PRACTICAL EXERCISE IN R

Description of dataset for practical exercises

The dataset for the practical exercise is derived from the 'Colorado study', which evaluates cost effectiveness of different reimbursement mechanisms for Medicaid cases with severe mental illness.[1] The study compares fee for service (FFS) with two different capitation methods. In one capitation model, services are provided directly (direct capitation or DC) by not-for-profit community mental health centres. In these practicals, we focus on this form of capitation and compare it to FFS. In FFS physicians are reimbursed according to how much of a service they provide and in capitation method physicians are reimbursed according to the number of patients registered.

The practical exercises are based on a subset of cases whose medical bill is reimbursed by either FFS (n=114) or DC (n=155). Note that the sample and (hence the results) have been deliberately changed from the published paper.

For each method the decision problem you should consider is whether DC is cost-effective compared to FFS for Medicaid patients with severe mental health illness.

The dataset is provided in the zip folder emailed to your previously.

We also provide an e-copy of these notes in this subfolder that you can copy and paste commands from.

Please note that while these data have been anonymised, permission for their use is only for educational purposes. Under no circumstances should they be used for research or other purposes.

Variable description for dataset

Variable name	**Description**
Id	Study id
Key Baseline measures	
W1paid	Cost prior to intervention, continuous
W1qaly	QALY prior to intervention, continuous
W1schiz	Schizophrenia, 1=yes; 0=no
W1bipolar	Bipolar, 1=yes; 0=no
W1age	Age (continuous)
W1male	0=women; 1=men
W1highcost	Previously high cost (0=no; 1=yes)
W1lowcost	Previously low cost (0=no; 1=yes)

[1]Grieve et al. 2008. "Evaluation of health care programs by combining cost with quality of life measures: a case study comparing capitation and fee for services." Health Services Research 43(4):1204-22.

W1use	Previously used any service (0=no; 1=yes)
W1nonwhite	Ethnicity, 1= white, 0=other
W1phs	Physical health at baseline, continuous
W1mhs	Mental health at baseline, continuous
W1gf	Global functioning at baseline, continuous
age2	Age squared
age3	Age cubed
priorcost2	W1paid squared
priorcost3	W1paid cubed
priorqaly2	W1qaly squared
priorqaly3	W1qaly cubed
Key outcomes	
totalcost	Total discounted cost in first and second follow-up period
totalqaly	Total discounted QALY in first and second follow-up period
Treatment variable	
treated	Treatment indicator =0 for FFS, 1 for DC

Part 1: Propensity score matching with R

In this exercise, we will perform propensity score matching in `R` using the `Matching` package.

Aim of the exercise

- Learn how to estimate a propensity score in R
- Consider how to assess the specification of the propensity score by checking covariate balance, using a broad range of balance statistics

Before opening R, copy the data file `notforprofit2013.dta` and save it in your own drive (e.g., desktop or external drive). Also save the R script from the same folder, named `Practical.R`. Using this script will make it easier for you to type some longer commands.

Now open R and change the working directory to your own drive (e.g., home directory or external drive) using for example the following command:

```
setwd("H:/Myname/Myfolder/")
```

(tip: in R, the file path contains a forward slash / not a backward slash like for example in Stata!).

Alternatively, you can manually set your working directory by going into the main R menu and choosing File > Change dir.

You can open the script for the R code by going to: File-> Open script and choosing the working directory you set previously. You can modify this script file, or can also create your own script file by choosing File > New script.

First, you need to ensure that the packages, "rgenoud" and "Matching" are installed before loading them. This can be done using the drop-down menu Packages > Install packages in R. To do this, you need to be connected to the internet, select a local "mirror", and look for the packages "rgenoud " and "Matching". It would be good to do this in advance of the short course (tip: in R, the choice of lower versus upper case DOES matter!).

Step 1: Read the data in R

Read the data (in Stata format) `notforprofit2013.dta` into the data frame,

```
dta3 <- read.dta(file = "notforprofit2013.dta")

dta3 <- as.data.frame(dta3)
```

and load the necessary libraries:

```
library(foreign)

library(boot)

library(rgenoud)

library(Matching)
```

Before starting the propensity score matching, you can familiarise yourself with the requisite matching function, using the help page in R. Type the following:

```
?Match()
```

Step 2: Estimate the propensity score

Estimate the propensity score using a logistic regression. For now, we will add potential confounders as linear terms. If you have time later, you can play around with making the model more flexible, for example by adding nonlinear terms.

```
pscore <- glm(treated ~ W1male + W1nonwhite + W1highcost + W1lowcost
+ W1schiz + W1bipolar+ W1age + W1qaly + W1paid + W1phs + W1mhs +
W1gf + W1use, data = dta3, binomial(link = "logit"))
```

You can inspect the propensity score model by typing

```
summary(pscore)
```

Save the linear predictor of treatment assignment using the code

```
pscore_est <- predict(pscore, data = dta3)
```

Append the predicted linear predictor of the propensity score (`pscore_est`) to the existing variables in the data set (`dta3`):

```
dta3<- cbind(dta3, pscore_est)
```

and remove the object `pscore_est`, so that we don't have duplicated objects (one inside, and one outside of the dataframe `dta3`).

```
rm(pscore_est)
```

Step 3: Perform propensity score matching

Before conducting 1 to 1 nearest-neighbour propensity score matching with replacement (default option), remember to attach your data frame (`attach(dta3)`). This enables us to refer to a variable in the dataset by simply giving their name, e.g. "treated". (Alternatively, you can refer to the object as `dta3$treated`, and you don't need to attach).

The syntax for propensity score matching is:

```
attach(dta3)

m_ps1 <- Match(Tr = treated, X = pscore_est, estimand="ATT")

detach(dta3)
```

Remember to detach the dataset using `detach(dta3)`.

Step 4: Assess balance

To assess the balance on covariates before and after matching both formal statistical tests and graphical tools can be used.

The function `MatchBalance` allows us to check if the results of `Match` have achieved balance on covariates by calculating four different sets of balance statistics. We would like to assess the balance on the following confounders: `W1age, W1qaly, W1paid, W1nonwhite, W1schiz` and `W1bipolar.`

The syntax for `MatchBalance` is:

```
set.seed(1122)
```

```
mb_ps1 <-MatchBalance(treated ~ W1age + W1qaly + W1paid + W1nonwhite
+ W1schiz + W1bipolar, match.out = m_ps1, data = dta3, nboots = 500)
```

Note that we set the seed prior to `MatchBalance` in order to obtain the exact same results every time `MatchBalance` is used. `nboots` is the number of bootstrap samples to be run. Bootstrapping is highly recommended because the bootstrapped Kolmogorov-Smirnov (KS) test provides correct coverage even when the distributions being compared are not continuous. At least 500 `nboots` are recommended for publication quality p-values. If you set `nboots` to zero then bootstrap resampling is not undertaken.

Which covariates are imbalanced before matching? Which after matching? Compare the p-value for the t-test with that for the KS test for `W1qaly`. What do you notice?

The QQ plot is a graphical aid that can be used to compare the distribution of a potential confounder variable between the control and the treatment group.

Empirical-QQ (eQQ) plots of `W1qaly` before and after matching can be produced by using the `qqplot` function in the following way:

```
attach(dta3)

qqplot(W1qaly[treated==0], W1qaly[treated==1])

detach(dta3)
```

However to get nicer plots we can use some options as shown below:

```
attach(dta3)

par(mfrow = c(1,2), las=2, oma=c(8,0.5,8,0.5))

qqplot(W1qaly[treated==0], W1qaly[treated==1], main="Before
matching", ylab = "Treatment observations", xlab="Control
observations", ylim=c(min(W1qaly[treated==1],
W1qaly[m_ps1$index.treated]), max(W1qaly[treated==1],
W1qaly[m_ps1$index.treated])),xlim=c(min(W1qaly[treated==0],
W1qaly[m_ps1$index.control]), max(W1qaly[treated==0],
W1qaly[m_ps1$index.control])))

abline(coef=c(0,1))


qqplot(W1qaly[m_ps1$index.control], W1qaly[m_ps1$index.treated],
main="After matching", ylab="Treatment observations", xlab="Control
observations", ylim=c(min(W1qaly[treated==1],
```

```
W1qaly[m_ps1$index.treated]), max(W1qaly[dta3$treated==1],
W1qaly[m_ps1$index.treated])),xlim=c(min(W1qaly[treated==0],
W1qaly[m_ps1$index.control]), max(W1qaly[treated==0],
W1qaly[m_ps1$index.control])))

abline(coef=c(0,1))

detach(dta3)
```

Note that `par(mfrow = c(1,2), las = 2, oma=c(8,0.5,8,0.5))` allows us to have 1x2 pictures on one plot. With `ylim` and `xlim` we make sure that the plots before and after matching, are on the same scale.

Assess the balance for the squared and cubed terms of `W1qaly (priorqaly2` and `priorqaly3`) by including them in `MatchBalance`. What do you conclude?

Step 5: Estimate the ATT

Only once the covariate balance post matching has been judged sufficient, should the estimated ATT be reported.

To estimate the ATT for `totalcost` after matching on the propensity score with non-linear terms, the code is:

```
attach(dta3)

m_ps_cost <- Match(Y=totalcost, Tr=treated, X=pscore_est,
estimand="ATT")

summary(m_ps_cost, full=TRUE)

detach(dta3)
```

Now modify the above code to estimate the ATT for the same matched dataset for `totalqaly`.

Step 6: Save and store the matched dataset

This part of code allows you to save and store the matched data:

```
match.ps.data <- rbind(dta3[m_ps1$index.treated,],
dta3[m_ps1$index.control,])

match.ps.data <- cbind(match.ps.data, weights=c(m_ps1$weights,
m_ps1$weights))
```

```
match.ps.data <- cbind(match.ps.data,
matchid=c(1:length(m_ps1$index.treated),1:length(m_ps1$index.control
)))

write.table(match.ps.data, file = "match.ps.data.csv", sep = ",",
col.names = NA, na="NA", qmethod = "double")
```

Note: The vectors `index.treated` and `index.control` store the original observation number for the treated and control observation, in each matched pair. An easy way to inspect the matched pairs is by binding these vectors together, with the weight attached to each pair.

```
matchedpairs=cbind(m_ps1$index.treated
,m_ps1$index.control,m_ps1$weights)
```

Now type `head(matchedpairs)` to see the first elements of the vector, and `dim(matchedpairs)` to check the dimensions of the matched data.

Notice that this dataset is longer (165 rows) than the number of treated observations in the original dataset (n=155). The reason for this is that for certain treated observations, more than one control observation can be matched, because they have the same Pscore (tie). In this case, the default option in the Matching package is *not* to break the tie, but to create more than one matched pairs. For example, the 1st (treated) observation is matched with the 97th and 144th (control) observations, and both matched pairs are stored with a weight of 0.5.

You can verify that the weights add up to the original number of treated observations, by typing:

```
sum(m_ps1$weights)
```

You can check how many of the original 114 control observations have been used in the matching process, by typing:

```
length(unique(m_ps1$index.control))
```

You may want to save the output of the propensity score matching so that later you can compare it to the Genetic Matching results. An easy way to do this is by simply copying the text output, as well as the generated plots into a Word document.

You can also save the R script file by moving the cursor to the "R editor" window, and selecting File menu > Save as.

Part 2: Genetic Matching with R

In this exercise we will perform multivariate matching in `R` by using the Genetic Matching algorithm, incorporated in the `R` package `Matching`.

Aim of the exercise

- Learn how to define those potential confounders we wish to achieve balance on (`BalanceMatrix`);
- Perform Genetic Matching (`GenMatch`);
- Learn how to perform multivariate matching on potential confounders (`X` matrix) (`Match`);
- Check covariate balance and compare the balance achieved with that from propensity score matching;
- Estimate the treatment effects (point estimates and 95% CI), after Genetic Matching.

By the end of the exercise you should be able to perform Genetic Matching and interpret the results.

Step 1 Load data and libraries, estimate Pscore as before

Make sure that the dataset and the `Matching()`, `foreign()and rgenoud()` and `boot()` libraries are loaded, as described previously. We will use the previously estimated propensity score, `pscore_est` so make sure you re-create it and add it to the dataset. (The script `Practical.R` repeats these steps.)

Before implementing GenMatch, you can familiarise yourself with the function, using the help page. Type the following:

```
?GenMatch()
```

Step 2: Specify the `X` and `BalanceMatrix` matrices

Include in the `X matrix` the estimated linear predictor (`pscore_est`) obtained in Part 1, and the following covariates: `W1male, W1nonwhite, W1highcost, W1lowcost, W1schiz, W1bipolar, W1use, W1age, W1qaly, W1paid, W1phs, W1mhs, W1gf`.

```
attach(dta3)

X <-cbind(pscore_est, W1male, W1nonwhite, W1highcost, W1lowcost,
W1schiz, W1bipolar, W1use, W1age, W1qaly, W1paid, W1phs, W1mhs,
W1gf)
```

For now, we include the same variables in the balance matrix as in the X matrix.

```
BalanceMatrix <- X
```

As we will later see, this does not have to be the case: the `BalanceMatrix` needs to include terms we think are the most important to achieve good balance on. (Note: do not use "detach" yet, as we carry on using the dataset in the next step.)

Step 3: Perform Genetic Matching and assess covariate balance

We first perform Genetic Matching, to obtain the `Weight.matrix`, which will then be used in the multivariate matching to calculate distance between the matched pairs. This step is the computationally intensive part of Genetic Matching. Call the function `GenMatch` and save the output in the object `gen1`:

```
gen1 <- GenMatch(Tr = treated, X = X, BalanceMatrix = BalanceMatrix,
pop.size = 500, unif.seed=2233, int.seed=2233)
```

Note that we set the seed inside the function `GenMatch` in order to obtain the exact same results every time `GenMatch` is used. While the recommended population size (`pop.size`) is at least 1000, in order to keep computation time short, we set it to 500. With this dataset, the algorithm will take less than two minutes to run.

In order to obtain the matched data, we perform the multivariate matching, using the X matrix and the weights, obtained from the previous step. Again, we save the output of the matching in an object, `m_gm1`. When calling this function, we need to reference the output object of Genetic Matching (`gen1`), previously stored.

```
m_gm1 <- Match(Tr = treated, X = X, Weight.matrix = gen1, estimand =
"ATT")
```

Don't forget to detach the data:

```
detach(dta3)
```

Note that the syntax for matching is the same as that used in the propensity score matching, the only new element is the weight matrix (`Weight.matrix`). While the `GenMatch` part calculated the weights (hence used more programming time), the `Match` part created the matched datasets and calculated the treatment effects.

Assess the balance on covariates before and after matching by using the function `MatchBalance`:

```
set.seed(1122)

mb_gm1 <- MatchBalance(treated ~ W1age + W1qaly + W1paid +
W1nonwhite + W1schiz + W1bipolar, match.out=m_gm1, data=dta3,
nboots=500)
```

Compare these results with those obtained with the propensity score matching in Part 1. What do you conclude?

Now use the `qqplot` function to visually assess the balance achieved on `W1qaly`:

```
par(mfrow = c(1,2), oma=c(8,0.5,8,0.5))

attach(dta3)

qqplot(W1qaly[treated==0], W1qaly[treated==1], main="Before
matching", ylab = "Treatment observations", xlab="Control
observations",
ylim=c(min(W1qaly[treated==1],W1qaly[m_gm1$index.treated]),
max(W1qaly[treated==1], W1qaly[m_gm1$index.treated])),
xlim=c(min(W1qaly[treated==0], W1qaly[m_gm1$index.control]),
max(W1qaly[treated==0], W1qaly[m_gm1$index.control])))

abline(coef=c(0,1))


qqplot(W1qaly[m_gm1$index.control],W1qaly[m_gm1$index.treated],
main="After matching", ylab="Treatment observations", xlab="Control
observations",
ylim=c(min(W1qaly[treated==1],W1qaly[m_gm1$index.treated]),
max(W1qaly[treated==1], W1qaly[m_gm1$index.treated])),
xlim=c(min(W1qaly[treated==0],W1qaly[m_gm1$index.control]),
max(W1qaly[treated==0], W1qaly[m_gm1$index.control])))

abline(coef=c(0,1))

detach(dta3)
```

Compare these plots with those obtained with the propensity score matching. What do you conclude?

To assess the balance on the squared and cubed terms of `W1qaly`, add `priorqaly2` and `priorqaly3` in `MatchBalance`. Then compare the KS test and t-test for these non-linear terms with those of `W1qaly`. What do you notice?

Step 4: Estimate the ATT

Now we use the matched data obtained in the last step, and estimate the ATT.

```
attach(dta3)

m_gm1_cost <- Match(Y= totalcost,Tr=treated, X=X, Weight.matrix =
gen1, estimand = "ATT")

summary(m_gm1_cost)

detach(dta3)
```

Modify the above code to estimate the treatment effect for `totalqaly`.

Step 5: Save and store the matched dataset

This part of code allows you to save and store the matched data:

```
match.gn.data <- rbind(dta3[m_gm1$index.treated,],
dta3[m_gm1$index.control,])

match.gn.data <- cbind(match.gn.data, weights=c(m_gm1$weights,
m_gm1$weights))

match.gn.data <- cbind(match.gn.data,
matchid=c(1:length(m_gm1$index.treated),1:length(m_gm1$index.control
)))

write.table(match.gn.data, file = "match.gn.data.csv", sep = ",",
col.names = NA, na="NA", qmethod = "double")
```

You're done! If you have time, think through any further analyses that you think are warranted.