EHR Working Group Workshop

Practical: Propensity scores

10th July 2024

In this practical, we will explore the relationship between treatment *(exposure)* and outcome using propensity score approaches for the simulated cohort data.

```
# Import necessary libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score
```

Exercise 1 Read in the analysis dataset (cohort.dta) and explore the data.

For today's session, since we wish to focus our attention on the estimation of causal effects we will remove patients with missing data. In real life, for variables that contain missing values we would instead use a range of techniques to handle the missing data.

```
data = pd.read_stata('cohort.dta')

# Remove patients with missing data
data = data.dropna()
print(data.describe())
```

Exercise 2 Estimate the effect of treatment on the outcome

```
model_1 = sm.Logit(data['outcome'], sm.add_constant(data['trt'])).fit()
print(model_1.summary())

# Repeat, adjusting for covariates
covariates = ['age', 'female', 'ses', 'smoke', 'alc', 'bmicat', 'nsaid_rx', 'cancer', 'hyper']
```

```
model_2 = sm.Logit(data['outcome'], sm.add_constant(data[['trt'] + covariates])).fit()

print(model_2.summary())
```

Exercise 3 Estimate the propensity score

As described in the earlier session, the propensity score is defined as the probability of exposure/treatment assignment conditional on measured (or observed) characteristics. Our aim is to identify an exposed group and a non-exposed group that are similar in terms of observed characteristics, on average. To do that, for each individual, we estimate their probability of being exposed (i.e. of initiating the study drug instead of the comparator drug). Here, we use a multiple logistic regression model to estimate those probabilities.

First, we will fit a multiple logistic regression model for being exposed to the study drug:

Fit model

```
model_ps = sm.Logit(data['trt'], sm.add_constant(data[covariates])).fit()
```

Using this model, we predict the probability of each individual being exposed given their observed confounders/characteristics. We will save these probabilities as a variable in the analysis dataset.

```
data['ps'] = model_ps.predict(sm.add_constant(data[covariates]))

print(model_ps.summary())
```

Exercise 4 Check distribution

After fitting the propensity score, it is important to investigate its distribution!

Ideally, we do not want the propensity score to completely explain the probability of being in the exposed or the unexposed group (i.e. we do not want scores at zero or 1). If this does occur, this would indicate that some individuals have almost no chance of being in the other exposure group, which would violate the positivity assumption (which states that each individual has a non-zero probability of being in either exposure group).

Draw a histogram of the estimated propensity scores. Are there very extreme propensity score values?

```
sns.histplot(data['ps'], kde=True)

plt.title('Histogram of Propensity Scores')

plt.show()


sns.kdeplot(data=data, x='ps', hue='trt')

plt.title('Kernel Density of Propensity Scores by Treatment')

plt.show()
```

Exercise 5 Check distribution cont.

Second, we wish to check that the range of propensity score values in the two exposed groups overlaps. Such overlap is sometimes called *common support*. A lack of common support (i.e. lack of overlap) would also indicate a violation of the positivity assumption. For example, if we have estimated propensity score values between 0.8 and 0.9 in the exposed group but the highest propensity score in the unexposed group is 0.78, this would indicate that we have a set of people in the exposed group who do not have anyone comparable to them in the unexposed group.

To explore the area of common support, you can use a histogram or plot a box plot stratifying by treatment initiated:

```
sns.histplot(data=data, x='ps', hue='trt', element='step', stat='density', common_norm=False)

plt.title('Histogram of Propensity Scores by Treatment')

plt.show()
```

```
# Summarise the propensity score

print(data.groupby('trt')['ps'].describe())
```

Based on the results above, how concerned are you about violations of the positivity assumption?

The study drug group has estimated propensity score values higher than the comparator group (0.6545169 vs 0. 6532809). However, these are so close that we wouldn't consider this concerning in practice. Similarly, the difference between the minimum values is very close.

We are not particularly concerned about common support here. However, it is worth bearing in mind that violations of positivity can occur that are not obvious in the plots and statistics we have looked at.

Exercise 6 Assessing covariate balance

Assessing differences in the distribution of covariates between exposed and non-exposed groups before incorporating any type of adjustment for confounding is a key step to identify potential confounding (by measured covariates). Here we will calculate, for each covariate, the standardized difference (StD) between exposed and non-exposed groups. We use StD instead of hypothesis testing because the StD is less influenced by sample size. We will consider values of StD of over 0.1 to indicate meaningful imbalance of that covariate between groups.

To calculate the standardized differences between two groups for continuous variables we use the means and standard deviations of the mean:

$$StD = \frac{(mean_{exposed} - mean_{unexposed})}{sd_{pool}}$$

where $mean_{exposed}$ and $mean_{unexposed}$ are the sample mean of the covariate in the exposed and unexposed groups, respectively, and "sd_{pool}" represents the standard deviation of the covariate in the entire sample. There are various ways of obtaining an estimate of the latter. We will calculate this from the sample standard deviations of the covariate in the two groups ($sd_{exposed}$and $sd_{unexposed}$) as:

$$sd_{pool} = \sqrt{\frac{({sd_{exposed}}^2 + {sd_{unexposed}}^2)}{2}}$$

Use this formula to calculate the standardized difference of age between the study drug and the comparator groups.

```
age_mean = data.groupby('trt')['age'].mean()

age_std = data.groupby('trt')['age'].std()

print("Mean Age by Treatment:")

print(age_mean)

print("\nStandard Deviation of Age by Treatment:")

print(age_std)
```

To calculate the standardized difference between two groups for categorical variables we use proportions and standard deviations of proportions:

$$StD = \frac{(prop_{exposed} - prop_{unexposed})}{sd_{pool}}$$

where $prop_{exposed}$ and $prop_{unexposed}$ are the sample proportion with the characteristic of interest in the exposed and unexposed groups, respectively, and sd_{pool} can be calculated as:

$$sd_{pool} = \sqrt{\frac{((prop_{exposed} \times (1 - prop_{exposed})) + (prop_{unexposed} \times (1 - prop_{unexposed})))}{2}}$$

Use these formulae to calculate the standardized difference of gender between the study drug and comparator groups.

```
female_prop = data.groupby('trt')['female'].value_counts(normalize=True).unstack().fillna(0)

print(female_prop)
```

Alternatively, you can use a loop to do this:

```
std_diffs = {}
```

```
for covariate in covariates:
        group1 = data[data['trt'] == 1][covariate]
        group2 = data[data['trt'] == 0][covariate]
        std_diffs[covariate] = standardized_difference(group1, group2)
print(std_diffs)
```

What are the differences between the different ways of obtaining the standardized differences?

How balanced are the covariates?

Exercise 7 Adjust on propensity score

```
interaction = data['trt'] * data['ps']
model_adjusted = sm.Logit(data['outcome'], sm.add_constant(data[['trt', 'ps',
interaction]])).fit()
print(model_adjusted.summary())
```

Exercise 8 Generate inverse probability treatment weights

Inverse probability weighting can be used to estimate the average treatment effect (ATE) using the formula below.

$$ATE = E(Y|X_{exposed}) - E(Y|X_{unexposed})$$

Generate the weights for the treated and untreated group

$$weight_{exposed} = \frac{1}{PS}$$

$$weight_{unexposed} = \frac{1}{(1 - PS)}$$

Initialize the 'wt' column with NaN

```
data['wt'] = np.nan
```

Set 'wt' based on 'trt' values

```
data.loc[data['trt'] == 1, 'wt'] = 1 / data['ps']
data.loc[data['trt'] == 0, 'wt'] = 1 / (1 - data['ps'])
```

Check balance after weighting (you may need a balance check function here)

Exercise 9 Estimate the average treatment effect using inverse-probability weights

```
model_ate = sm.Logit(data['outcome'], sm.add_constant(data['trt']),
freq_weights=data['weight']).fit()

print(model_ate.summary())
```

Exercise 10 Perform matching

Matching is one of the most common propensity score methods used in health research. It involves comparing each individual exposed to the study drug to a similar individual exposed to the comparator drug (i.e. they have similar propensity scores).

By selecting two groups with similar distributions of observed confounders, we attempt to replicate the situation of a trial. Matching involves matching exposed and unexposed individuals on their propensity score values. In most applications, we aim to achieve balance in the baseline covariates between the exposed and unexposed groups (i.e. matching on the propensity score should remove differences in observed confounders).

```
# Here, we use the pymatch library to perform 1:1 nearest neighbour matching without
replacement.

# Install the psmatch2 package

from pymatch.Matcher import Matcher

m = Matcher(data, data, yvar="trt", exclude=["outcome", "ps"])

m.fit_scores(balance=True, nmodels=10)

m.match(method="min", nmatches=1, threshold=0.01)

m.assign_weight_vector()


# Replace with caliper width of 0.2*SD(PS)


# Create matched dataset

matched_data = m.matched_data


# Now calculate the standardized differences to assess whether matching on the propensity
score has created balance on all measured confounders.

matched_std_diffs = {}

for covariate in covariates:

        group1 = matched_data[matched_data['trt'] == 1][covariate]

        group2 = matched_data[matched_data['trt'] == 0][covariate]
```

```
        matched_std_diffs[covariate] = standardized_difference(group1, group2)
print(matched_std_diffs)
```

In a real-life scenario, it is unlikely data would be balanced prior to matching. If imbalances still remain after matching, it may be that the functional form of continuous variables is not correctly specified or there may be some interactions between variables that need to be accounted for in the model.

Finally, let us re-examine the relationship between treatment and the outcome in our matched dataset.

```
model_matched = sm.Logit(matched_data['outcome'], sm.add_constant(matched_data['trt']), freq_weights=matched_data['_weight']).fit()
print(model_matched.summary())
```

Exercise 11 Complete the table using the estimates generate from earlier questions. How do the methods compare?

Method	**N**	**Estimate**	**SE**
Adjustment			
Weighting			
Matching			