Simple Linear Regression

Signal Data Science

In this assignment, you'll learn how to do a linear regression, the simplest of all machine learning techniques! We'll use data from the United Nations about infant mortality in different countries for this analysis.

There's sample code located in `day1Example.R`.

- Install the packages `car`, `Ecdat`, `HistData`, `ggplot2`, `dplyr`, `Rmisc`, and `GGally`. Load `day1Example.R` in RStudio.

You'll go through the existing code, alternating between figuring out what it does, answering questions here, and writing your own code to supplement what's already there.

Getting started

First, we have some preliminaries to take care of.

- Load the packages `car`, `ggplot2`, and `GGally`. Set `df = UN`.
- You can print out `df` by just typing `df` into the console, but you can also get a nice GUI for looking at data frames by running `View(df)` (case-sensitive). Does anything stand out to you? If there are any questions you'd like to answer with this data, write them down as comments in your R file.
- Packages in R are extensively documented online. Look at the reference manual for the `car` package to read about the `UN` data.

Viewing correlations and cleaning the data

- Use the `cor()` function on the data frame to find the correlation between infant mortality and GDP. What's wrong, and why is this happening? Look in the documentation for `cor()` to figure out how to tell the function to ignore entries with missing values.

- For readability, multiply the correlation matrix by 100 and `round()` it to whole numbers. Wrap all of this (including the above bullet point) into a function, `cor2`, which outputs a correlation matrix (ignoring entries with missing values) with rounded whole numbers.

Instead of making each function we use handle missing values (`NA`s), we can create a new data frame with incomplete rows excluded.

- Type `?na.fail` and read the documentation on `NA`-related functions; find one appropriate for the job and use it to make `df2`, a new data frame excluding rows containing missing values.

Visualizing distributions

We'll now start doing some transformations of the data, leading up to statistical analysis!

- Use the `ggpairs()` function (from the `GGally` package) on the data frame.
  - What do you notice about the distributions of GDP and infant mortality?
  - Figure out how to take a log transformation of the data and assign it to `ldf`, and examine it with `ggpairs()`.
  - Note the differences, and reflect on the appropriateness of a linear model for the untransformed vs. transformed data.

Running linear regressions

If you don't remember much about linear regressions, briefly skim the relevant sections in *Applied Predictive Modeling*.

- Run the lines that use the `lm` command to generate linear fits of infant mortality against GDP. You can type `linear_fit` and `summary(linear_fit)` in the console to get summaries of the results.

You'll note that the `summary()` command will print out a statistic denoted **Adjusted R-squared**, which can be interpreted as the *proportion of variance in the target variable explained by the predictors*. Take a look at StackExchange to briefly see how this statistic is calculated. In general, higher is better.

Plotting linear regressions

To plot the results of a simple linear regression, it's actually easier to let `ggplot2` fit the model for you.

- Run the line starting with `ggplot...` to plot a scatterplot of the data in `df2` along with a linear fit of infant mortality to GDP.
  - What happens when you remove the `method` argument in `geom_smooth()`?
  - Look at the documentation for `geom_smooth()` and determine what method it defaults to.
  - Find the documentation online for that method, **briefly** read about it, and explicitly call it in the `method` argument instead of `"lm"`. How good of an approximation is a linear model?
  - Make the same plots for the linear fit of log(infant mortality) vs. log(GDP). (*Hint:* To access column `c` of a dataframe `df`, use the syntax `df$c`.)

Looking at the residuals

A residual is a fancy word for prediction error; it's the difference given by `actual - predicted`.

Why is this important? By fitting a model to our data and looking at the residuals, we can visually inspect the results for evidence of heteroskedasticity.

- Run the first `qplot` command, which plots the residuals (`actual - predicted` values) of the simple linear fit. Is there evidence of heteroskedasticity?
- Run the second `qplot` command, which plots the residuals of the linear fit of the log-transformed data. Is the log-log transformation an improvement? Why or why not?

One of the assumptions of linear regression is that the variances of the distributions from which the errors are drawn have the same variance. If that's the case, then we shouldn't really see much structure in the plot of the residuals, so seeing structure in the plot of residuals is a warning sign that our model isn't working. For example, compare the top and bottom plots here (top has structure, bottom doesn't).

- Using the documentation and experimenting in the console, make sure you understand what `df$infant.mortality - exp(fitted(loglog_fit))` does.

- Generate and analyze a plot of residuals for the linear fit of log(infant mortality) vs. GDP.