Introduction to R - Part II

Adrian Rohit Dass

2025-01-17

Recap: Outline from Part 1

- Why use R?
- R Basics
- R for Data Manipulation
  - Reading-in data, sub-setting, creating new variables, etc.
- R for Statistical Analysis
- Descriptive and Regression Analysis
- Applied Example
- Other topics in R
  - Tidyverse
  - data.table
  - R Studio
  - R Markdown
- Applied Example 2
- R Resources

Recap: Survey of Health Economics Students

A survey conducted to previous IHPME health economics students suggested the following research interests

1) Working with data
   - Common tasks: reading in data, creating new variables, data subsets, etc.
   - Example packages: base, tidyverse, data.table, etc.
2) Applied econometrics
   - Common tasks: descriptive analysis, regression analysis, etc.
   - Example packages: stats, plm, lmtest, sandwich, etc.
3) Economic Evaluation
   - Common tasks: model building (Markov, Microsim, etc.), sensitivity analysis, etc.
   - Example packages: base, stats, ggplot2, etc.

The use of R in research

- Vilhuber, Turrito, and Welch (2020) found that most of the supplements in the American Economic Review (AER) use STATA.
  - They also found that open source software (including R) had a small presence.
- Masuadi et al. (2021) found that SPSS was the most used software in health services research.
  - R was used less often (9.4%).
- Jalal et al. (2017) showed that the use of R was increasing in the field of health decision making, but proportion was lower compared to other software such as Microsoft Excel.

Outline for Part II

- R as a programming language
  - functions, for loops, flow control, matrix algebra, etc.
  - examples from econometrics, economic evaluation, and data analysis will be provider throughout
- R Markdown for beamer presentations
- R Resources

R as a programming language

- The programming language in R is object oriented
  - Roughly speaking, this means that data, variables, vectors, matrices, characters, arrays, etc. are treated as "objects" of a certain "class" that are created throughout the analysis and stored by name.
  - We then apply "methods" for certain "generic functions" to these objects
- It can be used for tasks outside of data analysis, similar to other programming languages
- The language itself was designed for programming with data See Kleiber and Zeileis (2008) for more

For Statement in R

General syntax:

for(var in seq) expr

- var = variable used to index loop number (commonly "i")
- seq = sequence to loop over (i.e., 1:L, where L is the end of the loop)
- expr = Expression to evaluate

Example:

```
for (i in 1:3)
{
print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

Foreach function

Package: foreach

- Provides looping structure that returns a value
- General syntax:

foreach(..., .combine, .init, .final = NULL, .inorder = TRUE, .multicombine = FALSE, .maxcombine = if (.multicombine) 100 else 2, .errorhandling = c("stop", "remove", "pass"), .packages = NULL, .export = NULL, .noexport = NULL, .verbose = FALSE)

Online help file: https://www.rdocumentation.org/packages/foreach/versions/1.5.2/topics/foreach

Foreach function

Applied Example

- Read in 10 waves of a dataset
- Could be survey waves, administrative dataset, etc.
- Contain the same variables in each wave
- Goal is combine (stack) the datasets

Foreach function (code from example)

```
# Load libraries ----
library(foreach)
library(data.table)

# Load data ----
files <- paste0("Yearly Files/wdata", paste0(seq(1, 10, 1), ".csv")) # File locations and names
data.all <- foreach(i=1:length(files), .combine = "rbind") %do%
{
return(read.csv(files[i]))  # Object to return
}
```

Writing Functions in R

- Using R for data analysis typically involves the utilization a sequence of commands for inputs to produce outputs
- These sequences can be wrapped into a function, which can be called to avoid repeating these sequences by hand
- Functions can be used for many different purposes, including data formatting, Markov and Microsim models, applied econometrics, etc.

Writing Functions in R

```
my.toy.fun = function(x,y)
{
  z <- x + y
  z.squared <- z^2
  return(z.squared)
}

my.toy.fun(x = 4, y = 5)
```

```
## [1] 81
```

Function Example

- Instrumental Variable (IV) estimation is typically estimated using two-stage least squares (2SLS), which uses a linear function in both stages
- In many health applications, the second stage is non-linear
- The control function approach, or two-stage residual inclusion, has been suggested as an alternative to 2SLS in non-linear models (see for example Papke and Wooldridge (2008); Basu, Coe, and Chapman (2018))
- Standard errors need to be adjusted for the first step estimation, which can be done by jointly bootstrapping both steps (Cameron and Trivedi 2022). This can be achieved by writing a function in combination with the boot package.

Applied Example

- Analysis of health expenditure data for individuals 65+ in the U.S. (Medicare) (Cameron and Trivedi (2022) Chapter Seven)
- Outcome is log of total out-of-pocket expenditures on prescribed medications
- Endogenous variable is indicator for whether individual holds employer or union-sponsored health insurance
- Instrument is ratio of individual's social security income to income from all sources
- Other model variables include number of chronic conditions, age, female, whether black or Hispanic, and log of annual household income (in thousands of dollars).
- Data can be downloaded from here: https://www.stata-press.com/data/mus2.html

Code for applied example

```
rm(list = ls())
library(haven)
library(boot)
library(AER)

ivdata.all <- read_dta("mus2/mus207mepspresdrugs.dta",
                       col_select = c("ldrugexp", "hi_empunion", "ssiratio",
                                      "totchr", "age", "female", "blhisp", "linc")) # Load dataset and variables

ivdata <- ivdata.all[complete.cases(ivdata.all),] # Remove observations with missing values

boot.fun <- function(data, i) # Bootstrapping function
{
  boot.data <- data[i,]

  first <- lm(hi_empunion ~ ssiratio + totchr + age + female + blhisp + linc,
              data = boot.data) # First stage

  boot.data$v1hat <- residuals(first) # Estimate residuals from first stage

  second <- lm(ldrugexp ~ hi_empunion + totchr + age + female + blhisp + linc + v1hat,
               data = boot.data) # Second stage

  return(coefficients(second))
}

boot.results <- boot(ivdata, boot.fun, R = 500)

summary(boot.results) # Bootstrap Results
summary(ivreg(ldrugexp ~ hi_empunion + totchr + age + female + blhisp + linc |
                ssiratio + totchr + age + female + blhisp + linc,
              data = ivdata)) # Compare results with ivreg
```

Summary of applied example

- Bootstrapped standard errors are similar to those produced by 2SLS
- Control function example can be extended to case where second stage is non-linear
  - Example: Papke and Wooldridge (2008) investigate the impact of school funding on student math test pass rates
  - The authors use a control function approach with a fractional response model for the second stage
- Method can also be extended to discrete endogenous variables, but careful attention needs to be placed on the form of the residual. See, for example, Basu, Coe, and Chapman (2018), Geraci, Fabbri, and Monfardini (2018)

List function in R

- Generic vectors where each element can be virtually any type of object (Kleiber and Zeileis 2008)
- This allows us to combine scalars, numeric vectors, data frames, etc. into one object
- Objects can be extracted from list by name through '$' or [[ ]] (element-number wise extraction)

Example:

```
my.list <- list("id" = seq(1, 10, 1), "Explanation" = "Patient ID")

my.list$id # Extract ID
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
my.list[[1]] # Same as above
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Applied Example

Common task: Exporting regression results in R

- In R, some functions are available by default (OLS, GLM, etc.) whereas others are contained in packages written by other users
- This implies that user-written packages designed to work for default R packages may not work for other models
- In addition, some packages are designed to work with LaTeX, which is not necessarily helpful for users who work with Microsoft Word
- Finally, it may be difficult to achieve the desired formatting of results with existing packages

Is it possible to write our own function to create a regression results table?

Bonus challenge: Must be compatible with MS Word and LaTeX

A note on R and LaTeX

- You need to have a LaTeX distribution installed to typeset using this approach
- If you are an active LaTeX user, you likely have MiKTeX, MacTeX, or similar installed already, so no further action is needed
- If you're interested in LaTeX but don't have a LaTeX distribution, you may consider installing TinyTeX from the tinytex R package. To install through R:

```
tinytex::install_tinytex()
```

For more details, please see:

https://bookdown.org/yihui/rmarkdown-cookbook/install-latex.html

Applied Example (Continued)

- Analysis of Health Expenditure Data in Jones et al. (2013) Chapter Three
- The data covers the medical expenditures of US citizens aged 65 years and older who qualify for health care under Medicare.
  - Outcome of interest is total annual health care expenditures (measured in US dollars).
  - Other key variables are age, gender, household income, supplementary insurance status (insurance beyond Medicare), physical and activity limitations and the total number of chronic conditions.
  - Data can be downloaded from here (mus03data.dta): https://www.stata-press.com/data/musr.html

Regression Results Function Code

```r
# Load Libraries ----
library(lmtest)
library(sandwich)
library(foreign)
# Regression Results Function ----
reg.results.fun = function(model, digits)
{
  reg.results <- coeftest(model)
  beta <- reg.results[,1]
  se <- reg.results[,2]
  sig.stars <- symnum(reg.results[,4], corr = FALSE, na = FALSE,
                      cutpoints = c(0, 0.001, 0.01, 0.05, 0.1, 1),
                      symbols = c("***", "**", "*", ".", ""))

  results.table <- data.frame(cbind("Variable" = rownames(reg.results),
                                    "Beta" = paste(round(beta, digits), sig.stars, sep = ""),
                                    "SE" = round(se, digits)), row.names = NULL)

  return(list("coefficients" <- beta,
              "results.table" <- results.table))
}
# Load Data ----
cost.data.all <- read.dta("Expenditure Data/mus03data.dta")
cost.data <- cost.data.all[cost.data.all$totexp>0,]
# Regression ----
ols.cost.data <- lm(totexp ~ age + female + income + suppins + phylim + actlim + totchr, data = cost.data)
summary(ols.cost.data)
# Export Regression Results ----
ols.cost.data.results.all <- reg.results.fun(ols.cost.data, 2)
ols.cost.data.results <- ols.cost.data.results.all$results.table
save(ols.cost.data.results, file = "ols.cost.data.results.RData")
```

Code for R Markdown

```
# ---
# title: "Untitled"
# output:
#   pdf_document: default
#   html_document: default
#   word_document: default
# ---
#
# ```{r setup, include=FALSE}
# knitr::opts_chunk$set(echo = FALSE)
# ```
#
# ## Regression
#
# ```{r regression}
# load("ols.cost.data.results.RData")
# knitr::kable(ols.cost.data.results)
# ```
```

Flow Control in R: If Statements

An if/else statement in R takes the general form:

if (cond) { R code if true } else { R code if not true }

Example:

```
x <- 5

if(x>=4)
{
  print("x is greater than or equal to four")
} else {
  print("x is less than four")
}
```

```
## [1] "x is greater than or equal to four"
```

Applied Example (Continued)

- Common task: Creating output using heteroskedasticity robust standard errors
  - Typical summary() function in R only gives output using the standard OLS variance matrix (i.e., assuming homoskedasticity)
  - Can we modify our regression function to give results using heteroskedasticity robust variance matrix?
  - Can we also add a test for heteroskedasticity in the regression results function?

Code for applied example

```r
# Regression Results Function ----
reg.results.fun <- function(model, digits, robust = FALSE)
{
  if (robust==TRUE){
    reg.results <- coeftest(model, vcovHC(model, type = "HC1"))
    beta <- reg.results[,1]
    se <- reg.results[,2]
    sig.stars <- symnum(reg.results[,4],
                        cutpoints = c(0, 0.001, 0.01, 0.05, 0.1, 1),
                        symbols = c("***", "**", "*", ".", ""))

    results.table <- data.frame(cbind("Variable" = rownames(reg.results),
                                      "Beta.Robust" = paste(round(beta,digits), sig.stars, sep = ""),
                                      "SE.Robust" = round(se, digits)), row.names = NULL)
  } else {
  reg.results <- coeftest(model)
  beta <- reg.results[,1]
  se <- reg.results[,2]
  sig.stars <- symnum(reg.results[,4],
                      cutpoints = c(0, 0.001, 0.01, 0.05, 0.1, 1),
                      symbols = c("***", "**", "*", ".", ""))

  results.table <- data.frame(cbind("Variable" = rownames(reg.results),
                                    "Beta" = paste(round(beta,digits), sig.stars, sep = ""),
                                    "SE" = round(se, digits)), row.names = NULL)
  }

  test.hetero <- bptest(model)

  return(list("coefficients" = beta,
              "results.table" = results.table,
              "Het.Test" = test.hetero))
}
```

Code for applied example (contined)

```
# Load Data ----
cost.data.all <- read.dta("Expenditure Data/mus03data.dta")
cost.data <- cost.data.all[cost.data.all$totexp>0,]

# Regression ----
ols.cost.data <- lm(totexp ~ age + female + income + suppins + phylim +
                      actlim + totchr, data = cost.data)

# Export Regression Results ----
ols.cost.data.results.all <- reg.results.fun(ols.cost.data, 2)
ols.cost.data.results <- ols.cost.data.results.all$results.table

ols.cost.data.results.robust.all <- reg.results.fun(ols.cost.data, 2, robust = TRUE)
ols.cost.data.results.robust <- ols.cost.data.results.robust.all$results.table

ols.cost.data.results.combine <- merge(ols.cost.data.results,
                                       ols.cost.data.results.robust,
                                       by = "Variable")
```

Matrix Algrebra

Matrix algebra is often used to perform calculations in decision-making models. It can also be helpful to implement a statistical method that is not available in R.

- Matrix multiplication
  - %*%
- Element wise-multiplication
  - *
- Matrix inverse
  - solve()

Applied Example

Markov model with the following transitional probability matrix

$$\begin{bmatrix} \mathbf{H} & \mathbf{S} & \mathbf{D} \\ 0.9 & 0.08 & 0.02 \\ 0 & 0.8 & 0.2 \\ 0 & 0 & 1 \end{bmatrix}$$

Everybody in the model starts in H

Can we use matrix algebra to solve for the second period health states?

Matrix Algebra Code

```
rm(list = ls()) # Clear memory
A = matrix(0, nrow = 2, ncol = 3) #2X3 matrix of 0s (Two time periods, 3 health states)
P = rbind(c(0.9, 0.08, 0.02),
                  c(0,0.8, 0.2),
                   c(0,0,1)) #3x3 transitional probability matrix
A[1,] = c(1, 0, 0) #Initial health states
A[2,] = A[1,] %*% P #Health states in period 2
print(A) #Display matrix
```

```
##      [,1] [,2] [,3]
## [1,]  1.0 0.00 0.00
## [2,]  0.9 0.08 0.02
```

Parallel Processing in R

- Parallel computing: From Wikipedia: "Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time."
  - See here for more: https://en.wikipedia.org/wiki/Parallel_computing
- Modern day computers typically contain:
  - Single-core
  - Multicore (Dual, Quad, Hexa, Octo, etc.)
  - May also contain hyperthreading

Parallel Processing in R

Parallel processing can be used in many situations, including:

- Bootstrapping
- Microsimulation models
- Monte Carlo experiments
- Probabilistic Sensitivity Analysis

By utilizing parallel processing, we can significantly speed up the processing time of our calculations

Applied Example (stacking data. . . but faster)

- We experienced long load times when loading and stacking the previous datasets
- Can we use parallel processing to speed up the process?

Code for applied example

```
rm(list = ls())
library(foreach)
library(doParallel)

# Register Parallel ----
ncores = 2
registerDoParallel(cores = ncores)

# Load data ----
files <- paste0("Yearly Files/wdata", paste0(seq(1, 10, 1), ".csv"))
## Serial ----
t1 = Sys.time()
data.all <- foreach(i=1:length(files), .combine = "rbind") %do%
  {
    return(read.csv(files[i]))
  }
Sys.time() - t1
```

```
## Time difference of 23.69767 secs
```

```
## Parallel ----
t2 = Sys.time()
data.all <- foreach(i=1:length(files), .combine = "rbind") %dopar%
  {
    return(read.csv(files[i]))
  }
Sys.time() - t2
```

```
## Time difference of 12.41112 secs
```

Making Beamer Presentations with R Markdown

- Similar to typsetting word documents, R Markdown can also be used to create presentations
- This includes the LaTeX based beamer presentations
- Outside of R Markdown (i.e. through a LaTeX editor), creating presentations in beamer can be tedious and tables need to be in a certain format to be included
- R Markdown can simplify the process

Making Beamer Presentations with R Markdown (Continued)

- Go to File → New File → R Markdown → Presentation –> PDF (Beamer)
- New slides can be created with one line of code: #
  - Similar to documents, R chunks can be used to work with R and include R objects in output
- List of possible themes and colours can be found here: https://hartwork.org/beamer-theme-matrix/

Applied Example: Beamer Presentation

Can we create a beamer presentation in R Markdown that includes the regression results we generated previously?

Beamer Code

```
# ---
# title: "Untitled"
# header-includes:
#   - \usepackage{booktabs}
# output:
#   beamer_presentation:
#     theme: "Madrid"
# ---
# ```{r setup, include=FALSE}
# knitr::opts_chunk$set(echo = FALSE)
# ```
# # Regression
# ```{r regression}
# load("ols.cost.data.results.combine.RData")
# knitr::kable(ols.cost.data.results.combine, format = "latex", booktabs = T)
# ```
```

Conclusions

- R is a flexible programming language that can be used to perform tasks related to data manipulation, applied econometrics, and economic evaluation
- Contains a wide array of pre-canned routines to implement the statistical analysis we want to perform
- R Markdown can be used to create reports and presentations within R.

Additional Resources

- Applied Econometrics with R
  - Kleiber, C., & Zeileis, A. (2008). Applied econometrics with R. Springer Science & Business Media.
- R for Medical Decision Making
  - Jalal, H., Pechlivanoglou, P., Krijkamp, E., Alarid-Escudero, F., Enns, E., & Hunink, M. M. (2017). An overview of R in health decision sciences. Medical decision making, 37(7), 735-746.
  - Krijkamp, E. M., Alarid-Escudero, F., Enns, E. A., Jalal, H. J., Hunink, M. M., & Pechlivanoglou, P. (2018). Microsimulation modeling for health decision sciences using R: a tutorial.
  - Alarid-Escudero, F., Krijkamp, E. M., Enns, E. A., Yang, A., Hunink, M. G., Pechlivanoglou, P., & Jalal, H. (2021). A Tutorial on time-dependent cohort state-transition models in R using a cost-effectiveness analysis example. arXiv preprint arXiv:2108.13552.
- Posit Cheatsheets: - https://posit.co/resources/cheatsheets/

Conclusion

Thank you for listening. Good luck with R!

References I

Basu, Anirban, Norma B Coe, and Cole G Chapman. 2018. "2SLS versus 2SRI: A ppropriate methods for rare outcomes and/or rare exposures." *Health Economics* 27 (6): 937–55.

Cameron, Adrian Colin, and Pravin K Trivedi. 2022. *Microeconometrics using stata: Volume I*. Stata Press.

Geraci, Andrea, Daniele Fabbri, and Chiara Monfardini. 2018. "Testing Exogeneity of Multinomial Regressors in Count Data Models: Does Two-Stage Residual Inclusion Work?" *Journal of Econometric Methods* 7 (1): 20140019.

Jalal, Hawre, Petros Pechlivanoglou, Eline Krijkamp, Fernando Alarid-Escudero, Eva Enns, and MG Myriam Hunink. 2017. "An overview of R in health decision sciences." *Medical Decision Making* 37 (7): 735–46.

References II

Kleiber, Christian, and Achim Zeileis. 2008. *Applied econometrics with R*. Springer Science & Business Media.

Masuadi, Emad, Mohamud Mohamud, Muhannad Almutairi, Abdulaziz Alsunaidi, Abdulmohsen K Alswayed, and Omar F Aldhafeeri. 2021. "Trends in the Usage of Statistical Software and Their Associated Study Designs in Health Sciences Research: A Bibliometric Analysis." *Cureus* 13 (1).

Papke, Leslie E, and Jeffrey M Wooldridge. 2008. "Panel Data Methods for Fractional Response Variables with an Application to Test Pass Rates." *Journal of Econometrics* 145 (1-2): 121–33.

Vilhuber, Lars, James Turrito, and Keesler Welch. 2020. "Report by the AEA Data Editor." *AEA Papers and Proceedings* 110 (May): 764–75. https://doi.org/10.1257/pandp.110.764.