Linear Regression: Galton's Height Data

Signal Data Science

With the Galton height dataset, you'll be getting more experience with basic operations on data frames and linear regressions.

Don't worry if some of the ways in which R works seem opaque or confusing to you. R has a steep learning curve, and we don't expect you to be an expert in R after this single assignment. You'll study some important internal details of the language later, but first, it's important to get some intuition for the kind of stuff you can do with R and motivation, in the form of interesting datasets and questions, to learn the language at a deeper level.

Getting started

- Load the `HistData`, `dplyr`, and `Rmisc` packages and set `df = GaltonFamilies`. Take a look at it visually with `View(df)`.
- What variables does `df` include? Check the documentation to figure out what `midparentHeight` represents.

This time, the data frame has a lot of different columns. You can use `names()` to show the column names of `df` (you'll note that the output of `names(df)` is the same as the output of `colnames(df)`), and for a specific column `col` you can access it with the `$` operator, like so: `df$childHeight`. You can access and modify columns just like any other variable (with some small exceptions).

- Go back to the linear fits you made for the infant mortality dataset. Call `names(summary(linear_fit))` and figure out how to access the adjusted R-squared statistic directly instead of having to print out the entire summary of a linear fit every single time.

The `gender` variable is encoded as a **factor**, which we'll cover in greater depth later. For now, since we want to run linear regressions including gender, we want to turn it into a *binary numeric variable*, with values 0 and 1.

- Use a combination of arithmetic and `as.numeric()` to turn the `gender` variable into a column of 0s and 1s. Be sure to keep track of which gender you assign to each of 0 and 1.

Learning to use `dplyr`

The `dplyr` package is one of Hadley Wickham's most commonly used R packages and is particularly useful for the straightforward manipulation of data frames.

- Read through page 43 of Hadley Wickham's `dplyr` tutorial. Don't work through all the examples – just skim and refer back to the tutorial later if you need to.

Running linear regressions

In the questions below, "run a regression of A against X, Y and Z" should be understood as "first run a regression of A against X, Y and Z individually, perhaps in pairs, then run the regression against all three variables".

- Aggregate the data by family using the appropriate `dplyr` function. Before writing any R code to do so, think about what this aggregation actually means and what kinds of variables you want to calculate on a per-family basis.
  - The `Rmisc` package loads the `plyr` package as a dependency, which loads its own `summarise()` function and consequently overrides `dplyr`'s `summarize()`. To access the `summarize()` of dplyr, call `dplyr::summarise()`.
- Plot the average heights of children in each family against the mothers' heights, the fathers' heights, and the mid-parent heights. Afterward, use multiplot to display all three graphs at the same time.
  - The linked example uses `geom_line()`; ignore that part and just use it to learn about how to use `multiplot()` to combine plots.
- Compute and visually inspect the correlations between the variables in your aggregated dataset. Make a couple hypotheses about the data that you think you might be able to prove or disprove as you do further analysis and exploration.
- Compare the results of regressing average child heights against (fathers' heights and mothers' heights versus regressing against mid-parent heights. Interpret the results, paying particular attention to the adjusted R-squared statistic.
- Run a linear regression of the numbers of children in families against fathers' heights, mothers' heights, and average child heights. Looking at the summaries of the linear fits, do these regressions capture any statistically significant relationships? If so, with what p-values?

  - Here, "*p*-values" refers to the *p*-values associated with each coefficient in the regression, associated with testing for the null hypothesis that the value of each coefficient is equal to zero.

- Following the example usages of `ggplot()` that you've already seen along with the official documentation, use `ggplot()` in conjunction with `geom_histogram()` to make a histogram displaying the distribution of number of children per family.

Doing additional analysis

Let's return the original, *unaggregated* data. When we instruct you to run regressions, you should automatically assume that you should try to do as much interpretation as possible.

- Run a regression of child heights against mothers' heights, fathers' heights, mid-parent heights, and gender.
- Use the appropriate functions in `dplyr` to restrict to children of either gender and run regressions of child heights against mothers' heights and fathers' heights.
- Is it possible for us to analyze the effect of being firstborn vs. secondborn vs. thirdborn vs. . . . on child heights? Why or why not?