

Introduction to R

John Fox

McMaster University

ICPSR 2010

Outline

- Getting Started with R
- Statistical Models in R
- Data in R
- R Programming
- R Graphics
- Building R packages (or another topic)

Getting Started With R

What is R?

- A statistical programming language and computing environment, implementing the S language.
- Two implementations of S:
 - S-PLUS: commercial, for Windows and (some) Unix/Linux, eclipsed by R.
 - R: free, open-source, for Windows, Macintoshes, and (most) Unix/Linux.

Getting Started With R

What is R?

- How does a statistical programming environment differ from a statistical package (such as SPSS)?
 - A package is oriented toward combining instructions and rectangular datasets to produce (voluminous) printouts and graphs. Routine, standard data analysis is easy; innovation or nonstandard analysis is hard or impossible.
 - A programming environment is oriented toward transforming one data structure into another. Programming environments such as R are *extensible*. Standard data analysis is easy, but so are innovation and nonstandard analysis.

Getting Started With R

Why Use R?

- Among statisticians, R has become the de-facto standard language for creating statistical software. Consequently, new statistical methods are often first implemented in R.
- There is a great deal of built-in statistical functionality in R, and many (literally thousands of) add-on packages available that extend the basic functionality.
- R creates fine statistical graphs with relatively little effort.
- The R language is very well designed and finely tuned for writing statistical applications.
- (Much) R software is of very high quality.
- R is easy to use (for a programming language).
- R is free (in both of senses: costless and distributed under the Free Software Foundation's GPL).

Getting Started With R

This Workshop

- The purpose of this lecture series/workshop is to get participants started using R.
- The statistical content is largely assumed known.
- Much of the workshop is based on J. Fox and S. Weisberg, *An R Companion to Applied Regression, Second Edition*, Sage (in press).
- More advanced participants may prefer to read, or want to read in addition, W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S, Fourth Edition*. New York: Springer, 2002
- Additional materials and links are available on the web site for the first edition of the book:
<http://socserv.socsci.mcmaster.ca/jfox/Books/Companion/index.html>
- The book is associated with an R package (called **car**) that implements a variety of methods helpful for analyzing data with linear and generalized linear models.

Getting Started With R

This Workshop

- Other references are given on the workshop web site.
- Workshop web site:
<http://socserv.socsci.mcmaster.ca/jfox/Courses/R-course/index.html>

Statistical Models in R

Topics

- Multiple linear regression
- Factors and dummy regression models
- Overview of the `lm` function
- The structure of generalized linear models (GLMs) in R; the `glm` function
- GLMs for binary/binomial data
- GLMs for count data
- Traditional ANOVA and MANOVA for repeated-measures designs (time permitting)

Statistical Models in R

Arguments of the `lm` function

- `lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)`
- `formula`

<i>Expression</i>	<i>Interpretation</i>	<i>Example</i>
<code>A + B</code>	include both A and B	<code>income + education</code>
<code>A - B</code>	exclude B from A	<code>a*b*d - a:b:d</code>
<code>A:B</code>	all interactions of A and B	<code>type:education</code>
<code>A*B</code>	<code>A + B + A:B</code>	<code>type*education</code>
<code>B %in% A</code>	B nested within A	<code>education %in% type</code>
<code>A/B</code>	<code>A + B %in% A</code>	<code>type/education</code>
<code>A^k</code>	effects crossed to order k	<code>(a + b + d)^2</code>

Statistical Models in R

Arguments of the `lm` function

- `data`: A data frame containing the data for the model.
- `subset`:
 - a logical vector: `subset = sex == "F"`
 - a numeric vector of observation indices: `subset = 1:100`
 - a negative numeric vector with observations to be omitted: `subset = -c(6, 16)`
- `weights`: for weighted-least-squares regression
- `na.action`: name of a function to handle missing data; default given by the `na.action` option, initially `"na.omit"`
- `method`, `model`, `x`, `y`, `qr`, `singular.ok`: technical arguments
- `contrasts`: specify list of contrasts for factors; e.g.,
`contrasts=list(partner.status=contr.sum, fcategory=contr.poly)`
- `offset`: term added to the right-hand-side of the model with a fixed coefficient of 1.

Statistical Models in R

Review of the Structure of GLMs

- A generalized linear model consists of three components:
 - 1 A *random component*, specifying the conditional distribution of the response variable, y_i , given the predictors. Traditionally, the random component is an exponential family — the normal (Gaussian), binomial, Poisson, gamma, or inverse-Gaussian.
 - 2 A linear function of the regressors, called the *linear predictor*,

$$\eta_i = \alpha + \beta_1 x_{i1} + \dots + \beta_k x_{ik}$$

on which the expected value μ_i of y_i depends.

- 3 A *link function* $g(\mu_i) = \eta_i$, which transforms the expectation of the response to the linear predictor. The inverse of the link function is called the *mean function*: $g^{-1}(\eta_i) = \mu_i$.



Statistical Models in R

Review of the Structure of GLMs

- In the following table, the logit, probit and complementary log-log links are for binomial or binary data:

Link	$\eta_i = g(\mu_i)$	$\mu_i = g^{-1}(\eta_i)$
identity	μ_i	η_i
log	$\log_e \mu_i$	e^{η_i}
inverse	μ_i^{-1}	η_i^{-1}
inverse-square	μ_i^{-2}	$\eta_i^{-1/2}$
square-root	$\sqrt{\mu_i}$	η_i^2
logit	$\log_e \frac{\mu_i}{1 - \mu_i}$	$\frac{1}{1 + e^{-\eta_i}}$
probit	$\Phi(\mu_i)$	$\Phi^{-1}(\eta_i)$
complementary log-log	$\log_e[-\log_e(1 - \mu_i)]$	$1 - \exp[-\exp(\eta_i)]$



- Generalized linear models are fit with the `glm` function. Most of the arguments of `glm` are similar to those of `lm`:
 - The response variable and regressors are given in a model formula.
 - `data`, `subset`, and `na.action` arguments determine the data on which the model is fit.
 - The additional `family` argument is used to specify a *family-generator function*, which may take other arguments, such as a link function.

- The following table gives family generators and default links:

<i>Family</i>	<i>Default Link</i>	<i>Range of y_i</i>	$V(y_i \eta_i)$
gaussian	identity	$(-\infty, +\infty)$	ϕ
binomial	logit	$\frac{0, 1, \dots, n_i}{n_i}$	$\mu_i(1 - \mu_i)$
poisson	log	$0, 1, 2, \dots$	μ_i
Gamma	inverse	$(0, \infty)$	$\phi\mu_i^2$
inverse.gaussian	$1/\mu^2$	$(0, \infty)$	$\phi\mu_i^3$

- For distributions in the exponential families, the variance is a function of the mean and a dispersion parameter ϕ (fixed to 1 for the binomial and Poisson distributions).

Statistical Models in R

Implementation of GLMs in R

- The following table shows the links available for each family in R, with the default links as ■:

family	link			
	identity	inverse	sqrt	1/mu ²
gaussian	■	<input type="checkbox"/>		
binomial				
poisson	<input type="checkbox"/>		<input type="checkbox"/>	
Gamma	<input type="checkbox"/>	■		
inverse.gaussian	<input type="checkbox"/>	<input type="checkbox"/>		■
quasi	■	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
quasibinomial				
quasipoisson	<input type="checkbox"/>		<input type="checkbox"/>	



Statistical Models in R

Implementation of GLMs in R

family	link			
	log	logit	probit	cloglog
gaussian	<input type="checkbox"/>			
binomial	<input type="checkbox"/>	■	<input type="checkbox"/>	<input type="checkbox"/>
poisson	<input type="checkbox"/>			
Gamma	<input type="checkbox"/>			
inverse.gaussian	<input type="checkbox"/>			
quasi	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
quasibinomial		■	<input type="checkbox"/>	<input type="checkbox"/>
quasipoisson	■			

- The quasi, quasibinomial, and quasipoisson family generators do not correspond to exponential families.



Statistical Models in R

GLMs for Binary/Binomial and Count Data

- The response for a binomial GLM may be specified in several forms:
 - For binary data, the response may be
 - a variable or an S expression that evaluates to 0's ('failure') and 1's ('success').
 - a logical variable or expression (with TRUE representing success, and FALSE failure).
 - a factor (in which case the first category is taken to represent failure and the others success).
 - For binomial data, the response may be
 - a two-column matrix, with the first column giving the count of successes and the second the count of failures for each binomial observation.
 - a vector giving the *proportion* of successes, while the binomial denominators (total counts or numbers of trials) are given by the `weights` argument to `glm`.

Statistical Models in R

GLMs for Binary/Binomial and Count Data

- Poisson generalized linear models are commonly used when the response variable is a count (Poisson regression) and for modeling associations in contingency tables (loglinear models).
 - The two applications are formally equivalent. Poisson GLMs are fit in S using the `poisson` family generator with `glm`.
- Overdispersed binomial and Poisson models may be fit via the `quasibinomial` and `quasipoisson` families.

- Function definition
- Control structures:
 - Conditionals: if, ifelse, switch
 - Iteration: for, while, repeat
- Recursion

Programming Basics

Review of MLE of the Binary Logit Model: Estimation by Newton-Raphson

- 1 Choose initial estimates of the regression coefficients, such as $\mathbf{b}_0 = \mathbf{0}$.
- 2 At each iteration t , update the coefficients:

$$\mathbf{b}_t = \mathbf{b}_{t-1} + (\mathbf{X}'\mathbf{V}_{t-1}\mathbf{X})^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{p}_{t-1})$$

where

- \mathbf{X} is the model matrix, with \mathbf{x}'_i as its i th row;
- \mathbf{y} is the response vector (containing 0's and 1's);
- \mathbf{p}_{t-1} is the vector of fitted response probabilities from the previous iteration, the i th entry of which is

$$p_{i,t-1} = \frac{1}{1 + \exp(-\mathbf{x}'_i\mathbf{b}_{t-1})}$$

- \mathbf{V}_{t-1} is a diagonal matrix, with diagonal entries $p_{i,t-1}(1 - p_{i,t-1})$.
- 3 Step 2 is repeated until \mathbf{b}_t is close enough to \mathbf{b}_{t-1} . The estimated asymptotic covariance matrix of the coefficients is given by $(\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}$.

Programming Basics

Review of MLE of the Binary Logit Model: Estimation by General Optimization

- Another approach is to let a general-purpose optimizer do the work of maximizing the log-likelihood,

$$\log_e L = \sum y_i \log_e p_i + (1 - y_i) \log_e (1 - p_i)$$

- Optimizers work by evaluating the *gradient* (vector of partial derivatives) of the 'objective function' (the log-likelihood) at the current estimates of the parameters, iteratively improving the parameter estimates using the information in the gradient; iteration ceases when the gradient is sufficiently close to zero.
- For the logistic-regression model, the gradient of the log-likelihood is

$$\frac{\partial \log_e L}{\partial \mathbf{b}} = \sum (y_i - p_i) \mathbf{x}_i$$

Programming Basics

Review of MLE of the Binary Logit Model: Estimation by General Optimization

- The covariance matrix of the coefficients is the inverse of the matrix of second derivatives. The matrix of second derivatives, called the *Hessian*, is

$$\frac{\partial \log_e L}{\partial \mathbf{b} \partial \mathbf{b}'} = \mathbf{X}' \mathbf{V} \mathbf{X}$$

- The `optim` function in R, however, calculates the Hessian numerically (rather than using an analytic formula).

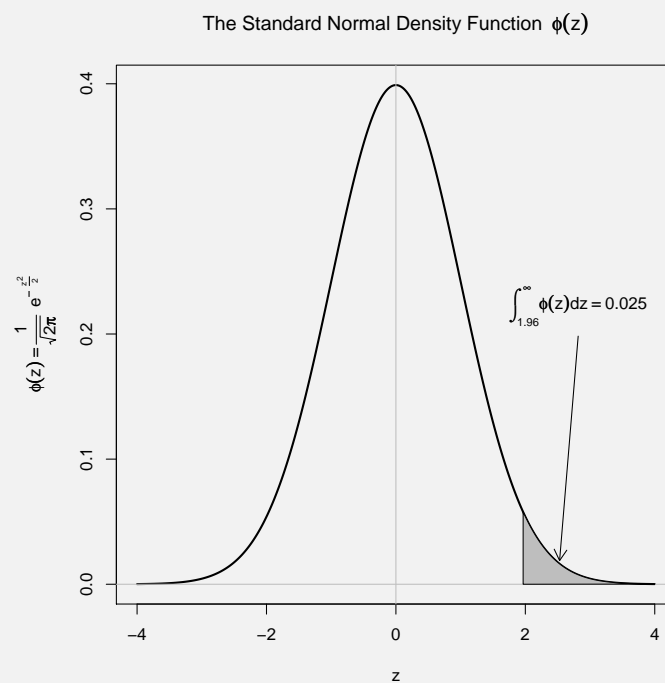
- Locating an error: `traceback()`
- Setting a breakpoint and examining the local environment of an executing function: `browser()`
- A simple interactive debugger: `debug()`
- Some other facilities: **debug** package, `debugger()` post-mortem debugger
- Measuring time and memory usage with `system.time` and `Rprof`

Object-Oriented Programming

The S3 Object System

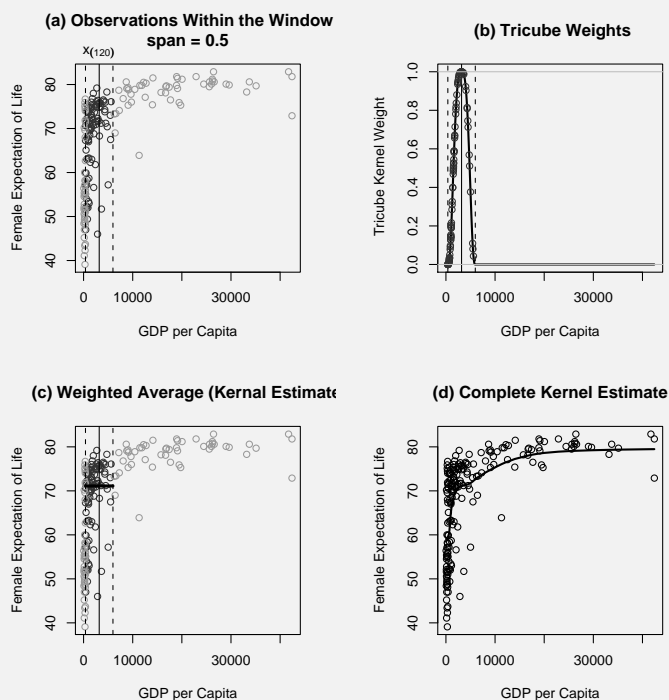
- S3 versus S4 objects
- How the S3 object system works
- Method dispatch, for *object* of class "*class*": `generic(object)`
⇒ `generic.class(object)` ⇒ `generic.default(object)`
 - For example, summarizing an object `mod` of class "`lm`": `summary(mod)`
⇒ `summary.lm(mod)`
- Objects can have more than one class, in which case the first applicable method is used.
 - For example, objects produced by `glm()` are of class `c("glm", "lm")` and therefore can *inherit* methods from class "`lm`".
- Generic functions: `generic <- function(object, other-arguments, ...) UseMethod("generic")`
 - For example, `summary <- function(object, ...) UseMethod("summary")`

- Traditional S graphics
 - points
 - lines
 - text
 - axes
 - frames
 - arrows
 - polygons
 - legends
 - curves
 - use of colour
- Trellis graphics (via the `lattice` package)



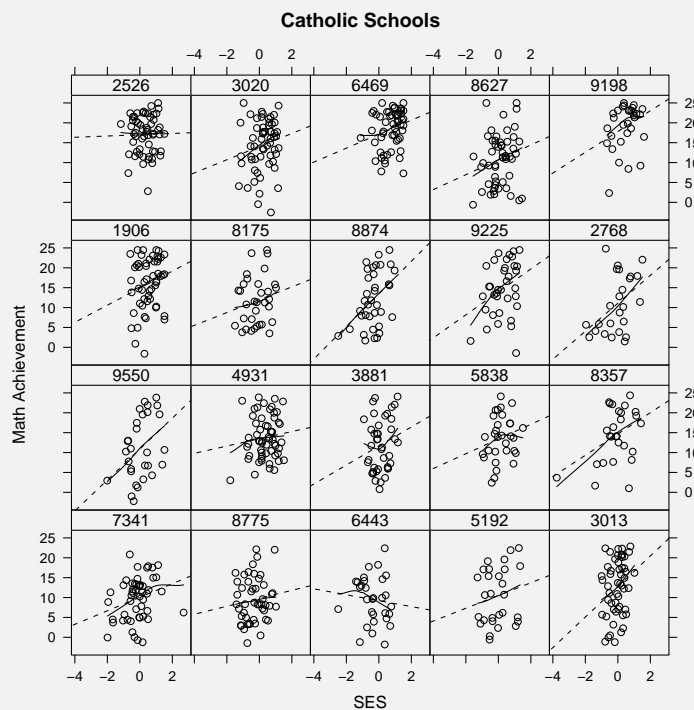
R Graphics

Example: Explanation of Nearest-Neighbor Kernel Regression



R Graphics

Example: Trellis Display



Building R Packages

- Primary reference for creating R Packages: *Writing R Extensions* (distributed with R).
- Create a *source package* containing “meta-information” describing the package; R code; data; documentation; and possibly other components.
- The source package is then checked and built into a universal `.tar.gz` format that can be distributed and installed under any operating system on which R runs, or built into a *binary package* for a particular system, such as Windows or Mac OS X.
- Under Windows, the ability to build packages requires installing software in addition to R: a working LaTeX, Perl, Unix-like command-line tools, and a C++ compiler, all of which are readily available. See the “Windows toolset” appendix in the *R Installation and Administration* manual and <http://www.murdoch-sutherland.com/Rtools/>.

Building R Packages

- To check a package, assuming that the R bin directory is on the Windows path, and that the package subdirectory is in the current directory:
R CMD check *package-name*
- To build the package, producing a `tar.gz` file:
R CMD build --force *package-name*
- To build a Windows binary package, producing a `.zip` file:
R CMD build --binary *package-name*
- To install directly:
R CMD INSTALL *package-name*

Building R Packages

Package Structure

- An R source package consists of a directory containing:
 - A DESCRIPTION file with meta information such as the package name, version, and author, and other packages on which the package depends.
 - An optional NAMESPACE file (for a package with a namespace), enumerating, e.g., the objects that are “exported” from the package.
 - An R subdirectory containing .R files with R code for creating objects such as functions.
 - A man (“manual”) subdirectory that includes .Rd documentation files (using LaTeX-like markup). All public objects in the package should be documented.
 - A data subdirectory containing data objects, such as text files that can be read as R data frames. Thus, a file named Duncan.txt would produce a data frame named Duncan.
 - Possibly an inst (“install”) subdirectory containing arbitrary files and subdirectories to be installed in the package.
 - Possibly other subdirectories (e.g., for compiled C code).

Building R Packages

- The function package.skeleton creates the “skeleton” of a source package for objects that are currently in the R workspace.
- Example: The **matrixDemos** package.