Introduction to R

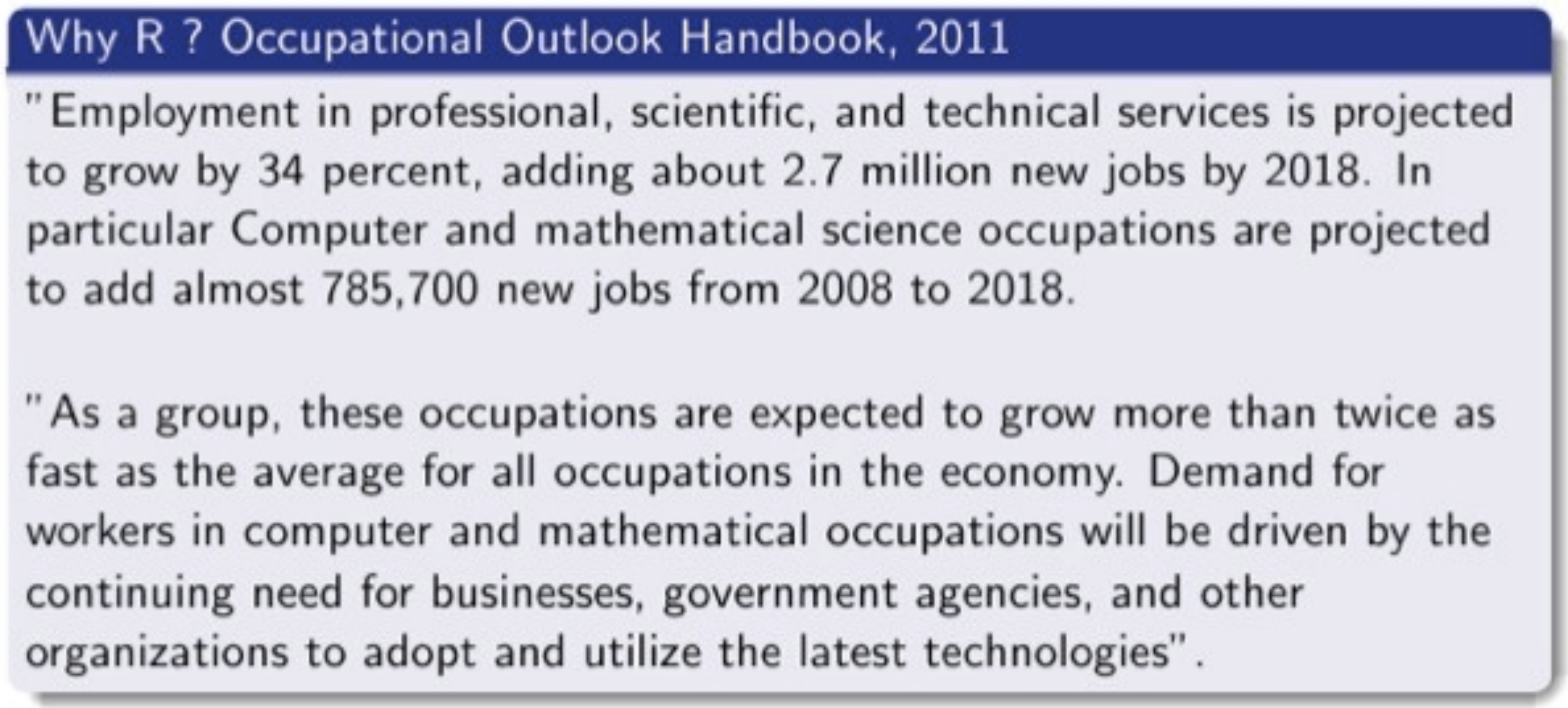
Steve Pittard

Dept of Biostatistics and Bioinformatics

Emory University

wsp@emory.edu

Introduction - Motivations

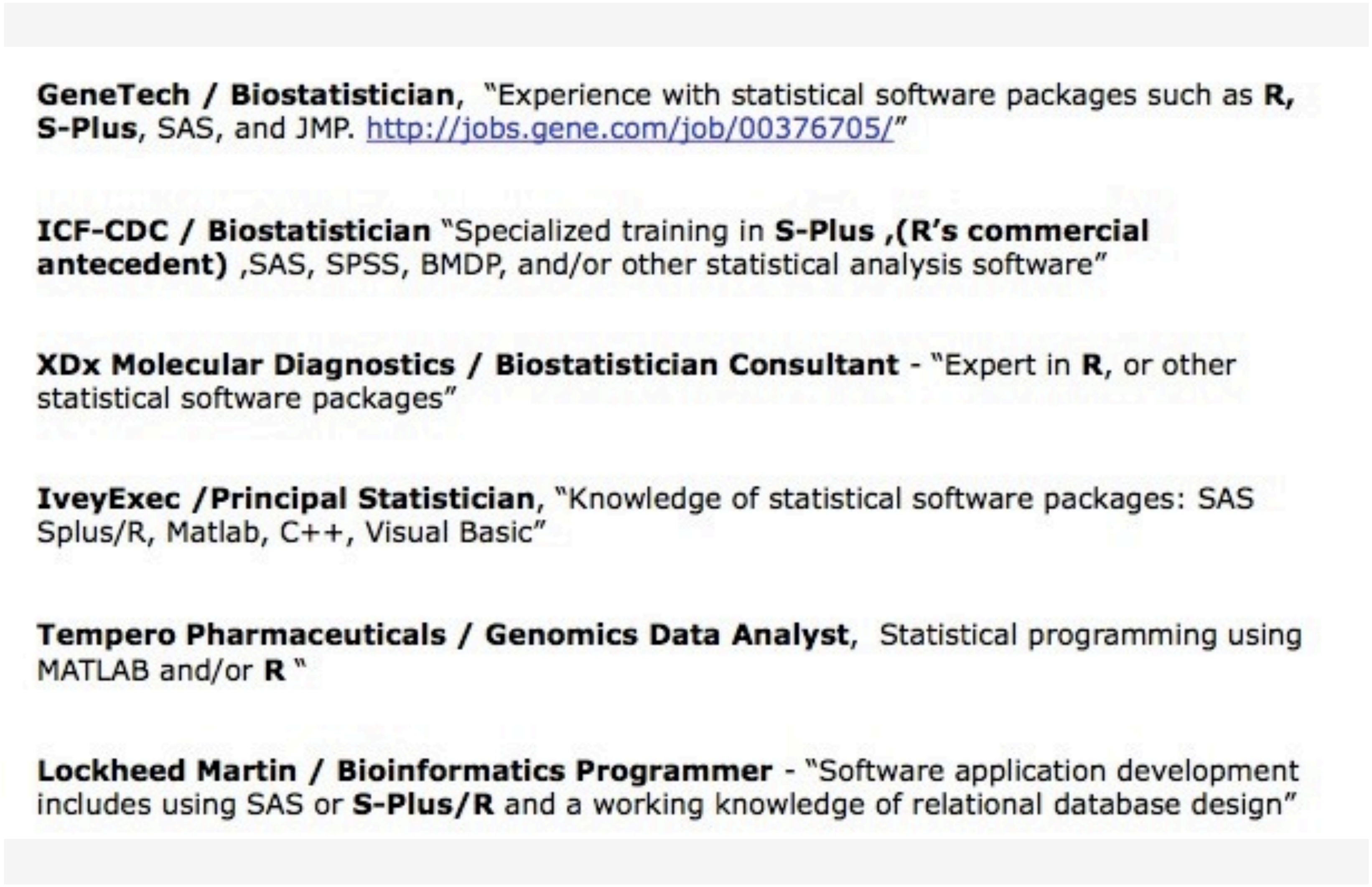


Why R ? Occupational Outlook Handbook, 2011

"Employment in professional, scientific, and technical services is projected to grow by 34 percent, adding about 2.7 million new jobs by 2018. In particular Computer and mathematical science occupations are projected to add almost 785,700 new jobs from 2008 to 2018.

"As a group, these occupations are expected to grow more than twice as fast as the average for all occupations in the economy. Demand for workers in computer and mathematical occupations will be driven by the continuing need for businesses, government agencies, and other organizations to adopt and utilize the latest technologies".

Introduction - Motivations



GeneTech / Biostatistician, "Experience with statistical software packages such as **R, S-Plus**, SAS, and JMP. http://jobs.gene.com/job/00376705/"

ICF-CDC / Biostatistician "Specialized training in **S-Plus ,(R's commercial antecedent)** ,SAS, SPSS, BMDP, and/or other statistical analysis software"

XDx Molecular Diagnostics / Biostatistician Consultant - "Expert in **R**, or other statistical software packages"

IveyExec /Principal Statistician, "Knowledge of statistical software packages: SAS Splus/R, Matlab, C++, Visual Basic"

Tempero Pharmaceuticals / Genomics Data Analyst, Statistical programming using MATLAB and/or **R** "

Lockheed Martin / Bioinformatics Programmer - "Software application development includes using SAS or **S-Plus/R** and a working knowledge of relational database design"

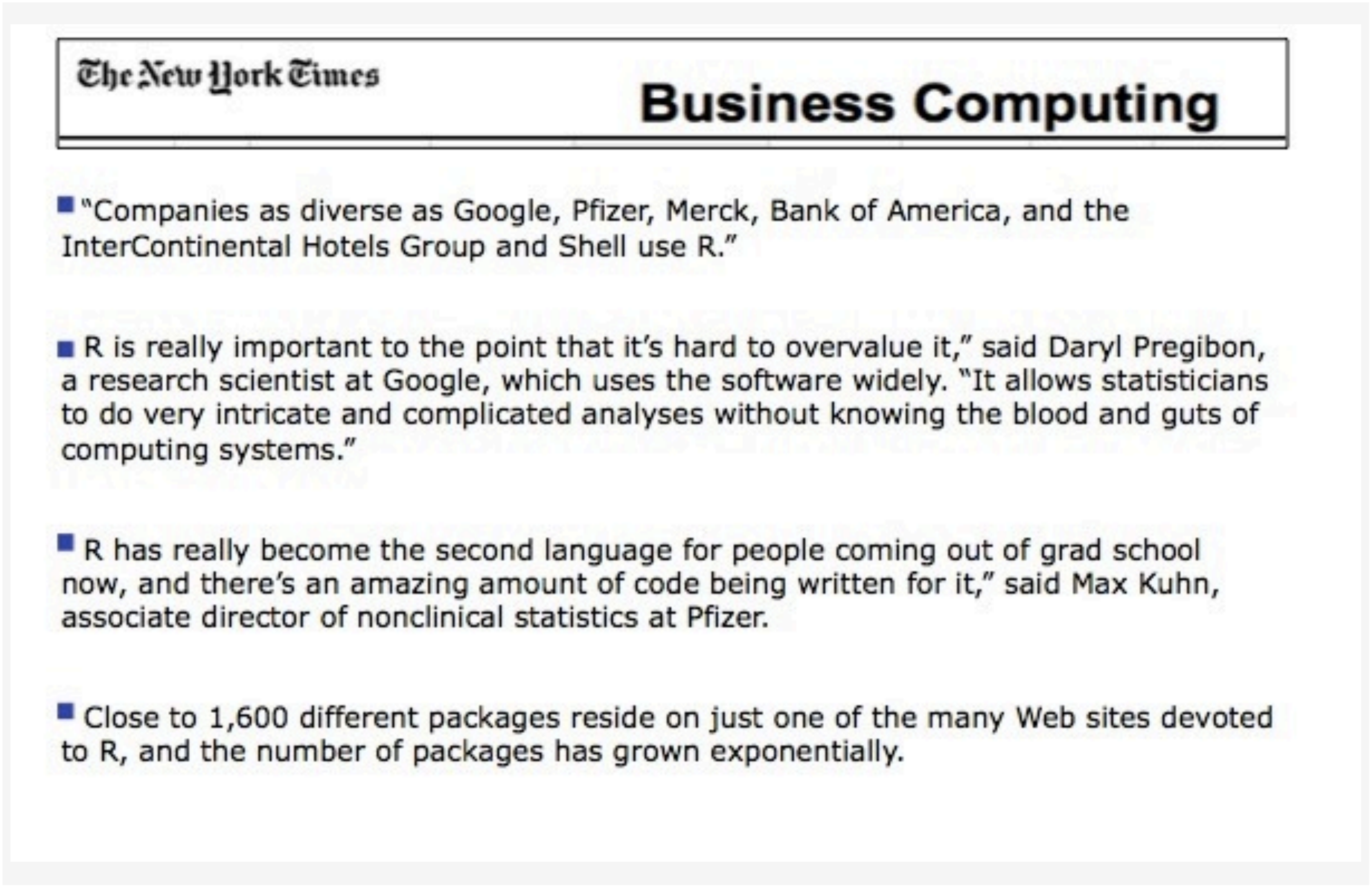
Introduction - Motivations

Data Analysts Captivated by R's Power The New York Times

Stuart Isett for The New York Times

R first appeared in 1996, when the statistics professors Robert Gentleman, left, and Ross Ihaka released the code as a free software package.

Introduction - Motivations

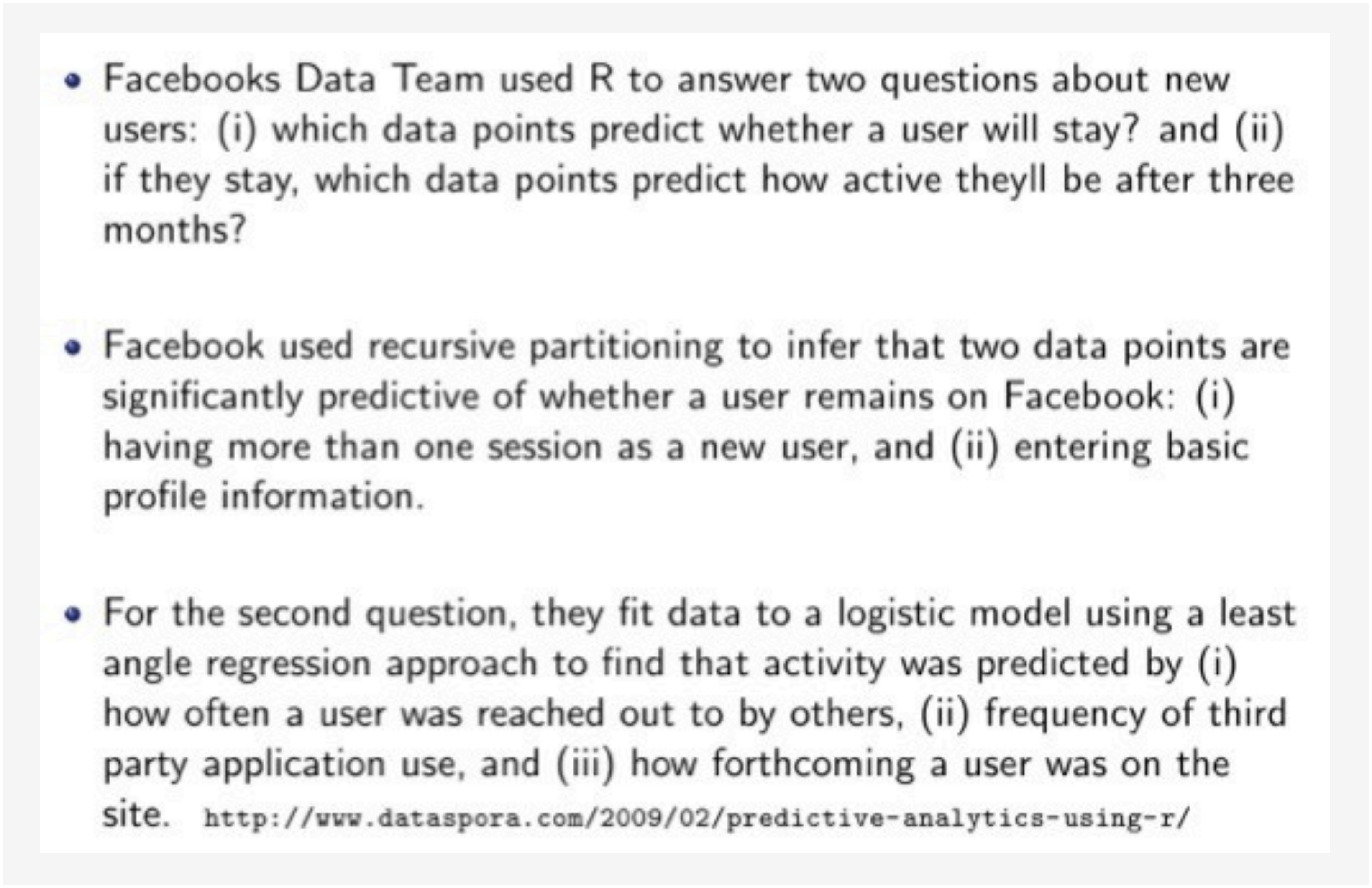


The New York Times

Business Computing

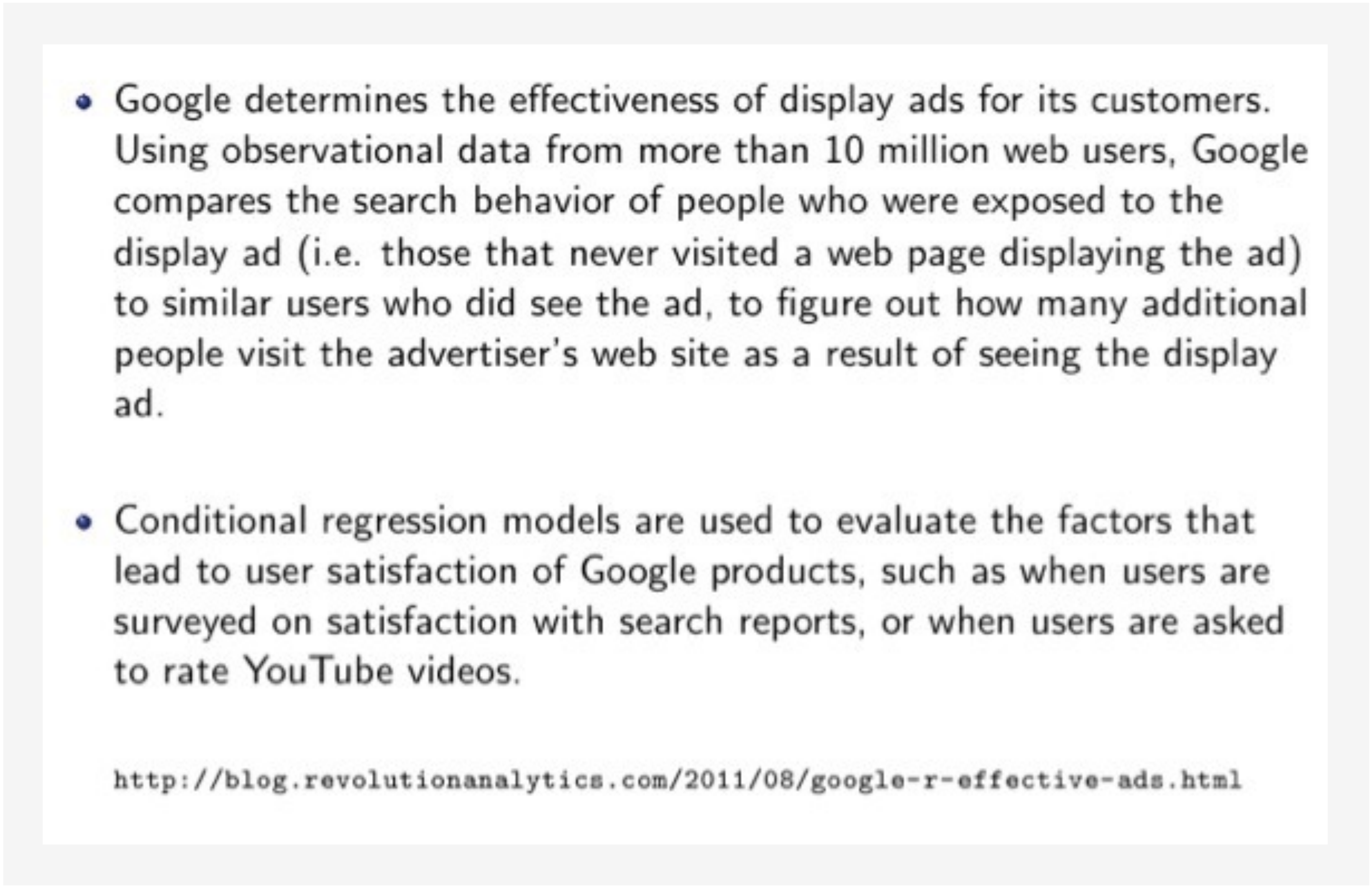
- "Companies as diverse as Google, Pfizer, Merck, Bank of America, and the InterContinental Hotels Group and Shell use R."
- R is really important to the point that it's hard to overvalue it," said Daryl Pregibon, a research scientist at Google, which uses the software widely. "It allows statisticians to do very intricate and complicated analyses without knowing the blood and guts of computing systems."
- R has really become the second language for people coming out of grad school now, and there's an amazing amount of code being written for it," said Max Kuhn, associate director of nonclinical statistics at Pfizer.
- Close to 1,600 different packages reside on just one of the many Web sites devoted to R, and the number of packages has grown exponentially.

Introduction - Motivations



- Facebooks Data Team used R to answer two questions about new users: (i) which data points predict whether a user will stay? and (ii) if they stay, which data points predict how active theyll be after three months?
- Facebook used recursive partitioning to infer that two data points are significantly predictive of whether a user remains on Facebook: (i) having more than one session as a new user, and (ii) entering basic profile information.
- For the second question, they fit data to a logistic model using a least angle regression approach to find that activity was predicted by (i) how often a user was reached out to by others, (ii) frequency of third party application use, and (iii) how forthcoming a user was on the site. `http://www.dataspora.com/2009/02/predictive-analytics-using-r/`

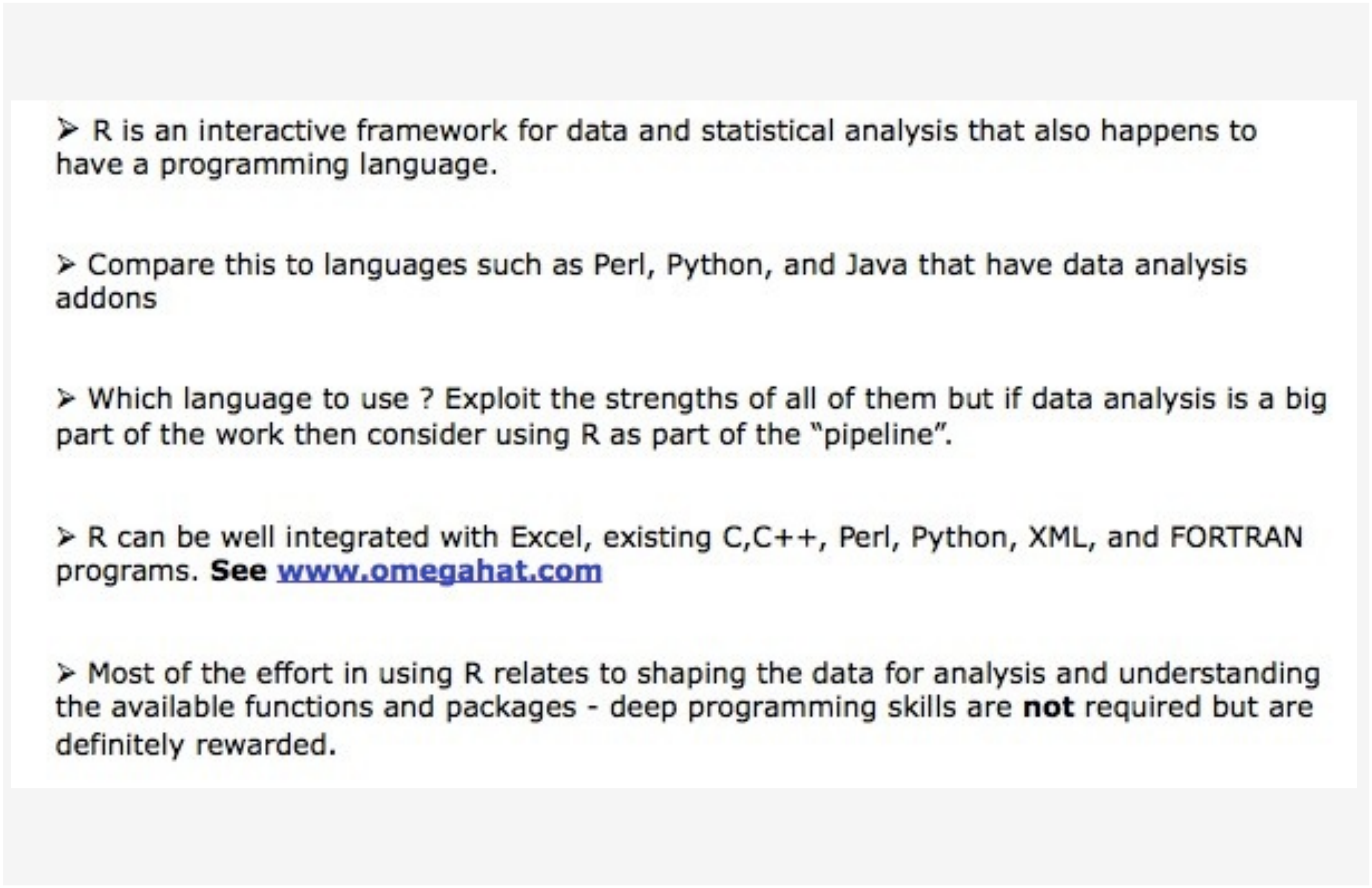
Introduction - Motivations



- Google determines the effectiveness of display ads for its customers. Using observational data from more than 10 million web users, Google compares the search behavior of people who were exposed to the display ad (i.e. those that never visited a web page displaying the ad) to similar users who did see the ad, to figure out how many additional people visit the advertiser's web site as a result of seeing the display ad.
- Conditional regression models are used to evaluate the factors that lead to user satisfaction of Google products, such as when users are surveyed on satisfaction with search reports, or when users are asked to rate YouTube videos.

`http://blog.revolutionanalytics.com/2011/08/google-r-effective-ads.html`

Introduction - Motivations



- R is an interactive framework for data and statistical analysis that also happens to have a programming language.

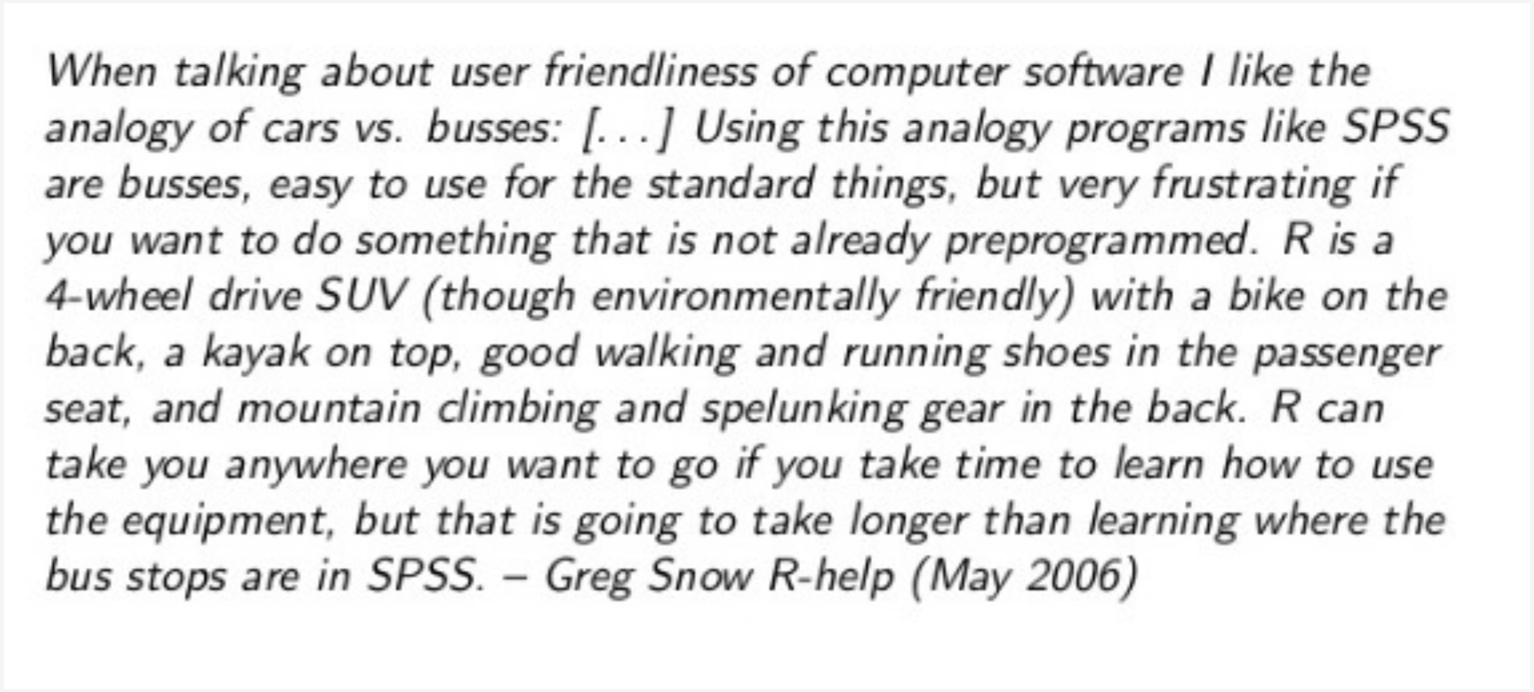
- Compare this to languages such as Perl, Python, and Java that have data analysis addons

- Which language to use ? Exploit the strengths of all of them but if data analysis is a big part of the work then consider using R as part of the "pipeline".

- R can be well integrated with Excel, existing C,C++, Perl, Python, XML, and FORTRAN programs. **See** www.omegahat.com

- Most of the effort in using R relates to shaping the data for analysis and understanding the available functions and packages - deep programming skills are **not** required but are definitely rewarded.

Introduction - Motivations

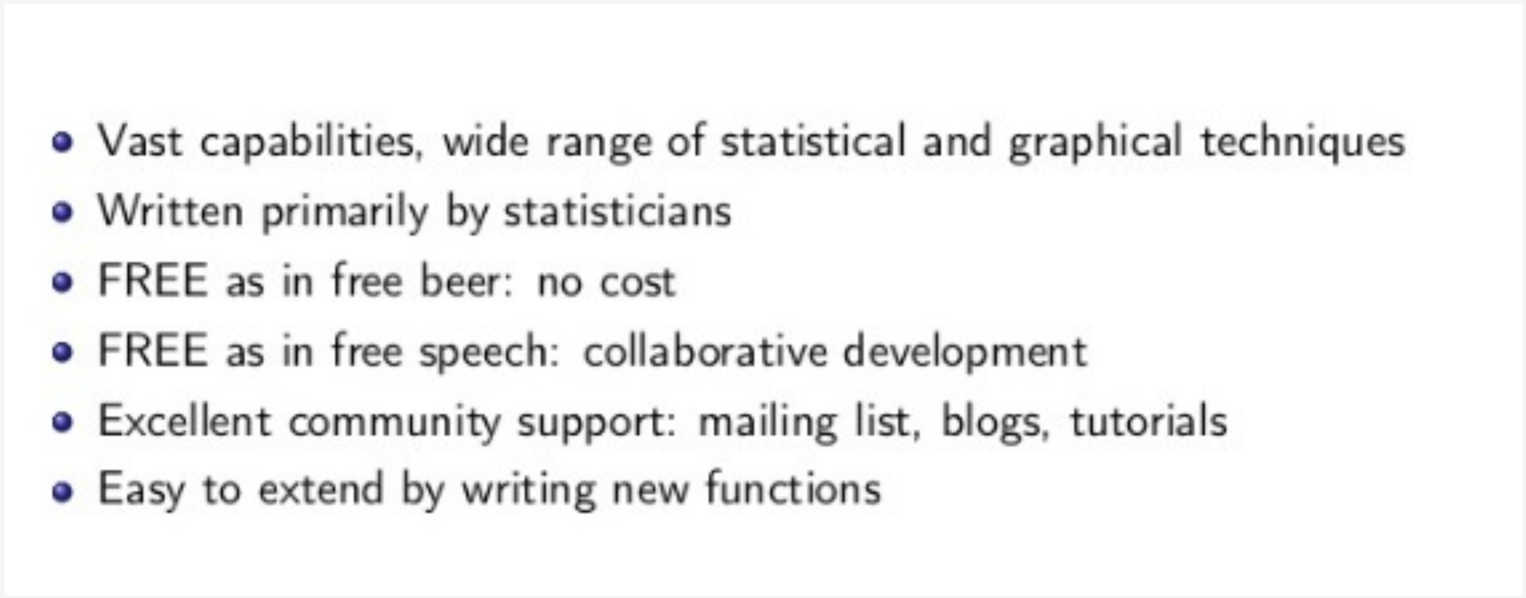


When talking about user friendliness of computer software I like the analogy of cars vs. busses: [. . .] Using this analogy programs like SPSS are busses, easy to use for the standard things, but very frustrating if you want to do something that is not already preprogrammed. R is a 4-wheel drive SUV (though environmentally friendly) with a bike on the back, a kayak on top, good walking and running shoes in the passenger seat, and mountain climbing and spelunking gear in the back. R can take you anywhere you want to go if you take time to learn how to use the equipment, but that is going to take longer than learning where the bus stops are in SPSS. – Greg Snow R-help (May 2006)

See Introduction to the R Statistical Computing Environment

http://www.slideshare.net/izahn/rintro

Introduction - Motivations

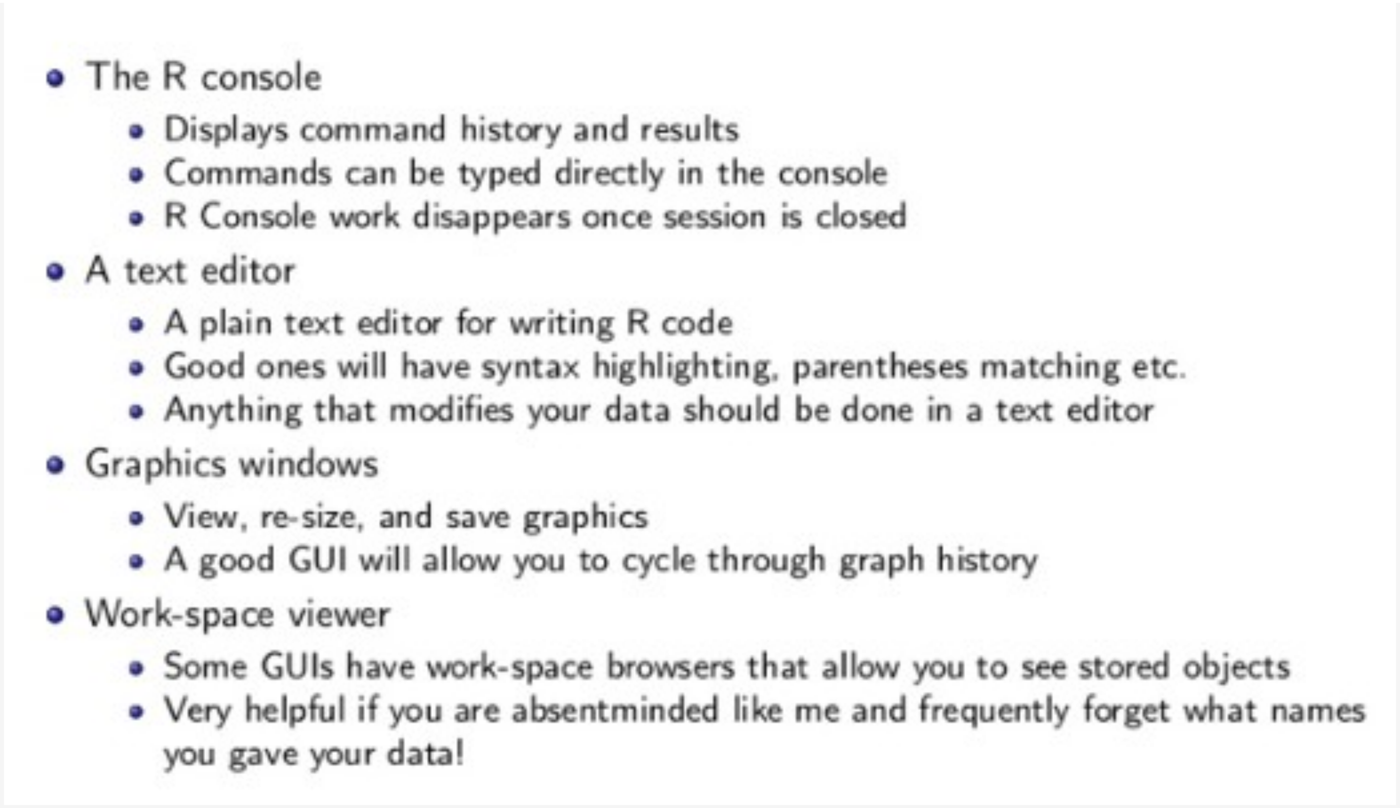


- Vast capabilities, wide range of statistical and graphical techniques
- Written primarily by statisticians
- FREE as in free beer: no cost
- FREE as in free speech: collaborative development
- Excellent community support: mailing list, blogs, tutorials
- Easy to extend by writing new functions

Taken from Introduction to the R Statistical Computing Environment

http://www.slideshare.net/izahn/rintro

Introduction - Motivations

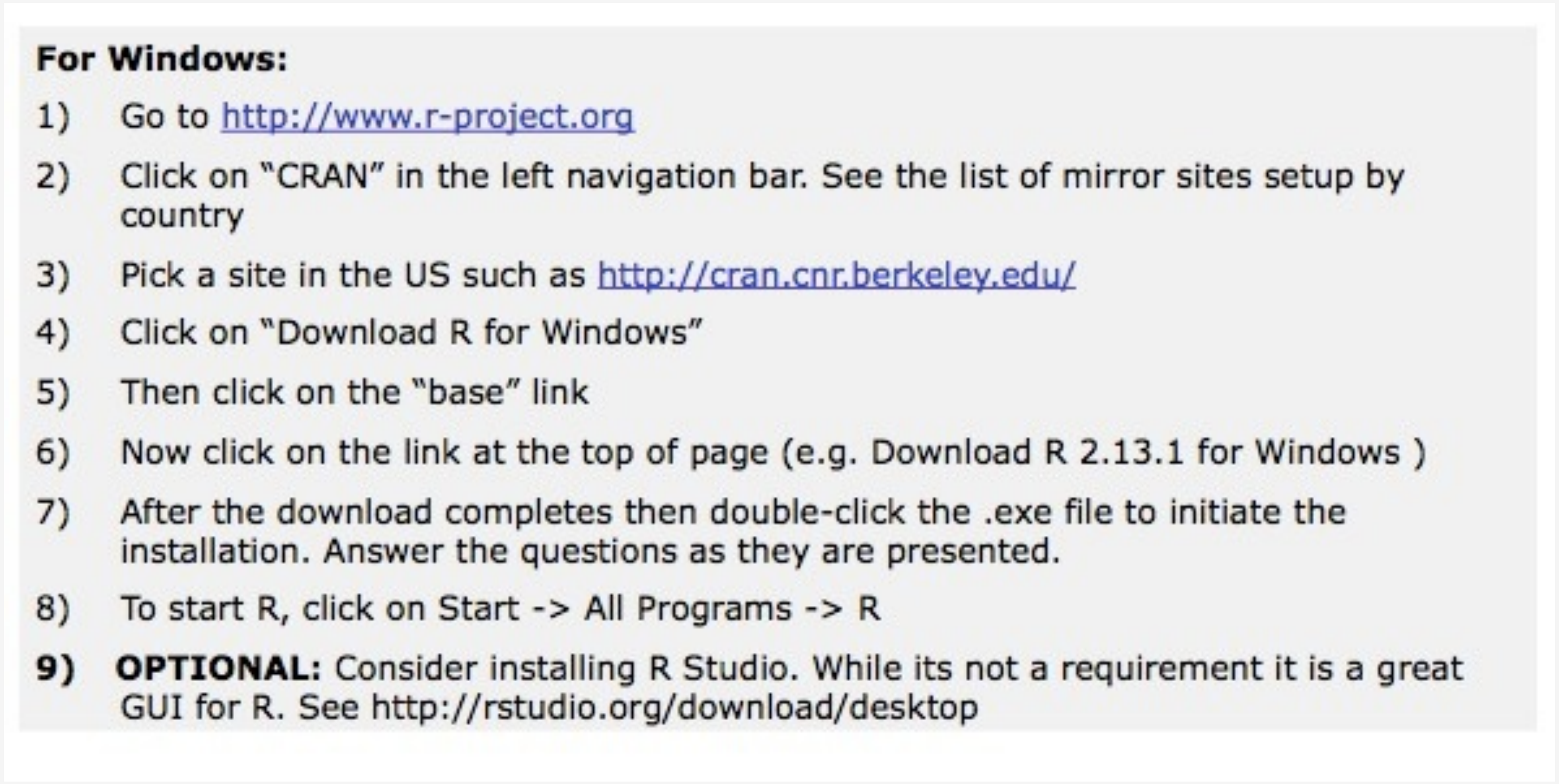


- The R console
  - Displays command history and results
  - Commands can be typed directly in the console
  - R Console work disappears once session is closed
- A text editor
  - A plain text editor for writing R code
  - Good ones will have syntax highlighting, parentheses matching etc.
  - Anything that modifies your data should be done in a text editor
- Graphics windows
  - View, re-size, and save graphics
  - A good GUI will allow you to cycle through graph history
- Work-space viewer
  - Some GUIs have work-space browsers that allow you to see stored objects
  - Very helpful if you are absentminded like me and frequently forget what names you gave your data!

Taken from Introduction to the R Statistical Computing Environment

http://www.slideshare.net/izahn/rintro

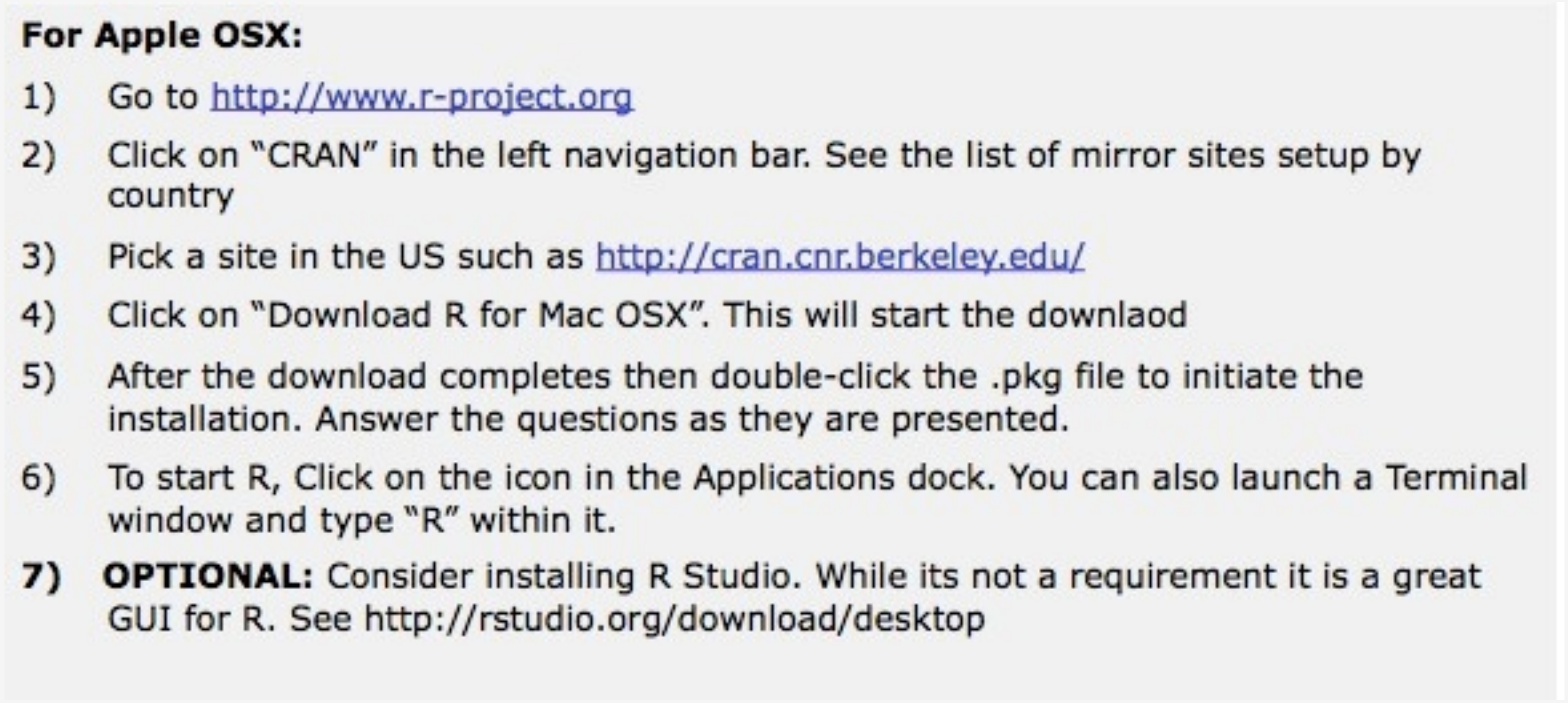
Installing



For Windows:

1) Go to http://www.r-project.org
2) Click on "CRAN" in the left navigation bar. See the list of mirror sites setup by country
3) Pick a site in the US such as http://cran.cnr.berkeley.edu/
4) Click on "Download R for Windows"
5) Then click on the "base" link
6) Now click on the link at the top of page (e.g. Download R 2.13.1 for Windows)
7) After the download completes then double-click the .exe file to initiate the installation. Answer the questions as they are presented.
8) To start R, click on Start -> All Programs -> R
9) **OPTIONAL:** Consider installing R Studio. While its not a requirement it is a great GUI for R. See http://rstudio.org/download/desktop

Installing



For Apple OSX:

1) Go to http://www.r-project.org
2) Click on "CRAN" in the left navigation bar. See the list of mirror sites setup by country
3) Pick a site in the US such as http://cran.cnr.berkeley.edu/
4) Click on "Download R for Mac OSX". This will start the downlaod
5) After the download completes then double-click the .pkg file to initiate the installation. Answer the questions as they are presented.
6) To start R, Click on the icon in the Applications dock. You can also launch a Terminal window and type "R" within it.
7) **OPTIONAL:** Consider installing R Studio. While its not a requirement it is a great GUI for R. See http://rstudio.org/download/desktop

Installing - Packages

It is important to note that R comes with a base set of packages as part of every install.

```
> getOption("defaultPackages")
[1] "datasets"  "utils"     "grDevices" "graphics"  "stats"     "methods"

> search()
 [1] ".GlobalEnv"   "package:lattice"   "package:stats" "package:graphics"
"package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods" "Autoloads"         "package:base"
```

While this might not look like a lot, each package has lots of useful stuff contained therein. If you wanted to drill down into one of these, here is one way to do it:

```
> library(help=”stats”)
```

Installing - Packages

```
> library(help=”stats”)
              Information on package ‘stats’

Description:

Package:       stats
Version:       2.15.1
Priority:      base
Title:         The R Stats Package
Author:        R Core Team and contributors worldwide
Maintainer:    R Core Team <R-core@r-project.org>
Description:   R statistical functions
License:       Part of R 2.15.1
Built:         R 2.15.1; universal-apple-darwin9.8.0; 2012-06-22 19:10:14 UTC; unix

Index:
.checkMFClasses         Functions to Check the Type of Variables passed
                        to Model Frames
AIC                     Akaike's An Information Criterion
ARMAacf                 Compute Theoretical ACF for an ARMA Process
ARMAtoMA                Convert ARMA Process to Infinite MA Process
Beta                    The Beta Distribution
Binomial                The Binomial Distribution
Box.test                Box-Pierce and Ljung-Box Tests
```

Installing - Packages

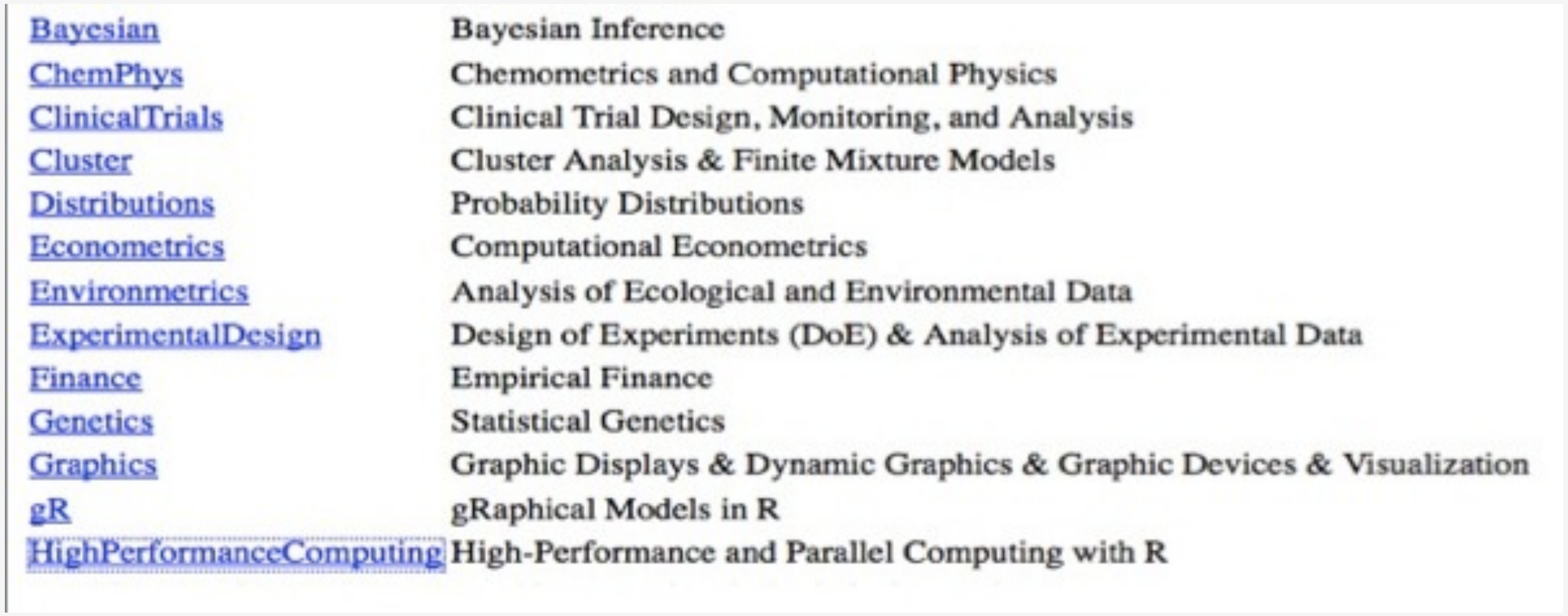
Many packages also come with test and example data sets that can be very useful when attempting to perfect your knowledge of the various functions. To see what test data sets are available in a give package, do something like:

```
> search()
 [1] ".GlobalEnv"        "package:lattice"   "package:stats"
"package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
"package:base"

> data(package="stats")

Data sets in package ‘datasets’:

AirPassengers                          Monthly Airline Passenger Numbers
1949-1960
BJsales                                Sales Data with Leading Indicator
BJsales.lead (BJsales)                 Sales Data with Leading Indicator
BOD                                    Biochemical Oxygen Demand
CO2                                    Carbon Dioxide Uptake in Grass Plants
DNase                                  Elisa assay of DNase
EuStockMarkets                         Daily Closing Prices of Major European
Stock Indice
..
```

Installing - CRAN

While the base R package has many cool functions, you will invariably want to install many of the add-on packages available from CRAN.

To obtain information on the wide variety of packages available for then visit http://cran.cnr.berkeley.edu Here are some of the available package categories. Use the "search" function to drill down:



Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
gR	gRaphical Models in R
HighPerformanceComputing	High-Performance and Parallel Computing with R

Installing - CRAN

If you are using an installation of R on Windows you can use a GUI to help you identify and install packages. However, if you are using a straight terminal interface such as when on Linux or OSX Terminal then you will need to something like the following.

Let's say that we have decided that we need the capabilities offered by the "actuar" function on CRAN. This package relates to Actuarial functions. To install it, start R.

```
> install.packages("actuar",dependencies=TRUE)
--- Please select a CRAN mirror for use in this session ---
Loading Tcl/Tk interface ... done

trying URL 'http://mirrors.nics.utk.edu/cran/bin/macosx/leopard/contrib/2.15/
actuar_1.1-5.tgz'
Content type 'application/x-gzip' length 1837121 bytes (1.8 Mb)
opened URL
==================================================
downloaded 1.8 Mb

> library(actuar)   # Brings the package into the workspace
```

Installing - CRAN

If you are using an installation of R on Windows you can use a GUI to help you identify and install packages. However, if you are using a straight terminal interface such as when on Linux or OSX Terminal then you will need to something like the following.

```
> library(actuar)   # Brings the package into the workspace


> search()
 [1] ".GlobalEnv"        "package:actuar"    "package:lattice"
"package:stats"     "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"  "package:methods"
"Autoloads"         "package:base"
```

Installing - CRAN

On occasion, (not often), you will need to install a package from a specific repository. For example, the OmegaHat project at www.omeghat.org has many interesting add-on packages that aren't necessarily linked into CRAN. To install one of these packages you would do something like:

```
> install.packages("GeoIP", repos = "http://www.omegahat.org/R")
```

On Linux and OSX it is possible, (though unlikely), that you will have to install a package using the "tar/gzipped" version of the file. This might happen if a colleague has written a package that hasn't been registered on CRAN. To do the install you do something like:

```
$ R CMD INSTALL GeoIP.tar.gz
```

It is also possible that the above *might* trigger a permissions error in which case you will have to use the "sudo" command or get the administrator of your system to help.

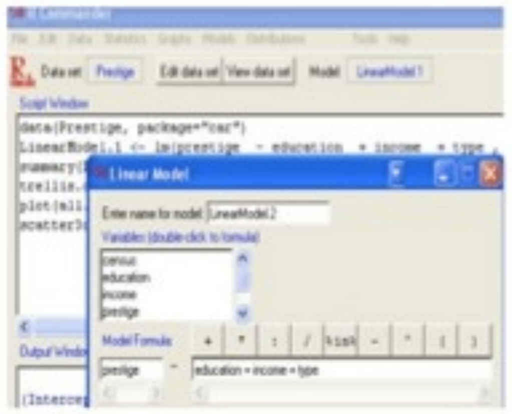
```
$ sudo R CMD INSTALL GeoIP.tar.gz
```

GUI ?

R is provided with a command line interface (CLI), which is the preferred user interface for power users because it allows direct control on calculations and is flexible. However, good knowledge of the language is required.

Because of this, the CLI can be intimidating for beginners. Thankfully, there are a number of R GUIs out there to make your life easier: R Commander, R-Studio, JGR, SciViews-R, R.app (OSX), Tinn-R (Windows only), ECLIPSE with STATET, EMACs

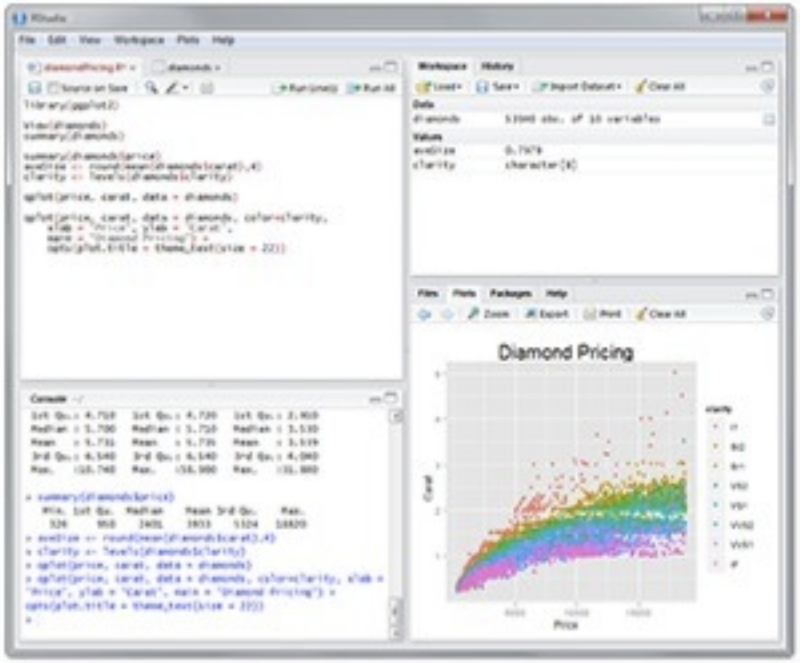
Note that on Windows you get default GUI that provides you with a reasonable amount of menu-drive capability. However, you could also use another GUI if you so desired.



```
data(Prestige, package="car")
LinearModel.1 <- lm(prestige ~ education + income + type
```

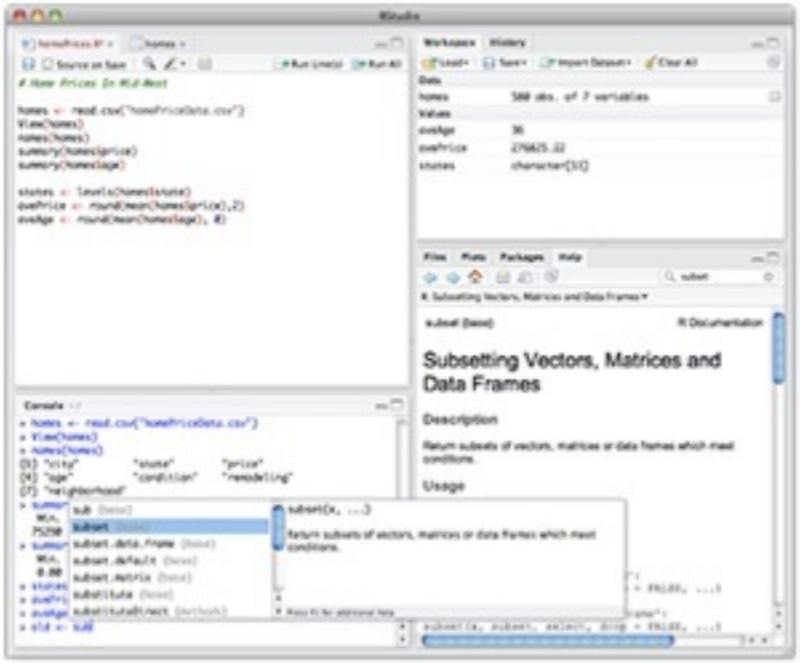
GUI ?

I like to use R-Studio. See http://rstudio.org



```
library(ggplot2)

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat),4)
clarity <- levels(diamonds$clarity)

qplot(price, carat, data = diamonds)

qplot(price, carat, data = diamonds, color=clarity,
    xlab = "Price", ylab = "Carat",
    main = "Diamond Pricing") +
    opts(plot.title = theme_text(size = 22))
```

Windows



```
# Home Prices In Mid-West

homes <- read.csv("homePriceData.csv")
View(homes)
names(homes)
summary(homes$price)
summary(homes$age)

states <- levels(homes$state)
avePrice <- round(mean(homes$price),2)
aveAge <- round(mean(homes$age), 0)
```

Subsetting Vectors, Matrices and Data Frames

Description

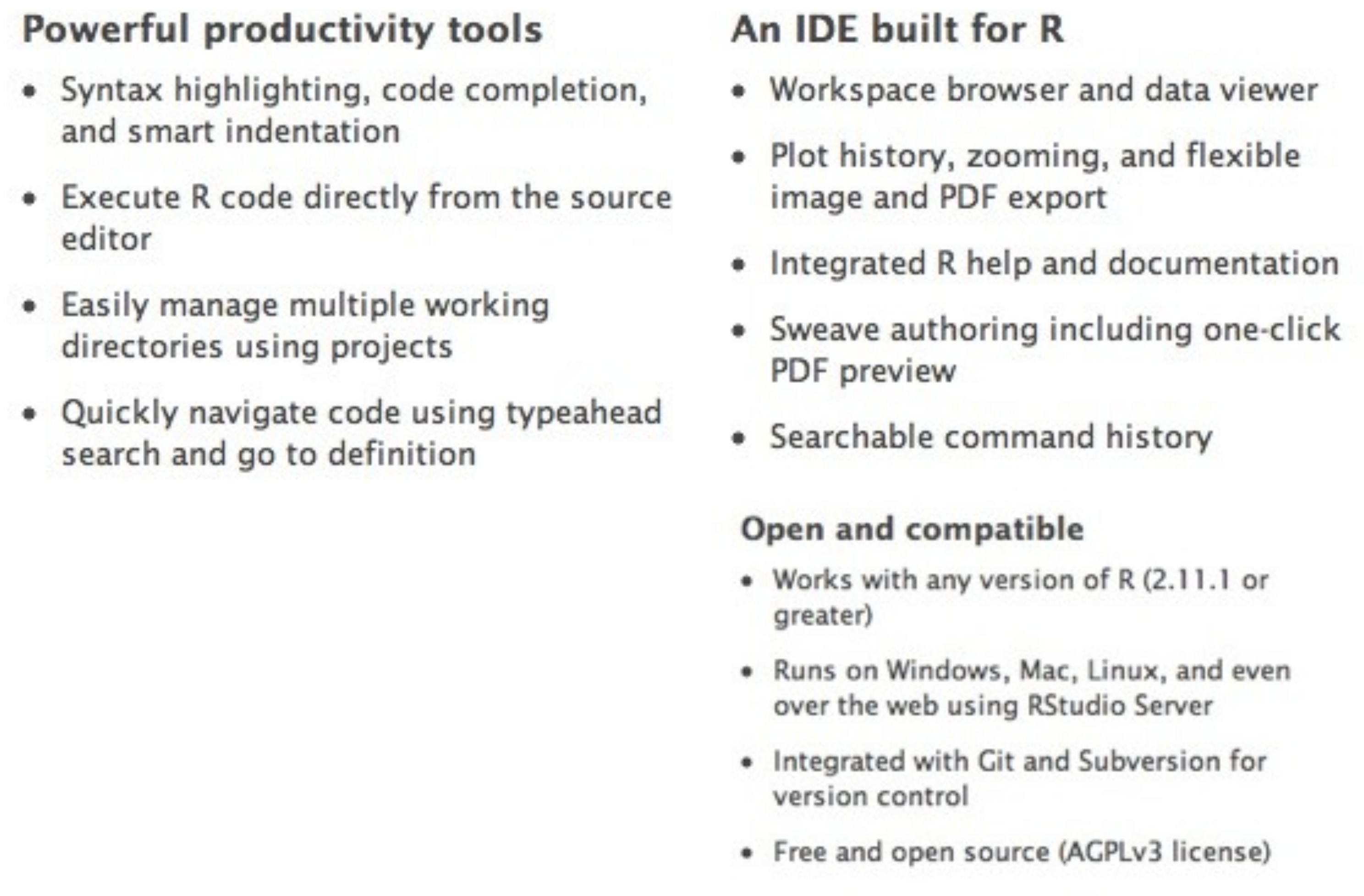
Return subsets of vectors, matrices or data frames which meet conditions.

Usage

Mac OS X

GUI ?

I like to use R-Studio. See http://rstudio.org



Powerful productivity tools

- Syntax highlighting, code completion, and smart indentation
- Execute R code directly from the source editor
- Easily manage multiple working directories using projects
- Quickly navigate code using typeahead search and go to definition

An IDE built for R

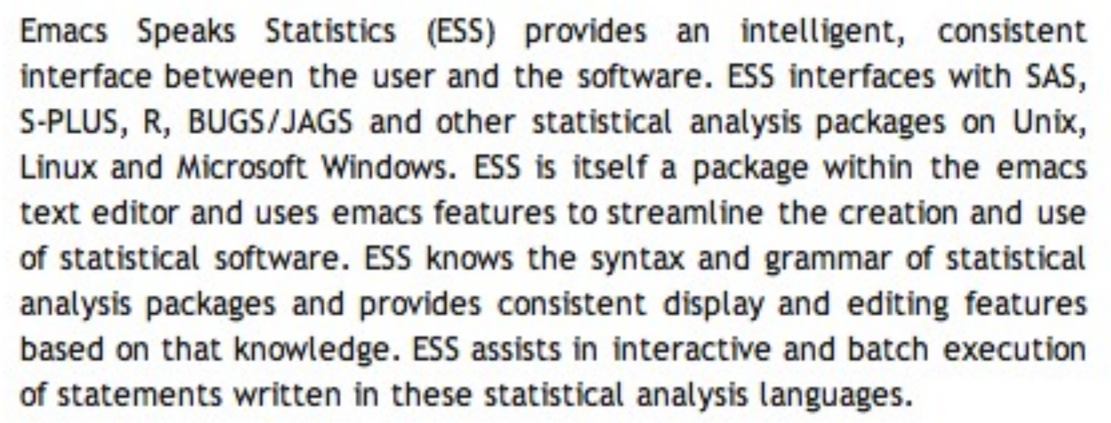
- Workspace browser and data viewer
- Plot history, zooming, and flexible image and PDF export
- Integrated R help and documentation
- Sweave authoring including one-click PDF preview
- Searchable command history

Open and compatible

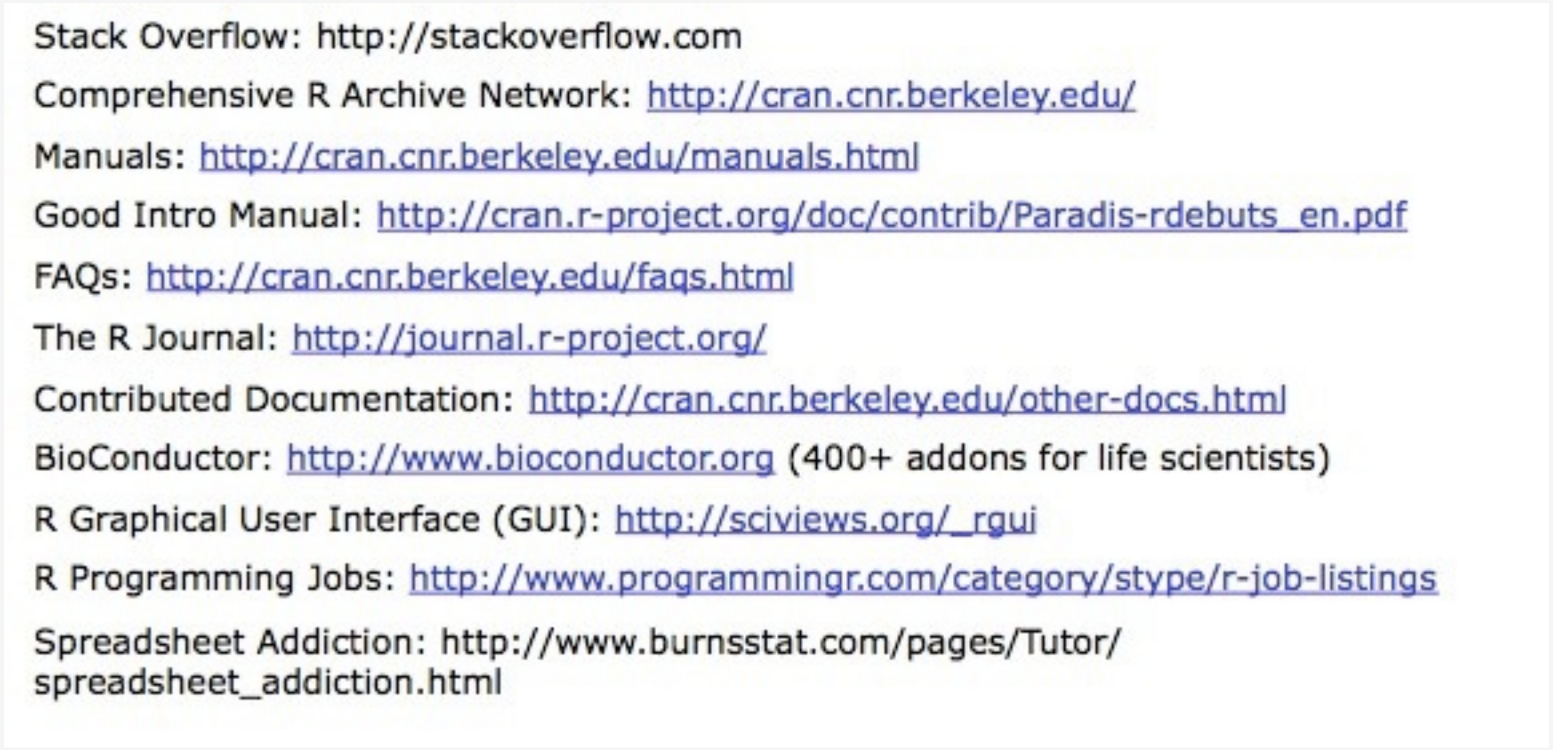
- Works with any version of R (2.11.1 or greater)
- Runs on Windows, Mac, Linux, and even over the web using RStudio Server
- Integrated with Git and Subversion for version control
- Free and open source (AGPLv3 license)

GUI ?

If you know EMACS then you will probably want to use ESS. See http://ess.r-project.org/index.php?Section=home


Emacs Speaks Statistics (ESS) provides an intelligent, consistent interface between the user and the software. ESS interfaces with SAS, S-PLUS, R, BUGS/JAGS and other statistical analysis packages on Unix, Linux and Microsoft Windows. ESS is itself a package within the emacs text editor and uses emacs features to streamline the creation and use of statistical software. ESS knows the syntax and grammar of statistical analysis packages and provides consistent display and editing features based on that knowledge. ESS assists in interactive and batch execution of statements written in these statistical analysis languages.

Finding Documentation for R



Stack Overflow: http://stackoverflow.com

Comprehensive R Archive Network: http://cran.cnr.berkeley.edu/

Manuals: http://cran.cnr.berkeley.edu/manuals.html

Good Intro Manual: http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf

FAQs: http://cran.cnr.berkeley.edu/faqs.html

The R Journal: http://journal.r-project.org/

Contributed Documentation: http://cran.cnr.berkeley.edu/other-docs.html

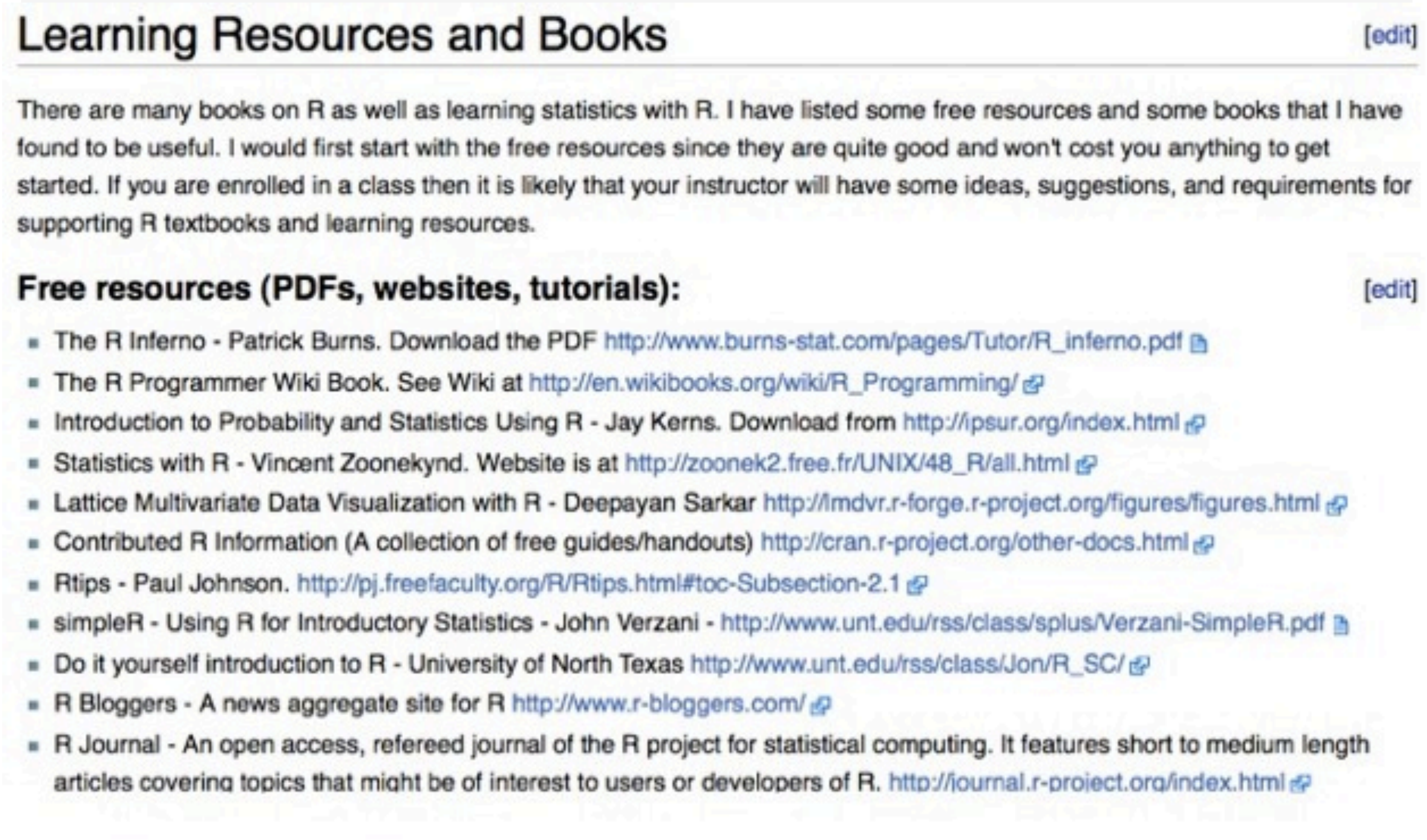
BioConductor: http://www.bioconductor.org (400+ addons for life scientists)

R Graphical User Interface (GUI): http://sciviews.org/_rgui

R Programming Jobs: http://www.programmingr.com/category/stype/r-job-listings

Spreadsheet Addiction: http://www.burnsstat.com/pages/Tutor/
spreadsheet_addiction.html

Finding Documentation for R



Learning Resources and Books

[edit]

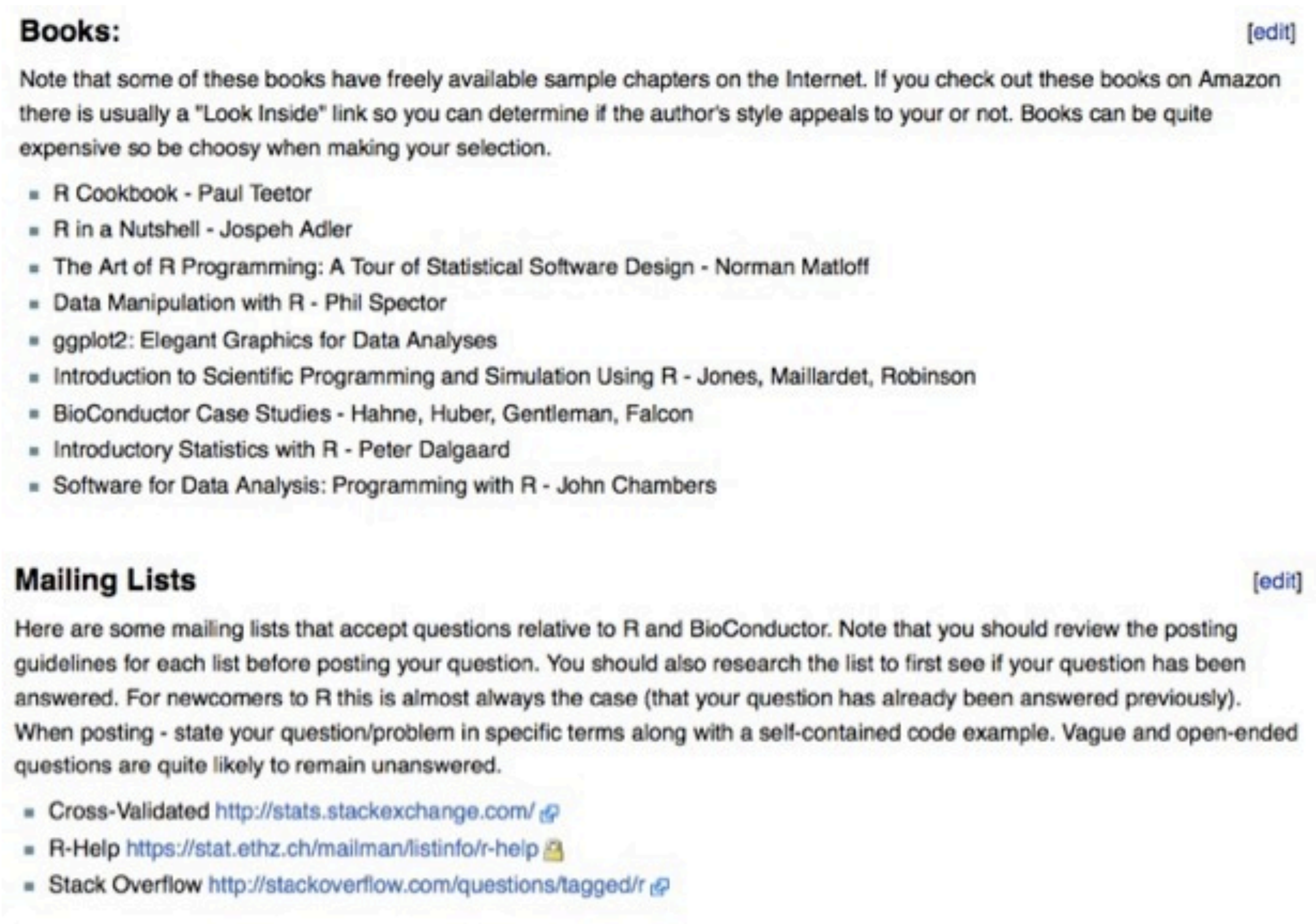
There are many books on R as well as learning statistics with R. I have listed some free resources and some books that I have found to be useful. I would first start with the free resources since they are quite good and won't cost you anything to get started. If you are enrolled in a class then it is likely that your instructor will have some ideas, suggestions, and requirements for supporting R textbooks and learning resources.

Free resources (PDFs, websites, tutorials):

[edit]

- The R Inferno - Patrick Burns. Download the PDF http://www.burns-stat.com/pages/Tutor/R_inferno.pdf
- The R Programmer Wiki Book. See Wiki at http://en.wikibooks.org/wiki/R_Programming/
- Introduction to Probability and Statistics Using R - Jay Kerns. Download from http://ipsur.org/index.html
- Statistics with R - Vincent Zoonekynd. Website is at http://zoonek2.free.fr/UNIX/48_R/all.html
- Lattice Multivariate Data Visualization with R - Deepayan Sarkar http://lmdvr.r-forge.r-project.org/figures/figures.html
- Contributed R Information (A collection of free guides/handouts) http://cran.r-project.org/other-docs.html
- Rtips - Paul Johnson. http://pj.freefaculty.org/R/Rtips.html#toc-Subsection-2.1
- simpleR - Using R for Introductory Statistics - John Verzani - http://www.unt.edu/rss/class/splus/Verzani-SimpleR.pdf
- Do it yourself introduction to R - University of North Texas http://www.unt.edu/rss/class/Jon/R_SC/
- R Bloggers - A news aggregate site for R http://www.r-bloggers.com/
- R Journal - An open access, refereed journal of the R project for statistical computing. It features short to medium length articles covering topics that might be of interest to users or developers of R. http://journal.r-project.org/index.html

Finding Documentation for R



Books: [edit]

Note that some of these books have freely available sample chapters on the Internet. If you check out these books on Amazon there is usually a "Look Inside" link so you can determine if the author's style appeals to your or not. Books can be quite expensive so be choosy when making your selection.

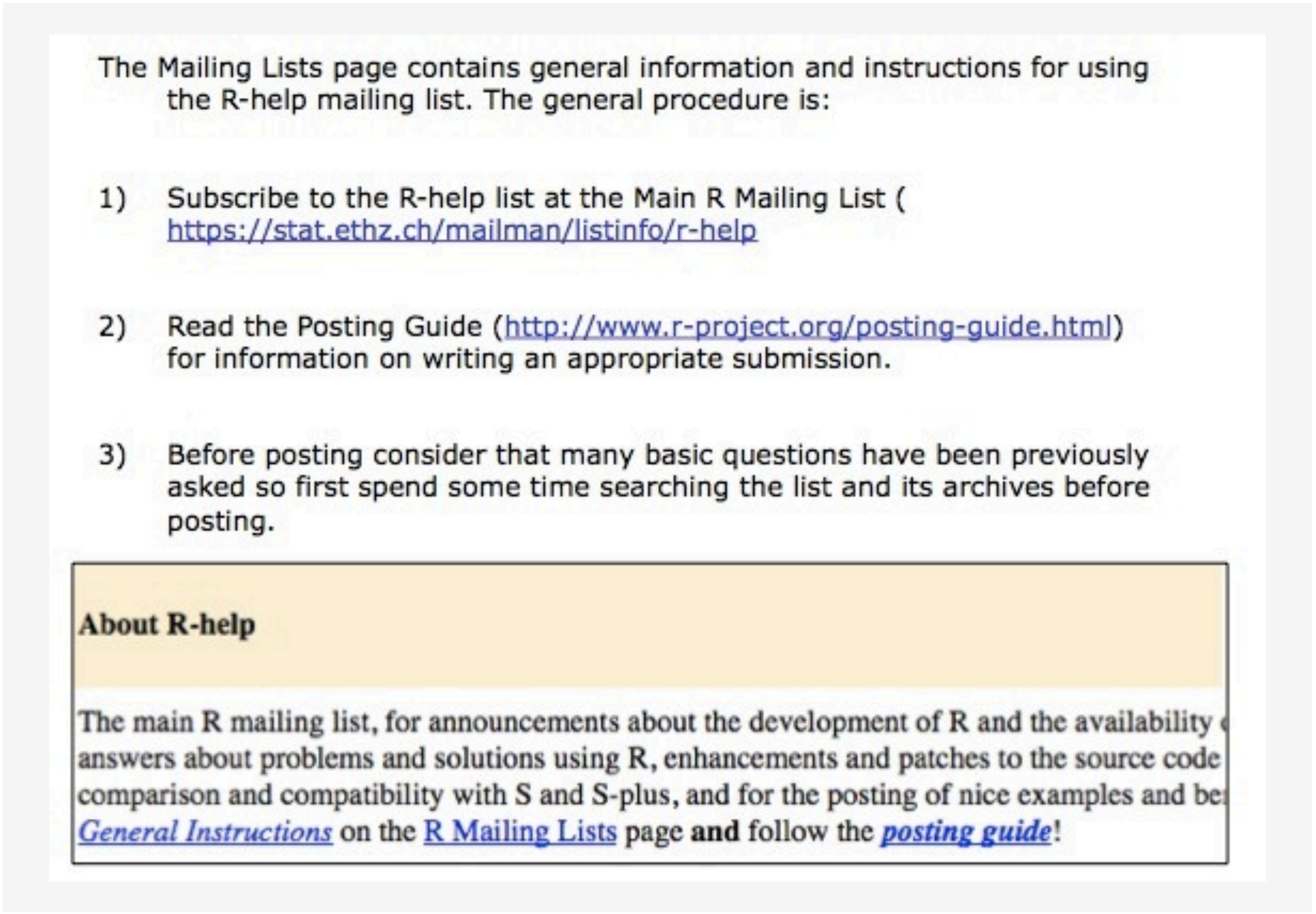
- R Cookbook - Paul Teetor
- R in a Nutshell - Jospeh Adler
- The Art of R Programming: A Tour of Statistical Software Design - Norman Matloff
- Data Manipulation with R - Phil Spector
- ggplot2: Elegant Graphics for Data Analyses
- Introduction to Scientific Programming and Simulation Using R - Jones, Maillardet, Robinson
- BioConductor Case Studies - Hahne, Huber, Gentleman, Falcon
- Introductory Statistics with R - Peter Dalgaard
- Software for Data Analysis: Programming with R - John Chambers

Mailing Lists [edit]

Here are some mailing lists that accept questions relative to R and BioConductor. Note that you should review the posting guidelines for each list before posting your question. You should also research the list to first see if your question has been answered. For newcomers to R this is almost always the case (that your question has already been answered previously). When posting - state your question/problem in specific terms along with a self-contained code example. Vague and open-ended questions are quite likely to remain unanswered.

- Cross-Validated http://stats.stackexchange.com/
- R-Help https://stat.ethz.ch/mailman/listinfo/r-help
- Stack Overflow http://stackoverflow.com/questions/tagged/r

Finding Documentation for R



The Mailing Lists page contains general information and instructions for using the R-help mailing list. The general procedure is:

1) Subscribe to the R-help list at the Main R Mailing List (https://stat.ethz.ch/mailman/listinfo/r-help

2) Read the Posting Guide (http://www.r-project.org/posting-guide.html) for information on writing an appropriate submission.

3) Before posting consider that many basic questions have been previously asked so first spend some time searching the list and its archives before posting.

About R-help

The main R mailing list, for announcements about the development of R and the availability answers about problems and solutions using R, enhancements and patches to the source code comparison and compatibility with S and S-plus, and for the posting of nice examples and be *General Instructions* on the R Mailing Lists page **and** follow the ***posting guide***!

Finding Documentation for R

Style Guides for R Programming

Ever try reading code written by someone else ?

Take some "style" hints from the big guys: See "Google's R Style Guide" and Hadley's Style Guide at:

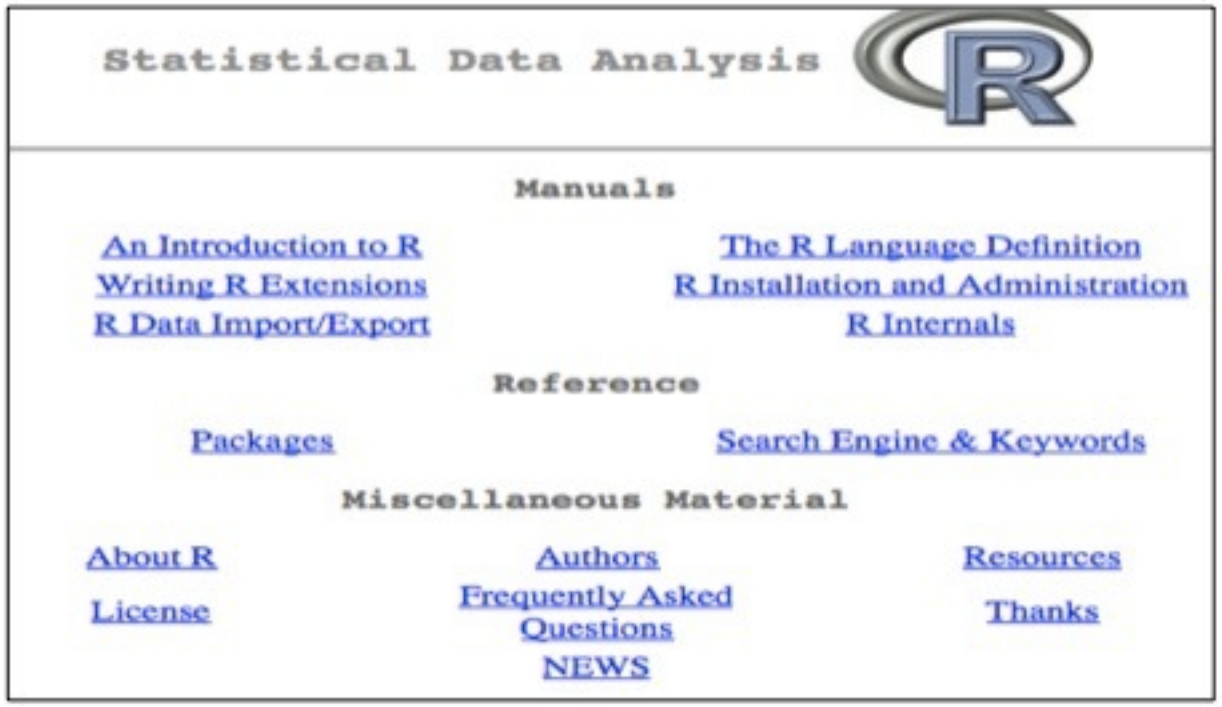
http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html

https://github.com/hadley/devtools/wiki/Style

Workspace - Getting Help

R has built-in assistance that can be very useful. Let’s take a look

```
> help.start()    # Launches a web browser with a table of contents.
```



Statistical Data Analysis

Manuals

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

Reference

Packages
Search Engine & Keywords

Miscellaneous Material

About R
License
Authors
Frequently Asked Questions
NEWS
Resources
Thanks

Workspace - Getting Help

R has built-in assistance that can be very useful. Let's take a look

```
help(function_name)   # Get help on function

?function_name        # Equivalent to the above

args(function_name)   # See what arguments the function accepts

example(function_name)    # See an example

?mean

example(mean)

mean> x <- c(0:10, 50)

mean> xm <- mean(x)

mean> c(xm, mean(x, trim = 0.10))
[1] 8.75 5.50

mean> mean(USArrests, trim = 0.2)
  Murder  Assault UrbanPop 
    7.42   167.60    66.20 
```

Workspace - Getting Help

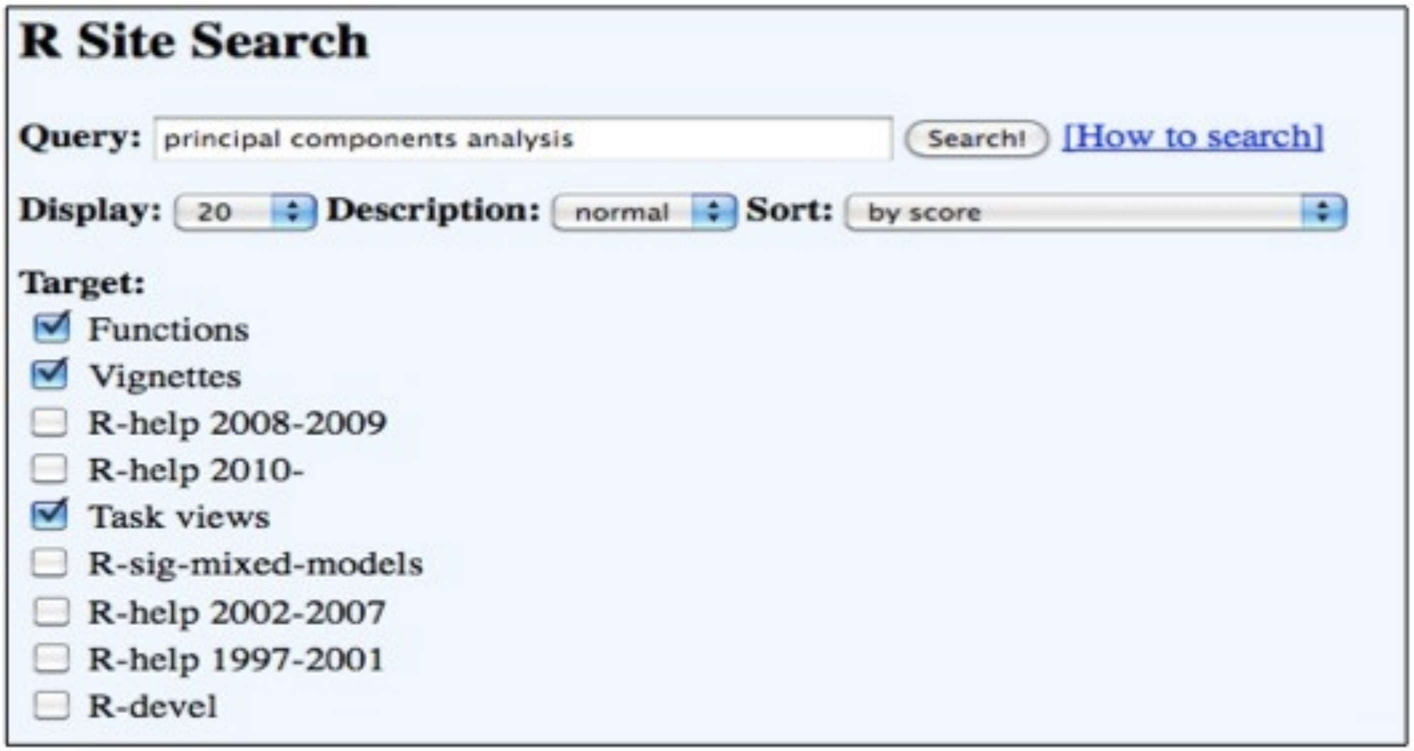
R has built-in assistance that can be very useful. Let's take a look

```
help.search("time series")

??"time series"
Help files with alias or concept or title matching 'time series' using
fuzzy matching:

boot::tsboot            Bootstrapping of Time Series
car::Hartnagel          Canadian Crime-Rates Time Series
datasets::austres       Quarterly Time Series of the Number of
                        Australian Residents
datasets::beavers       Body Temperature Series of Two Beavers
ggplot2::economics      US economic time series
```

Workspace - Getting Help

R has built-in assistance that can be very useful. Let's take a look

```
> RSiteSearch("correlation")
A search query has been submitted to http://search.r-project.org
The results page should open in your browser shortly
```



R Site Search

Query: principal components analysis [Search!] [How to search]

Display: 20 Description: normal Sort: by score

Target:

- [✓] Functions
- [✓] Vignettes
- [ ] R-help 2008-2009
- [ ] R-help 2010-
- [✓] Task views
- [ ] R-sig-mixed-models
- [ ] R-help 2002-2007
- [ ] R-help 1997-2001
- [ ] R-devel

R vs. SAS vs. MATLAB - Clash of the Titans

Let the debate begin….. But there are no easy answers here.
There are lots of good things about all three.

R

* Characteristics of R - R Journal, vol. 1/1 (journal.r-project)

1) Interactive language for data analysis and programming

2) Uses the functional programming model

3) Object-oriented

4) Modular - Accommodates add-on packages

5) Collaborative, Open source projects

* Everything that is in R is an object

* Everything that happens in R is a function call.

R

* The R main program is a classical read-eval-print loop
 This basically means that you get a command prompt at which to enter things:

```
>
```

Three things to keep in mind:

1) The GREAT thing about R is that there are many different ways to do things.

2) The WORST thing about R is that there are many different ways to do things

3) To be a good programmer in R one must first be a knowledgeable user of R.

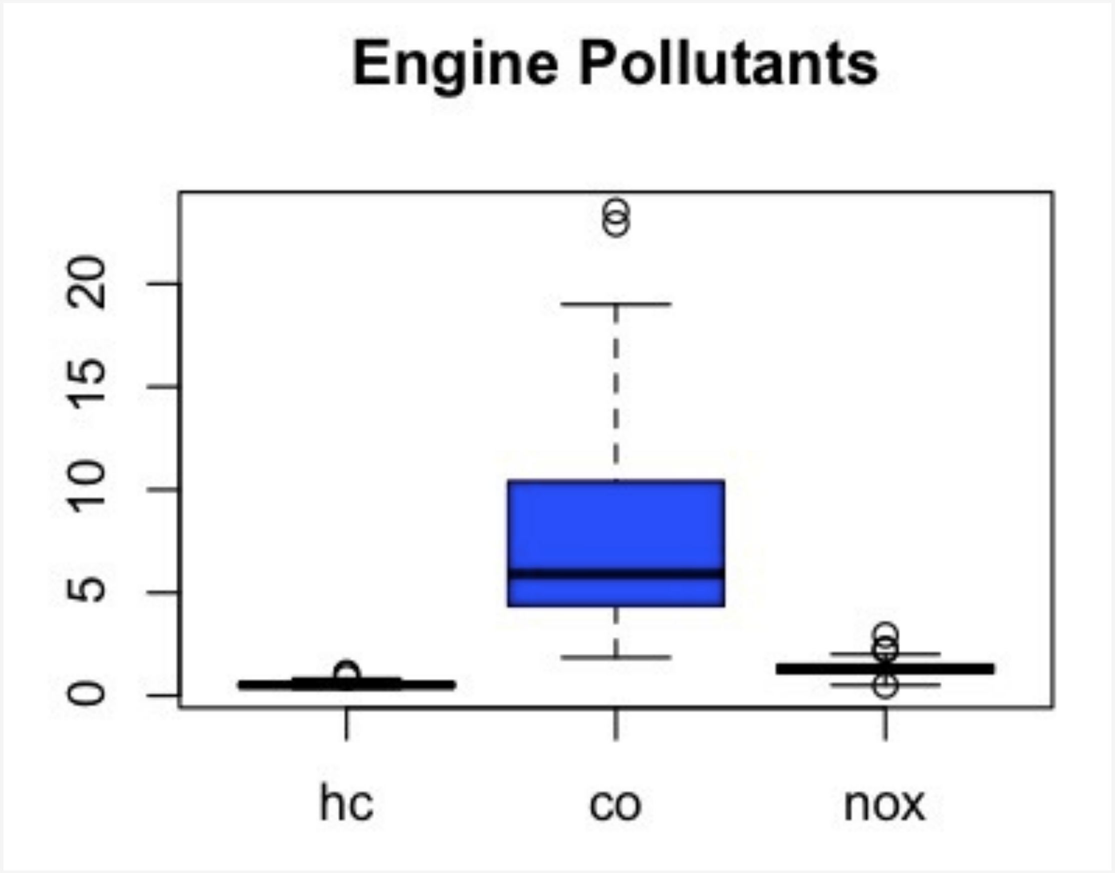
R - A Walkthrough

```
url = "http://www.bimcore.emory.edu/R/CSV.DIR/table_7_3.csv"
engine = read.table(url, sep = ",", header=TRUE)

engine=engine[,-1]

head(engine)         # 3 engine pollutants
    hc    co  nox
1 0.50  5.01 1.28
2 0.65 14.67 0.72
3 0.46  8.60 1.17
4 0.41  4.42 1.31
5 0.41  4.95 1.16

> summary(engine)
       en               hc                co               nox       
 Min.   : 1.00   Min.   :0.3400   Min.   : 1.850   Min.   :0.490  
 1st Qu.:12.75   1st Qu.:0.4375   1st Qu.: 4.388   1st Qu.:1.110  
 Median :24.50   Median :0.5100   Median : 5.905   Median :1.315  
 Mean   :24.00   Mean   :0.5502   Mean   : 7.879   Mean   :1.340  
 3rd Qu.:35.25   3rd Qu.:0.6025   3rd Qu.:10.015   3rd Qu.:1.495  
 Max.   :46.00   Max.   :1.1000   Max.   :23.530   Max.   :2.940  
```

http://www.cyclismo.org/tutorial/R/cholesterol.html

R - A Walkthrough

```
# Check out the data

boxplot(engine)
```



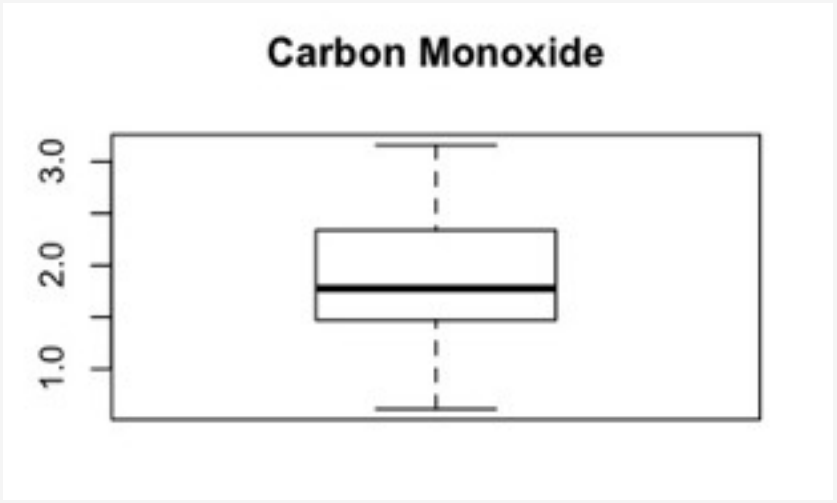
http://www.cyclismo.org/tutorial/R/cholesterol.html

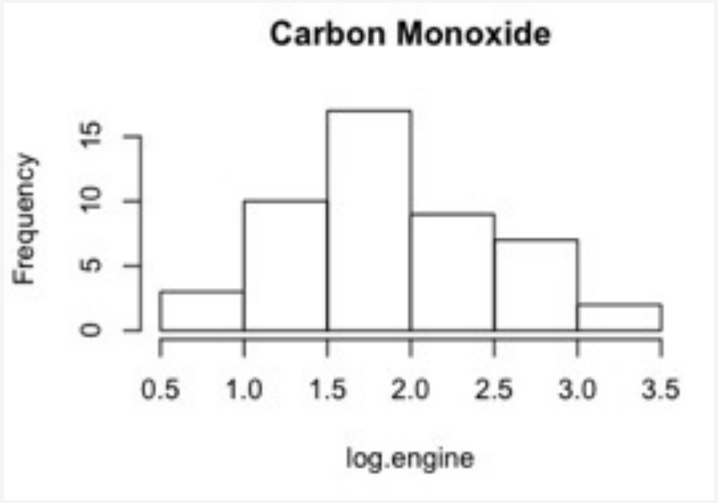
R - A Walkthrough

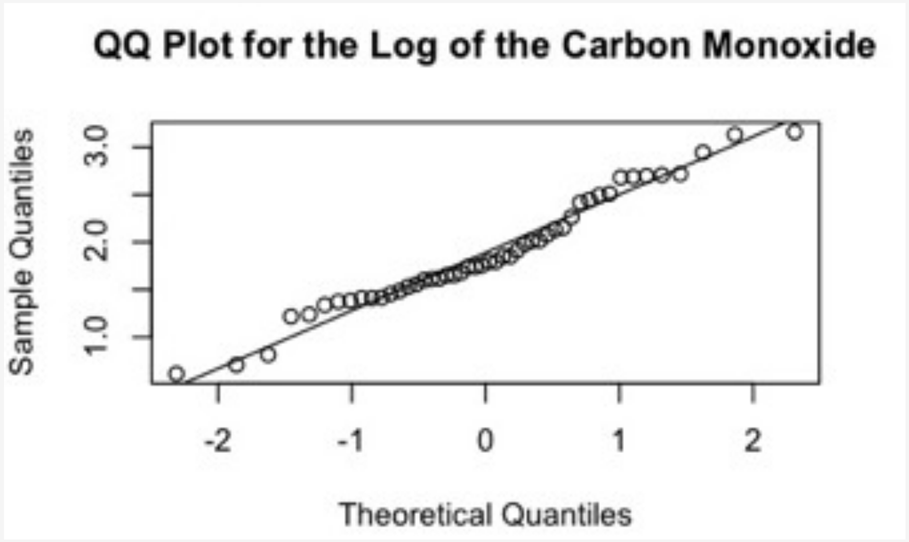
```
# Plot some helpful graphs

par(mfrow=c(1,3))

boxplot(engine$co,main="Carbon Monoxide")

hist(engine$co)

qqnorm(engine$co,main="Carbon Monoxide")

qqline(engine$co)
```

http://www.cyclismo.org/tutorial/R/cholesterol.html

R - A Walkthrough







R - A Walkthrough

```
# Null hypothesis is that the data is normal

shapiro.test(engine$co)

     Shapiro-Wilk normality test

data:  engine$co
W = 0.8357, p-value = 9.289e-06


# Take the log of the CO


log.engine = log(engine$co)

shapiro.test(log.engine)

     Shapiro-Wilk normality test

data:  log.engine
W = 0.9693, p-value = 0.2379
```

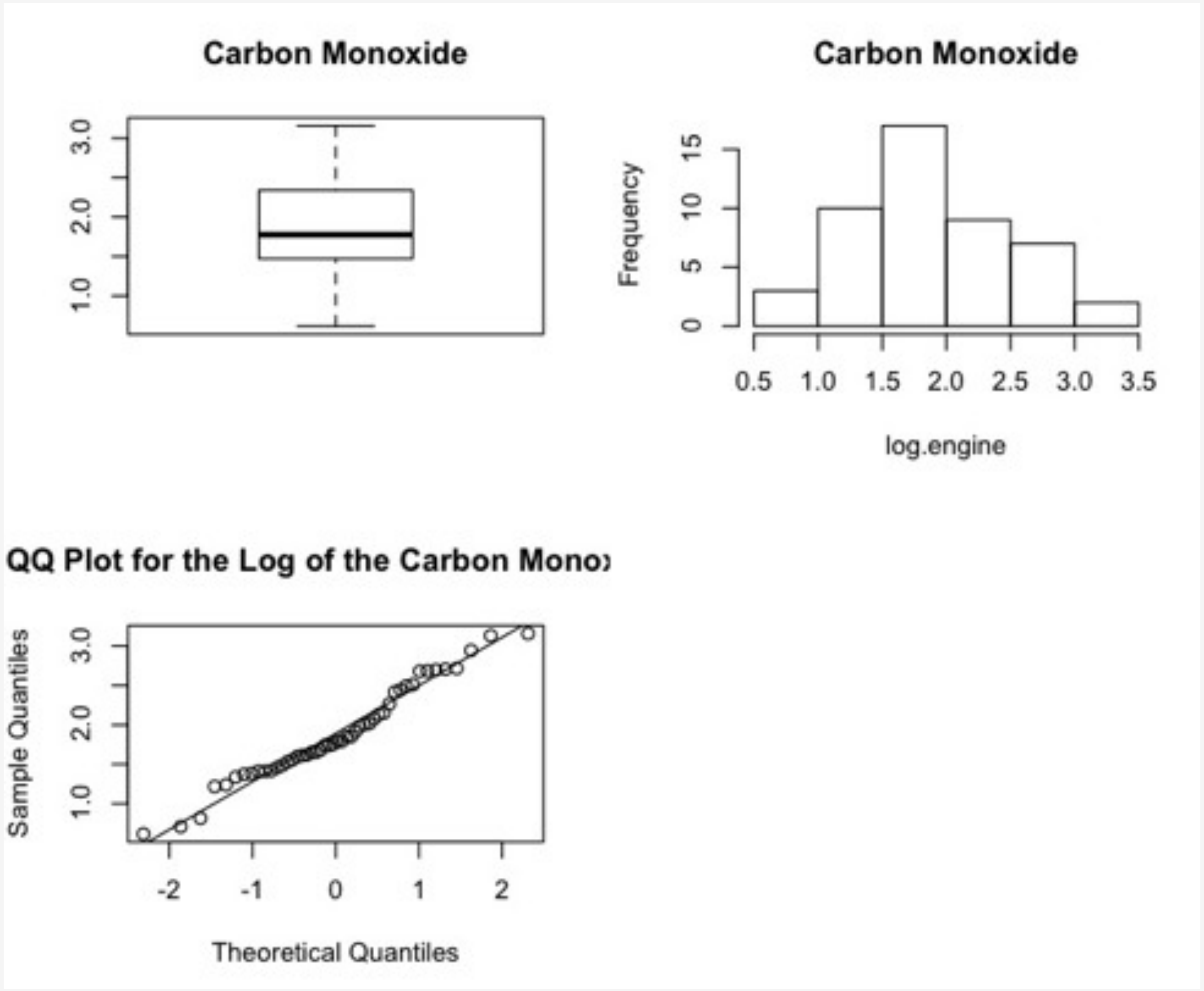
http://www.cyclismo.org/tutorial/R/cholesterol.html

R - A Walkthrough

```
par(mfrow=c(2,2))

log.engine = log(engine$co)

boxplot(log.engine,main="Carbon Monoxide")

hist(log.engine,main="Carbon Monoxide")

qqnorm(log.engine,main="QQ Plot for the Log of the Carbon Monoxide")

qqline(log.engine)
```

http://www.cyclismo.org/tutorial/R/cholesterol.html

R - A Walkthrough



R - A Walkthrough

```
# Let's build a confidence interval

my.mean = mean(log.engine)

my.sd = sd(log.engine)

n = length(log.engine)

# Get standard error

se = my.sd/sqrt(n)

error = se*qt(0.975,df=n-1)

left = my.mean - error

right = my.mean + error

c(left,right)
[1] 1.709925 2.057431

c(exp(left),exp(right))
[1] 5.528548 7.825840
```

R - A Walkthrough

```
# Test H0: mu  = 5.4
#      HA:mu  != 5.4

lNull <- log(5.4) - error

rNull <- log(5.4) + error

c(lNull,rNull)
[1] 1.512646 1.860152

my.mean
[1] 1.883678
```

So the mean is outside the range so we reject the null hypothesis. There is a low probability that we would have obtained our sample mean if the true mean really were 5.4.

http://www.cyclismo.org/tutorial/R/cholesterol.html

R - A Walkthrough

```
# We could have calculated a p-value by hand

p.val = 2*(1-pt((my.mean-log(5.4))/se,df=n-1))

> p.val
[1] 0.02692539

# But its easier to call a procedure to do it all !!!!

t.test(log.engine,mu = log(5.4),alternative = "two.sided")

     One Sample t-test

data:  log.engine
t = 2.2841, df = 47, p-value = 0.02693
alternative hypothesis: true mean is not equal to 1.686399
95 percent confidence interval:
 1.709925 2.057431
sample estimates:
mean of x
 1.883678
```

http://www.cyclismo.org/tutorial/R/cholesterol.html

R - A Walkthrough

To find the power we need to set a level for the mean and then find the probability that we would accept the null hypothesis if the mean is really at the prescribed level. Here we will find the power to detect a difference if the level were 7.

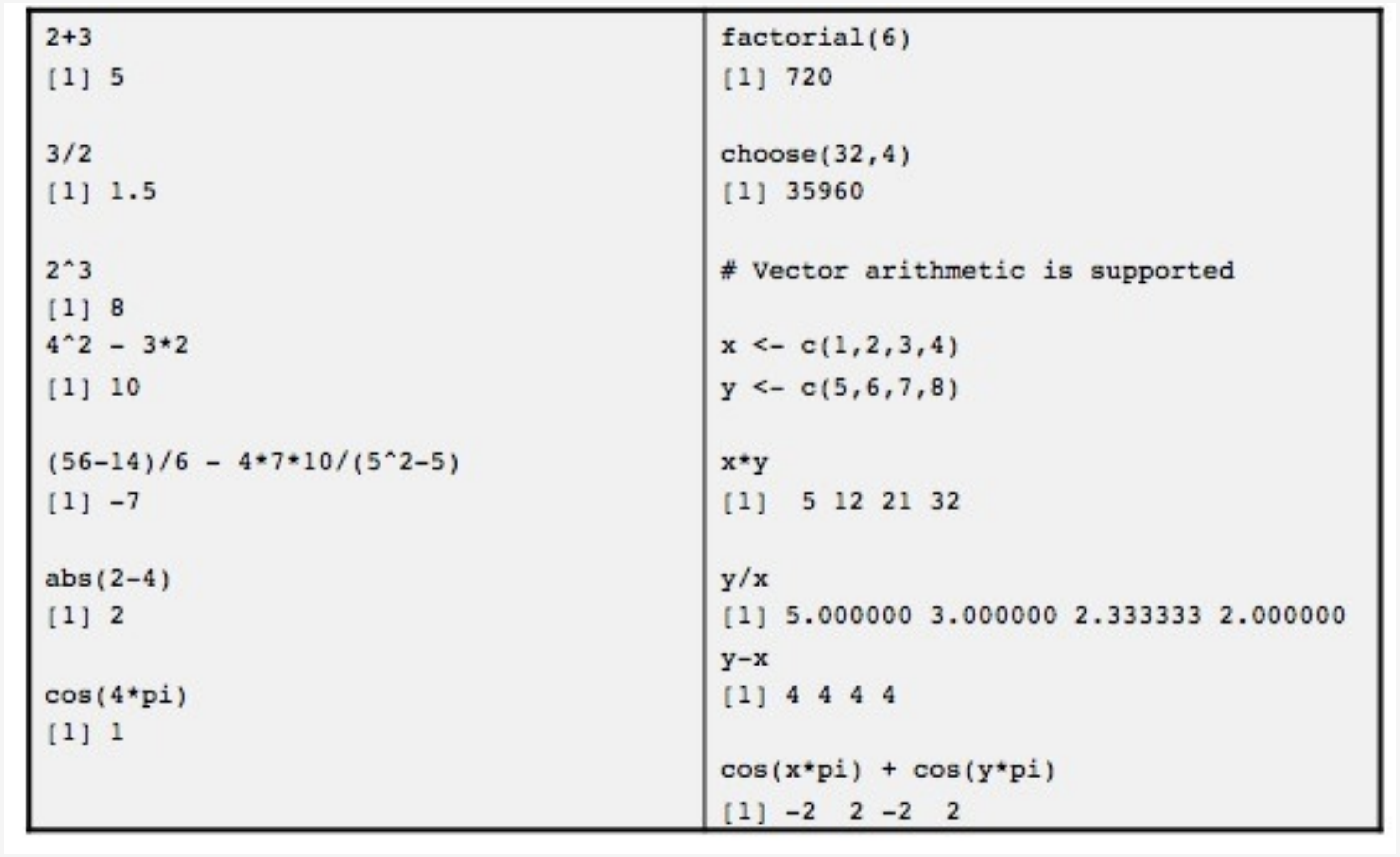
```
power.t.test(n=n,delta=log(7)-log(5.4),sd=s,sig.level=0.05,
              type="one.sample",alternative="two.sided",strict = TRUE)

     One-sample t test power calculation 

              n = 48
          delta = 0.2595112
             sd = 0.5983851
      sig.level = 0.05
          power = 0.8371421
    alternative = two.sided
```

http://www.cyclismo.org/tutorial/R/cholesterol.html

R - Your First Real Session

```
?mean                      # Get help on the mean function

example(kmeans)     # Run an example of kmeans (if it exists)

pi                 # Some popular quantities are built-in to R
[1] 3.141593

sqrt(2)             # Basic arithmetic
[1] 1.414214

print(pi)          # Print the comments of the pi variable
[1] 3.141593

X = 3; Y = 4  # Semicolon lets you enter 2 commands on the same line

Z = sqrt(X^2 + Y^2)    # Variables contain information


ls()               # List all variables in the “environment”
[1] "X" "Y" "Z"
```

R - Your First Real Session - Calculator



```
2+3
[1] 5

3/2
[1] 1.5

2^3
[1] 8
4^2 - 3*2
[1] 10

(56-14)/6 - 4*7*10/(5^2-5)
[1] -7

abs(2-4)
[1] 2

cos(4*pi)
[1] 1
```

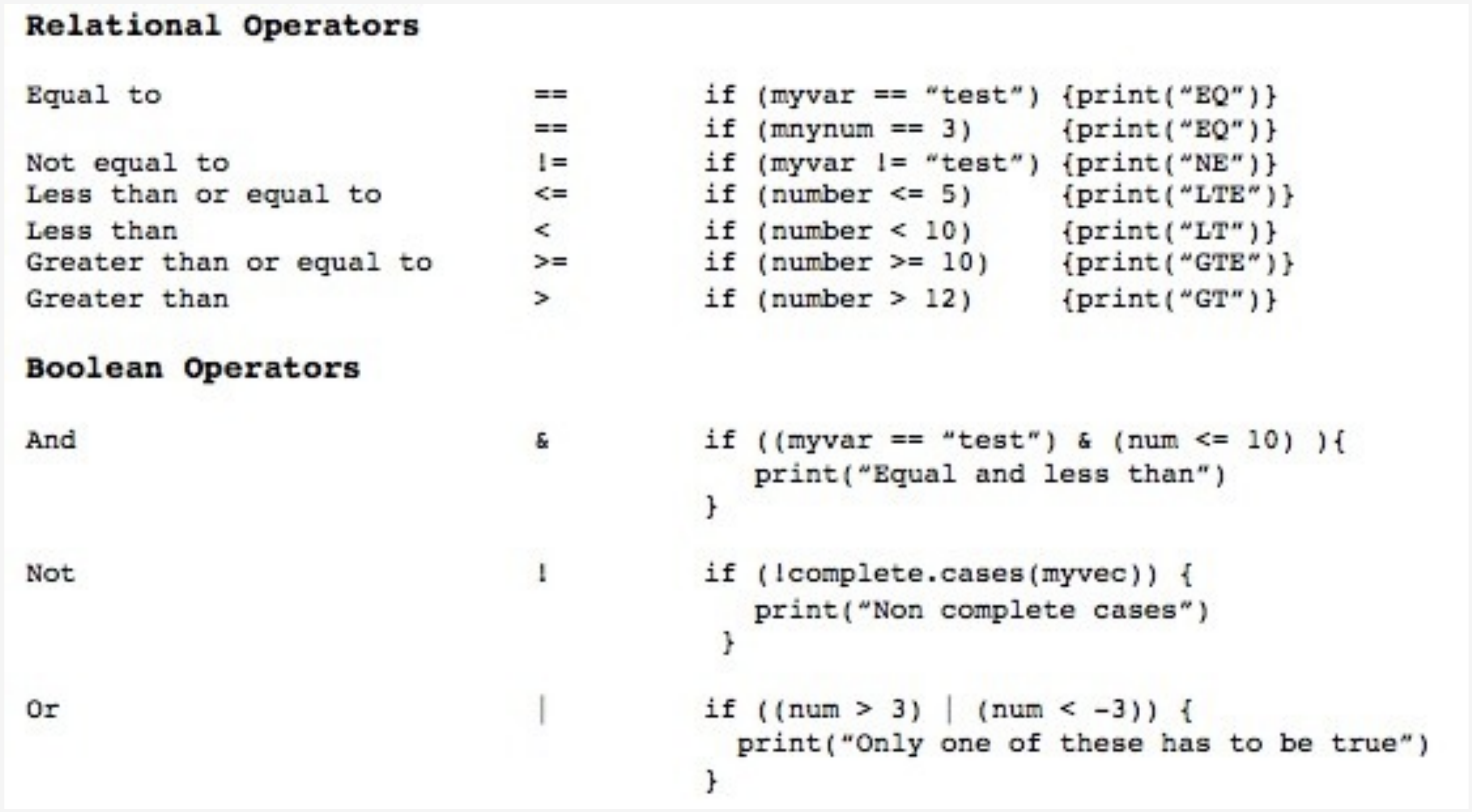
```
factorial(6)
[1] 720

choose(32,4)
[1] 35960

# Vector arithmetic is supported

x <- c(1,2,3,4)
y <- c(5,6,7,8)

x*y
[1]  5 12 21 32

y/x
[1] 5.000000 3.000000 2.333333 2.000000
y-x
[1] 4 4 4 4

cos(x*pi) + cos(y*pi)
[1] -2  2 -2  2
```

R - Your First Real Session - Calculator

```
log(10)
[1] 2.302585

log10(100)
[1] 2

sin(pi/2)
[1] 1

cos(pi/2)
[1] 6.123234e-17

> 1.3e6
[1] 1300000

> 9 %% 2
[1] 1

> 8 %% 2
[1] 0

> floor(5.7)
[1] 5
```

```
ceiling(6.8)
[1] 7

round(6.889,2)
[1] 6.89

3/0
[1] Inf

0/0
[1] NaN

is.finite(3)
[1]

x = c(1:8,NA)
x
[1]  1  2  3  4  5  6  7  8 NA

mean(x)
[1] NA

mean(x,na.rm=T)
[1] 4.5
```

R - Your First Real Session - Operators



```
Relational Operators

Equal to                     ==       if (myvar == “test”) {print(“EQ”)}
                             ==       if (mnynum == 3)     {print(“EQ”)}
Not equal to                 !=       if (myvar != “test”) {print(“NE”)}
Less than or equal to        <=       if (number <= 5)     {print(“LTE”)}
Less than                    <        if (number < 10)     {print(“LT”)}
Greater than or equal to     >=       if (number >= 10)    {print(“GTE”)}
Greater than                 >        if (number > 12)     {print(“GT”)}

Boolean Operators

And                          &        if ((myvar == “test”) & (num <= 10) ){
                                         print(“Equal and less than”)
                                      }

Not                          !        if (!complete.cases(myvec)) {
                                         print(“Non complete cases”)
                                       }

Or                           |        if ((num > 3) | (num < -3)) {
                                        print(“Only one of these has to be true”)
                                      }
```

R - Your First Real Session - Operators

Here are some popular math formulas rewritten in R. Note that many of these are implemented within R for convenient use. However it is helpful to gain a knowledge of operators and their precedence.

```
# a^2 + b^2 = c^2                    # Pythagorean Theorem

a = 2; b = 4

c = sqrt(a^2 + b^2)                  # To solve the PT for c

a = 2; b = 4; c = 1

(-b + sqrt(b^2 - 4*a*c)) / (2*a)      # First case quadratic formula solution

(-b - sqrt(b^2 - 4*a*c)) / (2*a)      # Second case quadratic formula solution

r = 4; h = 6; b = 3

circumference = 2*pi*r       # circumference of a circle

area = (b*h)/2                # Area of a triangle
```

R - Your First Real Session - Operators

You can create expressions with R for later evaluation.

```
area = expression( (b*h)/2 )

# Solve where b = 3 and h = 4

b = 3

h = 4

eval(area)
[1] 6
```

Note that you can easily make a function out of this also:

```
area <- function(b,h) {   # b and h are arguments / placeholders
    my.area = (b*h)/2
    return(my.area)
}

area(3,4)
[1] 6
```

R - Your First Real Session - Operators

You can create expressions with R for later evaluation.

```
r1 = expression((-b + sqrt(b^2 - 4*a*c)) / (2*a))

r2 = expression((-b - sqrt(b^2 - 4*a*c)) / (2*a))

# Solve for ax^2 + bx + c where a = 1, b=6, and c=8

a = 1 ; b=6 ; c=8

eval(r1)
[1] -2

eval(r2)
[1] -4

a*eval(r1)^2 + b*eval(r1) + c
[1] 0

a*eval(r2)^2 + b*eval(r2) + c
[1] 0
```

R - Your First Real Session - Operators

We could also create a function out of this:

```
my.quad <- function(a,b,c) {
       r1 = (-b + sqrt(b^2 - 4*a*c)) / (2*a)
       r2 = (-b - sqrt(b^2 - 4*a*c)) / (2*a)
       my.roots = c(r1,r2)
       return(my.roots)
}


# Solve for ax^2 + bx + c where a = 1, b=6, and c=8

my.quad(1,6,8)
[1] -2 -4
```

Workspace - Startup

This is one of the more neglected areas when discussing R. However, it is important to understand how one gets information into and out of R. You can setup a file called .Rprofile in your home directory that enables you to configure default options for many things in R.

So if your home directory is /home/wsp or /Users/fender (assumes Linux or OSX) then create a file in your home directory called .Rprofile. Note that you are not obligated to use this capability but the more experience you get with R the more likely you will be interested in using it.

```
$ touch ~/.Rprofile
$ vi .Rprofile
```

(Then edit it to contain things like on the next slide)

Workspace - Startup

```
# Things you might want to change

options(editor="notepad")
cd = setwd
pwd = getwd
lss = dir

# R interactive prompt
options(prompt="> ")
options(continue="+ ")

# General options
options(digits=3)
options(width = 130)
options(graphics.record=TRUE)

.First <- function(){
 library(Hmisc)
 cat("\nWelcome at", date(), "\n")
}

.Last <- function(){
 cat("\nGoodbye at ", date(), "\n")
}
```

Workspace - Navigating Directories and Files

This is one of the more neglected areas when discussing R. However, it is important to understand how one gets information into and out of R.

```
getwd()
[1] "/Users/fender/TEST.DIR"

setwd("/Users/fender")

getwd()
[1] "/Users/fender"

setwd("/Users/fender/TEST.DIR")

getwd()
[1] "/Users/fender/TEST.DIR"

dir()
[1] "coolpkg"        "coolpkg_1.0.tar.gz" "coolpkg.pdf"        "coolpkg.Rcheck"
"g.Rd"                 "stuff.R"

dir(,recursive=TRUE)
 [1] "coolpkg_1.0.tar.gz"                       "coolpkg.pdf"
 [3] "coolpkg.Rcheck/00check.log"               "coolpkg.Rcheck/00install.out"
 [5] "coolpkg.Rcheck/coolpkg-Ex.pdf"            "coolpkg.Rcheck/coolpkg-Ex.R"
 [7] "coolpkg.Rcheck/coolpkg-Ex.Rout"           "coolpkg.Rcheck/coolpkg-manual.log"
 [9] "coolpkg.Rcheck/coolpkg-manual.pdf"        "coolpkg.Rcheck/coolpkg/data/d.rda"
```

Workspace - Navigating Directories and Files

It is possible to manipulate files from within R. This doesn't mean that this is the best way to do this but it is a possibility.

```
R.home()
[1] "/Library/Frameworks/R.framework/Resources"

setwd(R.home())

list.files()
 [1] "bin"          "COPYING"      "doc"          "etc"
     "fontconfig"    "include"      "Info.plist"    "lib"
 [9] "library"      "man1"         "modules"      "NEWS"
      "NEWS.pdf"     "R"            "Rscript"       "share"


list.files(path = "doc")
 [1] "AUTHORS"          "COPYING"          "COPYRIGHTS"
     "CRAN_mirrors.csv" "FAQ"              "html"
 [7] "KEYWORDS"         "KEYWORDS.db"      "manual"   "NEWS.rds" "RESOURCES"
"THANKS"
```

Workspace -Navigating Directories and Files

It is possible to manipulate files from within R. This doesn't mean that this is the best way to do this but it is a possibility.

```
setwd("doc")

list.files()
 [1] "AUTHORS"           "COPYING"           "COPYRIGHTS"
"CRAN_mirrors.csv" "FAQ"              "html"
 [7] "KEYWORDS"          "KEYWORDS.db"       "manual"            "NEWS.rds"
"RESOURCES"        "THANKS"



path.expand("~")    # Finds you home directory in Linux

setwd(path.expand("~"))   # Sets your working directory to be your home
```

Workspace - Navigating Directories and Files

It is possible to manipulate files from within R. This doesn't mean that this is the best way to do this but it is a possibility.

```
my.vec = list.files()

my.vec
 [1] "AUTHORS"          "COPYING"      "COPYRIGHTS"       "CRAN_mirrors.csv"
"FAQ"                "html"
 [7] "KEYWORDS"         "KEYWORDS.db"      "manual"           "NEWS.rds"
"RESOURCES"          "THANKS"

length(my.vec)
[1] 12
```

Workspace - Navigating Directories and Files

One can also use the “system” command to run commands at the UNIX level and reroute the output back into the R session.

```
system("ls -l")

AUTHORS
COPYING
COPYRIGHTS
CRAN_mirrors.csv
FAQ
KEYWORDS
KEYWORDS.db
NEWS.rds
RESOURCES
THANKS
html
manual

system("ls -1",intern=T)
 [1] "AUTHORS"          "COPYING"          "COPYRIGHTS"
"CRAN_mirrors.csv" "FAQ"              "KEYWORDS"
 [7] "KEYWORDS.db"      "NEWS.rds"         "RESOURCES"        "THANKS"
"html"             "manual"
```

Workspace - Saving Workspace

If you haven't noticed already, whenever you quit R it will prompt you to to save your workspace. What this means is that you will have the ability to save a copy of all your current R objects (as evidenced by the ls()) command. Generally you will want to save it.

Behind the scenes it stores the information into a BINARY file called .Rdata in whatever directory you are currently in.

```
> q()
Save workspace image? [y/n/c]: y

Goodbye at  Mon Oct  1 14:26:47 2012

fenders-macbook:TEST.DIR fender$ ls .Rdata
.Rdata
```

When you start R back up in this directory it will load this file by default. A rookie mistake is when users start R from another directory expecting to find their objects reloaded

Workspace - Saving Workspace

If you want finer-grained control then you can specify your own file name.

```
save(list = ls(all=TRUE), file = "/Users/myhome/.MyRdata")
```

You can also elect to save only specific objects to a file.

```
 my.lm = lm(mpg ~ wt,mtcars)
ls(my.lm)
 [1] "assign"        "call"          "coefficients"  "df.residual"   "effects"
"fitted.values" "model"
 [8] "qr"            "rank"          "residuals"     "terms"         "xlevels"

save(my.lm,file="/Users/myhome/.MyLM")
```

You can retrieve the contents of this file from any future R session by doing:

```
load("/Users/myhome/.MyLM")
```

Workspace - Saving Workspace

In general I suggest that you create a master directory to contain your R projects. You can then create sub-directories to house your various efforts.

That way when you come back to them after some lapse in time (e.g. semester break, vacation, etc) you can look at your directory structure and know where to pick back up.

```
$ ls RProjects

|-RProjects
    |-Genomes
           |---1000_Genomes
           |---Centenarians
    |-HIV
           |---Replicates
    |-Influenza
```

Each subdirectory could have its own .RData file (or whatever you want to name it) to keep objects for that project intact. Don't pile up everything into one directory because it gets really confusing.

Workspace - Navigating

This is one of the more neglected areas when discussing R. However, it is important to understand how one gets information into and out of R.

```
ls()            # Check to see what objects there are in memory
character(0)

source("stuff.R",echo=TRUE)

f = function(x) x + y
g = function(x) x - y


ls()            # Ah, we now have two functions in the workspace.
[1] "f" "g"
```

Workspace - Navigating

This is one of the more neglected areas when discussing R. However, it is important to understand how one gets information into and out of R.

```
ls()          # Ah, we no have two functions in the workspace.
[1] "f" "g"

rm("g")

ls()
[1] "f"

rm(list = ls())   # Removes all objects in the workspace
                   # This can be dangerous - use with care
```

Workspace - Basic I/O - Variables

As in most programming languages, it is customary to store or hold the results of an operation in a variable name. In R such results are assigned with the symbols "<-" or "=". Variable names are case sensitive.

```
 X <- 2.5    # These two statements are equivalent

 X = 2.5 # Same as above

 X
[1] 2.5

mynewvar = X + 3

MYNEWVAR = X + 3      # Two different variables
```

R has several one-letter reserved words: c, q, s, t, C, D, F, I, and T

You cannot begin a variable name with a period character "."

Workspace - Basic I/O - Variables

Valid Names

Variable names in R are case-sensitive, so myvar is not the same as MYVAR

Variables names should not begin with numbers (e.g. 1) or symbols (%,_)

Variable names should not contain spaces in the name (my var)

Workspace - Basic I/O - Variables

Valid Variable Names	Invalid Variable Names
`mean.height`	`.mean.height`
`smoker`	`_myvariable`
`non.smoker`	`_Mean.height`
`temp.var`	`1variable`
`patient_id`	`1_variable`
`Eye.Color`	`%some.var`
`State_Population`	`some var`
`disease.state`	`“some var”`
`White_Cell_Count`	
`jobTitle`	

Scope

R uses lexical scoping, which we will explore in some detail later.

Since variables cannot be declared, (they pop into existence on first assignment) it is not always easy to determine the scope of a variable. You cannot tell just by looking at the source code of a function whether a variable is local to that function.

Workspace - Basic I/O

You can use the scan function to enter data from the command line. This is also useful when pasting in a row of numbers from a website.

```
my.input.vector = scan()
1: 8 9 10 11 23 48 73
8:
Read 7 items

my.input.vector
[1]  8  9 10 11 23 48 73
```

The cat command can be used to take some input and write it to a file:

```
cat("TITLE extra line", "2 3 5 7", file="ex.data")
cat("TITLE extra line", "2 3 5 7", "11 13 17",
+ file="ex.data", sep="\n")

dir(pattern="ex.data")
[1] "ex.data"
```

Workspace - Basic I/O

You can prompt users for input but this isn't really all that common. But let's say you wanted to prompt the user to enter a number:

```
n <- readline("enter a positive integer: ")
enter a positive integer: 23

n
[1] "23"
```

You could make this more involved by doing some error checking to make sure they enter in an integer.

```
n = -1

while(n < 1 ){
   n <- readline("enter a positive integer: ")
   n <- ifelse(grepl("\\D",n),-1,as.integer(n))
   if(is.na(n)){break}  # breaks when hit enter
}
enter a positive integer: sdfs
enter a positive integer: 45
```

Workspace - Basic I/O

It is also very easy to turn around and write data frames and matrices to .CSV file for use with any program that uses .CSV files (e.g. SAS, SPSS, Minitab, Excel, etc).

```
head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1

write.table(mtcars,file="mtcars.csv",
              row.names=TRUE,          # Row names get saved
              col.names=TRUE,          # Header gets saved
              sep=",")



$ head mtcars.csv
"mpg","cyl","disp","hp","drat","wt","qsec","vs","am","gear","carb"
"Mazda RX4",21,6,160,110,3.9,2.62,16.46,0,1,4,4
"Mazda RX4 Wag",21,6,160,110,3.9,2.875,17.02,0,1,4,4
```

Workspace - CSV files

You can also read text files a line at a time use the readLines() function. Again, this is not so common an activity but it is something you should have some exposure to. Let's say we have a comma delimited file.

Workspace - "Sinking" your work

From time to time you might want to capture the output of your work though as you already know this is possible using the "save" command to save the contents of your environment. The "cat", "write", and "write.table" commands allow you to save specific variables to a file.

The "sink" command exists to provide a way to redirect the output from R commands directly to a file. This might be something you would do if you were running a very long running batch job. This would let you see the progress of your program on an incremental basis.

Let's say we have the following R statements:

```
set.seed(123)
x <-rnorm(10)
y <-rnorm(10)

print(x)
cat ("y =", y, "\n")

t.test(x,y)
plot(x,y)
```

Workspace - "Sinking" your work

If we run this interactively the output from this would look like the following. We would also see a plot window because of the call to the plot command.

```
set.seed(123)
x <-rnorm(10)
y <-rnorm(10)

print(x)
 [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774  1.71506499
 [7]  0.46091621 -1.26506123 -0.68685285 -0.44566197

cat ("y =", y, "\n")
y = 1.224082 0.3598138 0.4007715 0.1106827 -0.5558411 1.786913 0.4978505 -1.966617
0.7013559 -0.4727914

t.test(x,y)
     Welch Two Sample t-test

data:  x and y
t = -0.3006, df = 17.872, p-value = 0.7672
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.0710488  0.8030562
sample estimates:
 mean of x  mean of y
0.07462564 0.20862196
```

Workspace - "Sinking" your work

If we wanted we could redirect all the output from the print,cat statement and t.test function to a file called "my.results.txt"

```
sink("my.results.txt") # All output will now go to "my.results.txt"

set.seed(123)
x <-rnorm(10)
y <-rnorm(10)

print(x)
cat ("y =", y, "\n")

t.test(x,y)
plot(x,y)

sink()                # This will turn off the sink
```

If we run this then we see only the statements as they are processed but not any of the output. To see the output we look at "my.results.txt"

Workspace - "Sinking" your work

If we run this then we see only the statements as they are processed but not any of the output. To see the output we look at "my.results.txt". Note that the result of the plot command will go by default to a file named "Rplots.pdf" located in the current working directory.

```
$ more my.results.txt

 [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774  1.71506499
 [7]  0.46091621 -1.26506123 -0.68685285 -0.44566197
y = 1.224082 0.3598138 0.4007715 0.1106827 -0.5558411 1.786913 0.4978505
-1.966617 0.7013559 -0.4727914

        Welch Two Sample t-test

data:  x and y
t = -0.3006, df = 17.872, p-value = 0.7672
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.0710488  0.8030562
sample estimates:
 mean of x  mean of y
0.07462564 0.20862196
```

Workspace - "Sinking" your work

Okay, an alternative approach to using sink is to run your R jobs as "batch" jobs. Let's put these commands into a file called test.R. Note the we don't have any "sink" commands.

```
set.seed(123)
x <-rnorm(10)
y <-rnorm(10)

print(x)
cat ("y =", y, "\n")

t.test(x,y)
plot(x,y)
```

We can run this as a batch job on Linux and OSX like:

```
$ R CMD BATCH test.R myout
```

This will put your results into a file "myout". As before the plots will be stashed into a file called "Rplots.pdf".

Workspace - "Sinking" your work

If you wanted more control over how your plot was named then you can use one of the functions designed to create plots in a known format (PNG, JPEG, PDF).

```
set.seed(123)
x <-rnorm(10)
y <-rnorm(10)

print(x)
cat ("y =", y, "\n")

t.test(x,y)

pdf("myplots.pdf")   # Redirects plots to myplots.pdf

plot(x,y)

dev.off()            # Turns off plot redirection
```

We can run this as a batch job on Linux and OSX like:

```
$ R CMD BATCH test.R myout
```

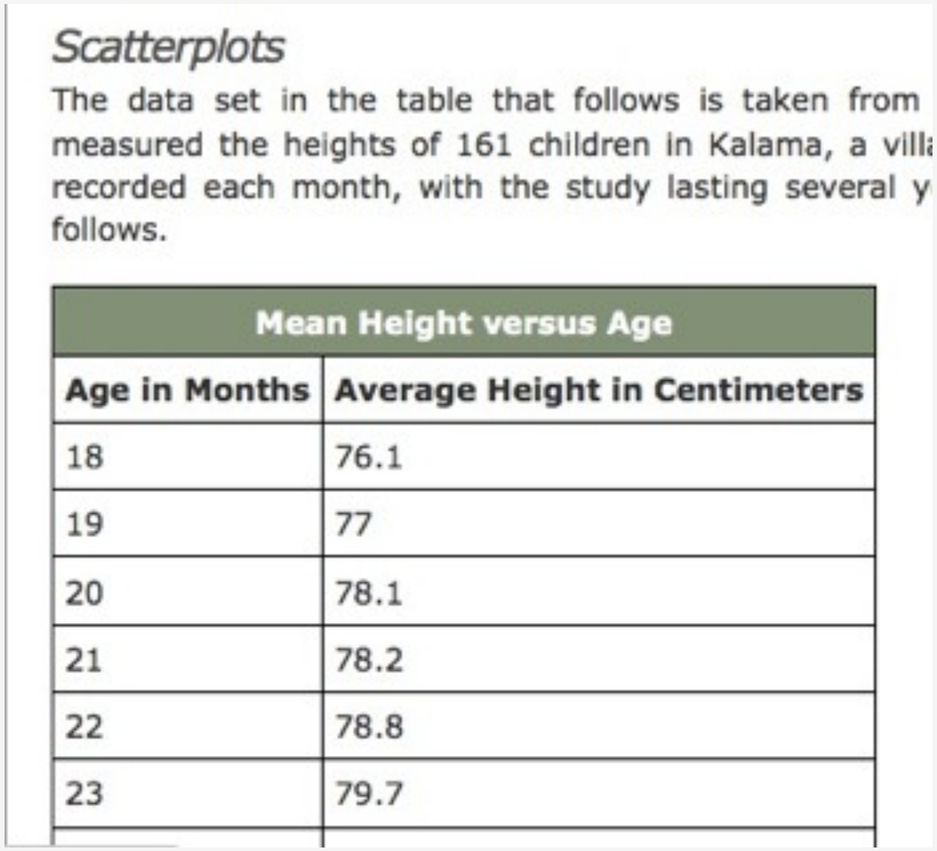
Workspace - Tables

Note that you can read CSV files directly from the Internet as long as you have the URL. AND there are dedicated functions to intelligently parse the data so you don't have to do all the things you had to do in the previous slide.

```
url = "http://www.bimcore.emory.edu/BIOS560R/DATA.DIR/hsb2.csv"

my.input = read.table(url,header=T,sep=",")

head(my.input)
  gender  id race ses schtyp  prgtype read write math science socst
1      0  70    4   1      1  general   57    52   41      47    57
2      1 121    4   2      1   vocati   68    59   53      63    61
3      0  86    4   3      1  general   44    33   54      58    31
4      0 141    4   3      1   vocati   63    44   47      53    56
5      0 172    4   2      1 academic   47    52   57      53    61
6      0 113    4   2      1 academic   44    52   51      63    61
```

Workspace - Tables

Sometimes you want to read data from the Internet but its not in a CSV file. Its in a HTML table. Take a look at this example. Let's say we want to get the data in the first table from here: http://msenux.redwoods.edu/math/R/regression.php



Scatterplots

The data set in the table that follows is taken from
measured the heights of 161 children in Kalama, a vill
recorded each month, with the study lasting several y
follows.

Mean Height versus Age	
Age in Months	Average Height in Centimeters
18	76.1
19	77
20	78.1
21	78.2
22	78.8
23	79.7

Workspace - Tables

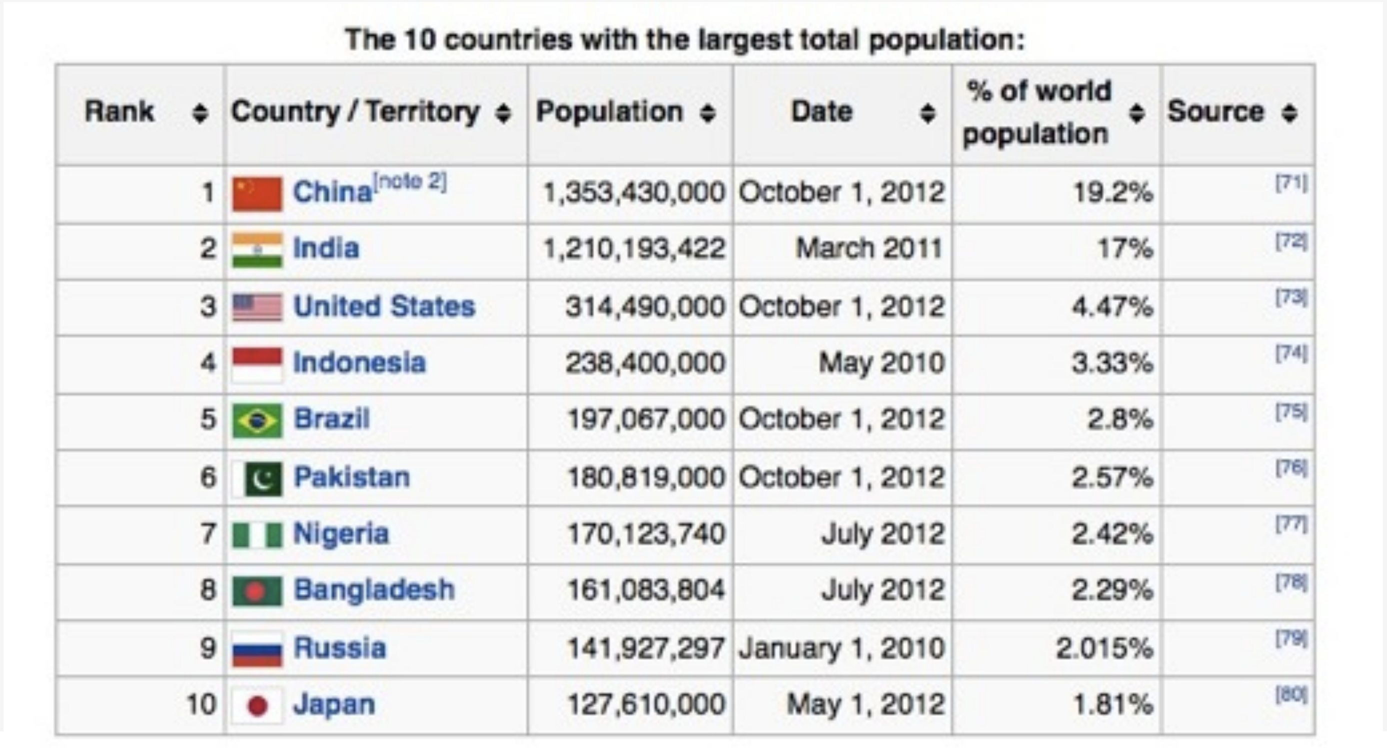
Let's write a little R code to get this information:

```
library(XML)
url = "http://msenux.redwoods.edu/math/R/regression.php"
my.table = readHTMLTable(url,which=1)

my.table
                      V1                              V2
1  Mean Height versus Age                           <NA>
2           Age in Months Average Height in Centimeters
3                      18                           76.1
4                      19                             77
..
..

my.table = my.table[-1:-2,]
names(my.table) = c("Age","Height")
head(my.table)
  Age Height
3  18   76.1
4  19     77
5  20   78.1
6  21   78.2
7  22   78.8
```

Workspace - Complicated Tables

Let's look at a more involved example. Look at the follow info on World Population http://en.wikipedia.org/wiki/World_population

This page has many tables. Let's get the data corresponding to the one that looks like:



The 10 countries with the largest total population:

Rank	Country / Territory	Population	Date	% of world population	Source
1	China[note 2]	1,353,430,000	October 1, 2012	19.2%	[71]
2	India	1,210,193,422	March 2011	17%	[72]
3	United States	314,490,000	October 1, 2012	4.47%	[73]
4	Indonesia	238,400,000	May 2010	3.33%	[74]
5	Brazil	197,067,000	October 1, 2012	2.8%	[75]
6	Pakistan	180,819,000	October 1, 2012	2.57%	[76]
7	Nigeria	170,123,740	July 2012	2.42%	[77]
8	Bangladesh	161,083,804	July 2012	2.29%	[78]
9	Russia	141,927,297	January 1, 2010	2.015%	[79]
10	Japan	127,610,000	May 1, 2012	1.81%	[80]

Workspace - Complicated Tables

This appears to be the 5th table on the page. Note that this is one of those things that you will probably have to try a few times before you get it right.

```
url = "http://en.wikipedia.org/wiki/World_population"

table.four = readHTMLTable(url, which=4) # Most densely populated countries

table.four
   Rank Country / Territory    Population            Date  % of world\npopulation Source
1     1       China[note 2] 1,353,430,000 October 1, 2012                   19.2%   [71]
2     2               India 1,210,193,422      March 2011                     17%   [72]
3     3       United States   314,490,000 October 1, 2012                   4.47%   [73]
4     4           Indonesia   238,400,000        May 2010                   3.33%   [74]
```

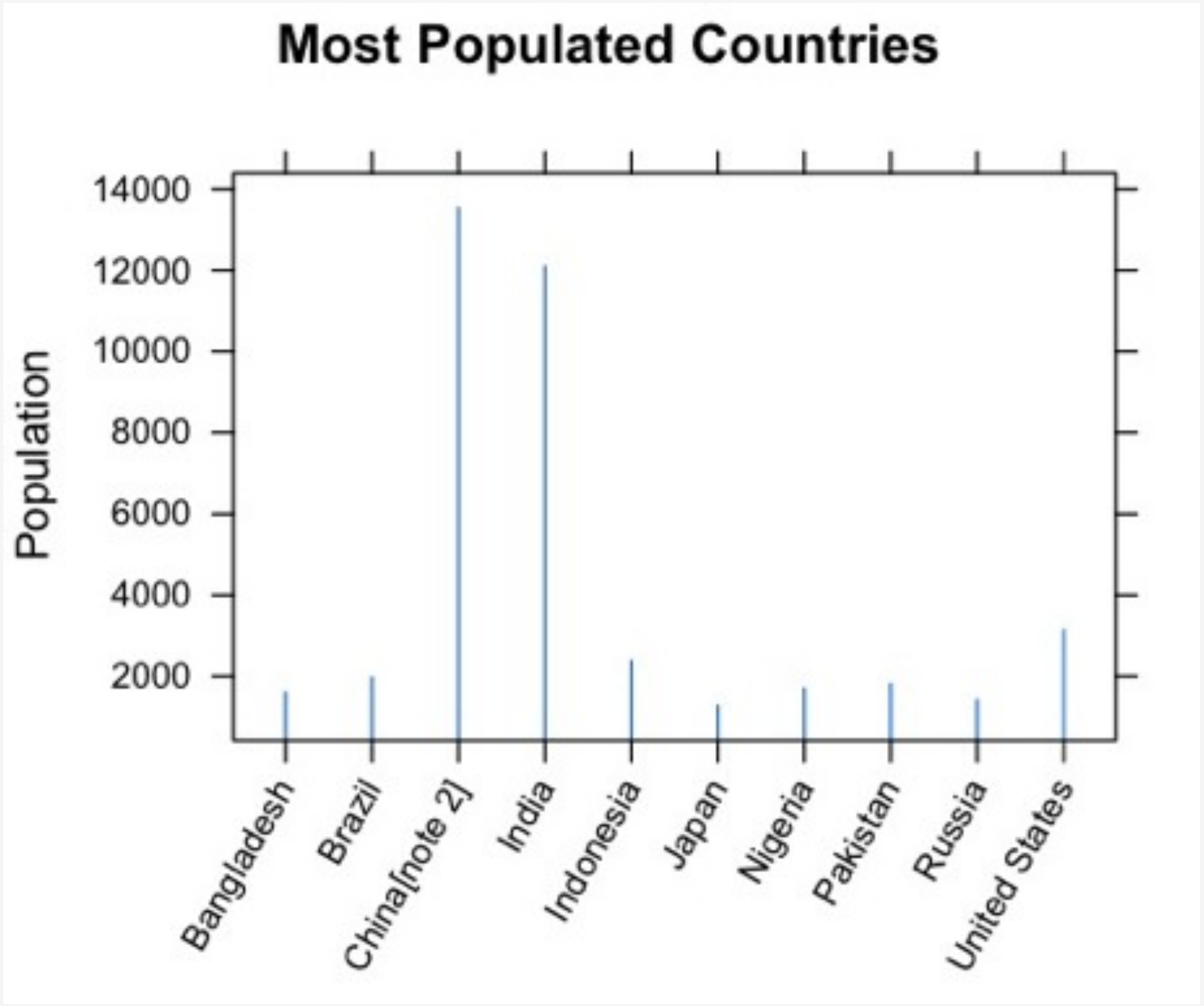
Okay, we got the data but its a little messy. Let's clean it up some.

Workspace - Complicated Tables

Okay, we got the data but its a little messy. Let's clean it up some.

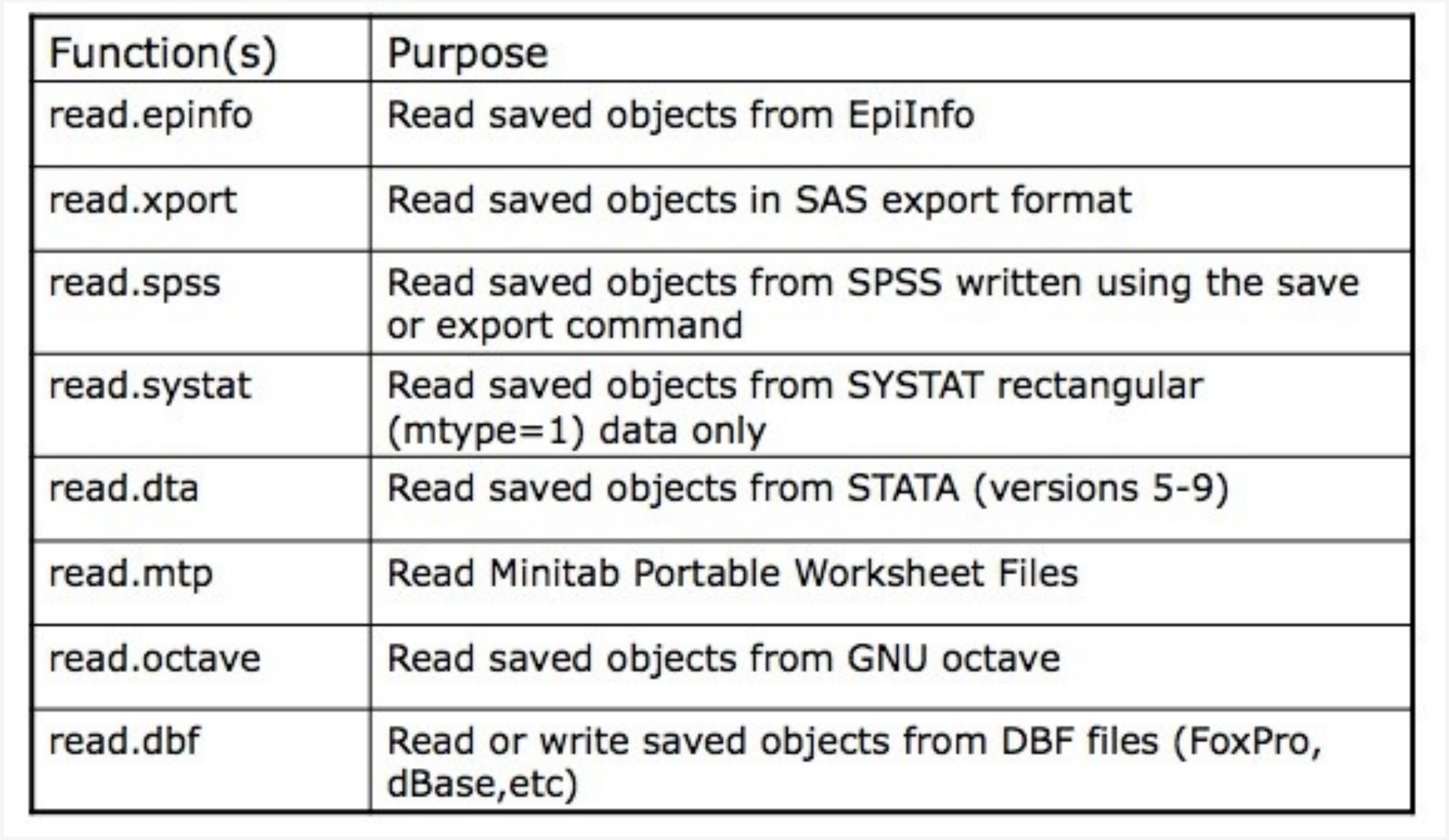
```
table.four = table.four[,-4:-6]  # Eliminate the 4th-6th column
table.four
   Rank Country / Territory    Population
1     1           China[73] 1,352,190,000
2     2               India 1,210,193,422


# Get rid of the commas in the numbers

table.four$Population=as.numeric(gsub(",","",table.five$Population))/100000

# Give the columns new names

names(table.four) = c("Rank","Country","Population")

# Plot the data
library(lattice)

xyplot(Population ~ Country,table.four,scales = list(x = c(rot=60)),
       type="h",main="Most Densely Populated Countries")
```

Workspace - Complicated Tables



Workspace - Other Stats Packages

Its also possible to read data sets from other statistical packages. However, as with Excel, I suggest you export those datasets into a CSV format and then import them into R.



Function(s)	Purpose
read.epinfo	Read saved objects from EpiInfo
read.xport	Read saved objects in SAS export format
read.spss	Read saved objects from SPSS written using the save or export command
read.systat	Read saved objects from SYSTAT rectangular (mtype=1) data only
read.dta	Read saved objects from STATA (versions 5-9)
read.mtp	Read Minitab Portable Worksheet Files
read.octave	Read saved objects from GNU octave
read.dbf	Read or write saved objects from DBF files (FoxPro, dBase,etc)

Workspace - Excel

You can even read directly from Excel spreadsheets but this is usually something you wouldn't do. Always try to save the data out of Excel into a CSV file and then import it into R.

If you simply MUST read the Excel spreadsheet directly then use
an approach similar to the following. Note that I've never done this so can't say that it will work. Also, this might work only on Windows machines.

```
> library(RODBC)
> channel <- odbcConnectExcel("examp.xls")
## list the spreadsheets
> sqlTables(channel)
       TABLE_CAT TABLE_SCHEM        TABLE_NAME   TABLE_TYPE REMARKS
     1 C:\\bdr            NA           Sheet1$ SYSTEM TABLE      NA
     2 C:\\bdr            NA           Sheet2$ SYSTEM TABLE      NA
     3 C:\\bdr            NA           Sheet3$ SYSTEM TABLE      NA
     4 C:\\bdr            NA Sheet1$Print_Area        TABLE      NA

     ## retrieve the contents of sheet 1, by either of
> sh1 <- sqlFetch(channel, "Sheet1")
> sh1 <- sqlQuery(channel, "select * from [Sheet1$]")
```

Workspace - Databases

Note that you can also read and write to relational databases. We will explore this much more in depth in a later module. You may wonder - Why should you care about connecting to relational databases from R. Good reasons include:

```
* There are limitations on the types of data that R handles well. Since
all data being manipulated by R are resident in memory, and several
copies of the data can be created during execution of a function,


* R is not well suited to extremely large data sets. Data objects that
are more than a (few) hundred megabytes in size can cause R to run out of
memory, particularly on a 32-bit operating system.


* R does not easily support concurrent access to data. That is, if more
than one user is accessing, and perhaps updating, the same data, the
changes made by one user will not be visible to the others.


* R does support persistence of data, in that you can save a data object
or an entire worksheet from one session and restore it at the subsequent
session, but the format of the stored data is specific to R and not
easily manipulated by other systems.
```

Workspace - Databases

Database management systems (DBMSs) and, in particular, relational DBMSs (RDBMSs) *are* designed to do all of these things well. Their strengths are:

1. To provide fast access to selected parts of large databases.
2. Powerful ways to summarize and cross-tabulate columns in databases.
3. Store data in more organized ways than the rectangular grid model of spreadsheets and R data frames.
4. Concurrent access from multiple clients running on multiple hosts while enforcing security constraints on access to the data.
5. Ability to act as a server to a wide range of clients.

The sort of statistical applications for which DBMS might be used are to extract a 10% sample of the data, to cross-tabulate data to produce a multi-dimensional contingency table, and to extract data group by group from a database for separate analysis.

Workspace - Databases

Note that you can also read and write to relational databases. We will explore this much more in depth in a later module. For now here is an example of how you might connect to a MySQL database.

```
m <- dbDriver("SQLite")

    # initialize a new database to a tempfile and copy some data.frame
    # from the base package into it

tfile <- tempfile()
con <- dbConnect(m, dbname = tfile)
data(USArrests)
dbWriteTable(con, "USArrests", USArrests)

    # query
rs <- dbSendQuery(con, "select * from USArrests")
d1 <- fetch(rs, n = 10)      # extract data in chunks of 10 rows
d1
     row_names Murder Assault UrbanPop Rape
1      Alabama   13.2     236       58 21.2
2       Alaska   10.0     263       48 44.5
3      Arizona    8.1     294       80 31.0
4     Arkansas    8.8     190       50 19.5
5   California    9.0     276       91 40.6
6     Colorado    7.9     204       78 38.7
```

Workspace - Databases

Note that you can also read and write to relational databases. We will explore this much more in depth in a later module. For now here is an example of how you might connect to a MySQL database.

```
## Select from the loaded table

dbClearResult(rs)

rs <- dbSendQuery(con, "select * from USArrests where
                                 Assault > 10 order by Murder")
d1 <- fetch(rs, n = 10)
d1
       row_names Murder Assault UrbanPop Rape
1   North Dakota    0.8      45       44  7.3
2          Maine    2.1      83       51  7.8
3  New Hampshire    2.1      57       56  9.5
4           Iowa    2.2      56       57 11.3
5        Vermont    2.2      48       32 11.2
6          Idaho    2.6     120       54 14.2
7      Wisconsin    2.6      53       66 10.8
8      Minnesota    2.7      72       66 14.9
9           Utah    3.2     120       80 22.9
10   Connecticut    3.3     110       77 11.1
```

Workspace - Databases

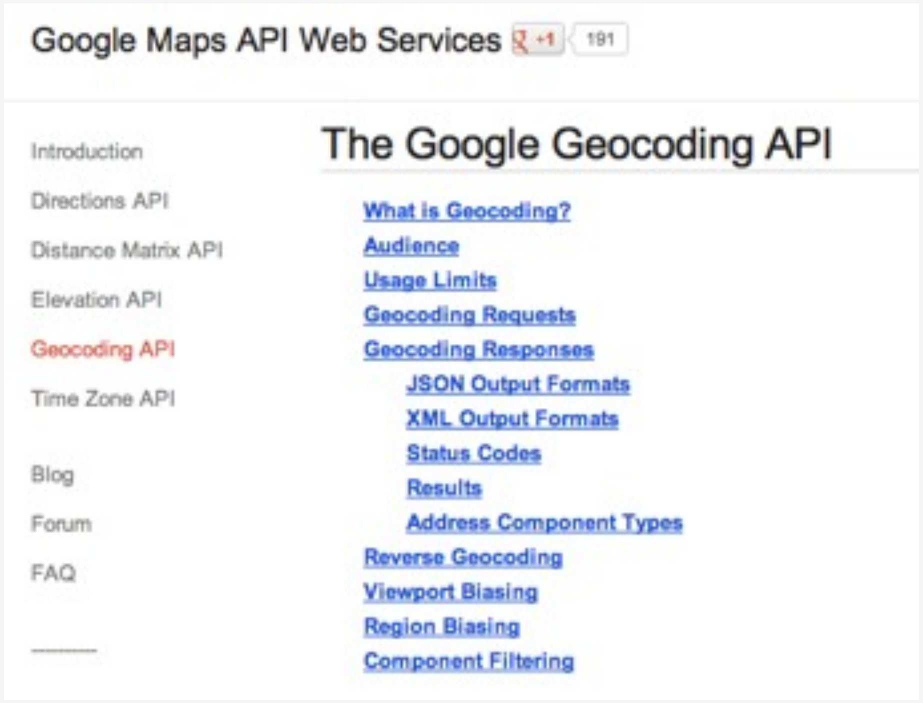
Another way to work with data sets as if they were databases is to use the "sqldf" package. We'll look more closely at this in the data frame section. This package is very useful if you already know SQL and are just learning R. sqldf is an add on package so you will first need to install it.

```
install.packages("sqldf", dependencies = TRUE)

library(sqldf)

data(mtcars)

sqldf("select * from mtcars where mpg > 20 AND cyl == 4")
```

Workspace - XML

R can also process XML files from the Internet, which is very powerful. As an example we'll access Google's GeoCoding pages to get the Latitude and Longitude for Atlanta, GA from within a program. Its just a simple example.

https://developers.google.com/maps/documentation/geocoding/


Google Maps API Web Services +1 191

- Introduction
- Directions API
- Distance Matrix API
- Elevation API
- Geocoding API
- Time Zone API
- Blog
- Forum
- FAQ

The Google Geocoding API

- What is Geocoding?
- Audience
- Usage Limits
- Geocoding Requests
- Geocoding Responses
  - JSON Output Formats
  - XML Output Formats
  - Status Codes
  - Results
  - Address Component Types
- Reverse Geocoding
- Viewport Biasing
- Region Biasing
- Component Filtering

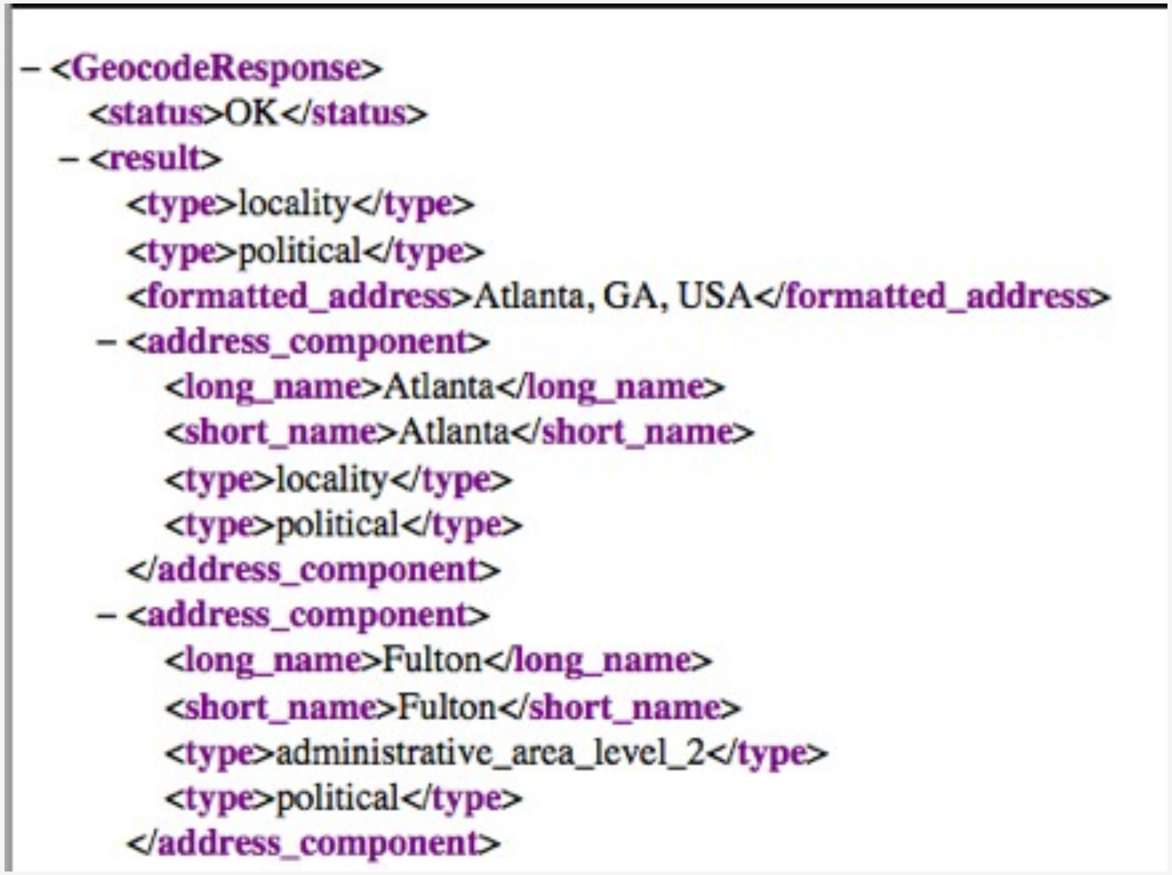
Workspace - XML

R can also process XML files from the Internet, which is very powerful. As an example we'll access Google's GeoCoding pages to get the Latitude and Longitude for Atlanta, GA from within a program. Its just a simple example.

```
library(RCurl)
library(XML)

my.url = "http://maps.googleapis.com/maps/api/geocode/xml?
address=Atlanta,GA&sensor=false"

txt = getURL(my.url)
hold = xmlTreeParse(txt,useInternalNodes=TRUE)

hold
<?xml version="1.0" encoding="UTF-8"?>
<GeocodeResponse>
  <status>OK</status>
  <result>
    <type>locality</type>

place = getNodeSet(hold,"//GeocodeResponse/result[1]/geometry/location[1]/*")
as.numeric(sapply(place,xmlValue))
[1]  33.74900 -84.38798
```

Workspace - XML

R can also process XML files from the Internet, which is very powerful. As an example we'll access Google's GeoCoding pages to get the Latitude and Longitude for Atlanta, GA from within a program. Its just a simple example.



```
- <GeocodeResponse>
    <status>OK</status>
  - <result>
      <type>locality</type>
      <type>political</type>
      <formatted_address>Atlanta, GA, USA</formatted_address>
    - <address_component>
        <long_name>Atlanta</long_name>
        <short_name>Atlanta</short_name>
        <type>locality</type>
        <type>political</type>
      </address_component>
    - <address_component>
        <long_name>Fulton</long_name>
        <short_name>Fulton</short_name>
        <type>administrative_area_level_2</type>
        <type>political</type>
      </address_component>
```

Workspace - XML

The interesting thing is that we can then use this information to query other resources. Say for example we can look at the Personal Weather Stations Web site and get quick assessment of the weather in the area based on a survey of weather stations.

```
my.url = "http://api.wunderground.com/auto/wui/geo/GeoLookupXML/index.xml?
query=33.749,-84.38798"

myweather = getURL(url)
hold = xmlTreeParse(myweather, useInternalNodes=TRUE)


qstr = "//location/nearby_weather_stations/pws/station[distance_mi<5]/
ancestor-or-self::station/id"

stationsXml = getNodeSet(hold,qstr)

stations = sapply(stationsXml,xmlValue)

> stations
[1] "KGAATLAN40" "KGAATLAN49" "KGAATLAN37" "KGAATLAN68" "KGAATLAN57"
[6] "KGAATLAN54" "KGAATLAN16"
```