An Introduction to R

Kendall Giles

Department of Statistical Sciences and Operations Research
Virginia Commonwealth University
and
Human Language Technology Center for Excellence
Johns Hopkins University

27 August 2009

It's all about the data

Google's chief economist Hal Varian:

The ability to take data—to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it*—that's going to be a hugely important skill in the next decades, not only at the professional level but even at the educational level for elementary school kids, for high school kids, for college kids. Because now we really do have essentially free and ubiquitous data. So* ***the complimentary scarce factor is the ability to understand that data and extract value from it****.*

R

- R is a language and environment for statistical computing and graphics
- From New York Times Jan 7, 2009 article: "Data Analysts Captivated by R's Power":

> *"R is really important to the point that it's hard to overvalue it," said Daryl Pregibon, a research scientist at Google, which uses the software widely. "It allows statisticians to do very intricate and complicated analyses without knowing the blood and guts of computing systems."*

Why R?

- Powerful: mirrors in many ways the S language developed at Bell Laboratories — is designed around a true computer language
- FREE: no costly licenses or fees.
- Extensible: Is part of the GNU project — user-contributions are welcomed and additional features are constantly being added
- Useful: Generated plots can be dropped into journal articles, at (statistics) conferences fairly common to see research results generated with R
- Insightful: R helps the student understand principles and concepts, unlike black-box packages like SPSS/SAS and Minitab
- Popular: R provides an API that can be utilized by other languages and programs
- Helpful: Many help files and online resources

Getting R

- Get it at: `http://www.r-project.org/`
- There is a FAQ that has installation instructions if you need it –> pick the binary version for OS

DEMO –> `http://www.r-project.org/`

Running R

DEMO: running R, after it is installed on your system

Examples:

- basic calculator: $2 + 2$; 3 ˆ 4; 31 $\%\%$ 7; 31 $\%/\%$ 7
- functions: sin(pi/4)
- variables: x <- 12
- "x <- c(1,5,9,10)" vs "c(1,5,9,10)" vs "sum(c(1,5,9,10))"
- x <- seq(from=2, to=8, by=2); x[1]; x[-2]
- x <- rep(2, times=11); x <- rep(seq(1,2),2)
- y <- x + 1; y <- x + c(0,0,1,1)
- plot(sin, -pi, 2*pi)

R general use

- How to get help:
  - help(c)
  - help.search("matrix")
  - apropos("matrix")
  - help.start() –> R's "An Introduction to R"
  - tutorial videos –> `http://www.stat.vcu.edu/help/R/index.html`
- Note: working directory, scripts file suggestions
- objects()

Data

- Most data we use tends to be large –> # observations AND/OR # features
- Some raw data must be pre-processed before it is in a usable form
- Small amounts of data can be directly entered using the function `c()`
  - e.g., `foo <- c(2.1, 3.2, 4.4)`
- Most of the larger datasets we will encounter are in ASCII text files

NB: Microsoft Word $\neq$ text editor

Types of data

- numeric: e.g., 1, 3.1415, 0
- character: e.g., "cancerous", "true", "one", "green"
- logical: TRUE

Reading in a vector of data

72
69.5
74
76
68
...

- SPECIAL NOTE ABOUT R's WORKING DIRECTORY: `getwd()`
- If the data in the file is just one vector (many observations, one variable), then you can read it in using `scan()`
  - e.g., to read in the file `mycounts.txt`:
    ```
    mydata <- scan("mycounts.txt")
    ```

Manipulating Matrices

72	100	...
69.5	120	...
74	110	...
76	89	...
68	156	...
...	...	...

Matrices

- `foo <- matrix(1:10, nrow=2, ncol=5)`
- `foo[2,3]`
- `foo[2,]`
- `foo[,1]`

Manipulating Data Frames

Data Frames are like matrices except the columns have names and the columns can be of different data types.

- `mycol <- c("blue", "green", "white")`
- `mycount <- c(10, 13, 16)`
- `car.inventory <- data.frame(mycol, mycount)`
- `names(car.inventory)`

Reading in multivariate data

- If the data have many observations and variables, these data are often represented as rows (observations) and columns (variables)
  - e.g., Fisher's Iris dataset: 50 samples from each of three species of Iris flower
  - NB: this dataset has numeric as well as categorical data, as well as variable (column) labels

```
mydata <- read.table("iris.txt", header=TRUE, sep=" ", as.is=TRUE)
names(mydata) mydata$sepal.length
```

Saving and Loading Variables

- The variables in memory can be listed using `objects()`
- R allows you to save a variable to disk

```
mydata <- read.table("iris.txt", header=TRUE, sep=" ", as.is=TRUE)
mydata <- mydata[,-2]
dump("mydata", "trimmed_iris.R")
```

- To read a variable back in to memory:

```
rm(mydata)
objects()
source("trimmed_iris.R")
mydata
```

Writing data to a file

- Sometimes you may want to write data to a file without the R syntax (like the "iris.txt" ASCII file)

```
write.table(mydata, "myiris.txt", quote=FALSE, row.names=FALSE)
```

Visualization

R has customizable functions to support the visual summarization and analysis of your data.

Some examples:

- timeseries `plot()`
- `boxplot()`
- scatterplot
- color scatterplots

How to write functions: 1

```
myhi <- function(foo)
{
     print(foo)
}

myhi()
```

How to write functions: 2

$$fv = pv(1 + ir)^n$$

```
interest.rate <- 0.01
present.value <- 100
num.periods <- 12
future.value = present.value*(1 + interest.rate)^num.periods

future.value <- function(num.periods, present.value, interest.rate)
{
     future.value <- present.value*(1 + interest.rate)^num.periods
     return(future.value)
}

fv <- future.value(12, 100, .01)
```

Algorithm Basics: for

R is built on a real programming language, so there are standard ways to control the flow of code.

`for` loop:

- e.g.,

```
for( i in 1:12) {
  cat( c(i, future.value(i, 100, .01), "\n") )
}
```

Algorithm Basics: if

`if`:

- e.g.,

```
for( i in 1:10 ) {
  foo <- sample( c(1:6), 1, replace=TRUE)
  if (foo > 3) {
    print("HI")
  }
  else {
    print("LOW")
  }
}
```

Algorithm Basics: while and repeat

- `while ( condition ) { statements }`
- `repeat { statements }`
  - `if (condition) break`

Packages

- R is an open system, so there are many packages available:
  - `library()`
  - `library(help="base")`
- To load a desired package:
  - `library("stats")`
- To search for a function:
  - `help.search("anova")`
- You can also look to R's Package Manager to see what packages are installed/loaded
- Use R's Package Installer to look for other user-contributed R packages
- You can also create your own packages and submit them to the R site for everyone to use

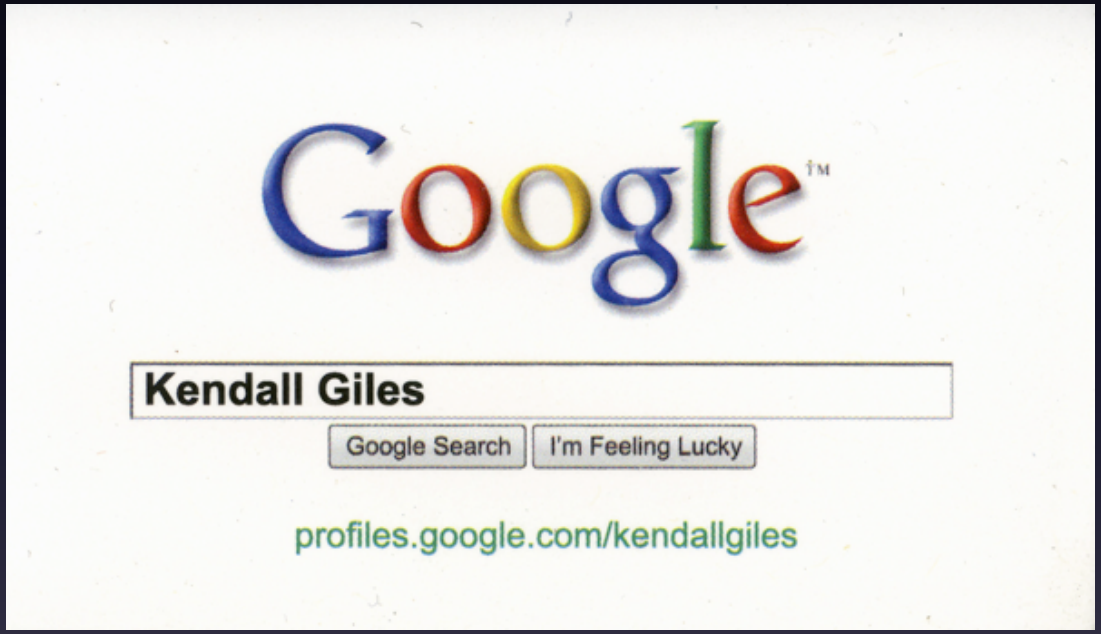
R API

- R provides an API so that you can call its routines from other languages:
  - e.g., you can make R routine calls from within your C code
- Some languages also provide interfaces to R:
  - e.g., you can also call R routines from with Python
  - `from rpy import *`
  - `foo1 = r.rnorm(100)`
  - `r.plot(r.table(r.rpois(100,5)), type="h", col="red", lwd=10, main="rpois(100, lambda=5)")`
- Or you can always have your language tell R to execute an R batch file from the command line:
  - `R CMD BATCH myscript.r`

Further Reading

- http://cran.r-project.org/doc/manuals/R-intro.pdf
- http://www.johndcook.com/R_language_for_programmers.html
- http://www.nytimes.com/2009/08/06/technology/06stats.html

Thank you!


Google
Kendall Giles
Google Search
I'm Feeling Lucky
profiles.google.com/kendallgiles