Economics 691-06: Assignment 4

Nihar Shah

Solutions are marked in blue.

Problem 1: Quasi-Experimental Methods, Part I

Hello Fresh is a meal-preparation service that operates in several countries. It sends subscribers the ingredients to cook meals a few times a week. Assume in this stylized problem that Hello Fresh has a single meal plan and operates in the US, Canada, and Australia. It collects feedback from consumers with monthly surveys, scored from 0 through 10.

Hello Fresh recently pioneered a low-sodium version of its meal plan. It does not have the logistical infrastructure to provide two different meal plans in the same country, but it still wants to understand the effect of the low-sodium plan on satisfaction ratings. It thus rolls out the low-sodium plan in the US only, without making changes in Canada and Australia. Note that this roll-out only occurred at the start of August; until then, all consumers received the regular meal plan. Hello Fresh asks you to use this data to estimate a treatment effect.

The data is in the file data_assignment4_1.csv. The file has 3,000 rows. Each row corresponds to a separate user, for which there are three columns. The first column is the user's country: US, CA (Canada), or AU (Australia). The second column is the month in which the survey took place: one of April, May, June, July, or August (2020). The third column is the outcome, i.e. the user's surveyed satisfaction from 0 through 10. There are exactly 1,000 users from each country.

```
# Load libraries: numpy, pandas, and linear regression
import numpy as np
import pandas as pd
import statsmodels.formula.api as sm

# Load in data
from google.colab import files
import io
uploaded = files.upload()
data = pd.read_csv(io.BytesIO(uploaded['data_assignment4_1.csv']))

# As some pre-processing, note (and drop) missing data
print("There are " + str(data.isnull().sum().sum()) + " missing rows.")
data = data.dropna()
```

```
There are 49 missing rows.
```

```
# Also, we define a treatment indicator to make the problem easier, which
# takes the value 1 when in the US in the month of August
data['treatment'] = 0
col_index = data.columns.get_loc('treatment')
data.iloc[np.where((data['country'] == 'US') &
                   (data['month'] == 'August'))[0],col_index] = 1
```

1. **(5 points)** Using the event study methodology, compute the effect of treatment (along with standard errors). Note any choices you make in determining the length of the pre-treatment period.

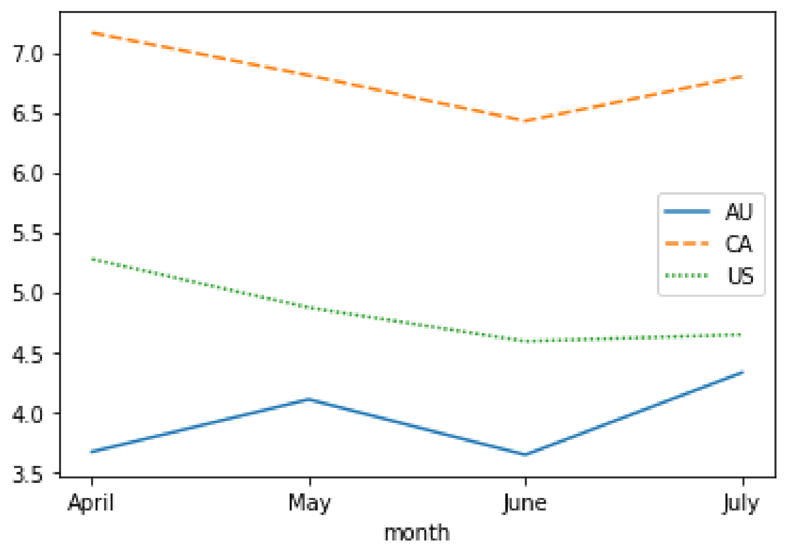
   The computation is below. Since consumers likely fully internalize the effect of the meal plan right away, we can keep the window length in the pre-treatment period short. Moreover, we can rely on linear regression for the standard errors, as the countries and sample sizes within them are fixed.

   ```
   # Compute the effect as an event study
   def event_study(df):
     result = sm.ols(formula = "satisfaction ~ treatment", data = df).fit()
     print("The parameter estimate is: " + str(round(result.params[1], 4)))
     print("The standard error is: " + str(round(result.bse[1], 4)))

   # We first try using a short pre-period
   # Since the result is significant, we do not need to risk a longer pre-period
   index = np.where((data['country'] == 'US') &
                    ((data['month'] == 'August') | (data['month'] == 'July')))[0]
   event_study(data.iloc[index,])

   The parameter estimate is: 0.8669
   The standard error is: 0.1889
   ```

2. **(10 points)** Now, we prepare to do the differences-in-differences analysis. To do so, examine the parallel trends assumption across the US, Canada, and Australia. For which countries does this assumption hold better or worse?

   We plot the parallel trends for the US, Canada, and Australia prior to the treatment month (August) below. The Canadian trends seem approximately equivalent to the US ones, and so parallel trends is likely to hold there, making Canada a valid control group. The Australian trends do not, and so Australia would not be a good candidate for the control group in the diff-in-diff analysis.

   ```
   # Plot parallel trends
   import matplotlib.pyplot as plt
   import seaborn as sns
   pivot = pd.pivot_table(data, values= 'satisfaction', index = 'month',
                          columns= 'country', aggfunc = np.mean)
   pivot = pivot.reindex(["April", "May", "June", "July"])
   sns.lineplot(data=pivot, sort = False)
   ```



3. **(5 points)** Using the differences-in-differences methodology and your assessment from the previous question, compute the effect of treatment (along with standard errors).

   The computation is below, using US and Canadian data. Moreover, we can rely on linear regression for the standard errors, as the countries and sample sizes within them are fixed.

```
# Compute the diff-in-diff effect, using Canada as the control group
def diff_diff(df):
  result = sm.ols(formula = "satisfaction ~ treatment + month + country",
                  data = df).fit()
  # Note that we have to find the right index, as there are multiple variables
  # in this regression
  j = np.where(result.params.index == 'treatment')[0][0]
  print("The parameter estimate is: " + str(round(result.params[j], 4)))
  print("The standard error is: " + str(round(result.bse[j], 4)))

index = np.where((data['country'] == 'US') | (data['country'] == 'CA'))[0]
diff_diff(data.iloc[index,])

The parameter estimate is: 1.1723
The standard error is: 0.2179
```

Problem 2: Quasi-Experimental Methods, Part II

Blue Apron is another meal-preparation company that competes with Hello Fresh, and that operates in the US only. Assume that in this stylized problem, it also used to have a single meal plan; and recently developed a low-sodium plan that it wants to test with consumers. Unlike Hello Fresh, Blue Apron has not run satisfaction surveys in the past, and so cannot rely on historical data. On the plus side and still unlike Hello Fresh, Blue Apron does have the logistical infrastructure to provide two different meal plans in the same country.

Blue Apron does not want to randomize due to the PR risk of consumers receiving low-sodium meals without their consent. As such, it opens an online queue. Blue Apron is able to serve low-sodium meals to the first 10,000 customers in the queue; although it cannot serve such meals to the remaining (who continue to receive their regular meals), as Blue Apron later realizes it does not have enough low-sodium meals. Blue Apron asks you to use this data to estimate a treatment effect.

The data is in the file data_assignment4_2.csv. The file has 30,000 rows. Each row corresponds to a separate user, for which there are four columns. The first column is the user's position in the queue, where NA indicates that the user chose not to sign up for the queue at all. The second column is whether the user received the treatment, i.e. low-sodium meals (indicated by 1), or continued to receive the regular meals (indicated by 0). The third column is the outcome, i.e. the user's surveyed satisfaction from 0 through 10. The fourth column is called "popups" and should be put aside until later in the problem.

```
# Load libraries: numpy, pandas, and linear regression
import numpy as np
import pandas as pd
import statsmodels.formula.api as sm

# Load in data
from google.colab import files
import io
uploaded = files.upload()
data = pd.read_csv(io.BytesIO(uploaded['data_assignment4_2.csv']))
```

1. **(5 points)** Why can we not regress satisfaction on treatment for a valid treatment effect? Explain this in concrete language relevant to this scenario.

   The choice to opt into low-sodium meals is not random, and so there will be selection biases. For instance, people who opt into low-sodium meals may just be less happy people in general; and that may thus show up as a spurious negative treatment effect.

2. You decide to pursue a regression discontinuity design, in which you analyze consumers around the queue cutoff of 10,000.

   (a) **(5 points)** Using the same concrete language relevant to this scenario, why can we compare consumers' outcomes just above and below the cutoff to get a valid treatment effect? In your answer, note any assumptions and whether you believe them to satisfied. (You do not need to formally test the assumptions.)

   While there may be selection biases that cause people to sign up for low-sodium meals (and at different speeds), we can plausibly believe those who joined the queue just above and just below the 10,000 cutoff are effectively identical (despite receiving different treatments). Formally, this means that we believe the continuity assumption. This can be violated if the assignment rule is known in advance and agents can manipulate around it, but because Blue Apron's method uses a queue (which is hard to manipulate) and may not have enough committed to the first 10,000 signups publicly anyways, it seems unlikely for manipulation to occur.

   (b) **(5 points)** Using the regression discontinuity methodology and all data, estimate the treatment effect (along with standard errors).

We implement the regression discontinuity design below, regressing outcomes on the queue position and including a dummy for treatment. Those consumers who did not participate in the queue are excluded by the regression, and this is deliberate. Moreover, we can rely on linear regression for the standard errors, as the cutoff and thus the sizes of the treated and control groups are fixed.

```
# Function to implement regression, and return coefficient and SE
def reg_discontinuity(df):
  result = sm.ols(formula = "satisfaction ~ treatment + queue", data = df).fit()
  # Note that we have to find the right index, as there are multiple variables
  # in this regression
  j = np.where(result.params.index == 'treatment')[0][0]
  print("The parameter estimate is: " + str(round(result.params[j], 4)))
  print("The standard error is: " + str(round(result.bse[j], 4)))

# Run regression with all data
reg_discontinuity(data)
```

```
The parameter estimate is: 1.2183
The standard error is: 0.0385
```

(c) **(5 points)** Use the same methodology but restrict to data within a narrow band around the cutoff: those with queue places between 9,000 and 11,000. Why might the result differ, and which result do you trust more?

The estimate from the narrow band likely has less bias, as it exploits the discontinuity in treatment more sharply without needing strong assumptions about how the position in the queue is generally related to the outcomes. However, because it has a smaller sample size, it also has higher variance. I personally might be inclined to favor the results from the narrow band regression in this case. However, arguments defending either estimate will be accepted as long as there is some discussion of the bias-variance tradeoff.

```
# Run regression with data in the narrow queue
reg_discontinuity(data.iloc[np.where(
                 (data['queue'] > 9000) & (data['queue'] < 11000))[0],])
```

```
The parameter estimate is: 1.0072
The standard error is: 0.1262
```

3. An engineering team at Blue Apron tells you that they actually were running a separate experiment over this timeframe. Specifically, they were running pop-ups for some users, which informed them about new developments at Blue Apron; and some pop-ups were indeed related to these low-sodium meals and the queue for them. This is the fourth column – "popups" – which tells if you users were receiving these pop-ups (indicated by 1) or not (indicated by 0). You decide to pursue an instrumental variables approach, in which pop-ups are the instrument.

(a) **(10 points)** Using the same concrete language relevant to this scenario, why might pop-ups be a valid instrument for this problem? In your answer, note any assumptions and whether you believe them to satisfied.

There are two key assumptions for the instrumental variables methodology to work. First, the instrument must satisfy relevance, i.e. it must predict the endogenous variable (treatment) with a sufficiently high F-statistic (e.g. over 10). We run that regression and find an F-statistic that exceeds 6500, so that assumption holds. Second, the instrument must satisfy the exclusion restriction, i.e. it must not be correlated with the outcome in any other way. This is a less defensible assumption in this context. It is possible that the pop-ups affected consumer satisfaction with Blue Apron in other ways, e.g. by informing them about new ways to cook leftover ingredients, new ways of interacting with the product, etc.

```
# Check the F-stat of regressing the treatment variable on the instrument
result = sm.ols("treatment ~ popups", data = data).fit()
print(result.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:              treatment   R-squared:                       0.179
Model:                            OLS   Adj. R-squared:                  0.179
Method:                 Least Squares   F-statistic:                     6551.
Date:                Sat, 19 Sep 2020   Prob (F-statistic):               0.00
Time:                        07:22:09   Log-Likelihood:                -17044.
No. Observations:               30000   AIC:                         3.409e+04
Df Residuals:                   29998   BIC:                         3.411e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.5333      0.003    152.788      0.000       0.526       0.540
popups        -0.3992      0.005    -80.939      0.000      -0.409      -0.389
==============================================================================
Omnibus:                     5322.212   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2015.287
Skew:                           0.440   Prob(JB):                         0.00
Kurtosis:                       2.085   Cond. No.                         2.62
==============================================================================
Warnings: Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

(b) **(10 points)** Using the instrumental variables methodology, estimate the treatment effect.

We can estimate the treatment effect by performing the two-stage process. First, we regress the endogenous variable (treatment) on the instrument (popups); and we use the resulting model to generate predicted values for the endogenous variable. Note that those consumers who did not participate in the queue are excluded by the regression, and this is deliberate. Second, we regress the outcome (consumer satisfaction) on the predicted endogenous variable, and report that parameter estimate.

```
# Write a helper function to perform the two-stage IV procedure, of first
# regressing treatment on the instrument, and second regression the outcome
# on the predicted (rather than actual) value of treatment
def iv_regression(df):
  result = sm.ols(formula = "treatment ~ popups", data = df).fit()
  df['treatment_pred'] = result.predict(df)
  result = sm.ols(formula = "satisfaction ~ treatment_pred", data = df).fit()
  return(result.params[1])

print("The parameter estimate is: " + str(round(iv_regression(data), 4)))
The parameter estimate is: 0.9925
```

(c) **(5 points)** Using the bootstrap, compute the standard errors of your treatment effect. Note that we could use regression methods to get the standard errors, as the sample sizes for the treatment and control groups are fixed. However, we cannot simply read the standard errors off from the OLS summary, as there are two sources of uncertainty, one corresponding to each stage of the two-stage process; and so we can only use regression methods that take this into account (e.g. the ivreg function in R).

As such, a much easier approach is to simply bootstrap, with sample sizes of 10,000 treated and 20,000 untreated observations.

```
# Set seed for bootstrap
np.random.seed(5)

# Implement bootstrap methodology, holding sample sizes in treated and
# control groups fixed
bootstrap = []
i = np.where(data['treatment'] == 1)[0]
j = np.where(data['treatment'] == 0)[0]
for k in range(1000):
  index = np.random.choice(i, len(i), replace = True)
  index = np.append(index, np.random.choice(j, len(j), replace = True))
  bootstrap.append(iv_regression(data.iloc[index,]))

print("The standard deviation is: " + str(round(np.std(bootstrap), 4)))

The standard deviation is: 0.0462
```

Problem 3: Streaming Bootstrap

(25 points) Write a function that can compute the standard errors for the mean of a data series via 1000 bootstrapped iterations. Assume that the data is provided in batches, and each batch must be deleted before the next batch is provided. Test the function on a dataset that comes in five separate batches, where each batch has 100,000 observations randomly and uniformly distributed from 30 to 50; and report the standard error.

Hint: the function to generate random Poisson values is numpy.random.poisson(λ, n) in Python and rpois(n, λ) in R.

The function is not particularly complex or lengthy, but it does take some planning and careful manipulation of arrays. For a given data size (say $n = 100,000$) and a given number of Bootstrap iterations (say $k = 1,000$), we generate $n \times k$ weights, from the Poisson distribution, and assign n weights to each of the k iterations. We then increment each iteration's numerator by the sum of all the data times the assigned weights; and increment each iteration's denominator by the sum of the assigned weights. Note that the function below does these multiplications and sums efficiently and parsimoniously, but you are encouraged to use nested loops and simple examples to develop intuition and build up the function.

That leaves us with a numerator and denominator for each of the k Bootstrap iterations that can be incremented with additional data later. Similarly, when we are ready to compute the standard error, we divide the numerator by denominator for each Bootstrap iteration; and the standard deviation of those values is the standard error.

```
# Load library and set parameters
import numpy as np
iterations = 1000

# Implement streaming bootstrap
# Can either start with fresh data, or take in an existing sum / n
def streaming_bootstrap(data, sum = np.zeros(iterations),
                        n = np.zeros(iterations), flag = True):

  # Get weights and increment data counter
  w = np.random.poisson(lam = 1, size = iterations * len(data))

  # Reshape weights and data for summation
  w = np.reshape(w, [len(data), iterations])
  data = np.reshape(data, [len(data), 1])

  # Multiply each data point by weights for all bootstrap iterations, and
  # then sum across each bootstrap iteration
  # Note the second argument to np.sum does the summation across columns,
  # as opposed to rows
  sum = np.sum(w * data, 0) + sum

  # Sum the weights for the denominator too
  n = np.sum(w, 0) + n

  # Return either summary statistics or standard deviation of estimates
  if flag:
    return([sum, n])
  print("The standard deviation is: " + str(round(np.std(sum/n), 4)))

# Process five batches of 100,000 observations
np.random.seed(10)
for i in range(5):
  data = np.random.uniform(10, 30, 100000)
  if i == 0:
    sum, n = streaming_bootstrap(data = data)
  elif i < 4:
    sum, n = streaming_bootstrap(data = data, sum = sum, n = n)
  else:
    streaming_bootstrap(data = data, sum = sum, n = n, flag = False)
```

```
The standard deviation is: 0.0081
```