Topic # 00: Data Analytics

A Tutorial on the Use of R

Instructor: Prof. Arnab Bisi, Ph.D.

Johns Hopkins Carey Business School

R Introduction

- S is a statistical high-level and interpreted programming language developed at the Bell laboratories around 1975 by John Chambers. The commercial implementation of S is called S-PLUS and appeared in 1988.
- R is an open-source implementation of S and was created in the early nineties by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. These days, R is maintained by the R core team.
- R has become very popular particularly in academia but also in industry. Much of R's success story is due to all the packages written for R by the R-community.

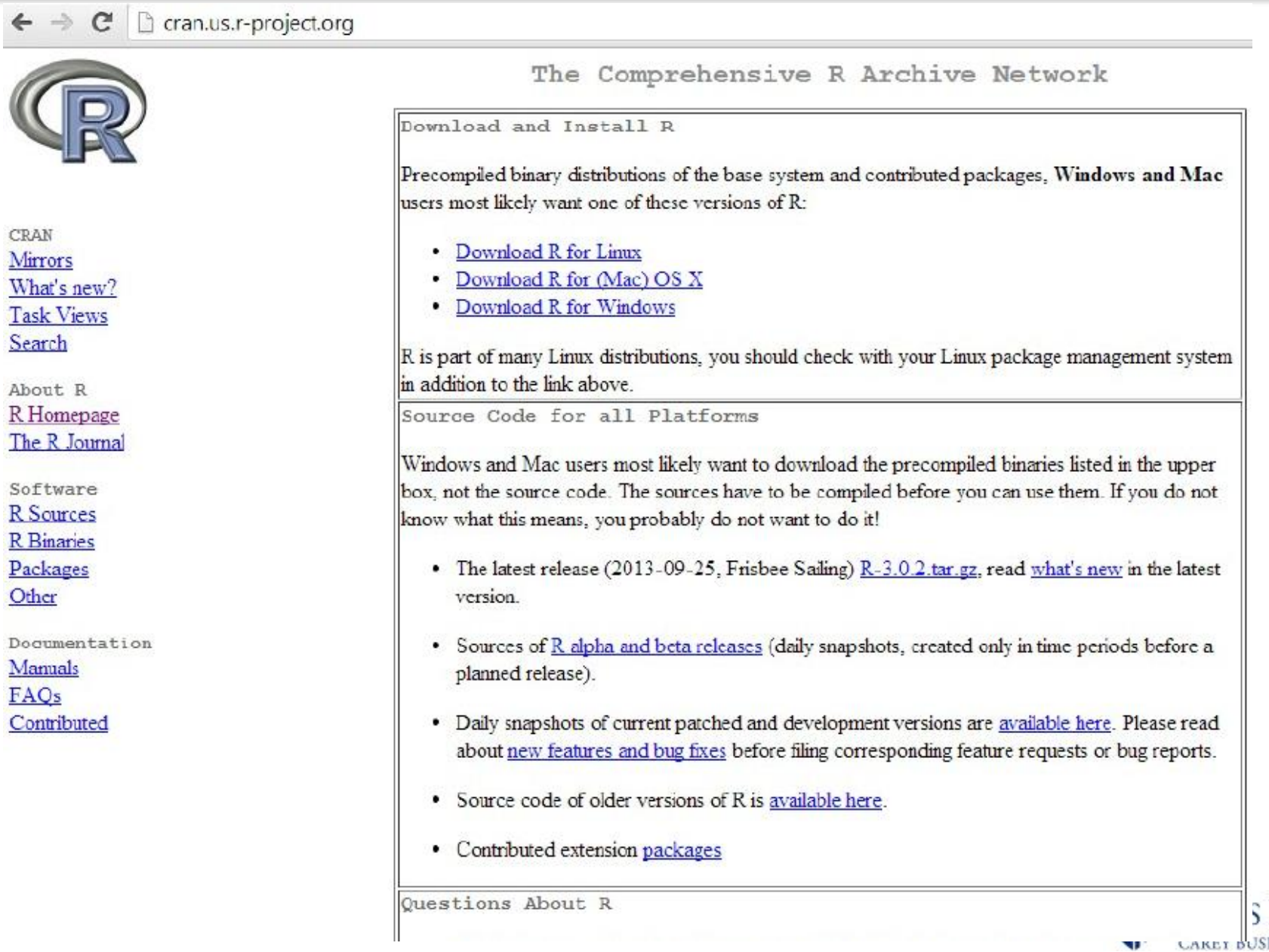
JOHNS HOPKINS
CAREY BUSINESS SCHOOL

What is R?

- A software package
- A programming language
- A toolkit for developing statistical and analytical tools
- An extensive library of statistical and mathematical software and algorithms
- A scripting language
- ...

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Install R


cran.us.r-project.org
The Comprehensive R Archive Network
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
Contributed
Download and Install R
Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R:
Download R for Linux
Download R for (Mac) OS X
Download R for Windows
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.
Source Code for all Platforms
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!
The latest release (2013-09-25, Frisbee Sailing) R-3.0.2.tar.gz, read what's new in the latest version.
Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
Source code of older versions of R is available here.
Contributed extension packages
Questions About R

Why R?

- R is free!
- R is cross-platform and runs on Windows, Mac, and Linux (as well as more obscure systems).
- R provides a vast number of useful statistical tools, many of which have been tested.
- R produces publication-quality graphics in a variety of formats. R plays well with FORTRAN, C, and scripts in many languages. R scales, making it useful for small and large projects.
- It is NOT Excel.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Why R-Studio?

- R Studio is free!
- R Studio is cross-platform and runs on Windows, and Mac, and looks the same on both platforms
- R Studio provides a vast number of useful tools, for the novice that handle pathways, input, and output and is consistent across platforms.
- R Studio makes it easy to experiment to find the right syntax

  and then save the correct command for later.
- R Studio helps handle the interface with Word.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Do We Still Need Excel?

- Excel is ”almost” free!
- Excel is cross-platform and runs on Windows, and Mac
- Excel provides a vast number of useful statistical tools, many of which have been tested and work well on “SMALL” data sets.
- Excel makes it easy to “see” your data
- Excel does a good job at sorting, using relative addresses, and plotting small data sets
- Excel can be VERY useful for data scrubbi
- It is NOT ideal for LARGE data sets.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

R commands:

```
> plot(rnorm(1000))
```

Assign a value to a variable:

```
> a=5
> b<-10
```

Simple math calculations

```
> a+b-a*b
```

The “ <− ” and “ = ” are both assignment operators.

The standard R prompt is a “ > ” sign.

If a line is not a complete R command, R will continue the next line with a “ + ”.

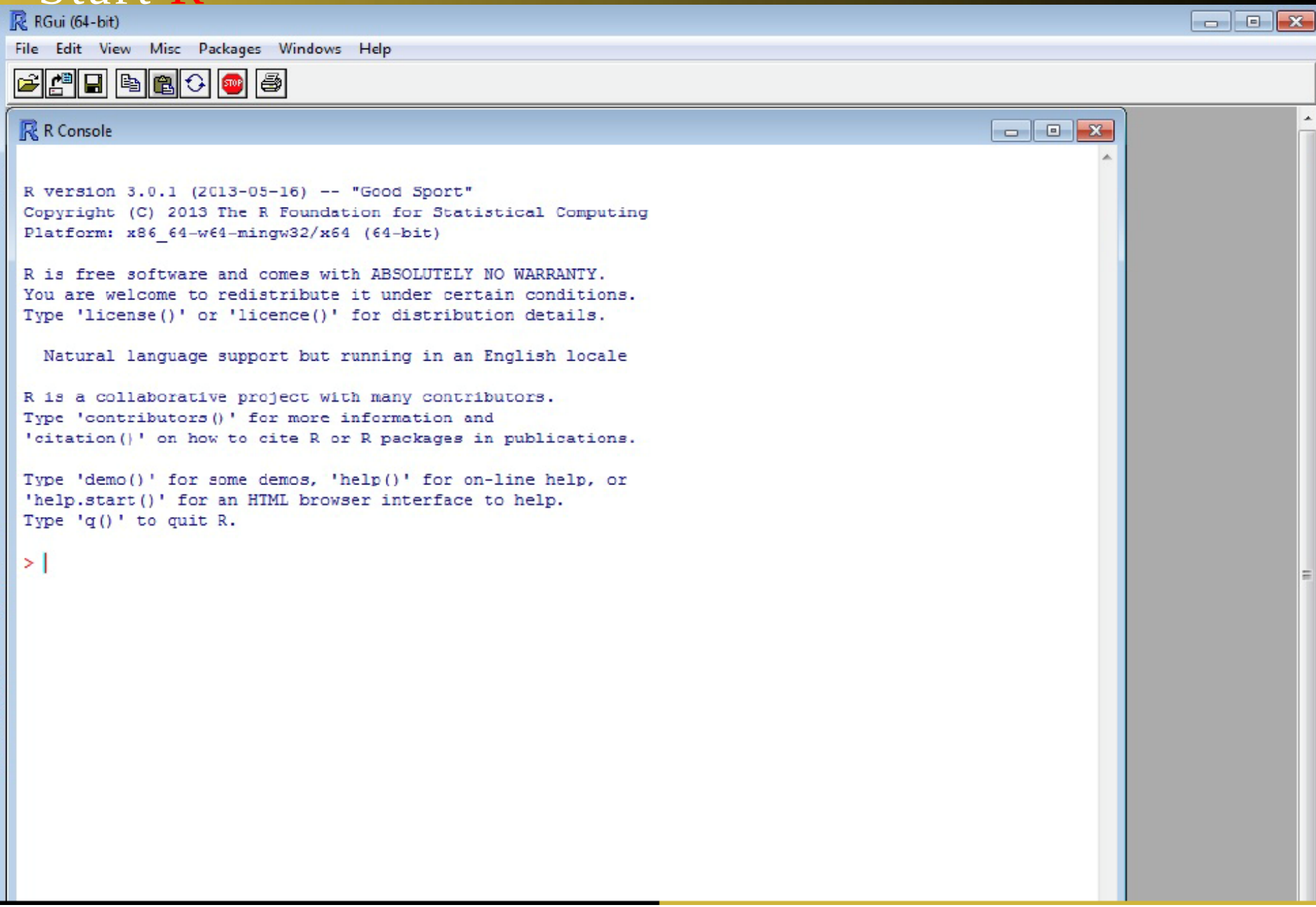
Display the names of the objects

```
> objects()
> ls()
```

Remove variables: `rm()`

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Start R

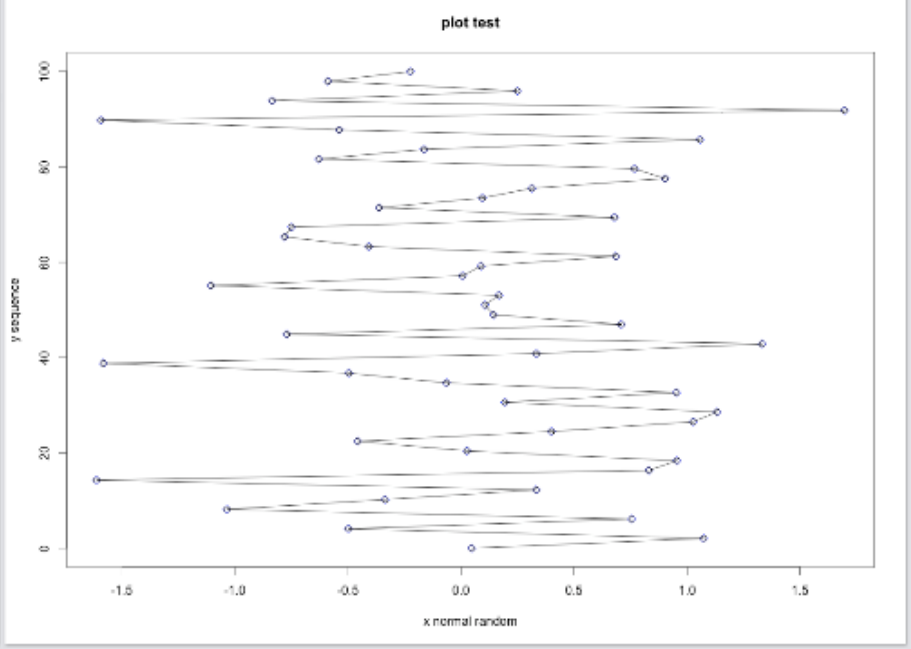

RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute 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.
>

Start R Studio

```
dev.off(3)
# setwd()
getwd()

# boxplot, (try larger size)
x=rpois(lambda=10,50)
boxplot(x)
par(mfrow=c(1,2))
boxplot(x)
boxplot(log(x))
par(mfrow=c(1,1))

# Read Data
# Use read.table() or read.csv() to read data into R
Auto=read.csv("Auto.csv",header=T,na.strings="?")

# read data from the Internet
theURL <- "http://www.jaredlander.com/data/TomatoX20First.csv"
tomato <- read.table(file=theURL,header=TRUE, sep=",")
```

```
> length(x)
[1] 3
> class(x)
[1] "list"
> x[[2]]
[1] "y"
> is.list(tab)
[1] TRUE
> tab[[2]]
 [1] -1.6664328 -0.8663095 -0.8768774 -0.3488891  0.2389225  1.4172195  0.6505842 -0.4588956  0.3766280  0.6528687
> names(tab)
[1] "col1" "col2" "col3"
> # Basic Plot Functions
> x=rnorm(50)
> y=seq(from=0,to=100,length.out=50)
> plot(x,y,xlab='x normal random',ylab='y sequence', main='plot test', pch=5,col=4)
> # plot(x)
> lines(x,y)
> points(x,y)
> # Save a png image to a file
> png('my.first.plot.png',width=480,height=360)
> dev.off(3)
quartz_off_screen
                4
> # setwd()
> getwd()
[1] "/Users/Chester/Desktop/R"
> # boxplot, (try larger size)
> x=rpois(lambda=10,50)
> boxplot(x)
> par(mfrow=c(1,2))
> boxplot(x)
> boxplot(log(x))
> par(mfrow=c(1,1))
> # Read Data
> # Use read.table() or read.csv() to read data into R
> Auto=read.csv("Auto.csv",header=T,na.strings="?")
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'Auto.csv': No such file or directory
>
```

Data	
aa	int [1:2, 1:3] 1 2 3 4 5 6
c.matrix	num [1:5, 1:2] 1 2 3 4 5 ...
m.normal	num [1:3, 1:2] -0.6475 -0.0638 0.2213 0.2804 -0.3351 ...
m2	num [1:3, 1:2] -6.475 -0.638 2.213 52.804 46.648 ...
mat	num [1:10, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
tab	10 obs. of 3 variables
Values	
a	int [1:5] 1 2 3 4 5
b	num [1:5] 1.333 -0.587 -1.429 -1.052 1.666
country	chr [1:7] "US" "UK" "China" "India" "Japan" "Korea" "Canada"
countryf	Factor w/ 7 levels "Canada","China",..: 7 6 2 3 4 5 1
my.math.score	95
my.math.score	95
MyMathScore	95
x	int [1:50] 13 9 11 7 12 8 5 13 8 14 ...
x2	num [1:3] 2 4 6
x3	num [1:3] 3 6 9



JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Rules for Names in R

- Any combination of letters, numbers, underscore, and "."
- May not start with numbers, underscore.
- R is case-sensitive.
- Variable names should be short, but descriptive

Camel caps: MyMathScore=95
Underscore: my_math_score=95
Dot separated: my.math.score=95

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

R Help Functions

- If you know the name of the function or object on which you want help:

```
> help(read.csv)
> help('read.csv')
> ?read.csv
```

- If you do not know the name of the function or object on which you want help:

```
> help.search('input')
> RSiteSearch('input')
> ??input
```

- Do Not forget our friend: Google

  Rseek.org

  stackoverflow.com

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Important data types in R

- Classes
  - Character, Numeric, Integer, Logical
- Objects
  - Vectors, Matrices, Data frames, Lists, Factors, Missing values
- Operations
  - Subsetting, Logical subsetting

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Vectors

- Assignment using function c()

```
> x <- c(1, 2, 3)
> c(1, 2, 3)->x
> y <- c(x, 2, x)
> z<-c(1e3,100)
```

- Vector arithmetic

  Elementary arithmetic operators: +, -, *, /, ^
  Common arithmetic functions: log, exp, sin, cos, tan, $\sqrt{x}$,...
  Other important functions: range(), length(), max(), min(), sum(), prod(), mean(), var(), sort()

- Generating regular sequences

```
seq(-5, 5, by=1) -> x
x <- seq(length=10, from=-5, by=.5) x <-
rep(x, times=5)
x <- rep(x, each=5)
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Vector Operations

- Operations on a single vector are typically done element-by-element
- If the operation involves two vectors:
  - Same length: R simply applies the operation to each pair of elements.
  - Different lengths, but one length a multiple of the other: R reuses the shorter vector as needed
  - Different lengths, but one length not a multiple of the other: R reuses the shorter vector as needed and delivers a warning
- Typical operations include multiplication, addition, subtraction, division, exponentiation, but many operations in R operate on vectors and are then called "vectorized".

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Vector Operations

```
> x=1:6
> y=2
> x*y
[1]  2  4  6  8 10
> y=c(1,10)    12
> x*y
[1]  1 20  3 40  5 60
> x/y
[1] 1.0 0.2 3.0 0.4 5.0 0.6
```

Logical vectors

- Logical vectors are generated by conditions: x<-5>4
  - E.g., x<-5>4
  - Logical operators are <, <=, >, >=, ==, !=
  - Logical expressions: &, |, !
- Missing values
  - missing value: NA, z<-c(1:5,NA)
  - function is.na(x) gives a logical vector
  - miss value: NULL; NULL cannot exist within a vector; if used, it simply disappears
  - z<- c(1,NULL,3)

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Logical Vector Operations

```
> #create a numerical vector
> x=1:6
> #test whether x>3, create a logical vector
> x>3
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
> x<=3
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
> x!=3
[1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
> x==3
[1] FALSE FALSE  TRUE FALSE FALSE FALSE
> # we can also assign the results to a variable
> y=(x>3)
> y
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

Indexing Vectors

- In programming, an index is used to refer to a specific element or set of elements in an vector (or other data structure).
- R uses [ ] to perform indexing

```
> x=seq(0,2,.2)
> #create a new vector from the 5th element of x
> x[4]
[1]
0.6
> y=x[4:]
Error: unexpected ']' in "y=x[4:]"
> y=x[4:length(x)]
> y
[1] 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
```

Indexing can use other vectors for the indexing

```
> x[c(5,6,8)]
[1] 0.8 1.0 1.4
> z=2:5
> x[z]
[1] 0.2 0.4 0.6 0.8
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Indexing Vectors and Logical Vectors

- Combine indexing vectors and logical vectors

```
> x=1:10
> y=(x>5)
> summary(y)
   Mode   FALSE    TRUE    NA's
logical       5       5       0
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    3.25    5.50    5.50    7.75   10.00
> x[y]
[1]  6  7  8  9 10
> x[!y]
[1] 1 2 3 4 5
> x[x>5]
[1]  6  7  8  9 10
```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Factors

- A factor is a special type of vector, normally used to hold a categorical variable in many statistical functions.
- Such vectors have class "factor".
- Factors in R often appear to be character vectors when printed, but you will notice that they do not have double quotes around them.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Factors: Examples

Sample code

```
> country<-c("US","UK","China","India","Japan","Korea","Canada")
> # convert to factor
> countryf<-factor(country)
> country
[1] "US"     "UK"     "China"  "India"  "Japan"  "Korea"
> countryf   "Canada"
[1] US     UK     China  India  Japan  Korea  Canada
Levels: Canada China India Japan Korea UK US
> # convert factor back to character vector
> as.character(countryf)
[1] "US"     "UK"     "China"  "India"  "Japan"  "Korea"  "Canada"
> # convert to numeric
vector
> as.numeric(countryf)
[1] 7 6 2 3 4 5 1
> as.numeric(country)
[1] NA NA NA NA NA NA NA
Warning message:
NAs introduced by coercion
```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

- A matrix is a rectangular array. It can be viewed as a collection of column vectors all of the same length and the same type (i.e., numeric, character or logical).
- A data frame is also a rectangular array. All of the columns must be the same length, but they may be of different types.
- The rows and columns of a matrix or data frame can be given names.
  E.g.,

```
> aa=1:6
> dim(aa)<-c(2,3)
> aa
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Matrix Operations

Create a Matrix

```
> a<-1:5
> b<-rnorm(5)
> # make a matrix by column binding
> c.matrix<-cbind(a,b)
> # names of rows and columns
> rownames(c.matrix)
NULL
> colnames(c.matrix)
[1]
 "a"  "b"
```

Indexing for matrices

```
> c.matrix[4,2]
         b
0.6641357
> c.matrix[1,]
         a          b
 1.0000000  0.2573384
> c.matrix[,2]
[1]  0.2573384 -0.6490101 -0.1191688 0.6641357  1.1009691
> c.matrix[c.matrix>1]
      2     <NA>     <NA>     <NA>     <NA>
2.000000 3.000000  4.000000  5.000000  1.100969
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Matrix Operations

```
> # create a matrix with 2 columns and 3 rows
> # filled with random normal
> m.normal=matrix(rnorm(6),nrow=3)
> m2=m.normal*10
> m2
          [,1]       [,2]
[1,]  1.437715 -14.375862
[2,] -1.177536  -7.970895
[3,] -9.120684  12.540831
> m2[,2]=m2[,2]+50
> summary(m2)
       V1              V2
 Min.   :-9.1207   Min.   :35.62
 1st Qu.:-5.1491   1st Qu.:38.83
 Median :-1.1775   Median :42.03
 Mean   :-2.9535   Mean   :46.73
 3rd Qu.: 0.1301   3rd Qu.:52.28
 Max.   : 1.4377   Max.   :62.54
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Matrices vs. Data Frames

```
> x=1:10
> y=rnorm(10)
> mat<-cbind(x,y)
> class(mat[,1])
[1] "numeric"
> z=paste0('a',1:10)
> tab<-cbind(x,y,z)
> class(tab) [1]
"matrix"
> mode(tab[,1] )
[1] "character"
> head(tab,4)
```

```
     x   y                                z
[1,] "1" "0.772142185804 53"              "a1"
[2,] "2" "-0.21951562675344"             "a2"
[3,] "3" "-0.424810283377287"            "a3"
[4,] "4" "-0.418980099421959"            "a4"
```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Matrices vs. Data Frames

```
> tab<-data.frame(x,y,z)
> class(tab)
[1] "data.frame"
> head(tab)
  x          y   z
1 1  0.7721422  a1
2 2 -0.2195156  a2
3 3 -0.4248103  a3
4 4 -0.4189801  a4
5 5  0.9969869  a5
6 6 -0.2757780  a6
> mode(tab[,1])
[1] "numeric"
> rownames(tab)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
> rownames(tab)<-paste0("row",1:10)
> rownames(tab)
 [1] "row1"  "row2"  "row3"  "row4"  "row5"  "row6"  "row7"
 "row8"
```

Data Frames

- Data frame columns can be referred to by name using the "dollar sign" operator $

```
> tab$x
 [1]  1  2  3  4  5  6  7  8  9 10
> attach(tab)
The following object is masked _by_ .GlobalEnv:
    x, y, z
> x
 [1]  1  2  3  4  5  6  7  8  9 10
```

- Column names can be set, which can be useful for referring to data later

```
> colnames(tab)
[1] "x" "y"
"z"
> colnames(tab)<-c('a','b','c')
> colnames(tab)
[1] "a" "b"
"c"
> colnames(tab)<-paste0('col',1:3)
> colnames(tab)
[1] "col1" "col2" "col3"
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

List

- A list is a collection of objects that may be the same or different types.
- The objects generally have names, and may be indexed either by name (e.g. `my.list$name3`) or component number (e.g. `my.list[[3]]`).
- A data frame is a list of matched column vectors.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

List: Examples

- Create a list

```
> x=list(1,"y",c(2,4,6))
> x

> length(x)
[1] 3
> class(x)
[1]
"list"
> x[[2]]
[1] "y"
> is.list(tab)
[1] TRUE
> tab[[2]]
 [1]    0.7721422 -0.2195156 -0.4248103 -0.4189801
 [9]    1.2993123 -0.8732621  0.9969869 -0.275778
> names(tab)
[1] "a" "b" "c"
```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Basic Plot Functions

- The command `plot(x,y)` will plot vector `x` as the independent variable and vector `y` as the dependent variable.
- Within the command line, you can specify the title of the graph, the na of the x-axis, and the name of the y-axis.
  - `main='title'`
  - `xlab='name of x axis'`
  - `ylab='name of y axis'`

```
> x=rnorm(50)
> y=seq(from=0,to=100,length.out=50)
> plot(x,y,xlab='x normal random',ylab='y sequence',
+ main='plot test', pch=5,col=4)
```

- `plot(x)`: x is a vector, produce a scatter plot
- The command `lines(x,y)` adds a line segment to the plot. The
- command `points(x,y)` adds points to the plot.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

- Graphics Devices and Saving Plots
- To make a plot directly to a file use: `png()`, `postscript()`, etc.
- R can have multiple graphics "devices" open.
  - To see a list of active devices: `dev.list()`
  - To close the most recent device: `dev.off()`
  - To close device 3: `dev.off(3)`
  - To use device 5: `dev.set(5)`
- Save a png image to a file

```
> png("my.first.plot.png",width=480,height=360)
> dev.off(3)

> setwd()
> getwd()
```

BoxPlot, Histogram

- boxplot, (try larger size)

```
> x=rpois(lambda=10,50)
> boxplot(x)
> par(mfrow=c(1,2))
> boxplot(x)
> boxplot(log(x))
> par(mfrow=c(1,1))
```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Read Data

- Function `read.table()`
  - Data is stored in a format referred to as a `Data Frame`
  - Use function `fix()` to view the data in a spreadsheet like window
- Missing values
  - Use `read.table()` or `read.csv()` to read data into R
  - `header=T` tells R that the first line contains variable names
  - `na.strings` tells R that it sees a particular character or set of characters, it should be treated as a missing element.
  - Example:

    ```
    Auto=read.csv("Auto.csv",header=T,na.strings="?")
    ```
  - read data from the Internet

    ```
    theURL <-
    "http://www.jaredlander.com/data/Tomato%20First.csv" tomato <-
    read.table(file=theURL,header=TRUE, sep=",") head(tomato)
    ```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Probability Distributions

- R provides a set of functions to evaluate
  - The Cumulative distribution function $\Pr(X \leq x)$, e.g., `pnorm(2,mean=5,sd=10)`
  - The probability density function and the quantile function, e.g., `> dnorm(2,mean=5,sd=10)`
    `> qnorm(.38,mean=5,sd=10)`
  - Simulate from the distribution
    `> z=rnorm(mean=5,sd=100,n=10)`

- Prefix names
  - 'd' for the density, computes $f(x)$
  - 'p' for the CDF, computes $F(x) = \Pr(X \leq x) = \int_{-\infty}^{x} f(t)dt$
  - 'q' for the quantile function, computes x such that $\Pr(X \leq x) = p$
  - 'r' for the random variables, returns a random variable

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Distribution, R Name, Additional Arguments

Distribution	R name	Additional arguments
uniform	unif	min, max
binomial	binom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
Student's	t	df, ncp
F	f	df1, df2, ncp
chi-squared	chisq	df, ncp
...		

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Example: Uniform Distribution

- Uniform Distribution
  - Density function:

$$f(x|a,b) = \frac{1}{b-a}, \quad \forall a \leq x \leq b.$$

  - Cumulative distribution function:

$$F(x) = \frac{x-a}{b-a}, \quad \forall a \leq x \leq b.$$

- R code

```
> dunif(x=8,min=5,max=15)
> punif(10,min=5,max=15)
> qunif(.8,min=5,max=15)
> runif(10,min=5,max=15)
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Example: Normal Distribution

- Generate Random Variable:
  - R code: `rnorm(n, mean = 0, sd = 1)`
- Role of function `set.seed()`: Setting a seed ensures reproducible results from random processes in R
  - Compare results

```
> set.seed(5)
> rnorm(3,mean=10,sd=20)
[1]  -6.81711  37.68719 -15.10984

> rnorm(3,mean=10,sd=20)
[1]    11.40286    44.22882   -2.05816

> set.seed(5)
> rnorm(3,mean=10,sd=20)
[1]  -6.81711  37.68719 -15.10984
```

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Sample Function

- Use sample function to pick five numbers at random from the set 1:40

```
> sample(1:40,5)
```

- By default, sample picks random elements without replacement, i.e., no number appears twice.

```
> sample(c("H","T"),10,replace=T)
[1] "T" "H" "H" "H" "T" "H" "T" "T" "T" "T"
```

- Sampling with different probability

```
> sample(c("success","failure"),10,
replace=T,prob=c(0.8,0.2))
 [1] "success" "success" "failure"
"success" "success" "success" "success"
"failure" "success" "success"
```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

- plot density, noting that "l" is lower-case letter "l", not the digit "1"

```
> x<-seq(-4,4,0.01)
> plot(x,dnorm(x),type="l")

> x<-0:50
> plot(x,dbinom(x,size=50,prob=.33),type="h")
```

- plot cumulative distribution functions

```
> x<-seq(-4,4,0.01)
> plot(x,pnorm(x),type="l")

> x<-0:50
> plot(x,pbinom(x,size=50,prob=.33),type="h")
```

Define a Function

- Use function `function()` to define a function
- For example, define a function $f(x) = 3x^{-4}, \quad x \geq 1$.

```
> f<-function(x){3*x^(-4)}
> f(2)
[1] 0.1875
```

- Verify whether a function is a well-defined density function, check

  its integral from a to b $\int_a^b f(x)dx$ using function `integrate()`

```
> integrate(f,1,Inf)
1 with absolute error < 1.1e-14
```

- Hint: not sure about the arguments of a function, use `args()` in R: e.g., `> args(integrate)`

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Simulation: Generate Random Variables Following Any Distribution

- Use *Inverse Transformation Method*
  - Simply set the Cumulative Distribution Function (CDF) equal to the uniform variable, i.e., $F(X) = U \sim Unif(0,1)$.
  - Express the variable using the uniform variable, i.e., $X = F^{-1}(U)$.

Theorem

$F^{-1}(U)$ follows the CDF F(x), i.e., $\Pr(F^{-1}(U) \leq x) = F(x)$.

- Example: the CDF $F(x) = Int[(1,x)\ 3x^{-4}dx = 1 - x^{-3}$
  - Set $F(x) = U \sim Unif(0,1)$, i.e., $1 - x^{-3} = U$.
  - Then, $x = (1-U)^{-1/3}$
- Sample R code

```
> set.seed(13)
> U=runif(1000)
> X=(1-U)^(-1/3)
```

- X follows CDF F(x)

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Inverse Transformation Method, Cont.

- Online integrator:
  `http://integrals.wolfram.com/index.jsp`
- Indefinite integral: $\int f(x)dx \sim F(x)$
- Definite integral: $\int_a^b f(x)dx = F(b) - F(a).$

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Questions,
Comments?

See you next time.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL