Introduction to R: Econometrics

Cristian Huse

University of Oldenburg

Overview

- The objective of this topic is to guide you through an empirical project using standard tools within Econometrics
- This is important since R has a number of packages available, but it is not always clear which package to use
  - Therefore, you will get an overview of libraries used in Econometrics, at least at the time of writing
- The session will also give you a suggestion of workflow when using R/RStudio

Overview (cont'd)

- For pedagogical purposes, the focus of the session will be on applied econometrics, in particular regression analysis
  - I'm assuming that you already know all of the necessary theoretical background and will revisit it using R
  - If that is not the case, it is your responsibility to go back to your textbook or view a tutorial (Florian Heiß from the Uni Düsseldorf has lots of resources, from textbook to youtube channel, all free, see References)
- The session is obviously biased
  - It focuses on Econometrics
  - It does not discuss how to obtain, clean, merge, transform data etc

Project: Fuel Demand

- We will work with part of the fuel data used in Huse (2018), which estimates the demand for gasoline (g) and ethanol (e) in the Swedish market;
- Our aim is to estimate demand equations of the following form:

$$ln(q_{et}) = \alpha_{1e} + \sum_{k=e,g} \gamma_{ek} ln(p_{kt}) + \theta_e Z_{et} + \varepsilon_t^e$$

$$ln(q_{gt}) = \alpha_{1g} + \sum_{k=e,g} \gamma_{gk} ln(p_{kt}) + \theta_g Z_{gt} + \varepsilon_t^g$$

- Here, t denotes months, q_{ft} the sales of fuel $f = e, g$ at period t, p_{ft} the price of fuel f at period t, Z_{ft} are demographics, controls, and fixed-effects, ε_{ft} are error terms

Project: Fuel Demand (2)

- We will first estimate each equation using OLS, and our main parameters of interest are the price elasticities of demand ($\gamma_{ee}, \gamma_{gg}, \gamma_{eg}, \gamma_{ge}$). Then, given the potential **endogeneity** of prices, we will employ **instruments** ($w^e_{k_e t}, w^g_{k_g t}$) and estimate each equation using 2SLS (the paper still estimates them jointly using 3SLS)

- As in much of demand estimation, prices and quantities are taken to be jointly determined, therefore the issue of endogeneity and the need of an instrument

- Concretely, potential instruments for the (Swedish) domestic gasoline and ethanol prices would be different international commodity prices – e.g., different oil prices for the former and sugar and food prices for the latter
  - Why? Because they are correlated with, but not determined by, domestic fuel prices (i.e, exogenous)

Lab Session: Empirical Project

- Our aim in the session is to replicate Huse (2018)
  - Start with descriptive statistics (Supplementary Table 1)
  - Continue with the six plots (Figure 1)
  - Create new variables, in particular, instruments
  - Estimate the demand for fuels using OLS (parts of Supplementary Table 3)
  - Estimate the demand for fuels using 2SLS (parts of Supplementary Table 4)
  - We won't be using 3SLS at this point
- **Note:** You don't have the same data which was used in the paper (distribution network data is not available) and are using a different software (R instead of Stata), so some differences may occur

Structure of the Session

- Setup
  - Check whether packages are installed, load packages
- Load data
  - Exact commands will depend on file format (.csv, .xls etc)
  - It is possible to convert/load data in other formats (e.g., from Stata)
  - While not our focus, there is often substantial work in cleaning, merging, and transforming data
- Exploratory data analysis, descriptive statistics
  - Format of each variable, summary statistics, plots of key relations of interest
- Econometric analysis
  - Regression, e.g., OLS, IV
  - Not discussed here: time series, panel data, discrete choice
- Reporting results
  - Regression tables
  - Plots of key effects from estimates, e.g., marginal effects
- Saving, exporting any files, if needed

Comments: Starting your Code

- A personal tip here: you might want to start each piece of code with a header providing details about author, name and purpose of code, first version, latest version of it:

```
## IntroR_Metrics.R
## by Cristian Huse -- cristian.huse@uol.de
## Current version: 20210930
## First version:   20210101
## This code provides an introduction to R focusing
## on applied econometrics
```

- Note
  - If you are naming files with a date chunk, the format YYYYMMDD is the most convenient, as it allows ordering more easily (alternatively, version control);
  - The "#" makes the line "invisible" to the compiler;

Comments: Loading Data

- The final aim is to have a **data.frame**, ideally with no **NA**s
- Several alternatives are available, depending on, e.g., file format
  - Base R: read.csv()
  - library(data.table): fread()
  - library(readxl): read_excel()
  - library(xlsx): read.xlsx()
  - library(haven): read_dta() for Stata .dta files
- Key thing is to check the help for the precise options
  - header, separator, range etc

Comments: Summary Statistics

- Again, there are many alternatives out there
- A personal favourite is library(modelsummary)
- Other alternatives exist, e.g.,
  - table()
  - summary()

Comments: Plots

- I opted to use Base R for plots this time. The common structure across most plots is:

plot(): starts plot, sets basic parameters, e.g., title, labels, limits
lines(): is added to the above plot (can have many, alt. **points()**)
legend(): adds a legend to the plot, including colour, label etc

- Instead of using two vertical axes, I opted to scale the data in some plots for the sake of simplicity

- Note that typically a plot needs to have the "x-data" ordered
  - (compare the last plot with the previous ones and/or drop the order command to see what happens)

- You can also combine plots in a matrix, save plots in different file formats etc

Comments: OLS

- The basic command for regression models is **lm()**
  - The optional argument data exists because several data.frame's can co-exist in R

```
lm(y ~ x1 + x2 + x3 + ..., data = df)
```

- The traditional way to extract regression output is via **summary()**, but nicer versions are also available

```
ols1 <- lm(y ~ x, data=df)
summary(ols1)

library(broom)
tidy(ols1, conf.int = TRUE)
glance(ols1)
```

Comments: OLS with HC Standard Errors

- One way to get (heteroskedasticity-consistent = HC) robust standard errors is to use **library(estimatr)**

```
library(estimatr)
ols1_robust <- lm_robust(y ~ x, data = df)
ols1_robust

#compare with
tidy(ols1_robust, conf.int = TRUE)
```

- Note that these are not the same as those computed by Stata if that is relevant (discussion):

```
lm_robust(y ~ x, data = df, se_type = "stata")
```

Comments: OLS with HAC Standard Errors

- One way to get (heteroskedasticity and autocorrelation consistent = HAC) robust standard errors is to use **library(sandwich)**

```
library(sandwich)
library(lmtest)
ols1_hac <- lmtest::coeftest(ols1, vcov = NeweyWest)
ols1_hac
tidy(ols1_hac, conf.int = TRUE)
```

Comments: OLS with Clustered Standard Errors

- Clustered standard errors typically pertains to panel data, so here follows a quick example using **estimatr::lm_robust()**

```
lm_robust(y ~ x, data = df, clusters = my_cluster_var)
```

Comments: OLS with Fixed-Effects

- We saw it before, but for the sake of completeness...
- Month and year fixed-effects

```
summary(lm(y ~ x + as.factor(month) + as.factor(year),
           data = df))
```

- Interaction effects
  - $x1 : x2$: interacts the variables, i.e., $x1 * x2$
  - $x1/x2$: "nests" the second variable within the first, i.e., $x1 + x1 : x2$
  - $x1 * x2$: includes all original and interaction terms, i.e., $x1 + x2 + x1 : x2$

```
ols_ie <- lm(y ~ x1 * x2, data = df)
summary(ols_ie)
```

Comments: Panel Data

- As often happens, there are several alternatives
  - **library(lfe)**: quite similar to the Stata library **reghdfe**
  - **library(fixest)**: our favourite, new and very efficient alternative, see here
- Using fixest, the fixed-effects appear after a | and the default is to cluster the standard errors by the fixed effect variable (note how the output will mention se's details)

```
library(fixest)
ols_fe <- feols(y ~ x | fe1, data = df)
# Fixed effect(s) go after the "|"
ols_fe

# If standard se's
feols(y ~ x | fe1, data = df, se = 'standard')
```

Comments: Panel Data (2)

- **fixest** also handles high-dimensional FEs. . .

```
library(fixest)
ols_hdfe <- feols(y ~ x | fe1 + fe2, data = df)
ols_hdfe
```

- . . . and multiway clustering – the following are equivalent

```
ols_hdfe1 <- feols(y ~ x | fe1 + fe2,
                   se = 'twoway', data = df)
ols_hdfe2 <- feols(y ~ x | fe1 + fe2,
                   cluster = c('fe1', 'fe2'), data = df)
ols_hdfe3 <- feols(y ~ x | fe1 + fe2,
                   cluster = ~ fe1 + fe2, data = df)
```

Comments: Panel Data (3)

- A note on reporting results: **fixest** also has tools to compare models, report results etc

```
etable(ols_fe, ols_hdfe)

coefplot(list(ols_fe, ols_hdfe))
# Optional
legend("bottomleft", col = 1:2, lwd = 1, pch = c(20, 17),
       legend = c("FE and no clustering", "HDFE and
                  twoway clustering"))
```

Comments: IV

- The following packages mostly conduct estimation using **2SLS = Two-stage Least Squares**. We will focus on three packages.

- **Option 1: ivreg::ivreg()**
- One syntax is $y \sim ex|en|in$, where the right-hand variables are endogenous, exogenous, and instruments

```
library(ivreg)

# 3-part formula
iv <- ivreg(y ~    ## LHS: Dependent variable
              x |  ## 1st part RHS: Exog. variable(s)
              w |  ## 2nd part RHS: Endog. variable(s)
              z,   ## 3rd part RHS: Instruments
           data = df
           )
summary(iv)
```

Comments: IV (2)

- It is also possible to use the syntax $y \sim ex + en | ex + in$

```
library(ivreg)

# 2-part formula
iv <-ivreg(y ~    ## LHS: Dependent variable
             x + w | ## 1st part RHS: ex + en
             x + z | ## 2nd part RHS: ex + in
           data = df
           )
summary(iv)
```

- See an example here for details

Comments: IV (3)

- **Option 2: estimatr::iv_robust()**
- It follows the syntax $y \sim ex + en | ex + in$

```
library(estimatr)

# 2-part formula with robust SEs
iv_reg_robust <-
  iv_robust(y ~
              x + w |
              x + z,
            data = df
            )
summary(iv_reg_robust, diagnostics = TRUE)
```

Comments: IV (4)

- **Option 3: fixest::feols()**
- This combines tools of IV and panel data, which is often the case in Applied Microeconomics (e.g., Environmental Economics, Industrial Organization)
- It follows the syntax $y \sim ex|fe|en \sim in$
- Example with no FEs:

```
library(fixest)

iv_feols <-
  feols(y ~ x |  #y ~ ex
          w ~ z, #en ~ in (1st-stage, final slot)
        data = df
        )
summary(iv_feols, stage = 1) #Shows 1st stage in detail
iv_feols
```

Comments: IV (5)

- Example with FEs:

```
library(fixest)

iv_feols <-
  feols(y ~ x |      #y ~ ex
        mo + yr | #Include FEs slot
        w ~ z,    #en ~ in (1st stage, final slot)
              data = df
              )
summary(iv_feols, stage = 1)#Shows 1st stage in detail
iv_feols
```

Comments IV (6)

- The above packages deal with methods where estimation is done equation-wise
- There are cases, however, where system estimation is important, e.g., the errors of the equations are potentially correlated, on top of endogeneity
- This calls for the use of **3SLS = Three-stage Least Squares** which is performed using the package **systemfit** (note it can also do SUR and 2SLS estimation)
- Resources (check the vignette)

Comments IV (7)

```r
install.packages("systemfit")
eq1 <- y1 ~ y2 + x1
eq2 <- y2 ~ y1 + x2
eqSystem <- list(demand = eq1, supply = eq2)
fitols <- systemfit(eqSystem) #OLS
fitsur <- systemfit(eqSystem, method = "SUR") #SUR
# 3SLS
# same instruments across equations
fit3sls1 <- systemfit(eqSystem, method = "3SLS",
                      inst = ~ z1 + z2 + z3)
# different instruments across equations
fit3sls2 <- systemfit(eqSystem, method = "3SLS",
                      inst = list(~ z2 + z3,
                                  ~ z1 + z2 + z3))
print(fit3sls1)
print(fit3sls2)
```

Other Commands (for completeness)

- Logit models

```r
# glm()
glm_logit = glm(am ~ cyl + hp + wt, data = mtcars,
                family = binomial)
summary(glm_logit)

# **fixest**
feglm(am ~ cyl + hp + wt, data = mtcars,
      family = binomial)

# marginal effects
library(mfx)
logitmfx(glm_logit, atmean = TRUE, data = mtcars)
```

Other Commands (a selection of libraries)

- A selection of libraries based on methods:
  - Discrete-choice methods: mlogit
  - Difference-in-differences (with variable timing, etc.): did and DRDID
  - Synthetic control: tidysynth, gsynth and scul
  - Count data: pscl
  - Lasso: biglasso
  - Causal forests: grf

Comments: Regression Tables

- As before, there are several alternatives out there. A great one is **modelsummary**, see here for great examples

- A rough table with output of three regressions, stars to denote significance, 3 decimal places

```
library(modelsummary)
modelsummary(list(reg1, reg2, reg3),
             stars = c("*"=.1, "**"=.05, "***"=.01),
             fmt = 3)

# Save as latex file, drop intercept and most gof stats
modelsummary(list(reg1, reg2, reg3),
             stars = c("*"=.1, "**"=.05, "***"=.01),
             fmt = 3,
             coef_omit = "Intercept",
             gof_omit = "AIC|BIC|Log.Lik.|R2 Adj.|R2 Withi
             output = "my_first_table.tex")
```

Comments: Regression Tables (2)

```
# Omit the intercept, report only *.*lpg, *.*lpe terms
modelsummary(list(reg1, reg2, reg3),
             stars = c("*"=.1, "**"=.05, "***"=.01),
             fmt = 3,
             coef_omit = ("^(?!.*lpg)(?!.*lpe)(\\(Int)"),
             gof_omit = "AIC|BIC|Log.Lik.|R2 Adj.|R2 Withi
             )
```

```
# Label coefficients and table columns (specifications)
cm <- c('lpg'    = 'log(Gasoline price)',
        'lpe'    = 'log(Ethanol price)')
models <- list("OLS1"=reg1, "OLS2"=reg2, "OLS3"=reg3)
modelsummary(models, coef_map = cm,
             stars = c("*"=.1, "**"=.05, "***"=.01),
             fmt = 3,
             gof_omit = "AIC|BIC|Log.Lik.|R2 Adj.|R2 Withi
             )
```

Instructions for the Lab Session Econometrics with R

- Please load the file **IntroR_Metrics.R**
- Go through the file line-by-line, consulting the help whenever needed – this is your exercise for next week (book 2-3 hours)

Take-aways

- The aim of this slide deck (and associated lab session) is to provide you with an overview of how one can apply econometric techniques using R/RStudio;
- With this overview, you should be able to execute a standard empirical project using tools from Econometrics which don't involve much data cleaning or processing;
- You are not expected to **memorize** everything that was covered in the slides, but you should invest time and focus on the associated .R files – make sure you **understand** the commands used and/or the underlying logic;

Take-aways (2)

Repeating myself:

- From my own experience, the best way to learn is to get your hands dirty with data:
  - Go through the files in detail
  - Take something you know and have done before and re-do the project using a new language
  - There are countless channels, tutorials, books, and communities, e.g., Stack Overflow
- As in everything, the contents and the approach pursued here are biased, incomplete, and reflect (my) personal taste

Selected References

- An Introduction to R
- R Data Import/Export
- Wickham & Grolemund's R for Data Science
- Introduction to Econometrics with R or here
- Florian Heiß's
  - Using R for Introductory Econometrics
  - YouTube channel

Appendix

Setup: Package Management

- Installing packages one-by-one can become tedious, but one can use **pacman**, a package management tool to automatize the process:

```
## Load and install today's packages
if (!require("pacman")) install.packages("pacman")
pacman::p_load(mfx, tidyverse, hrbrthemes, estimatr,
               ivreg, fixest, sandwich, lmtest,
               margins, vtable, broom, modelsummary,
               data.table,fastverse)
## Make sure we have at least version 0.6.0 of ivreg
if (numeric_version(packageVersion("ivreg")) <
    numeric_version("0.6.0"))
  install.packages("ivreg")
## Optional -- ggplot2 plotting theme
theme_set(hrbrthemes::theme_ipsum())
```