---

title: "2D Regression Visualization Final Draft"

author: "Riya Makhijani, Ruchita Patel, Keenan M. Reimer, Rowan Kardos"

date: "June 3, 2016"

output: html\_document

---

###Introduction:

The goal of our project is to create an interactive app that allows users to visualize regression plots using example or custom data sets. The app provides the user with the option to choose a csv file from their computer and upload it on to the app. If they choose not to utilize this feature, there are three included datasets: the “diamonds” dataset from ggplot2 , “mpg” dataset from ggplot2, and the “iris” dataset from R. The diamond dataset contains prices, carat weight, color, and many other attributes of diamonds for almost 54,000 diamonds. The mpg dataset contains a subset of the fuel economy data from [ttp://fueleconomy.gov](http://fueleconomy.gov/). Its scope is limited car models that had a new release every year between 1999 and 2008 with variables such as model, manufacturer, engine displacement, and much more. The iris data set, which is our default data set, gives the measurements in centimeters of the sepal length and width and petal length and width, respectively, for 50 flowers from 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*. The user has the option to visualize a linear regression plot with a quantitative response variable (y), a quantitative explanatory variable (x) and an arbitrary categorical variable. They can choose what type of linear regression model they would like to see plotted: basic regression (y ~ x) or an interaction model (y ~ x \* z) where z is the categorical variable they want to plot an interaction with. Our app also contains the option to plot confidence bands, specifically the mean response for a given x and a specific category level**;** the user can input the level of confidence they want (default being 95%). Another option we included is whether the user wants to include a non-linear best fit line (LOESS). Lastly, there is a way for them to see the two core plots for residual analysis, which are the normality of residuals plot and the residuals vs predicted plot. The combination of all of these features makes data visualization easier for any user.

###Statistical Concepts:

\*\*Regression:\*\*

Regression analysis is a useful statistical method that allows statisticians to better understand the relationships between variables. The app2 has the capability to overlay a linear regression model onto a plot of the data, providing immediate feedback to the user. It can also present an interaction plot with the corresponding interaction statistics.

\*\*Confidence Intervals:\*\*

Confidence intervals are a statistical tool that help statisticians expand their understanding of data beyond their specific sample. Confidence intervals are essentially interval estimates of a certain population parameter. The confidence interval plotted through our app is of the mean response for a given x and a specific category level (if categories are applicable). The user can also input the level of confidence they want; we have chosen the default to be 95.

\*\*Residual Analysis:\*\*

Two key assumptions needed for linear regression are that the error terms are normally distributed and have constant variance. Our app shows a graph of the normality of the residuals and a residual versus predicted plot. Residuals are actually a representation of the errors, so normality of the residuals means normality of the errors. The user should be aware of any patterns or trends in the residual vs predicted plot, as that might cause the results (p-value, confidence, etc) to be invalid. Similarly, if the normality of the residuals does not follow the line of best fit, then the results also might be invalid.

\*\*Meaningful Statistics:\*\*

Our basic linear regression model displays the adjusted r^2 value and the p-value for the model (global p-value for the basic regression model, and interaction p-value for the interaction model) in addition to the slope and the intercept of the regression line on the model. The adjusted r^2 is a statistical measure of how close the data is fitted to the regression line, adjusted for the number of predictors in the model. The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model, so in general a higher value indicates a better fitting line than a lower r^2 value. Other statistics included were the parameters needed to compute the p-value for the interaction: degrees of freedom for interaction and error, the F value, as well as the p-value itself.

\*\*Interaction\*\*

Statistical interaction means the effect of explanatory variable(s) on the response variable depends on the value of another explanatory variable(s), in our case the categorical variable. A small P-interaction value means that there is statistically a significant relationship between the two explanatory variables and that their relative values (to each other) have an impact on the response variable.

###Code:

In order to make a shiny app, you need to have two elements. These elements are the server file (server.R) and the user interface file (ui.R). Both of these work together to get the app running. The code will be included in this folder; you must make sure both the files are in the same folder and you must run the ui.r file to get the app. (If that does not work try running the server file, make sure their names are ui.R and server.R). If you just want to see the app running, follow this link:

<https://kreimer314.shinyapps.io/RegressionVisualization/>

You can find the source code online below:

<https://drive.google.com/open?id=0B4V_0jp9DSsKVjNHVDRySXQ4REE>

###Results/Conclusions of the App:

We have several preloaded datasets (iris, diamonds, and mpg) integrated into the app. The different options that we have provided are: you can choose what explanatory, response, and categorical variables you want to use, you can choose between simple regression and interaction models, and you can also choose to use your own dataset to graph. We have built in warning and fail-safe mechanisms that will prevent the user from doing something they probably did not intend; however, these mechanisms can be easily overridden if the user wants. We have also included a tab that has a residual analysis plot and a normality plot. In the example dataset, an interesting graph to look at is the interaction regression model with sepal length as the explanatory variable, sepal width as the response, and species as the categorical variable. There you see most of the code in action: an interaction model with colored data points and line for each category, as well as the needed degrees of freedom, F-statistic, and p-value. Overall, we believe that we have made data visualization through our app pretty straightforward and useful for any user.

###Objectives:

Our primary objective was to create an app that can give a user who does not know much about coding a way to view plots and retrieve statistical regression information about a dataset. We used ggplot to make the graphs, so the output looks as if the user coded it themselves, and we used R to perform powerful statistical analysis. We believe that we have successfully completed our task, as our app achieves this goal whilst only requiring that the user understand the meaning of explanatory, response, and categorical variables. In other words, unless they just want to play around with graphs, they need to understand some statistics, but no knowledge of coding is required.

###Data/Sources:

Since our app is interactive and allows for the user to read in their own file, we cannot describe every dataset used in our project. However, we do have multiple pre-loaded datasets from R and ggplot2 that we have thoroughly described in our introduction. Initially we found an app that was similar to what we wanted to accomplish. The code for that was published online and a license was provided enabling us to re-use and modify it in any way we like. We removed certain things of this app and then made the changes to include all the options that we have described. A link to this source can be found here: <http://shiny.stat.calpoly.edu/3d_regression/>

###Recommendation:

ShinyApp syntax is fairly straight forward in the context of user interface (UI) coding and it fits nicely into the standard R workflow; however, it quickly becomes unwieldy as apps grow in complexity. The fact that the UI design is ‘code-first’ means that it can be difficult to know: a) what your code will look like once it works, b) where to put the code to get it to show up where you intend, and c) which UI element best fits your task and how to use it. Other UI development environments have graphical interfaces,which makes things much easier; however, ShinyApp is fine for simple to moderately complex apps.

For future groups using ShinyApps, the most important thing to understand is the way that ui.R and server.R work together. The ui.R file contains the code necessary for creating and organizing the different UI elements that will appear in your app. The server.R file contains functions that will be called to perform various jobs in your app. There is a special list called input, which is implicitly accessible in server.R, that contains the values of UI elements in ui.R that have ID’s. The value stored in the UI element can be referenced with input$Element\_ID. There is an intelligent system ‘under-the-hood’ in ShinyApps that remembers which UI elements are accessed by which functions in server.R. This system looks for access to the input list to make these connections. Any time a UI element with a given ID is used or changed by the user, all functions that access the value stored in the element (by its ID) are automatically recalled.

We would highly recommend future groups to try their hands at a ShinyApp. Even if they lack programming experience, they should be able to achieve something relatively simple. The combination of ShinyApp syntax and base R syntax presents a technical hurdle that is very typical of any language being used to construct an ergonomic user interface, but we believe the challenge of working it out to be worth while.