Introduction To R
Part II

Adrian Rohit Dass

January 14th, 2022

Recap: Outline from Part I

- Why use R?
- R Basics
- R for Database Management
    - Reading-in data, merging datasets, reshaping, recoding variables, sub-setting data, etc.
- R for Statistical Analysis
    - Descriptive and Regression Analysis
- Other topics in R
    - Tidyverse
    - Parallel Processing
    - R Studio
        - R Markdown
- Applied Example
- R Resources

Results from Survey

- In December 2021, Dr. Hancock Howard polled the students registered in this course for potential topics to cover in this part II session
- The results of the survey can be roughly grouped into the following categories
  - Applied econometrics
  - Economic Evaluation
  - Working with data
- Given the wide scope of areas of interest, it may not be helpful to focus on one particular topic in depth (i.e. models for medical decision making)

Outline

- The purpose of this talk is to provide an overview of the use of R for programming
  - This will allow us to write our own routines to go beyond the pre-canned functions covered in session 1
  - This should also be useful for all users of R, regardless of research interest
- We will also cover the integration of R and document typsetting programs (MS Word and LaTeX), parallel processing in R, and creating LaTeX presentations using RStudio

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 & Zeileis (2008) for more

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 (Continued)

```
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 inputs

Temporary variables that are not stored in R memory

Function output(s)

List function in R

- Generic vectors where each element can be virtually any type of object (Kleiber & 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)

Ex:

```
my.list = list("id" = seq(1, 10, 1), "Explanation" = "Patient ID")
my.list$id # Extract ID
my.list[[1]] # Same as above, but different method
```

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

```
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))
}
```

R Markdown code

~~~
---
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
}
```

Can be omitted if empty

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?

Updated Regression Results Function Code

```
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], 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.Robust" = paste(round(beta, digits), sig.stars, sep = ""),
                          "SE.Robust" = round(se, digits)), row.names = NULL)
 } else { same as before (see previous slide)}
 test.hetero = bptest(model)
 return(list("coefficients" = beta,
          "results.table" = results.table,
          "Het.Test" = test.hetero))
}
```

Flow Control in R: For Loops

- Main argument in for loop is typically a sequence.
- Takes the general form:

```
for(var in seq)
{
R code to loop
}
```

Applied Example: Monte Carlo Experiment

- We will conduct a Monte Carlo study to investigate the performance of OLS
- The data generating process takes the following form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

Where β_0 = 1, β_1 = 2, β_2 = 3, $\epsilon \sim N(0,1)$

- Monte Carlo studies are generally used to study the performance of methods in estimating the true population parameters
- May be of interest to those learning econometrics as well as those involved in models for decision making (i.e. probabilistic sensitivity analysis).

Code for Monte Carlo Exercise

```
rm(list = ls())
set.seed(12345)
b0 = 1
b1 = 2
b2 = 3
n = 1000
S = 10000
beta.hat = matrix(0, nrow = S, ncol = 3)
p = Sys.time()
for (i in 1:S)
{
  x1 = rnorm(n, mean = 1, sd =1)
  x2 = rnorm(n, mean = 1, sd =1)
  e = rnorm(n, mean = 0, sd =1)
  y = b0 + b1*x1 + b2*x2 + e
  data = data.frame(cbind(y, x1, x2))
  ols = lm(y~x1 + x2, data = data)
  beta.hat[i,] = coefficients(ols)
}
comp.time = Sys.time() - p
comp.time
mc.results = colMeans(beta.hat)
mc.results
```

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 (Continued)

- 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

Parallel Processing in R (Continued)

- There are many packages to perform parallel processing in R, including
- parallel
  - Available in R by default
  - Handles large chunks of computations in parallel
  - https://stat.ethz.ch/R-manual/R-devel/library/parallel/doc/parallel.pdf
- doParallel
  - “parallel backend” for the “foreach” package
  - provides a mechanism needed to execute foreach loops in parallel
  - https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf

Parallel Processing in R (Continued)

- There are many ways for perform parallel processing, including forking and sockets
- Forking is only available on Unix operating systems, and thus cannot be done using Windows.
  - It is generally faster than sockets, and all variables and packages loaded in R main session are automatically passed to the other cores
  - Generally the preferred method on these types of operating systems
- Sockets are available on Unix and Windows operating systems
  - In this case, variables and packages are not automatically passed to the other cores
- See https://bookdown.org/rdpeng/rprogdatascience/parallel-computation.html for more

Applied Example: Monte Carlo Experiment

- Using the same Monte Carlo experiment, can we speed up the calculations using parallel processing?

Monte Carlo Code

```
library(doParallel)
ncores = 2
registerDoParallel(cores = ncores)
p2 = Sys.time()
beta.hat.par = foreach(i=1:S, .combine = 'rbind', .multicombine = TRUE) %dopar%
  {
    x1 = rnorm(n, mean = 1, sd =1)
    x2 = rnorm(n, mean = 1, sd =1)
    e = rnorm(n, mean = 0, sd =1)
    y = b0 + b1*x1 + b2*x2 + e
    data = data.frame(cbind(y, x1, x2))
    ols = lm(y~x1 + x2, data = data)
    beta.hat = coefficients(ols)
  }

comp.time2 = Sys.time() - p2
comp.time2
```

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 (i.e. \begin{frame} \end{frame} to create slides, \begin{itemize} \end{itemize} to create bulleted lists, etc.) 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?

R Markdown 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

- Using the programming capabilities within R allows us to write our own functions that expand the functionality of R
  - This is particularly helpful when no R package/function is available
- Use of parallel processing allows for speedier computation times for various tasks in R, including applied econometrics and decision making models
- R and LaTeX can work in harmony with each other to produce beautiful documents and presentations with R Markdown

Additional Resources

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.

Applied Econometrics with R

- Kleiber, C., & Zeileis, A. (2008). *Applied econometrics with R*. Springer Science & Business Media.

RStudio Cheatsheets

- https://www.rstudio.com/resources/cheatsheets/

Thanks for Listening

Good luck with R!