ps_6

March 14, 2019

1 Problem Set 6: Welfare Instrumental Variables

In [1]:
```
import numpy as np
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt
import seaborn as sns
import requests
from IPython.display import display
import statsmodels.api as sm
import statsmodels.formula.api as smf
from linearmodels.iv import IV2SLS
%matplotlib inline
sns.set_style('darkgrid')
```

In [2]:
```
def fetch_and_cache(data_url, file, data_dir="data", force = False):
    """
    (Credit: John DeNero)
    Download and cache a url and return the file object.
    Dependent: from pathlib import Path
    import requests
    data_url: the web address to download
    file: the file in which to save the results.
    data_dir: (default="data") the location to save the data
    force: if true the file is always re-downloaded
    return: The pathlib.Path to the file.
    """
    data_dir = Path(data_dir)
    data_dir.mkdir(exist_ok=True)
    file_path = data_dir/Path(file)
    if force and file_path.exists():
        file_path.unlink()
    if force or not file_path.exists():
        print('Downloading...', end = ' ')
        resp = requests.get(data_url)
        with file_path.open('wb') as f:
            f.write(resp.content)
```

```
            print('Done!')
        else:
            import time
            created = time.ctime(file_path.stat().st_ctime)
            print("Using cached version downloaded at", created)
        return file_path

    data_urls = r"https://bcourses.berkeley.edu/files/74717033/download?download_frd=1"
    file_names = r"welfare.csv"

    file = fetch_and_cache(data_urls, file_names)

    welfare_df = pd.read_csv(file)
    display(welfare_df.head())
```

```
Using cached version downloaded at Fri Mar  8 14:20:01 2019
```

	imm	hsgrad	agelt25	age35p	treatment	working_at_baseline	anykidsu6
0	0	0	0	0	1	0	1
1	0	1	0	1	1	0	0
2	0	0	0	0	1	0	0
3	0	0	0	0	0	1	0
4	0	1	0	0	0	0	1

	nevermarried	ft15	ft20	ft24	ft48	welfare15	welfare20	welfare24
0	1	0	0.0	0.0	0.0	1	1	1
1	1	1	1.0	1.0	NaN	0	0	0
2	1	0	0.0	0.0	0.0	1	1	1
3	0	0	0.0	0.0	0.0	1	0	0
4	1	0	0.0	0.0	0.0	1	1	1

	welfare48
0	1
1	0
2	0
3	0
4	1

```
In [3]: welfare_df.dropna(inplace = True)
```

1.1 1)

We are interested in casual model of working full time affects welfare participation. Our casual model is

$$(1)\ y_i = \beta_0 + \beta_1 FT_i + u_i$$

where y_i is an indicator equal to 1 if parent i is on welfare in period t, and FT_i is indicator for full time status in the month. We are going to use "assigned to the treatment group" as our instrument, so $z_i = T_i := Treatment_i$. We will allow for different coefficients for different time horizon effects.

```
In [4]: months = [15, 20, 24, 48]

        fts = ["ft{}".format(month) for month in months]
        welfares = ["welfare{}".format(month) for month in months]
```

1.1.1 a)

Estimate first stage models for the probability of working FT in months 15, 20, 24, 48, using treatment as the instrument and no other controls.

$$FT_i = \pi_0 + \pi_1 T_i + \eta_i$$

There will be separate estimates of π_0, π_1 for each of months 15, 20, 24, 48.

```
In [5]: models = ["{} ~ 1 + treatment".format(ft) for ft in fts]

        results = []
        for model in models:
            results.append(smf.ols(formula = model, data = welfare_df, missing = 'drop').fit()
            #Must drop missing values since linearmodels (which is being used for IV estimatio

        pis = pd.DataFrame(
            {ft:result.params for ft, result in zip(fts, results)}
        )
        display(pis)
```

	ft15	ft20	ft24	ft48
Intercept	0.151399	0.162850	0.161154	0.233673
treatment	0.142694	0.114427	0.107509	0.050577

1.1.2 b)

Estimate reduced form models for the probability of being on welfare in months 15, 20, 24, 48, using treament as the instrumental variable and no other control variables.

$$y_i = \gamma_0 + \gamma_1 T_i + \nu_i$$

There will be separate estimates of γ_0, γ_1 for each of months 15, 20, 24, 48.

```
In [6]: models = ["{} ~ 1 + treatment".format(welfare)
                  for welfare in welfares]

        results = []
        for model in models:
            results.append(smf.ols(formula = model, data = welfare_df, missing = 'drop').fit()
```

```
gammas = pd.DataFrame(
    {welfare:result.params for welfare, result in zip(welfares, results)}
)
display(gammas)
```

	welfare15	welfare20	welfare24	welfare48
Intercept	0.809584	0.769720	0.738762	0.591603
treatment	-0.147977	-0.122058	-0.107507	-0.041972

1.1.3 c)

Esimate the casual model (1) by OLS for each of months 15,20,24,48.

```
In [7]: models = ["{} ~ 1 + {}".format(welfare, ft)
                  for welfare, ft in zip(welfares, fts)]

        results = []
        for model in models:
            results.append(smf.ols(formula = model, data = welfare_df, missing = "drop").fit()

        ols_betas = pd.DataFrame(
            {welfare:result.params for welfare, result in zip(welfares, results)}
        )
        display(ols_betas)
```

	welfare15	welfare20	welfare24	welfare48
Intercept	0.861634	0.832976	0.798990	0.708896
ft15	-0.568337	NaN	NaN	NaN
ft20	NaN	-0.566939	NaN	NaN
ft24	NaN	NaN	-0.532323	NaN
ft48	NaN	NaN	NaN	-0.534459

1.1.4 d)

Estimate the causal model by IV in each of months 15, 20, 24, 48.

```
In [8]: models = ["{0} ~ 1 + [{1} ~ treatment]".format(welfare, ft)
                  for welfare, ft in zip(welfares, fts)]

        results = []
        for model in models:
            results.append(IV2SLS.from_formula(
                formula = model, data = welfare_df
            ).fit())

        betas = pd.DataFrame(
```

```
            {welfare:result.params for welfare, result in zip(welfares, results)}
        )
        display(betas)
```

	welfare15	welfare20	welfare24	welfare48
Intercept	0.966589	0.943431	0.899911	0.785521
ft15	-1.037020	NaN	NaN	NaN
ft20	NaN	-1.066693	NaN	NaN
ft24	NaN	NaN	-0.999974	NaN
ft48	NaN	NaN	NaN	-0.829871

1.1.5 e)

Verify that in each month, $\hat{\beta}_1^{IV} = \hat{\gamma}_1/\hat{\pi}_1$.

```
In [9]: for welfare, ft, month in zip(welfares, fts, months):
            g_over_p = gammas[welfare].loc["treatment"] / pis[ft].loc["treatment"]
            print("for month {2}:\n beta_hat: {0} \n gamma_pi_ratio: {1}" .format(
                betas[welfare].loc[ft], g_over_p, month))
```

```
for month 15:
 beta_hat: -1.037019699629429
 gamma_pi_ratio: -1.0370196996294272
for month 20:
 beta_hat: -1.0666934218687558
 gamma_pi_ratio: -1.0666934218687738
for month 24:
 beta_hat: -0.9999741121268855
 gamma_pi_ratio: -0.9999741121268908
for month 48:
 beta_hat: -0.8298710946635879
 gamma_pi_ratio: -0.8298710946635985
```

We clearly see that $\hat{\beta}_1^{IV} = \hat{\gamma}_1/\hat{\pi}_1$ holds in every month observed.

1.2 2)

Now repeat steps (a) - (e) from question (1) for month 15 only, but including as controls the variables {imm, hsgrad, agelt25, age35p, working_at_baseline, anykidsu6, nevermarried} in your first stage model, your reduced form model, and your OLS and IV versions of the causal model. Because SSP was a randomized experiment, T_i is (approximately) uncorrelated with all these control variables. Does their addition affect your different estimates?

```
In [10]: additional_variables = "imm + hsgrad + agelt25 + age35p + working_at_baseline + anyki
```

1.2.1 a)

```
In [11]: models = ["{} ~ 1 + treatment + {}".format(ft, additional_variables) for ft in fts]

         results = []
         for model in models:
             results.append(smf.ols(formula = model, data = welfare_df, missing = 'drop').fit(
             #Must drop missing values since linearmodels (which is being used for IV estimati

         pis = pd.DataFrame(
             {ft:result.params for ft, result in zip(fts, results)}
         )
         display(pis)
```

	ft15	ft20	ft24	ft48
Intercept	0.089979	0.080885	0.092306	0.157485
treatment	0.141824	0.113821	0.106826	0.048997
imm	-0.052285	-0.025376	-0.028026	-0.014446
hsgrad	0.092977	0.100354	0.101047	0.128781
agelt25	0.014932	0.021334	0.037112	0.047044
age35p	-0.042554	-0.022191	-0.036641	-0.032199
working_at_baseline	0.271514	0.244647	0.231532	0.190155
anykidsu6	-0.021173	-0.007451	-0.011784	-0.035446
nevermarried	-0.009805	-0.002040	-0.015590	0.006294

1.2.2 b)

```
In [12]: models = ["{} ~ 1 + treatment + {}".format(welfare, additional_variables)
                   for welfare in welfares]

         results = []
         for model in models:
             results.append(smf.ols(formula = model, data = welfare_df, missing = 'drop').fit(

         gammas = pd.DataFrame(
             {welfare:result.params for welfare, result in zip(welfares, results)}
         )
         display(gammas)
```

	welfare15	welfare20	welfare24	welfare48
Intercept	0.853086	0.816950	0.795875	0.649518
treatment	-0.146921	-0.120974	-0.106730	-0.039979
imm	0.035602	0.046574	0.057187	0.028725
hsgrad	-0.106124	-0.114205	-0.122399	-0.156687
agelt25	-0.029367	-0.033101	-0.042715	-0.090572
age35p	0.062799	0.053035	0.045522	0.050122
working_at_baseline	-0.222330	-0.223736	-0.218009	-0.193014
anykidsu6	0.017565	0.012456	-0.009699	0.046499

```
nevermarried          0.040659   0.051376   0.066910   0.047003
```

1.2.3 c)

```
In [13]: models = ["{} ~ 1 + {} + {}".format(welfare, ft, additional_variables)
                   for welfare, ft in zip(welfares, fts)]

         results = []
         for model in models:
             results.append(smf.ols(formula = model, data = welfare_df, missing = "drop").fit(

         ols_betas = pd.DataFrame(
             {welfare:result.params for welfare, result in zip(welfares, results)}
         )
         display(ols_betas)
```

	welfare15	welfare20	welfare24	welfare48
Intercept	0.864487	0.828490	0.813167	0.719825
age35p	0.041043	0.042165	0.028297	0.034329
agelt25	-0.020830	-0.021300	-0.024015	-0.067026
anykidsu6	0.008149	0.010162	-0.014012	0.029275
ft15	-0.537471	NaN	NaN	NaN
ft20	NaN	-0.532697	NaN	NaN
ft24	NaN	NaN	-0.493343	NaN
ft48	NaN	NaN	NaN	-0.498130
hsgrad	-0.057406	-0.061818	-0.073507	-0.092814
imm	0.008339	0.033772	0.044002	0.021714
nevermarried	0.035033	0.049985	0.058946	0.050059
working_at_baseline	-0.075483	-0.092631	-0.103084	-0.098090

1.2.4 d)

```
In [14]: models = ["{0} ~ 1 + [{1} ~ treatment] + {2}".format(welfare, ft, additional_variable
                   for welfare, ft in zip(welfares, fts)]

         results = []
         for model in models:
             results.append(IV2SLS.from_formula(
                 formula = model, data = welfare_df
             ).fit())

         betas = pd.DataFrame(
             {welfare:result.params for welfare, result in zip(welfares, results)}
         )
         display(betas)
```

```
                      welfare15  welfare20  welfare24  welfare48
```

```
Intercept             0.946299   0.902918   0.888098   0.778018
age35p                0.018716   0.029449   0.008914   0.023850
agelt25              -0.013898  -0.010426  -0.005636  -0.052187
anykidsu6            -0.004368   0.004536  -0.021472   0.017577
ft15                 -1.035941        NaN        NaN        NaN
ft20                       NaN  -1.062845        NaN        NaN
ft24                       NaN        NaN  -0.999099        NaN
ft48                       NaN        NaN        NaN  -0.815952
hsgrad               -0.009806  -0.007545  -0.021443  -0.051608
imm                  -0.018563   0.019603   0.029186   0.016938
nevermarried          0.030502   0.049208   0.051333   0.052138
working_at_baseline   0.058942   0.036285   0.013314  -0.037857
```

1.2.5 e)

```
In [15]: for welfare, ft, month in zip(welfares, fts, months):
             g_over_p = gammas[welfare].loc["treatment"] / pis[ft].loc["treatment"]
             print("for month {2}:\n beta_hat: {0} \n gamma_pi_ratio: {1}" .format(
                 betas[welfare].loc[ft], g_over_p, month))
```

```
for month 15:
 beta_hat: -1.0359407430749634
 gamma_pi_ratio: -1.0359407430749892
for month 20:
 beta_hat: -1.0628446924873316
 gamma_pi_ratio: -1.0628446924873447
for month 24:
 beta_hat: -0.9990992195414421
 gamma_pi_ratio: -0.9990992195414519
for month 48:
 beta_hat: -0.8159517190649126
 gamma_pi_ratio: -0.8159517190650901
```

The inclusion of the control variables seems to have negligible impact on our estimates.