Problem Set 4

Tessie Dong, Derek Li, Andi Liu

Due Jan 26th, 2024

Part 1: Describe the Data (10 p)

1. Fill Table 1's columns 5 and 6 using, respectively, the data in `nswpsid.csv` and in `nswcps.csv`. **Notes**: You want to limit attention to observations with `treat=0`. You filled columns 3 and 4 in PSet 3.

```
# Load data
nswpsid <- read_csv("starter-files/nswpsid.csv")
nswcps <- read_csv("starter-files/nswcps.csv")
nswpsid_treat0 <- nswpsid %>% filter(treat == 0)
nswcps_treat0 <- nswcps %>% filter(treat == 0)
```

```
summary_cps <- summarise_all(nswcps_treat0, list(mean))
summary_psid <- summarise_all(nswpsid_treat0, list(mean))
```

Variable	Definition	NSW		PSID-1	CPS-1
		Treated	Control	Control	Control
[1]	[2]	[3]	[4]	[5]	[6]
age	Age in years	25.82	25.05	34.85	33.22
edu	Education in years	10.35	10.09	12.12	12.03
nodegree	1 if education < 12	0.71	0.83	0.31	0.30
black	1 if Black	0.84	0.83	0.25	0.07
hisp	1 if Hispanic	0.06	0.11	0.03	0.07
married	1 if married	0.19	0.15	0.87	0.71
u74	1 if unemployed in '74	0.71	0.75	0.09	0.12
u75	1 if unemployed in '75	0.60	0.68	0.10	0.11
re74	Real earnings in '74 (in '82 $)	2,096	2,107	19429	14017
re75	Real earnings in '75 (in '82 $)	1,532	1,267	19063	13631
re78	Real earnings in '78 (in '82 $)	6,349	4,555	21,554	14,847
treat	1 if received offer of training	1	0	0	0
Sample Size		185	260	2,490	15,992

Table 1: Sample averages for the NSW data (treated and control groups), PSID-1 data, and CPI-1 data.

2. Briefly comment on the completed Table 1. **Hint**: Are the PSID-1 and CPS-1 samples "good" control groups?

   Answer: I would argue that these samples are not the best control groups - this is mostly because many of the OPV covariates from the PSID and CPS exhibit large differences from the characteristics of the NSW sample. For example, the average age of the NSW sample is 25.82, while the average age of the PSID sample is 34.5, and there are large differences in income across the three samples. This suggests that the populations from which PSID and CPS were drawn are not very similar to the population of the NSW sample - making comparisons between treated individuals in the NSW sample and "untreated" individuals in the PSID and CPS samples less reliable, in our opinion.

3. Why do you think that Dehajia and Wahba constructed their "observational datasets" by pulling together the treated sample from NSW and a sample of individuals drawn from either the PSID or the CPS data? **Hint:** Both PSID and CPS include information on whether an individual enrolled in a training course during the previous 12 months. Thus, Dehajia and Wahba could have exploited exclusively observational variation in whether an individual enrolled in a training program. Why do you think that they chose not to follow this approach?

   Answer: We believe that Dehajia and Wahba chose to pool the NSW and PSID/CPS datasets because they wanted to have a larger sample size to work with. This is because the NSW sample is relatively small, and the PSID/CPS samples are much larger. By pooling the NSW and PSID/CPS samples, Dehajia and Wahba are able to increase the sample size of their dataset. In addition, by analyzing samples drawn from different distributions (i.e. PSID/CPS datasets), they could increase the generalizability of their results to the population.

Part 2: Regression-based Estimation of TEs (90 p)

Objective: You use the `nswpsid.csv` dataset to estimate the treatment effect (TE) of the offer of training via regression-based approaches associated with the following three specifications of the outcome equation:

$$re78_i = \alpha + \rho D_i + u_i,\ i = 1, ..., 2675, \tag{1}$$
$$re78_i = \alpha + \rho D_i + \mathbf{x}_i'\beta + u_i,\ i = 1, ..., 2675, \tag{2}$$
$$re78_i = \rho D_i + g(\mathbf{x}_i) + u_i,\ i = 1, ..., 2675, \tag{3}$$

Subscript i denotes an individual. Also: 1) $re78_i$ represents the data field `re78`; 2) D_i represents the data field `treat`; 3) $\mathbf{x}_i$ represents a $K \times 1$ vector of observed pre-determined variables (OPVs); and, 4) $g(\cdot)$ is an unknown and possibly non-linear function (i.e., a generalization of $\alpha + \beta'\mathbf{x}_i$). Table 2's column [1] references the regression specification. Column [2] gives the name of the approach. Column [3] indicates the regression coefficient of interest. You complete columns [4] and [5] with the estimate of the regression coefficient and its standard error (SE).

Reference Model	**Name of the Estimation Approach**	**Parameter of Interest**	**Estimate**	**SE**
[1]	[2]	[3]	[4]	[5]
exp. (1)	Treatment-Control Comparison (TCC)	ρ		
exp. (2)	Reg-Adj. Treatment-Control Comparison (Adj. TCC)	ρ		
exp. (3)	Double Machine Learning (DML)	ρ		

Table 2: Treatment Effect Estimates Based on Three Regression-Based Approaches Applied to Observational Data.

Background: Heteroschedasticity-Robust Standard Errors. In econometrics, the conditional variance is called the skedastic function. Homoschedasticity obtains when the unobservable in a regression specification has the same conditional variance for all values of the explanatory variable(s). For example, in specification (1) there is only one explanatory variable D_i, and it takes only two values, therefore homoshedasticity obtains if $Var[u_i|D_i = 1] = Var[u_i|D_i = 0]$. If this assumption fails, we say that the model exhibits heteroschedasticity. As a rule, we are better off reporting heteroschedasticity-robust SEs, i.e., SEs computed in a way that allows for heteroschedasticity, because they are valid whether or not homoschedasticity holds.

Background: "Partialling-Out" Interpretation of OLS in a MLRM. Simple linear-in-parameter regression models (SLRM) are of the form

$$y_i = \alpha + \beta x_i + u_i \tag{4}$$

where x_i is a single regression covariate. MLRMs are of the form:

$$y_i = \alpha + \beta_1 x_{1,i} + \cdots + \beta_K x_{K,i} + u_i \text{ with } K > 1. \tag{5}$$

In PSet1 you derived the form of the OLS estimator of the slope coefficient in SLRM (4), namely

$$\hat{\beta} = \frac{\sum_{i=1}^n (x_i - \bar{x}) y_i}{\sum_{i=1}^n (x_i - \bar{x})^2} \underbrace{=}_{\text{also equivalent to}} \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}. \tag{6}$$

Note that if you regress x_i on a constant, the fitted value is $\hat{x}_i = \bar{x}$, thus the regression residuals are $\hat{r}_i \equiv x_i - \hat{x}_i = x_i - \bar{x}$. Similarly, if you regress y_i on a constant, the fitted value is $\hat{y}_i = \bar{y}$, thus the regression residuals are $\hat{v}_i \equiv y_i - \hat{y}_i = y_i - \bar{y}$. Accordingly, we can rewrite $\hat{\beta}$ in expression (6) as:

$$\hat{\beta} = \frac{\sum_{i=1}^n \hat{r}_i y_i}{\sum_{i=1}^n \hat{r}_i^2} \underbrace{=}_{\text{also equivalent to}} \frac{\sum_{i=1}^n \hat{r}_i \hat{v}_i}{\sum_{i=1}^n \hat{r}_i^2}. \tag{7}$$

Similar steps yield a very compact representation of the OLS estimator of the slope coefficient**s** in a MLRM. For example, the OLS estimator of β_1 in MLRM (5) can be written as:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} \hat{r}_{1,i} y_i}{\sum_{i=1}^{n} \hat{r}_{1,i}^2} \underbrace{=}_{\text{also equivalent to}} \frac{\sum_{i=1}^{n} \hat{r}_{1,i} \hat{v}_{1,i}}{\sum_{i=1}^{n} \hat{r}_{1,i}^2}, \tag{8}$$

where $\hat{r}_{1,i}$ denotes the residuals from regressing $x_{1,i}$ on a constant and all remaining regression covariates, i.e., $\{x_{2,i}, \ldots, x_{K,i}\}$ and $\hat{v}_{1,i}$ denotes the residuals from regressing y_i on a constant and all remaining regression covariates, i.e., $\{x_{2,i}, \ldots, x_{K,i}\}$. Similar expressions hold for $\hat{\beta}_2$, $\hat{\beta}_3$, etc.

4. (30 p) These questions pertain to the specification in expression (1) thus you obtain the Treatment-Control Comparison (TCC) Estimator of the treatment effect of the offer of training.

   a. (8 p) Estimate ρ. See table below for estimate of ρ

```r
# Run the regression
m1 <- lm(re78 ~ treat, data = nswpsid)

# Extract coefficients
coefficients <- summary(m1)$coefficients["treat", c("Estimate", "Std. Error")]

# Get additional test statistics
test_stats <- lmtest::coeftest(m1, vcov. = vcov(m1))

# Combine into a data frame
result_df <- data.frame(Estimate = test_stats[,"Estimate"],
                        Std_Error = test_stats[,"Std. Error"],
                        t_value = test_stats[2, "t value"],
                        Pr = formatC(test_stats[2, "Pr(>|t|)"],
                                     format = "e",
                                     digits = 3))

# Create a table using kable
knitr::kable(result_df)
```

	Estimate	Std_Error	t_value	Pr
(Intercept)	21553.92	303.6414	-13.16871	2.033e-38
treat	-15204.78	1154.6143	-13.16871	2.033e-38

   b. (10 p) Compute heteroschedasticity-robust SEs.

```r
# Calculate robust standard errors
robust_se <- lmtest::coeftest(m1, vcov. = sandwich::vcovHC(m1, type = "HC0"))

# Create a data frame for the table
result_df2 <- data.frame(Estimate = robust_se[,"Estimate"],
                         Robust_Std_Error = robust_se[,"Std. Error"],
                         t_value = robust_se[,"t value"],
                         Pr = formatC(robust_se[2, "Pr(>|t|)"],
                                      format = "e",
                                      digits = 3))

# Generate the table using kable
kable(result_df2)
```

	Estimate	Robust_Std_Error	t_value	Pr
(Intercept)	21553.92	311.6684	69.15658	1.506e-108
treat	-15204.78	655.6691	-23.18971	1.506e-108

   c. (2 p) Verify that $\hat{\rho}$ in **4d** equals $(\overline{re78}^{D=1} - \overline{re78}^{D=0})$, i.e., the difference between the average post-training earnings of the treated and of the control individuals. This fact explains the name of the estimator, and is consistent with what you derived in previous Psets.

```r
nswpsid_treat1 = nswpsid %>% filter(treat == 1)

estimate = mean(nswpsid_treat1$re78) - mean(nswpsid_treat0$re78)
```

The result in Q4a is -15204.78 which is equal to the difference between the average post-training earnings of the treated and of the control individuals, which is also equal to -15204.78.

d. (10 p) Intuitively explain why the TCC approach may not deliver a credible estimate of the average effect of the treatment of interest. **Hint**: Use the result in **4c** to think about what this approach uses to proxy for the missing data, i.e., for the control units' mean of the potential outcome w/ treatment, and for the treated units' mean of the potential outcome w/out treatment.

Answer: The TCC approach uses the control group as a proxy for the missing potential outcome of the treatment group if they had not been treated, and vice versa. However, if the control group is not comparable with the treated group (i.e. the samples are drawn from different distributions), the proxy is invalid. For instance, from the summary statistics provided, we can see that there are differences in characteristics such as age, education, and race between the treated and control groups. These differences suggest that the groups may not be comparable and thus the TCC approach may not deliver a credible estimate of the average effect of the treatment of interest.

5. (20 p) These questions pertain to the specification in expression (2) thus you obtain the Regression-Adjusted Treatment-Control Comparison (Adj. TCC) Estimator of the treatment effect of the offer of training.

   a. (10 p) Add to the model estimated in **4** the following OPVs as regression covariates: `age`, `agesq`, `edu`, `nodegree`, `black`, `hisp`, `re74`, and `re75`. Report $\hat{\rho}$ and its heteroschedasticity-robust SE. **Programming Guidance:** Add column `agesq` (`age` squared) to your dataframe using, e.g., `dplyr::mutate( )`.

```r
# add age squared to the dataframe
nswpsid <- nswpsid %>% mutate(agesq = age^2)
m2 <- lm(re78 ~ treat + age + agesq + edu +
         nodegree + black + hisp + re74 + re75,
         data = nswpsid)
coeff <- summary(m2)$coefficients["treat", c("Estimate", "Std. Error")]
test_stats <- lmtest::coeftest(m2, vcov. = vcov(m2))
result_df <- data.frame(Estimate = coeff["Estimate"],
                        Std_Error = coeff["Std. Error"],
                        t_value = test_stats[2, "t value"],
                        Pr = formatC(test_stats[2, "Pr(>|t|)"],
                                     format = "e",
                                     digits = 3))
knitr::kable(result_df, digits = 3, caption = "OLS Regression Results")
```

Table 5: OLS Regression Results

	Estimate	Std_Error	t_value	Pr
Estimate	217.944	866.197	0.252	8.014e-01

   b. (10 p) Intuitively explain why the Adj. TCC approach may be regarded as an improvement over the TCC approach when it comes to credible identification/estimation of average treatment effects. **TODO**

6. (20 p) Consider again the specification in expression (2) estimated in **5**. Here you implement two procedures, as detailed below, to verify the "partialling-out" interpretation of OLS coefficients in MLRM.

   a. (8 p) Procedure A:

      i. (4 p) First Stage: Regress `treat` on a constant and the OPVs listed in **5a**; obtain the residuals. **Programming Guidance:** If you run `s1 <- lm(treat ~ x1 + x2, data = dt)`, retrieve the residuals as `s1$residuals`.

```
nswpsid_const <- nswpsid %>% mutate(constant = 1)
# regress treat on a constant and the OPVs listed in 5a
m2 <- lm(treat ~ constant + age + agesq +
                 edu + nodegree + black +
                 hisp + re74 + re75,
         data = nswpsid_const)
# get residuals
resid_m2 <- m2$residuals
```

      ii. (4 p) Second Stage: Regress `re78` on a constant and the residuals from **6(a)i**.

```
# regress re78 on a constant and the residuals from 6a
m3 <- lm(re78 ~ resid_m2, data = nswpsid)
# get coefficients
coeff <- summary(m3)$coefficients["resid_m2", c("Estimate", "Std. Error")]
test_stats <- lmtest::coeftest(m3, vcov. = vcov(m3))
result_df <- data.frame(Estimate = coeff["Estimate"],
                        Std_Error = coeff["Std. Error"],
                        t_value = test_stats[2, "t value"],
                        Pr = formatC(test_stats[2, "Pr(>|t|)"],
                                     format = "e",
                                     digits = 3))
knitr::kable(result_df, digits = 3, caption = "OLS Regression Results")
```

Table 6: OLS Regression Results

	Estimate	Std_Error	t_value	Pr
Estimate	217.944	1344.279	0.162	8.712e-01

   b. (8 p) Procedure B:

      i. First Stage: Same as **6(a)i**.

      ii. (4 p) First Stage: Regress `re78` on a constant and the OPVs listed in **5a**; obtain the residuals.

```
# regress re78 on a constant and the OPVs listed in 5a
m3 <- lm(re78 ~ constant + age + agesq +
                 edu + nodegree + black +
                 hisp + re74 + re75,
         data = nswpsid_const)
# get residuals
resid_m3 <- m3$residuals
```

      iii. (4 p) Second Stage: Regress the residuals from **6(b)ii** on the residuals from **6(b)i**.

```
# regress residuals from 6b on residuals from 6a
m4 <- lm(resid_m3 ~ resid_m2)
# get coefficients
```

```r
coeff <- summary(m4)$coefficients["resid_m2", c("Estimate", "Std. Error")]
test_stats <- lmtest::coeftest(m4, vcov. = vcov(m4))
result_df <- data.frame(Estimate = coeff["Estimate"],
                        Std_Error = coeff["Std. Error"],
                        t_value = test_stats[2, "t value"],
                        Pr = formatC(test_stats[2, "Pr(>|t|)"],
                                     format = "e",
                                     digits = 3))
knitr::kable(result_df, digits = 3, caption = "OLS Regression Results")
```

Table 7: OLS Regression Results

	Estimate	Std_Error	t_value	Pr
Estimate	217.944	864.9	0.252	8.011e-01

c. (4 p) Verify that the estimates of the slope coefficient from **6(a)ii** and **6(b)iii** are numerically identical to $\hat{\rho}$ obtained in **5a**. Use this fact to give meaning to the expression "partialling-out" interpretation of OLS in a MLRM. **Hint**: Think about what steps **6(a)i** and **6(b)ii** accomplish.
Answer: TODO

7. (20 p) Consider the partially-linear specification in expression (3). Here you estimate ρ via the the Double Machine Learning (DML) estimation procedure of Robinson (1988), as detailed below.

   a. (2 p) Install four R packages: `DoubleML`, `data.table`, `mlr3`, and `mlr3learners`.

```
library("DoubleML")
library("data.table")
library("mlr3")
library("mlr3learners")
```

   b. (2 p) If your data is not already a `data.table` object convert it. **Programming Guidance:** Assuming that your dataframe is called `df`, use `dt <- data.table::as.data.table(df)`. `data.table` is an extension of `data.frame` and allows for fast manipulation of very large data.

```
dt = data.table::as.data.table(nswpsid)
```

   c. (2 p) Collect all the original OPVs in a list named, for example, `pretreat_colnames`. Note: Henceforth when we refer to these OPVs in mathematical expressions we use the notation $\mathbf{x}_i$.

```
pretreat_colnames = colnames(dt)[-c(1,9)]
```

   d. (2 p) Specify data and variables for the causal model by running the script:

```
dml_data_psid <- DoubleML::DoubleMLData$new(dt,
                         y_col = "re78",
                         d_cols = "treat",
                         x_cols = pretreat_colnames)
```

   and look at the following object.

```
dml_data_psid <- DoubleML::DoubleMLData$new(dt,
                                            y_col = "re78",
                                            d_cols = "treat",
                                            x_cols = pretreat_colnames)
```

```
dml_data_psid
```

```
## ================= DoubleMLData Object ==================
##
##
## ------------------ Data summary      ------------------
## Outcome variable: re78
## Treatment variable(s): treat
## Covariates: age, edu, black, hisp, married, re74, re75, u74, u75, nodegree, agesq
## Instrument(s):
## No. Observations: 2675
```

   e. (2 p) Suppress messages from the `mlr3` package by adding
   `lgr::get\_logger("mlr3")\$set\_threshold("warn")`
   to your script.

```
lgr::get_logger("mlr3")$set_threshold("warn")
```

   f. (2 p) Here you mimic the first stage of Procedure B in **6b**. Namely, you specify the model for the two regression functions $l(\mathbf{x}) = E[\texttt{re78}_i|\mathbf{x}_i = \mathbf{x}]$ and $m(\mathbf{x}) = E[\texttt{treat}_i|\mathbf{x}_i = \mathbf{x}]$. In **6b** you used a linear-in-parameter model and a priori decided which OPVs to include and which transformations

to apply to the OPVs to include (e.g., you excluded `u74`, you used both `age` and `agesq`, you left as-is the other included OPVs). Instead here you do not a priori exclude any OPVs, and you use flexible models, which accommodate complex non-linearities. Run the script:

```
# Specify a RF model as the learner model for l(x)=E[re78|X=x]
ml_l_rf <- mlr3::lrn("regr.ranger")

# Specify a RF model as the learner model for m(x)=E[treat|X=x]
ml_m_rf <- mlr3::lrn("classif.ranger")
```

The above script uses a Random Forest (RF) model for both conditional expectations functions.

```
ml_l_rf <- mlr3::lrn("regr.ranger")
ml_m_rf <- mlr3::lrn("classif.ranger")
```

g. (2 p) Here you initialize & parametrize the model object which you later use to perform estimation. Run the script:

```
# Set seeds for cross-fitting
set.seed(3141)

# Set the DML specification
obj_dml_plr <- DoubleML::DoubleMLPLR$new(dml_data_psid,
                                         ml_l = ml_l_rf, ml_m = ml_m_rf,
                                         n_folds = 2,
                                         score = "partialling out",
                                         apply_cross_fitting = TRUE)
```

The above script: (i) utilizes the data object generated in **7d**, namely `dml_data_psid`; (ii) utilizes the models for the first stage regressions picked in **7f**, namely `ml_l_rf` and `ml_m_rf`; (iii) specifies that we want to split the sample into 2 parts (`n_folds = 2`), and (iv) that we want to use the "partialling out'' approach to estimate causal impacts (`score = "partialling out"`), and (v) that we want to apply cross-fitting (`apply_cross_fitting = TRUE`).

```
set.seed(3141)

obj_dml_plr <- DoubleML::DoubleMLPLR$new(dml_data_psid,
                                         ml_l = ml_l_rf, ml_m = ml_m_rf,
                                         n_folds = 2,
                                         score = "partialling out",
                                         apply_cross_fitting = TRUE)
```

h. (2 p) Here you fit the DML model defined in **7g**. Run the script:

```
obj_dml_plr$fit()
obj_dml_plr
```

At a high level the above script implements all of the following operations: (i) fits the two models for the first stage selected in **7f**, (ii) gets residuals, (iii) regresses the residuals for the outcome variables onto the residuals for the treatment indicator to obtain the DML estimate of ρ in expression (3). Note: You specified `n_folds = 2` and requested `apply_cross_fitting = TRUE` in **7g** thus the 2-stage estimation procedure proceed as follows. First the entire data is split into two sub-samples, call them A and B (hence the term "2 folds"). Sample A is used to fit the 1st stage models. These fitted models are used to compute residuals in sample B and these residuals are used to fit the 2nd stage model using only data in sample B. Denote the resulting estimate $\hat{\rho}_{AB}$. Then the samples are swapped (hence the

term "cross fitting' '). That is, sample B is used to fit the 1st stage models. Sample A is used to fit the 2nd stage model. Denote the resulting estimate $\hat{\rho}_{BA}$. The DML estimate is the average of $\hat{\rho}_{AB}$ and $\hat{\rho}_{BA}$.

```
obj_dml_plr$fit()
obj_dml_plr
```

```
## ================= DoubleMLPLR Object ==================
##
##
## ------------------ Data summary      ------------------
## Outcome variable: re78
## Treatment variable(s): treat
## Covariates: age, edu, black, hisp, married, re74, re75, u74, u75, nodegree, agesq
## Instrument(s):
## No. Observations: 2675
##
## ------------------ Score & algorithm ------------------
## Score function: partialling out
## DML algorithm: dml2
##
## ------------------ Machine learner   ------------------
## ml_l: regr.ranger
## ml_m: classif.ranger
##
## ------------------ Resampling        ------------------
## No. folds: 2
## No. repeated sample splits: 1
## Apply cross-fitting: TRUE
##
## ------------------ Fit summary       ------------------
##  Estimates and significance testing of the effect of target variables
##       Estimate. Std. Error t value Pr(>|t|)
## treat    -603.6     1146.0  -0.527    0.598
```

i. (4 p) Take a look at the output, i.e., at the object `obj_dml_plr`. How does the DML estimate of average treatment effect compare to the estimates based on specifications (1) and (2)? **TODO**