Applied Econometrics with R

Chapter 7

Programming

Programming

Overview

Overview

Data analysis typically involves

- using or writing software that can perform the desired analysis,
- a sequence of commands or instructions that apply the software to the data, and
- documentation of the commands and their output.

Here: Go beyond using off-the-shelf software. Use R tools for

- simulation (of power functions),
- bootstrapping a regression model,
- maximizing a likelihood,
- reproducible econometrics using `Sweave()`.

Programming

Simulations

Simulations

Simulations typically involve 3 steps:

- simulating data from some data-generating process (DGP),
- evaluating the quantities of interest (e.g., rejection probabilities, parameter estimates, model predictions), and
- iterating the first two steps over a number of different scenarios.

Example: compare power of two tests for autocorrelation

- Durbin-Watson test
- Breusch-Godfrey test

Recall: Durbin-Watson test is not valid in presence of lagged dependent variables.

Simulations

Data-generating processes are

$$\begin{aligned} \text{trend:} \quad y_i &= \beta_1 + \beta_2 \cdot i + \varepsilon_i, \\ \text{dynamic:} \quad y_i &= \beta_1 + \beta_2 \cdot y_{i-1} + \varepsilon_i, \end{aligned}$$

- regression coefficients $\beta = (0.25, -0.75)^\top$,
- $\{\varepsilon_i\}$, $i = 1, \dots, n$, is stationary AR(1), derived from standard normal innovations and with lag 1 autocorrelation ϱ.
- starting values are 0 (for both y and ε).

Goal: Analyze power properties of both tests (for size $\alpha = 0.05$) on both DGPs with

- autocorrelations $\varrho = 0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 0.99$ and
- sample sizes $n = 15, 30$.

Simulations

Step 1: DGP with all parameters

```
R> dgp <- function(nobs = 15, model = c("trend", "dynamic"),
+    corr = 0, coef = c(0.25, -0.75), sd = 1)
+ {
+    model <- match.arg(model)
+    coef <- rep(coef, length.out = 2)
+
+    err <- as.vector(filter(rnorm(nobs, sd = sd), corr,
+      method = "recursive"))
+    if(model == "trend") {
+      x <- 1:nobs
+      y <- coef[1] + coef[2] * x + err
+    } else {
+      y <- rep(NA, nobs)
+      y[1] <- coef[1] + err[1]
+      for(i in 2:nobs)
+        y[i] <- coef[1] + coef[2] * y[i-1] + err[i]
+      x <- c(0, y[1:(nobs-1)])
+    }
+    return(data.frame(y = y, x = x))
+ }
```


Simulations

Step 2: evaluation for a single scenario

```
R> simpower <- function(nrep = 100, size = 0.05, ...)
+ {
+   pval <- matrix(rep(NA, 2 * nrep), ncol = 2)
+   colnames(pval) <- c("dwtest", "bgtest")
+
+   for(i in 1:nrep) {
+     dat <- dgp(...)
+     pval[i,1] <- dwtest(y ~ x, data = dat,
+       alternative = "two.sided")$p.value
+     pval[i,2] <- bgtest(y ~ x, data = dat)$p.value
+   }
+
+         return(colMeans(pval < size))
+ }
```

Simulations

Step 3: iterated evaluation over all scenarios

```
R> simulation <- function(corr = c(0, 0.2, 0.4, 0.6, 0.8, 0.9,
+    0.95, 0.99), nobs = c(15, 30), model = c("trend", "dynamic"),
+    ...)
+  {
+    prs <- expand.grid(corr = corr, nobs = nobs, model = model)
+    nprs <- nrow(prs)
+
+    pow <- matrix(rep(NA, 2 * nprs), ncol = 2)
+    for(i in 1:nprs) pow[i,] <- simpower(corr = prs[i,1],
+      nobs = prs[i,2], model = as.character(prs[i,3]), ...)
+
+    rval <- rbind(prs, prs)
+    rval$test <- factor(rep(1:2, c(nprs, nprs)),
+      labels = c("dwtest", "bgtest"))
+    rval$power <- c(pow[,1], pow[,2])
+    rval$nobs <- factor(rval$nobs)
+    return(rval)
+  }
```


Simulations

Now set random seed (reproducibility!) and call `simulation()`:

```
R> set.seed(123)
R> psim <- simulation()
```

Remarks:

- `simulation()` calls `simpower()`, and `simpower()` calls `dgp()`.
- Argument `...` is simple mechanism for passing on further arguments to other functions – in `simpower()` to `dgp()`.
- Precision from only 100 replications not sufficient for professional applications.

Simulations

Inspect simulation results:

```
R> tab <- xtabs(power ~ corr + test + model + nobs, data = psim)
R> ftable(tab, row.vars = c("model", "nobs", "test"),
+    col.vars = "corr")
                     corr    0  0.2  0.4  0.6  0.8  0.9 0.95 0.99
model   nobs test
trend   15   dwtest       0.05 0.10 0.21 0.36 0.55 0.65 0.66 0.62
             bgtest       0.07 0.05 0.05 0.10 0.30 0.40 0.41 0.31
        30   dwtest       0.09 0.20 0.57 0.80 0.96 1.00 0.96 0.98
             bgtest       0.09 0.09 0.37 0.69 0.93 0.99 0.94 0.93
dynamic 15   dwtest       0.02 0.01 0.01 0.00 0.00 0.00 0.01 0.02
             bgtest       0.05 0.01 0.06 0.12 0.17 0.16 0.22 0.24
        30   dwtest       0.01 0.02 0.01 0.02 0.00 0.00 0.04 0.21
             bgtest       0.02 0.03 0.11 0.39 0.60 0.64 0.60 0.76
```


Simulations

Remarks:

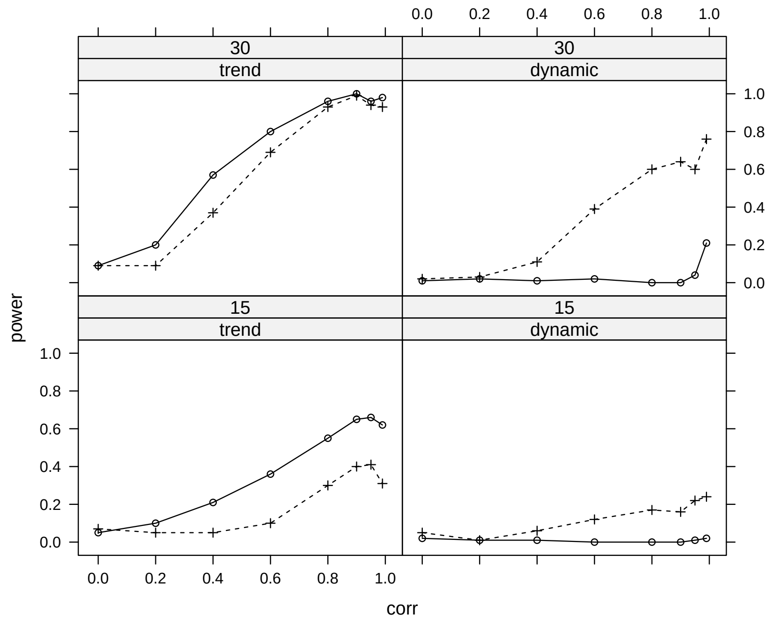
- `xtabs()` helps to turn "`data.frame`" into "`table`" that classifies `power` outcome by the four design variables.
  Use `ftable()` for printing resulting four-way table (creates "flat" two-way table).
- Supplying `corr` as column variable and `test` as last row variable as table is aimed at comparing power curves

Graphical comparison: using trellis graphics.

```
R> library("lattice")
R> xyplot(power ~ corr | model + nobs, groups = ~ test,
+    data = psim, type = "b")
```

Scatterplot for `power ~ corr`, conditional on combinations of `model` and `nobs`, grouped by `test` within each panel.

Simulations



Simulations

Details:

- **lattice** (Sarkar 2008) implements trellis layouts.
- Written in the **grid** graphics system (Murrell 2005).
- More flexible (and more complex) than default R graphics.
- `xyplot()` generates trellis scatterplots.

Results:

- Durbin-Watson test somewhat better in trend model. Advantage over Breusch-Godfrey test diminishes with increasing ϱ and n.
- For dynamic model, Durbin-Watson test has almost no power except for very high correlations. Breusch-Godfrey test performs acceptably.

Programming

Bootstrapping a Linear Regression

Bootstrapping a Linear Regression

Idea:

- Conventional regression output relies on asymptotic approximations. Often not very reliable in small samples or models with substantial nonlinearities.
- Possible remedy is bootstrapping.

In R:

- basic recommended package is **boot** (Davison and Hinkley, 1997)
- function `boot()` implements classical nonparametric bootstrap (sampling with replacement) and other resampling techniques.

Bootstrapping a Linear Regression

Bootstrap and econometrics:

- Observational data are standard in economics, hence consider responses *and* regressors as random.
- Suggests to use pairs bootstrap (resample observations). Method should give reliable standard errors even under (conditional) heteroskedasticity.

Example: bootstrap standard errors and confidence intervals for `Journals` data (Stock and Watson 2007) by case-based resampling.

Basic regression was

```
R> data("Journals")
R> journals <- Journals[, c("subs", "price")]
R> journals$citeprice <- Journals$price/Journals$citations
R> jour_lm <- lm(log(subs) ~ log(citeprice), data = journals)
```


Bootstrapping a Linear Regression

Function `boot()` takes several arguments, required are

- `data` – the data set,
- `R` – the number of bootstrap replicates,
- `statistic` – a function returning the statistic to be bootstrapped. Function must take data set and index vector providing the indices of the observations included in current bootstrap sample.

Example: required statistic given by convenience function

```
R> refit <- function(data, i)
+    coef(lm(log(subs) ~ log(citeprice), data = data[i,]))
```

Now call `boot()`:

```
R> library("boot")
R> set.seed(123)
R> jour_boot <- boot(journals, refit, R = 999)
```


Bootstrapping a Linear Regression

```
R> jour_boot

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = journals, statistic = refit, R = 999)


Bootstrap Statistics :
     original       bias    std. error
t1*   4.7662 -0.0010560      0.05545
t2*  -0.5331 -0.0001606      0.03304
```

```
R> coeftest(jour_lm)

t test of coefficients:

               Estimate Std. Error t value Pr(>|t|)
(Intercept)      4.7662     0.0559    85.2   <2e-16
log(citeprice)  -0.5331     0.0356   -15.0   <2e-16
```


Bootstrapping a Linear Regression

```
R> boot.ci(jour_boot, index = 2, type = "basic")

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = jour_boot, type = "basic", index = 2)

Intervals :
Level      Basic
95%   (-0.5952, -0.4665 )
Calculations and Intervals on Original Scale
```

```
R> confint(jour_lm,  parm = 2)

                 2.5 %  97.5 %
log(citeprice) -0.6033 -0.4628
```

Results: Conventional and bootstrap standard errors and confidence intervals (for slope coefficient) are essentially identical, i.e., conventional versions valid.

Bootstrapping a Linear Regression

Remarks:

- **boot** has further functions for resampling, e.g. `tsboot()` for block resampling from time series.
- Block resampling from time series also via `tsbootstrap()` from **tseries**.
- Maximum entropy bootstrap in **meboot**.

Programming

Maximizing a Likelihood

Maximizing a Likelihood

Example: Generalized Cobb-Douglas production function (Zellner and Revankar, *JAE* 1998)

$$Y_i e^{\theta Y_i} = e^{\beta_1} K_i^{\beta_2} L_i^{\beta_3},$$

- can be seen as transformation applied to the dependent variable encompassing the level (with classical Cobb-Douglas for $\theta = 0$),
- allows returns to scale to vary with the level of output.

Multiplicative error gives logarithmic form

$$\log Y_i + \theta Y_i = \beta_1 + \beta_2 \log K_i + \beta_3 \log L_i + \varepsilon_i.$$

$\rightarrow$ nonlinear in parameters, only for known θ can estimate by OLS.

Maximizing a Likelihood

Solution: simultaneous estimation of regression coefficients and transformation parameter using maximum likelihood (ML).

Assumption: $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ i.i.d. Resulting (log-)likelihood is

$$\mathcal{L} = \prod_{i=1}^{n} \left\{ \phi(\varepsilon_i/\sigma) \cdot \frac{1 + \theta Y_i}{Y_i} \right\},$$

$$\ell = \sum_{i=1}^{n} \{\log(1 + \theta Y_i) - \log Y_i\} + \sum_{i=1}^{n} \log \phi(\varepsilon_i/\sigma).$$

where

- $\varepsilon_i = \log Y_i + \theta Y_i - \beta_1 - \beta_2 \log K_i - \beta_3 \log L_i$
- $\phi(\cdot)$ is PDF of standard normal distribution.
- Note $\partial\varepsilon_i/\partial Y_i = (1 + \theta Y_i)/Y_i$.

Maximizing a Likelihood

Task: Function maximizing log-likelihood wrt $(\beta_1, \beta_2, \beta_3, \theta, \sigma^2)$. Use `Equipment` data from Greene (2003).

3 Steps:

- code the objective function,
- obtain starting values for an iterative optimization, and
- optimize the objective function using the starting values.

Remarks:

- Since `optim()` by default performs minimization, we minimize the *negative* log-likelihood.
- Our function `nlogL()` is function of vector parameter `par` comprising five elements.
- R provides functions for the logarithms of standard distributions, including normal density `dnorm(..., log = TRUE)`.

Maximizing a Likelihood

Step 1: code log-likelihood

```
R> data("Equipment", package = "AER")
R> nlogL <- function(par) {
+   beta <- par[1:3]
+   theta <- par[4]
+   sigma2 <- par[5]
+
+   Y <- with(Equipment, valueadded/firms)
+   K <- with(Equipment, capital/firms)
+   L <- with(Equipment, labor/firms)
+
+   rhs <- beta[1] + beta[2] * log(K) + beta[3] * log(L)
+   lhs <- log(Y) + theta * Y
+
+   rval <- sum(log(1 + theta * Y) - log(Y) +
+     dnorm(lhs, mean = rhs, sd = sqrt(sigma2), log = TRUE))
+   return(-rval)
+ }
```


Maximizing a Likelihood

Step 2: obtain starting values

- fit classical Cobb-Douglas form by OLS,
- starting value for $\beta = (\beta_1, \beta_2, \beta_3)^\top$ is resulting vector of coefficients, `coef(fm0)`,
- starting value for θ is 0,
- starting value for disturbance variance is mean of squared residuals from Cobb-Douglas fit.

Thus

```
R> fm0 <- lm(log(valueadded/firms) ~ log(capital/firms) +
+    log(labor/firms), data = Equipment)
R> par0 <- as.vector(c(coef(fm0), 0, mean(residuals(fm0)^2)))
```


Maximizing a Likelihood

Step 3: search for the optimum from starting values.

```
R> opt <- optim(par0, nlogL, hessian = TRUE)
```

- By default, `optim()` uses Nelder-Mead method (further algorithms available).
- Set `hessian = TRUE` to obtain standard errors.

Extract estimates, standard errors and value of objective function:

```
R> opt$par
[1] 2.91469 0.34998 1.09232 0.10666 0.04275
R> sqrt(diag(solve(opt$hessian)))[1:4]
[1] 0.36055 0.09671 0.14079 0.05850
R> -opt$value
[1] -8.939
```

Results suggest that θ is greater than 0.

Maximizing a Likelihood

Remarks:

- For practical purposes, solution needs to be verified (local optimum?).
- Function is specialized to data set under investigation.
  If a reusable function is needed, a proper function `GCobbDouglas(formula, data, ...)` should be coded.

Programming

Reproducible Econometrics Using `Sweave()`

Reproducible Econometrics Using `Sweave()`

R and reproducible econometrics:

- R is mostly platform independent – runs on Windows, Mac OS, and various flavors of Unix.
- R is open source – inspection of full source code possible.
- R supports literate programming – `Sweave()` allows for mixing R and LaTeX code.

These slides are produced using `Sweave()` functionality. For compiling,

- first the whole source code is executed, its output (text and graphics) is "weaved" with the LaTeX text,
- then pdfLaTeX is run to produce the final slides in PDF (portable document format).

Therefore, it is assured that the input and output displayed are always in sync with the versions of the data, code, packages, and R itself.

Reproducible Econometrics Using `Sweave()`

Example:

- We start out from the file `Sweave-journals.Rnw`.
- Mainly looks like a LaTeX file, but contains R code chunks beginning with `<<...>>=` and ending in `@`.
- File can be processed by R upon calling
  ```
  R> Sweave("Sweave-journals.Rnw")
  ```
  This replaces original R code by valid LaTeX code and weaves it into `Sweave-journals.tex`
- In place of R chunks, new file contains verbatim LaTeX chunks with input and output of R commands and/or an `\includegraphics{}` statement for the inclusion of figures generated along the way.
- `Sweave-journals.tex` can be processed as usual by LaTeX, producing the final document.

Reproducible Econometrics Using `Sweave()`

Remarks:

- `.Rnw` abbreviates "R noweb" – **noweb** is literate-programming tool whose syntax is reused in `Sweave()`.
- Additional environments (`Sinput`, `Soutput`, and `Schunk`, etc.) defined in style file `Sweave.sty` – part of the local R installation and automatically included with system-dependent path.
- In addition to "weaving", there is second basic operation for literate-programming documents, called "tangling". Here amounts to extracting R code included in `.Rnw` file.

```
R> Stangle("Sweave-journals.Rnw")
```

produces file `Sweave-journals.R` containing R code from the two R chunks.

Reproducible Econometrics Using `Sweave()`

Remarks:

- Basic weaving procedure can be refined in many ways. Can insert control options in `<<...>>=` , e.g., `echo=FALSE` (code not displayed), also optional name for the chunk. See `?RweaveLatex` for more details.
- We already used `fig=TRUE` (figures required). By default, both EPS (encapsulated PostScript) and PDF files are generated so that the associated LaTeX sources can be compiled either with plain LaTeX (for DVI documents) or pdfLaTeX (for PDF documents).
- Running LaTeX also possible from within R using `texi2dvi()` from **tools** package.
- Source code for example is also contained in vignette in folder `~/AER/inst/doc` of **AER**. View PDF document by calling

```
R> vignette("Sweave-journals", package = "AER")
```


Reproducible Econometrics Using `Sweave()`

Using `\Sexpr{}`: Often want to avoid verbatim sections in reports or papers and use LaTeX formulas and equations instead.

Example: display regression equation with estimated coefficients.

```
\[
  \log(\mbox{subscriptions}) \quad = \quad
    \Sexpr{round(coef(journals_lm)[1], digits = 2)}
    \Sexpr{if(coef(journals_lm)[2] < 0) "-" else "+"}
    \Sexpr{abs(round(coef(journals_lm)[2], digits = 2))}
    \cdot \log(\mbox{price per citation})
\]
```

Output in processed document is The fitted regression line is

$$\log(\text{subscriptions}) \quad = \quad 4.77 - 0.53 \cdot \log(\text{price per citation})$$

Reproducible Econometrics Using `Sweave()`

Tables: Often simpler to directly use R's text processing functionality and put together the full LaTeX code within R.

Example: table of coefficients for a regression model

Table: Hand-crafted regression summary for `Journals` data.

	Estimate	Std. error	*t* statistic	*p* value
(Intercept)	4.766	0.056	85.249	< 0.001
log(price/citations)	-0.533	0.036	-14.968	< 0.001

Furthermore: Packages like **xtable**, **memisc**, or **texreg** put together "nice" summary tables in LaTeX.