STAT/CS 94 Fall 2015 Adhikari HW11, Due: 11/18/15

Problem 1 Comprehension Comprehension

In this problem, we'll introduce you to a new piece of Python syntax, *list comprehensions*. Here is a piece of code you might find familiar by now:

```
results = []
for element in some_list:
    value = do_computation(element)
    results.append(value)
```

That's a lot of code to express something fairly simple: Make a list of the things you get when you do some computation (for example, calling `do_computation`) on each element of `some_list`. In fact, you have seen a simpler way to do that when your list is a column in a table: `table.apply(do_computation, 'column_name')`. But what if you don't want to use tables? And what if you don't want to define a function that does everything you want to do with each element? That's where list comprehensions are useful. We can rewrite our code above as: `results = [do_computation(element) for element in some_list]`. The syntax for a basic list comprehension is:

```
[<expression possibly involving varname> for <varname> in <some_list>]
```

The value of the expression is the list you get when you evaluate the expression `len(<some_list>)` times, each time defining `<varname>` to be the next element of `<some_list>`. `<some_list>` can also be a Numpy array or another sequence-like thing. For example, here is a list comprehension whose value is a list (note: not an array) of the first 5 perfect squares (including 0): `[x*x for x in np.arange(5)]`.

Here are some basic exercises to get familiar with list comprehensions.

(a) Write a list comprehension whose value is a list of the first 10 even numbers (including 0).

(b) Write a list comprehension whose value is a list of the square roots of the first 7 whole numbers (including 0).

(c) Suppose someone has put the English names of the 12 months of the year in a list called `month_names`. Write a list comprehension whose value is a list of the lengths of the names of the months.

(d) Suppose you have performed a 3-nearest-neighbors regression on a dataset with 2 columns and defined a function called `regression_prediction` that takes a single number as its argument and returns your regression prediction for that number. Write a list comprehension whose value is a list of your regression predictions for the numbers 0.5, -1.0, and 5.0, in that order.

Answer:

(a) `[2*x for x in np.arange(10)]` (you could also use the filtering syntax for list comprehensions, which we didn't talk about: `[x for x in np.arange(20) if x \% 2 == 0]`)

(b) `[math.sqrt(x) for x in np.arange(7)]`

(c) `[len(name) for name in month_names]`

(d) `[regression_prediction(x) for x in [0.5, -1.0, 5.0]]`

Problem 2 Data-Driven Design

For this problem **and the remainder of the problems in this assignment**, here are four useful functions, which you may assume have already been defined for you:

1. `bootstrap_mean(table, column_label)`:
    - Bootstraps `table[column_label]` using 10,000 repetitions
    - Returns an approximate 95% confidence interval for population mean of `column_label`
2. `bootstrap_slope(table, column_x_label, column_y_label)`:
    - Bootstraps rows of `table` using 10,000 repetitions
    - For each bootstrap sample, performs a regression of `column_label_y` on `column_label_x`
    - Returns an approximate 95% bootstrap confidence interval for the true slope
3. `perm_test_means(table, column_label, group_label)`:
    - Randomly permutes `table[column_label]` 10,000 times to test equality of distributions by comparing sample means
    - `group_label` denotes Group A by 0 and Group B by 1
    - Returns the approximate P-value of the test
4. `bootstrap_AB_test_means(table, column_label, group_label)`:
    - Bootstraps `table[column_label]` to test equality of distributions by comparing sample means, using 10,000 repetitions
    - `group_label` denotes Group A by 0 and Group B by 1
    - Returns the approximate P-value of the test

In this problem and each one below, **give a precise statement of the null hypothesis or explain why no null hypothesis is needed**. Say which (if any) of the above functions should be used, and, as appropriate, write the code you would run to call the function. (You may include code to augment tables if needed.) Please explain how you would use the output of the function to answer the question. You may assume that all samples are large, as are the numbers of repetitions of the sampling procedures. If you use regression, you may assume the regression model is justified.

Please note that the questions are stated rather loosely; this is how questions arise in practice. You, as the data scientist, have to decide how to make the question precise.

Now, for the question: Researchers would like to study the effect of a website's design on the amount of money people spend at that site. Each of 500 visitors to the site is randomly assigned to view one of two versions of the site. The table `site` contains the data. There is one row for each of the 500 visitors. The column `'version'` contains the code 1 or 2 depending on which version the visitor viewed. The column `'money'` contains the amount of money, in dollars, spent by the visitor on the site. Explain how you would test whether the version makes a difference to the amount of money spent. **Answer:** First, a note on this problem and the next 3. Part of the purpose of these problems was to present more realistic, open-ended scenarios. It is true in general, but especially true for these problems, that there are reasonable answers that don't exactly match ours. We'll accept any answer with a clear and convincing justification.

Back to the problem. We are not told anything about the process for choosing the 500 visitors, so the techniques we've seen in this class cannot tell us anything about the *general population* of visitors, only these 500. So we take as our population the following 500 pairs of numbers: the 500 amounts of money that would have been spent by each user if they *had been assigned* version 1, and the 500 amounts of money that would have been spent under version 2. Our null hypothesis is that the means of these two groups of 500 numbers are equal. (Note that another possible interpretation of the question is that it asks whether the amount of money hypothetically spent by *each visitor* is the same for each assigned version. But it's not clear how to test that, and probably the researchers care only about the average revenue anyway.)

We only observe the money spent for the website version each user was actually assigned to – only half of the 1000 numbers. But since the assignment process was random, the observations we have constitute a random sample from the population we're interested in, and we can use the bootstrap methods we've seen in class. Our test statistic is the difference in mean money spent among the two groups. We will compute an approximate P-value with a bootstrap test (we could equally well use a permutation test) as follows:

```
# (bootstrap_AB_test_means expects groups to be labeled 0 and 1)
site['zero-one-version'] = site['version'] - 1
p_value = bootstrap_AB_test_means(site, 'money', 'zero-one-version')
```

Then, if `p_value` is less than, say, .01, we will reject our null hypothesis and declare that the version made a difference in the amount of money spent among these 500 visitors.

Problem 3 Population Puzzler

The table `usa` contains one row for each of the years 1970 through 2010. In each row, the column `'century'` contains a 0 if the year is in the 20th century and 1 if it is in the 21st. The column `'population'` contains the US population for that year. How would you go about deciding whether the US population is on average higher in the 21st century than in the 20th?

Answer: We are interested in comparing the average of the US population numbers in the years 1900-1999 to the average of the US population numbers in the years 2000-2015. We do not observe all of the population data, so we cannot answer the question just by computing means of the data. But note first that *we don't really need to see the data* to answer this question with pretty high confidence. You probably already know some background facts: population has steadily increased in most countries over the last century, in the absence of major catastrophes; and there have been no major population-decreasing events in the US in the 21st century. Together, these imply that the average in the 21st century is higher than the 20th century.

The techniques we've seen in this class would allow us to answer the question if we observed subsets of the two centuries' data *that were selected at random from all the data we were interested in.* Here, we have not been assured that the sample is random, and in fact the data from the 20th century were pretty clearly selected by just taking the last 30 years of that century. The data don't tell us anything about the early 20th century. Therefore it would not be reasonable to use the techniques from this class to answer this question. In particular, null hypothesis testing is not useful.

Problem 4 Education Elucidation

The table `couples` consists of one row for each of 600 couples chosen randomly from all the married couples in a city. The column `'educ_h'` contains the husband's educational level in years, and `'educ_w'` contains the wife's educational level in years. Explain how you would decide whether or not the husbands in the city are about as educated as their wives.

Answer: As always, before using statistics, it is important to first understand how we would answer the question if we had access to the entire population (the educational levels of all the married male-female married couples in the city). First, we should notice that the question is asking something about the relationships between pairs of data points – husbands' education levels and wives'. A simple concrete form of the statement that husbands' education levels are "about the same" as their wives' is that the difference between the two is 0 *on average*, *in the city.* That will be our null hypothesis.

Now we just need to understand how to test that null hypothesis using a random sample. We will simply compute a confidence interval for the population mean of the differences, and reject the null if that interval does not include 0. To do that, we can run the following code:

```
couples['diffs'] = couples['educ_h'] - couples['educ_w']
confidence_interval = bootstrap_mean(couples, 'diffs')
reject_null = confidence_interval[0] > 0 or confidence_interval[1] < 0
```

Here are some potential alternative answers:

(1) It's not totally satisfying to know that the average difference in education levels is 0; we might instead want to know whether the typical *size* of the difference is small. For example, we could use as our test statistic the average absolute difference, `np.mean(np.abs(couples['educ_h'] - couples['educ_w']))`. It is implausible that it would be 0, since that would require that every husband is exactly as educated as his wife. So we could say, somewhat arbitrarily, that our null hypothesis is that the average absolute difference between husbands' and wives' education levels *in the city* is less than 1 year. To test this hypothesis, we could still use `bootstrap_mean` on the average absolute difference; we would reject the null if the confidence interval did not overlap the interval $[0, 1]$.

(2) We could view this as a regression problem: We are regressing husbands' education levels on wives' education levels. Our null hypothesis would be that the line `husband_ed = wife_ed` approximately fits the population. Concretely, our null would have to have 3 components: The slope and intercept of the least-squares fit line in the population are close to 1 and close to 0, respectively. (Concretely, say they are in the ranges [0.9,1.1] and [-0.1, 0.1], respectively.) Further, the correlation of that line is at least, say, .9. But now we must use the sample to jointly test those three things – that is, we must decide whether at least one of them is false. You can do that, but we haven't learned how to do it in this class, and the given functions are not really up to the task. So this is not the best answer.

(3) We could view this as asking whether the *fixed* line `husband_ed = wife_ed` fits the data well. We could compute the mean squared residual for that line and take as our null hypothesis that its value in the population is small, say less than 1. In the sample, this amounts to `np.mean((couples['educ_h'] - couples['educ_w'])**2` – essentially the same as alternative (1), but using squared distance rather than absolute difference.

Problem 5 Prolificacy Problem

(continuing the previous problem) The column `'children'` of the table `'couples'` contains the number of children in the couple's household. Explain how you would decide whether the wife's educational level makes a difference to the number of children, among families in the city.

Answer: In contrast with problem 4, we are not trying to decide whether the wife's educational level *matches* some other thing, but only whether it makes *any difference at all* in the number of children. Numbers of children are in different units (children) than years of education (years), so it is natural to draw a scatter diagram and turn to regression. In this class, we have only seen a way to test whether there is an approximate *linear* relationship between two things of different units, so that is what we'll test. Concretely, our null hypothesis is that the slope of the least-squares fit line of number of children on wife's education level, for all the families in the city, is 0. If we reject this hypothesis, then there is a linear relationship between the two things, so we can say that the wife's educational level makes a difference to the number of children.

Given our random sample of the population of families, we will compute an approximate 95% confidence interval for the population slope using the bootstrap, and then reject the null hypothesis if the confidence interval does not include 0. To do this, we can run the following code:

```
confidence_interval = bootstrap_slope(couples, 'educ_w', 'children')
accept_null = confidence_interval[0] <= 0 and confidence_interval[1] >= 0
```