Go Bigger

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.

And 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:

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

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

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.

Units change on just one term...

So hmmm... maybe meters of climb isn't the unit of measurement we want to use. Let's change it so that we can get the amount of time added per km of climb. That means 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:

```r
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
```

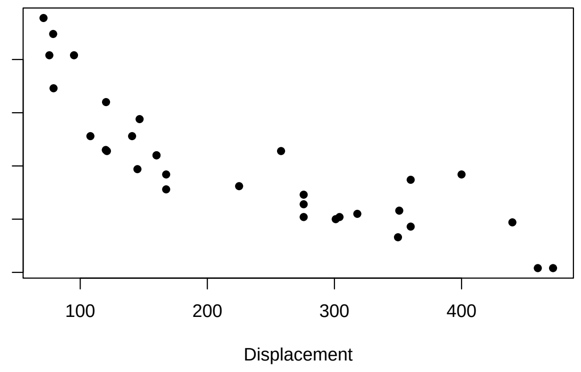
Which makes perfect sense and behaves just as we saw in the simple linear example from Chapter 5: we divided our x_2 by 1000, so to get the same result from $\beta_2 \times x_2$ the value of β_2 would need to increase by a factor of 1000. And now our interpretation includes that the winning time increase 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 value for the associated variable will simply change by the inverse. 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 term by term.

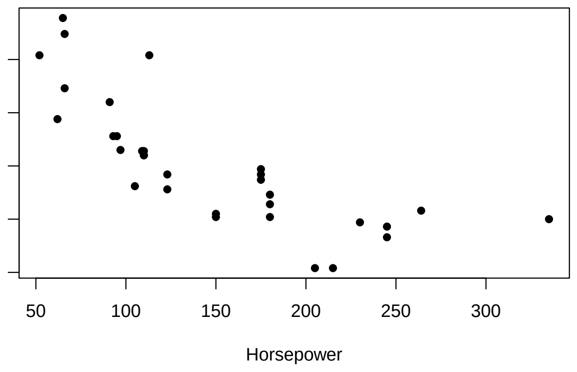
Plotting and transformations

When working in more dimensions we now have a challenge in how we can visualize the data. Generally we can only plot two continuous dimensions (sometimes three) in a way our brains can understand simply because .

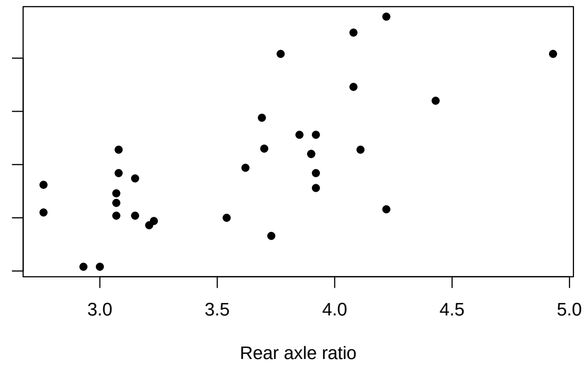
A good first step is to look at the simple pairwise relationships between each x predictor and the outcome y. Let's expand beyond our two-x hill race model and look at a larger data set with multiple predictors: the mtcars dataset we've seen several times before. Figure 1 shows mpg data plotted against displacement, horsepower, read axle ratio, and weight, all from `mtcars`.



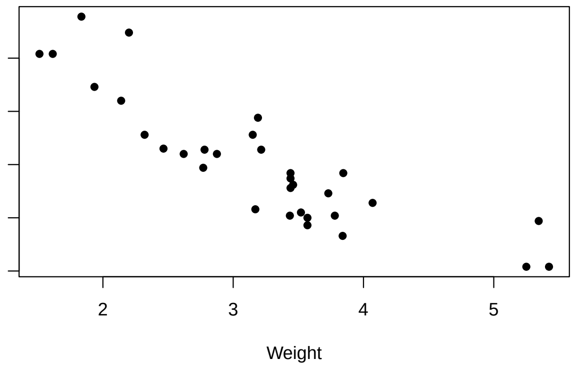
(a) Displacement and MPG



(b) Horsepower and MPG



(c) Rear axle ratio and MPG



(d) Weight and MPG

Figure 1: Motor Trend Car Data

This shows us that all four pictured independent variables have a clear relationship with fuel efficiency as measured by MPG, but some have curvature in that relationship. Fitting the model without any transformations, the resulting fitted vs. residual plot comes out as:

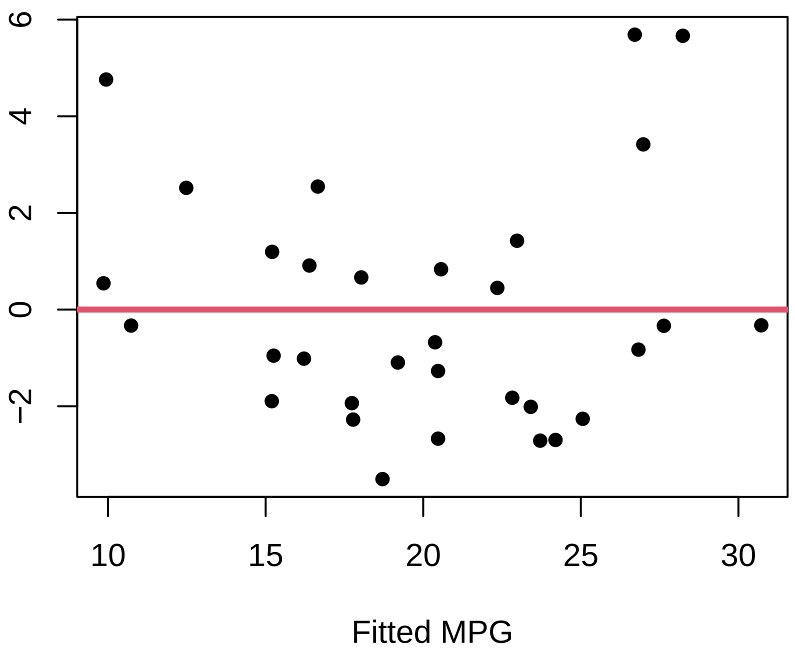
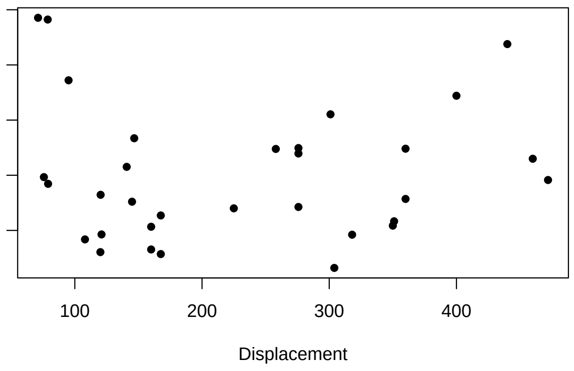


Figure 2: Motor Trend Fitted vs. Residual Plot

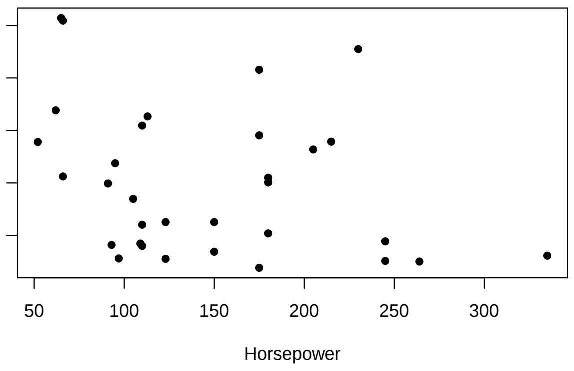
Clearly the curvature seen in the initial plots is playing role with how well our multiple regression model fits.

Transformations

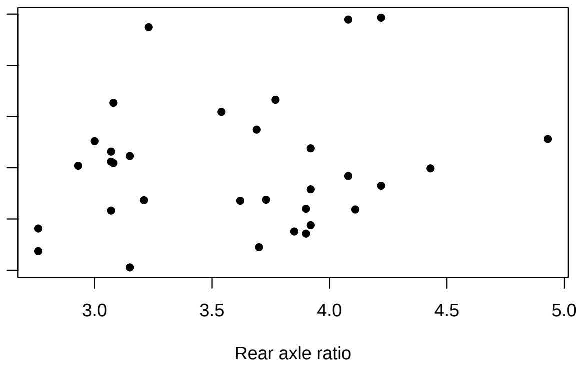
A good first step is to look at the simple pairwise relationships between each x predictor and the outcome y. Let's expand beyond our two-x hill race model and look at the `UScereals` data in the `MASS` data. Figure 7 below shows our four predictors: protein, fat, carbohydrates, and sugars each plotted against the dependent variable: calories per serving.



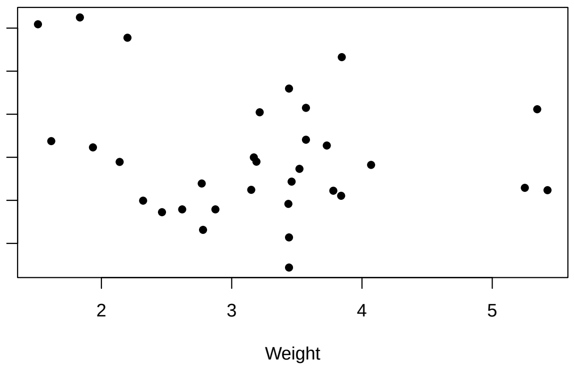
(a) Displacement



(b) Horsepower

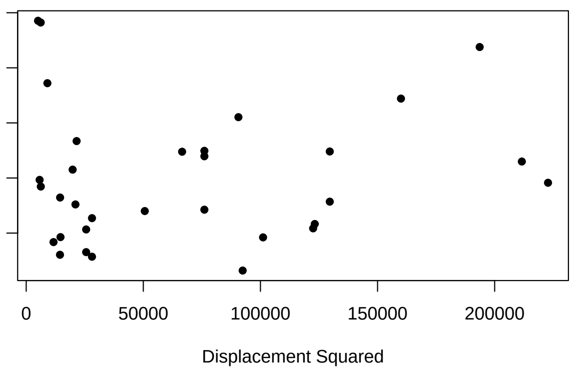


(c) Rear axle ratio

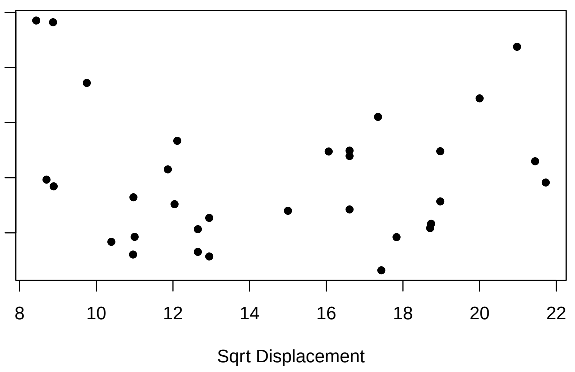


(d) Weight

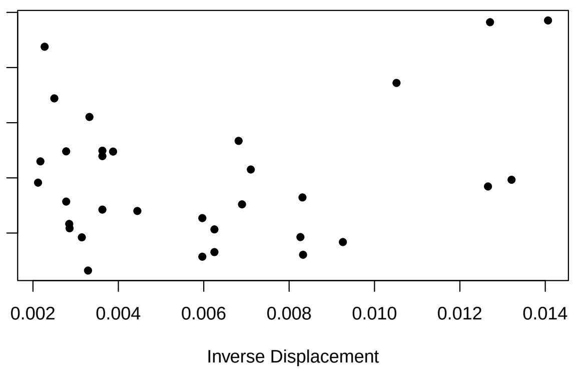
Figure 3: Motor Trend Partial Residual Plots



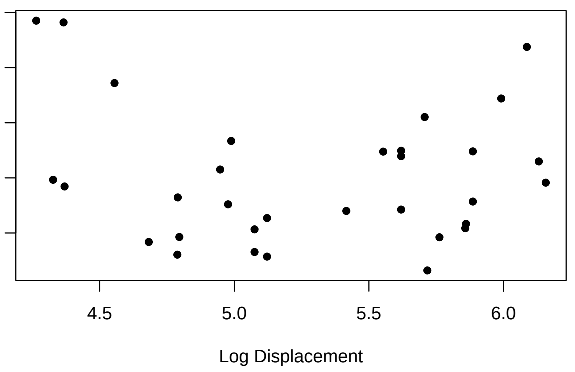
(a) Displacement Squared



(b) Square-root Displacement

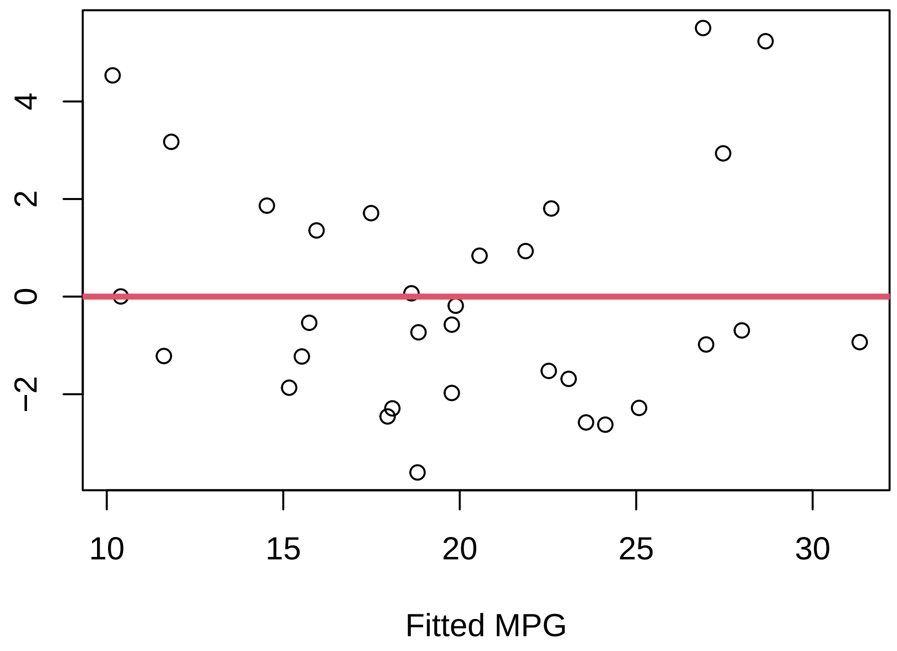


(c) Inverse Displacement



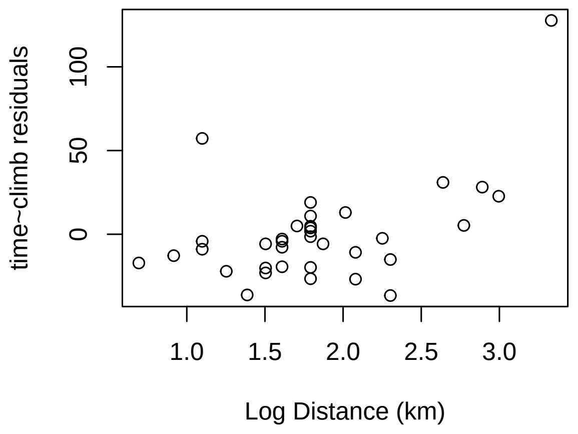
(d) Log Displacement

Figure 4: Displacement Transformations and Residuals

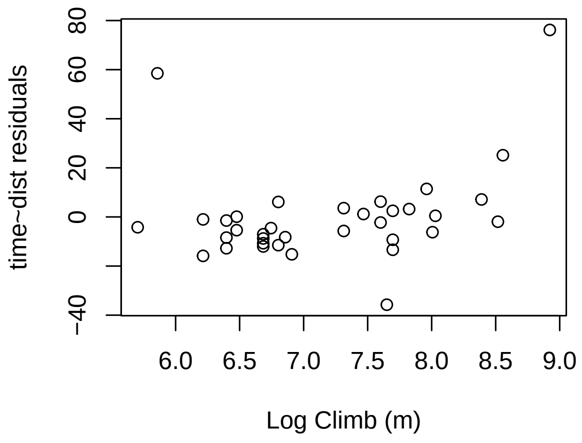


(a) Model with Displacement^2 plus HP, Axle Ratio, and Wt

Figure 5: Motor Trend Car Data

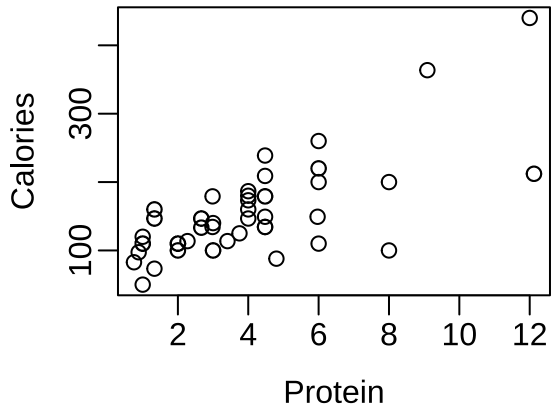


(a) Log(dist) vs. residuals of time~climb

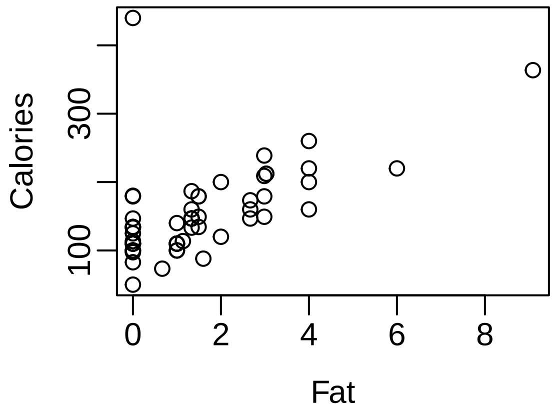


(b) Log(climb) vs. residuals of time~dist

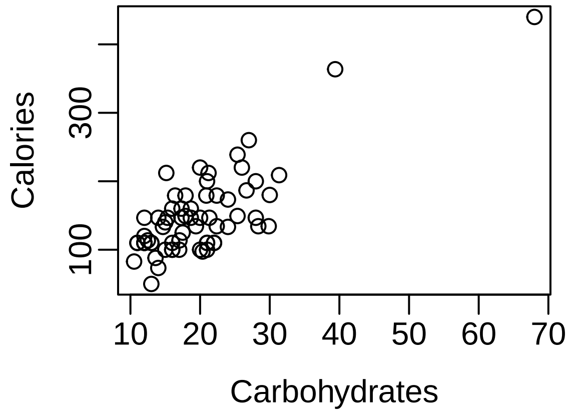
Figure 6: Scottish Hill Races, Log transformations



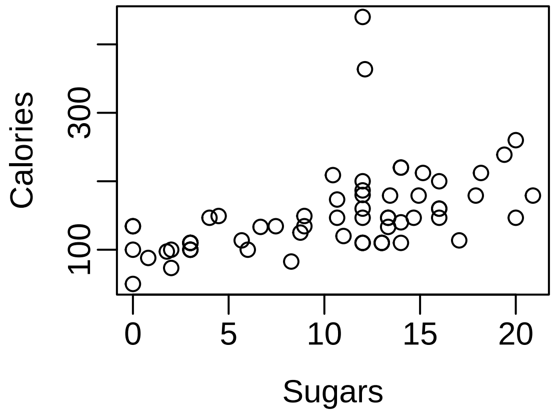
(a) Protein vs. Calories



(b) Fat vs. Calories



(c) Carbohydrates vs. Calories



(d) Sugars vs. Calories

Figure 7: Calories in Cereal

Each of our four initial plots look fairly linear and appropriate for inclusion in our model. Protein and sugars might introduce some heterskedasticity, and there is one clear outlier in the fat plot and a few on the sugar plot, but these plots all look like a reasonable place to start. So we fit the multiple regression model and then create the fitted vs. residual plot to get a sense of how closely your model is following the data across all dimensions combined.

Confession: In the case of a single predictor you could have been using an X vs. residual plot this whole time, but with multiple regression you need the fitted value to serve as a combination of all independent terms in the model. Therefore, it's not bad to be in the habit of evaluating *fitted* vs. residual plots from the start.

Just as in the simple one-x case, If the fitted vs. residual model has curvature, you have a curvature problem to address. If the fitted vs. residual plot shows a change in residual variance, you have a heteroskedasticity problem to address. Once a problem is identified, you'll follow up with more plots to uncover which of the X are the root cause. It might be one X, but it also might be more than one.

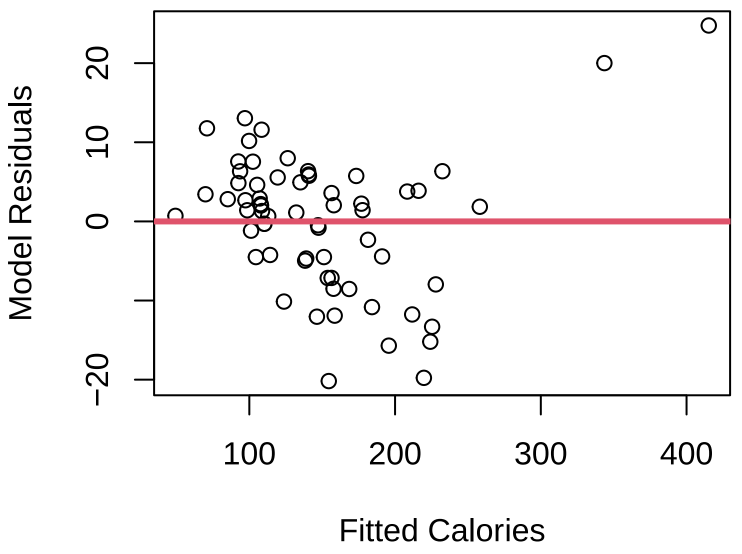


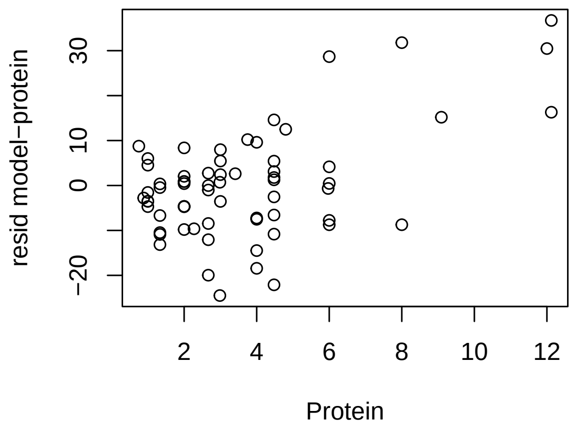
Figure 8: Cereal Calories fitted vs. residuals

This is where the partial regression plot comes in. To view how each X variable relates to your Y after the impact of the other X's has been considered you can plot x_i vs. the residuals from a model predicting Y using all $x_1 .... x_p$ except x_i. These are the four plots shown in Figure 9.

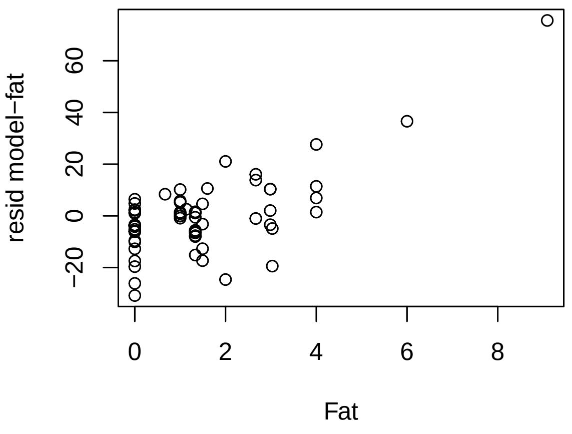
The encouraging take-away shown in these plots is that every variable is adding something new to our knowledge of calories. This is known by the fact that all plots show a definite relationship between their x_i and the residuals from the model not including x_i.

These plots also show us that the only relationship that isn't linear and shows a curve similar to that seen in Figure 8 is the one between protein and the residuals of the model not including protein. So what can be done to make that relationship linear? Transformations. Figure 10 shows the results of the four basic transformations applied to protein and then plotted against the partial residuals from the model without protein. As should have been anticipated given the parabolic shape, the protein-squared model is the most linear.

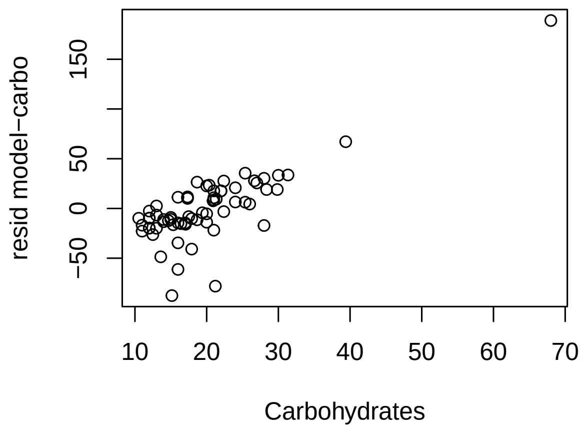
To picture this a bit better, you can space out the values of protein-squared less than 30 by first adding a constant to protein and then squaring it. This can make the plot more clear, but as demonstrated through the lines below, this doesn't fundamentally change the relationship between Protein and Calories since there's a constant in our model already.



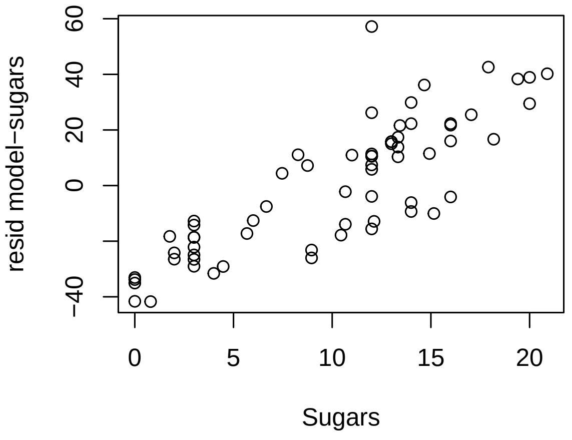
(a) Protein partial regression plot



(b) Fat partial regression plot

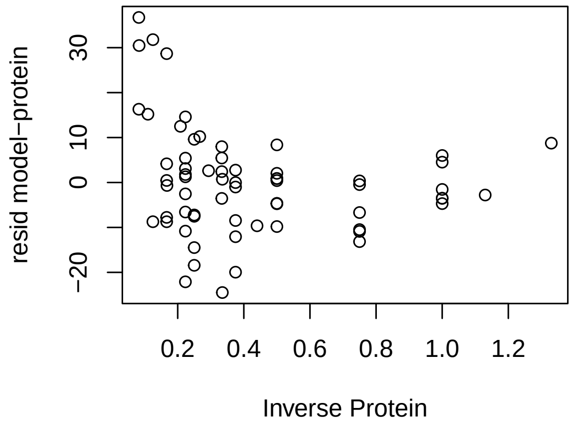


(c) Carbo partial regression plot

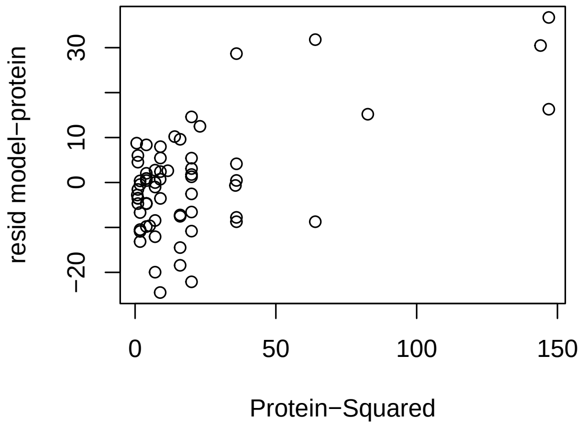


(d) Sugars partial regression plot

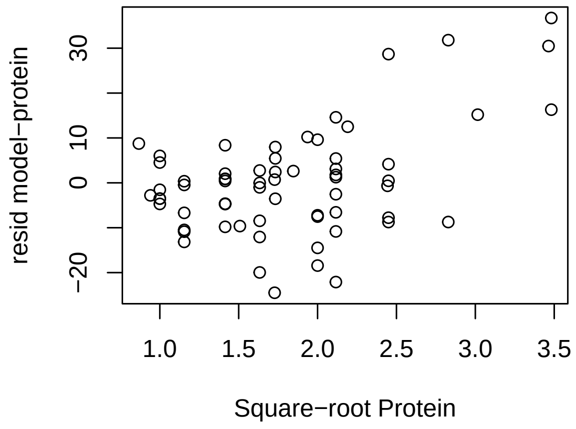
Figure 9: Cereal Partial Regression Plots



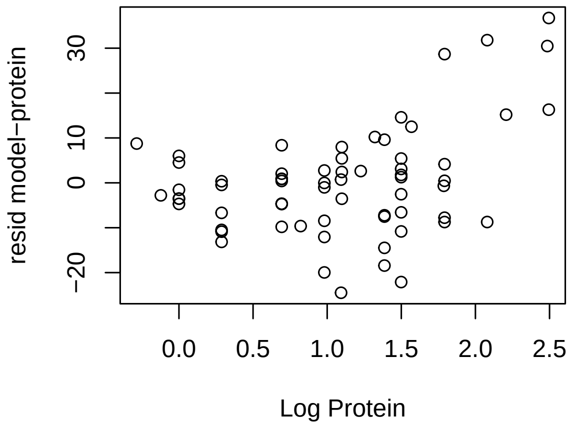
(a) Inverse Protein partial regression plot



(b) Protein squared partial regression plot



(c) SQRT Protein regression plot



(d) Log Protein partial regression plot

Figure 10: Transformed Protein Partial Regression Plots

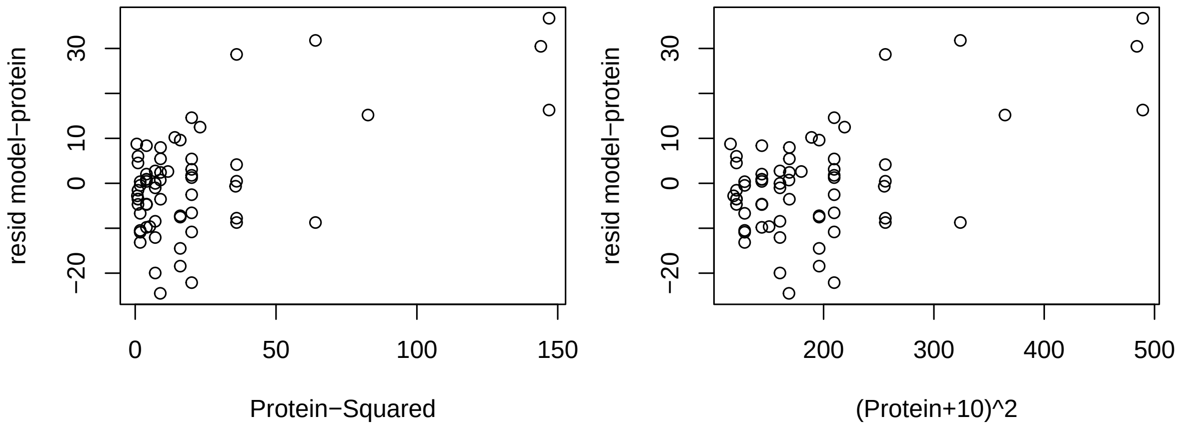
Consider for any constant c:

$$y = \hat{\beta}_0 + \hat{\beta}_1(x_1 + c) + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \varepsilon$$

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_1 c + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \varepsilon$$

$$\widehat{\beta_{0new}} + \hat{\beta}_0 + \hat{\beta}_1 c$$

$$y = \widehat{\beta_{0new}} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 + \varepsilon$$



(a) Protein Squared

(b) (Protein+10) Squared

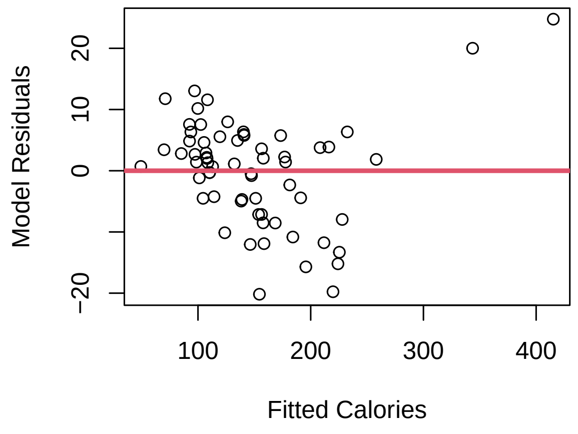
Figure 11: Transformed Protein Partial Regression Plots

From here, we know that $protein^2$ is a better linear predictor for calories than $protein$ directly so we change our model from

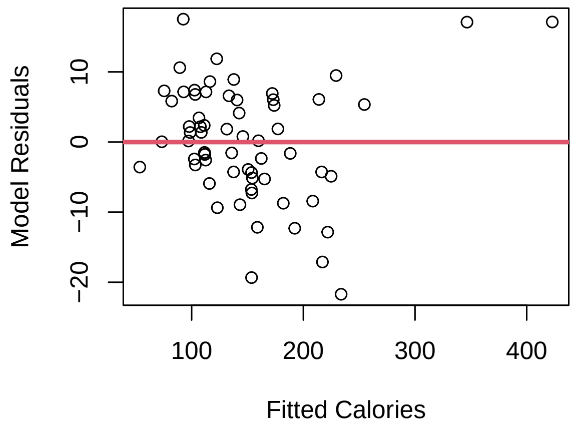
$$calories = \beta_0 + \beta_1 protein + \beta_2 fat + \beta_3 carbo + \beta_4 sugars + \varepsilon$$

to

$$calories = \beta_0 + \beta_1 protein^2 + \beta_2 fat + \beta_3 carbo + \beta_4 sugars + \varepsilon$$



(a) Original



(b) Transformed protein

Figure 12: Cereal Calories fitted vs. residuals, protein squared

Multicolinearity

A good first step is to look at the simple pairwise relationships between each x predictor and the outcome y. Let's expand beyond our two-x hill race model and look at a larger data set with multiple predictors. Figure 13 shows data from the `road` dataset contained in the `MASS` library.

From first glance, the scatterplot matrix in Figure 13 makes clear that there is an extreme outlier in our data in the population density domain. With a closer look at the data we can see that while most rows of `road` are US states as explained in the help menu for the data frame, row 9 is actually not a state at all; it is Washington DC. This explains why row 9 has a population density of over 12,000 while the next highest is only 655 (for Massachusetts).

```
road[order(road$popden, decreasing=TRUE)[1:5],]
```

```
      deaths drivers popden rural temp fuel
DC       115      35  12524   0.0   44   23
Mass     766     255    655  17.0   37  166
Conn     325     167    518   5.1   37   95
Maryl    616     157    314  29.0   44  113
Dela     118      30    226   3.4   41   20
```

To exclude the district from our analysis, step one is to create a new `road2` data frame that omits Washington DC. Then, we'll examine a scatterplot matrix for the new data.

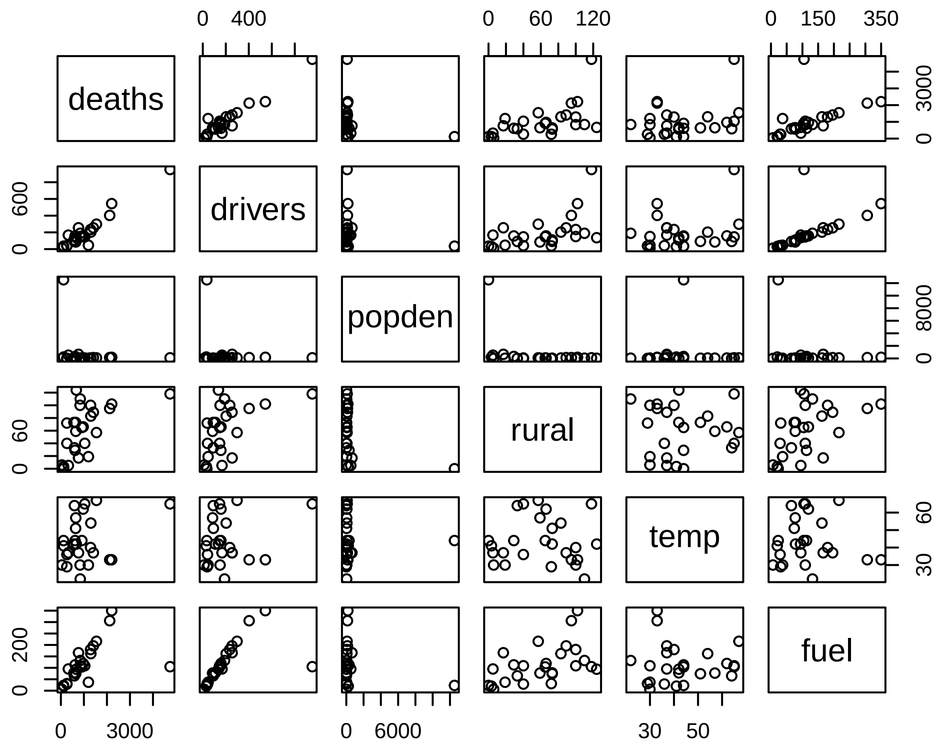


Figure 13: US Driver Deaths

```r
road2<-road[road$popden<1000,]
plot(road2)
```

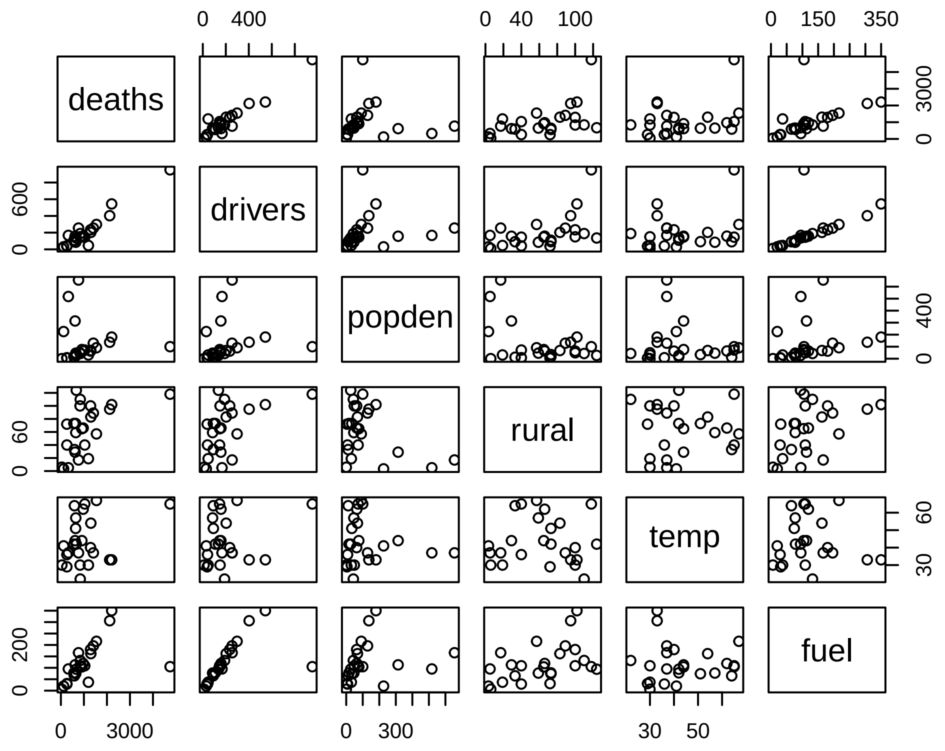


Figure 14: US Driver Deaths, without DC

From Figure 14 we can see that every possible predictor for road deaths has a definite relationship with the deaths column of our data set. Some are stronger than others, but drivers, popden, rural, temp, and fuel all appear to be at least somewhat linearly related to deaths.

Inference

The same methods and R commands used for inference earlier continue working in multiple dimensions as well. `confint` will give confidence intervals for the s and `predict.lm` will give point estimates, confidence intervals, and prediction intervals for particular x values of interest.

Interactions