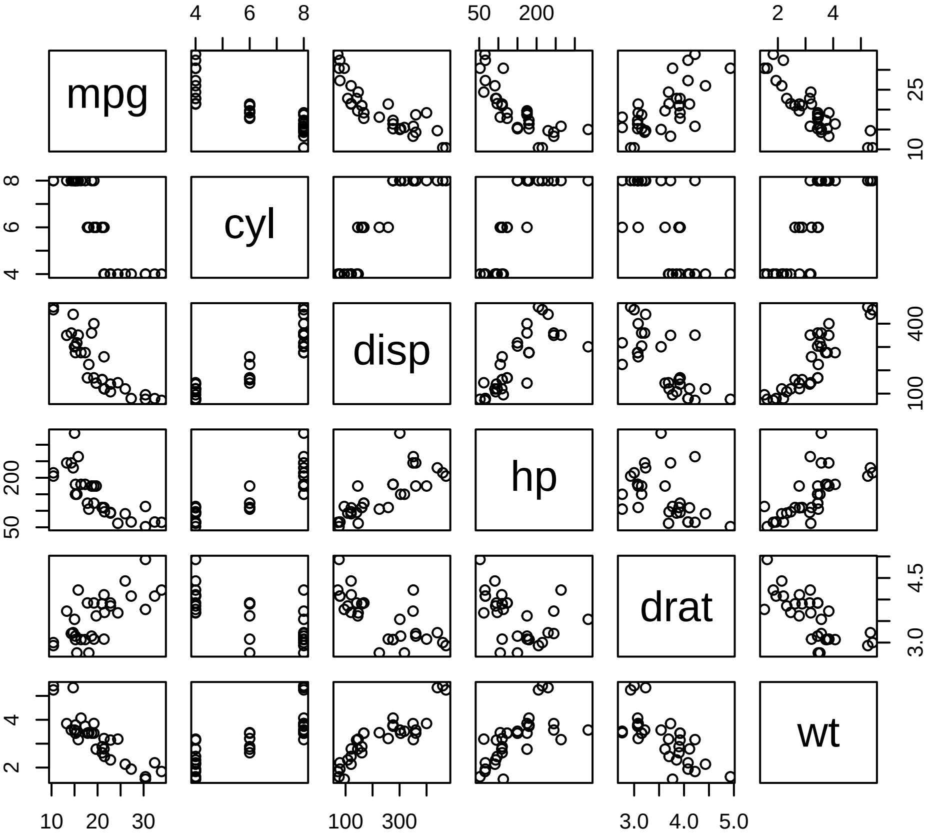


Figure 7.7: Motor Trend Car Data

given apparent curvature in the mpg-displacement and mpg-hp plots, but all indicate they could be useful in a model for MPG. Other scatterplots in the matrix show us that many of these terms are closely related to each other as well so we know there is overlap in the value they offer to predict MPG.

We start with the model $mpg = \beta_0 + \beta_1 wt + \beta_2 cyl + \varepsilon$ because vehicle weight and the number of engine cylinders should have the strongest influence on fuel efficiency. From there, the question is how much would our model improve if drat, the rear axle ratio, were added? From our initial scatterplot matrix drat's relationship with mpg appears reasonably linear so there shouldn't be any transformations to worry about.

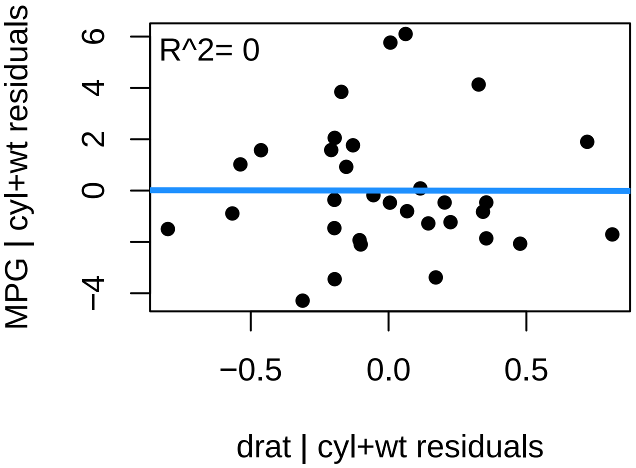


Figure 7.8: Added Variable Plot, Rear Axle Ratio

The added variable plot for drat given in Figure 7.8 shows that there is no real additional value that drat offers when cylinders and weight are already in the model. The relationship between the residuals from drat modeled by cylinders+weight and the residuals of mpg modeled by cylinders+weight appears completely random; has a best fit line that is virtually horizontal, and a correlation that is zero.

After seeing this plot, it should come as no surprise that when a model is fit predicting MPG as a function of cylinders+weight+axle ratio the axle ratio term is not statistically significant.

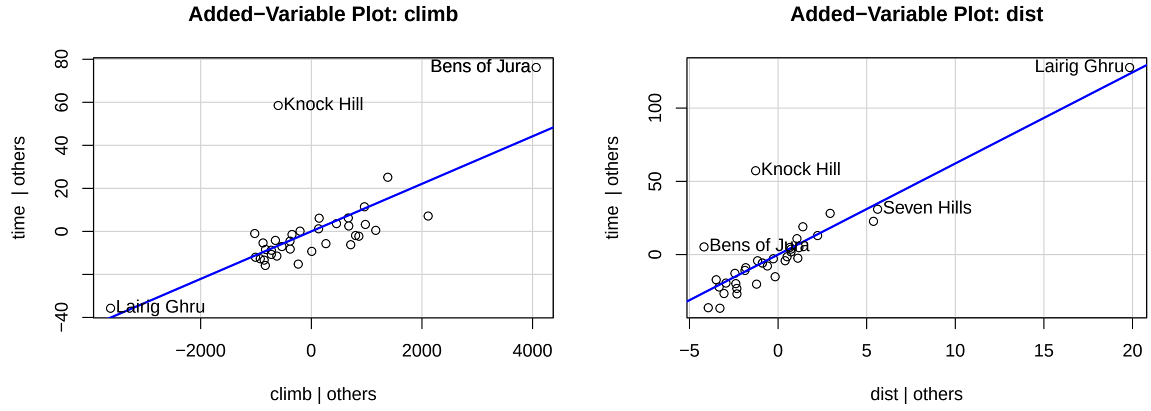
```r
m_cars<-lm(mpg~cyl+wt+drat, data=mtcars)
summary(m_cars)$coef
```

```
               Estimate Std. Error    t value     Pr(>|t|)
(Intercept) 39.76765828  6.8728698  5.7861795 3.258387e-06
cyl         -1.50957715  0.4464205 -3.3815136 2.142188e-03
wt          -3.19473392  0.8293065 -3.8522956 6.235168e-04
drat        -0.01620074  1.3230926 -0.0122446 9.903173e-01
```

In R

Figure 7.4 and Figure 7.8 show added variable plots put together manually using basic plotting functionality in R. There's also the `avPlot` function in the `car` library that will produce added variable plots directly with much less effort. All you need to specify in `avPlot` is the name of the full fitted model object, and the name of the variable you want to consider as the possible add/drop term. The variable name needs to be put in quotation marks. First, an example using the `avPlot` function on the hill race data:

```r
mhills<-lm(time~climb+dist, data=hills)
avPlot(mhills, "climb")
avPlot(mhills, "dist")
```



(a) Climb plot

(b) Distance plot

Figure 7.9: Added Variable Plots

And here is the `mtcar` model example with identification of unusual points removed:

```r
mcars<-lm(mpg~cyl+wt+drat, data=mtcars)
avPlot(mcars, "drat", id=FALSE)
```

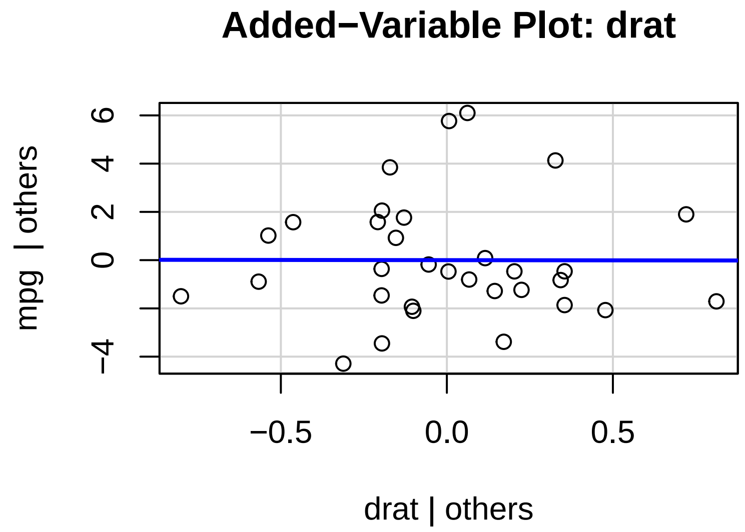
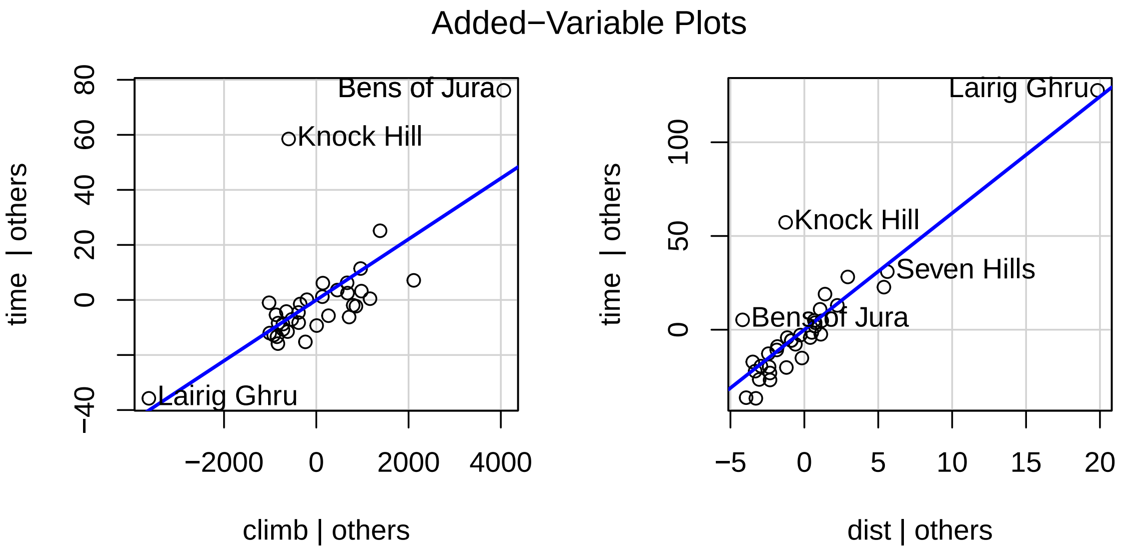


Figure 7.10: Added Variable Plot for Axle Ratio

If you want to quickly produce all added variable plots for a model, you can use the `avPlots` command. Just add that `s` to the command name and simply providing the model object will create all plots.

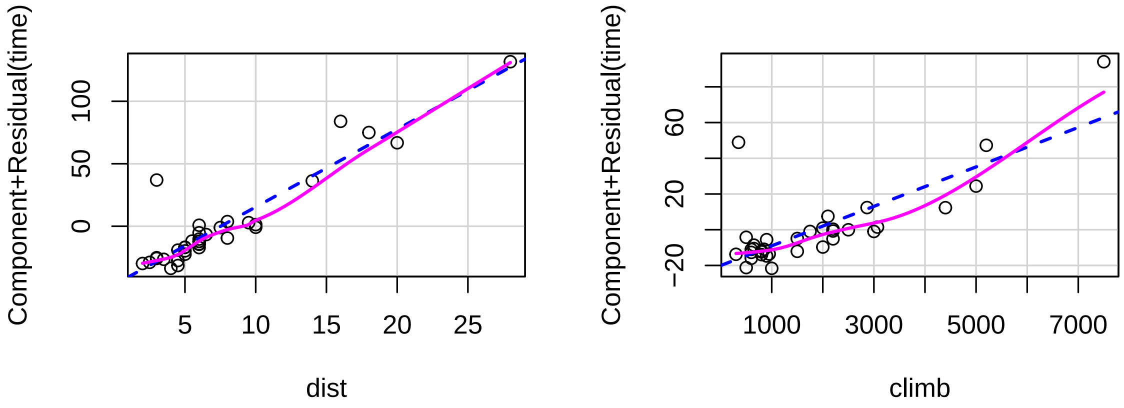
```
avPlots(mhills)
```



7.5.2 Partial Residual Plot

There's another useful plot to know about when working with multiple regression: the ***Partial Residual Plot***. A partial residual plot is another way to look at how each predictor plays a role in a model with multiple independent variables.

To investigate x_i using a partial residual plot, first fit a model with all independent terms of interest included. The x-axis is then x_i, and the y-axis is the residuals from your full model $+\beta_i x_i$. Figure 7.11 shows these plots for distance and elevation climb as predictors for race time in the hill race data. The partial residual plots produced by R by default have two helpful overlaid lines: a dashed blue line showing the best linear fit, and a more free-form line showing the pattern followed by the points.



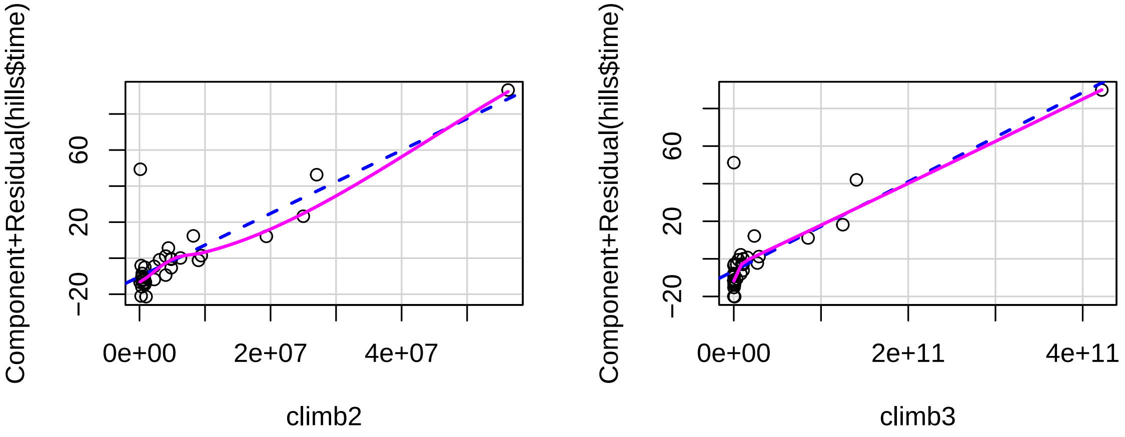
(a) Distance

(b) Elevation Climb

Figure 7.11: Partial residual plots, Hill Races

In this example, the pink line for distance follows the blue dashed line fairly well as shown in Figure 7.11 (a). The curvature shown in the pink line for the partial residual plot for climb in Figure 7.11 (b) however, indicates a transformation on climb might make for an even better model. Specifically, it appears that squaring climb might straighten the relationship out.

To investigate this, consider the plots in Figure 7.12 showing (a) the partial residual plot when $climb^2$ is used in the model and (b) the partial residual plot when $climb^3$ in the model. The $climb^2$ model is a definite improvement over the original, but still not quite right. The $climb^3$ model straightens the pink curve further to the point that it aligns nearly perfectly with the dashed blue line.

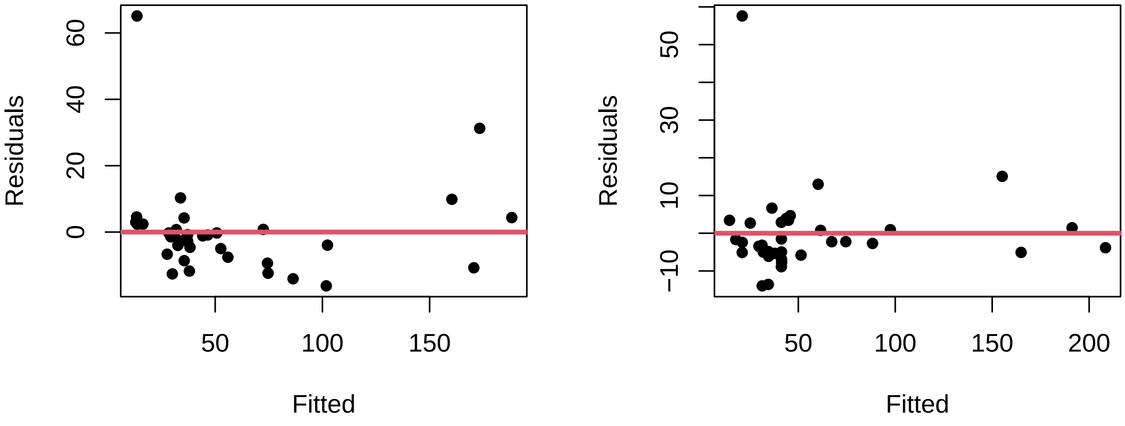


(a) Climb Squared

(b) Climb Cubed

Figure 7.12: Partial residual plots, Climb transformed

Using this, we can update our model for winning time to use *climb*^{3}, and then evaluate the fit and model assumptions. Figure 7.13 shows the fitted vs. residual plots for both the basic model with *climb* directly and the approach and the *climb*^{3} model.



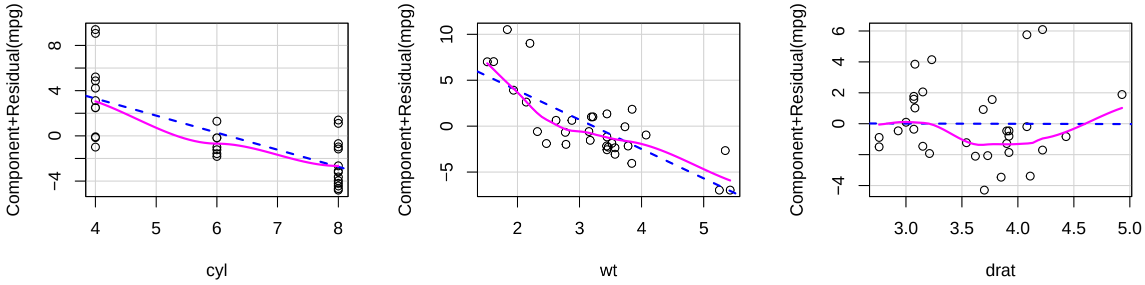
(a) Dist + Climb model

(b) Dist + Climb^3 model

Figure 7.13: Fitted vs. Residual Plots

This shows there is one extreme outlier with a residual greater than 50 that is present in both approaches. We can also see that the basic model does exhibit some curvature that appears fully corrected in the *climb*^{3} model. Homoskedasticity appears much improved in the new model as well.

How do partial residual plots look for our more complex `mtcars` example? Consider Figure 7.14. The plot for weight shows slight, but non-consistent deviation from the line. The plot for cylinders suggests perhaps a simple transformation might improve the fit. And the plot for drat, the measure of the rear axle ratio, definitely shows curvature away from a horizontal line. This horizontal line confirms what we saw with the added variable plot: inclusion of the term won't significantly help predict MPG at all.



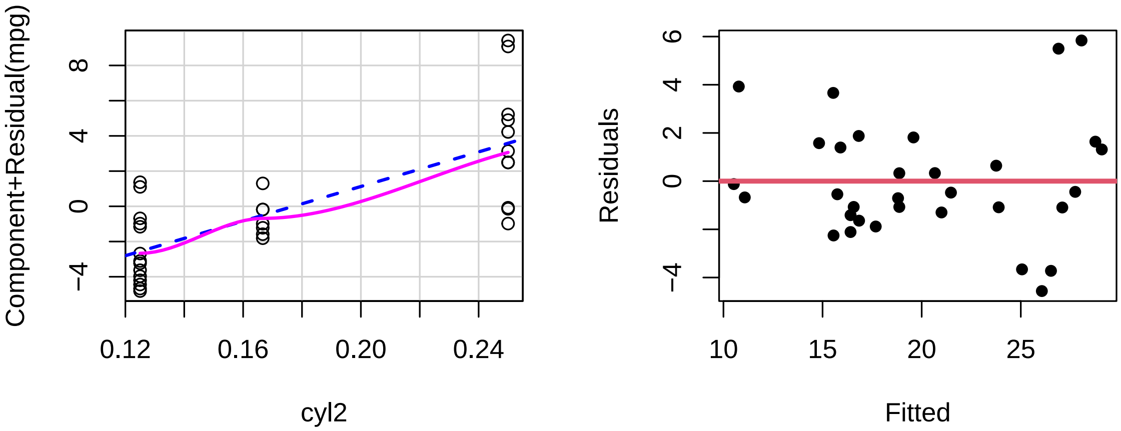
(a) Cylinders

(b) Weight

(c) Rear Axle Ratio

Figure 7.14: Partial Residual Plots

If cylinders is transformed to be included in the model as $1/cylinders$ then the updated partial residual plot becomes what is shown in Figure 7.15 (a) and the resulting fitted vs. residual plot for the weight+inverse cylinders model is as shown in Figure 7.15 (b). The R^2 of the model fit with cylinders inverted is very slightly increased over the initial model with simply weight+cylinders, 0.837 vs 0.830, making it a matter of scientist discretion if the added complexity of inverting cylinders is worth the minimal fit improvement.



(a) Inverse Cylinders Partial Residual Plot,Fitted vs. Residual

(b) Inverse Cylinders Partial Residual Plot,Fitted vs. Residual

Figure 7.15: Weight + 1/cylinders

```
mod_car<-lm(mpg~wt+cyl, data=mtcars)
mod_car2<-lm(mpg~wt+I(1/cyl), data=mtcars)

c(summary(mod_car)$r.square, summary(mod_car2)$r.square)
```

```
[1] 0.8302274 0.8372329
```

In R

Like the `avPlot` function, the partial residual plot function is in the `car` library. `crPlot` creates the partial residual plots with a minimum of two inputs: the full fitted model object, and the name of the variable you want to focus on. Make sure you put the variable name in quotation marks.

The code to produce Figure 7.11 was simply:

```
modh<-lm(time~dist+climb, data=hills)
crPlot(modh, "dist")
crPlot(modh, "climb")
```

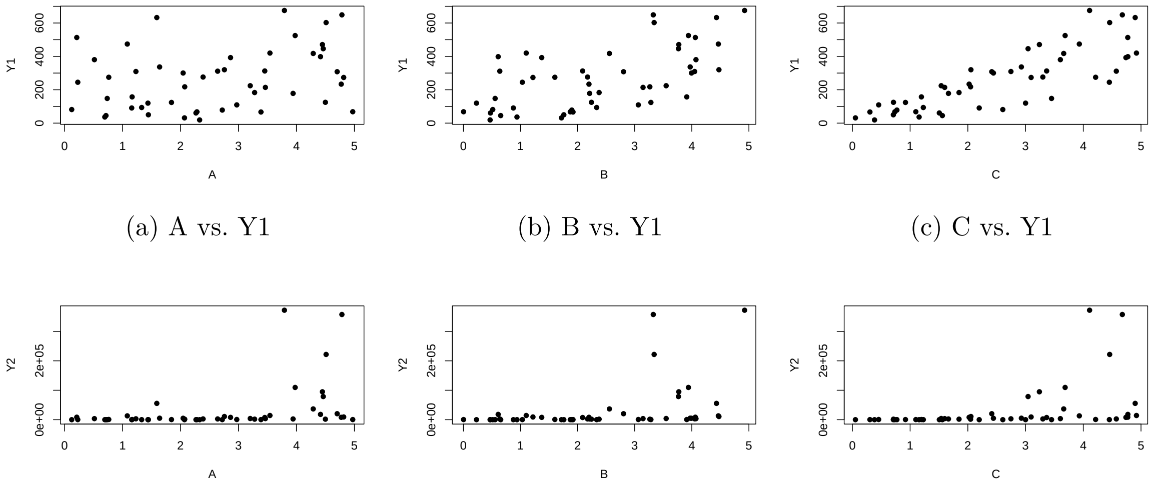
As seen with `avPlot`, there is also a plural version of the `crPlot` command. `crPlots` will create all possible partial residual plots for a specified model. Where does the name "crPlot" come from? The y-axis is the component of interest plus the model residual: "cr" is short for "component + residual".

7.5.3 Inverse Response Plot

When in a situation where transforming Y is your best course of action, an ***Inverse Response Plot*** can help. For this we'll create two dummy examples so the real relationship is fully known. Let's create 50 rows of three independent columns, "A", "B", and "C", where each is a created from a random sample from a Uniform[0,5]. Then we'll make Y1 and Y2 where $Y_1 = (A+2B+3C)^2+\epsilon$ and $Y_2 = e^{A+B+C}+\epsilon$ using a Normal(0, 1) for the ϵ noise component.

```
set.seed(123)
A<-runif(50, 0, 5)
B<-runif(50, 0, 5)
C<-runif(50, 0, 5)

Y1<-(A+2*B+3*C)^2 + rnorm(50)
Y2<-exp(A+B+C) + rnorm(50)
```

If we start by looking at simple plots of A, B, and C plotted against Y1, and against Y2, it is not immediately clear that transforming our Ys is what needs to happen (see Figure 7.16).



(a) A vs. Y1

(b) B vs. Y1

(c) C vs. Y1

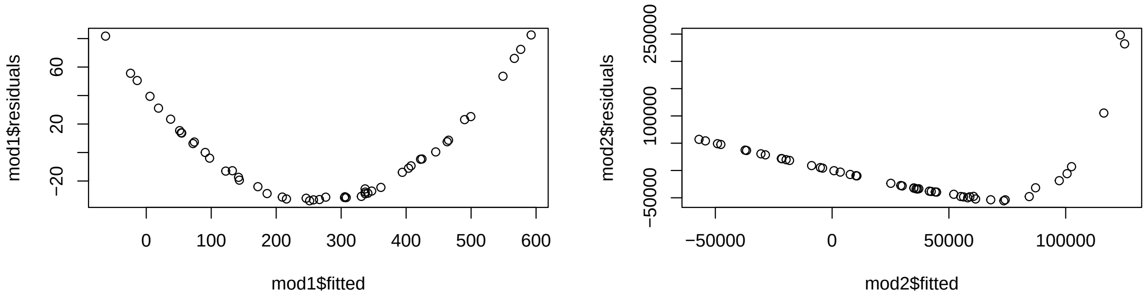
(d) A vs. Y2

(e) B vs. Y2

(f) C vs. Y2

Figure 7.16: Simple Scatter Plots

If we fit the basic models for Y1 and Y2 as functions of A, B, and C we get fitted vs. residual plots as shown in Figure 7.17 which clearly indicate there is a strong non-linear relationship at play and a transformation somewhere is warranted.

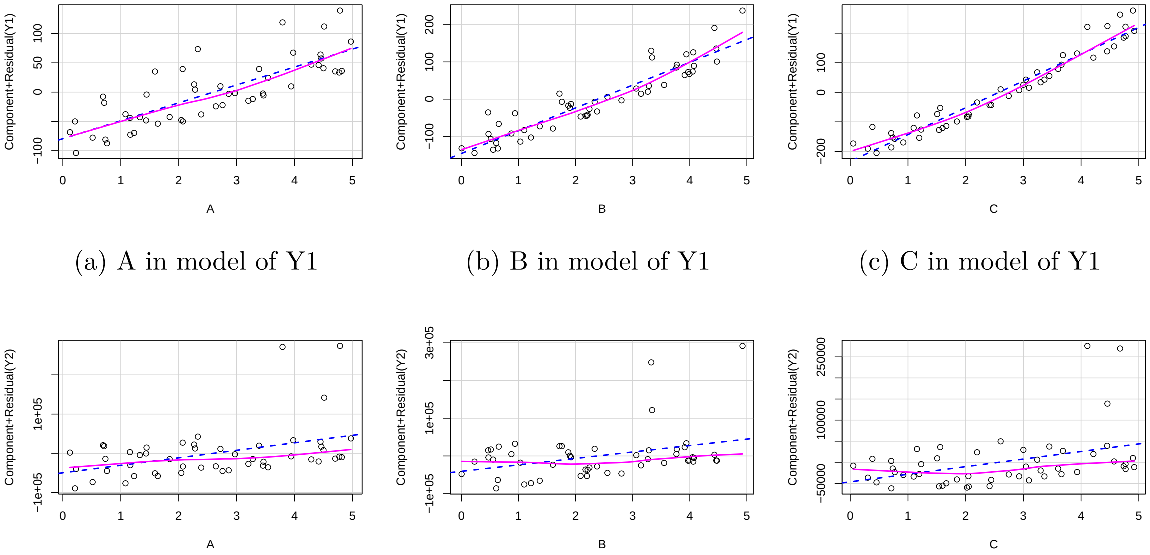


(a) Y1~A+B+C Fitted vs. Residual

(b) Y2~A+B+C Fitted vs. Residual

Figure 7.17: Fitted vs. Residuals

So let's look at the partial residual plots for A, B, and C to find out where to transform. These are shown in Figure 7.18 and none look to have significant curvature in their pink line fits.



(a) A in model of Y1

(b) B in model of Y1

(c) C in model of Y1

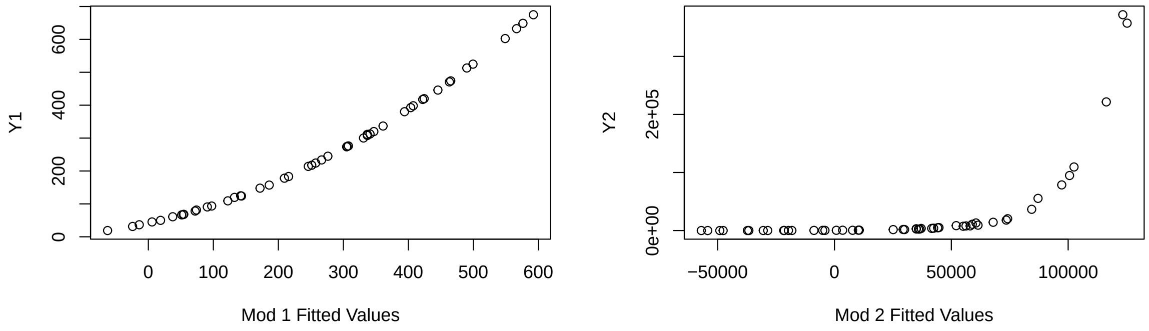
(d) A in model of Y2

(e) B in model of Y2

(f) C in model of Y2

Figure 7.18: Partial Residual Plots

If none of the X predictor terms are the source, consider the response Y as needing the transformation by looking at a plot of the fitted value of Y on the x-axis, vs the observed Y values on the y-axis. This is called the ***Inverse Response Plot*** and if Y needs a transformation, the shape of the curve in the inverse response plot will tell you which one you need.



(a) Inverse Response Y1

(b) Inverse Response Y2

Figure 7.19: Inverse Response Plots

Sure enough, the inverse response plot for Y1 is a curve that follows an x^2 curve shape which aligns with us needing to apply a square-root to Y1 to make it linear, and the inverse response plot for Y2 is a curve that follows an exponential curve shape indicating to us that a log transformation for Y2 is the way to go. Both just as our data was created.

7.6 Multicollinearity

When two or more independent variables are highly correlated with each other, your multiple regression model suffers from a problem known as multicollinearity. At issue is the fact that this correlation undermines the assumption that independent random variables are in fact *independent*, which is critical for accurately estimating their individual impacts on the dependent variable you're predicting.

Sometimes multicollinearity arises because of a choice the modeler has made that is easily corrected. For example, in a health study rather than including both the weight of a patient and the patient body mass index, you can just select one as a measure of patient size and move on. Often though the choice isn't quite so simple as the variables might very naturally move together while at the same time each providing valuable insight.

The problem is that having highly correlated independent variables makes the standard errors of the estimated regression coefficients increase. This means your confidence intervals are wider and you're less able to identify when a term is significantly different from zero or any other key value of interest. Multicollinearity can also lead to coefficients changing notably, even changing signs unexpectedly, with relatively minor changes to the data or the model.

For an example, let's look at the `fgl` data frame from the `MASS` library. This data frame considers the refractive index of 214 fragments of six different types of glass and their oxide makeup. We'll start with the full model that includes all available predictors of refractive index:

```r
mod_glass<-lm(RI~., data=fgl)
summary(mod_glass)
```

```
Call:
lm(formula = RI ~ ., data = fgl)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.2131 -0.3786 -0.0186  0.3697  4.0317 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 11.68095   67.73022   0.172 0.863248    
Na           0.60928    0.66840   0.912 0.363100    
Mg           1.32017    0.67269   1.963 0.051086 .  
Al          -0.92790    0.70416  -1.318 0.189098    
Si          -0.61637    0.68460  -0.900 0.369022    
K            0.74007    0.68789   1.076 0.283286    
Ca           2.47150    0.66700   3.705 0.000273 ***
Ba           1.97569    0.70198   2.814 0.005374 ** 
Fe           0.36329    0.73879   0.492 0.623442    
typeWinNF    0.09996    0.17352   0.576 0.565209    
typeVeh     -0.88579    0.26111  -3.392 0.000835 ***
typeCon      0.39910    0.39687   1.006 0.315810    
typeTabl     0.39263    0.42512   0.924 0.356822    
typeHead     1.58234    0.43890   3.605 0.000394 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9503 on 200 degrees of freedom
Multiple R-squared:  0.9081,	Adjusted R-squared:  0.9021 
F-statistic:   152 on 13 and 200 DF,  p-value: < 2.2e-16
```

This model summary tells us a few things. First, you'll notice that many of our independent variables do not appear significant - many p-values are well above any reasonable α level. You can also see that two glass types that are most different from each other are vehicle window glass, with a -0.88579 slope and vehicle headlight glass with a 1.58 slope. This kind of makes sense that they would differ from the other types because the thick safety glass of vehicle windows is very different from the type of glass used to make regular home windows or tableware. Similarly, glass for headlights is also very different from your drinking glass but in an entirely different way. These observations aside though, the $R^2 = 0.9081$ does indicate the model does a fairly good job at predicting the refractive index of glass given the glass components.

Now consider a reduced model that reduces glass type down to only three levels: vehicle, headlight, or other, and only considers the Al, Si, Ca, and Ba elements.

```r
vehicle<-as.factor(fgl$type=="Veh")
headlight<-as.factor(fgl$type=="Head")
mod_glass2<-lm(RI~Al+Si+Ca+Ba+vehicle+headlight, data=fgl)
summary(mod_glass2)
```

```

Call:
lm(formula = RI ~ Al + Si + Ca + Ba + vehicle + headlight, data = fgl)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.3816 -0.4509  0.0376  0.4418  3.9111 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   108.21998    7.34898  14.726  < 2e-16 ***
Al             -2.19413    0.17487 -12.547  < 2e-16 ***
Si             -1.61520    0.09887 -16.336  < 2e-16 ***
Ca              1.39476    0.05230  26.671  < 2e-16 ***
Ba              0.89572    0.20551   4.359 2.06e-05 ***
vehicleTRUE    -0.93091    0.26208  -3.552 0.000473 ***
headlightTRUE   0.63323    0.31455   2.013 0.045397 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.015 on 207 degrees of freedom
Multiple R-squared:  0.8914,    Adjusted R-squared:  0.8883 
F-statistic: 283.3 on 6 and 207 DF,  p-value: < 2.2e-16
```

Our intercept is wildly different now. What do you notice about the coefficients for Al, Si, Ca and Ba for the second model compared to the first? The Al and Si slopes are more than twice what they were in the earlier model while the Ca and Ba ones decreased significantly. And look at the standard errors on those new coefficient estimates. Our large model had standard errors that were all in the .66 to .74 range, but now they are much much smaller. Si went from having a standard error of 0.6846 to one of only .09887. The model coefficient of determination, R^2, is only slightly decreased and still strong at 0.8941. What is going on here? Multicollinearity. Here is the correlation matrix (rounded to three decimal places) on our continuous independent variables:

```
round(cor(fgl[,2:9]),3)
```

```
       Na     Mg     Al     Si      K     Ca     Ba     Fe
Na  1.000 -0.274  0.157 -0.070 -0.266 -0.275  0.327 -0.241
Mg -0.274  1.000 -0.482 -0.166  0.005 -0.444 -0.492  0.083
Al  0.157 -0.482  1.000 -0.006  0.326 -0.260  0.479 -0.074
Si -0.070 -0.166 -0.006  1.000 -0.193 -0.209 -0.102 -0.094
K  -0.266  0.005  0.326 -0.193  1.000 -0.318 -0.043 -0.008
Ca -0.275 -0.444 -0.260 -0.209 -0.318  1.000 -0.113  0.125
Ba  0.327 -0.492  0.479 -0.102 -0.043 -0.113  1.000 -0.059
Fe -0.241  0.083 -0.074 -0.094 -0.008  0.125 -0.059  1.000
```

Looks like Mg is fairly strongly correlated with a few of the other terms. That's why the model wasn't hurt much by it's removal.

In addition to checking correlation matrices, the *variance inflation factor* is a good metric for flagging where multicollinearity is likely an issue. The variance inflation factor is defined as:

$$VIF_i = \frac{1}{1 - R_i^2}$$

Where R_i^2 is the R^2 from a regression model predicting variable i using all other predictors. So to examine the influence of Mg in our model we would look at how well Mg could be modeled linearly as a function of NA, Al, Si, K, Ca, Ba, Fe, and our two glass types:

```
mod_Mg<-lm(Mg~Na+Al+Si+K+Ca+Ba+Fe+vehicle+headlight, data=fgl)
summary(mod_Mg)$r.square
```

```
[1] 0.995202
```

Seems Mg can be almost entirely explained by the other terms. With a high R^2_{Mg}, that means VIF_{Mg} will be large. In fact, $\frac{1}{1-.9952} = 208.42$. Values of $VIF > 5$ are generally considered to be worth a closer look.

In R

Fortunately, R will pretty easily provide you with all VIF metrics when provided with a full model under consideration. In the `car` library lives the `vif` function. Running all VIFs for our glass model is as simple as:

```r
mod_glass<-lm(RI~Na+Mg+Al+Si+K+Ca+Ba+Fe+vehicle+headlight, data=fgl)
vif(mod_glass)
```

```
        Na         Mg         Al         Si          K         Ca         Ba 
 65.145740 208.421848  27.958365  62.254625  44.602864 200.586359  27.131346 
        Fe    vehicle  headlight 
  1.218031   1.053694   3.388083 
```

Taking away Mg as the term with the largest VIF, our new model with VIFs looks like:

```r
mod_glass2<-lm(RI~Na+Al+Si+K+Ca+Ba+Fe+vehicle+headlight, data=fgl)
vif(mod_glass2)
```

```
       Na        Al        Si         K        Ca        Ba        Fe   vehicle 
 2.038928  1.882341  1.551700  1.776061  1.648278  2.245932  1.092993  1.047940 
headlight 
 3.277242 
```

Showing with that one omission, the bulk of our multicollinearity issue has been cleared up.

8 Building and Selection

8.1 Measures of model strength

8.1.1 Test Set and Training Set

It may seem counter-intuitive but sometimes the best choice you can make is to NOT use all of your data when fitting your model. If you begin by splitting your data into two, you can fit the model on one set, known as the "training set", and then test how well the model performs on the remaining "test set". This helps protect you from creating a model that is so well tuned to your specific data that it isn't generalizable to the full population and won't produce helpful predictions.

Discretion is allowed when deciding the relative sizes of the training and testing sets. At most you would split the data 50/50 putting half in each set randomly. In some cases with limited data you might shift that to putting as much as 80% in the training set and holding out the remaining 20% for a sanity check.

In R

Below is an example using the reduced glass refraction index data just seen at the end of Section 7.6. The fgl glass data frame has 214 rows to start with.

```
library(MASS)
library(caret)
```

```
# set seed for reproducability
set.seed(1234)

# randomly select 150 rows for the training set
train_index<-sample(1:214, 150)

# put row numbers not selected into the test set
test_index<-(1:214)[!is.element(1:214, train_index)]
```

```r
# create the two data sets
glass_train<-fgl[train_index,]
glass_test<-fgl[test_index,]

# fit the model with the training set
glass_model<-lm(RI~Al+Si+Ca+Ba, data=glass_train)

# find mean squared error in training set
mean(glass_model$residuals^2)
```

```
[1] 1.160116
```

```r
# find mean squared error in test set
fit_values<-predict(glass_model, glass_test)
test_errors<-fit_values-glass_test$RI
mean(test_errors^2)
```

```
[1] 0.9209567
```

```r
# if you want a formal test, can do t-test of errors
t.test(glass_model$residuals^2, test_errors^2)
```

```

    Welch Two Sample t-test

data:  glass_model$residuals^2 and test_errors^2
t = 0.58113, df = 152.37, p-value = 0.562
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5738999  1.0522177
sample estimates:
mean of x mean of y
1.1601156 0.9209567
```

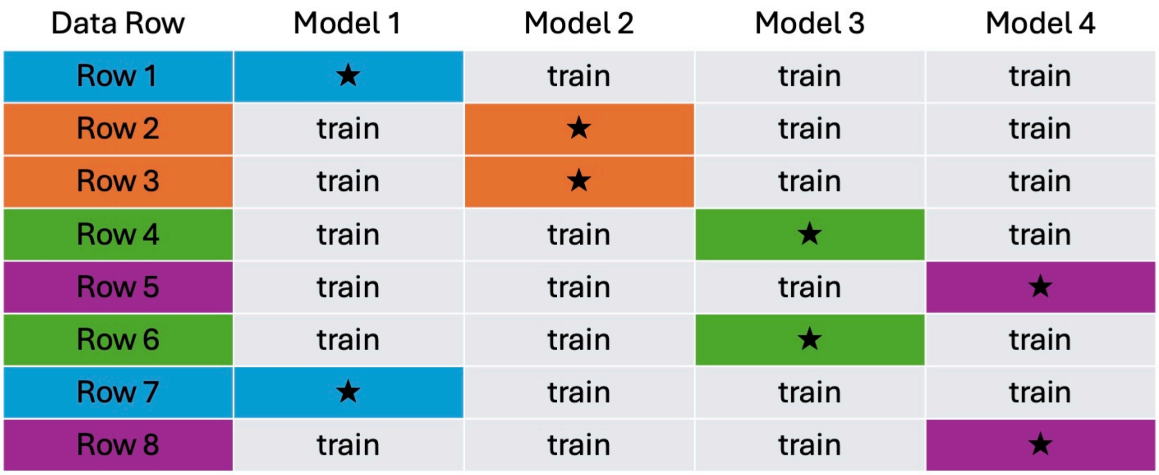
This shows that there is no evidence our model fitted using the training data doesn't perform just as well on our test data. With this we can be confident that future refraction index values predicted using the model are reasonably accurate.

8.1.2 K-fold cross-validation

What if we just got lucky with our test set and training set sampling? K-fold cross-validation takes the idea of test/training set validation to the next level. Instead of separating the data into just two parts, a test set and a training set, k-fold cross validation splits the data into k parts. Often k is set to 5 or 10, but there's no reason you can't select any other number provided you have enough rows to reasonably split.

Once the data is split, the model is then fit k times; each time using k-1 groups as the training set and the held out group as the test set. Figure 8.1 below illustrates what would happen with 4-fold cross validation on a data set with only 8 rows.

First the 8 rows are randomly grouped into four groups of 2. The model is then run 4 times. In the example, rows 1 and 7 are in fold 1. The first time the model is fit, fold 1 points are set aside for testing and the model is trained on rows 2,3,4,5,6, and 8. For model 2, our second fold containing rows 2 and 3 are set aside for testing, and the model is trained using rows 1,4,5,6,7, and 8. In the end you're left with four examples of how well the model fit using the approach you've selected fits data that was not involved in the fitting.



Data Row	Model 1	Model 2	Model 3	Model 4
Row 1	★	train	train	train
Row 2	train	★	train	train
Row 3	train	★	train	train
Row 4	train	train	★	train
Row 5	train	train	train	★
Row 6	train	train	★	train
Row 7	★	train	train	train
Row 8	train	train	train	★

Figure 8.1: Four-fold cross validation diagram

In R

Continuing with the glass refractive index scenario, we'll complete 10-fold cross validation of a model containing Al, Si, Ca, and Ba as linear predictors using the `train` function in the `caret` library.

```
# set seed for reproducability
set.seed(1234)

# Define the parameters. Method is cross validation, k=10.
train_params <- trainControl(method = "cv", number = 10)

# Train your k models.  Specify lm is type of model,
#     and put in the training parameters set above
glass_models <- train(RI ~ Al+Si+Ca+Ba, data=fgl,
  method = "lm",trControl = train_params)

# View the results of the cross-validation
glass_models$resample
```

```
        RMSE  Rsquared       MAE Resample
1  0.4672234 0.9570430 0.3934590   Fold01
2  0.6537417 0.9465408 0.5239035   Fold02
3  1.0238911 0.8434257 0.6182354   Fold03
4  1.4876875 0.7103225 1.0625717   Fold04
5  1.1690601 0.8168152 0.9380207   Fold05
6  1.0556602 0.8918505 0.8826193   Fold06
7  0.5904965 0.9885397 0.4773744   Fold07
8  1.2919068 0.7462664 0.7501642   Fold08
9  1.3814389 0.7731487 0.8307277   Fold09
10 1.0430018 0.8797080 0.7720872   Fold10
```

These results indicate that over the 10 folds, the worst RMSE (square-root of mean squared error) among test holdouts happened when Fold4 was withheld from training. In this worst-case, the RMSE was 1.487. In the best-case, when Fold 1 was withheld, the RMSE was only 0.467. These are RMSEs calculated on the fold that was used as testing, not the folds that were used in training. If you prefer, the R^2 values for the testing holdouts, and the MAE (mean absolute error) for the testing holdouts are also given for each of the k folds. Some variability over the folds is to be expected, and these ten look reasonably consistent indicating the `RI~Al+Si+Ba+Ca` model approach is not over-fitting and should apply generally to the population.

8.1.3 Adjusted R^2

Back in Chapter 2 we defined R^2, the coefficient of determination, as:

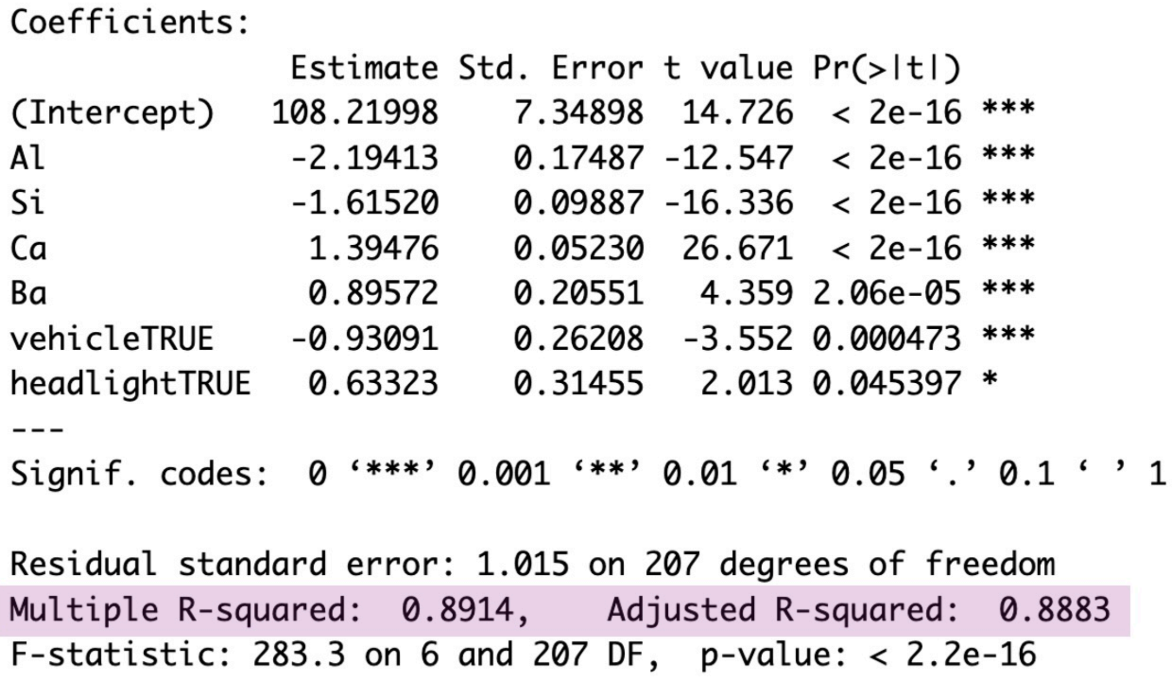
$$R^2 = 1 - \frac{SSError}{SSTotal} = \frac{SSReg}{SSTotal}$$

So whenever the sum of squared residuals, $SSError$, decreases, the R^2 goes up. In the world of multiple regression this creates a bit of a problem because every time you add a new term to the regression model the sum of squared residuals will go down whether the new term is really helpful or not. That's why judging two models against each other using R^2 just won't work. Enter adjusted R^2.

$$R^2_{adj} = 1 - \frac{SSError/(n-p-1)}{SSTotal/(n-1)}$$

Where n is the number of observations and p is the number of predictor variables. This makes it so just adding terms won't necessarily increase your R^2 - it will only improve if the increase in p is adding real value to your model.

Getting this adjusted R^2 from R is simple. The basic model summary output includes this adjusted R^2 right after the R^2 value you've already come to know and love. Below is a highlighted example from the reduced glass refraction index model. The basic R^2 with no adjustment is given as 0.8914, and the R^2_{adj} is given as 0.8883. Since the two are close in value, that is an indication that every term in the linear model is providing additional insight into the value of refraction index.



```
Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    108.21998    7.34898  14.726  < 2e-16 ***
Al              -2.19413    0.17487 -12.547  < 2e-16 ***
Si              -1.61520    0.09887 -16.336  < 2e-16 ***
Ca               1.39476    0.05230  26.671  < 2e-16 ***
Ba               0.89572    0.20551   4.359 2.06e-05 ***
vehicleTRUE     -0.93091    0.26208  -3.552 0.000473 ***
headlightTRUE    0.63323    0.31455   2.013 0.045397 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.015 on 207 degrees of freedom
Multiple R-squared:  0.8914,    Adjusted R-squared:  0.8883 
F-statistic: 283.3 on 6 and 207 DF,  p-value: < 2.2e-16
```

8.1.4 AIC and BIC

AIC (Akaike's Information Criterion) and BIC (Bayes Information Criterion) are two metrics used to compare the relative strength of competing models. The specific values of AIC and BIC have no real meaning, but the relative size of AIC and BIC of two or more possible models can help you decide which model to use.

Calculation of both AIC and BIC includes the log of the likelihood of the model under review. What is a model's likelihood? Simply put: it's the probability of seeing the observed data if the fitted model is the truth. The process to figuring out this likelihood is beyond our scope so we'll rely on R to do the work for us, but it's good to have a sense of what it's measuring.

$$AIC = 2(p+2) - 2ln(\hat{L})$$

and

$$BIC = ln(n)(p+2) - 2ln(\hat{L})$$

where $\hat{L}$ is the likelihood, p is the number of independent predictors in your model, and n is the number of observations. Notice the two are quite similar; the only difference is AIC has the number of estimated parameters multiplied by 2, while BIC penalizes larger parameter counts by a factor of $ln(n)$.

For both AIC and BIC, the model with the lower value is the better choice. The `stats` library in R (a library usually installed by default) contains both an `AIC` and a `BIC` function for calculating these measures.

Here is an example considering two different models for glass refraction index; one with glass type indicators and one without:

```r
mod_glass2<-lm(RI~Na+Al+Si+K+Ca+Ba+Fe+vehicle+headlight, data=fgl)

mod_glass3<-lm(RI~Na+Al+Si+K+Ca+Ba+Fe, data=fgl)

AIC(mod_glass2)
```

```
[1] 598.3572
```

```r
AIC(mod_glass3)
```

```
[1] 625.5007
```

```r
BIC(mod_glass2)
```

```
[1] 635.383
```

```r
BIC(mod_glass3)
```

```
[1] 655.7945
```

In this case AIC and BIC agree that the model including glass type is better than the model without type. It is not always the case that AIC and BIC will agree though. That's why for scientific rigor you should select which measure you want to use prior to calculating them.

8.2 Model Building

With R^2_{adj}, AIC, and BIC we now have three different approaches to measure the quality of a model. This opens the door to ways we can build a multiple regression model piece by piece.

8.2.1 Forward Selection

The forward selection process is exactly what is sounds like: adding one variable at a time. At each step of this approach, each potential predictor is considered and one that improves R_{adj} the most, or decreases AIC or BIC the most, is added to the model. The process continues until adding one more term does not improve your performance metric.

8.2.2 Backward Elimination

This approach to model building is also exactly what it sounds like. With backward selection you begin with a large model including all possible predictors and then remove them one by one. Which one should be removed at each step can be decided based on which has the highest p-value for the test on $\hat{\beta}$, by which one, when removed, increases R_{adj} the most, or by which one, when removed, decreases AIC or BIC the most. The process stops when removing any additional terms will only hurt the model quality.

8.2.3 Step-wise

The step-wise approach is a combination of forward selection and backward elimination. In this algorithm, the model can either grow or shrink but only ever one term at a time. The modeler uses discretion to add or subtract terms at each iteration, possibly alternating between adding and subtracting, but other paterns are possible as well. The process stops when it's found that neither adding nor subtracting a term will improve the decision metric (R^2_{adj}, AIC or BIC).

8.3 Building Examples in R

8.3.1 Forward

In the openintro library you will find a data frame called gifted that contains information on 36 gifted children. Along with their scores for a test of analytical skills, the data frame contains each child's mother's and father's IQs, the age at which the child first said "mommy" or "daddy", the age when the child first counted to 10 successfully, and the average number of hours per week the child reads with a parent, watches education TV, and watches cartoons.

To build a model working with forward selection and the AIC metric, we can use the `step` function built into the `stats` library as follows:

```
# specify an empty intercept-only model to start
mod_start<-lm(score~1, data=gifted)

# define the largest model possible as an upper bound on scope
mod_full<-lm(score~., data=gifted)

# specifying k=2 indicates using AIC.  Each step will be printed out.
step(mod_start, scope=list(lower=mod_start, upper=mod_full),
     direction="forward", k=2)
```

```
Start:  AIC=111.33
score ~ 1

           Df Sum of Sq    RSS     AIC
+ motheriq  1   244.838 505.47  99.111
+ count     1   222.211 528.09 100.687
+ read      1   206.958 543.35 101.712
+ edutv     1   102.861 647.44 108.023
+ speak     1    53.846 696.46 110.650
+ cartoons  1    45.074 705.23 111.100
<none>                  750.31 111.331
+ fatheriq  1    26.542 723.76 112.034

Step:  AIC=99.11
score ~ motheriq

           Df Sum of Sq    RSS     AIC
+ read      1   227.205 278.26  79.622
+ count     1   211.162 294.31  81.640
+ speak     1    38.747 466.72  98.240
+ fatheriq  1    30.712 474.76  98.854
+ edutv     1    27.814 477.65  99.073
<none>                  505.47  99.111
+ cartoons  1     2.272 503.20 100.949

Step:  AIC=79.62
score ~ motheriq + read

           Df Sum of Sq    RSS    AIC
+ fatheriq  1    43.605 234.66 75.486
<none>                  278.26 79.622
+ speak     1    11.834 266.43 80.057
+ edutv     1     5.904 272.36 80.850
+ count     1     3.617 274.65 81.151
+ cartoons  1     0.563 277.70 81.549

Step:  AIC=75.49
score ~ motheriq + read + fatheriq

           Df Sum of Sq    RSS    AIC
<none>                  234.66 75.486
+ speak     1   12.5784 222.08 75.503
+ edutv     1    9.7910 224.87 75.952
+ count     1    4.3398 230.32 76.814
+ cartoons  1    0.8398 233.82 77.357


Call:
lm(formula = score ~ motheriq + read + fatheriq, data = gifted)

Coefficients:
```

```
(Intercept)     motheriq        read     fatheriq
    44.3656       0.4282     12.7663       0.3215
```

Now let's examine each step of the output above. We specified in the `step` command that we wanted to begin with an empty intercept-only model and that the biggest model that could be considered is the full model with all predictors available in the `gifted` data frame.

At the start, our intercept-only model has an AIC of 111.33, and the first table in the output tells us the RSS (Residual Sum of Squares or SSError) and AIC for models that include the intercept plus each possible term one by one; each term gets a new row in the table. In the first round, R found that adding `motheriq` to the intercept produced an AIC of 99.111, adding `count` led to an AIC of 100.687, adding `read` led to an AIC of 101.712… and so on through to adding `fatheriq` making for an AIC of 112.034, an actual increase over the baseline start of 111.33. Since lower AIC is better, `motheriq` gets added to the model, and the second round begins.

Step two begins with the `score~motheriq` model. An intercept is included but not explicitly stated here. The table shows the results from adding each possible remaining predictor to the score to the model and finds that adding in `read` is the most helpful and will decrease the model AIC to 79.622.

Two more rounds of forward selection continue. The third round examines adding another term to the `score~motheriq+read` model, and selects adding in `fatheriq`. Round four looks to add to the `score~motheriq+read+fatheriq` model, but discovers adding any of the remaining possible terms will result in a higher AIC so the search ends and the `score~motheriq+read+fatheriq` is returned as the final model.

8.3.2 Backward

What if the backward elimination approach is taken? Same `gifted` data as used in the forward selection example, but now let's start with the full model and remove terms one at a time until all $\hat{\beta}$ p-values are below $\alpha = 0.05$.

We'll start by checking the VIFs to determine if high multicollinearity should be taken care of first.

```
vif(mod_full)
```

```
fatheriq motheriq    speak    count     read    edutv cartoons
1.196214 1.172998 1.182755 6.886769 6.896808 8.214684 8.373075
```

```
summary(mod_full)$coef
```

```
               Estimate  Std. Error    t value     Pr(>|t|)
(Intercept) 75.5084856 24.02617507  3.1427593 3.934153e-03
fatheriq     0.2524925  0.13756072  1.8354988 7.707469e-02
motheriq     0.4000681  0.07290507  5.4875205 7.328886e-06
speak        0.1876429  0.14766697  1.2707168 2.142862e-01
count        0.2064897  0.26631214  0.7753673 4.446222e-01
read         7.5440549  5.58640059  1.3504321 1.876921e-01
edutv       -4.2024429  2.24503495 -1.8718831 7.170324e-02
cartoons    -3.3389901  2.01808208 -1.6545363 1.091872e-01
```

Since time spent watching educational tv and time spent watching cartoons are strongly correlated with each other (correlation of -0.9234), it is not surprising that the two have high VIFs. For step one, we'll remove `cartoons` because of the two, it is the one with the higher $\widehat{\beta}$ p-value.

```
mod_2<-lm(score~fatheriq+motheriq+speak+count+read+edutv, data=gifted)
vif(mod_2)
```

```
fatheriq motheriq    speak    count     read    edutv
1.016676 1.154849 1.158823 6.650745 6.759192 1.217195
```

```
summary(mod_2)$coef
```

```
               Estimate  Std. Error    t value     Pr(>|t|)
(Intercept) 49.8812173 18.90916625  2.6379385 0.0132717059
fatheriq     0.3406675  0.13056188  2.6092414 0.0142021633
motheriq     0.3850639  0.07447438  5.1704205 0.0000157951
speak        0.2223968  0.15048032  1.4779129 0.1502106664
count        0.2880610  0.26943473  1.0691311 0.2938293264
read         6.2384283  5.69364778  1.0956822 0.2822341840
edutv       -0.7741723  0.88969865 -0.8701512 0.3913636419
```

In our model without `cartoons`, `read` and `count` still have VIFs greater than 5. They are strongly positively correlated (0.9103) so even though `edutv` has the highest p-value, I'll drop `count` this round since it has the higher p-value between `count` and `read`.

```
mod_3<-lm(score~fatheriq+motheriq+speak+read+edutv, data=gifted)
vif(mod_3)
```

```
fatheriq motheriq    speak     read    edutv 
1.016335 1.136772 1.054395 1.072252 1.190811 
```

```
summary(mod_3)$coef
```

```
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 47.0554351 18.76811965  2.507200 1.781474e-02
fatheriq     0.3381140  0.13085088  2.583965 1.488079e-02
motheriq     0.3950256  0.07406516  5.333487 9.083768e-06
speak        0.1741009  0.14388176  1.210028 2.357167e-01
read        11.8220180  2.27313104  5.200764 1.321418e-05
edutv       -0.9142163  0.88209867 -1.036411 3.082929e-01
```

High multicollinearity is all cleared up now so the focus can be solely on p-values. Now we'll drop `edutv` for having the highest p-value.

```
mod_4<-lm(score~fatheriq+motheriq+speak+read, data=gifted)
summary(mod_4)$coef
```

```
              Estimate  Std. Error  t value     Pr(>|t|)
(Intercept) 42.7092528 18.31549501 2.331865 2.637951e-02
fatheriq     0.3242503  0.13032091 2.488091 1.842563e-02
motheriq     0.4207323  0.06987185 6.021485 1.154340e-06
speak        0.1898180  0.14325136 1.325070 1.948325e-01
read        12.2089372  2.24494494 5.438413 6.108775e-06
```

Next to drop is speak.

```
mod_5<-lm(score~fatheriq+motheriq+read, data=gifted)
summary(mod_5)$coef
```

```
              Estimate  Std. Error  t value     Pr(>|t|)
(Intercept) 44.3655659 18.48732491 2.399783 2.239759e-02
fatheriq     0.3214775  0.13183396 2.438503 2.047870e-02
motheriq     0.4282474  0.07045893 6.077973 8.662970e-07
read        12.7663191  2.23107427 5.722050 2.430603e-06
```

And now we'll stop because all remaining terms have a p-value smaller than our $\alpha = 0.05$ threshold. The final model of `score~fatheriq+motheriq+read` matches the one obtained using AIC and forward selection, but that need not be the case. In fact, let's look at what happens if we were to perform our backward elimination without multicollinearity checks along the way.

Our first step would have us remove `count` because it's p-value of 0.4446 is the highest in `mod_full`. So step two would now be:

```
mod_2<-lm(score~fatheriq+motheriq+speak+read+edutv+cartoons, data=gifted)
summary(mod_2)$coef
```

```
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 75.7756404 23.85794658  3.176117 3.526914e-03
fatheriq     0.2430752  0.13607806  1.786292 8.451231e-02
motheriq     0.4082659  0.07163665  5.699121 3.646928e-06
speak        0.1511945  0.13901910  1.087581 2.857370e-01
read        11.5226177  2.19353467  5.252991 1.255677e-05
edutv       -4.5968140  2.17157275 -2.116813 4.297596e-02
cartoons    -3.6286687  1.96951508 -1.842417 7.566028e-02
```

Then `speak` is removed...

```
mod_3<-lm(score~fatheriq+motheriq+read+edutv+cartoons, data=gifted)
summary(mod_3)$coef
```

```
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 79.0467761 23.73960954  3.329742 2.312701e-03
fatheriq     0.2373610  0.13639031  1.740307 9.205533e-02
motheriq     0.4121118  0.07176701  5.742357 2.870557e-06
read        11.9013189  2.17231039  5.478646 6.030978e-06
edutv       -4.8881415  2.16154516 -2.261411 3.114315e-02
cartoons    -3.8202332  1.96759143 -1.941578 6.163171e-02
```

Then `fatheriq` is removed...

```
mod_4<-lm(score~motheriq+read+edutv+cartoons, data=gifted)
summary(mod_4)$coef
```

```
               Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 112.5034019 14.37637560  7.825575 7.855003e-09
motheriq      0.4177214  0.07400329  5.644633 3.383735e-06
read         11.6036137  2.23529988  5.191077 1.241780e-05
edutv        -6.0534053  2.12140697 -2.853486 7.638249e-03
cartoons     -5.1125813  1.88075148 -2.718372 1.064514e-02
```

And now since all p-values are below our 0.05 threshold we would stop with the `score~motheriq+read+edutv+cartoons` model. Checking at this stopping point for multicollinearity would make us then question having both `edutv` and `cartoons` in the model.

```
vif(mod_4)
```

```
motheriq     read    edutv cartoons 
1.143219 1.044483 6.938065 6.878850 
```

```
mod_5<-lm(score~motheriq+read+edutv, data=gifted)
summary(mod_5)
```

```


Call:
lm(formula = score ~ motheriq + read + edutv, data = gifted)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.6035 -1.7634 -0.0054  1.4456  6.7931 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 87.74186   12.18228   7.202 3.52e-08 ***
motheriq     0.40044    0.08076   4.959 2.24e-05 ***
read        11.99889    2.44313   4.911 2.57e-05 ***
edutv       -0.79303    0.95214  -0.833    0.411    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.917 on 32 degrees of freedom
Multiple R-squared:  0.637, Adjusted R-squared:  0.603 
F-statistic: 18.72 on 3 and 32 DF,  p-value: 3.405e-07
```

Removing cartoons as having the higher of the two p-values leaves us with `score~motheriq+read+edutv`, a model that's a bit different from the one found through forward selection with AIC and backward elimination with multicollinearity considered up front.

8.3.3 Step-wise

Lastly we'll use the same step function used in our forward selection example to perform step-wise model selection using BIC. The scope of models to be considered is

```
# we begin with the mod_5 at the end of the backward elimination example.
# scope ranges from intercept-only to the full model as before
# specifying k=ln(n) indicates using BIC to make decisions
stepwise<-step(mod_5, scope=list(lower=mod_start, upper=mod_full),
     direction="both", k=log(nrow(gifted)))
```

```
Start:  AIC=87.18
score ~ motheriq + read + edutv

           Df Sum of Sq    RSS     AIC
+ cartoons  1    52.426 219.93  83.070
+ fatheriq  1    47.491 224.87  83.869
- edutv     1     5.904 278.26  84.372
<none>                  272.36  87.184
+ speak     1    10.237 262.12  89.388
+ count     1     2.686 269.67  90.410
- read      1   205.296 477.65 103.824
- motheriq  1   209.278 481.64 104.123

Step:  AIC=83.07
score ~ motheriq + read + edutv + cartoons

           Df Sum of Sq    RSS     AIC
<none>                  219.93  83.070
+ fatheriq  1    20.167 199.77  83.192
+ speak     1     6.877 213.06  85.510
+ count     1     0.475 219.46  86.576
- cartoons  1    52.426 272.36  87.184
- edutv     1    57.767 277.70  87.883
- read      1   191.180 411.11 102.007
- motheriq  1   226.047 445.98 104.937
```

The output looks much like the output created in our forward selection case with one major difference: at each step the table of possible decisions includes both + terms and - terms. In step 1 the function considers adding `cartoons`, `fatheriq`, `speak`, or `count`, and also taking away `edutv`, `read`, or `motheriq`. Though labeled "AIC" the last column is actually a calculated BIC because we specified `k=log(n)`. In R's `step` function, the value of k specifies the multiplier to be put with the (p+2) measure of model size:

$$step's\ AIC = k(p+2) - 2ln(\hat{L})$$

This means *k=2* indicates you want traditional AIC, and *k=ln(n)* indicates BIC. Remember `log(n)` in R is using natural log.

8.4 Best model of size p

In the `leaps` library you can find a helpful function called `regsubsets`. The `regsubsets` function allows you to have R search every possible model with p predictors and tell you which is best.

Here is an example with our gifted data where regsubsets is asked to consider models ranging from having one predictor (in addition to an intercept) up to four predictors. As inputs we provide the form of the full model, the data set, `nbest` tells R how many "best" models we want of each size, and `nvmax` tells R how large a model we want to work up to.

```r
best_mods<-regsubsets(score~., data=gifted, nbest=1, nvmax=4)
summary(best_mods)
```

```
Subset selection object
Call: regsubsets.formula(score ~ ., data = gifted, nbest = 1, nvmax = 4)
7 Variables  (and intercept)
         Forced in Forced out
fatheriq     FALSE      FALSE
motheriq     FALSE      FALSE
speak        FALSE      FALSE
count        FALSE      FALSE
read         FALSE      FALSE
edutv        FALSE      FALSE
cartoons     FALSE      FALSE
1 subsets of each size up to 4
Selection Algorithm: exhaustive
         fatheriq motheriq speak count read edutv cartoons
1  ( 1 ) " "      "*"      " "   " "   " "  " "   " "
```

```
2  ( 1 ) " "      "*"      " "   " "   "*"  " "  " "
3  ( 1 ) "*"      "*"      " "   " "   "*"  " "  " "
4  ( 1 ) "*"      "*"      "*"   "*"   " "  " "  " "
```

From the top section of this output you can see there are "forced in" and "forced out" options we did not apply to any of the potential independent variables. The lower section then gives the results. The row labeled "1" has an asterisk under `motheriq` indicating the best one-predictor model uses `motheriq`. The row labeled "2" has asterisks for `motheriq` and `read` telling us the best two-predictor model includes those terms. The best three-predictor model is shown to be `score~fatheriq+motheriq+read`, and the best four–predictor model found was `score~fatheriq+motheriq+speak+count`.

Notice the best three-predictor model is not simply a subset of the best four-predictor model. While our forward selection, backward elimination, and stepwise models change only one term at a time making each successive model have only one variable different, the work done by `regsubsets` is exhaustive and doesn't require one-at-a-time changes.

8.5 Lasso Regression

Recall from way back in Chapter 2 that our best linear fit solution comes from wanting to set the $\widehat{\beta}_i$ values in such a way that $\sum(y_i - \hat{y}_i)^2$ is minimized. The lasso technique modifies this goal so that instead we minimize:

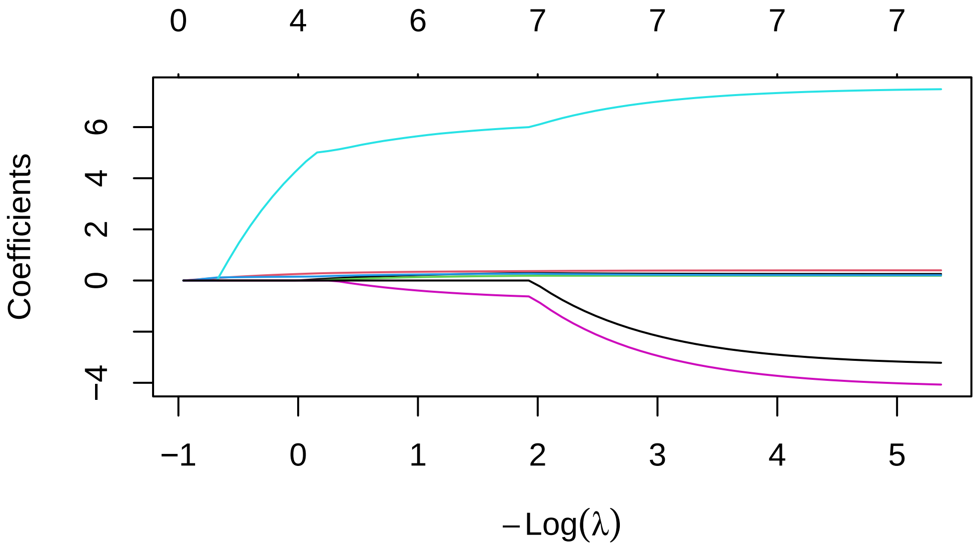
$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \tag{8.1}$$

With this change, we are making the decision that minimizing the sum of squared errors is important, but we also want to minimize the sum of the absolute value of our β parameters weighted by a λ tuning parameter. The value in doing this, is that now we can start with all possible independent variables and see which get eliminated through a $\widehat{\beta} = 0$ fit.

In R

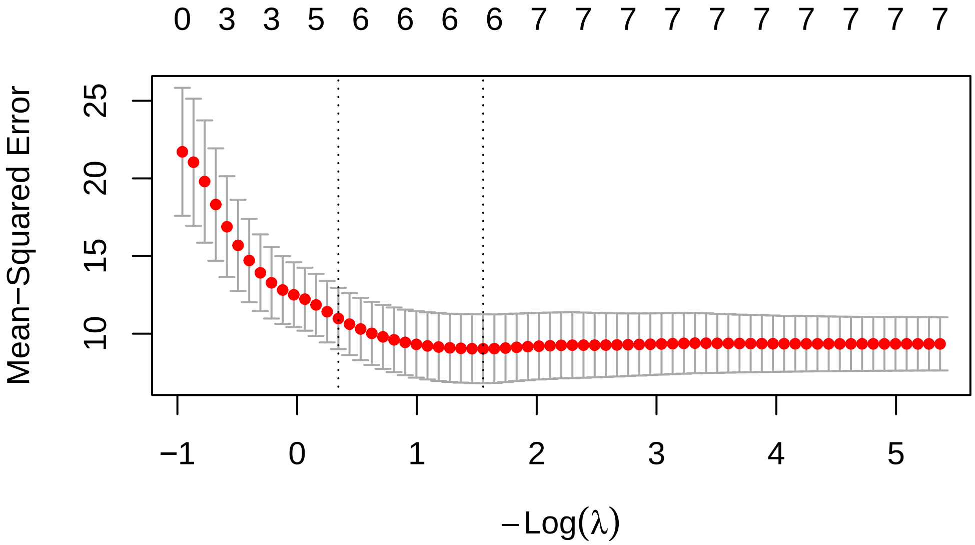
We'll apply this approach in R to our gifted data using the `glmnet` function in the `glmnet` library.

```
# first need our data in format of x and y matrices
mod_x<-as.matrix(gifted[,-1])
mod_y<-as.matrix(gifted[,1])

# use cross validation to explore different values of lambda
set.seed(1234)
step1<-cv.glmnet(mod_x, mod_y, alpha=1)
plot(step1$glmnet.fit)
```



This first plot looks at the values of the β coefficients over a wide range of λ values. Each β has one colored line associated with it. The different consideredλs are along the x-axis, but in the form of $-log(\lambda)$, and the y-axis indicates the value of the βs for each corresponding λ. Along the top of the plot the number of non-zero β coefficients is shown.

```
plot(step1)
```



This plot shows the mean-squared error for our model under various values of λ. Vertical reference lines are drawn at the minimum mean-squared error, and at the minimum plus one standard error. The λs that correspond to each of these key values are retrievable through `step1$lambda.min` and `step1$lambda.1se`.

```
step1$lambda.min
```

```
[1] 0.2115333
```

Then we can fit the model using the identified best λ and see which coefficients are non-zero:

```
step2<-glmnet(mod_x, mod_y, alpha=1, lambda=step1$lambda.min)
lasso.coef<-predict(step2, type="coefficients")
lasso.coef
```

```
8 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept) 63.4282931
fatheriq     0.2681240
motheriq     0.3592196
speak        0.1696803
count        0.2590080
```

```
read         5.8841220
edutv       -0.5542266
cartoons     .
```

If not satisfied with only eliminating cartoons, we could change to a larger, non-optimal lambda:

```
step2<-glmnet(mod_x, mod_y, alpha=1, lambda=1.5)
lasso.coef<-predict(step2, type="coefficients")
lasso.coef
```

```
8 x 1 sparse Matrix of class "dgCMatrix"
                     s0
(Intercept) 130.0439277
fatheriq      .
motheriq      0.1734218
speak         .
count         0.1340442
read          2.1009536
edutv         .
cartoons      .
```

And here we see `motheriq`, `count`, and `read` are the three terms selected as having a non-zero coefficient with $\lambda = 1.5$. To use lasso strictly as a model selection tool, from here you would then return to the lm function to fit a model with these three terms are base any predictions on the $\hat{\beta}$s that result from simply fitting the least-squares criterion, not that seen in Equation 8.1.

```
mod_result<-lm(score~motheriq+count+read, data=gifted)
summary(mod_result)$coef
```

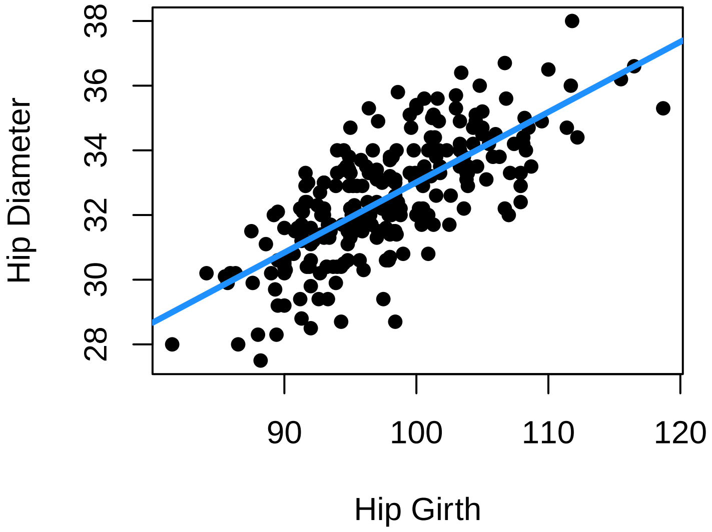
```
              Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 85.3970316 11.39507108 7.4942079 1.565612e-08
motheriq     0.4157893  0.07710859 5.3922568 6.345091e-06
count        0.1820829  0.28048936 0.6491617 5.208663e-01
read         8.9042595  5.88322934 1.5134986 1.399682e-01
```

8.6 Principal Components Analysis (PCA)

Now we're going to consider a much larger data set: the `bdims` data frame in the `openintro` library contains 25 measures on 507 physically active adults. In addition to gender, age, weight, and height, we have access to hip girth, bicep girth, knee diameter, and all kinds of body measurements. It's great information, but as you might suspect these measurements are often highly correlated with each other. Larger hip girths tend to go with larger waists, small ankles make you more likely to have smaller wrists, and so on.

Principal Components Analysis is a way we can incorporate much of the valuable information offered by these 21 body measurements, without making our model have 21 separate slopes and a host of multicollinearity issues. How?

Let's start smaller and focus on two of our body measurements: `hip_gi`, the hip girth as measured at the same level as `bit_di`, the bitrochanteric diameter. Bitrochanteric diameter is basically hip width at the top of the femur but below the pelvis. Obviously these two measurements should be correlated and they are. Limiting to only the 247 men in the `bdims` data frame, the two hip measurements are plotted here in Figure 8.2.



(a) Original data

Figure 8.2: Male Hip Measurements

Now look at the plot with the regression line overlaid and think about rotating the entire scatter plot to right a bit so that regression line was the new x-axis. Go ahead, pick up your book or screen (or turn your head) and rotate the graph so the blue line is our new horizontal. If you knew where along that new x-axis a person lied, you'd know a lot about their hip size. You'd have a combination of hip girth and hip diameter information with just that one blue-axis variable. The vertical variation up and down from that line is the relatively small amount of information you're giving up by only using one variable instead of the original two.

We could repeat this process over and over with pairs of body measurements we know are related, at each point picking the one-dimensional combination of them and effectively cutting our model size in half while giving up relatively little. But 1) that's time consuming and 2) what if we don't have the proper knowledge to know which terms to pair?

Stop and think about how that blue line does such a good job simplifying the two dimensions into one main direction of variability and then one minor direction of variability. What if we could take our full 21 dimensions of body measurements and get the computer to tell us the axes that align with the highest variability and the least variability? That is what we can do with ***principal components analysis***, often referred to simply as ***PCA***.

The linear algebra that's driving this change in basis is beyond our scope here, but instead we will focus on how to apply the method in R.

In R

First, the data must be numeric in nature. For our body measurements, we'll look only at women, and only at the first 21 columns which cover `bia_di` (biacromial diameter) through `wri_gi` (wrist girth). There are a few different functions in R that will perform PCA, but we'll use the `prcomp` option that comes in the `stats` library.

```
womens_bdims<-bdims[bdims$sex==0,1:21]

# or if you prefer:

womens_bdims<-bdims |>
  filter(sex==0) |>
  select(bia_di:wri_gi)

pca_body<-prcomp(womens_bdims, scale.=TRUE)
```

Note that when `prcomp` is called, the `scale.=TRUE` is added in. What this does is tell R that before finding the axes of highest variability, the data should be scaled. This keeps us from labeling the dimensions that just happen to have bigger numbers from dominating the findings. If all body measurements were taken in cm but we transformed knee diameter into mm, we'd think a difference of 100 in the knee measurement was huge and a difference of 10cm in a waist was tiny. The `scale.=TRUE` takes each column and converts it to z-scores using the column means and standard deviations. This was a 3-standard deviation high measure on calf girth gets the exact same attention as a 3-standard deviation high measure on ankle girth. Always scale in PCA.

Figure 8.3 shows the first four principal components. The `prcomp` function will create as many principal components (new axes) as there are original dimensions to your data. They are ordered based on the amount of variability they can explain. Notice that every plot in the scatter plot matrix looks like just a random blob. That will always happen as these new axes will be uncorrelated with each other by design.

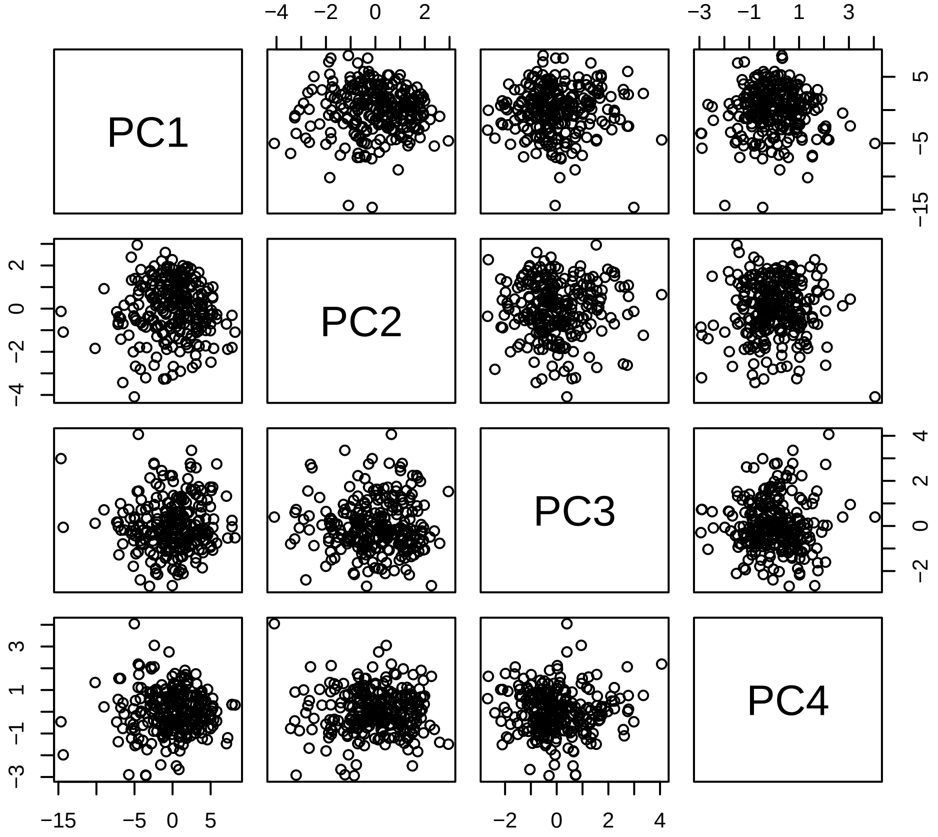


Figure 8.3: PCA Results

A `biplot` can be used to get a sense of how the original columns relate to the new principal components. Two biplots showing PC1 vs. PC2 and PC3 vs. PC4 are below in Figure 8.4.

```
par(mfrow=c(1,2))
par(mar=c(4,5,2,2))
biplot(pca_body)
biplot(pca_body, choices=c(3,4))
```

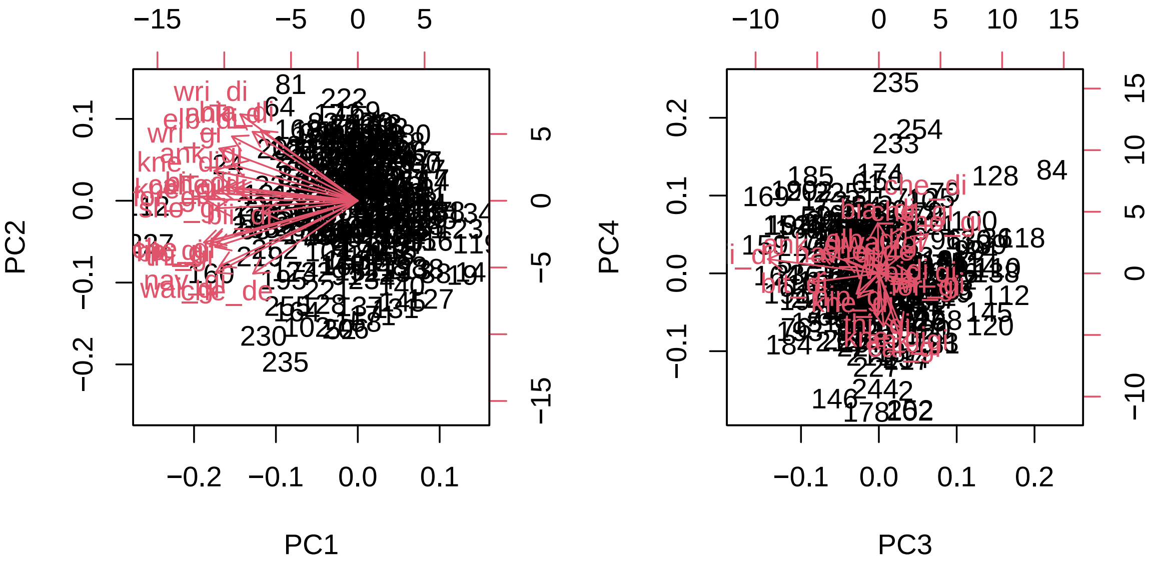


Figure 8.4: PCA Results

Both of these **biplots** are a bit hard to read because there are 21 red vectors all on top of each other. The first gives some discernable information though. It shows that PC1 involves a negative multiple of every column, while PC2 looks to be about half positive and half negative. The numeric information for how the original columns map to the principal components is given in the `$rotation` element of our completed `prcomp` object.

Here are the mappings for the first two principal components, rounded to five decimal places. They confirm the all-negative loadings for PC1 and the roughly even positive/negative split for PC2.

```
round(pca_body$rotation[,1],5)
```

```
  bia_di   bii_di   bit_di   che_de   che_di   elb_di   wri_di   kne_di 
-0.16001 -0.15629 -0.21022 -0.17278 -0.20168 -0.20772 -0.19393 -0.23672 
  ank_di   sho_gi   che_gi   wai_gi   nav_gi   hip_gi   thi_gi   bic_gi 
```

```
-0.17357 -0.23440 -0.24326 -0.23461 -0.22822 -0.25381 -0.23532 -0.24366
  for_gi   kne_gi   cal_gi   ank_gi   wri_gi 
-0.25137 -0.24002 -0.22807 -0.20682 -0.22877 
```

```
round(pca_body$rotation[,2],5)
```

```
  bia_di   bii_di   bit_di   che_de   che_di   elb_di   wri_di   kne_di 
 0.31718 -0.05449  0.06264 -0.32902  0.05383  0.29060  0.39074  0.13547 
  ank_di   sho_gi   che_gi   wai_gi   nav_gi   hip_gi   thi_gi   bic_gi 
 0.31147 -0.03461 -0.18167 -0.32502 -0.29562 -0.18830 -0.20851 -0.19328 
  for_gi   kne_gi   cal_gi   ank_gi   wri_gi 
 0.00609  0.03437  0.04863  0.16233  0.23538 
```

Interested in how much of the total variability can be seen in each principal component? That information is seen in the `$sdev` element of the `prcomp` object and can be seen in a bar chart with the plot command:

```
# convert standard deviations to variance by squaring
variances<-pca_body$sdev^2

# look at variance for first 8 components, rounded
round(variances[1:8],4)
```

```
[1] 12.3825  1.6555  1.2828  1.0451  0.8028  0.5475  0.4608  0.4358
```

```
# proportion of total variance for first 8 components, rounded
round(variances[1:8]/sum(variances),4)
```

```
[1] 0.5896 0.0788 0.0611 0.0498 0.0382 0.0261 0.0219 0.0208
```

```
# see plot of variance per component
plot(pca_body)
```

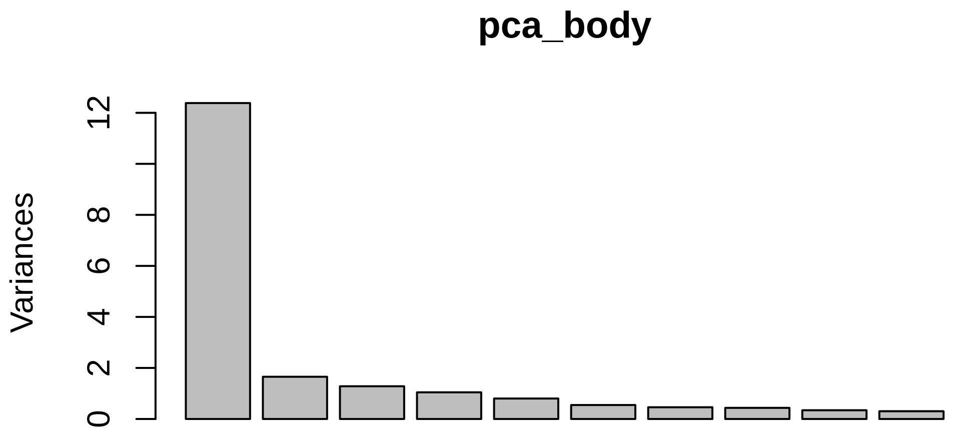


Figure 8.5: Component Variance Plot

But what does any of this have to do with regression? Well, now that we have our principal components we can use them as predictors in our model to maximize the information from our data set while minimizing the number of β parameters that need estimating.

```
#create BMI outcome to estimate with the measurements
bmi<-bdims$wgt/(bdims$hgt/100)^2

#limit to only the women
bmi_f<-bmi[bdims$sex==0]

#fit model with first seven principal components
mod_bmi<-lm(bmi_f~pca_body$x[,1:7])

#look at model summary
summary(mod_bmi)
```

```
Call:
lm(formula = bmi_f ~ pca_body$x[, 1:7])
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-3.8904 -0.7937 -0.0327  0.7054  4.1316 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)          22.27793    0.07581 293.853  < 2e-16 ***
pca_body$x[, 1:7]PC1 -0.77013    0.02159 -35.677  < 2e-16 ***
pca_body$x[, 1:7]PC2 -0.88505    0.05904 -14.992  < 2e-16 ***
pca_body$x[, 1:7]PC3  0.35202    0.06707   5.249 3.25e-07 ***
pca_body$x[, 1:7]PC4 -0.22522    0.07430  -3.031  0.00269 ** 
pca_body$x[, 1:7]PC5 -0.05495    0.08478  -0.648  0.51751    
pca_body$x[, 1:7]PC6  0.24817    0.10266   2.417  0.01634 *  
pca_body$x[, 1:7]PC7 -0.11019    0.11190  -0.985  0.32573    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.222 on 252 degrees of freedom
Multiple R-squared:  0.8595,	Adjusted R-squared:  0.8556 
F-statistic: 220.2 on 7 and 252 DF,  p-value: < 2.2e-16
```

This model summary shows us that just because one component covers more variability than another, doesn't mean that component is more important when it comes to predicting our outcome. PC6 covers only 2.61% of the total variability compared to PC5's 3.82%, but when it comes to predicting a person's BMI PC6 is more helpful than PC5. Since PC1 and PC2 are so strongly related to BMI, it turns out a model with just those two predictors has an $R^2_{adj} = 0.8337$.- not bad for a 3-β model on a complex problem.

You can run forward selection, backward elimination, or step-wise building algorithms on the principal components just as you would the raw data - but now with the advantage of not having multicollinearity to navigate.

PCA isn't without it's downsides. Using this type of approach and model might be very powerful, but it isn't nearly as interpretable as a model built on the original data. Depending on the context of the work, interpretability might take a back seat to predictive power which makes PCA a strong option. Also, sometimes looking at the PCA loadings themselves can provide insight into the interdependencies and relationships within your data.

Bottom line: PCA is a powerful tool to have in your tool box, but it shouldn't be the first tool you turn to when fitting a model.

9 Well that's weird

9.1 Studentized Residuals

So you fit a model and all of your residuals are within ±1. That's great! Isn't it? Well it all depends on the scale of the data you're working with. If you are trying to model the weight of adults based on waist circumference than being consistently within ±1 lb would be great. If instead you are trying to predict a student's blood alcohol level based on the number of beers they've had a ±1 error is pretty lousy since no one has a blood alcohol over 1 (or even 0.5).

Simply put, studentized residuals are a standardized residual measure to help you assess the size of the errors without needing to know detail around the context of the data. A studentized residual is obtained by dividing the raw residual by an estimate of its standard deviation. What is the estimate of its standard deviation? Refer back to Equation 7.3 in Chapter 7 and you'll see this is something we've already worked with and H, the hat matrix, plays a key role.

The studentized residual for the ith observation is given by:

$$t_i = \frac{e_i}{\sigma\sqrt{(1-h_{ii})}}$$

where:

- e_i = residual for the ith observation
- σ = mean squared error estimate as seen before
- h_{ii} = the ith diagonal of the hat matrix, $H = X(X^TX)^{-1}X^T$

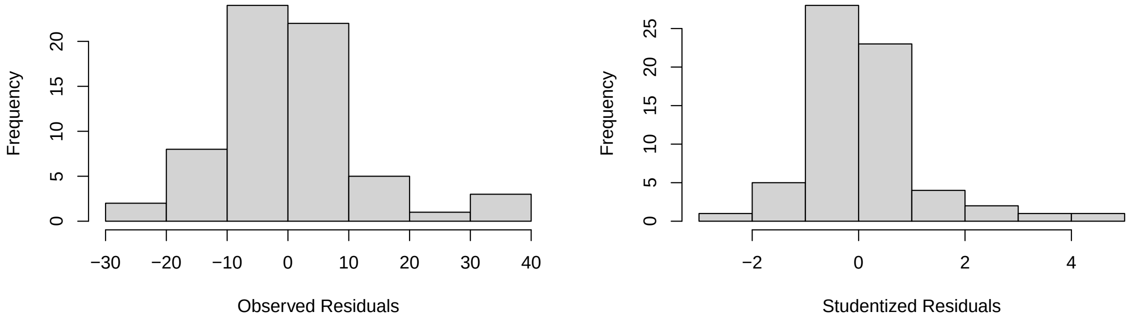
Fundamentally, this approach is no different than the z-score formula you likely saw in intro statistics or the test-statistic format you've seen in basic hypothesis testing. The t_i studentized residual is just the observed error, minus the expected error which is always zero by design, then divided by a standard error.

So now that you have studentized residuals what can you do with them? Their main use is in being able to quickly identify outlier observations and have a consistent scale in which to think of just how much of an outlier they actually are. An outlier point with a studentized residual of 2-3 isn't too bad, but an studentized residual of 17 is worth a look no matter what the scale of your y variable is.

In R

To get the studentized residuals for a fitted model you can use the `rstudent` function contained in the `stats` library. The example here explores calorie content of cereals from the `UScereal` data set in the `MASS` library.

```
model_cereal<-lm(calories~fat+sugars+carbo, data=UScereal)
stud.res<-rstudent(model_cereal)

hist(model_cereal$residuals, main="", xlab="Observed Residuals", breaks=8)
hist(stud.res, main="", xlab="Studentized Residuals", breaks=8)
```



(a) Observed Residuals

(b) Studentized Residuals

Figure 9.1: Residuals for Cereal Model

A summary of studentized residuals shows the most extreme outlying point has $t_i = 4.654$ which is definitely worth a second look. You can refer to a Student's t-distribution with $n - (p + 1)$ degrees of freedom and see that a residual even 2.5 or larger should only happen about 0.76% of the time.

```
pt(2.5, nrow(UScereal)-(3+1), lower.tail = FALSE)
```

```
[1] 0.007562028
```

9.2 Leverage and Cook's Distance

Outlier points in regression are not necessarily a problem. Where problems do occur is when the outlier is in a position that causes it to have undue influence over the regression fit. Consider Figure 9.2 below with two distinct outliers on the far right both at an x-value of 40.

In the case of the navy outlier point, it is far removed from the typical x values, but lies on the same x-y line that seems to fit the non-outlier points so it doesn't exert an unusual amount of influence on the blue line fit.

The red outlier point however lies much higher than the linear pattern of non-outliers would expect. Because of this, inclusion of that one point when fitting the model moves the line from the blue fit to the one shown in orchid.

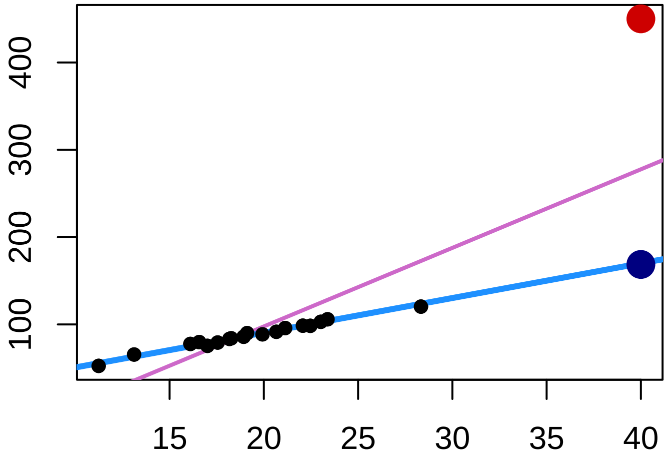


Figure 9.2: Example of Leverage

To identify outlier points that might be influential, you can use the ***leverage*** metric. The leverage for a point is simply that point's corresponding diagonal element of the "hat matrix". On average, leverage will be about $(p+1)/n$, and any point with a leverage greater than 2.5 times that is worth a closer look.

Another popular diagnostic metric to assess when a point might be unduly influential is ***Cook's Distance***. Cook's distance combines information from both the residual size and the leverage and is defined as:

$$D_i = \frac{e_i^2}{(p+1)\hat{\sigma}^2} \times \frac{h_{ii}}{(1-h_{ii})^2} = \frac{t_i^2}{p+1} \times \frac{h_{ii}}{1-h_{ii}}$$

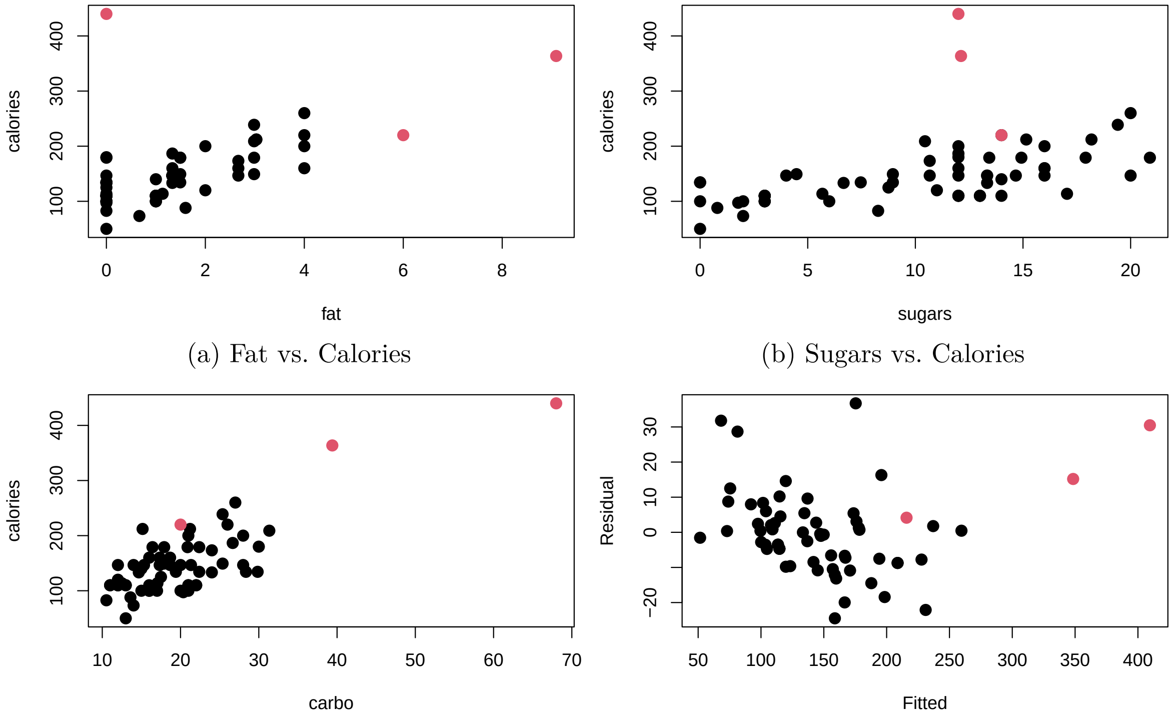
where:

- e_i = residual for the i^{th} observation
- t_i = studentized residual for the i^{th} observation
- p = the number of independent variables in your model
- $\hat{\sigma}^2$ = the observed standard error of residuals
- h_{ii} = the i^{th} diagonal of the hat matrix, $H = X(X^TX)^{-1}X^T$

This highlights that Cook's distance increases if an observation has a high studentized residual, has a high leverage, or both. A rule of thumb is to examine points with $D_i > 4/n$, where n is the total number of observations. Ultimately though, any value that stands out from most others is worthy of a closer inspection.

In R

The functions for leverage and Cook's distance are built into the `stats` library that loads by default. Continuing with the cereal nutrition example we find that there are three points with high leverage values.



(a) Fat vs. Calories

(b) Sugars vs. Calories

(c) Carbohydrates vs Calories

(d) Model Fitted vs. Residuals

Figure 9.3: Leverage Example

The `hatvalues` function returns the hat matrix diagonal for a fitted model object. From there it is a simple task to identify those that exceed our threshold of two and a half times $(p+1)/n$. Figure 9.3 above shows four plots of the cereal data with high leverage points marked in red. Leverage calculations are shown below.

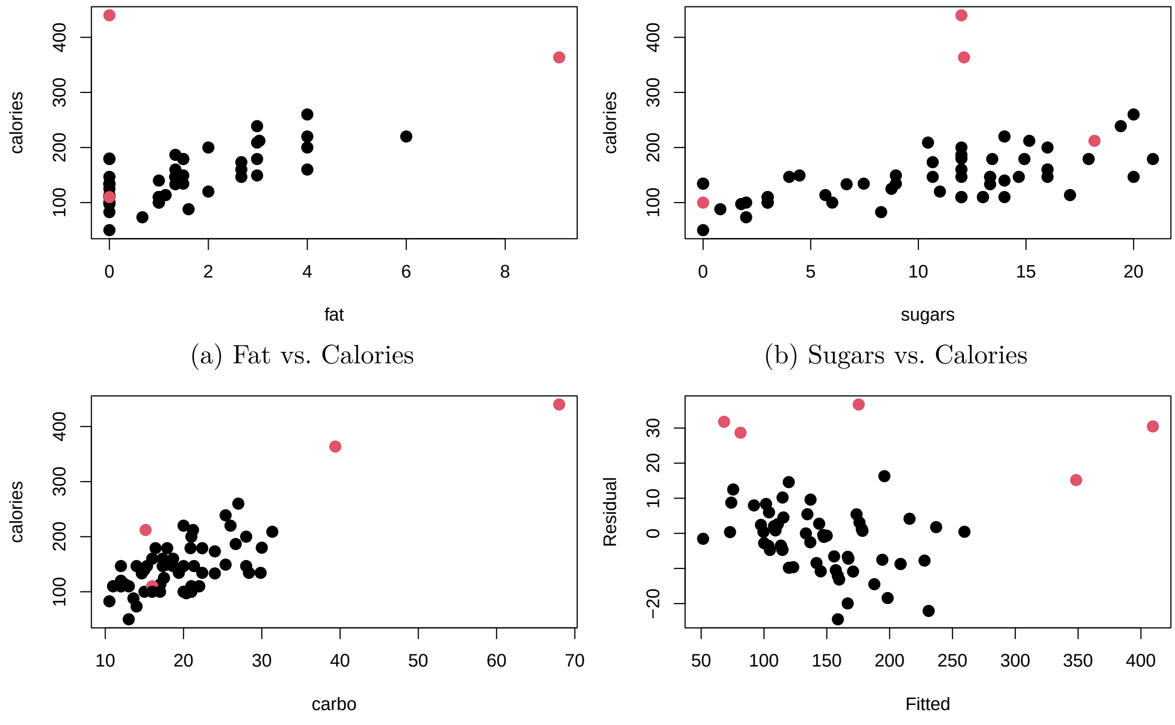
```
model_cereal<-lm(calories~fat+sugars+carbo, data=UScereal)

lev_cereal<-hatvalues(model_cereal)
((2+1)/nrow(UScereal))*2.5
```

```
[1] 0.1153846
```

```
sort(lev_cereal, decreasing=TRUE)[1:6]
```

```
                 Grape-Nuts          Great Grains Pecan           Cracklin' Oat Bran
                 0.61399054                  0.42563479                   0.14574239
Fruitful Bran Shredded Wheat spoon size         Oatmeal Raisin Crisp
                 0.08625599                  0.08324148                   0.08255166
```

The process is similar for calculating Cook's distance. There are four cereals flagged with the Cook's distance metric with Grape-Nuts becoming identified as a very significant standout with a $D_i > 6$. Here in Figure 9.4 are the same four cereal plots with red points indicating those with a Cook's distance greater than the $4/n$ threshold.



(a) Fat vs. Calories

(b) Sugars vs. Calories

(c) Carbohydrates vs Calories

(d) Model Fitted vs. Residuals

Figure 9.4: Cook's Distance Example

Below shows using the `cooks.distance` function, and a listing of the top six values.

```
cd_cereal<-cooks.distance(model_cereal)
4/nrow(UScereal)
```

```
[1] 0.06153846
```

```
sort(cd_cereal, decreasing=TRUE)[1:6]
```

```
               Grape-Nuts        Great Grains Pecan                 100% Bran
               6.43414435                0.50046528                0.13898120
All-Bran with Extra Fiber                 Special K Nutri-Grain Almond-Raisin
               0.12924058                0.06608620                0.04776351
```

9.3 DFFits and DFBetas

To understand the exact influence yielded by a row of your data, ***DFFits*** and ***DFBetas*** are used. DFFits will tell you how the fit of your model changes with the inclusion and exclusion of each point. DFBetas tells you how the estimated $\hat{\beta}$ parameters change with the inclusion and exclusion of each point.

For example, in our `calories~fat+sugars+carbo` model in the UScereal data frame, the DFFit for Cinnamon Toast Crunch is -0.237. That means that when the model is fit including Cinnamon Toast Crunch (CTC) in the training set, the fitted value for CTC is $0.237\sigma_{CTC}$ lower than when the model is fit excluding CTC from the training set. Note that it is in units of σ specific to the CTC observation, not the broader σ of the model. In general, the formula for DFFits for observation i is:

$$DFFits_i = \frac{\hat{y}_i - \hat{y}_{(i)i}}{\sqrt{\hat{\sigma}^2_{(i)} h_{ii}}}$$

where:

- $\hat{y}_i$ = the fitted value for the i^{th} observation when all data trains the model
- $\hat{y}_{(i)i}$ = the fitted value for the i^{th} observation when model built without observation i
- $\hat{\sigma}^2_{(i)}$ = the observed mean squared error of residuals in model built without observation i
- h_{ii} = the i^{th} diagonal of the full model hat matrix, $H = X(X^TX)^{-1}X^T$

DFBetas is the same idea, but looking at the difference in β values with and without each observation rather than at the difference in fits.

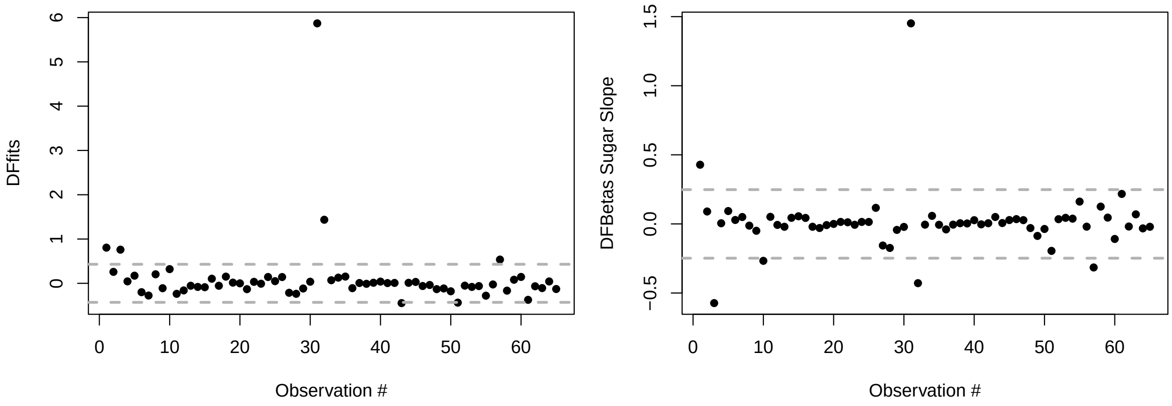
$$DFBetas_j = \frac{\hat{\beta}_j - \hat{\beta}_{j(i)}}{\sqrt{\hat{\sigma}^2_{j(i)}(X^TX)^{-1}_{jj}}}$$

where:

- $\hat{\beta}_j$ = the estimate for the β_j when all observations train the model
- $\hat{\beta}_{j(i)}$ = the estimate for the β_j when the model is built without the i^{th} observation
- $\hat{\sigma}^2_{j(i)}$ = the observed mean squared error of residuals in model built without observation i
- X = the X matrix of data including the column of 1s for the intercept
- $(X^TX)^{-1}_{jj}$ = the j^{th} diagonal of the $(X^TX)^{-1}$ matrix

You can refer back to Chapter 7 to review the derivation of $\hat{\sigma}^2(X^TX)^{-1}$ as the variance matrix for the β vector.

Simple plots of DFfits and DFBetas are often helpful in spotting influential observations. An example of this is in Figure 9.5. As a general rule, DFfits values greater than $2\sqrt{p/n}$ or less than $-2\sqrt{p/n}$ are worthy of a second look so reference lines have been included in Figure 9.5 (a) to show those thresholds. For Figure 9.5 (b) reference lines are included at $\pm 2/\sqrt{n}$, the general rule of thumb for evaluating DFBetas.



(a) DFFits

(b) DFBetas for sugars

Figure 9.5: DFFits and DFBetas

In R

Here are the series of steps to calculate out the DFFits value of Cinnamon Toast Crunch, the 11^{th} row of USCereals, in R.

```
mod_cereal<-lm(calories~fat+sugars+carbo, data=UScereal)
fit1<-mod_cereal$fitted.values[11]

mod_noCTC<-lm(calories~fat+sugars+carbo, data=UScereal[-11,])
fit2<-predict(mod_noCTC, UScereal[11,])

CTC_hat<-hatvalues(mod_cereal)[11]
CTC_sig<-sqrt(summary(mod_noCTC)$sigma^2 * CTC_hat)

(fit1-fit2)/CTC_sig
```

```
Cinnamon Toast Crunch
           -0.2372347
```

The easier way is to use the dffits function that is part of the stats library in R. The only input needed is the fitted model object, and DFFits for all observations will be returned.

```
mod_cereal<-lm(calories~fat+sugars+carbo, data=UScereal)
dfs<-dffits(mod_cereal)
dfs[11]
```

```
Cinnamon Toast Crunch
           -0.2372347
```

For DFBetas, here is the DFBeta for the sugars slope considering the Cinnamon Toast Crunch row of data.

```
  mod_cereal<-lm(calories~fat+sugars+carbo, data=UScereal)
  fit1<-mod_cereal$coef[3]

  mod_noCTC<-lm(calories~fat+sugars+carbo, data=UScereal[-11,])
  fit2<-mod_noCTC$coef[3]

  sig1<-summary(mod_noCTC)$sigma^2
  xmat<-as.matrix(cbind(rep(1, 65), UScereal[,c(4,7,8)]))
  inv<-solve(t(xmat)%*%xmat)
  siguse<-sqrt(sig1*inv[3,3])

  (fit1-fit2)/siguse
```

```
    sugars 
0.08030943 
```

To do this automatically in R you can use the `dfbetas` function. The `dfbetas` function returns a full matrix of values with each row corresponding to an observation, and each column corresponding to a different β. For the sugar slope change associated with omitting Cinnamon Toast Crunch we'll want the 11^{th} row and the 3^{rd} column.

```
dfb<-dfbetas(mod_cereal)
dfb[11,]
```

```
(Intercept)         fat      sugars       carbo 
-0.06653261 -0.19889228  0.05119340  0.07823310 
```

Be aware that in addition to the `dfbetas` function, there is also a `dfbeta` function. The `dfbeta` function without the trailing s, provides the raw difference in β fits without scaling by the standard error of β. This can also be useful sometimes, but it is important to know which one you're looking at.

```
dfb_raw<-dfbeta(mod_cereal)
dfb_raw[11,]
```

```
(Intercept)         fat      sugars       carbo 
-0.32113477 -0.20768424  0.01484826  0.01446437 
```

Benson, Janette B. 1993. "Season of Birth and Onset of Locomotion: Theoretical and Methodological Implications." *Infant Behavior and Development* 16 (1): 69–81. https://doi.org/10.1016/0163-6383(93)80029-8.

Çetinkaya-Rundel, Mine, David Diez, Andrew Bray, Albert Y. Kim, Ben Baumer, Chester Ismay, Nick Paterno, and Christopher Barr. 2024. "Openintro: Datasets and Supplemental Functions from 'OpenIntro' Textbooks and Labs." https://doi.org/10.32614/CRAN.package.openintro.

Geiman, Claire, and Eric Long. 2023. "Data from: Allometric Brain Reduction in an Insular, Dwarfed Population of Black-Tailed Deer." Dryad. https://doi.org/10.5061/DRYAD.GQNK98STM.