Notebook Objective

In this notebook, I use an instrumental variable to estimate the effect of a time limited welfare eligibility on family well-being.

Data are from the Family Transition Program (FTP). FTP was the first welfare reform initiative in which some families reached a time limit on their welfare eligibility and had their benefits canceled. The program took place in Escambia County, Florida from 1994 to 1999. Key findings from the study, as well as additional background information can be found here.

Preparing the Data

In [19]:
```python
from linearmodels import IV2SLS # pip install linearmodels
import numpy as np
import pandas as pd
from scipy.stats import pearsonr
import statsmodels.formula.api as smf
```

I leverage both the administrative data (which contains official employment and income records) and survey data (which contains participant self-reported data on well-being) in the analysis. We'll need to load, merge, and clean the data before estimating treatment effects.

In [2]:
```python
def admin_data_loader():
    """Loads ftp administrative dataset."""

    path = '/Users/danielchen/Desktop/UChicago/Year Two/Autumn 2020/Program Evaluation/Problem Sets/Pro
blem Set 2/ftp_ar.dta'
    df = pd.read_stata(path)

    return df


def survey_data_loader():
    """Loads ftp survey dataset."""

    path = '/Users/danielchen/Desktop/UChicago/Year Two/Autumn 2020/Program Evaluation/Problem Sets/Pro
blem Set 1/ftp_srv.dta'
    df = pd.read_stata(path)

    return df


def ftp_merger(dataframe1, dataframe2):
    """Merges the two ftp datasets."""

    df = pd.merge(dataframe1, dataframe2, on='sampleid')

    return df
```

In [3]:
```python
admin = admin_data_loader()
survey = survey_data_loader()
df = ftp_merger(admin, survey)
df
```

Out[3]:

	sampleid	e_x	cflag	longtdec	b_aidst	gender	ethnic	marital	afdctime	afdchild	...	emppq1_y	yrearn_y	yrearnsq_y	pearn1_y
0	1	0	1.0	1	2.0	2.0	5.0	5.0	2.0	3.0	...	1	5700	32490000	600
1	100	0	NaN	2	2.0	2.0	1.0	1.0	2.0	2.0	...	1	2350	5522500	1100
2	1000	0	NaN	1	2.0	2.0	1.0	5.0	2.0	3.0	...	1	7500	56250000	1600
3	1004	0	1.0	1	1.0	2.0	1.0	4.0	5.0	1.0	...	1	9600	92160000	1700
4	1007	0	1.0	5	1.0	2.0	1.0	3.0	4.0	3.0	...	0	400	160000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1724	994	1	1.0	1	1.0	2.0	1.0	3.0	3.0	3.0	...	1	35100	1232010000	8500
1725	995	1	1.0	5	2.0	2.0	1.0	1.0	6.0	1.0	...	0	0	0	0
1726	996	0	1.0	7	2.0	2.0	1.0	1.0	5.0	1.0	...	1	300	90000	100
1727	997	0	1.0	6	1.0	2.0	1.0	1.0	1.0	1.0	...	1	3300	10890000	1800
1728	999	0	NaN	5	2.0	2.0	1.0	5.0	7.0	3.0	...	1	1100	1210000	900

1729 rows × 2830 columns

The merged dataframe contains duplicate columns indicated by the "_x" or "_y" suffixes attached to certain column names. The next two functions remove duplicate columns and cleans up the remaining names.

In [4]:
```python
def drop_y_columns(dataframe):
    """Drops duplicate columns."""

    df = dataframe.copy()

    cols_to_drop = [col for col in df if col.endswith('_y')]
    df = df.drop(cols_to_drop, 1)

    return df


def colunm_renamer(dataframe):
    """Removes '_x' from column names after merging."""

    df = dataframe.copy()

    col_names = [col for col in df.columns.values]
    new_names = [col_name[:-2] if col_name.endswith('_x') else col_name for col_name in col_names]
    df.columns = new_names

    return df
```

In [5]:
```python
df = drop_y_columns(df)
df = colunm_renamer(df)
df
```

Out[5]:

	sampleid	e	cflag	longtdec	b_aidst	gender	ethnic	marital	afdctime	afdchild	...	nkids0	nkids1	nkids2	nkidsge3	ageykid
0	1	0	1.0	1	2.0	2.0	5.0	5.0	2.0	3.0	...	0.0	1.0	0.0	0.0	10.0
1	100	0	NaN	2	2.0	2.0	1.0	1.0	2.0	2.0	...	0.0	0.0	0.0	1.0	11.0
2	1000	0	NaN	1	2.0	2.0	1.0	5.0	2.0	3.0	...	0.0	0.0	1.0	0.0	1.0
3	1004	0	1.0	1	1.0	2.0	1.0	4.0	5.0	1.0	...	0.0	0.0	0.0	1.0	7.0
4	1007	0	1.0	5	1.0	2.0	1.0	3.0	4.0	3.0	...	0.0	0.0	0.0	1.0	5.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1724	994	1	1.0	1	1.0	2.0	1.0	3.0	3.0	3.0	...	0.0	0.0	0.0	1.0	6.0
1725	995	1	1.0	5	2.0	2.0	1.0	1.0	6.0	1.0	...	0.0	0.0	0.0	1.0	6.0
1726	996	0	1.0	7	2.0	2.0	1.0	1.0	5.0	1.0	...	0.0	1.0	0.0	0.0	8.0
1727	997	0	1.0	6	1.0	2.0	1.0	1.0	1.0	1.0	...	0.0	0.0	0.0	1.0	1.0
1728	999	0	NaN	5	2.0	2.0	1.0	5.0	7.0	3.0	...	1.0	0.0	0.0	0.0	0.0

1729 rows × 1982 columns

The cleaned dataframe contains 1729 rows or observations for 1729 unique families.

Understanding Key Variables + Summary Statistics

- `e` is the treatment dummy where 0 means that a family was randomly assigned to the control group and 1 means that a family was randomly assigned to the treatment group. The treatment group had their benefits time limited - in addition to receiving a variety benefits that is beyond the analysis scope of this notebook - of while the control group did not.
- `fmi2` is from the survey data. Families were asked if they were believed to have been subject to the time limit or not. Possible responses also include "don't know" or "no response".

First, let's get a broad overview of how many people believed that they were subject to the time limit versus those who did not believe that they were subject to the time limit.

In [6]:
```python
def summary_stats(dataframe):
    """Returns a dataframe showing how many people believed in the time limit
    vs. how many people did not.
    """

    df = dataframe.copy()

    categories = [
        'Believed Subject to Time Limit',
        "Didn't Believe Subject to Time Limit",
        "Don't Know"
    ]

    sum_table = pd.DataFrame({'CATEGORY': categories,
                              'COUNTS': df['fmi2'].value_counts()})

    sum_table = sum_table.append({'CATEGORY': 'Valid Responses',
                                  'COUNTS': len(df['fmi2'])},
                                  ignore_index=True)

    return sum_table
```

In [7]:
```python
summary_stats(df)
```

Out[7]:

	CATEGORY	COUNTS
0	Believed Subject to Time Limit	666
1	Didn't Believe Subject to Time Limit	365
2	Don't Know	118
3	Valid Responses	1729

For this analysis, I will only be working with observations where the family believed that they were subject to the time limit or the opposite. I will not be working with those who don't know. I'll subset the data and create a new dummy variable for these people. The new dummy variable is referred to as `TLyes`.

In [8]:
```python
def new_dummy_creator(dataframe):
    """Creates a new treatment variable. 1 for those who believed in time limit.
    0 for those who did not. Everyone else is dropped.
    """

    df = dataframe.copy()

    df = df[(df['fmi2'] == 1) | (df['fmi2'] == 2)]

    df['TLyes'] = [1 if val == 1 else 0 for val in df['fmi2']]

    return df
```

In [9]:
```python
df = new_dummy_creator(df)
```

Next, I'd like to get a sense of whether or not there was confusion around the time limit. While a family may have been randomly assigned to the treatment group, and therefore had their benefits time limited, it's possible that the family may have not believed that they were subject to the time limit and vice versa. Below, I cross-tabulate the assignment variable `e` against the self-reported belief variable `TLyes`.

In [10]:
```python
def xtab_generator(dataframe):
    """Generates crosstabs of original dummy variable vs. new dummy variable."""

    df = dataframe.copy()

    tabs = pd.crosstab(index=df['e'], columns=df['TLyes'],
                       margins=True, margins_name='Total',
                       rownames=['Original Treatment'],
                       colnames=['Time Limit Belief'])

    return tabs
```

In [11]:
```python
xtab_generator(df)
```

Out[11]:

Time Limit Belief	0	1	Total
Original Treatment			
0	300	205	505
1	65	461	526
Total	365	666	1031

From the table above, it's clear that participants were confused as to whether or not the time limit applied to their families. Of the 505 families originally assigned to control, roughly 60% were correct in identifying that the time limit did not apply to them. However, a plurality (40.6%) incorrectly thought that their benefits were time limited when then they in reality were not. Participants assigned to the treatment group better understood the guidelines dictating their benefits as 87.6% of these 562 families correctly identified that their benefits were time limited. Conversely, the remaining 12.4% thought that they had no limits when, in reality, they did.

Estimating Effects via Ordinary Least Squares (OLS)

It's possible to determine the effect of the time limit on well-being with a simple OLS regression. There are a number of covariates which contain missing data. I'll impute the means for each one of these controls where there is a `NaN`.

In [12]:
```python
def mean_imputer(dataframe):
    """Takes in the admin and survey merged data and returns a dataframe with
    covariate columns containing NA values filled with the mean of that column.
    """

    df = dataframe.copy()

    columns = [
        'male',
        'agelt20',
        'age2534',
        'age3544',
        'agege45',
        'black',
        'hisp',
        'otheth',
        'martog',
        'marapt',
        'nohsged',
        'applcant',
    ]
    df[columns] = df[columns].fillna(df[columns].mean())

    return df
```

In [13]:
```python
df = mean_imputer(df)
```

With no more missing data, I'm going to create a helper function that retrieves paramaters such as the estimated Betas and standard errors from statsmodels's regression output. As I'm running multiple regressions, the output will be more clearly summarized in a new dataframe.

In [14]:
```python
def paramater_retriever(list_object, parameter):
    """Retrieves a specified parameter from ols regression output."""

    if parameter == 'coefficients':
        values = [item.params for item in list_object]
    elif parameter == 'standard error':
        values = [item.bse for item in list_object]
    elif parameter == 'pvalues':
        values = [item.pvalues for item in list_object]
    elif parameter == 'conf_int_low':
        values = [item.conf_int()[0] for item in list_object]
    elif parameter == 'conf_int_high':
        values = [item.conf_int()[1] for item in list_object]

    values = [value[1] for value in values]

    return values
```

In [15]:
```python
def time_limit_ols(dataframe):
    """Runs OLS to estimate effect of believing in the time limit on welfare
    receipt during years 1-4 of the sample period.
    """

    df = dataframe.copy()

    welfare_vars = [
        'vrecc217',
        'vrecc2t5',
        'vrecc6t9',
        'vrec1013',
        'vrec1417',
    ]
    ind_vars = [
        'TLyes',
        'male',
        'agelt20',
        'age2534',
        'age3544',
        'agege45',
        'black',
        'hisp',
        'otheth',
        'martog',
        'marapt',
        'nohsged',
        'applcant',
        'yremp',
        'emppq1',
        'yrearn',
        'yrearnsq',
        'pearn1',
        'recpc1',
        'yrrec',
        'yrkrec',
        'rfspc1',
        'yrrfs',
        'yrkrfs',
    ]
    right_hand_side = ' + '.join([var for var in ind_vars])

    formulas = [var + ' ~ ' + right_hand_side for var in welfare_vars]
    regressions = [smf.ols(formula=formula, data=df).fit() for formula in formulas]

    ols_results = pd.DataFrame({
        'Welfare_Variable': welfare_vars,
        'Coefficient': paramater_retriever(regressions, 'coefficients'),
        'Std_Error': paramater_retriever(regressions, 'standard error'),
        'p_value': paramater_retriever(regressions, 'pvalues'),
        'Conf_Low': paramater_retriever(regressions, 'conf_int_low'),
        'Conf_High': paramater_retriever(regressions, 'conf_int_high')})

    ols_results['t_stat'] = ols_results['Coefficient'] / ols_results['Std_Error']

    ols_results = ols_results[[
        'Welfare_Variable',
        'Coefficient',
        'Std_Error',
        't_stat',
        'p_value',
        'Conf_Low',
        'Conf_High'
    ]]

    return ols_results
```

In [16]:
```python
time_limit_ols(df)
```

Out[16]:

	Welfare_Variable	Coefficient	Std_Error	t_stat	p_value	Conf_Low	Conf_High
0	vrecc217	0.027352	0.017174	1.592680	0.111546	-0.006348	0.061053
1	vrecc2t5	0.034013	0.019017	1.788532	0.073991	-0.003305	0.071330
2	vrecc6t9	0.007862	0.027806	0.282743	0.777432	-0.046702	0.062426
3	vrec1013	-0.016129	0.031071	-0.519116	0.603794	-0.077101	0.044842
4	vrec1417	-0.072005	0.031382	-2.294493	0.021967	-0.133587	-0.010424

The table above returns OLS results where the `Welfare_Variable` column refers to binary variables that indicate whether a family ever received benefits throughout different points over the four year study period. The `Coefficient` column contains the estimated Betas of believing in the time limit. The data suggest that in year three (`vrec1013`) and year four (`vrec1417`), families who believed in the time limit were less roughly 1% and 7% less likely to ever receive benefits relative to families who did not believe in the time limit.

The interpretation is plausible. If I believed that my benefits were restricted after a certain period of time, then I would be less likely to rely on these benefits as time passes. Though the interpretation is logical, this doesn't guarantee that our estimates are *correct*. In the next section, I explain problems with using ordinary least squares.

Limitations of Ordinary Least Squares

Ordinary Least Squares relies on the assumption that the independent variable is *exogenous* for valid estimates. However, the treatment variable is `TLyes`, and this variable is *endogenous*. As seen in the Understanding Key Variables + Summary Statistics section, there was clearly some confusion about who the time limits actually applied to. There may have been genuine confusion as to the treatment received by the treated, but families may also be *self-selecting* into treatment and control groups.

In other words, because individuals seek to maximize their outcomes, individuals and families may self-select into the group that does not believe in time limited benefits. Arguably, it's better to receive unlimited benefits as opposed to limited benefits, so individuals may not believe in the time limit even if they were randomly assigned to the treatment group which imposed a time limit.

In summary, OLS will not return consistent estimates of the effect of believing in the time limit. In the next section, I explore the possibility of using an instrumental variable and elaborate on the required assumptions.

Instrumental Variable Approach

When the explanatory variable (in this case `TLyes`) is correlated with the error term, OLS will return biased results. The instrumental variable introduces a third variable that changes the explanatory variable but has no direct effect on the dependent variable. It is only through the explanatory variable that the instrument has an effect on the dependent variable.

In this scenario, we need a third variable, Z, that influences the exogenous part of treatment `TLyes` (or the part that is not correlated with the endogenous part of the error term). By this logic, Z must be uncorrelated with the error term.

In the following subsections, I will review the assumptions necessary to use an instrumental variable and evaluate whether or not they hold up in this context. I propose using `e` or the original experimental treatment indicator as an instrument for `TLyes`.

1. Exclusion Restriction

The instrument must not be correlated with the error term and must be exogenous. Mathematically speaking, we can say that $cor(z, u) = 0$ and $E(Y_{0i}|Z = 1, D = 0) = E(Y_{0i}|Z = 0, D = 0)$. In other words, the untreated potential outcome for an individual when the instrument is turned on is equal to the untreated potential outcome for an individual when the instrument is turned off. The instrument itself has no effect on the potential outcome conditional on the same unit being assigned to control or treatment.

This condition is fundamentally untestable because we never observe the error term, however, it is *likely met*. Because the original experimental dummy (`e`) is exogenous through random assignment, it's plausible that the assignment itself has no direct effect on any of the dependent variables. It is only through the belief (or disbelief) in the time limit (`TLyes`) that there is an effect on the variables of interest.

2. Relevance

The instrument must be correlated with the treatment dummy.

In this scenario, the assignment to treatment or control must be correlated with the belief in the time limit. If the instrument changes, then we also expect the belief in the time limit to also change.

This condition is *satisfied*. Below, I test the correlation between the two variables and test against the decision rule. When using an instrumental variable, the decision rule states that our F-stat must be greater than 10 when running a first-stage regression where the endogenous variable is explained by the instrument (`TLyes` ~ `e`).

In [17]:
```python
def correlation_test(dataframe, var1, var2):
    """Returns Pearson's R coefficient."""

    df = dataframe.copy()

    correlation, _ = pearsonr(df[var1], df[var2])

    return correlation


def first_stage_test(dataframe):
    """Prints regression output where TLyes is explained by e."""

    df = dataframe.copy()

    output = smf.ols(formula='TLyes ~ e', data=df).fit().summary()

    print(output)
```

In [24]:
```python
correlation_test(df, 'e', 'TLyes')
```

Out[24]: 0.49181417014926404

Pearson's R is about .5, suggesting that when `e` moves, so does `TLyes`.

In [25]:
```python
first_stage_test(df)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  TLyes   R-squared:                       0.242
Model:                            OLS   Adj. R-squared:                  0.241
Method:                 Least Squares   F-statistic:                     328.3
Date:                Mon, 04 Jan 2021   Prob (F-statistic):           6.72e-64
Time:                        14:23:10   Log-Likelihood:                -559.62
No. Observations:                1031   AIC:                             1123.
Df Residuals:                    1029   BIC:                             1133.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.4059      0.019     21.887      0.000       0.370       0.442
e              0.4705      0.026     18.119      0.000       0.420       0.521
==============================================================================
Omnibus:                       57.896   Durbin-Watson:                   2.090
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               31.800
Skew:                          -0.268   Prob(JB):                     1.24e-07
Kurtosis:                       2.328   Cond. No.                         2.64
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

From the output above, the t-stat is roughly 18. The F-stat is simply the t-stat squared, and 18^2 is clearly greater than 10.

Instrumental Variable Estimation

In this section, I estimate the effect of believing in the time limit on employment, welfare receipt, and income outcomes where `e` is the instrument for `TLyes`. The dataframe below displays the results for each regression.

In [26]:
```python
def iv_estimation(dataframe):
    """Takes in a dataframe and returns estimates where random assignment
    serves as an instrument for believing in the time limit.
    """

    df = dataframe.copy()

    dependent_variables = [
        # Employment variables
        'vempq217',
        'vempq2t5',
        'vempq6t9',
        'vemp1013',
        'vemp1417',
        # Welfare variables
        'vrecc217',
        'vrecc2t5',
        'vrecc6t9',
        'vrec1013',
        'vrec1417',
        # Income variables
        'tinc217',
        'tinc2t5',
        'tinc6t9',
        'tinc1013',
        'tinc1417'
    ]

    df['CONSTANT'] = 1
    controls = [
        'male',
        'agelt20',
        'age2534',
        'age3544',
        'agege45',
        'black',
        'hisp',
        'otheth',
        'martog',
        'marapt',
        'nohsged',
        'applcant',
        'yremp',
        'emppq1',
        'yrearn',
        'yrearnsq',
        'pearn1',
        'recpc1',
        'yrrec',
        'yrkrec',
        'rfspc1',
        'yrrfs',
        'yrkrfs',
        'CONSTANT'
    ]

    iv_models = [IV2SLS(df[dep_var], df[controls], df['TLyes'], df['e']).fit() for dep_var in dependent
_variables]

    iv_results = pd.DataFrame({'Variable': dependent_variables,
                               'Coefficient': [model.params[-1] for model in iv_models],
                               'Std_Error': [model.std_errors[-1] for model in iv_models],
                               't_stat': [model.tstats[-1] for model in iv_models],
                               'p_value': [model.pvalues[-1] for model in iv_models]})

    return iv_results
```

In [29]:
```python
iv_estimation(df)
```

Out[29]:

	Variable	Coefficient	Std_Error	t_stat	p_value
0	vempq217	0.054892	0.042294	1.297855	1.943373e-01
1	vempq2t5	0.069899	0.059466	1.175448	2.398153e-01
2	vempq6t9	0.236567	0.060066	3.938420	8.201985e-05
3	vemp1013	0.267877	0.059908	4.471472	7.768306e-06
4	vemp1417	0.079870	0.059562	1.340957	1.799343e-01
5	vrecc217	0.002752	0.035092	0.078411	9.375009e-01
6	vrecc2t5	0.008770	0.038303	0.228966	8.188954e-01
7	vrecc6t9	-0.014991	0.055916	-0.268096	7.886254e-01
8	vrec1013	-0.131234	0.063073	-2.080680	3.746320e-02
9	vrec1417	-0.398856	0.066351	-6.011292	1.840502e-09
10	tinc217	4699.456443	2063.357267	2.277578	2.275175e-02
11	tinc2t5	292.806157	480.596033	0.609256	5.423546e-01
12	tinc6t9	1154.322187	595.716788	1.937703	5.265947e-02
13	tinc1013	2002.333511	693.309335	2.888081	3.876001e-03
14	tinc1417	1249.994589	785.365854	1.591608	1.114728e-01

Effect on Employment

In analyzing the employment variables (those that start with "vemp"), the IV estimates reveal that those who believed in the time limit were more likely to be employed throughout the study. However, these differences are only statistically significant throughout the middle of the study. By the end, employment levels were no different - statistically speaking - between those who believed in the time limit versus those who did not.

Effect on Welfare Receipt

Over the course of the entire study, when looking at welfare receipt variables (those that start with "vrec"), there was no statistical difference in the amount of welfare received between those who believed in the time limit and those who did not (variable `vrecc217`). However, when looking at the period specific effects, by year three (`vrec1013`) and year four (variable `vrec1417`), those who believed in the time limit were between 13% and 40% more likely to be employed than those who did not believe in the limit. These differences are statistically significant at the 95% level of confidence.

When comparing these results to the OLS results above, it's clear that OLS has not only underestimated the effect of believing in the time limit in year three and year four, but OLS has also misidentified statistical significance.

A note on standard errors:

Finally, when contrasting the IV estimates to OLS, it's clear that IV estimates will always return *larger* standard errors compared to OLS. In this scenario, the standard errors are larger by about two-fold. Intuitively, this makes sense because we're using less data to explain outcomes. We're using the variation in the perceived time limit explained by the assignment to treatment or control that is uncorrelated with the disturbance term. Since we're working with less data, we are less certain of our results and, consequently, the standard errors increase.

Effect on Income

Income is the last outcome of interest (variables starting with "tinc"). The data show that those who believed in the time limit had greater levels of income than those who did not. However, the increase is only statistically different from zero in year three of the study (`tinc1013`).