Self Assessment

Self Assessment

Today, you'll be completing a short assessment so that we can get a sense of where you're at.

- Type your answers in a new R script file with comments indicating where the answer to each question begins.
- Write down the current time. Please email us (at `signaldatascience@gmail.com`) with your R script attached after **2 hours** have passed.
- Work individually. You can however consult R documentation, look at old assignments, use the Internet, etc., but don't copy and paste code verbatim.
- Make your code as clear, compact, and efficient as possible. Use everything that you've learned!

Libraries you will find useful: `ggplot2`, `psych`, `dplyr`.

Don't worry if you can't figure out some of the questions or if it seems like you'll need more than the allotted time to finish the self-assessment. Just put in your best effort!

Part 1: R and Probability

Here's an interview question from *Euclid Analytics*:

> Suppose that X is uniformly distributed over $[0, 1]$. Now choose $X = x$ and let Y be uniformly distributed over $[0, x]$. Is it possible for us to calculate the "expected value of X given $Y = y$", *i.e.*, $\mathbb{E}(X|Y = y)$?

(If you don't know what *expected value* is, you can think of it as the mean of each possible outcome weighted by its probability.)

Now, we don't know the answer yet, but maybe we can get some sense of what it might look like by doing some Monte Carlo simulations. To that end:

- A *single trial* of the process described in the problem will yield a pair of values (x, y), where the probability distribution which y is drawn from depends on the value of x. Simulate k trials of this process for $k = 1000000$. (You may find the `runif()` function helpful.)
  - Plot the simulated values with `qplot()`.
- Since we're interested in the *expected value* of X given some $Y = y$, we can approximate this by separating our values of Y into *bins* (of equal width) and taking the *mean* of X within each bin.
  - Write code that does so for a bin width of w, setting $w = 0.01$ (equivalently, setting the number of bins to $1/0.01 = 100$).
  - Use `qplot()` to view the results. Do they make sense?

Now, suppose that a magic fairy whispers into your ear:

> Here's the answer, my friend! It just so happens that $\mathbb{E}(X|Y = y) = \frac{y-1}{\ln y}$.

In light of this revelation, you want to verify your computational results from earlier. To that end:

- Generate a lot of different values of Y and calculate the corresponding values of $\mathbb{E}(X|Y = y)$ according to the equation above.
  - Graph them using `qplot()`. Does this graph match your simulated results?
- Make a *single* dataframe with both your Monte Carlo-simulated results *and* your direct calculation of the theoretical result.
  - Make a single graph with (1) a scatterplot of the Monte Carlo-simulated results and (2) a smooth line connecting the points corresponding to the theoretical values. This should just basically be both of your previous graphs plotted at the same time.

Part 2: Data Analysis

We'll be looking at psychological test data.

- Load the `msq` dataset from the `psych` library and call `help()` on it to see what it's about. For convenience, set it equal to the variable `df`.
- Compute the fraction of missing values for each feature, sorted in descending order. The first of these should be 0.528747 for `"kindly"`.
- Make a new dataframe with columns `"active"` through `"scornful"` as well as Extraversion and Neuroticism.

- Replace each missing value with the mean of the column which the missing value is in.
- Create each of the following plots:
  - Histograms for Extraversion and Neuroticism (`geom_histogram()`)
  - Density plots for Extraversion and Neuroticism (`geom_density()`)
  - A scatterplot of Extraversion scores vs. Neuroticism scores, with a smoothed nonlinear fit to the points overlaid on top
- Run linear regressions of Extraversion and Neuroticism against all the other features. Neuroticism should *not* be one of the predictors for Extraversion and vice versa, but `"active"` through `"scornful"` should be included simultaneously.
- Use the `coef()` function on each of the two linear models to view the coefficients associated with each linear fit.
  - Print out the top 10 coefficients for each of the two linear fits. They should be ordered by absolute value (largest to smallest).