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()`.
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.

Data-generating processes are

- trend: \( y_i = \beta_1 + \beta_2 \cdot i + \varepsilon_i \),
- dynamic: \( y_i = \beta_1 + \beta_2 \cdot y_{i-1} + \varepsilon_i \),

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

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
+ } else {
+ y <- rep(NA, nobs)
+ for(i in 2:nobs)
+ x <- c(0, y[1:(nobs-1)])
+ }
+ return(data.frame(y = y, x = x))
+ }

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
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) + }
```

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

```r
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")
```

<table>
<thead>
<tr>
<th>model</th>
<th>nobs</th>
<th>test</th>
<th>corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>trend</td>
<td>15</td>
<td>dwtest</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>dwtest</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.31</td>
</tr>
<tr>
<td>dynamic</td>
<td>15</td>
<td>dwtest</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>dwtest</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.76</td>
</tr>
</tbody>
</table>
```

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
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

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 $\varrho$ and $n$.
- For dynamic model, Durbin-Watson test has almost no power except for very high correlations. Breusch-Godfrey test performs acceptably.

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

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)
```

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> jour_boot <- boot(journals, refit, R = 999)
R> boot.ci(jour_boot, index = 2, type = "basic")
```

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

---

Christian Kleiber, Achim Zeileis © 2008–2017

Applied Econometrics with R – 7 – Programming – 16 / 35

Christian Kleiber, Achim Zeileis © 2008–2017

Applied Econometrics with R – 7 – Programming – 17 / 35

Christian Kleiber, Achim Zeileis © 2008–2017

Applied Econometrics with R – 7 – Programming – 18 / 35

Christian Kleiber, Achim Zeileis © 2008–2017

Applied Econometrics with R – 7 – Programming – 19 / 35
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`.

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

\[ \log Y_i + \theta Y_i = \beta_1 + \beta_2 \log K_i + \beta_3 \log L_i + \varepsilon_i, \]

→ nonlinear in parameters, only for known \( \theta \) can estimate by OLS.

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 \( \partial \varepsilon_i / \partial Y_i = (1 + \theta Y_i) / Y_i. \)
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 \texttt{optim()} by default performs minimization, we minimize the negative log-likelihood.
- Our function \texttt{nlogL()} is function of vector parameter \texttt{par} comprising five elements.
- R provides functions for the logarithms of standard distributions, including normal density \texttt{dnorm(\ldots, \log = TRUE)}.

Maximizing a Likelihood

Step 1: code log-likelihood

\begin{verbatim}
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)
+   lhs <- log(Y) + theta * Y
+   rval <- sum(log(1 + theta * Y) - log(Y) +
+                 dnorm(lhs, mean = rhs, sd = sqrt(sigma2), log = TRUE))
+   return(-rval)
+ }
\end{verbatim}

Step 2: obtain starting values

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

Thus
\begin{verbatim}
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)))
\end{verbatim}

Step 3: search for the optimum from starting values.

\begin{verbatim}
R> opt <- optim(par0, nlogL, hessian = TRUE)
\end{verbatim}

By default, \texttt{optim()} uses Nelder-Mead method (further algorithms available).

Set \texttt{hessian = TRUE} to obtain standard errors.

Extract estimates, standard errors and value of objective function:
\begin{verbatim}
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
\end{verbatim}

Results suggest that \(\theta\) 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.

Reproducible Econometrics Using \texttt{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 – \texttt{Sweave()} allows for mixing R and \LaTeX{} code.

These slides are produced using \texttt{Sweave()} functionality. For compiling,

- first the whole source code is executed, its output (text and graphics) is “weaved” with the \LaTeX{} text,
- then \texttt{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.

Example:

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

Remarks:

- \texttt{.Rnw} abbreviates “R noweb” – \texttt{noweb} is literate-programming tool whose syntax is reused in \texttt{Sweave()}.  
- Additional environments (\texttt{Sinput}, \texttt{Soutput}, and \texttt{Schunk}, etc.) defined in style file \texttt{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 \texttt{.Rnw} file.

\begin{verbatim}
R> Stangle("Sweave-journals.Rnw")
\end{verbatim}

produces file \texttt{Sweave-journals.R} containing R code from the two R chunks.

Christian Kleiber, Achim Zeileis © 2008–2017

\section*{Reproducible Econometrics Using \texttt{Sweave()}}

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

\textbf{Example}: display regression equation with estimated coefficients.

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

Output in processed document is

\begin{verbatim}
log(subscriptions) = 4.77 - 0.53 \cdot \log(price per citation)
\end{verbatim}

Christian Kleiber, Achim Zeileis © 2008–2017

\section*{Reproducible Econometrics Using \texttt{Sweave()}}

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

\textbf{Example}: table of coefficients for a regression model

\begin{table}
\centering
\begin{tabular}{lrrr}
\hline
 & Estimate & Std. error & \textit{t} statistic & \textit{p} value \\
\hline
(Intercept) & 4.766 & 0.056 & 85.249 & < 0.001 \\
log(price/citations) & \text{-}0.533 & 0.036 & \text{-}14.968 & < 0.001 \\
\hline
\end{tabular}
\caption{Hand-crafted regression summary for Journals data.}
\end{table}

\textbf{Furthermore}: Packages like \texttt{xtable}, \texttt{memisc}, or \texttt{texreg} put together “nice” summary tables in \LaTeX{}. 

Christian Kleiber, Achim Zeileis © 2008–2017