

# R: An Overview and Some Current Directions

Luke Tierney

Department of Statistics & Actuarial Science  
University of Iowa

Korean Statistical Society  
November, 2006



# What Is R?

- R is a language for statistical computing and graphics.
- A member of the S language family.
  - S was developed at Bell Labs (John Chambers et al.).
  - S-plus is the other major family member.
  - ACM Software System Award, 1998.
  - De facto standard for computing in Statistical research.
  - Documented in many books, e.g. Venables and Ripley.
- Can view R as a different implementation or dialect of S.
- There are some important differences, but much code written for S-plus runs unaltered under R.
- R is a major framework for making available new statistical methodology.



# History and Development Model

- R is an Open Source project.
- Originally developed by Robert Gentleman and Ross Ihaka in the early 1990's for a Macintosh computer lab at U. of Auckland, NZ.
- Developed by the R-core group since mid 1997,

Douglas Bates

John Chambers

Peter Dalgaard

Robert Gentleman

Kurt Hornik

Stefano Iacus

Ross Ihaka

Friedrich Leisch

Thomas Lumley

Martin Maechler

Duncan Murdoch

Paul Murrell

Martyn Plummer

Brian Ripley

Duncan Temple Lang

Luke Tierney

Simon Urbanek



# Basic Design of R

- Basic philosophy: Good statistical analysis involves
  - exploring the data
  - allowing the data to guide the choice of analysis tools
  - adapting tools as needed for the appropriate analysis
- R is an interactive system
  - contrasts to batch-oriented systems (e.g. SAS)
- R is a high level language
  - contrasts to pure GUI systems (e.g. JMP)
  - allows analysis to be documented, repeated
  - allows new methods to be programmed



# Extending the R System

- Writing simple R functions is a natural part of working in R.
- Collections of functions that implement a particular analysis are often best organized into a *package*.
- The R package system provides a framework for developing, documenting, and testing extension code.
- Packages can include R code as well as foreign code (C, FORTRAN).
- Many R packages are made available through the CRAN repository <http://cran.r-project.org> (recent count: 840).



# Some Basics

- R uses a command line interface: *read-evaluate-print* loop
  - you type an expression
  - R reads the expression, evaluates it, and prints the result
- Some simple examples:

```
> 2 + 3
```

```
[1] 5
```

```
> exp(-2)
```

```
[1] 0.1353353
```

```
> log(100, base = 10)
```

```
[1] 2
```



# Variables and Vectors

- A variable `x` containing some uniform random numbers:

```
> x <- runif(4)
> x
```

```
[1] 0.1137034 0.6222994 0.6092747 0.6233794
```

- Some vectorized operations:

```
> x + 1
```

```
[1] 1.113703 1.622299 1.609275 1.623379
```

```
> log(x)
```

```
[1] -2.1741619 -0.4743339 -0.4954860 -0.4725999
```



# Numerical Summaries

- mean and standard deviation:

```
> mean(x)
```

```
[1] 0.4921642
```

```
> sd(x)
```

```
[1] 0.2523886
```

- median and inter-quartile range:

```
> median(x)
```

```
[1] 0.6157871
```

```
> IQR(x)
```

```
[1] 0.1371875
```

- sorting and ranking:

```
> sort(x)
```

```
[1] 0.1137034 0.6092747 0.6222994 0.6233794
```

```
> rank(x)
```

```
[1] 1 3 2 4
```



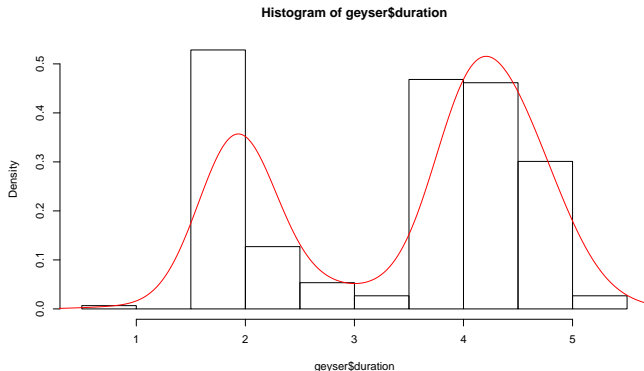


# Simple Graphics

## One Variable

Durations of eruptions of the Old Faithful geyser:

```
> library(MASS)
> hist(geyser$duration, prob = TRUE)
> lines(density(geyser$duration), col = "red")
```

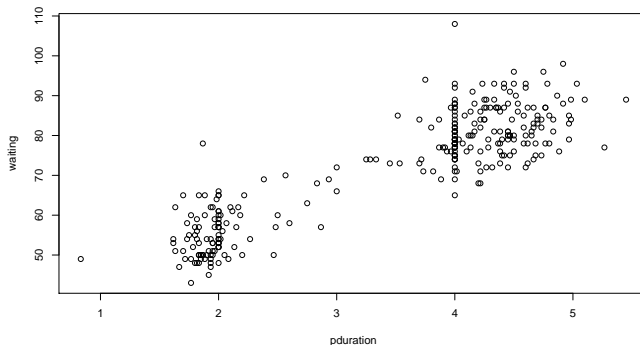


# Simple Graphics

## Two Variables

Duration and Waiting time until the next eruption:

```
> geyser2 <- geyser[-1, ]  
> geyser2$pduration <- geyser$duration[-299]  
> plot(waiting ~ pduration, data = geyser2)
```

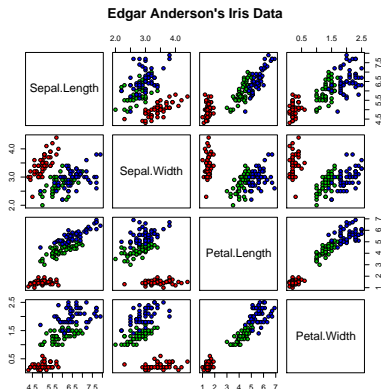


# More Complex Graphics

## Scatterplot Matrices

Measurements of 50 flowers for each of 3 species of iris:

```
> pairs(iris[1:4], main = "Edgar Anderson's Iris Data", pch = 21,  
+       bg = c("red", "green3", "blue")[unclass(iris$Species)])
```

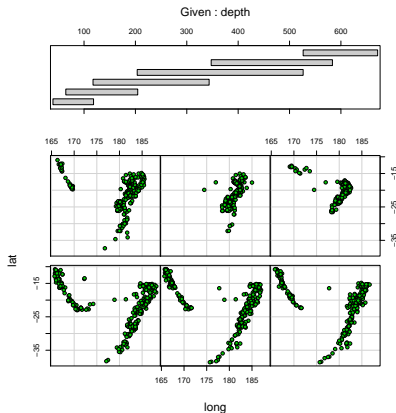


# More Complex Graphics

## Conditioning Plots

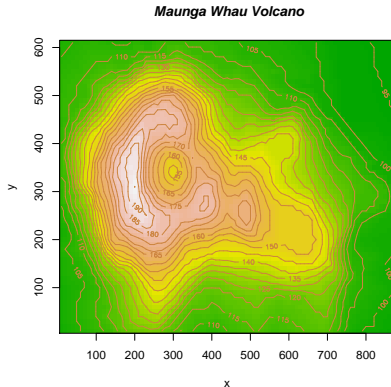
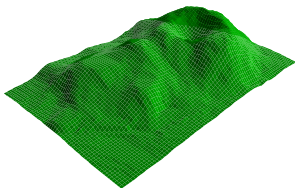
Locations of 1000 seismic events near Fiji:

```
> coplot(lat ~ long | depth, data = quakes, pch = 21, bg = "green3")
```



# More Complex Graphics

## Perspective, Image, and Contour Plots



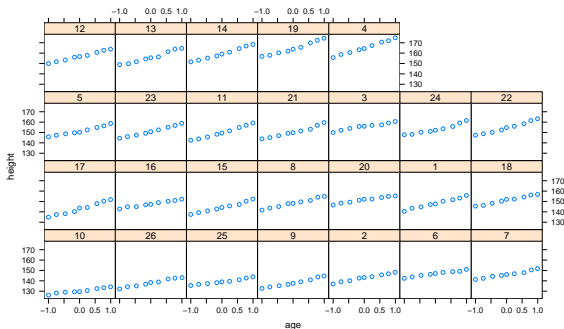
R Graph Gallery <http://addictedtor.free.fr/graphiques/> and `demo(graphics)` give more examples

# Lattice Graphics

## A Conditioning Plot

Height (cm) and standardized ages for a sample of 26 boys from Oxford, England.

```
> library(nlme)
> library(lattice)
> xyplot(height ~ age | Subject, data = Oxboys)
```

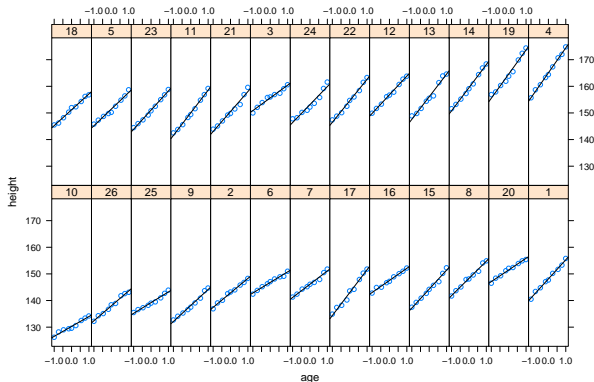


# Lattice Graphics

## A Conditioning Plot

Same plot with a different aspect ratio and regression lines:

```
> xyplot(height ~ age | Subject, data = Oxboys,  
+         aspect = "xy", type = c("p", "r"), col.line = "black")
```



# Statistical Models in R

- Basic R supports fitting a range of models.
- Many more are supported in contributed packages.
- Some of the models available include
  - linear, nonlinear, and generalized linear models
  - mixed models
  - survival models
  - and many others
- Most modeling functions specify models with a *formula*
  - Linear: `height ~ weight`
  - Nonlinear: `Weight ~ b0 + b1 * 2 ^ (-Days/th)`
  - Survival: `Surv(time, status) ~ dose + diet`
- Formulas are also used in specifying some plots.





# Models in R

## Linear Models

A simpler linear model for waiting time as a function of previous eruption duration for the Old Faithful geyser:

```
> summary(lm(waiting ~ pduration, data = geyser2))
```

Call:

```
lm(formula = waiting ~ pduration, data = geyser2)
```

Residuals:

Min	1Q	Median	3Q	Max
-14.6940	-4.4954	-0.0966	3.9544	29.9544

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	34.9452	1.1807	29.60	<2e-16 ***
pduration	10.7751	0.3235	33.31	<2e-16 ***

---

Signif. codes: 0 \*\*\* 0.001 \*\* 0.01 \* 0.05 . 0.1 1

Residual standard error: 6.392 on 296 degrees of freedom

Multiple R-Squared: 0.7894, Adjusted R-squared: 0.7887

F-statistic: 1110 on 1 and 296 DF, p-value: < 2.2e-16



# Models in R

## Linear Mixed Models

Linear mixed effects model with random slope and intercept for Oxford boys growth data:

```
> lme(height ~ age, data = Oxboys, random = ~age | Subject)
```

Linear mixed-effects model fit by REML

Data: Oxboys

Log-restricted-likelihood: -362.0455

Fixed: height ~ age

(Intercept)	age
149.371753	6.525469

Random effects:

Formula: ~age | Subject

Structure: General positive-definite, Log-Cholesky parametrization

StdDev	Corr
--------	------

(Intercept)	8.081077	(Intr)
-------------	----------	--------

age	1.680717	0.641
-----	----------	-------

Residual	0.659889
----------	----------

Number of Observations: 234

Number of Groups: 26



# Simulations in R

- Simulations are useful for
  - exploring and understanding problems
  - approximating otherwise intractable results
- R provides facilities for
  - simulating from many univariate distributions
  - simulating from some multivariate distributions
  - sampling with and without replacement
  - controlling the random number generator

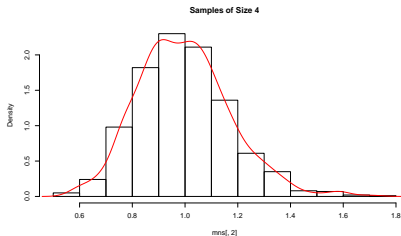
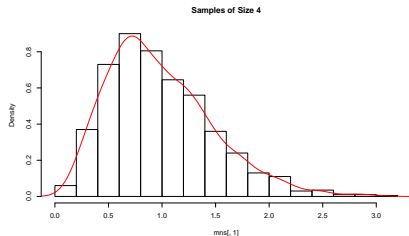


# Simulations in R

## Illustrating the Central Limit Theorem

A standard illustration of the CLT using exponential samples of size 4 and 32:

```
> rmat <- matrix(rexp(1000 * 32), nrow = 32)
> mns <- cbind(colMeans(rmat[1:4, ]), colMeans(rmat))
> hist(mns[, 1], prob = TRUE, main = "Samples of Size 4")
> lines(density(mns[, 1]), col = "red")
> hist(mns[, 2], prob = TRUE, main = "Samples of Size 4")
> lines(density(mns[, 2]), col = "red")
```



# Simulations in R

## Simulation-Based Inference

- R is often used for simulation-based inference
  - simulation-based exact tests
  - bootstrapping
  - Bayesian inference via MCMC
- The `boot` package is the most used bootstrapping framework.
- MCMC support:
  - several generic MCMC packages
  - several interfaces to BUGS
  - more complex model samplers are often coded directly in R.



# Some Aspects of Software Development in R

- Features of R that support implementing new methods:
  - High level language
  - Interface to lower level languages
  - Effective package mechanism
  - Documentation and testing support
  - Profiling tools
  - Support for embedding in other applications
  - [Sweave](#) for integrating R with  $\text{\LaTeX}$
- Some features of the language:
  - Lazy evaluation
  - Lexical scope
  - Name spaces
  - Error handling facilities
  - Object-oriented programming support



# Some Aspects of Software Development in R

## Lexical Scope

- Nested functions capture their defining environment
- This is a major difference from S-plus
- A simple example: creating a Bernoulli log-likelihood function.

```
> mkBernoulliLogLik <- function(x) {  
+   n <- length(x)  
+   k <- sum(x)  
+   function(p) k * log(p) + (n - k) * log(1 - p)  
+ }  
> f <- mkBernoulliLogLik(rbinom(20, 1, 0.3))  
> f
```

```
function (p)  
k * log(p) + (n - k) * log(1 - p)  
<environment: 0x24eea10>
```



# Some Aspects of Software Development in R

## Name Spaces

- Name spaces insure that only explicitly exported functions and variables are globally visible.
- Private functions and variables are seen by code defined in a package but not by code in other packages.
- Functions in a package see only private functions and explicit imports.
- Essential for reliable working of large number of independently developed packages.
- Implemented using lexical scope.





# Some Aspects of Software Development in R

## Some Open Issues

- Too many object systems (S3, S4)
- Too many graphics systems (standard, grid)
- Event loop support less than ideal
- Insufficient support for dynamic graphics
- Keeping workspace in memory has drawbacks
- Improvements in documentation system
- More support in detecting coding errors
- Performance improvement



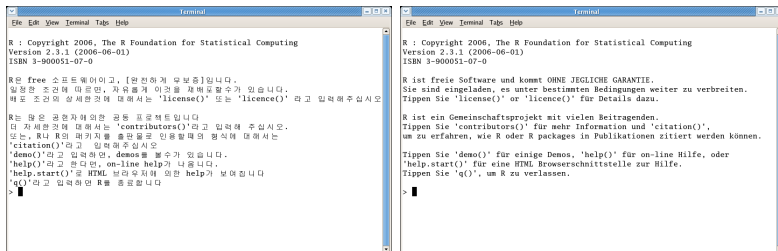
# Some Recent Developments

- New major releases occur twice a year.
- Many changes are minor improvements and bug fixes.
- Other changes are larger and more noticeable.
- Some recent changes:
  - Substantial changes and improvements to S4 classes and methods.
  - Performance improvements by moving some key functions to C.
  - Experimental support for memory profiling.
  - Enhancements to support for embedding.
  - Support to share C code among several packages.
  - Improved support for FORTRAN 90/95 in packages
  - Lazy loading of package to reduce memory and improve start-up.
  - Support for internationalization and localization.



# Some Recent Developments

## R console on Fedora Core 5 in Korean and German locales:



```
R : Copyright 2006, The R Foundation for Statistical Computing
Version 2.3.1 (2006-06-01)
ISBN 3-900051-07-0

R은 free 소프트웨어이고, [완전하게] 무료입니다.
일정한 조건에 따르면, 자유롭게 이것을 재배포할 수가 있습니다.
배포 조건과 상세한 것에 대해서는 'license()' 또는 'licence()' 라고 입력해 주십시오.

R은 많은 공헌자에 의한 공동 프로젝트입니다.
더 자세한 것에 대해서는 'contributors()'라고 입력해 주십시오.
또는, R나 R의 패키지를 출판물로 인쇄할 때의 형식에 대해서는
'citation()'라고 입력해 주십시오.
'demo()'라고 입력하면, demos를 볼 수가 있습니다.
'help()'라고 한다면, on-line help가 나옵니다.
'help.start()'로 HTML 브라우저에 의한 help가 보아집니다.
'q()'라고 입력하면 R를 종료합니다.
>
```

```
R : Copyright 2006, The R Foundation for Statistical Computing
Version 2.3.1 (2006-06-01)
ISBN 3-900051-07-0

R ist freie Software und kommt OHNE JEGLICHE GARANTIE.
Sie sind eingeladen, es unter bestimmten Bedingungen weiter zu verbreiten.
Tippen Sie 'license()' or 'licence()' für Details dazu.

R ist ein Gemeinschaftsprojekt mit vielen Beitragenden.
Tippen Sie 'contributors()' für mehr Information und 'citation()',
um zu erfahren, wie R oder R packages in Publikationen zitiert werden können.

Tippen Sie 'demo()' für einige Demos, 'help()' für on-line Hilfe, oder
'help.start()' für eine HTML Browserschnittstelle zur Hilfe.
Tippen Sie 'q()', um R zu verlassen.

>
```



- Code analysis tools
  - static analysis for error detection
- Byte code compilation
  - performance improvement
  - error detection
- Parallel computing
  - user-directed message passing ([snow](#) package)
  - automated parallel vectorization
  - using vectorized BLAS implementations
- Graphics
  - 3D contour plots
  - other 3D volume visualization methods



# Parallel Computing in R

## The snow Package

- **snow**: Simple Network of Workstations
- Should make handling “embarrassingly parallel” problems easy.
- Built on **rpvm**, **Rmpi**, or sockets.
- Master R process starts a cluster of worker R processes.
- Tasks are split onto the workers, results collected on the master.
- A simple parallel bootstrap:

```
> library(snow)
> cl <- makeCluster(10)
> clusterSetupRNG(cl)
> clusterEvalQ(cl, library(boot))
> b <- clusterCall(cl, boot, nuke.data, nuke.fun, R = 100, m = 1,
+   fit.pred = new.fit, x.pred = new.data)
> stopCluster(cl)
```



# Parallel Computing in R

The snow Package

- Other functions available include
  - `clusterApply`: parallel version of `lapply`
  - `clusterApplyLB`: load-balancing version of `clusterApply`.
- Current work is looking into:
  - Allowing user interrupts of parallel computations.
  - Better handling of R level errors.
  - Handling network/node failure
  - Managing sessions of parallel computations
  - Ways of limiting unintended data transfer.
- Key goal: retain the simplicity of the current system.



# Parallel Computing in R

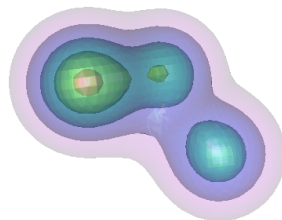
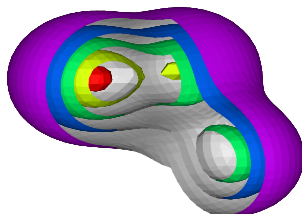
## Automatic Parallelization of Vectorized Arithmetic

- Dual-core processors are becoming common.
- Dual dual-core processor will be available in many workstations.
- Speedups of 2 to 4 times are useful
  - but *not* enough to justify manual parallelization
  - automated parallelization may be able to take advantage.
- Basic idea: if  $x$  and  $y$  are vectors and  $f$  is a vectorized primitive operation, use 2 (or 4) threads to compute  $f(x,y)$ .
- Problem: Synchronization overhead.
- Possible solutions:
  - careful tuning to each primitive operation
  - compilation to fuse primitive operations
- Design issue: How to allow packages to define new operations that take advantage of the parallel vectorization framework.



# Isosurfaces in R

- Isosurfaces are useful for visualizing
  - functions of three variables
  - volume data recorded on a 3D grid (e.g medical images)
- The Marching Cubes algorithm is used to construct isosurfaces.
- Visualization of nested surfaces is challenging.
- Some examples:





# Conclusions

- R has been successful on a number of dimensions:
  - providing a valuable tool for data analysis
  - providing a framework for disseminating new methodology
  - enabling the development of other large projects (Bioconductor)
  - serving as a framework for statistical computing research
  - illustrating the value of the open source development model
- Success does have its costs:
  - changes now affect a large number of users; this can limit innovation
  - the desire to support new, less experienced users can come at the cost of not supporting advanced use as well
- Nevertheless, R can be expected to evolve and improve for a number of years to come.



- Books

- Dalgaard, *Introductory Statistics with R*
- Venables and Ripley, *Modern Applied Statistics with S*
- Many others

- Web resources

- Several good introductions
- R graphics gallery <http://addictedtor.free.fr/graphiques/>
- R home page <http://www.r-project.org>

- Mailing lists, list archives, [RSiteSearch](#)

- R Wiki <http://wiki.r-project.org/>

