Comparing 2 Tools for Linear Regression

2022-11-20
Jayson Webb PhD

Overview

The purpose of this data experiment is to compare the experience and outcomes of creating a linear regression model with 2 different tools:

- R `lm` library. R is an open-source software environment for statistical computing and graphics. The `lm` library is used to fit linear models to data frames in the R Language.
- H20 makes it easy to create and deploy machine learning models including linear regression. I used the free version running locally on my HP Dev One laptop for this project.

I undertook this project to review and sharpen my own skills with regression but also to learn more about `h20`. I was already fairly familar with using `lm` in the `R` environment. I also hope that people might find some part of this useful or instructive, including having some code for a simulated data set (github repository). A secondary puropse of this project was to use Obsidian to write this up via markdown. Also a fun learning experience.

Insights

I wanted to capture some of the high level insights I gained from doing this project.

- The independent variables with larger range and variance were more important in the models.
- You shouldn't expect to be able to discover theoretical relationships among variables from a regression equation. You might, but you can't count on it.
- Interpretability of a model is an illusion for practical applications, according to the bullet point above. If your model does a good job of predicting data is hasn't seen before, you have captured something important, even if the model parameters don't reflect the underlying "physics" of the situation.
- I will invest more time learning H2O. After some initial getting used to the workflow, it is very quick, easy and intuitive to create models that perform well (at least for my purposes, which is mostly recreational data science).

- AutoML tools like those provided by H2O are very useful and easy to use and will increasingly become a part of my toolkit. `R` also has automl libraries like `caret` which is also available in Python as `pycaret`.

Next Steps

- Simulate more data. How would these approaches work with thousands or millions of data points?
- Error variance will play a larger role. How will these approaches be effected by a larger contribution of the random variate ϵ.

Simulated Data

The rationale for using simulated data was to have a set of known data properties so that their effects in the linear regression model could be predicted. Having well understood data helps with comparing output across models produced by different tools. Even though it does afford predictability, the relationships among the variables are complex and contain non-linearities, which makes it a challenge for linear regression.

The simulated regression data contains three independent variables (*a*,*b*,*c*) and one dependent variable (*output*). The formula for creating outputs from the three independent variables is shown in Equation 1. Two data sets were created: one for creating the original models and the other for testing the models.

Independent Variables

The values for the independent variables *a*, *b*, and *c* were selected randomly from integers with different ranges but the same population mean μ = 50. The ranges and the actual means and standard deviations are shown in the table below for one set of 100 simulated rows of data. This is the set that we used to create the regression models.

Table 1. Independent variables in the main data set

Variable	Range	Mean	sd
a	20 - 80	47.4	17.58
b	10 - 90	50.9	23.85
c	30 - 70	50.4	11.79

One interesting learning from this project is that the importance of the variables in the models I created was directly related to the range and sd, with larger values of both being associated with more variable importance.

Dependent Variable

The dependent variable *output* ($\bar{x}$ = 23.8, $\text{sd} =$ 2.79, range = [15.5,28.8]) was constructed from the independent variables *a*, *b*, and *c* and ϵ, a random variate in the range (0, 1), as follows. In the equation, log refers to log_{10}.

Equation 1. Dependent Variable Formula

$$output = -10 * log[\frac{log(b) + \epsilon_1}{a^2} + \frac{log(c) + \epsilon_2}{b^2} + \frac{log(a) + \epsilon_3}{c^2}]$$

The subscripts on $epsilon$ in the formula indicate that this was a unique random variable for each independent variable value.

What follows is a numerical example showing the calculation. We will just use $\epsilon = 0.5$, so we will only get an answer that approximates what is in the actual data, since those ϵ values were randomly generated.

In our simulation, the following values for *a,b,c* (79, 27, 42) generated an *output* value = 23.3 (we rounded the output values to 1 decimal point, thus adding a little more noise). Plugging the *a,b,c* values into Equation 1 above would look like the following.

Equation 2. Example Calculation

$$-10 * log[\frac{log(27) + 0.5}{79^2} + \frac{log(42) + 0.5}{27^2} + \frac{log(79) + 0.5}{42^2}] = 23.39$$

Considering that we used a fixed rather than a random ϵ, that's pretty close to the value generated by the simulation (23.39 vs 23.3).

The figure below shows the relationships among the dependent and independent variables. At first glance, it appears that *b* (B in the diagram) has the strongest linear relationship with *output* and that the independent variables are fairly uncorrelated.

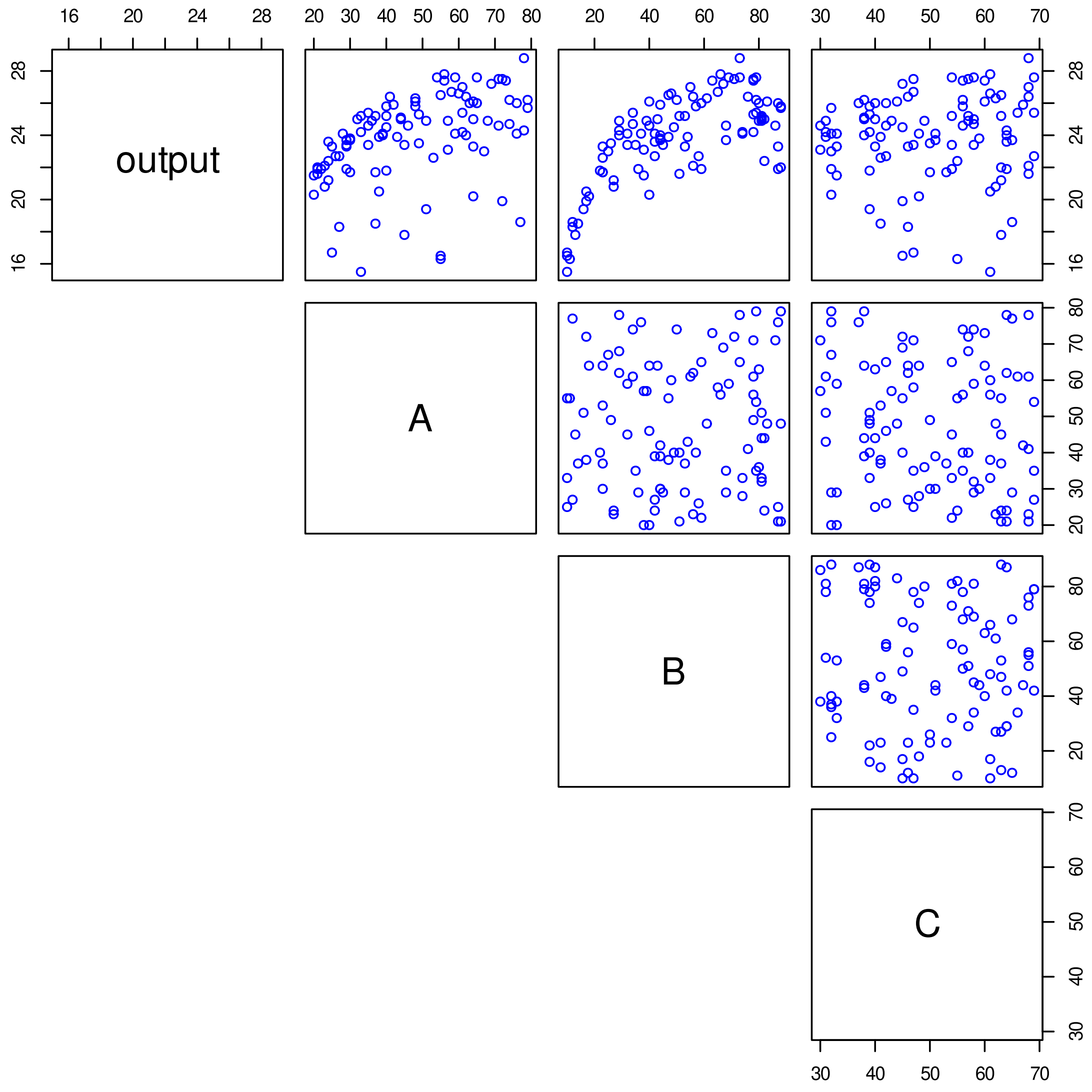

Figure 1 - Relationship among the independent (shown as capital letters) and dependent variables

Predicted Effects

The regression **main effects** should be positive for each independent variable because increasing the value of any of the independent variables should increase the value of output variable, all else being equal. The main effects are the denominators of the fractions in Equation 1. When you take the log of a fraction (note, we're taking the log in the numerator, but also the log of the sum of three fractions), the absolute value gets larger as the denominator gets larger. For example, $\log_{10}(1/10) = -1$, $\log_{10}(1/100) = -2$, $\log_{10}(1/1000) = -3$. We turn negative values into positive values by multiplying by -10 in Equation 1.

The two-way **interaction effects** should each be negative. As Equation 1. shows, variable *b* supresses the main effect of *a*, *c* supresses the main effect of *b*, and *a* suppresses *c*. Using the same logic as for the main effect, when we make the fraction larger by increasing the numerator, the absolute value of the log gets smaller.

As we explore regression models, we should see the pattern described above with positive main effects and negative two-way interactions.

R Regression

I used the `lm` library in `R` to create a model with the following structure:

$$output = a + b + c + a * b + a * c + b * c + a * b * c$$

I included all main effects, two-way interactions and the three-way interaction. Below is the summary output for the `lm` model.

Regression Output 1: The original model.

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.069e+01  5.009e+00   2.134 0.035505 *
A            2.148e-01  9.330e-02   2.302 0.023569 *
B            2.346e-01  8.953e-02   2.621 0.010268 *
C            1.507e-01  9.457e-02   1.594 0.114445
A:B         -4.479e-03  1.593e-03  -2.812 0.006012 **
A:C         -3.769e-03  1.765e-03  -2.135 0.035423 *
B:C         -3.752e-03  1.682e-03  -2.230 0.028166 *
A:B:C        1.068e-04  3.057e-05   3.493 0.000735 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.478 on 92 degrees of freedom
Multiple R-squared:  0.7386,    Adjusted R-squared:  0.7187
F-statistic: 37.14 on 7 and 92 DF,  p-value: < 2.2e-16
```

We see that the main effects are positive and the two-way interactions are negative, as we predicted. We'll talk about what the three-way interaction means in the next section on interpreting the model. Later, we'll try to improve the model to do better than an adjusted $R^2 = .72$. But first, let's talk about interpreting this model.

Interpreting the model

One interesting thing about working with data that you designed and has known properties is to see what the regression model will reflect back to you. What do the coefficients for the intercept, the main effects and the interactions mean? Do they **tell us anything about the underlying "reality" of the data?** As already mentioned, one thing the model gets right is the sign of the main effects and interactions. For everything else, the model doesn't really reflect reality. Equation 1 involves logs, ratios, and squares. But as we'll see later, it can give us reasonable predictions with transformations to the data. To discuss interpretability here, we have to make **some assumptions**. First, let's assume that we don't have a theory of the underlying variables. That is, we're not dealing with something like IQ, height, or income where we might have theory and world experience to tell us what to expect about how the independent variables lead to the dependent variable. Rather, we're dealing with a black box that produces numbers when we turn the three knobs a, b, and c to different settings. Inspecting the data suggests that **the three-way interaction is signficant because of the over corrections due to the two-way interactions**. This gives us decent predictions, but doesn't reflect the underlying reality that the main effects are proportional to the squared independent variable values and we wouldn't discover that the interactions are proportional to the log_{10} of one of the variables times the squared reciprocal of the other. Can we even trust the fact that the signs of the main effects and interactions point in the right direction? In a scenario where we didn't create the data, we wouldn't know this. But, I think we have to put some kind of stake in the ground and I think trusting the signs of the effects based on the raw data is something we can hang a hat on. One thing we'll want to watch out for as we try to improve the predictions is beware any transformation of the data that causes the sign of any of the effects to change.

Getting better predictions

Centering and scaling

A standard thing to do to make a regression equation more interpretable and to possibly improve the fit with the data is to center the data about the mean (i.e. subtracting the mean from each value resulting in a mean of 0) and sometimes to give each variable a standard deviation of 1. I tried both of those things on this data and all of the effects and interactions were positive. That is, the two-way interactions were no longer negative. I decided above to reject any data transformations that changed the signs of the original main effects and interactions, so I didn't use this approach.

Adding quadratic components

Figure 2 shows the predicted vs actual data for the original regression.

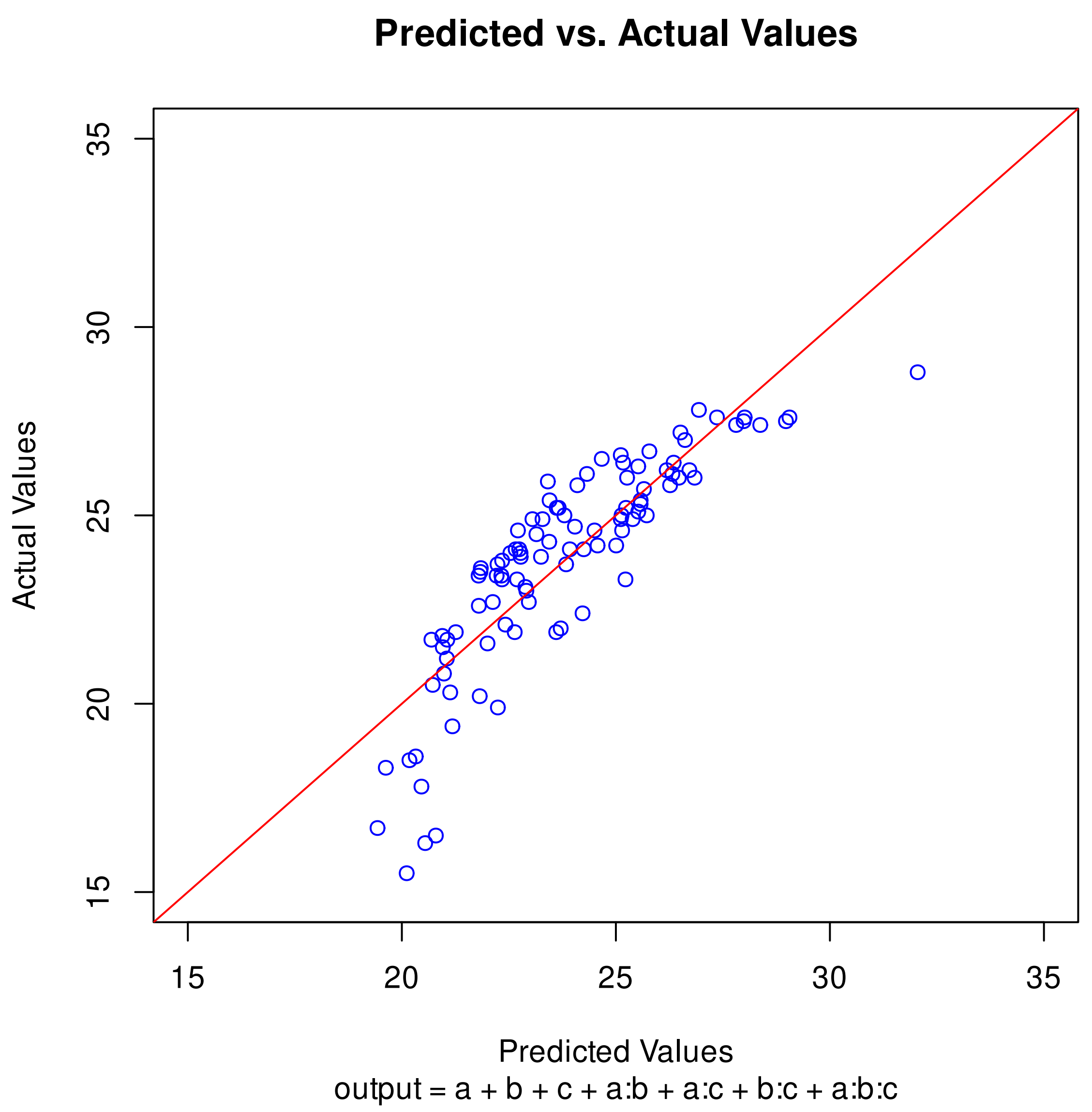

Figure 2 - Predicted vs actual values for the original regression in R

In the original regression, shown above, the predictions seem ok in the middle of the data range, but then overpredict on either end. This pattern can indicate that a quadratic component needs to be added to the data. So, I added independent variables to the data that were the squares of the original independent variables and called them *a2*, *b2* and *c2*. I also added the interactions involving the squared variables. The resulting regression table is shown below.

Regression Output 2: Adding quadratic components.

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.602e+00  1.049e+01  -0.344 0.732052
a2          -2.843e-03  1.012e-03  -2.808 0.006179 **
```

```
b2          -2.357e-03  6.700e-04  -3.517 0.000703 ***
c2          -1.617e-03  1.186e-03  -1.363 0.176543
A            6.391e-01  2.585e-01   2.472 0.015425 *
B            4.905e-01  2.209e-01   2.221 0.029033 *
C            4.181e-01  2.606e-01   1.604 0.112433
A:B         -8.009e-03  4.320e-03  -1.854 0.067239 .
A:C         -1.109e-02  5.133e-03  -2.160 0.033586 *
B:C         -5.549e-03  4.318e-03  -1.285 0.202293
a2:b2        3.264e-07  2.049e-07   1.593 0.114845
a2:c2        7.419e-07  3.552e-07   2.089 0.039703 *
b2:c2        1.384e-07  2.342e-07   0.591 0.556172
A:B:C        2.099e-04  8.555e-05   2.454 0.016184 *
a2:b2:c2    -1.964e-10  7.795e-11  -2.519 0.013630 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7559 on 85 degrees of freedom
Multiple R-squared:  0.9368,    Adjusted R-squared:  0.9264
F-statistic: 90.01 on 14 and 85 DF,  p-value: < 2.2e-16
```

To my surprise, the model now accounts for almost 93% of the variance in the *output* variable - MUCH better than the 72% we started with. It does seem odd that not all of the variables are significant. I looked at a model with just the quadratic components and it only accounted for about 53% of the variance in the *output* variable. It also seems odd that the signs for the quadratic components are reversed compared to the corresponding linear components. Below, we'll take the log of all the variables.

Taking the log

I have found the taking the log of data often improves the fit of a model. I think it helps tame any remaining non-linearities. So, I **took the log of the independent and dependent variables** (adding 1 to each value first in case there were any values < 1). Things improved yet again as shown in the regression output below.

Regression Output 3: Taking the log

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -88.060     28.969  -3.040  0.00315 **
a2          -136.694     42.971  -3.181  0.00205 **
b2          -132.887     39.840  -3.336  0.00126 **
```

```
c2          -133.860     42.603  -3.142  0.00231 **
A            295.451     92.871   3.181  0.00205 **
B            287.459     86.274   3.332  0.00128 **
C            289.485     92.102   3.143  0.00230 **
A:B          -74.320     22.240  -3.342  0.00124 **
A:C          -74.735     23.659  -3.159  0.00219 **
B:C          -72.678     21.970  -3.308  0.00138 **
a2:b2         17.281      5.164   3.346  0.00122 **
a2:c2         17.381      5.502   3.159  0.00219 **
b2:c2         16.887      5.098   3.312  0.00136 **
A:B:C         18.321      5.665   3.234  0.00174 **
a2:b2:c2      -2.137      0.661  -3.233  0.00174 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.01825 on 85 degrees of freedom
Multiple R-squared:  0.9803,    Adjusted R-squared:  0.9771
F-statistic: 302.5 on 14 and 85 DF,  p-value: < 2.2e-16
```

We now account for almost 98% of the variance, and all of our effects are significant, but we still see the pattern that quadratic components have signs opposite those of the linear counterparts. We also have two significant three way interactions now. If our coefficients were unintrepetable before, they are even more so now, but we have set our sights on prediction and we seem to have done a pretty good job of that. I also examined a model without the quadratic components, just taking the log of all variables and including all main effects and interactions. That one accounted for about 90% of the variance, but all the main effect and interaction coefficients were negative, which is undesirable as dicussed elsewhere in this paper.

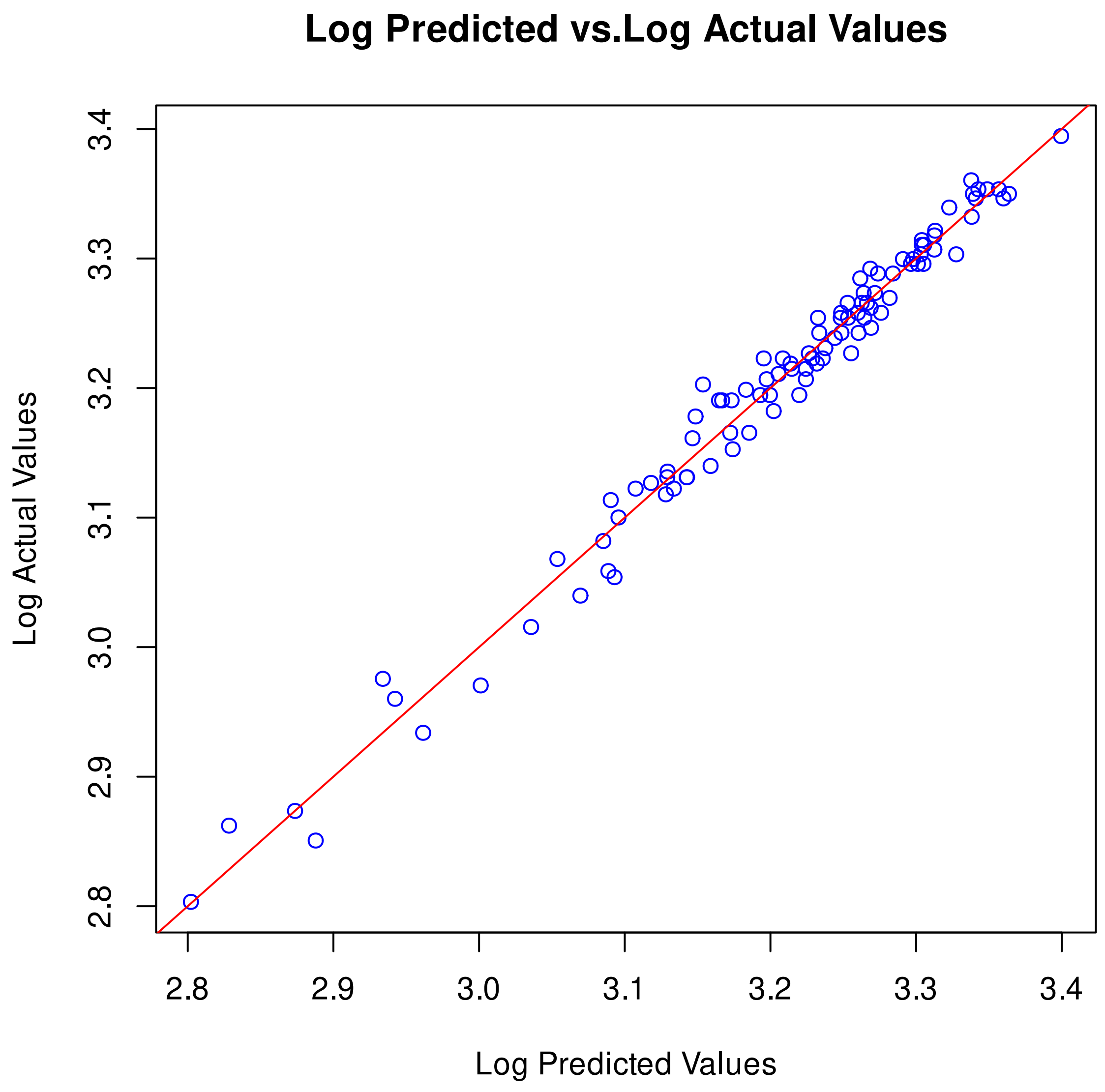

Figure 3 - Log Predicted vs log actual values for the regression equation shown in Regression Output 3 above, which includes linear and quadratic components for main effects and interactions.

Predicting with new data

To get our high level of prediction, did we commit the sin of overfitting? Let's test our equation with a new set of data to check.

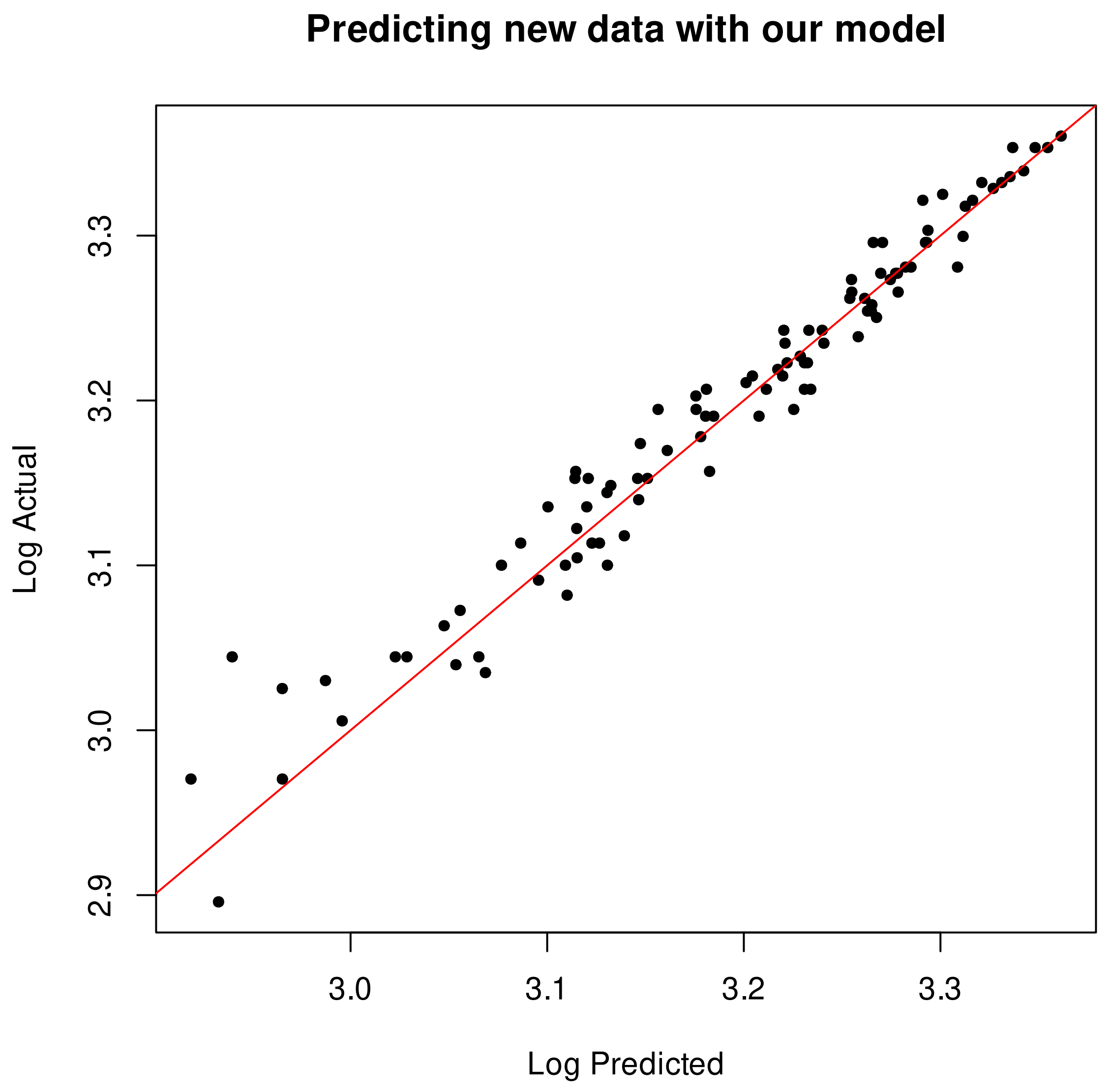

Figure 4 - Predicting new data with our model

Using our equation with new data, our model accounts for about 96% of the variance in *output* ($R^2 = .9584$), although we are a bit weak predicting lower values for *output*. When the log data is translated back to the original scale, our fit is actually a little better ($R^2 = .9628$).

Summary of R linear regression

We have a model that does a good job of predicting new data (data that wasn't used to develop the model). But, we don't have the ability to interpret our model. That is, we don't have an understanding of the "physics" of how our independent variables produce our dependent variable. We did "discover" that the main effects are positive and the two-way interactions are negative, or "inhibitory", as they were designed to be. However, the model also suggests there is a three-way interaction which in practical terms is useful to compensate for the over-correction of the two-way interactions in the model, but wasn't part of the original data design. So, in the end, we can't really trust the model in terms of being a true reflection of the underlying dynamics among the variables. Given our imaginary scenario (there is no theory, this is just a machine that produces numbers, and our interest is in prediction), that's ok. We didn't come in with any theory of the relationship of inputs to outputs. But one lesson to take from this is that **you shouldn't expect to be able to discover theoretical relationships from a regression equation.**

Even though we can't interpret our model, it's not a black box in that we know what transformations were made to the data, adding quadratic components and taking the log of all variables. But, did we have any theoretical ground to stand on to do those transformations? Or, were we just trying to get more variance accounted for? Well, to be honest, it was the latter. So, in that case, not being a "black box" is an illusion, if I'm being honest with myself. Also, was linear regression even the right approach? Maybe not. One advantage of the AutoML approach, one example of which we'll see next, is that many different model types are evaluated, and model ensembles are created, that attempt to solve the problem of predicting continuous variables, which is really what this exercise is about, even though I titled it "Linear Regression".

H20 AutoML

I decided to use the AutoML feature of H20 to create models using the main data set (regdata.csv) and predict the values in the holdout data set (reg2data.csv). After getting it installed on my HP Dev One Linux machine and doing some initial exploration, I was ready to go. H2O has a browser-based environment that guides you through workflows, from loading and parsing data to creating, running and evaluating models. I used that environment for this project.

One of the parameters you can set in AutoML limits how many models are generated and how long they will run. Without constraints, the AutoML ran for almost an hour on this small data. It did a great job, but for these purposes, that's too much time. The data reported below was based on a settings that limited the number of models investigated and ran for about 5 minutes.

XGBoost for the win

The model I chose was an XGBoost model. H2O lets you inspect all of the parameters of the model, which I won't go through here. Instead, I'll skip to the prediction of the holdout data.

The model accounted for about 94% of the variance in the holdout data - similar to the 95% that our `lm` model predicted. When I let the training run for an hour (rather than just

5 minutes) those models accounted for over 98% of the variance. The *mae* (mean absolute error) was .47, which is not automatically reported by `lm`. H2O lets you easily combine the predictions with the original data and export them so you can plot them in any tool you like.

I really liked using H2O for this task and will continue to invest time in understanding it. Among the positives and negatives:

Positives

- I didn't have to do any data transformations. Just submitted the data right to AutoML. This data was already quite "clean", so it will be interesting to see how it does with messier data.
- Produced better predictions on the holdout data than the `lm` approach.
- Produced more model diagnostics, like *mae*.
- AutoML automatically evaluates dozens(?) of models (depending on the limiting settings mentioned above) and even creates ensembles (which didn't perform as well on this data as the individual models). Makes me less likely to be on shaky theoretical ground in my approach.
- It's easy to select among just the best of the models evaluated.
- Easy to export the models and reuse them later, and probably to share with others.

Negatives

- The models produced are a bit of a black box, but as I said in the introductory insights, I think that's an illusion to think the more "hand crafted" models are not. The regression equation I created in `R` didn't really capture the underlying "physics" of the data, even though it gave some illusion of understanding.