R for Econometrics

Bas Machielsen
a.h.machielsen@uu.nl

2023-09-21

Setup

Installing R

- The R Programming Language
- (Google): Install R `<Operating System>`
  - Windows: https://cran.r-project.org/bin/windows/base/
  - MacOS: https://cran.r-project.org/bin/macosx/
  - Linux: https://www.cloudsigma.com/installing-r-on-ubuntu-21-04-a-tutorial/

Installing RStudio

- The RStudio integrated development environment (IDE)
  - All OS: https://posit.co/download/rstudio-desktop/

Econometrics with R

- In RStudio, open an Rmarkdown document: (File > New File > Rmarkdown)
- A convenient way to make assignments interactively
  - Possibility to combine text/interpretation with econometrics
- Different forms of output (you can write a report or presentation)
- Good resource: https://www.econometrics-with-r.org/

Downloading / Loading packages

- Downloading a package (`install.packages()`)
  - Only once per PC (like installing a video game)
- Loading a package (`library()`)
  - Every time you open R (like opening a video game)
  - Loading a package gives your console/Rmarkdown session access to

Importing Stata datasets

- First library we need: `haven`
  - Reads STATA (`.RAW`, `.DAT` file extensions) into R
  - Once: `install.packages('haven')`, then every session: `library(haven)`
- Pay attention to your working directory
  - In RMarkdown: working directory is the same as the directory where the file is located
  - In the console: working directory is next to your R version
  - Also: `getwd()`

Econometrics Models and Tests

Descriptive Statistics

```r
# Load the package
library(haven)
attend <- haven::read_stata('ATTEND.DTA')
```

```r
summary(attend)
```

```
##      attend         termgpa          priGPA           ACT       
##  Min.   : 2.00   Min.   :0.000   Min.   :0.857   Min.   :13.00  
##  1st Qu.:24.00   1st Qu.:2.138   1st Qu.:2.190   1st Qu.:20.00  
##  Median :28.00   Median :2.670   Median :2.560   Median :22.00  
##  Mean   :26.15   Mean   :2.601   Mean   :2.587   Mean   :22.51  
##  3rd Qu.:30.00   3rd Qu.:3.120   3rd Qu.:2.942   3rd Qu.:25.00  
##  Max.   :32.00   Max.   :4.000   Max.   :3.930   Max.   :32.00  
##                                                                 
##      final          atndrte           hwrte            frosh       
##  Min.   :10.00   Min.   :  6.25   Min.   : 12.50   Min.   :0.0000  
##  1st Qu.:22.00   1st Qu.: 75.00   1st Qu.: 87.50   1st Qu.:0.0000  
##  Median :26.00   Median : 87.50   Median :100.00   Median :0.0000  
##  Mean   :25.89   Mean   : 81.71   Mean   : 87.91   Mean   :0.2324  
##  3rd Qu.:29.00   3rd Qu.: 93.75   3rd Qu.:100.00   3rd Qu.:0.0000  
##  Max.   :39.00   Max.   :100.00   Max.   :100.00   Max.   :1.0000  
##                                   NA's   :6                        
##       soph           skipped          stndfnl        
##  Min.   :0.0000   Min.   : 0.000   Min.   :-3.30882  
##  1st Qu.:0.0000   1st Qu.: 2.000   1st Qu.:-0.78782  
##  Median :1.0000   Median : 4.000   Median : 0.05252  
##  Mean   :0.5765   Mean   : 5.853   Mean   : 0.02966  
##  3rd Qu.:1.0000   3rd Qu.: 8.000   3rd Qu.: 0.68277  
##  Max.   :1.0000   Max.   :30.000   Max.   : 2.78361  
## 
```

Descriptive Statistics

- Better using the `modelsummary` package

```
library(modelsummary)
datasummary(formula = attend + priGPA + final ~ mean +
              median + min + max + sd,
            output = 'markdown',
            data=attend)
```

	mean	median	min	max	sd
attend	26.15	28.00	2.00	32.00	5.46
priGPA	2.59	2.56	0.86	3.93	0.54
final	25.89	26.00	10.00	39.00	4.71

Linear Regression

- Basic function: `lm`
- Syntax: `y ~ x1 + x2 + ..`

```
model1 <- lm(attend ~ priGPA + ACT, data = attend)
model2 <- lm(attend ~ priGPA + ACT + priGPA:ACT, data = attend)

# Or simply, use this:
model2 <- lm(attend ~ priGPA*ACT, data = attend)
```

Linear Regression [2]

```r
summary(model1)
```

```
##
## Call:
## lm(formula = attend ~ priGPA + ACT, data = attend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.919  -2.165   0.680   3.083   9.477
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.22413    1.24291   19.49   <2e-16 ***
## priGPA       5.52339    0.34659   15.94   <2e-16 ***
## ACT         -0.54930    0.05408  -10.16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.601 on 677 degrees of freedom
## Multiple R-squared:  0.2906, Adjusted R-squared:  0.2885
## F-statistic: 138.7 on 2 and 677 DF,  p-value: < 2.2e-16
```

Linear Regression [3]

```r
summary(model2)
```

```
##
## Call:
## lm(formula = attend ~ priGPA * ACT, data = attend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.6607  -2.1462   0.7732   3.0258   9.3390
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.67033    5.85535   6.433 2.36e-10 ***
## priGPA       0.29145    2.25339   0.129   0.8971
## ACT         -1.12504    0.25090  -4.484 8.60e-06 ***
## priGPA:ACT   0.22152    0.09428   2.350   0.0191 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.586 on 676 degrees of freedom
## Multiple R-squared:  0.2963, Adjusted R-squared:  0.2932
## F-statistic: 94.89 on 3 and 676 DF,  p-value: < 2.2e-16
```

Joint Hypothesis Tests

- The car library does joint hypothesis tests in the `linearHypothesis` function:

```
library(car)
linearHypothesis(model2, c("ACT=0", "priGPA:ACT=0"), white.adjust="hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## ACT = 0
## priGPA:ACT = 0
##
## Model 1: restricted model
## Model 2: attend ~ priGPA * ACT
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1    678
## 2    676  2 48.866 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Heteroskedasticity & Autocorrelation Test

```r
library(lmtest)

# Breusch-Pagan (Heteroskedasticity)
bptest(model1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model1
## BP = 66.975, df = 2, p-value = 2.862e-15
```

```r
# Breusch-Godfrey (Serial Correlation)
bgtest(model1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1
## LM test = 0.024495, df = 1, p-value = 0.8756
```

- After loading, check `lmtest::`, see what functions pop up for more.

Robust (& Other) Standard Errors

- Best way to do this is to use the `fixest` package

```
library(fixest)

feols(final ~ atndrte + priGPA, vcov = "hetero", data=attend)
```

```
## OLS estimation, Dep. Var.: final
## Observations: 680
## Standard-errors: Heteroskedasticity-robust
##              Estimate Std. Error   t value  Pr(>|t|)
## (Intercept) 17.966111   0.975502 18.417289 < 2.2e-16 ***
## atndrte     -0.005504   0.010916 -0.504258   0.61424
## priGPA       3.237554   0.368095  8.795439 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 4.37913   Adj. R2: 0.13167
```

Panel Data Models

Fixed Effects

- Either the `plm` package (next slide) or the `fixest` package (my favorite)
  - I use the `LaborSupply` dataset (try `?LaborSupply` in the console after loading `plm`)

```
library(fixest)
data(LaborSupply)
#feols(y ~ x1 + x2 | fixed_effect_1 + fixed_effect_2, vcov = "hetero", data = data)
#?feols

# No. of hours works explained by wages, no. kids, age + person fixed effects
feols(lnhr ~ lnwg + kids + age | id, data=LaborSupply)
```

```
## OLS estimation, Dep. Var.: lnhr
## Observations: 5,320
## Fixed-effects: id: 532
## Standard-errors: Clustered (id)
##      Estimate Std. Error  t value Pr(>|t|)
## lnwg 0.165590   0.086449 1.915468  0.05597 .
## kids 0.005915   0.007801 0.758205  0.44866
## age  0.000941   0.001396 0.674306  0.50041
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## RMSE: 0.220772     Adj. R2: 0.335606
##                 Within R2: 0.016611
```

The plm package

- The plm package: https://cran.r-project.org/web/packages/plm/vignettes/A_plmPackage.html
  - Set your data as panel data

```
library(plm)
df <- pdata.frame(LaborSupply, index=c("id", "year"),
                  drop.index=TRUE,
                  row.names=TRUE)

#plm(y ~ x1 + x2, data = panel_dataframe)
model <- plm(formula = lnhr ~ lnwg + kids + age,
             data=df, model = "within")
```

The `plm` package [2]

- Summary:

```
summary(model, vcov = vcovHC)
```

```
## Oneway (individual) effect Within Model
##
## Note: Coefficient variance-covariance matrix supplied: vcovHC
##
## Call:
## plm(formula = lnhr ~ lnwg + kids + age, data = df, model = "within")
##
## Balanced Panel: n = 532, T = 10, N = 5320
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -4.0060844 -0.0614894  0.0015329  0.0789580  1.2864960
##
## Coefficients:
##        Estimate Std. Error t-value Pr(>|t|)
## lnwg 0.16559004 0.08634321  1.9178  0.05519 .
## kids 0.00591482 0.00779155  0.7591  0.44781
## age  0.00094133 0.00139429  0.6751  0.49963
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    263.68
## Residual Sum of Squares: 259.3
## R-Squared:      0.016611
## Adj. R-Squared: -0.093134
## F-statistic: 2.92589 on 3 and 531 DF, p-value: 0.033344
```

Time-series Data

AR Models

- Use the FERTIL3.DTA dataset

```
fertil <- read_stata("FERTIL3.DTA")
```

- ar.ols allows you to estimate AR models:

```
ar.ols(fertil$gfr,
       order.max = 1,
       demean = F,
       intercept = T)
```

```
##
## Call:
## ar.ols(x = fertil$gfr, order.max = 1, demean = F, intercept = T)
##
## Coefficients:
##      1
## 0.9777
##
## Intercept: 1.305 (2.513)
##
## Order selected 1  sigma^2 estimated as  17.69
```

AR Models [2]

- Also possible with `lm`:

```r
lm(gfr ~ lag(gfr, 1), data = fertil)
```

```
##
## Call:
## lm(formula = gfr ~ lag(gfr, 1), data = fertil)
##
## Coefficients:
## (Intercept)  lag(gfr, 1)
##      1.3049       0.9777
```

Creating Time-series Data

- You can use the `D` command from the `collapse` package to construct differences and lags
  - You can use the `Lag` command from the `Hmisc` package to construct lags

```
library(Hmisc); library(collapse); library(tidyverse)

fertil <- fertil |>
  mutate(gfr_lag = Hmisc::Lag(gfr, 1), # Lag one period
         gfr_diff = collapse::D(gfr), # Differ one period
         gfr_diff2 = collapse::D(gfr, diff = 2)) # Differ two periods
```

Stationarity Tests

- Augmented Dickey-Fuller test:

```
library(tseries)
adf.test(na.omit(fertil$gfr))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(fertil$gfr)
## Dickey-Fuller = -1.8829, Lag order = 4, p-value = 0.623
## alternative hypothesis: stationary
```

```
tseries::adf.test(na.omit(fertil$gfr))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(fertil$gfr)
## Dickey-Fuller = -1.8829, Lag order = 4, p-value = 0.623
## alternative hypothesis: stationary
```

```
tseries::adf.test(na.omit(fertil$gfr_diff2))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(fertil$gfr_diff2)
## Dickey-Fuller = -6.9009, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Instrumental Variables

Instrumental variables

- Instrumental variable estimation has a different syntax depending on the package you use
  - I use panel data packages because they work the best
- With `fixest`:

```
#feols(y ~ x1 + x2 | fe1 + fe2 | endog ~ instrument, data = data)
#feols(y ~ x1 + x2 | endog ~ instrument, data = data)
```

- With `plm`:

```
#plm(y ~ x1+x2+x3 | x3+z1+z2, data = data) # x3 is exogenous, x2 and x1 endogenous
```

- But also: check out the `ivreg` package

Reporting

Modelsummary

- The package `modelsummary` allows you to report tables very easily
- Example Code:

```
library(modelsummary)
# Models
model1 <- feols(cyl ~ mpg, data=mtcars)
model2 <- feols(cyl ~ hp+drat, data=mtcars)

# Tests
bp1 <- bptest(cyl ~ mpg, data=mtcars)$p.value |> round(3)
bp2  <- bptest(cyl ~ hp+drat, data=mtcars)$p.value |> round(3)

# Table
modelsummary(list(model1, model2),
             stars=c("*"=0.1, "**"=0.05, "***"=0.01), # Stars logic
             gof_map = c("adj.r.squared", "nobs"), # Statistics to report
             add_rows = data.frame("Breusch-Pagan Stat. $p$-val", bp1, bp2))
```

Output

- Output:

	(1)	(2)
(Intercept)	11.261***	8.617***
	(0.593)	(1.232)
mpg	-0.253***	
	(0.028)	
hp		0.017***
		(0.002)
drat		-1.365***
		(0.289)
:——————————-	———-:	———-:
R2 Adj.	0.717	0.814
Num.Obs.	32	32
Breusch-Pagan Stat. *p*-val	0.282	0.331

Note: ^^ * $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$