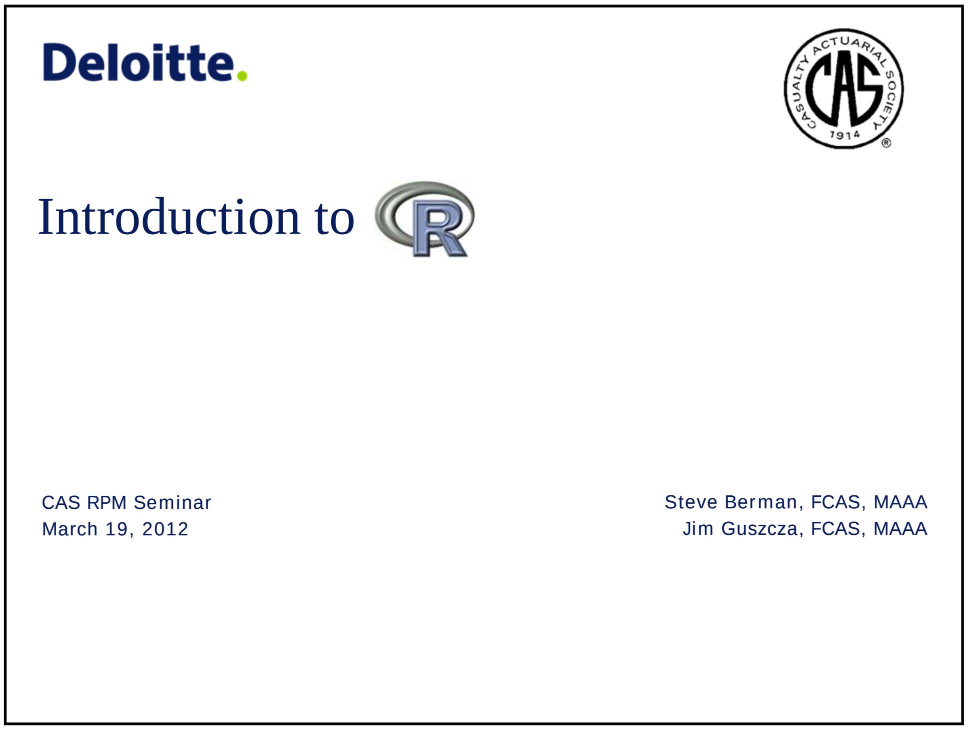
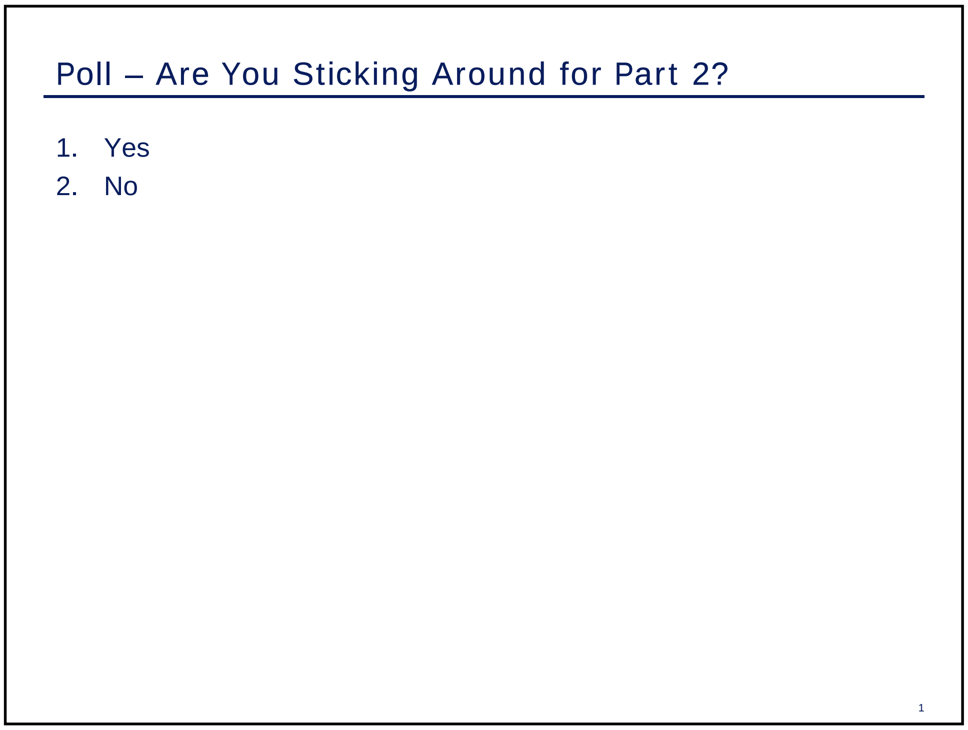


Deloitte.

Introduction to R

CAS RPM Seminar
March 19, 2012

Steve Berman, FCAS, MAAA
Jim Guszcza, FCAS, MAAA



Poll – Are You Sticking Around for Part 2?

1. Yes
2. No

1

Poll – How Much Do You Know About R?

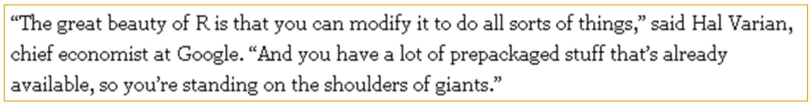
1. Isn’t that the 16th letter of the alphabet?
2. Something – I just installed it….
3. Spent a little time, looking for more
4. Occasional User
5. Power User (e.g. Jim Guscsza!)

R Background

R Background

R is an open-source, object-oriented statistical programming language

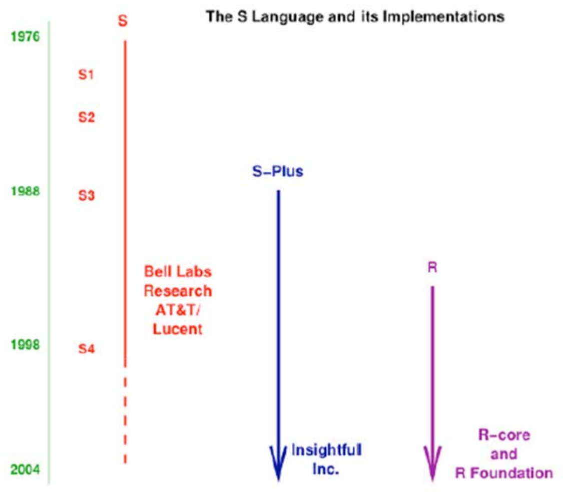
- History:
  - R is based on the S statistical programming language developed by John Chambers at Bell Labs in the 1980's
  - The commercial package S-plus is based on the S language
  - R is an open-source implementation of the S language
  - Developed by Robert Gentlemen and Ross Inhaka in New Zealand
  - At some point rewritten in C
- Features:
  - R is a high-level, object-oriented programming environment
  - R has advanced graphical capabilities
  - Statisticians around the world contribute add-on packages… therefore:


"The great beauty of R is that you can modify it to do all sorts of things," said Hal Varian, chief economist at Google. "And you have a lot of prepackaged stuff that's already available, so you're standing on the shoulders of giants."

4

R Evolution

- S is the original language
- S-plus is a commercial implementation of S
- R is an open-source implementation of S
- R is very similar to, but not identical with, other implementations of S



5

Facets of R

- In a recent article John Chambers discussed 6 “Facets of R”
  1. An *interface* to computational <u>procedures</u> of many kinds
  2. *Interactive*, hands-on in real time
  3. *Functional* in its model of programming
  4. *Object-oriented*, “everything is an object”
  5. *Modular*, built from standardized pieces
  6. *Collaborative*, a world-wide, open-source effort

- Interactive interface: Chambers was influenced by APL
  - One of the rare interactive scientific computing environments
  - Gives user ability to express novel computations
  - Heavy emphasis on matrices and arrays
  - <u>But</u>: unlike R, APL had no interface to procedures

- In the days before spreadsheets, APL was very popular in the actuarial community

“Facets of R”, John M. Chambers, *The R Journal* Vol. 1/1, May 2009

Modular and Collaborative: A Network ExteRnality

- Hal Varian’s “giant” has grown at an exponential rate.

- The open-source nature of R has encouraged top researchers from around the world to contribute new, often highly advanced, packages.

- Result: a powerful “network effect”.
  - The value of a product increases as more people use it.

- R has become something like the Wikipedia of the statistics world.

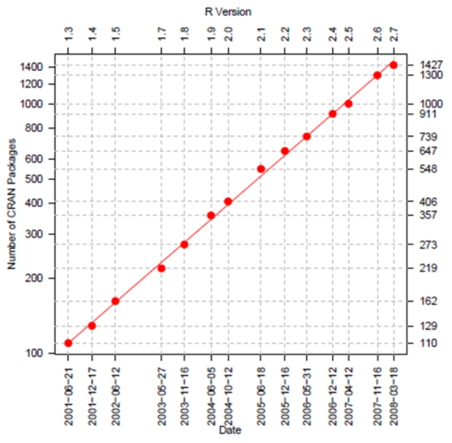
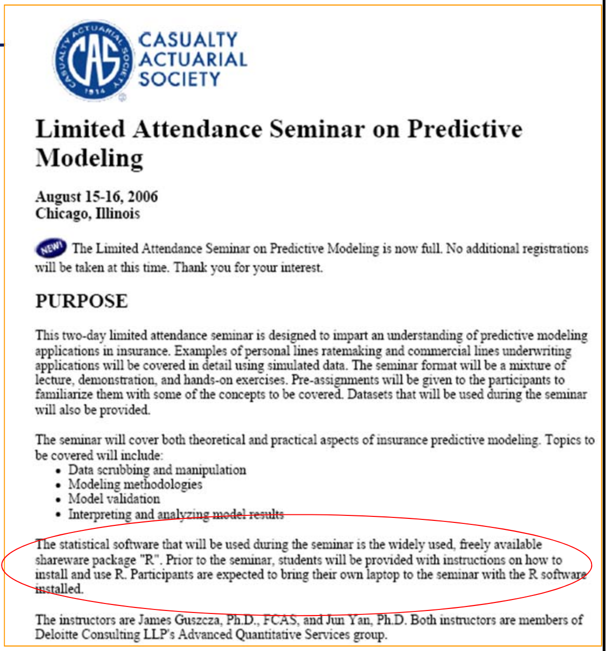


Figure 1: The number of R packages on CRAN has grown exponentially since R version 1.3 in 2001. *Source of Data*: `https://svn.r-project.org/R/branches/`.

Growing interest in R

- August 2006



CASUALTY ACTUARIAL SOCIETY

Limited Attendance Seminar on Predictive Modeling

August 15-16, 2006
Chicago, Illinois

NEW! The Limited Attendance Seminar on Predictive Modeling is now full. No additional registrations will be taken at this time. Thank you for your interest.

PURPOSE

This two-day limited attendance seminar is designed to impart an understanding of predictive modeling applications in insurance. Examples of personal lines ratemaking and commercial lines underwriting applications will be covered in detail using simulated data. The seminar format will be a mixture of lecture, demonstration, and hands-on exercises. Pre-assignments will be given to the participants to familiarize them with some of the concepts to be covered. Datasets that will be used during the seminar will also be provided.

The seminar will cover both theoretical and practical aspects of insurance predictive modeling. Topics to be covered will include:

- Data scrubbing and manipulation
- Modeling methodologies
- Model validation
- Interpreting and analyzing model results

The statistical software that will be used during the seminar is the widely used, freely available shareware package "R". Prior to the seminar, students will be provided with instructions on how to install and use R. Participants are expected to bring their own laptop to the seminar with the R software installed.

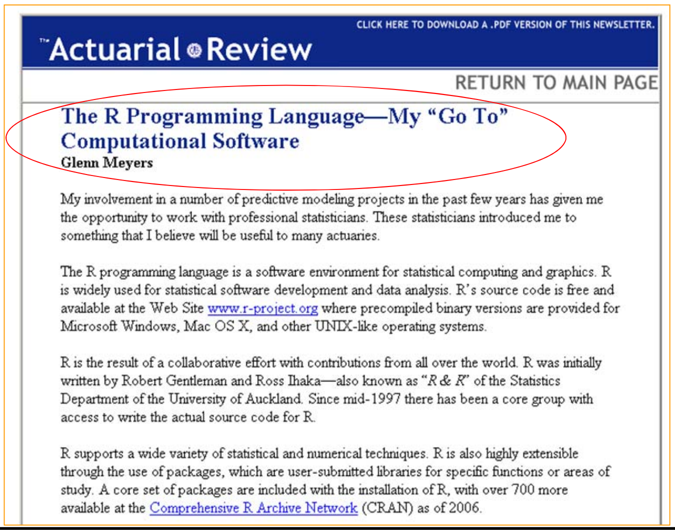
The instructors are James Guszcza, Ph.D., FCAS, and Jun Yan, Ph.D. Both instructors are members of Deloitte Consulting LLP's Advanced Quantitative Services group.

8

Growing interest in R

- November 2006

http://www.casact.org/newsletter/index.cfm?fa=viewart&id=5311



CLICK HERE TO DOWNLOAD A .PDF VERSION OF THIS NEWSLETTER.

Actuarial Review

RETURN TO MAIN PAGE

The R Programming Language—My "Go To" Computational Software
Glenn Meyers

My involvement in a number of predictive modeling projects in the past few years has given me the opportunity to work with professional statisticians. These statisticians introduced me to something that I believe will be useful to many actuaries.

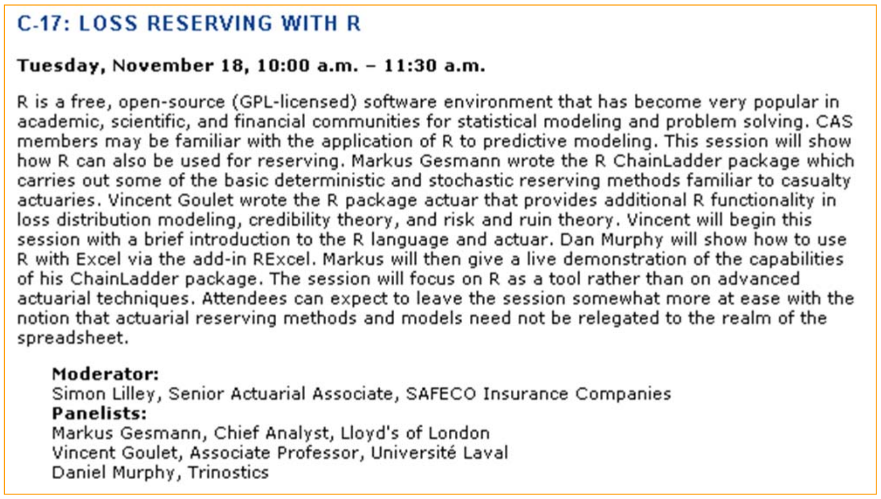
The R programming language is a software environment for statistical computing and graphics. R is widely used for statistical software development and data analysis. R's source code is free and available at the Web Site www.r-project.org where precompiled binary versions are provided for Microsoft Windows, Mac OS X, and other UNIX-like operating systems.

R is the result of a collaborative effort with contributions from all over the world. R was initially written by Robert Gentleman and Ross Ihaka—also known as "*R & R*" of the Statistics Department of the University of Auckland. Since mid-1997 there has been a core group with access to write the actual source code for R.

R supports a wide variety of statistical and numerical techniques. R is also highly extensible through the use of packages, which are user-submitted libraries for specific functions or areas of study. A core set of packages are included with the installation of R, with over 700 more available at the Comprehensive R Archive Network (CRAN) as of 2006.

Growing interest in R

• November 2008 – CAS Annual Meeting, Seattle



C-17: LOSS RESERVING WITH R

Tuesday, November 18, 10:00 a.m. – 11:30 a.m.

R is a free, open-source (GPL-licensed) software environment that has become very popular in academic, scientific, and financial communities for statistical modeling and problem solving. CAS members may be familiar with the application of R to predictive modeling. This session will show how R can also be used for reserving. Markus Gesmann wrote the R ChainLadder package which carries out some of the basic deterministic and stochastic reserving methods familiar to casualty actuaries. Vincent Goulet wrote the R package actuar that provides additional R functionality in loss distribution modeling, credibility theory, and risk and ruin theory. Vincent will begin this session with a brief introduction to the R language and actuar. Dan Murphy will show how to use R with Excel via the add-in RExcel. Markus will then give a live demonstration of the capabilities of his ChainLadder package. The session will focus on R as a tool rather than on advanced actuarial techniques. Attendees can expect to leave the session somewhat more at ease with the notion that actuarial reserving methods and models need not be relegated to the realm of the spreadsheet.

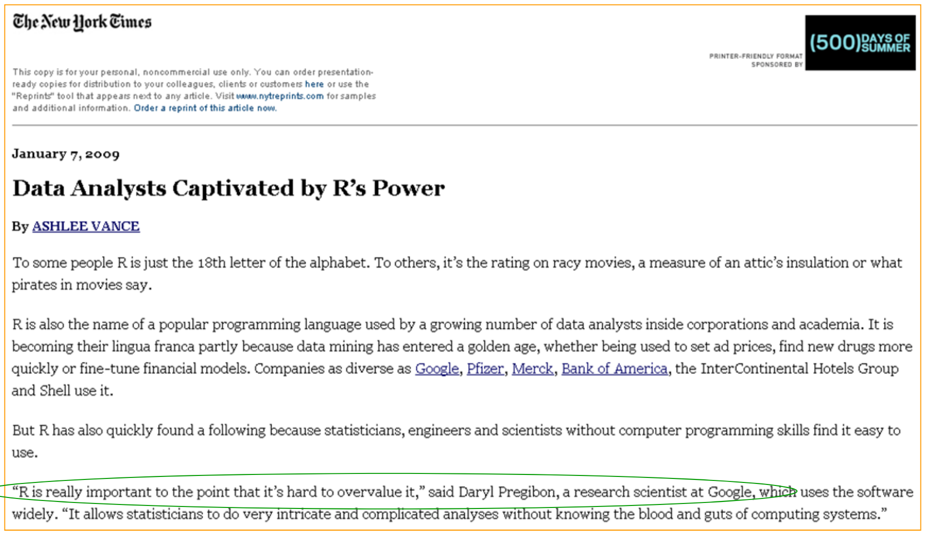
Moderator:
Simon Lilley, Senior Actuarial Associate, SAFECO Insurance Companies
Panelists:
Markus Gesmann, Chief Analyst, Lloyd's of London
Vincent Goulet, Associate Professor, Université Laval
Daniel Murphy, Trinostics

10

Growing interest in R

• January 2009

http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=1&pagewanted=print



The New York Times

PRINTER-FRIENDLY FORMAT SPONSORED BY (500) DAYS OF SUMMER

This copy is for your personal, noncommercial use only. You can order presentation-ready copies for distribution to your colleagues, clients or customers here or use the "Reprints" tool that appears next to any article. Visit www.nytreprints.com for samples and additional information. Order a reprint of this article now.

January 7, 2009

Data Analysts Captivated by R's Power

By ASHLEE VANCE

To some people R is just the 18th letter of the alphabet. To others, it's the rating on racy movies, a measure of an attic's insulation or what pirates in movies say.

R is also the name of a popular programming language used by a growing number of data analysts inside corporations and academia. It is becoming their lingua franca partly because data mining has entered a golden age, whether being used to set ad prices, find new drugs more quickly or fine-tune financial models. Companies as diverse as Google, Pfizer, Merck, Bank of America, the InterContinental Hotels Group and Shell use it.

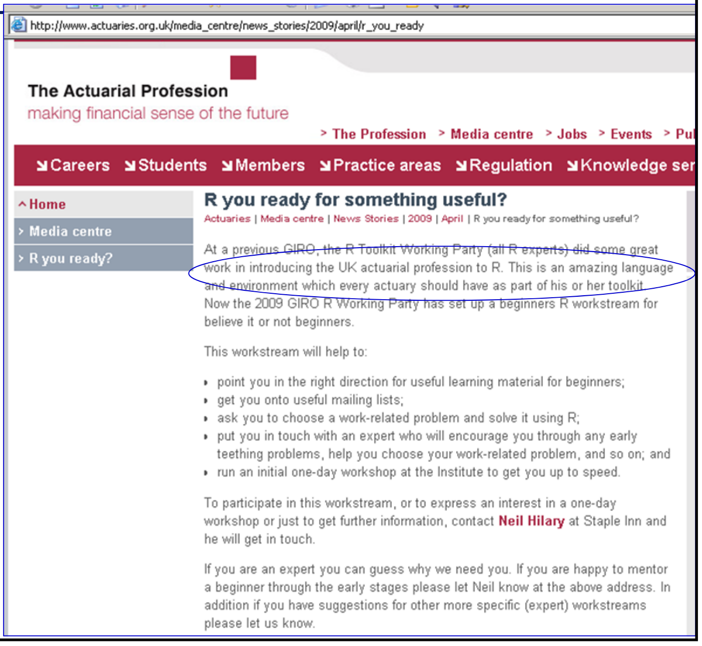
But R has also quickly found a following because statisticians, engineers and scientists without computer programming skills find it easy to use.

"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."

Growing interest in R

- April 2009
  http://www.actuaries.org.uk/media_centre/news_stories/2009/april/r_you_ready

- **Interest in the UK actuarial community**


http://www.actuaries.org.uk/media_centre/news_stories/2009/april/r_you_ready

The Actuarial Profession
making financial sense of the future

> The Profession > Media centre > Jobs > Events > Pu

Careers Students Members Practice areas Regulation Knowledge ser

Home
Media centre
R you ready?

R you ready for something useful?
Actuaries | Media centre | News Stories | 2009 | April | R you ready for something useful?

At a previous GIRO, the R Toolkit Working Party (all R experts) did some great work in introducing the UK actuarial profession to R. This is an amazing language and environment which every actuary should have as part of his or her toolkit. Now the 2009 GIRO R Working Party has set up a beginners R workstream for believe it or not beginners.

This workstream will help to:

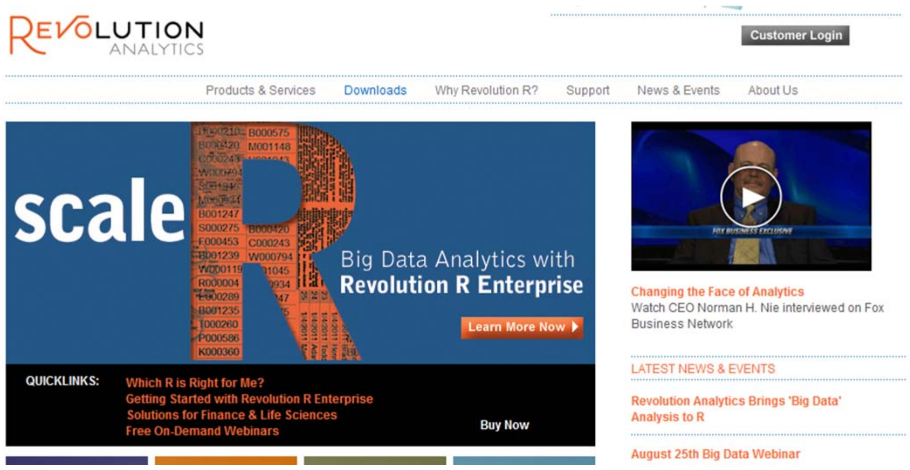
- point you in the right direction for useful learning material for beginners;
- get you onto useful mailing lists;
- ask you to choose a work-related problem and solve it using R;
- put you in touch with an expert who will encourage you through any early teething problems, help you choose your work-related problem, and so on; and
- run an initial one-day workshop at the Institute to get you up to speed.

To participate in this workstream, or to express an interest in a one-day workshop or just to get further information, contact Neil Hilary at Staple Inn and he will get in touch.

If you are an expert you can guess why we need you. If you are happy to mentor a beginner through the early stages please let Neil know at the above address. In addition if you have suggestions for other more specific (expert) workstreams please let us know.

On to Bigger Things?

- A company that aspires to be to R what Redhat is to Linux
  - Enterprise versions of R



Big Data Analytics with Revolution R Enterprise

QUICKLINKS: Which R is Right for Me?
Getting Started with Revolution R Enterprise
Solutions for Finance & Life Sciences
Free On-Demand Webinars

Changing the Face of Analytics
Watch CEO Norman H. Nie interviewed on Fox Business Network

LATEST NEWS & EVENTS
Revolution Analytics Brings 'Big Data' Analysis to R
August 25th Big Data Webinar

13

Installing R

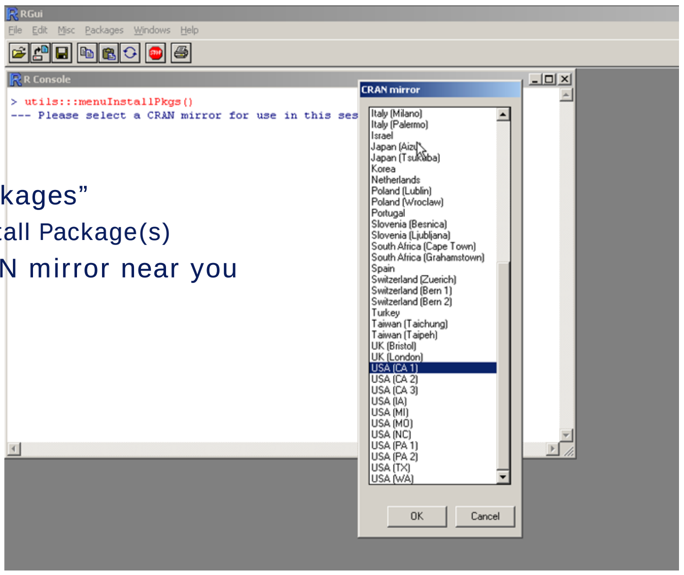
- Go to http://cran.r-project.org/
    - Or just type "R" into Google and click "I feel lucky"
- Click on "Download CRAN" on the left of the screen
- Click on one of the USA CRAN mirror sites
- Click on "Windows (95 and later)"
- Click on "base"
- Right-click on R-2.14.1-win32.exe (or latest version)
    - "Save target as" into any directory

- After you've downloaded this setup program, double-click on it and follow the instructions

- For those with permissions issues, follow the instructions at http://personal.bgsu.edu/~mrizzo/Rmisc/usbR.htm to install on a flash drive

14

Add-on Packages



```
> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this ses
```

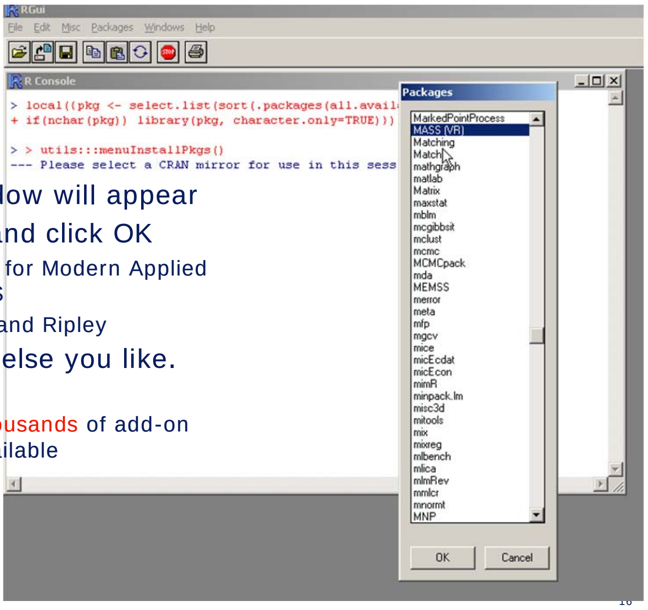
CRAN mirror

Italy (Milano)
Italy (Palermo)
Israel
Japan (Aizu)
Japan (Tsukuba)
Korea
Netherlands
Poland (Lublin)
Poland (Wroclaw)
Portugal
Slovenia (Besnica)
Slovenia (Ljubljana)
South Africa (Cape Town)
South Africa (Grahamstown)
Spain
Switzerland (Zuerich)
Switzerland (Bern 1)
Switzerland (Bern 2)
Turkey
Taiwan (Taichung)
Taiwan (Taipeh)
UK (Bristol)
UK (London)
USA (CA 1)
USA (CA 2)
USA (CA 3)
USA (IA)
USA (MI)
USA (MO)
USA (NC)
USA (PA 1)
USA (PA 2)
USA (TX)
USA (WA)

- Click on "Packages"
    - Select "Install Package(s)
- Select a CRAN mirror near you

15

Add-on Packages



```
> local({pkg <- select.list(sort(.packages(all.avail
+ if(nchar(pkg)) library(pkg, character.only=TRUE)})

> > utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this sess
```

- “Packages” window will appear
- Select “MASS” and click OK
  - MASS stands for Modern Applied Statistics in S
  - By Venables and Ripley
- … add anything else you like.
  - It’s all free
  - There are thousands of add-on packages available

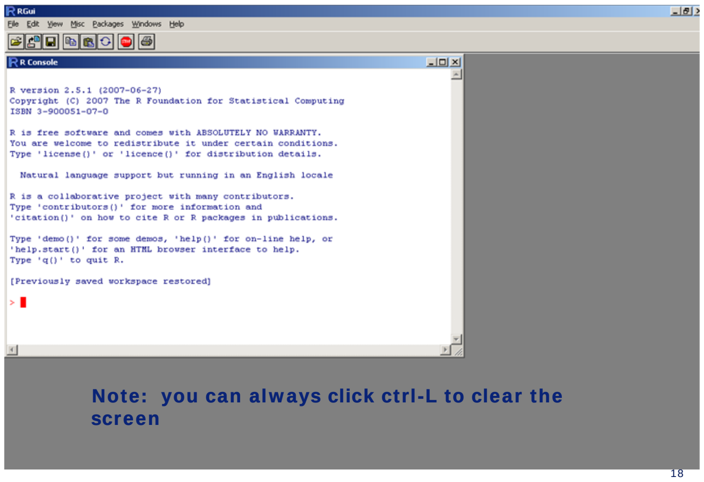
R – Basic Elements

RGui
Executing code
Functions
Assignments
Getting Help

Vectors
Matrices
Data Frames
Controls

Getting Started with R

- Double-click on the “R” icon to start the program
- You will see the Console screen. Code can be typed in here and run immediately



```
R version 2.5.1 (2007-06-27)
Copyright (C) 2007 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

>
```

Note: you can always click ctrl-L to clear the screen

18

R Basics - Packages

- Test to see whether your additional libraries were successfully added.
  - Type “library(MASS)”
    - library function loads in installed package into your current R session
    - All elements of package available until session closed
      - Note: R is case-sensitive!
  - If there are no error messages you’re ok
  - Type “library()” to see list of currently installed packages

19

R Basics – Command Line

- This screen gives you the “command line”.
  - Type commands at the red “>”
- You can use R as a calculator using standard operators
  - Type “2+3” at the command line and hit enter
  - Similarly “2-3”, “2*3”, “2/3”, “2^3” (or “2**3”)
- Use UP arrow at prompt to bring back previously submitted lines

Scripts

- Entering in codes one line at a time gets tiring! And not very reusable, either
- Scripts allow you to save code and load later
- Select File / New script to bring up a scripting window, and start entering code
- Use Windows to flip between scripts and console, or Tile them both on screen
- Can run single lines of code, blocks of code, or entire scripts
- Ctrl-L, Ctrl-A, Ctrl-R combo (clear, select all, run)

Interactive vs. Batch Mode

- At least three ways to run R
- Executing code from the Console Window or from a script is "Interactive Mode"
  - Only one stream can be running at a time
  - Lots of flexibility in what you want to run and the order
  - Can get intermediate results
  - Good when debugging
- Can run from a Command prompt as well or a batch file ("Batch Mode")
  - Useful if you know program will run correctly
  - Have multiple files processing at same time
  - R CMD BATCH *filename*
  - Output is saved to .Rout file

22

Functions and Statements

- R has a wide array of functions, both in the base load set and the packages. Some numeric functions:

abs	absolute value
log	natural logarithm
log10	base 10 logarithm
%%	modulus
%/%	integer division
floor	get lowest integer
ceiling	get highest integer
max	maximum
min	minimum

- Functions are called similar to Excel
  - Ex: `abs`(-3.5) (returns 3.5)
- Functions can take in any number of parameters but return at most a single object
- Some functions have optional parameters – can enter in parameters in order they are defined or refer to them by name
- Statements have similar syntax but do not return a result

23

String Functions

- `cat` – catenates and prints vector of strings
- `paste` – converts to characters and catenates
- `tolower`, `toupper` – case conversion

24

Help

- Don't exactly know the parameters for a function, or what it does? Want to do something but don't know the function? Get help!
- At console window, type "?" followed by function name, or use the help menu
  - Ex: "?summary", or "help(summary)"
- Use "??" followed by keyword to do search
  - Ex: "??regression"
  - Or try searching Google ("R linear regression")

25

Comments, Whitespace, etc.

- Code can span multiple lines
- Code can have white space, indentations, etc.
- Hash (#) comments out the rest of the line
- There is no multiple line comment in R (like /* */ construct in C or SAS

26

Assignments

- Suppose you want to set the variable *x* to equal 5
- Type “x <- 5” (Combine the less than sign “<“ and the minus sign “-”)
  - Also:
    - x=5
    - 5 -> x
    - assign(‘x’, 5)
- In words: “*x* gets 5”
- Now type “x” at the command line
- Now type “objects()”
  - x has been saved as an R object
- Equivalent is ls() (“list”, like Unix command)
- Now type “rm(x)” (“remove”)
  - To remove the object x if we’re done with it
- Now type “objects()” again
  - The object x is gone

27

Knowledge check – which sets x to 8?

1. x<- 2 + 2 * 2
2. assign(8, x)
3. x -> 8
4. x = 8

Workspaces

- Scripts allow you to store code, <u>not</u> data
  - Use .R suffix
- All data is stored in a single area called the workspace
- Workspace contains all variables as well as functions that have been created or loaded
  - Use File / Load Workspace, File / Save Workspace
  - Stores data and also loaded function definitions
  - Uses .RData suffix
- Because all data is in memory at the same time, you need to be careful with what variables are saved – it is not hard to run out of memory, depending on your system resources

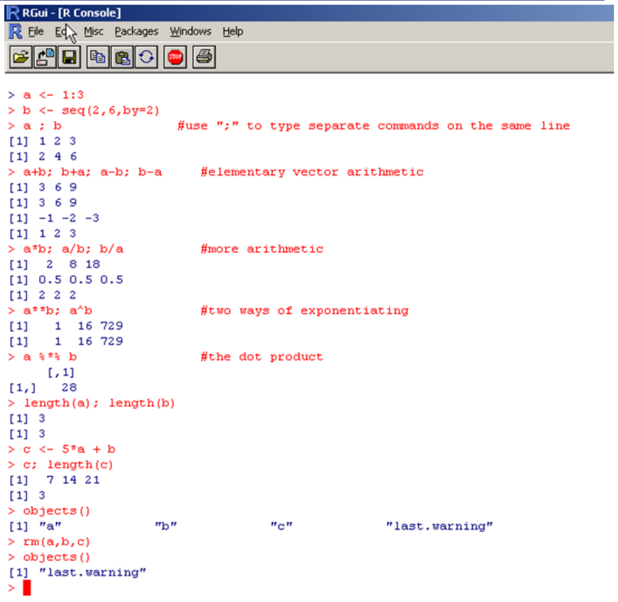
R Basics - Vectors

- A vector is a sequence of elements of the same type
- R handles vectors very naturally.
  - Type “c(1,2,3,4,5)” at the command line and hit enter
  - “c” stands for “concatenate”
  - This is how to create a vector of numbers
  - Alternately:
    - Type “1:5”
    - Type “seq(1,5)”
- Note: do not have to declare or dimension variables

30

Working with Vectors

- R handles vectors very naturally
- Type these commands into your R session to gain comfort.

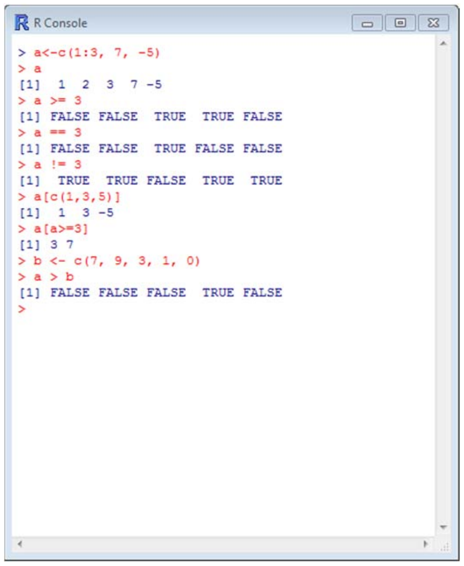


```
> a <- 1:3
> b <- seq(2,6,by=2)
> a ; b                   #use ";" to type separate commands on the same line
[1] 1 2 3
[1] 2 4 6
> a+b; b+a; a-b; b-a      #elementary vector arithmetic
[1] 3 6 9
[1] 3 6 9
[1] -1 -2 -3
[1] 1 2 3
> a*b; a/b; b/a           #more arithmetic
[1]  2  8 18
[1] 0.5 0.5 0.5
[1] 2 2 2
> a**b; a^b               #two ways of exponentiating
[1]   1  16 729
[1]   1  16 729
> a %*% b                 #the dot product
     [,1]
[1,]   28
> length(a); length(b)
[1] 3
[1] 3
> c <- 5*a + b
> c; length(c)
[1]  7 14 21
[1] 3
> objects()
[1] "a"            "b"            "c"            "last.warning"
> rm(a,b,c)
> objects()
[1] "last.warning"
> 
```

31

Filtering on Vectors

- Reference individual elements of vector using brackets
- Can use integer elements or boolean conditions



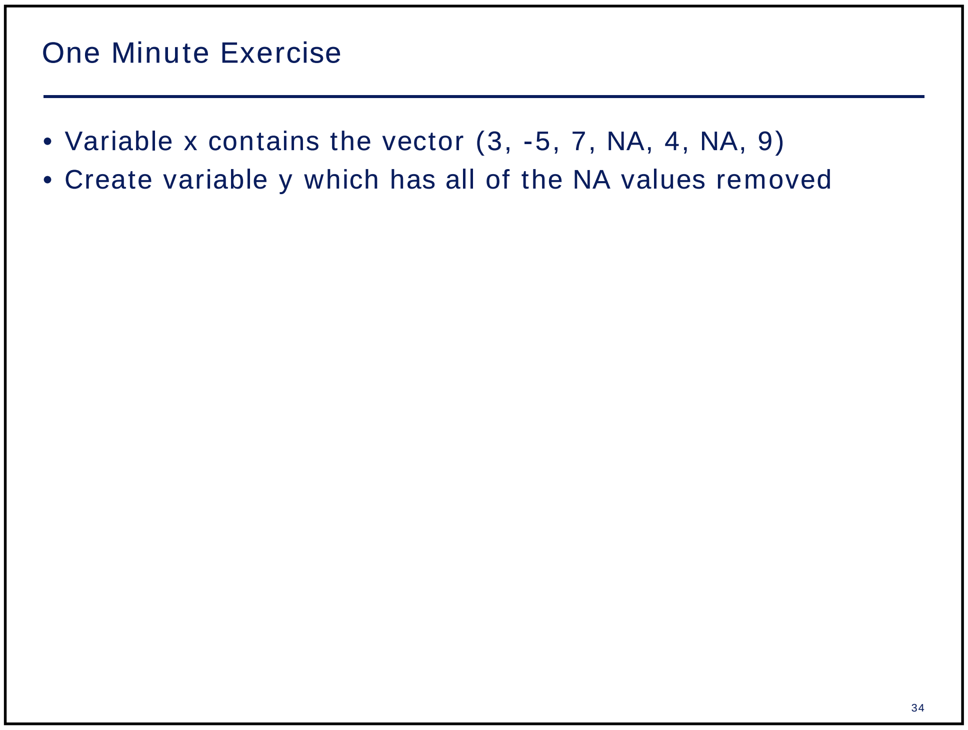
```
> a<-c(1:3, 7, -5)
> a
[1]  1  2  3  7 -5
> a >= 3
[1] FALSE FALSE  TRUE  TRUE FALSE
> a == 3
[1] FALSE FALSE  TRUE FALSE FALSE
> a != 3
[1]  TRUE  TRUE FALSE  TRUE  TRUE
> a[c(1,3,5)]
[1]  1  3 -5
> a[a>=3]
[1] 3 7
> b <- c(7, 9, 3, 1, 0)
> a > b
[1] FALSE FALSE FALSE  TRUE FALSE
>
```

32

Special Values and Coercion

- NA is the R version of a missing value
  - Missing values as any part of an operand generally return missing values (ex: 3 + NA = NA)
  - Can test for missing values with is.na() function
- Similarly, NULL is a reserved word for an undefined object
- NaN = Not a Number (usually math error)
- Inf = infinity
- Can change the type of a variable using functions like `as.integer`, `as.double`, `as.vector`, etc.

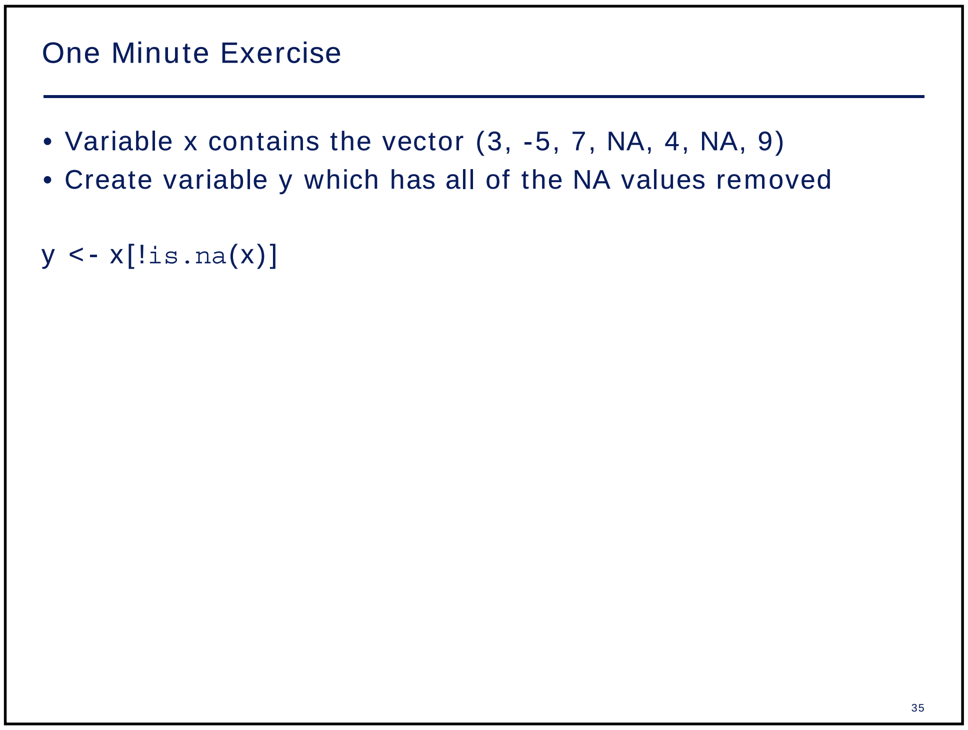
33



One Minute Exercise

- Variable x contains the vector (3, -5, 7, NA, 4, NA, 9)
- Create variable y which has all of the NA values removed

34



One Minute Exercise

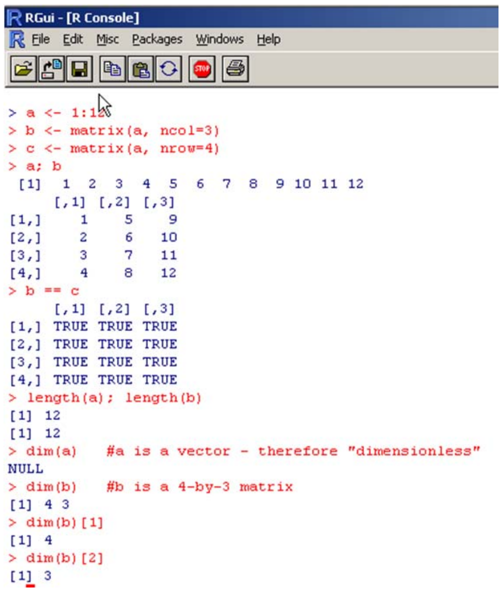
- Variable x contains the vector (3, -5, 7, NA, 4, NA, 9)
- Create variable y which has all of the NA values removed

```
y <- x[!is.na(x)]
```

35

Working with Matrices

- A matrix is an 2-dimensional array
- This screen illustrates how to create a matrix from a vector
- Vectors have length, matrices have dimension
- Use array() function if 2 dimensions not enough…

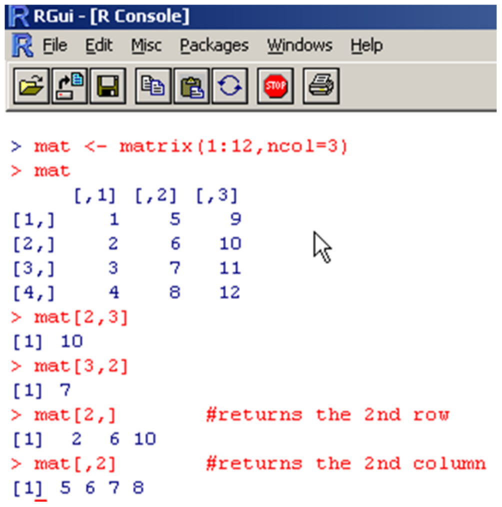


```
> a <- 1:12
> b <- matrix(a, ncol=3)
> c <- matrix(a, nrow=4)
> a; b
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> b == c
     [,1] [,2] [,3]
[1,] TRUE TRUE TRUE
[2,] TRUE TRUE TRUE
[3,] TRUE TRUE TRUE
[4,] TRUE TRUE TRUE
> length(a); length(b)
[1] 12
[1] 12
> dim(a)    #a is a vector - therefore "dimensionless"
NULL
> dim(b)    #b is a 4-by-3 matrix
[1] 4 3
> dim(b)[1]
[1] 4
> dim(b)[2]
[1] 3
```

36

Working with Matrices

- R is designed to handle matrices naturally
- The bracket notation "mat[row, column]" allows you to access any element of a matrix
- Notice what happens when you leave the row or column entry blank

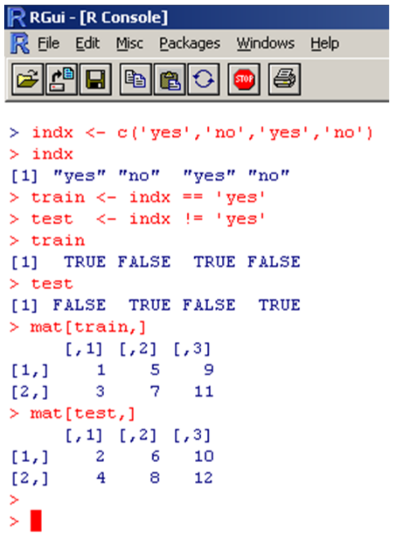


```
> mat <- matrix(1:12,ncol=3)
> mat
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> mat[2,3]
[1] 10
> mat[3,2]
[1] 7
> mat[2,]        #returns the 2nd row
[1]  2  6 10
> mat[,2]        #returns the 2nd column
[1] 5 6 7 8
```

37

Working with Matrices

- We can get fancier by creating an index.
  - Let's use an index to divide the matrix into disjoint sets of rows.
- Think about how this trick can be used in predictive modeling projects.
  - We divide a dataset either by a random number or some other dimension.

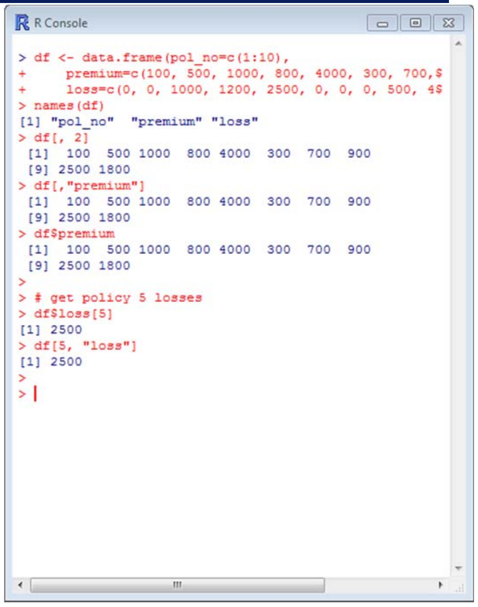


```
> indx <- c('yes','no','yes','no')
> indx
[1] "yes" "no"  "yes" "no"
> train <- indx == 'yes'
> test  <- indx != 'yes'
> train
[1]  TRUE FALSE  TRUE FALSE
> test
[1] FALSE  TRUE FALSE  TRUE
> mat[train,]
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11
> mat[test,]
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    4    8   12
>
> 
```

38

Data Frames

- A data frame is a matrix-like structure whose columns may be of differing types (numeric, logical, factor, character, etc.)
- Like an Excel table or a SAS dataset
- Columns have names
- All of the matrix functions apply to data frames
- Also can reference the columns by their names (*data_frame*$*col_name*)



```
> df <- data.frame(pol_no=c(1:10),
+     premium=c(100, 500, 1000, 800, 4000, 300, 700,$
+     loss=c(0, 0, 1000, 1200, 2500, 0, 0, 0, 500, 4$
> names(df)
[1] "pol_no"  "premium" "loss"
> df[, 2]
 [1]  100  500 1000  800 4000  300  700  900
 [9] 2500 1800
> df[,"premium"]
 [1]  100  500 1000  800 4000  300  700  900
 [9] 2500 1800
> df$premium
 [1]  100  500 1000  800 4000  300  700  900
 [9] 2500 1800
>
> # get policy 5 losses
> df$loss[5]
[1] 2500
> df[5, "loss"]
[1] 2500
>
> |
```

39

Knowledge check

You have the following data frame (HairEye):

```
    Hair   Eye Freq
1  Black Brown   32
2  Brown Brown   53
3    Red Brown   10
4  Blond Brown    3
5  Black  Blue   11
6  Brown  Blue   50
7    Red  Blue   10
8  Blond  Blue   30
9  Black Hazel   10
10 Brown Hazel   25
11   Red Hazel    7
12 Blond Hazel    5
13 Black Green    3
14 Brown Green   15
15   Red Green    7
16 Blond Green    8
```

Which of these statements returns a different value?

A. HairEye[10, 3]

B. HairEye[10,]$Freq

C. HairEye[,3][10]

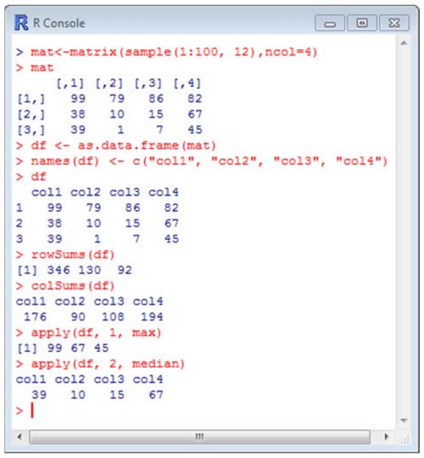
D. HairEye[HairEye$Hair==“Brown & HairEye$Eye==“Hazel,]$Freq

E. HairEye[3, “Freq”]

40

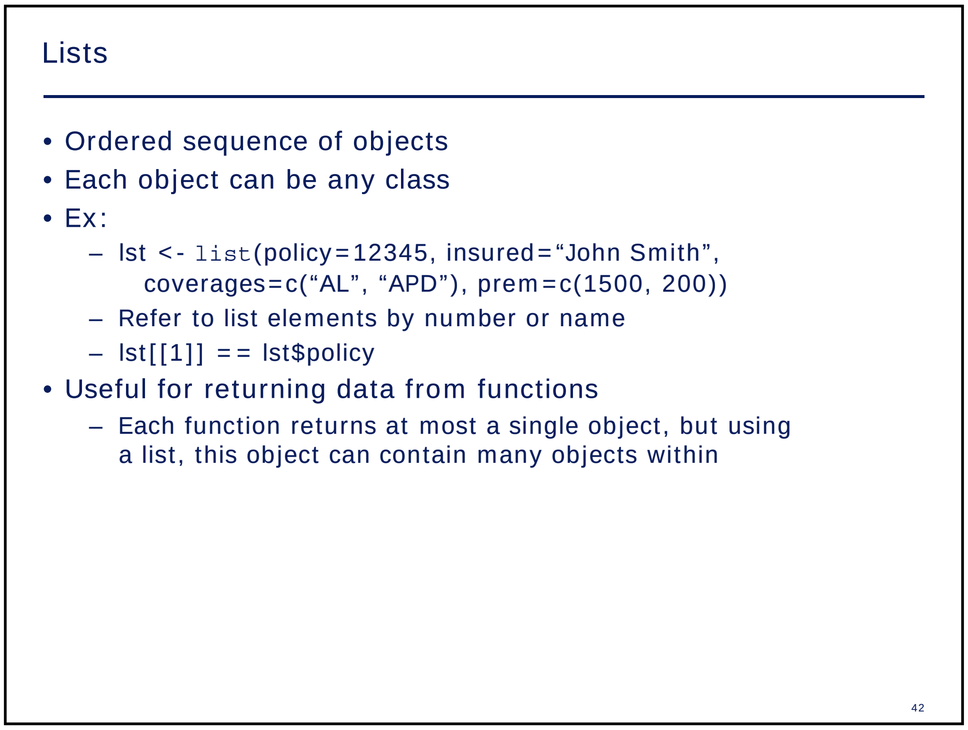
Data Frames

- Some common data manipulations:
  - `rbind` – combine two data frames by row
  - `cbind` – combine two data frames by column
  - `order` – determine order of records in a data frame – used for sorting
  - `merge` – combine two datasets across a common key
  - Methods to aggregate data
    - `rowsum`, `rowSums`, `colSums` – only perform sums
    - `aggregate` – allows different functions
    - `apply` – apply a function across rows or columns of data frame
    - `sapply` – apply a function across columns of data frame
    - `tapply` – apply function to a “ragged array”



```
> mat<-matrix(sample(1:100, 12),ncol=4)
> mat
     [,1] [,2] [,3] [,4]
[1,]   99   79   86   82
[2,]   38   10   15   67
[3,]   39    1    7   45
> df <- as.data.frame(mat)
> names(df) <- c("col1", "col2", "col3", "col4")
> df
  col1 col2 col3 col4
1   99   79   86   82
2   38   10   15   67
3   39    1    7   45
> rowSums(df)
[1] 346 130  92
> colSums(df)
col1 col2 col3 col4
 176   90  108  194
> apply(df, 1, max)
[1] 99 67 45
> apply(df, 2, median)
col1 col2 col3 col4
  39   10   15   67
>
```

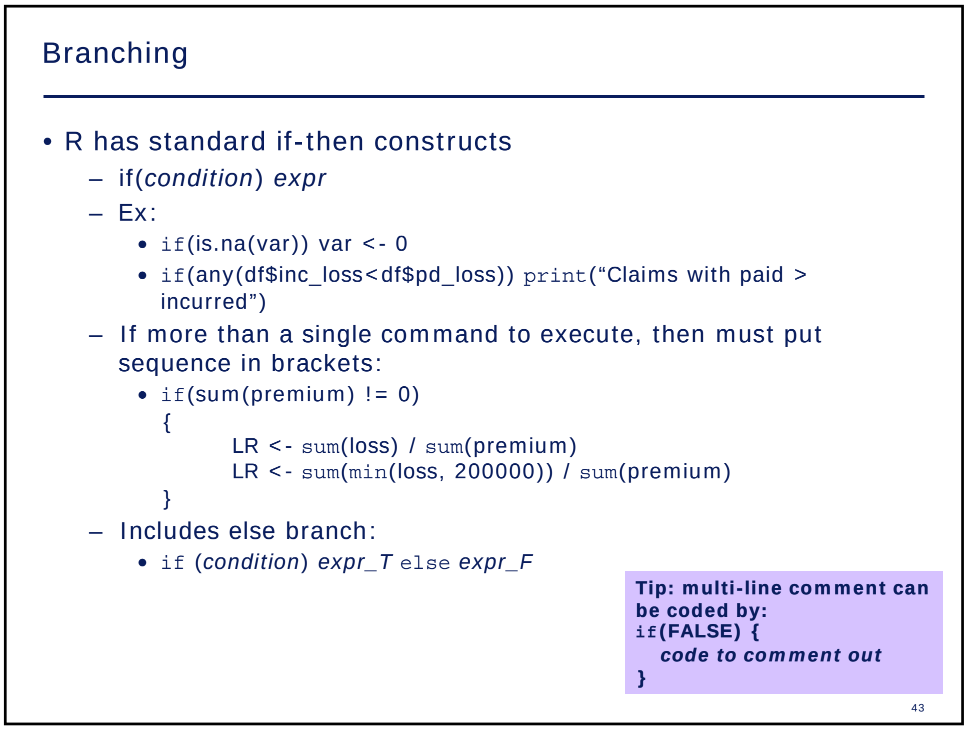
41



Lists

- Ordered sequence of objects
- Each object can be any class
- Ex:
  - lst <- `list`(policy=12345, insured="John Smith", coverages=c("AL", "APD"), prem=c(1500, 200))
  - Refer to list elements by number or name
  - lst[[1]] == lst$policy
- Useful for returning data from functions
  - Each function returns at most a single object, but using a list, this object can contain many objects within

42



Branching

- R has standard if-then constructs
  - if(*condition*) *expr*
  - Ex:
    - `if`(is.na(var)) var <- 0
    - `if`(any(df$inc_loss<df$pd_loss)) `print`("Claims with paid > incurred")
  - If more than a single command to execute, then must put sequence in brackets:

    -
      ```
      if(sum(premium) != 0)
      {
              LR <- sum(loss) / sum(premium)
              LR <- sum(min(loss, 200000)) / sum(premium)
      }
      ```
  - Includes else branch:
    - `if` (*condition*) *expr_T* `else` *expr_F*

Tip: multi-line comment can be coded by:
`if`(FALSE) {
  code to comment out
}

43

Looping

- For loops:
  - `for` (*name* `in` *expr_1*) *expr_2*
  - Ex: `for(i in 1:5) x <- x + df[i]`
  - Looping expression does not need to be evenly distributed
  - `for(j in c(1, 3, 6, 10)) print(sum(df[,j]))`
- Other loops:
  - `repeat` *expr*
  - `while` (*condition*) *expr*
- Note: avoid loops when not necessary! They are usually considerably slower to execute

44

User-Defined Functions

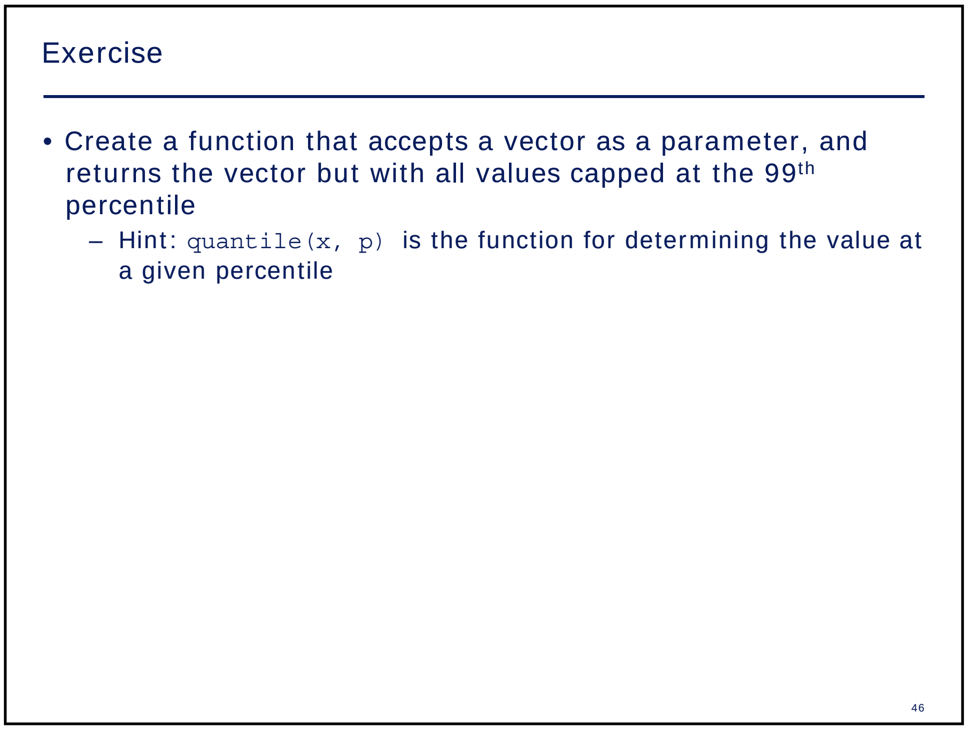
- It's easy to create functions to be used in programs
  - *function_name* <- `function`(parameters) {

    code

    `return`*(value)*

    }
  - Tip: save common functions in separate script, use source() function at top of script to include contents of a script in another script

45



Exercise

- Create a function that accepts a vector as a parameter, and returns the vector but with all values capped at the 99th percentile
  - Hint: `quantile(x, p)` is the function for determining the value at a given percentile

46