RegViz

STA3100 Final Project

Matthew Devaney

2024-04-30

Contents

1 Abstract

A linear model is a statistical concept which aims to represent a linear relationship in a given data set. I created an application, RegViz, in order to allow a user to dynamically interact with linear models. This application includes various model statistics and graphics and most importantly aims to be a user-friendly approach to linear regression in R.

2 Introduction

A linear model is a concept that attempts to model data which follow a linear relationship. Specifically, it models the relationship between a response, $\hat{y}$, and any given amount of predictor variables. Combining these two pieces together, we arrive at the following equation:

$$\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} + \epsilon_i \tag{1}$$

$$\text{for each predictor} : i = 1, \ldots, n$$

And a practical example may look like:

$$\mathbf{mpg} = 35.846 - 3.35902\mathbf{cyl6} - 3.18588\mathbf{cyl8} - 0.02312\mathbf{hp} - 3.1814\mathbf{wt}$$

Figure 1: Created with RegViz

As you can see, this equation follows a linear relationship between the response and the predictors. $\hat{y}_i$ should take on a range of continuous values, while each x_i may be either continous or categorical. Additionally, the term ϵ_i must be added to account for error in the model, as it is not possible to account for every single source of variance in the data. The goal is to minimize this error as much as possible, in order to increase the model's ability to represent the relationship in the data. This can be done in multiple ways, one of them being changing around which predictor variables are included in equation (1). In doing so, some interesting and valuable patterns can start to emerge.

This can typically be done using a programming tool such as **R** (R Core Team 2023). However, it can be difficult to quickly see the changes between models with different parameters using such a tool. A typical linear regression model in R code looks as follows:

```r
# Load in a data set
data <- mtcars
rownames(data) = NULL
# Check correlation between independent variables
cor(data)
# Convert categorical variables to factors
data$cyl <- as.factor(data$cyl)
# Choose parameters for the model and create it
model <- lm(mpg ~ hp + wt + cyl, data = data)
# View summary of the model to assess
summary(model)
```

All of that and not a single graphic or visualization for us to look at! And when we inevitably want to experiment with other models or parameters, we must use the same [`lm`, `summary`] paradigm over and over again.

That is why I created an application, RegViz, with the help of Shiny (Chang et al. 2024), to provide an easy-to-use, interactive, and visually appealing way for users to experiment with linear models. No longer do you need to worry about annoying R code, instead you can simply upload a formatted data set and use a series of check boxes, drop downs, and buttons to view all kinds of information about your model.

3 Application

The process of dealing with linear models should generally follow these steps:

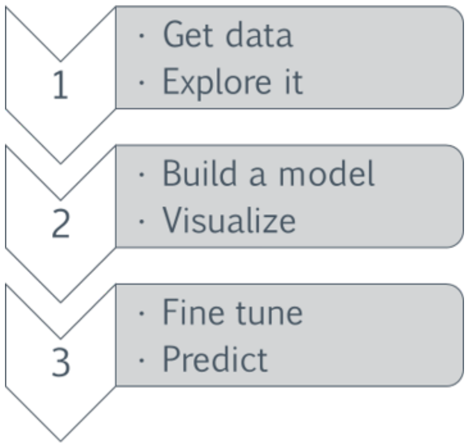


Figure 2: Steps of creating a linear model

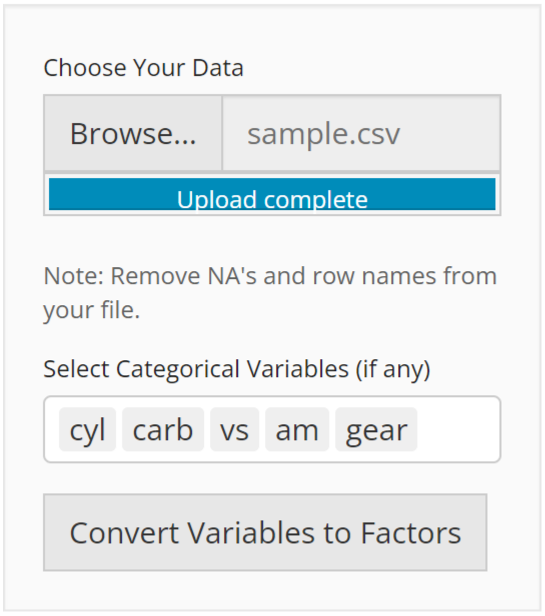
We'll explore how this application accomplishes these tasks in a user-friendly manner in this section.

3.1 Getting Data

While the app cannot choose data for the user (that's their job!), the app will make it easy to upload a csv-formatted file. After that, the user can choose which variables should be treated as categorical and therefore be converted into factors. This is a very important step because factors are how R knows to assign dummy variables to each category when performing linear regression.

3.2 Exploring Data

There are a few important assumptions to consider when performing a linear regression analysis. One of which is the idea of multicollinearity, where 2 or more of the predictors in the model are highly correlated. When this occurs, it makes it difficult to determine the effect of each individual predictor, and this can make the model less reliable. Thankfully, there are some tools to determine whether or not multicollinearity is occuring between predictors. Some of these are the correlation matrix and the variation inflation factor, or VIF (Team, Potters, and Li 2023), pictured in figure (4). In the correlation matrix, the strength of the correlation corresponds to the size of the square while the sign corresponds to the color. A VIF score > 10 indicates that siginificant multicollinearity is occuring between the predictors.



Choose Your Data

Browse... sample.csv

Upload complete

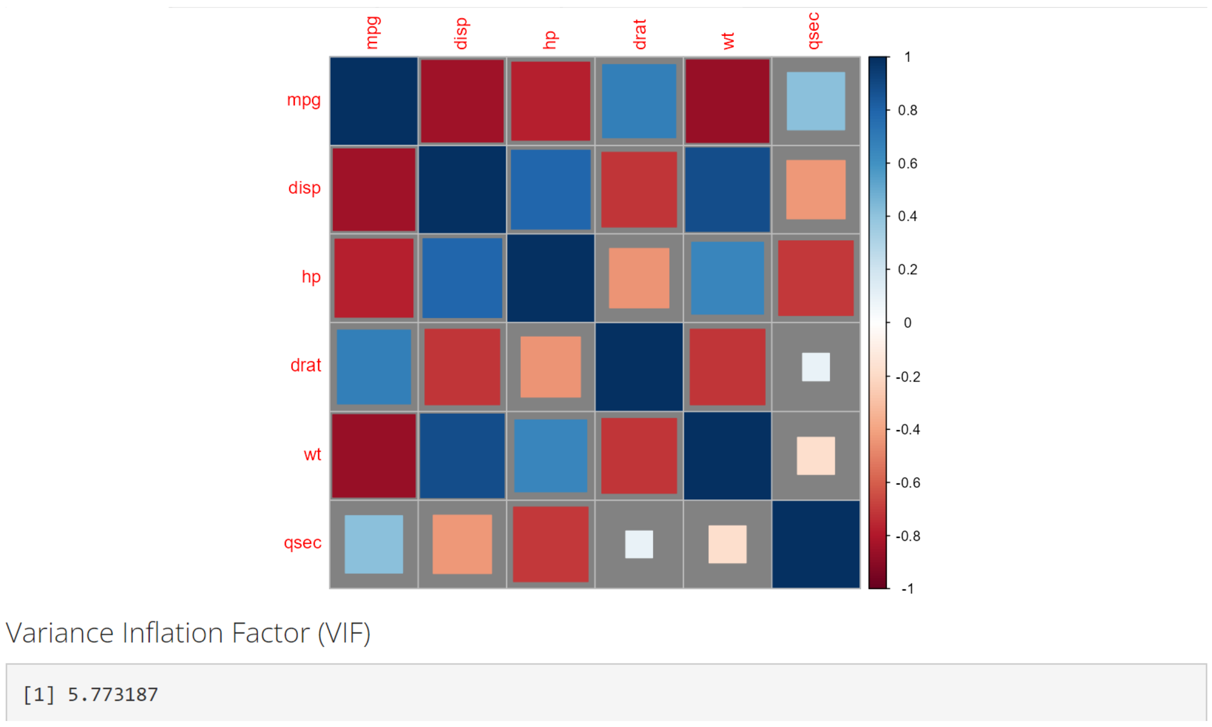
Note: Remove NA's and row names from your file.

Select Categorical Variables (if any)

cyl carb vs am gear

Convert Variables to Factors

Figure 3: Choosing data and factors



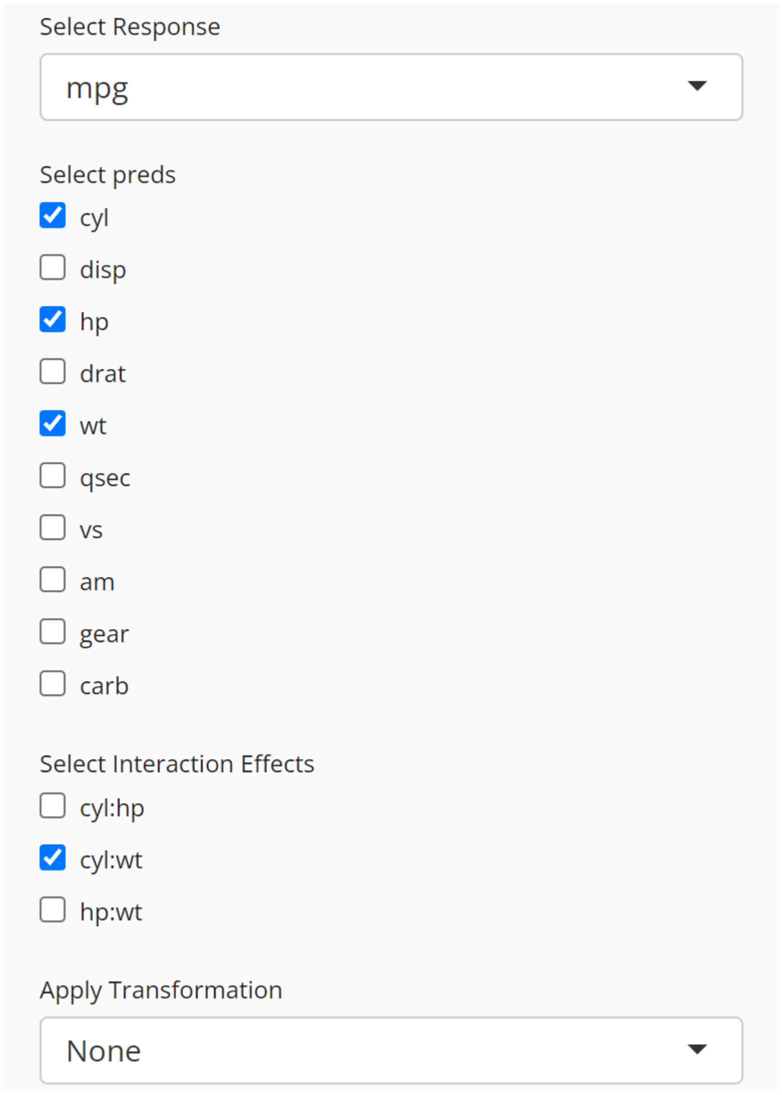
Variance Inflation Factor (VIF)

```
[1] 5.773187
```

Figure 4: Viewing multicollinearity

3.3 Building the Model

At this point, the data is loaded and formatted, and any signs of multicollinearity have been checked. Now, the model can be created. This can easily be done by selecting a response variable along with one or multiple predictors as shown in figure (5). In addition, the user can select certain interaction effects to be included in the model to get a different result. Not pictured are additional model outputs such as the summary, which **dynamically** updates as the user selects different parameters along with all of the other content.



Select Response

mpg

Select preds

- [✓] cyl
- [ ] disp
- [✓] hp
- [ ] drat
- [✓] wt
- [ ] qsec
- [ ] vs
- [ ] am
- [ ] gear
- [ ] carb

Select Interaction Effects

- [ ] cyl:hp
- [✓] cyl:wt
- [ ] hp:wt

Apply Transformation

None

Figure 5: Choosing model parameters

3.4 Visualize

A pitfall of traditional R code is that while making one graphic is easy, making multiple is not as quick. This application automatically generates multiple informative graphics to help the user get a picture of their model quickly. Below are some of the graphs that can be generated. Figure (6) (Plotly) demonstrates the very important model statistic, residuals. A residual is given by $e_i = y_i - \hat{y}_i$ and represents the difference between the actual data value and the value predicted by the model. Ideally, the data points would be randomly distributed and close to the line $x = 0$ for the best model fit. The second figure is a plot of added variable plots, which allows us to graphically see the effect of a single parameter in a multiple linear regression model.

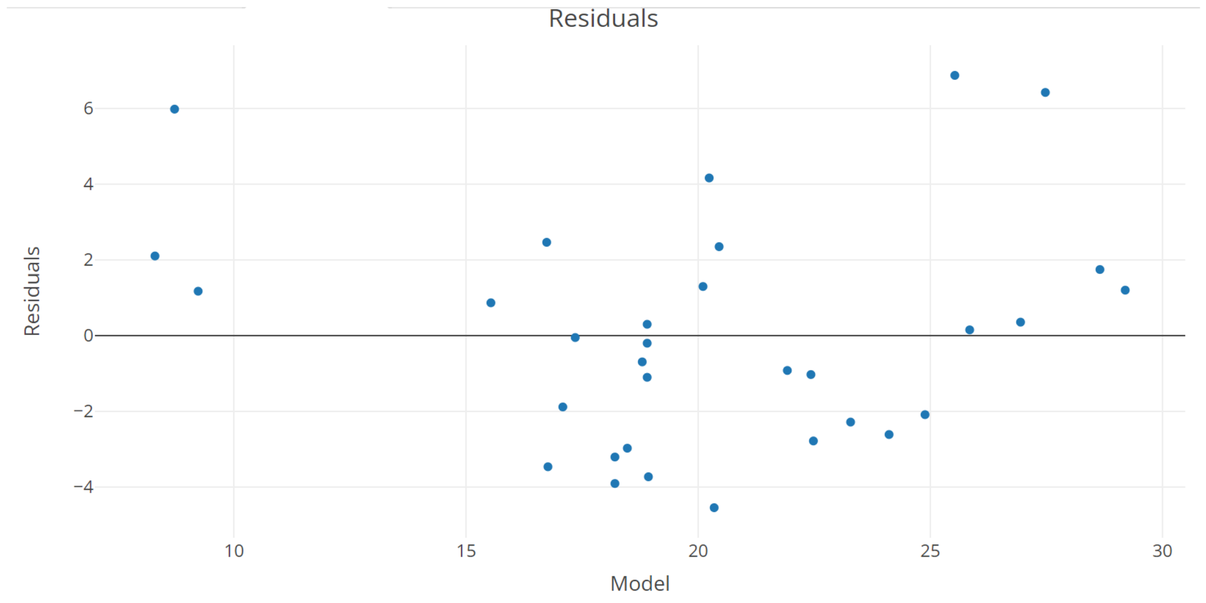


Figure 6: Residuals of a linear regression model

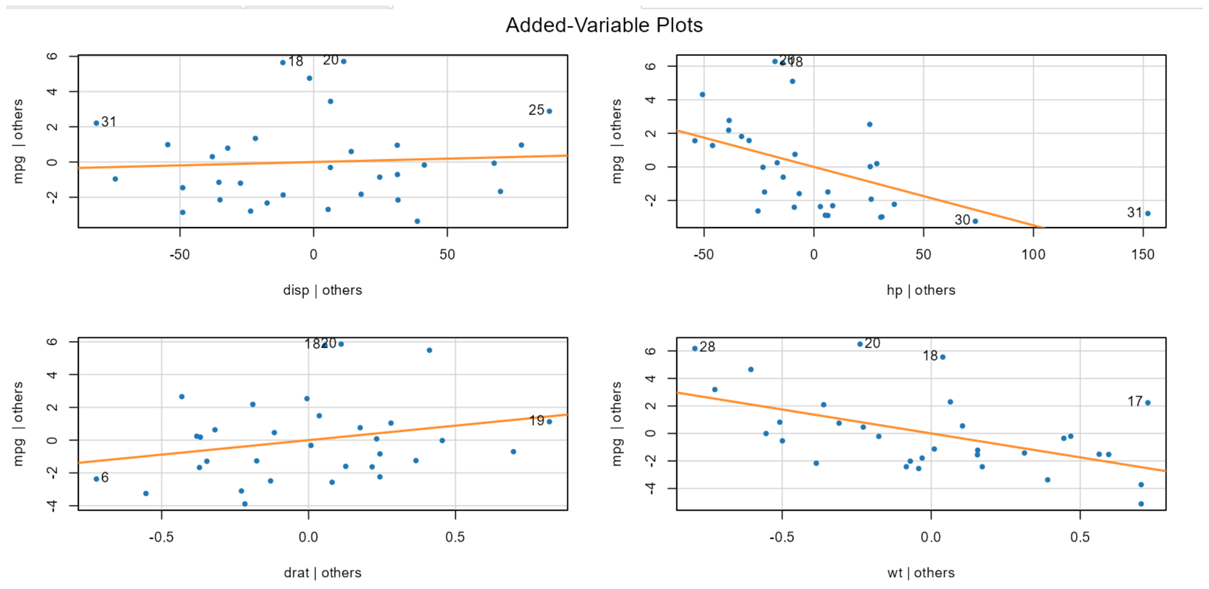
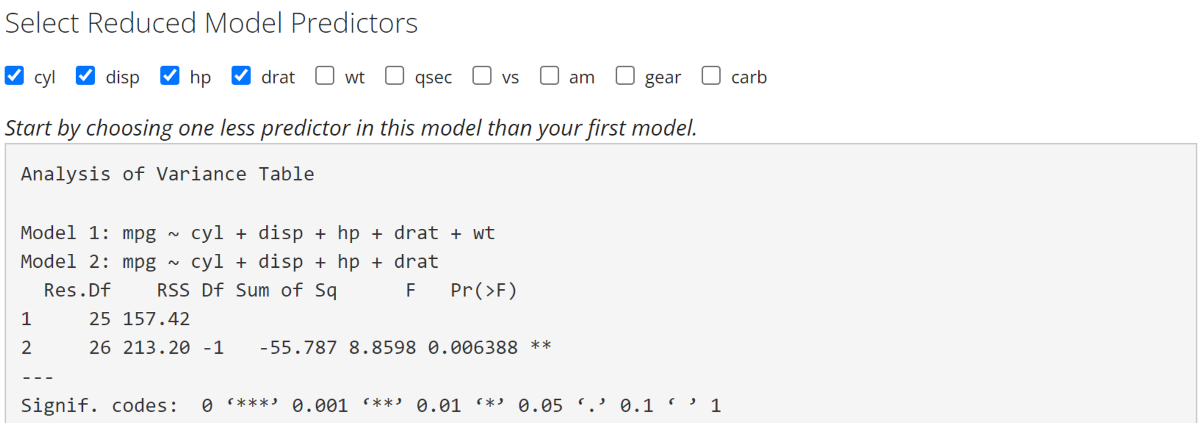


Figure 7: Four added variable plots

3.5 Fine Tuning

After creating a model, one might ask, can we do better? Thankfully, there are some techniques we can use to determine superior model. All of these techniques revolve around changing the parameters of the model and this application makes it easy to do that. One example of model optimization is the partial F test, which essentially determines the importance of a predictor by comparing a model containing that predictor and another model without it. Also, as seen in figure (3), the user has the ability to perform a nonlinear transformation (e.g. square root) on the predictors in order to improve the model fit if the relationship in the data isn't exactly linear.


Select Reduced Model Predictors
cyl [✓] disp [✓] hp [✓] drat [✓] wt qsec vs am gear carb

Start by choosing one less predictor in this model than your first model.

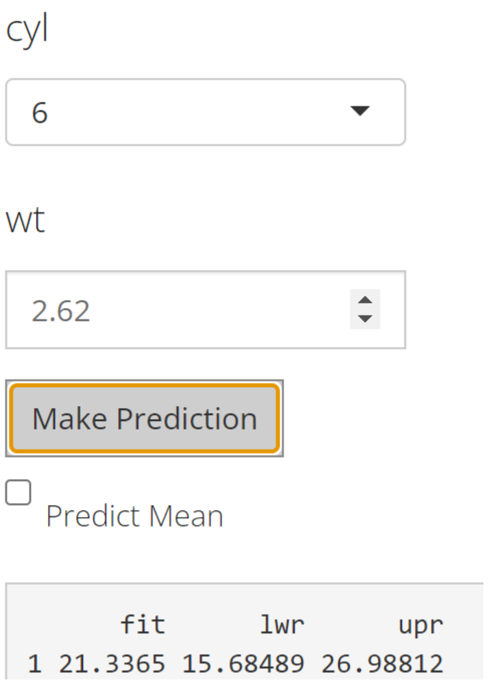
```
Analysis of Variance Table

Model 1: mpg ~ cyl + disp + hp + drat + wt
Model 2: mpg ~ cyl + disp + hp + drat
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     25 157.42
2     26 213.20 -1   -55.787 8.8598 0.006388 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 8: Results of a partial F test

3.6 Predict

Finally, the model should be able to be used to predict new data values! In traditional R code, having to extract all the coefficients from your model and then pair each one up with its respective predictor value is a pain. With RegViz, you can seamlessly input the value of each predictor and create a predicition using your model.



cyl: 6

wt: 2.62

```
      fit      lwr      upr
1 21.3365 15.68489 26.98812
```

Figure 9: Predicting the response with 3 predictors

4 Conclusion

This application aims to illustrate results from linear regression models in a user-friendly manner. It includes various model statistics, graphics, and, most importantly, allows the user to dynamically change the model to however they see *fit*. In the future, I would like to add additional features to this application such as polynomial regression, additional tool tips, and interactivity with data outlier detection and removal. Overall, I hope that this application could be beneficial to someone trying to understand how linear regression models work and how they change.

5 Works Cited

R Core Team (2023) Posit team (2023) Chang et al. (2024) Team, Potters, and Li (2023) Plotly

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2024. *Shiny: Web Application Framework for r.* https://CRAN.R-project.org/package=shiny.

Plotly. "Plotly — Plotly.com." https://plotly.com/r/.

Posit team. 2023. *RStudio: Integrated Development Environment for r.* Boston, MA: Posit Software, PBC. http://www.posit.co/.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Team, The Investopedia, Charles Potters, and Timothy Li. 2023. "Variance Inflation Factor (VIF) — Investopedia.com." https://www.investopedia.com/terms/v/variance-inflation-factor.asp.