An Intermediate Course in Statistical Computing

Richard M. Heiberger
Temple University

I have just completed teaching the latest incarnation of a course on Statistical Computing for second year graduate students in Statistics. This talk summarizes my course and includes responses to comments made by Peter Dalgaard in his talk at the UseR 2009 conference. My topics include

1. Elementary numerical analysis: Floating point representation of numbers. Prevention of disastrous cancellation.
2. Numerical Techniques: Numerical linear algebra. Numerical differentiation and quadrature (integration). Pseudo-random number generation.
3. Parsing. Precedence of operations. Use of the same parser for arithmetic, `?plotmath`, and model formulas.
4. Interprocess communication. Accessing the rest of the computer and the rest of the internet directly from inside R.
5. Graphics. Incremental development of complex graphical displays.
6. Packages. The course requires each student to construct a small R package. The goal is to understand the discipline and the mechanism of packaging software so others can use it. We cover methods, function design, debugging, and documentation.

1 Floating Point

> FAQ 7.31. Why doesn't R think these numbers are equal?
> The only numbers that can be represented exactly in R's numeric type are integers and fractions whose denominator is a power of 2. Other numbers have to be rounded to (typically) 53 binary digits accuracy.

This really is a *Frequently Asked Question*, a variant appears about once a week on the R-help list. I use the current week's question as HW and as an exam question. This is one of the most important issues, and I think one of the most difficult. Floating point representation of numbers is not intuitive.

I used three approaches.

1. Read the Goldberg article referenced in the FAQ for background.
2. Run the `machar` program (Cody's subroutine which detects machine characteristics) on our PC and on several pocket calculators. This is quite a fascinating exercise. On our individual PC, and on our Unix server, we found (no great surprise) the values in `.Machine`. I also displayed the values from running `machar program` on an IBM 360 and on a DEC VAX.
3. Look at several examples:

(a) Display floating point numbers in several formats.

```
> .66 == (1-.34)
[1] FALSE
> .66 - (1-.34)
[1] 1.110223e-16
```

I found several hex displays of numbers in R. None are pedagogically what I want. The best is `sprintf("%+13.13a", x)`

decimal	`x`	`sprintf("%+13.13a", x)`	`sprintf("%+17.17f", x)`
0.66	`0.66`	`+0x1.51eb851eb851fp-1`	`+0.66000000000000003`
$1 - .34$	`0.66`	`+0x1.51eb851eb851ep-1`	`+0.65999999999999992`
$.66 - (1 - .34)$	`1.110223e-16`	`+0x1.0000000000000p-53`	`+0.00000000000000011`
		`+0x0.0000000000001p-1`	

Note that both hex displays have repeated hex digits of `51eb8`. In the first `51eb` was rounded to `51f` and in the second, `51eb` was chopped to `51e`

The hex representation I really want is an "engineering"-style extension of `%a` format that would allow me to specify powers in units of $2^4 = 16$ (similar to the ENG format in the TI-83 and similar calculators) or in user specified units. In the example above, I manually calculated the unnormalized difference displayed with the same exponent as the two numbers that are subtracted.

(b) Show a simple example of disastrous cancellation

```
a^2 - b^2 != (a+b)*(a-b) ## for some surprising values of a and b
```

	Decimal 100 Hex `+0x64`		Decimal 134217728 = Hex `+0x8000000`	
`x`	100	`+0x1.9000000000000p+6`	134217728	`+0x1.0000000000000p+27`
`a <- x+1`	101	`+0x1.9400000000000p+6`	134217729	`+0x1.0000002000000p+27`
`b <- x+2`	102	`+0x1.9800000000000p+6`	134217730	`+0x1.0000004000000p+27`
`a^2`	10201	`+0x1.3ec8000000000p+13`	18014398777917440	`+0x1.0000040000000p+54`
`b^2`	10404	`+0x1.4520000000000p+13`	18014399046352900	`+0x1.0000008000001p+54`
`b^2 - a^2`	203	`+0x1.9600000000000p+7`	268435460	`+0x1.0000004000000p+28`
`(b+a) * (b-a)`	203	`+0x1.9600000000000p+7`	268435459	`+0x1.0000003000000p+28`

The outlined 0 in the decimal column for `a^2` with `x=+0x8000000` should be a 1 if we had arbitrary precision arithmetic. The marker ↕ in the hex column for `a^2` with `x=+0x8000000` shows that one more hex digit would be needed to precisely indicate the squared value.

(c) HW: calculate the sum of squares of three numbers in 2-digit base-10 arithmetic. This requires rounding to 2 digits at *every* intermediate step. We begin by parsing the expression into Polish notation to see the complete set of intermediate steps.

```
library(codetools)
showTree(quote(

   2^2 + 11^2 + 15^2

))

(+ (+ (^ 2 2) (^ 11 2)) (^ 15 2))


2  11  15  numbers
4 121 225  square each
4 120 220  round each to 2 digits (explain "round to even")
 124  220  add the first two
 120  220  round the sum of the first two to 2 digits
   340     add the partial sum and the third

## R  ## full precision of 53 binary digits
> 2^2 + 11^2 + 15^2
[1] 350
```

(d) Representable Numbers

If the question and answer are both in the range of representable numbers, then it should be possible to get there.

```
> ## Pythagoras
```

```
> Mod( 3   + 4i )
[1] 5
```

```
> Mod( 3e154   + 4e154i )
[1] 5e+154
```

```
> sqrt(3^2 + 4^2)
[1] 5
```

```
> sqrt(3e154^2 + 4e154^2)
[1] Inf
```

```
> .Machine$double.xmax
[1] 1.797693e+308
```

```
> 1e154^2
[1] 1e+308
```

```
> 3e154^2
[1] Inf
```

```
> sqrt((3e154/4e154)^2 + (4e154/4e154)^2) * 4e154
[1] 5e+154
```

2 Numerical Techniques

1. Numerical linear algebra.

   $X = QR$ decomposition and linear models.

   Eigenvalue decomposition $S = V\Lambda V'$.

2. Numerical differentiation and quadrature (integration).

   $$\left.\frac{dy}{dx}\right|_a = \frac{y(a+\delta) - y(a-\delta)}{2\delta}$$

   $$\int_a^b f(x)\,dx = \sum_{i=1}^{n} f(x_i)$$

3. Pseudo-random number generation.

   Most computer programs give the *same* answer each time you type the same expression.
   Pseudo-random generators give a *different* answer each time you type the same expression.

3 Parsing

1. Precedence of Operations

```
    1: 3 ^2

   (1: 3)^2
    1:(3 ^2)
```

```
>     1: 3 ^2
[1] 1 2 3 4 5 6 7 8 9
>    (1: 3)^2
[1] 1 4 9
>     1:(3 ^2)
[1] 1 2 3 4 5 6 7 8 9
```

```
    1: 3 +2

   (1: 3)+2
    1:(3 +2)
```

```
>     1: 3 +2
[1] 3 4 5
>    (1: 3)+2
[1] 3 4 5
>     1:(3 +2)
[1] 1 2 3 4 5
```

```
## showTree

    1: 3 ^2  ## (: 1 (^ 3 2))
```

```
    1: 3 +2    ## (+ (: 1 3) 2)
```

2. Arithmetic, Model Formulas, and `plotmath` use the same parser

```
library(codetools)
showTree(quote(

   (a+b)^2

))
```

```
(^ ("(" (+ a b)) 2)
```

arithmetic	`(a+b)^2`	if $(a = 2, b = 3)$, then $5^2 = 25$
model formula	`y ~(a+b)^2`	`y ~  a + b + a:b`
`plotmath`	`expression( (a+b)^2 )`	$(a + b)^2$

4 Interprocess communication

Elementary examples use `system` and `shell` to access other software on the current computer system. More complex capabilities are built on those. The goal is to use each type of software on your system at its strength and then combine them.

Typical examples are

C and Fortran: Much of the hard work of R is written in C and Fortran. Computational linear algebra in R, for example the `lm` and related routines, use the numerical algorithms from LAPACK and LINPACK. User writing their own packages often will include C or Fortran subroutines in their packages.

`read.file, source`: R functions takes a `url` as argument and therefore can read any file, anywhere in the world, directly from within R. This example is from R-Help. It shows how to update one function within a package by getting a copy of the development source

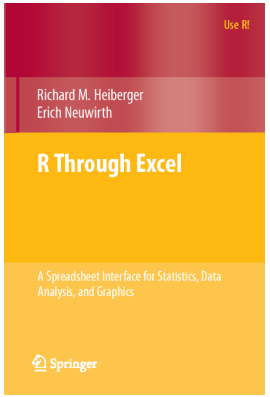
```
## Pull in development version of the zoo function na.locf.zoo

source("http://r-forge.r-project.org/.../na.locf.R?revision=725&root=zoo")
```

Timer: The computer provides time services. The `tcltk` package has tools that allow R to use those services to control calculations. This example runs in the background and prints `Hello you!` on the screen every second. It does not interfere with other use of the computer.

```
library(tcltk)

z <- function () {
    cat("Hello you!\n")
    .id <<- tcl("after", 1000, z)
}

.id <<- tcl("after", 1000, z)

tcl("after", "info", .id)   # To get info about this scheduled task
tcl("after", "cancel", .id) # To cancel the currently scheduled task
```

RExcel: Microsoft Excel is the most widely used spreadsheet program. Many of our clients and students use it as their data management system and as their working environment.

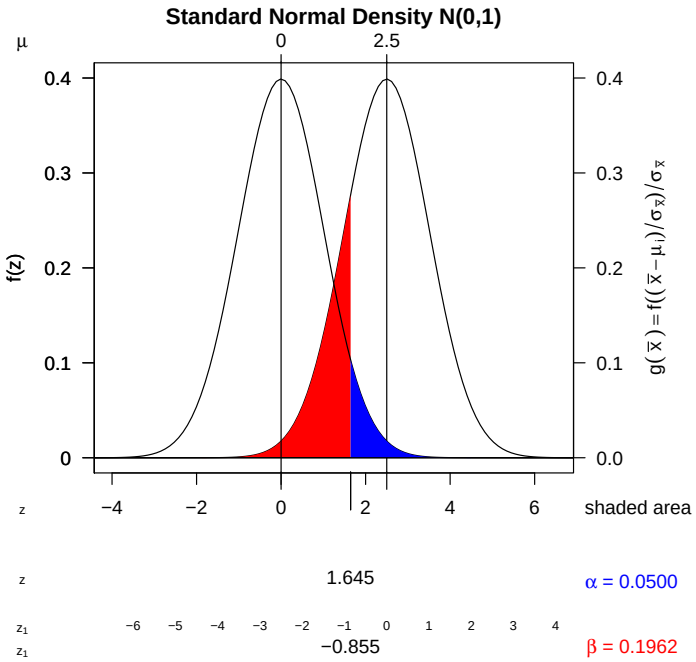
On Windows `RExcel` and `statconnDCOM` (`http://rcom.univie.ac.at`) access COM—the Microsoft interprocess communications system, and seamlessly integrates the entire set of R's statistical and graphical methods into Excel.



I show one example here, from our book *R through Excel*

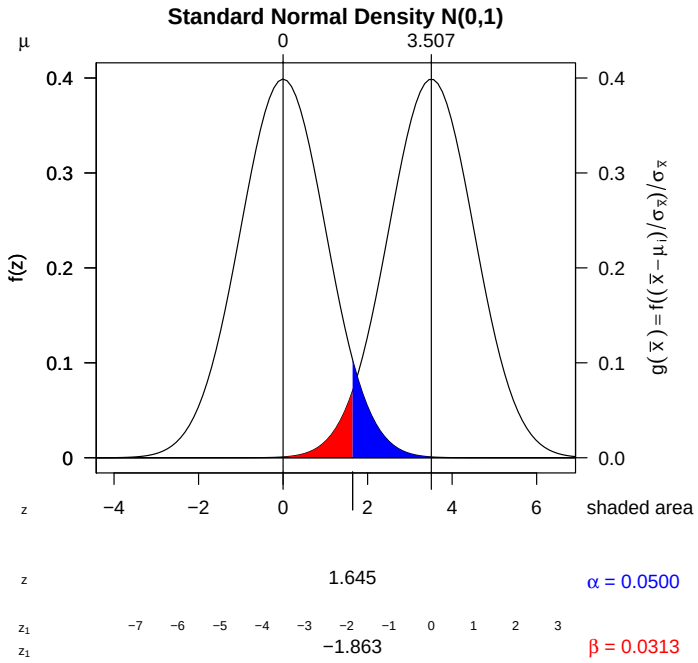
There are many more examples included with the software. There are several other talks at this UseR! 2010 conference illustrating applications of RExcel.

	A	B	C	D	E
1			Show		
2	Optional user input		Slider		on Graph
3	μ_0	0	☐		Display
4	μ_1	2.5	☑	◀ ▶	Display
5	z		☐		
6	σ	1			
7	n				
8	v				
9					
10	☐ α left	☑ α right	α:	◉ prob or hypoth	
11		0.050	0.050	○ confidence interval	
12		◀ ▶			



Placing values in the cells in Excel provides live control of the curve displayed in the R graph. The slider on μ_1 in Excel, smoothly moves the normal curve centered at μ_1 and adjusts the corresponding area illustrating β, the probability of the Type II Error.

	A	B	C	D	E
1			Show		
2	Optional user input		Slider		on Graph
3	μ_0	0	☐		Display
4	μ_1	3.5075	☑	◀ ▶	Display
5	z		☐		
6	σ	1			
7	n				
8	v				
9					
10	☐ α left	☑ α right	α:	◉ prob or hypoth	
11		0.050	0.050	○ confidence interval	
12		◀ ▶			

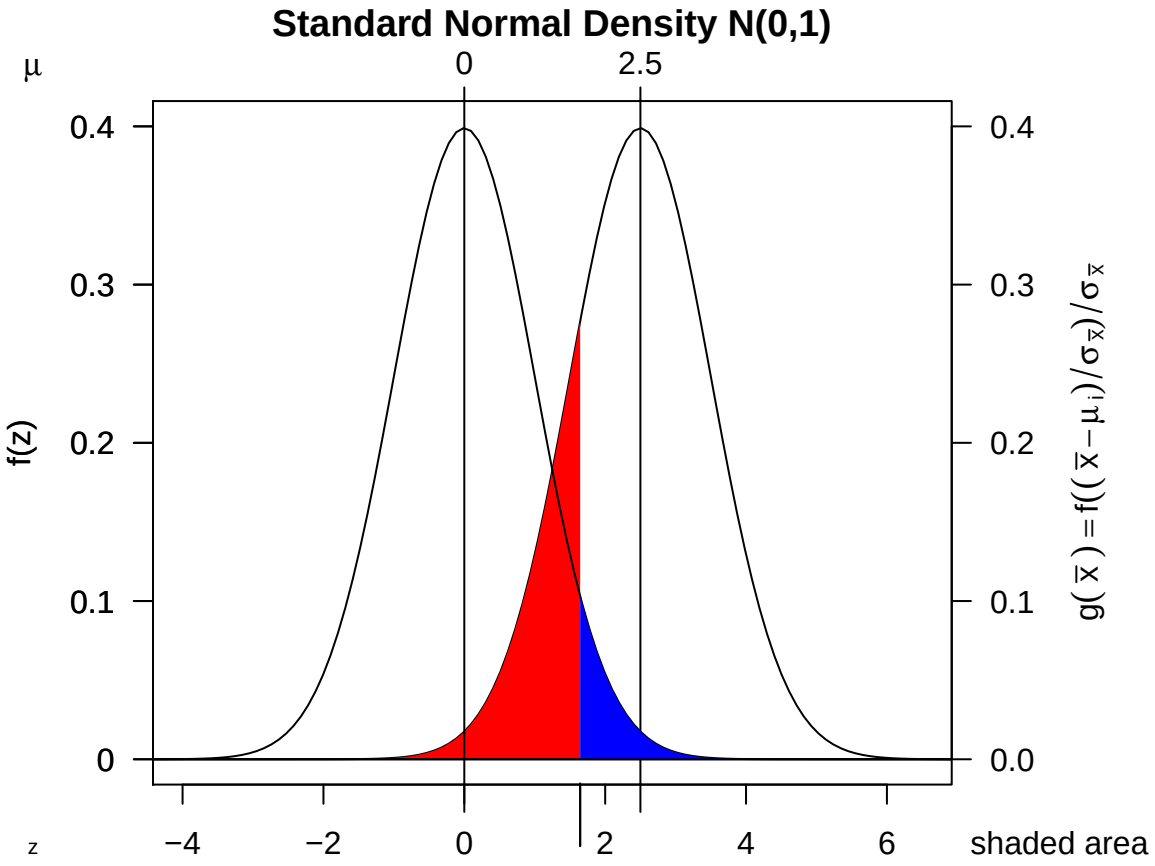


Placing values in the cells in Excel provides live control of the curve displayed in the R graph. The slider on μ_1 in Excel, smoothly moves the normal curve centered at μ_1 and adjusts the corresponding area illustrating β, the probability of the Type II Error.

5 Graphics

Graphical design techniques that can be addressed with the power graph are

1. Both curves are drawn with the same type of statement, but with shifted center.
2. Both highlighted areas are constructed with the same software. In one case it is parameterized to highlight an area to the right of a critical value and in the other case to the left of a (different) critical value.
3. The label in right margin shows an application of `plotmath`.



6 Packages

Modern Computing builds on existing software components. Therefore I include a unit on building an R package. My initial assignment is to design and construct a package that passes `R CMD check`. At least one package from the class is now on CRAN. There are several others that, with a little more work, could be placed on CRAN.

Many topics grow organically from both my private reading and from class presentation of the students' packages. This unit lasted several weeks. Most students submitted at least four versions of their package, each containing changes and recommendations from both `R CMD check` and from my comments during class.

1. Specific items that are covered, and that are enforced by the software, are
   (a) Coherent documentation of your package. The `.Rd` structure.
   This leads to a discussion on proper use of writing tools (LaTeX and MS Word).
   (b) Valid examples.
   (c) `plot.myclass` and how it fits with generic functions.
   (d) R-style of writing coordinated sets of functions work on user-defined classes. Separate functions for analyzing data, displaying tables, displaying plots.

2. Use of R documentation

   (a) Answers to most of your questions on packages, including how to read the error messages that `R CMD check` will find are in the Extensions manual.
   `c:/Progra 1/R/R-2.10.1/doc/manual/R-exts.html`
   Now that you all have completed the initial process of getting something to run, the details of `R-exts.html` will make much more sense.

   (b) Copyright and licensing

   (c) Read the R FAQ and the R Windows FAQ and find something you didn't know before.

   (d) Function and topic documentation
   ```
   ?functionname
   ??topic
   RSiteSearch("topic")
   ```

3. System Coordination

   (a) Source packages `tar.gz` and compiled packages `zip` for Windows.

   (b) File compression (both `zip` and `tar.gz`).

   (c) Environment Variables, specifically `PATH`.
   Understanding these are needed to make `Rtools` work on Windows.

   (d) Why does `source("myfile.doc")` usually not work?
   The answer includes a discussion of the distinction between
   word processors and text editors.

   (e) Version numbering.

4. Programming

   (a) Comments: Use when informative. Repeating the R statement is silly.

```
## Examples:
X.qr <- qr(X)
Q <- qr.Q(X.qr) ## assign matrix Q from QR decomposition of X (redundant)
R <- qr.R(X.qr) ## isolate Q and R factors (informative)
```

   (b) Efficiency of coding. Matrix and vector operations. `system.time()`

```
A <- matrix(rnorm(40), 8, 5)
B <- matrix(rnorm(30), 5, 6)

system.time(for (m in 1:10000) {                ## 32.66 seconds
C <- matrix(0, nrow(A), ncol(B))
for (i in 1:8)
   for (j in 1:6)
     for (k in 1:5)
        C[i,j] <- C[i,j] + A[i,k] * B[k,j]
})

system.time(for (i in 1:10000)  C <- A %*% B )  ##  0.03 seconds
```

   (c) Proper indentation of code. R-aware editors.
   I recommend ESS with emacs. There are others.

(d) Loops for sequential algorithms, not for parallel algorithms. Distinguish the concepts. Use `apply` family of functions when possible.

(e) Missing values: `na.action=na.exclude`, `na.rm=TRUE`.

(f) Boundary situations: for example `A[, i, drop=FALSE]` to retain the matrix class on a single column from a matrix.

(g) Time classes: Gabor Grothendieck and Thomas Petzoldt. R Help Desk: Date and time classes in R. R News, 4(1):29-32, June 2004.
`http://cran.r-project.org/doc/Rnews/Rnews_2004-1.pdf`

(h) Debugging: `recover`, `trace`, `print.default`, `str`, `options(error=recover)`.

7 Summary

1. Elementary numerical analysis.
2. Numerical Techniques.
3. Parsing.
4. Interprocess communication.
5. Graphics.
6. Packages.

8 References

1. Baier, T. and Neuwirth, E. (2007) "Excel :: Com :: R". *Computational Statistics*, 22 (1): 91–108. `http://www.springerlink.com/content/uv6667814108258m/fulltext.pdf` You can download this paper for no charge if your library subscribes.

2. Heiberger, Richard M., and Erich Neuwirth (2009). *R through Excel: A Spreadsheet Interface for Statistics, Data Analysis, and Graphics*, Springer–Verlag, New York. Series: Use R!
   `http://www.springer.com/978-1-4419-0051-7`

3. R Development Core Team (2010). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, `http://www.R-project.org`

Appendix

Floating Point characteristics of several computers

Hardware	arithmetic base `ibeta`	number of digits `it`	rounding method irnd
Intel i386	2	53	2
Intel x86_64	2	53	2
IBM 360, single	16	6	0
IBM 360, double	16	14	0
DEC VAX, single	2	24	1
DEC VAX, double	2	56	1
TI-83	16	11	0

1. HW: run `machar.c` manually on your calculator to determine `ibeta` and `it`. You are the CPU. The while loop in C is similar to the while loop in R which you can see with

```
?Syntax
?Control
```

From the TI documentation

> To maximize accuracy, the TI-83 Plus carries more digits internally than it displays. Values are stored in memory using up to 14 digits with a two digit exponent.

```
a         1.759218604e13

b         2
t      a+b 1.759218604e13
i      t-a 0                (a+b)-a != b

b      b+b 4
t      a+b 1.759218604e13
i      t-a 0                (a+b)-a != b

b      b+b 8
t      a+b 1.759218604e13
i      t-a 0                (a+b)-a != b

b      b+b 16
t      a+b 1.759218604e13
i      t-a 16               (a+b)-a == b
```