STAR_notebook-percentileoutcome

February 11, 2023

[1]:
```
import dame_flame
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import heapq
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.preprocessing import LabelBinarizer
from sklearn.linear_model import LinearRegression
from scipy import stats
```

```
C:\Users\Neha\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[2]:
```
STAR_Students = pd.read_spss('STAR_Students.sav')
```

[3]:
```
#STAR_Students['gksurban'].value_counts()
# d = {"RURAL": 0, "URBAN":1, "SUBURBAN": 2, "INNER CITY": 3}

df_trunc = STAR_Students.loc[:, STAR_Students.columns.intersection(
    ['gkclasstype', 'gender', 'race', 'gkfreelunch', 'gkschid', 'gktmathss',␣
 ↪'gktreadss', 'g1freelunch', 'g2freelunch', 'g3freelunch'])]


d = {"WHITE": 1, "BLACK": 0, "ASIAN": 1, "HISPANIC": 0, "OTHER": 0,
     "NATIVE AMERICAN": 0}
df_trunc['race'] = df_trunc['race'].map(d)

d = {"NON-FREE LUNCH": 0, "FREE LUNCH": 1}
df_trunc['gkfreelunch'] = df_trunc['gkfreelunch'].map(d)
df_trunc['g1freelunch'] = df_trunc['g1freelunch'].map(d)
df_trunc['g2freelunch'] = df_trunc['g2freelunch'].map(d)
df_trunc['g3freelunch'] = df_trunc['g3freelunch'].map(d)
```

```
d = {"MALE": 1, "FEMALE": 0}
df_trunc['gender'] = df_trunc['gender'].map(d)
#df_trunc['gktgen'] = df_trunc['gktgen'].map(d)

d = {"WHITE": 1, "BLACK": 0}
#df_trunc['gktrace'] = df_trunc['gktrace'].map(d)

d = {"SMALL CLASS": int(1), "REGULAR CLASS": int(0),
     "REGULAR + AIDE CLASS": int(0)}
df_trunc['ksmall'] = df_trunc['gkclasstype'].map(d)

# df_trunc = df_trunc.dropna().copy()

# Create age variable counting months
#df_trunc['age'] = df_trunc['birthyear']*12 + df_trunc['birthmonth']
df_trunc = df_trunc.drop(columns=['gkclasstype'])
# Bin age into deciles
#df_trunc['age'] = pd.qcut(df_trunc['age'], q=10, labels=False)

df_trunc = df_trunc.rename(columns={"ksmall": "treated"}) ## NOTE TO SELF --␣
 ↪COME BACK TO WE SHOULDNT HAVE TO DO THIS
```

[4]:
```
for i in df_trunc.index:
    if df_trunc.loc[i, 'g1freelunch'] == 1 or df_trunc.loc[i, 'g2freelunch'] ==␣
 ↪1 or df_trunc.loc[i, 'g3freelunch'] == 1 or df_trunc.loc[i, 'gkfreelunch']␣
 ↪== 1:
        df_trunc.loc[i, 'gkfreelunch'] = 1
    else:
        df_trunc.loc[i, 'gkfreelunch'] = 0
```

[5]:
```
df_trunc = df_trunc.drop(columns=['g1freelunch', 'g2freelunch', 'g3freelunch'])
```

[6]:
```
df_trunc=df_trunc.dropna(subset=['treated'])
```

[7]:
```
df_trunc_untreated = df_trunc[df_trunc['treated'] == 0]
df_trunc_treated = df_trunc[df_trunc['treated'] == 1]
```

[8]:
```
for i in df_trunc_treated.index:
    df_trunc_treated.loc[i, 'gktreadss'] = stats.
 ↪percentileofscore(df_trunc_untreated['gktreadss'], df_trunc_treated.
 ↪loc[i,'gktreadss'])
    df_trunc_treated.loc[i, 'gktmathss'] = stats.
 ↪percentileofscore(df_trunc_untreated['gktmathss'], df_trunc_treated.
 ↪loc[i,'gktmathss'])
```

```
C:\Users\Neha\AppData\Roaming\Python\Python36\site-
```

```
packages\pandas\core\indexing.py:1765: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  isetter(loc, value)
```

[9]:
```
# Percentile the math and reading and then average them
# but do different percentiles for the small class size people and the large␣
 ↪people.
df_trunc_untreated = df_trunc[df_trunc['treated'] == 0]
df_trunc_untreated['gktreadss'] = df_trunc_untreated['gktreadss'].
 ↪rank(pct=True)*100
df_trunc_untreated['gktmathss'] = df_trunc_untreated['gktmathss'].
 ↪rank(pct=True)*100
df_trunc_untreated['outcome'] = df_trunc_untreated[['gktreadss', 'gktmathss']].
 ↪mean(axis=1)

#df_trunc_treated = df_trunc[df_trunc['treated'] == 1]
#df_trunc_treated['gktreadss'] = df_trunc_treated['gktreadss'].
 ↪rank(pct=True)*100
#df_trunc_treated['gktmathss'] = df_trunc_treated['gktmathss'].
 ↪rank(pct=True)*100
df_trunc_treated['outcome'] = df_trunc_treated[['gktreadss', 'gktmathss']].
 ↪mean(axis=1)
```

```
C:\Users\Neha\Anaconda3\lib\site-packages\ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
C:\Users\Neha\Anaconda3\lib\site-packages\ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
C:\Users\Neha\Anaconda3\lib\site-packages\ipykernel_launcher.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\Neha\Anaconda3\lib\site-packages\ipykernel_launcher.py:11:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  # This is added back by InteractiveShellApp.init_path()
```

[10]:
```
df = pd.concat([df_trunc_treated, df_trunc_untreated])
df = df.drop(columns=['gktreadss', 'gktmathss'])
```

[11]:
```
df = df.dropna()
```

[12]:
```
# Also extremely wrong because didn't do fixed effects
reg = LinearRegression().fit(df.loc[:, df.columns != 'outcome'], df['outcome'])
```

[13]:
```
#y = df.loc[:,['outcome']]
#x = df.loc[:, ['gender', 'race', 'gkfreelunch', 'treated']]
#x = sm.add_constant(x)

md = smf.mixedlm(formula="outcome ~ gender+race+gkfreelunch+treated", data=df,␣
 ↪groups=df['gkschid'])
md.fit().summary()
```

[13]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
           Mixed Linear Model Regression Results
============================================================
Model:             MixedLM  Dependent Variable:  outcome
No. Observations:  5873     Method:              REML
No. Groups:        79       Scale:               492.2281
Min. group size:   34       Likelihood:          -26651.8008
Max. group size:   138      Converged:           Yes
Mean group size:   74.3
------------------------------------------------------------
                  Coef.  Std.Err.   z    P>|z| [0.025 0.975]
------------------------------------------------------------
Intercept         40.322    1.638 24.610 0.000 37.111 43.533
gender[T.1]       -4.596    0.583 -7.881 0.000 -5.738 -3.453
gkfreelunch[T.0]  12.124    0.701 17.303 0.000 10.750 13.497
race               9.109    1.171  7.778 0.000  6.813 11.404
treated            1.048    0.639  1.639 0.101 -0.205  2.302
```

```
Group Var        142.876    1.094
============================================================

"""
```

[15]:
```
fes = pd.get_dummies(df['gkschid'])
fes = fes.drop(columns=[161183.0])
y = df.loc[:,['outcome']]
x = df.loc[:, ['gender', 'race', 'gkfreelunch', 'treated']]
x = pd.concat([fes,x],axis=1)
x = sm.add_constant(x)
model = sm.OLS(y,x)
model.fit().summary()
```

[15]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  outcome   R-squared:                       0.303
Model:                            OLS   Adj. R-squared:                  0.293
Method:                 Least Squares   F-statistic:                     30.67
Date:                Sat, 11 Feb 2023   Prob (F-statistic):               0.00
Time:                        14:12:17   Log-Likelihood:                -26495.
No. Observations:                5873   AIC:                         5.316e+04
Df Residuals:                    5790   BIC:                         5.371e+04
Df Model:                          82
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          60.4540      2.550     23.708      0.000      55.455      65.453
112038.0      -30.3663      3.777     -8.040      0.000     -37.770     -22.962
123056.0      -17.6622      3.722     -4.745      0.000     -24.959     -10.366
128068.0      -19.3952      3.699     -5.244      0.000     -26.646     -12.144
128076.0      -26.5331      3.622     -7.326      0.000     -33.634     -19.433
128079.0      -27.3936      3.595     -7.621      0.000     -34.441     -20.347
130085.0      -18.6121      3.395     -5.483      0.000     -25.267     -11.957
159171.0        8.0403      3.119      2.577      0.010       1.925      14.156
161176.0      -17.4628      3.303     -5.288      0.000     -23.937     -10.989
162184.0      -13.7533      3.576     -3.846      0.000     -20.764      -6.743
164198.0       -9.0253      3.702     -2.438      0.015     -16.282      -1.769
165199.0        3.5199      3.780      0.931      0.352      -3.891      10.931
166203.0      -19.7707      3.454     -5.723      0.000     -26.543     -12.999
168211.0       -9.7285      3.130     -3.108      0.002     -15.865      -3.592
168214.0        2.3116      3.683      0.628      0.530      -4.909       9.532
169219.0        1.4997      3.868      0.388      0.698      -6.084       9.083
169229.0       -2.5168      2.874     -0.876      0.381      -8.150       3.117
```

169231.0	-27.5091	3.754	-7.327	0.000	-34.869	-20.149
169280.0	-4.2365	3.736	-1.134	0.257	-11.560	3.087
170295.0	-3.5818	3.502	-1.023	0.306	-10.447	3.284
173312.0	12.3096	3.661	3.363	0.001	5.133	19.486
176329.0	3.8031	3.470	1.096	0.273	-3.000	10.606
180344.0	-8.0471	3.161	-2.546	0.011	-14.244	-1.850
189378.0	-23.0814	3.429	-6.732	0.000	-29.803	-16.360
189382.0	-10.9020	3.518	-3.099	0.002	-17.799	-4.005
189396.0	-21.9393	3.573	-6.140	0.000	-28.944	-14.934
191411.0	-4.5879	3.910	-1.173	0.241	-12.253	3.077
193422.0	0.4441	3.615	0.123	0.902	-6.644	7.532
193423.0	-4.4012	3.378	-1.303	0.193	-11.023	2.221
201449.0	2.5399	2.999	0.847	0.397	-3.339	8.419
203452.0	-14.6782	3.151	-4.658	0.000	-20.856	-8.501
203457.0	3.5650	4.071	0.876	0.381	-4.417	11.547
205488.0	-11.5113	3.685	-3.124	0.002	-18.735	-4.288
205489.0	-15.9441	3.418	-4.665	0.000	-22.644	-9.244
205490.0	-29.1020	3.763	-7.733	0.000	-36.480	-21.725
205491.0	-17.4203	3.437	-5.069	0.000	-24.157	-10.683
205492.0	8.3959	3.471	2.419	0.016	1.592	15.200
208501.0	-12.1246	3.542	-3.423	0.001	-19.069	-5.181
208503.0	-27.0787	3.595	-7.533	0.000	-34.126	-20.032
209510.0	-19.2566	3.170	-6.075	0.000	-25.470	-13.043
212522.0	-5.7732	3.167	-1.823	0.068	-11.982	0.435
215533.0	1.4952	2.965	0.504	0.614	-4.318	7.309
216536.0	-13.5881	3.096	-4.389	0.000	-19.658	-7.519
218562.0	-2.1177	3.661	-0.578	0.563	-9.295	5.060
221571.0	-38.3167	3.136	-12.220	0.000	-44.464	-32.170
221574.0	-25.1809	3.571	-7.052	0.000	-32.181	-18.181
225585.0	-19.7043	3.329	-5.920	0.000	-26.229	-13.179
228606.0	-6.7410	3.350	-2.012	0.044	-13.309	-0.174
230612.0	7.8317	3.614	2.167	0.030	0.747	14.916
231616.0	-0.4239	3.654	-0.116	0.908	-7.587	6.739
234628.0	-6.6276	3.126	-2.120	0.034	-12.755	-0.500
244697.0	-16.5775	3.275	-5.061	0.000	-22.999	-10.156
244708.0	-21.8620	3.241	-6.745	0.000	-28.216	-15.508
244723.0	-21.1293	3.255	-6.491	0.000	-27.510	-14.748
244727.0	-3.9904	3.509	-1.137	0.256	-10.870	2.889
244728.0	-16.6571	3.975	-4.191	0.000	-24.449	-8.865
244736.0	11.9808	3.974	3.015	0.003	4.190	19.772
244745.0	6.0689	3.365	1.804	0.071	-0.528	12.666
244746.0	1.9122	3.926	0.487	0.626	-5.785	9.609
244755.0	0.4558	3.115	0.146	0.884	-5.651	6.563
244764.0	-5.9233	4.521	-1.310	0.190	-14.786	2.939
244774.0	-3.5177	3.306	-1.064	0.287	-9.998	2.962
244776.0	-7.7545	3.119	-2.486	0.013	-13.869	-1.640
244780.0	30.0246	3.781	7.942	0.000	22.613	37.436

```
244796.0      -10.1077      3.862     -2.617      0.009     -17.679      -2.536
244799.0      -10.6438      3.717     -2.864      0.004     -17.930      -3.358
244801.0      -12.1337      3.420     -3.547      0.000     -18.839      -5.428
244806.0       18.3390      3.093      5.929      0.000      12.275      24.403
244818.0      -16.6510      3.412     -4.880      0.000     -23.339      -9.963
244831.0      -11.3624      3.645     -3.117      0.002     -18.508      -4.217
244839.0        9.5274      3.382      2.817      0.005       2.898      16.157
252885.0       -4.9612      3.478     -1.427      0.154     -11.779       1.857
253888.0       -9.6905      4.003     -2.421      0.016     -17.537      -1.844
257899.0      -17.8431      3.083     -5.787      0.000     -23.888     -11.798
257905.0       -2.2907      2.948     -0.777      0.437      -8.070       3.488
259915.0      -12.8277      3.634     -3.530      0.000     -19.952      -5.704
261927.0       -8.5114      3.262     -2.609      0.009     -14.906      -2.117
262937.0        5.3416      3.497      1.528      0.127      -1.513      12.196
264945.0       -1.2649      3.144     -0.402      0.687      -7.428       4.898
gender         -4.5507      0.583     -7.802      0.000      -5.694      -3.407
race            9.6029      1.232      7.794      0.000       7.188      12.018
gkfreelunch   -12.1275      0.704    -17.219      0.000     -13.508     -10.747
treated         1.0724      0.640      1.676      0.094      -0.182       2.327
==============================================================================
Omnibus:                      129.749   Durbin-Watson:                   1.942
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               77.836
Skew:                          -0.127   Prob(JB):                     1.25e-17
Kurtosis:                       2.496   Cond. No.                         100.
==============================================================================

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

[16]:
```
# Do mice
from sklearn.tree import DecisionTreeRegressor

from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

imp = IterativeImputer(max_iter=10, random_state=1,
                       estimator=DecisionTreeRegressor())
imp.fit(df)
tmp_df = pd.DataFrame(data=np.round(imp.transform(df)),
                      columns=df.columns,
                      index=df.index)
# convert floats to ints because MICE creates floats
cols = list(df.columns)
cols.remove("outcome")
tmp_df[cols] = tmp_df[cols].astype('int64')
```

[17]:
```
df = tmp_df
```

[18]:
```
len(df)
```

[18]:
```
5873
```

[19]:
```
df.head()
```

[19]:

	gender	race	gkschid	gkfreelunch	treated	outcome
263	0	0	205492	1	1	78.0
281	1	1	189382	0	1	52.0
285	0	1	201449	1	1	76.0
295	0	1	161176	1	1	44.0
308	1	1	189382	0	1	85.0

[20]:
```
# Do the matching

models = []
random_seeds = [1111, 2222, 3333, 4444]
for i in range(4):
    matching_df, holdout_df = train_test_split(df, test_size=0.2,␣
 ↪random_state=random_seeds[i])
    model_dame = dame_flame.matching.DAME(
        repeats=False, verbose=0, adaptive_weights='decisiontree',
        missing_holdout_replace=1, missing_data_replace=1,
        early_stop_pe=True)
    model_dame.fit(holdout_data=holdout_df)
    model_dame.predict(matching_df)
    models.append(model_dame)
```

```
4655 units matched. We finished with no more treated units to match
4660 units matched. We finished with no more treated units to match
4659 units matched. We finished with no more treated units to match
4661 units matched. We finished with no more treated units to match
```

[ ]:
```
for i in range(len(models)):
    ate, var = dame_flame.utils.post_processing.
 ↪var_ATE(matching_object=models[i])
    print("ATE of trial", i, ":", ate,". Variance: ", var)
```

```
treated_col treated
```

[ ]:
```
# compute stuff for plot
# Create the plot
match_dfs = []
for i in models:
    match_dfs.append(i.input_data)
```

```
for i in range(4):
    colname = 'cates'
    match_dfs[i][colname] = dame_flame.utils.post_processing.CATE(
        models[i], match_dfs[i].index)

dame_len_groups = []
dame_cate_of_groups = []

for i in range(4):

    model_dame = models[i]
    groups = list(range(len(model_dame.units_per_group)))

    dame_cate_of_group = []
    dame_len_group = []
    dame_len_treated = []
    maxcate = 0.0
    maxgroupnum = 0
    index = 0

    flame_cate_of_group = []
    flame_len_group = []
    large_groups = []
    for group in model_dame.units_per_group:
        dame_cate_of_group.append(dame_flame.utils.post_processing.CATE(
            model_dame, group[0]))
        dame_len_group.append(len(group))

        # find len of just treated units
        df_mmg = df.loc[group]
        treated = df_mmg.loc[df_mmg["treated"] == 1]

    dame_len_groups.append(dame_len_group)
    dame_cate_of_groups.append(dame_cate_of_group)
```

[ ]:
```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize = (19,13),
                                             sharex=True, sharey=True)
fig.text(0.5, 0.05, 'Number of Units in Matched Group', ha='center',
         fontsize=26)
fig.text(0.05, 0.5, 'Estimated Treatment Effect of Matched Group',
         va='center', rotation='vertical', fontsize=26)
fig.suptitle("CATE Estimates from DAME for Four Random Samples from STAR␣
 ↪Dataset", fontsize=28, y=0.91)
ax1.axhline(y=0.0, color='r', linestyle='-')
ax2.axhline(y=0.0, color='r', linestyle='-')
ax3.axhline(y=0.0, color='r', linestyle='-')
```

```
ax4.axhline(y=0.0, color='r', linestyle='-')

ax1.tick_params(labelsize=26)
ax2.tick_params(labelsize=26)
ax3.tick_params(labelsize=26)
ax4.tick_params(labelsize=26)

ax1.scatter(dame_len_groups[0], dame_cate_of_groups[0], color="purple",
            alpha = 0.25)
#ax1.text(0.15, 0.9,'ATE: '+str(round(ates[0],2)), ha='center', va='center',
#         transform=ax1.transAxes, fontsize=26)

ax2.scatter(dame_len_groups[1], dame_cate_of_groups[1], color="green",
            alpha = 0.25)
#ax2.text(0.15, 0.9,'ATE: '+str(round(ates[1],2)), ha='center', va='center',
#         transform=ax2.transAxes, fontsize=26)

ax3.scatter(dame_len_groups[2], dame_cate_of_groups[2], color="blue",
            alpha = 0.25)
#ax3.text(0.15, 0.9,'ATE: '+str(round(ates[2],2)), ha='center', va='center',
#         transform=ax3.transAxes, fontsize=26)

ax4.scatter(dame_len_groups[3], dame_cate_of_groups[3], color="magenta",
            alpha = 0.25)
#ax4.text(0.15, 0.9,'ATE: '+str(round(ates[3],2)), ha='center', va='center',
#         transform=ax4.transAxes, fontsize=26)

plt.subplots_adjust(wspace=.02, hspace=.02)
## plt.savefig('cate-graph4.png', dpi = 200)
```

[ ]:
```
list_star_covars = []
for modelid in range(len(models)):

    # Pull out the groups with 10 or more units in the matched group
    model = models[modelid]
    large_groups = []
    for group in model.units_per_group:
        if len(group) >= 12.5:
            large_groups.append(group)

    covariates = set(models[modelid].input_data.columns) - set(['gktreadss',␣
↪'treated', 'cates'])
    # Which covars did the large group match on?
    star_covars = dict()
    for group in large_groups:
        group_star_covars = []
        matched_df = models[modelid].df_units_and_covars_matched.loc[group]
```

```
        for covar in covariates:
            if '*' in matched_df[covar].values:
                group_star_covars.append(covar)
        cate_of_group = models[modelid].input_data.loc[group[0], 'cates']
        star_covars[cate_of_group] = group_star_covars

    list_star_covars.append(star_covars)
```

[ ]:
```
list_star_covars
```

[ ]:
```
## Check the matched group with the most units in each trial -- also which␣
↪covariates did they use and which units in their MMG?
```

[ ]:
```
## Run DAME and FLAME and show why we chose DAME for this dataset. What happens␣
↪if we run FLAME?
flame_models = []
random_seeds = [1111, 2222, 3333, 4444]
for i in range(4):
    matching_df, holdout_df = train_test_split(df, test_size=0.2,␣
↪random_state=random_seeds[i])
    model_flame = dame_flame.matching.FLAME(
        repeats=False, verbose=3, adaptive_weights='decisiontree',
        missing_holdout_replace=1, missing_data_replace=1,
        early_stop_pe=True)
    model_flame.fit(holdout_data=holdout_df)
    result_flame = model_flame.predict(matching_df)
    flame_models.append(model_flame)
```

[ ]:
```
# whats the var of the ates?
for model in fmodels:
    print(dame_flame.utils.post_processing.var_ATE(matching_object=model))
```

[ ]:
```

```