Demand for economics journals

Data set from Stock & Watson (2007), originally collected by T. Bergstrom, on subscriptions to 180 economics journals at US libraries, for the year 2000.

10 variables are provided including:

- `subs` – number of library subscriptions,
- `price` – library subscription price,
- `citations` – total number of citations,

and other information such as number of pages, founding year, characters per page, etc.

**Of interest:** relation between demand and price for economics journals. Price is measured as price per citation.

Demand for economics journals

Load data and obtain basic information:

```r
R> library("AER")
R> data("Journals", package = "AER")
R> dim(Journals)
[1] 180 10
R> names(Journals)
[1] "title"   "publisher"  "society"   "price"
[5] "pages"   "charpp"    "citations" "foundingyear"
[9] "subs"    "field"
```

Plot variables of interest:

```r
R> plot(log(subs) ~ log(price/citations), data = Journals)
```

Fit linear regression model:

```r
R> j_lm <- lm(log(subs) ~ log(price/citations), data = Journals)
R> abline(j_lm)
```
Determinants of wages


Model: wage equation in semi-logarithmic form (with regressors education and quadratic polynomial in experience).

Comparison: OLS and LAD estimator (and further regression quantiles).

In R:
- use `lm()` again for more complex model,
- use `rq()` from `quantreg` for quantile regression (with the same type of interface),
- employ R's graphics capabilities for visualization and graphical comparison.
Determinants of wages

Load data:

```r
R> data("CPS1985", package = "AER")
R> cps <- CPS1985
```

OLS regression:

```r
R> cps_lm <- lm(log(wage) ~ experience + I(experience^2) +
+ education, data = cps)
```

Fitted mean function:

```r
R> cps2 <- data.frame(education = mean(cps$education),
+ experience = min(cps$experience):max(cps$experience))
R> cps2 <- cbind(cps2, predict(cps_lm, newdata = cps2,
+ interval = "prediction"))
```

Visualization:

```r
R> plot(log(wage) ~ experience, data = cps)
R> lines(fit ~ experience, data = cps2, col = 2)
```

Determinants of wages

Quantile regression for $\tau = 0.2, 0.35, 0.5, 0.65, 0.8$:

```r
R> library("quantreg")
R> cps_rq <- rq(log(wage) ~ experience + I(experience^2) +
+ education, data = cps, tau = seq(0.2, 0.8, by = 0.15))
```

Fitted quantile regressions:

```r
R> cps2 <- cbind(cps2,
+ predict(cps_rq, newdata = cps2))
```

Visualization:

```r
R> plot(log(wage) ~ experience, data = cps)
R> for(i in 6:10) lines(cps2[,i] ~ experience,
+ data = cps2, col = 2)
```

Graphical comparison of OLS and regression quantiles:

```r
R> plot(summarize(cps_rq))
```
Determinants of wages

Bivariate kernel density estimate of experience and log(wage):

```r
R> library("KernSmooth")
R> cps_bkde <- bkde2D(cbind(cps$experience, log(cps$wage)),
+                 bandwidth = c(3.5, 0.5), gridsize = c(200, 200))
R> image(cps_bkde$x1, cps_bkde$x2, cps_bkde$fhat,
+        col = rev(gray.colors(10, gamma = 1)),
+        xlab = "experience", ylab = "log(wage)")
R> box()
R> lines(fit ~ experience, data = cps2)
R> lines(lwr ~ experience, data = cps2, lty = 2)
R> lines(upr ~ experience, data = cps2, lty = 2)
```

Visualize with fitted OLS regression and confidence bounds:
R system for statistical computing and graphics

- R project homepage: https://www.R-project.org/,
- open-source software project,
- released under the GNU General Public License (GPL),
- full sources available online from Comprehensive R Archive
  Network (CRAN),
- binary versions for Microsoft Windows, various flavours of Linux
  (including Debian, Red Hat, SUSE, and Ubuntu), and for
  MacOS X,
- CRAN has a world-wide network of mirrors, see:
  https://CRAN.R-project.org/mirrors.html.

Installation

Installation of binary versions is straightforward:
- go to CRAN, pick up the version for your operating system, follow
  instructions in readme file,
- Microsoft Windows: download and run setup .exe file,
- Mac OS X: Installer package .pkg for base system and
  platform-specific GUI, along with additional programming tools (as
  disk image .dmg files),
- Linux: pre-packaged binaries for various flavors (.deb or .rpm
  files), also interfaced in various update managers (apt, yum, etc.).

Installation

Installation from source:
- possible on numerous (and also exotic) platforms,
- easy when compilers ship with the operating system (e.g.,
  Unix/Linux) in the usual configure/make/install steps,
- compilers are also available for Windows but require some more
  installation/configuration.

Manual: R Installation and Administration.

Packages

R is highly extensible by means of packages:
- packages can contain R code, source code (e.g., C, Fortran), data,
  manual pages, further documentation, examples, demos, ...
- package can depend on other packages (that need to be available
  for using the package),
- “base” packages: contained in the R sources,
- “recommended” packages: included in every binary distribution,
- “contributed” packages: available from the CRAN servers
  (currently more than 10,000) at
  https://CRAN.R-project.org/web/packages/.
Packages

Installing and loading packages:
- if connected to the internet, simply type
  \texttt{install.packages("AER")}
- additionally on Windows and Mac: GUI installer menus,
- packages are installed in \textit{libraries} (= collections of packages),
- library paths can be specified (see \texttt{?library}),
- packages are loaded by the command \texttt{library()}, e.g.,
  \texttt{library("AER")},
- \texttt{library()} lists all currently installed packages.

\textbf{CRAN task views:} provide overview of packages for certain tasks
(e.g., econometrics, finance, social sciences, Bayesian statistics, ...).
https://CRAN.R-project.org/web/views/

User interfaces and development environments

\textbf{Base R:} Command line interface (CLI), possibly enhanced by some
limited graphical user interface (GUI) capabilities on Windows and Mac.

Additionally:
- Various integrated development environments (IDEs).
- Various GUIs interfacing certain statistical functionality.
- See https://www.R-project.org/GUI/ for an overview.

\textbf{Popular choices:}
- IDE: RStudio is freely available, open source, and relatively easy
to use. See https://www.RStudio.com/products/RStudio/.
- Basic-statistics GUI: R Commander is an R package providing an
extensible GUI intended primarily for introductory statistics. See
https://CRAN.R-project.org/package=Rcmdr.

Philosophy

\textbf{In most other econometrics packages:} an analysis leads to a large
amount of output containing information on estimation, model
diagnostics, specification tests etc.

\textbf{In R:}
- analysis is broken down into a series of steps,
- intermediate results are stored in \textit{objects},
- minimal output at each step (often none),
- objects can be manipulated and interrogated to obtain the
  information required (e.g., \texttt{print()}, \texttt{summary()}, \texttt{plot()}).

\textbf{Fundamental design principle:} “Everything is an object.”

\textbf{Examples:} vectors and matrices are objects, but also functions and
even function calls ⇒ facilitates programming tasks.
Handling objects

List all objects in the global environment (i.e., the user's workspace):

R> objects()

[1] "CPS1985" "Journals" "cps" "cps2" "cps_bkde"
[6] "cps_lm" "cps_rq" "i" "j_lm"

More objects are available in the attached packages.

R> search()

[1] ".GlobalEnv" "package:KernSmooth"
[5] "package:AER" "package:survival"
[7] "package:sandwich" "package:lmtest"
[9] "package:zoo" "package:car"
[17] "Autoloads" "package:base"

Handling objects

Creating objects:

R> x <- 2
R> x

[1] 2
R> objects()

[1] "CPS1985" "Journals" "cps" "cps2" "cps_bkde"
[6] "cps_lm" "cps_rq" "i" "j_lm"

Removing objects with remove() or rm():

R> remove(x)
R> objects()

[1] "CPS1985" "Journals" "cps" "cps2" "cps_bkde"
[6] "cps_lm" "cps_rq" "i" "j_lm"

Handling objects

The global environment ".GlobalEnv" is always at the first position.

Several attached packages including the **base** package at its end.

R> objects("package:base")

shows the names of more than thousand objects defined in **base** (including the function objects()).

Objects can easily be created by assigning a value to a name, using the assignment operator `<-`.

Calling functions

For a function, `foo()` say:

- Typing an objects name at the prompt, `foo`, prints the object.
- For a function this prints the source code.
- If it is called with parentheses, `foo()`, it is a function call.
- If there are no arguments or all have defaults, `foo()` is a valid function call.
- A function call may use the arguments in any order, provided the name of the argument is given.
- If names of arguments are not given, R assumes they appear in the order of the function definition.
- If an argument has a default, it may be left out in a function call.


### Calling functions

**Example:** The function \( \log() \) has two arguments, \( x \) (a numeric scalar or vector), and \( \text{base} \) (the base with respect to which logarithms are computed).

\[
R> \log(x = 16, \text{base} = 2)
\]

\[
[1] 4
\]

The following calls all yield equivalent output:

\[
R> \log(16, 2)
\]
\[
R> \log(x = 16, 2)
\]
\[
R> \log(16, \text{base} = 2)
\]
\[
R> \log(\text{base} = 2, x = 16)
\]

### Classes and generic functions

Every object has a *class* that can be queried using \( \text{class()} \).

For each class, certain methods to *generic* functions can be available, e.g., \( \text{summary()} \) and \( \text{plot()} \).

**Examples:**

- "\text{data.frame}": a list with a certain structure (preferred format for holding data),
- "\text{lm}": linear-model objects (returned by \( \text{lm()} \)).

### Quitting R

One exits R by using the \( \text{q()} \) function:

\[
R> \text{q()}
\]

R asks whether to save the workspace:

- \( n \) (no): exit R without saving anything,
- \( y \) (yes): save all currently defined objects in .RData and the command history in .Rhistory, both in the working directory.
File management

Working directory:

- query with `getwd()`
- change with `setwd()`
- if available, `.RData` and/or `.Rhistory` are loaded upon startup,
- `dir()` lists available files.

More generally:

- directories can be listed with `dir()`
- saved workspaces can be loaded using `load()`
- R objects can be saved (in binary format) by `save()`.

Getting Help

Help pages

**Documentation:** The help page for any function or data set can be accessed using either `?` or `help()`:

```r
R> ?options
R> help("options")
```

**Examples:** At the bottom of a help page, there are typically practical examples of how to use that function. These can easily be executed:

```r
R> example("options")
R> example("lm")
```

Searching for help

If the exact name of a command is not known, the functions to use are `help.search()` and `apropos()`.

`help.search()` returns help files with aliases or concepts or titles matching a “pattern” using fuzzy matching. For example, searching for the pattern “option” will yield a (long) list of help pages, including the function `options()` used above.

```r
R> help.search("option")
options(base) Options Settings
```

`apropos()` lists all functions whose names include the pattern entered. As an illustration,

```r
R> apropos("help")
[1] "help" "help.request" "help.search" "help.start"
```
Vignettes

**More advanced:** Vignettes are PDF files generated from integrated files containing both R code and documentation in \textit{\LaTeX} format ⇒ all commands can be extracted and executed, reproducing the analysis.

Typically less technical information and written more in the style of tutorials.

For an example, see
\begin{verbatim}
R> vignette("strucchange-intro", package = "strucchange")
\end{verbatim}

These slides and accompanying R scripts are actually written using the same tools.

Demos

A demo is an interface to run some demonstration R scripts. Type
\begin{verbatim}
R> demo()
\end{verbatim}
for a list of available topics.

**Examples:** "graphics", "lm.glm".

For beginners, running
\begin{verbatim}
R> demo("graphics")
\end{verbatim}
is recommended.

Manuals

R also comes with a number of manuals:

- An Introduction to R
- R Data Import/Export
- R Language Definition
- Writing R Extensions
- R Installation and Administration
- R Internals

FAQs

CRAN hosts several collections of frequently asked questions (FAQs).

[https://CRAN.R-project.org/faqs.html](https://CRAN.R-project.org/faqs.html)

**R FAQ:** useful information for all platforms (Linux, Mac, Unix, Windows).

**R Mac OS X FAQ:** additional Mac-specific information.
[https://CRAN.R-project.org/bin/macosx/RMacOSX-FAQ.html](https://CRAN.R-project.org/bin/macosx/RMacOSX-FAQ.html)

**R Windows FAQ:** additional Windows-specific information.
[https://CRAN.R-project.org/bin/windows/base/rw-FAQ.html](https://CRAN.R-project.org/bin/windows/base/rw-FAQ.html)
The R Journal: online journal launched in 2009, following up on the R News newsletter launched in 2001, published about two times per year. Features include recent developments in R, a “programmer’s niche”, and examples analyzing data with R. 
https://journal.R-project.org/

Journal of Statistical Software: open-access journal that publishes articles and code snippets (as well as book and software reviews) on the subject of statistical software and algorithms. It has a growing number of publications on R packages, a special volume on Econometrics in R was published in Volume 27 (2008). 
https://www.jstatsoft.org/

Books: rapidly growing list of books on R or on statistics using R. Prominent examples include


Development model

As R is an open-source project, its development model is quite different from many other econometrics software packages.

Extensibility: a key feature in R’s success is the extensibility through packages. These can contain everything that the base system contains:

- R code (obviously),
- code in compiled languages (such as C, C++, or Fortran),
- data sets, demo files, test suites, vignettes, or further documentation.

Every R user can easily become an R developer by submitting his or her packages to CRAN.
Development model

**Base system:** Unlike the CRAN packages, base R is maintained by the R core team:
- major releases (i.e., versions x.y.0) annually,
- free read access to the development version in the SVN repository.

**Version control:** SVN stands for Subversion, see https://subversion.apache.org/

Mailing lists

For communication between R users and developers, two means are particularly useful: CRAN packages (see above) and various mailing lists.

**R-help:** asking for help on using R.

**R-devel:** discussing issues related to the development of R.

Furthermore, bugs can be reported and feature requests made. The posting guide discusses some good strategies for doing this effectively. https://www.R-project.org/posting-guide.html

**Special interest groups:** SIGs are mailing lists for special topics, including a list devoted to finance and (financial) econometrics:
R-SIG-Finance.

History of S

1976 John Chambers and co-workers at Bell Labs begin work on a project that will become S (S1).
1981 Licenses for a new portable Unix version of S outside Bell Labs (S2, brown and blue book).
1988 Statistical software package S-PLUS based on S.
1992 Object orientation and statistical modeling toolbox included (S3, white book).
1993 Exclusively licensed to MathSoft (now Insightful).
2004 S implementation sold to Insightful.

A Brief History of R
History of R

1991  Ross Ihaka and Robert Gentleman begin work on a project that will ultimately become R.
1993  First binary copies of R on Statlib.
1995  R release of sources under the GPL.
1997  R development core team is formed.
1998  Comprehensive R Archive Network (CRAN).
1999  First DSC meeting in Vienna, first R core meeting.
2000  R 1.0.0 is released.
2001  R News launched.
2002  R Foundation established.
2004  First useR! conference in Vienna.
2004  R 2.0.0 is released.
2007  R-Forge server launched.
2013  R 3.0.0 is released.

R in econometrics

- Racine and Hyndman (2002), “Using R to Teach Econometrics”, *Journal of Applied Econometrics*, 17, 175–189. (Uses R 1.3.1.)
- Kleiber and Zeileis (2008), *Applied Econometrics with R*, Springer-Verlag, New York. (Uses R 2.7.0.)