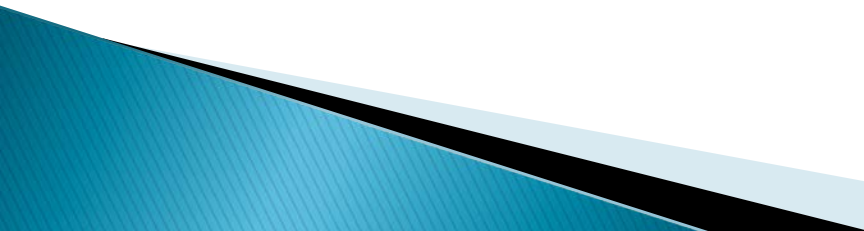


# Introduction to R and RStudio


Part 1a: Introduction to R

Rob Cribbie  
Department of Psychology  
York University

# What is R?

- R is a computer language, with exceptional capabilities for statistical analysis and graphics
  - R is the open source version of S
    - S was developed at Bell Laboratories in the 1970s, and is commercially sold as S-Plus
  - R was initially written in the 1990s by Ross Ihaka and Robert Gentleman, and is now supported by an international R-core team (in addition to thousands of people who contribute to R)
- 

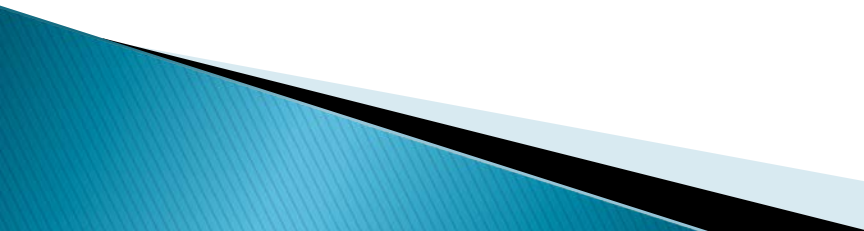
# What can R do?

- Manage data
  - Perform mathematical operations on data points, vectors and matrices
  - Perform basic and advanced statistical analyses through either the preloaded functions or user contributed functions
  - Generate a wide range of useful and interesting graphics
  - Can be used as a simple and effective programming language which includes conditionals, loops, if-else, etc. (plus any user contributed or created functions)
- 

# R Advantages

- Advantages of R over other software packages:
  - Free!!
  - Completely customizable
  - Available for all popular operating systems
  - Active user community
  - Forces you to think about your analyses
  - Always cutting edge on statistics, graphics, etc.
    - E.g., because of the availability of user contributed ‘functions’ the gap between the introduction of new statistical procedures in the literature and their availability in R is often much shorter than with other packages

# R Disadvantages

- NOT USER FRIENDLY! ... at least initially
    - Steep Learning Curve
  - Help files are sometimes difficult to use
  - Error messages are sometimes (often?) uninformative
  
  - But don't worry ... we are going to get by all these issues!
- 

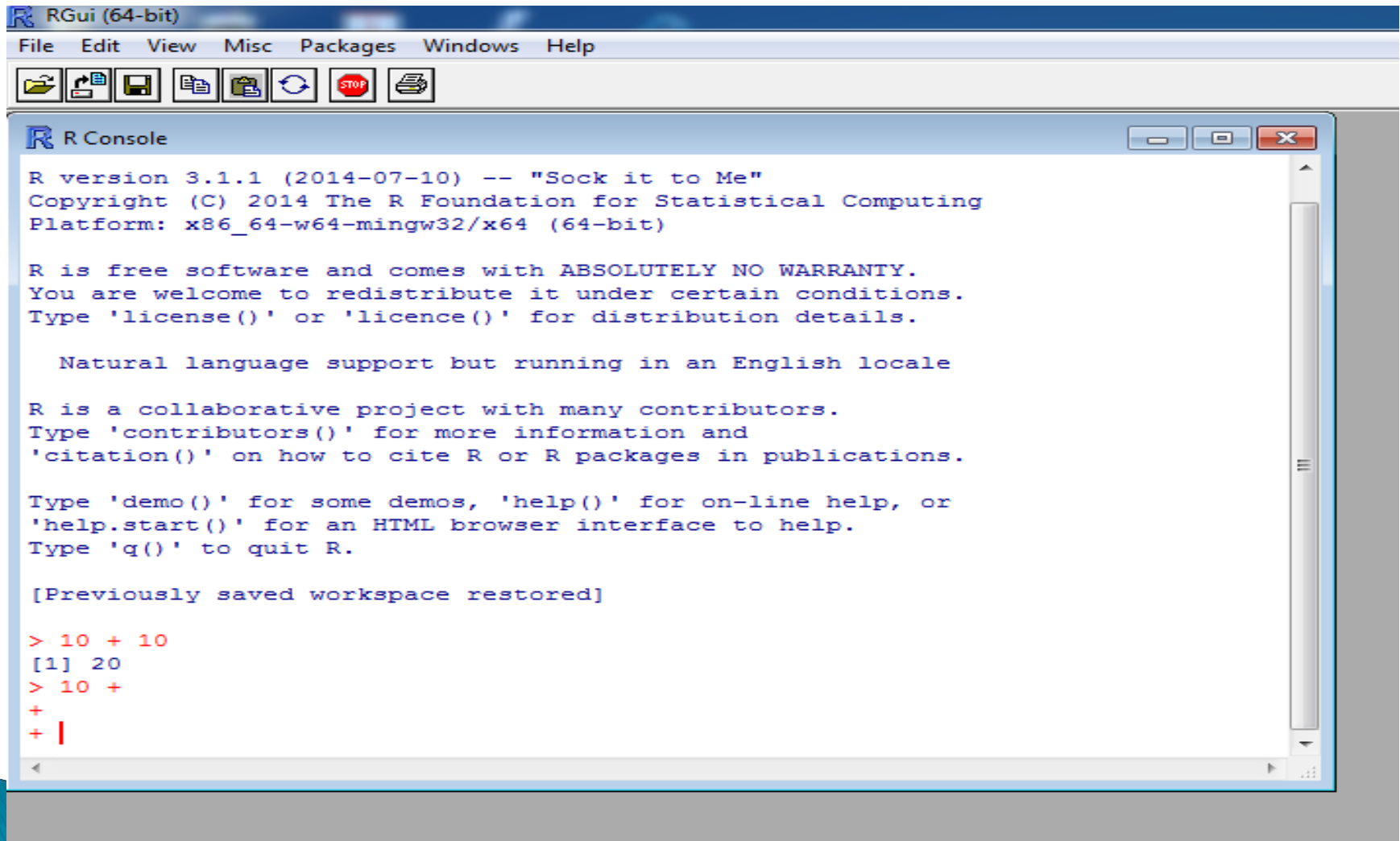
# How to Get R

- The R webpage is: <http://www.r-project.org/>
  - This webpage contains documentation, manuals, faqs, links to books, a newsgroup, an R search engine, etc.
- To install R on your computer:
  - Click on 'CRAN' under the 'Download' heading, and then select a mirror site
  - Click on the appropriate operating system (e.g., Windows, MAC, Linux)
  - Click on 'base', to download the base package
  - Click on 'Download R for Windows' (assuming you are using Windows)
    - The exact name of the file will change based on the version number

# Running R

- Like any other Windows program, R can be run by clicking on the icon or running R from the ‘Programs’ menu
- When you start R, the first thing you will see is the prompt (`>`), which is R's way of saying “Go Ahead ... Do something”
  - If you see a “+” in place of the prompt, that means that your last command was not complete
- Don't forget that R is case sensitive!

# What does R look like?





# Getting Help with R

- A lot of valuable information including links to a mailing list, an 'R search' website, and other helpful information can be found through CRAN
  - <http://cran.r-project.org/>
- I have also found that 'googling' the topic of interest, followed by r, can be effective
  - E.g.: google 'anova r' or 'latent class analysis r'
- If you know the R function you are interested in, you can get help on it in R by typing
  - >?function
  - Example: >?t.test

# Help file for 't.test'

```
t.test {stats}
```

## Student's t-Test

### Description

Performs one and two sample t-tests on vectors of data.

### Usage

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

```
## Default S3 method:
```

```
t.test(x, y = NULL,  
       alternative = c("two.sided", "less", "greater"),  
       mu = 0, paired = FALSE, var.equal = FALSE,  
       conf.level = 0.95, ...)
```

```
## S3 method for class 'formula'
```

```
t.test(formula, data, subset, na.action, ...)
```

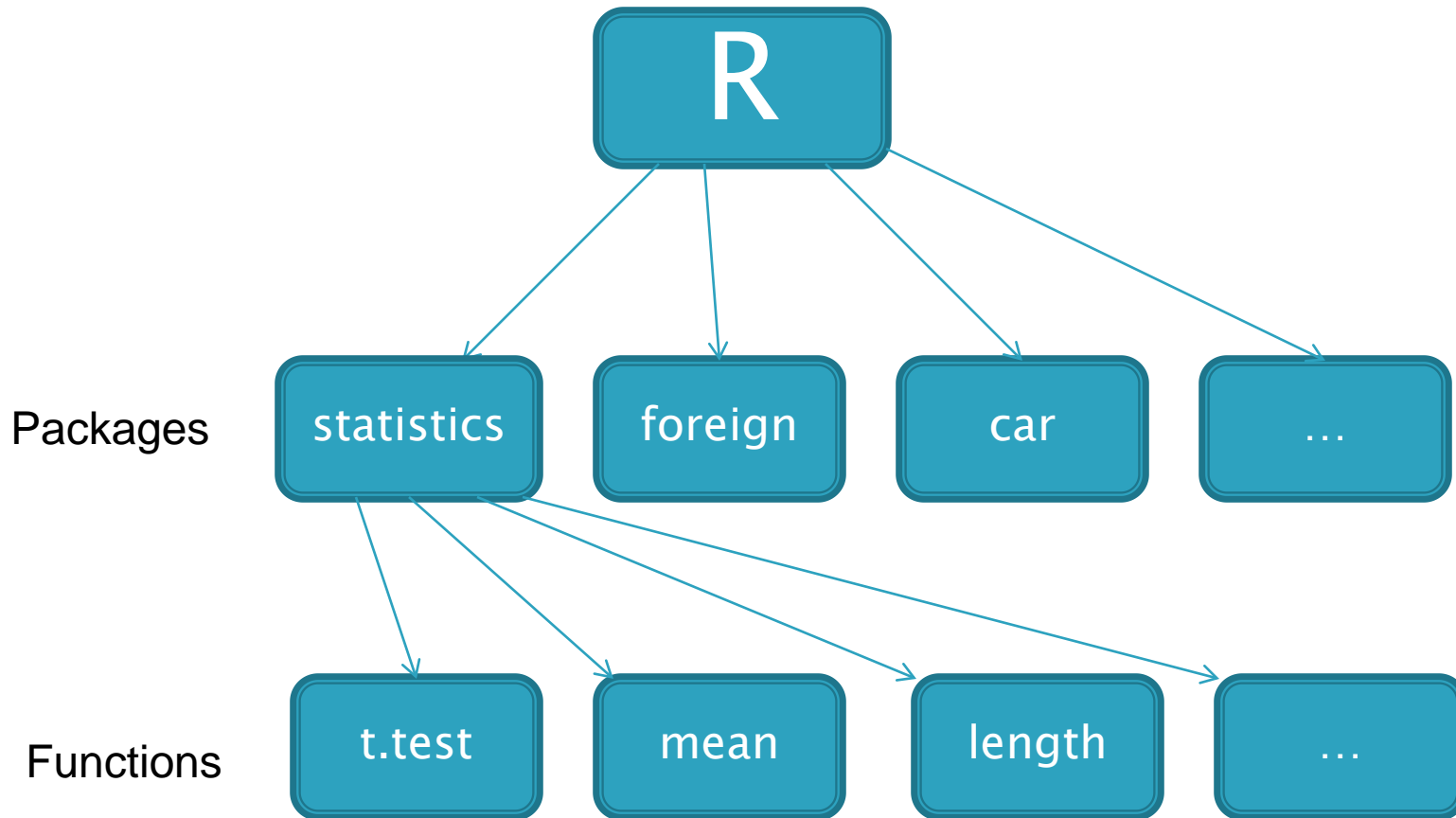
### Arguments

<code>x</code>	a (non-empty) numeric vector of data values.
<code>y</code>	an optional (non-empty) numeric vector of data values.
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "less" or "greater".
<code>mu</code>	a number indicating the true value of the mean (or difference in means).
<code>paired</code>	a logical indicating whether you want a paired t-test.

# Getting Help with R

- There are also numerous R Reference Cards that are available online that can make remembering many popular R commands simpler
- E.g.,
  - <https://cran.r-project.org/doc/contrib/Short-refcard.pdf>
  - <https://cran.r-project.org/doc/contrib/Baggott-refcard-v2.pdf>
  - <https://cran.r-project.org/doc/contrib/refcard.pdf>

# Understanding How R Works



Although packages contain functions, they can also contain other types of files (e.g., datasets)

# Packages and Functions

- One of the tricky parts to learning R is understanding what functions are automatically available, and how to access functions that are not automatically available
  - Functions are R commands, that often have specific defaults and options
    - ‘t.test’ is an R function that computes a t-test
  - However, ‘t.test’ is also part of a package called ‘statistics’
    - The ‘statistics’ package is automatically installed when you install R and is automatically loaded (‘libaried’) each time you start R

# Packages and Functions, cont'd

- Because the 'statistics' package is automatically loaded each time you start R, all of the functions available in the 'statistics' package are available at any time
- Other packages are installed when you install R, but are not loaded each time you start R
  - The package 'foreign', which has functions for reading external data sets (e.g., SPSS, Excel) is installed when you install R, but is not loaded each time you start R
- Other packages need to be both installed and loaded before they are accessible (e.g., 'car', 'psych', which we will make use of)

# More on Functions

- All functions contain 'arguments' for running the function
- E.g., the 'mean' function:

Default

`mean(x, trim = 0, na.rm = FALSE, ...)`

- Some objects are necessary because they have no default (e.g., 'x'), where some are unnecessary because they have defaults (e.g., 'trim')

# More on Functions

- Because R is developed by a community of developers and users, one of the advantages is that there are functions available to do almost anything you need to do, and further, there are also often different functions for doing similar things that each have their own advantages and disadvantages
- E.g., `hist` (*graphics* package) vs `histogram` (*lattice* package)



# Loading Installed Packages

- What if we try the following command:
  - `> histogram(x)`
  - Error: could not find function "histogram"
- Why do we get an error?
  - The function "histogram" is part of the package "lattice", which is part of the default installation but is not loaded each time you start R
- To load a new package we can use the drop down menus or just type
  - `> library(lattice)`
  - To get help with the package type either `help(lattice)` or `?lattice`

# Installing Packages

- Most packages are not installed as part of the default installation of R
  - The reason for this is that there are more than six thousand packages (and this number is growing fast) and there are probably only a small subset of those that will ever be of interest to you
- To install a new package in R, click on “Packages” and then “Install package(s)...”
- If you know the name of the package you want to install you can just use:
  - `install.packages("package_name")`

# R as a Fancy Calculator

```
> 2+2*3/6
```

```
[1] 3
```

```
> sqrt(2)
```

```
[1] 1.414214
```

```
> factorial(5) # 5!
```

```
[1] 120
```

```
> log(10)
```

```
[1] 2.302585
```

```
> round(3.8)
```

```
[1] 4
```

```
> round(x=3.234, digits=1)
```

```
[1] 3.2
```

```
> 3^2
```


```
[1] 9
```

```
> 3^1.5
```

```
[1] 1.732051
```

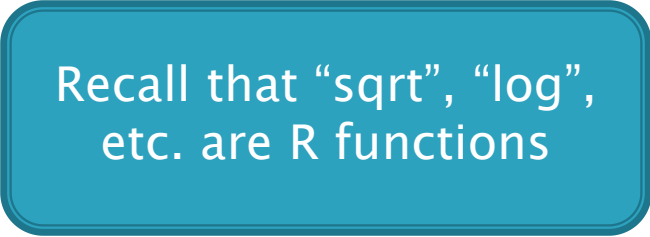
```
exp(1)
```

```
[1] 2.718282
```



“to the  
exponent”

Two arrows originate from the box: one points to the '2' in the expression  $3^2$  and the other points to the '1.5' in the expression  $3^{1.5}$ .



Recall that “sqrt”, “log”,  
etc. are R functions

# R as a Fancy Calculator, cont'd

```
> pi
```

```
[1] 3.141593
```

```
> 1:10
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
> rep(x=0,times=10) # repeat '0' ten times
```

```
[1] 0 0 0 0 0 0 0 0 0 0
```

```
> seq(from=0,to=10,by=2) # 0 to 10, by 2
```

```
[1] 0 2 4 6 8 10
```

```
> 1+3:10 # same as 1+(3:10), note order of operations
```

```
[1] 4 5 6 7 8 9 10 11
```

Google 'R reference card' for lots of different collections of popular R functions

# Objects

- If we type the following, the answer is not stored in any way:

```
>2+2
```

```
[1] 4
```

- On the other hand, if we type the following, the answer is stored in the object “a”

```
a<-2+2
```

- We can see the value of ‘a’ by typing either:

```
a
```

```
print(a)
```

# A Reminder about Case Sensitivity

- As mentioned earlier, all input into R is case sensitive
- Here is an example:

```
> a<-2
```

```
> a
```

```
[1] 2
```

```
> A
```

```
Error: object 'A' not found
```

```
> Rep(0,time=3)
```

```
Error: could not find function "Rep" (the real command  
is 'rep')
```

# Vectors

➤ Vectors are similar to variables

```
> x<-c(3,4,2,3,5,6)
```

- The ' $<-$ ' combination is used as a “gets” command
- 'c' stands for 'combine' or 'concatenate'

```
> y<-c(5,7,3,3,4,7)
```

```
> mean(x)
```

```
[1] 3.833333
```

```
> sd(y)
```

```
[1] 1.834848
```

# Assigning vectors/objects

- Technically, to create objects, you could use:

```
> x=10 (instead of x <- 10)
```

```
> x
```

```
[1] 10
```

```
> x=c(10,12,15)
```

```
> x
```

```
[1] 10 12 15
```

- However, most R users prefer “<-” to “=” to avoid confusion



# Assigning vectors/objects

- For example, imagine you wanted to compute the median of a variable `x`, where 'x' is created from within the median function

```
median(x = 1:10)
```

```
x
```

```
## Error: object 'x' not found
```

```
median(x <- 1:10)
```

```
x
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

# Types of Vectors

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
```

```
b <- c("one","two","three") # character vector
```

```
c <- c("hat","coat","boot") # character vector
```

- For obvious reasons, mathematical and statistical operations can be performed on numerical vectors, but not on character vectors

# Types of Vectors, cont'd

- All elements of a vector must be of the same type

- For example:

```
> x <- c(3, 5, "A")
```

```
> x
```

```
[1] "3" "5" "A"
```

- In this case R automatically converted all of the elements of `x` to characters (since it would be hard to convert 'A' to numeric)

# Types of Vectors, cont'd

- We can gain information regarding our vectors through the functions “str” (structure) and “class”

```
> aa<-c(4,6,5,3,6)
```

```
> class(aa)
```

```
[1] "numeric"
```

```
> str(aa)
```

```
num [1:5] 4 6 5 3 6
```

```
> aa<-c("a", "b", "c")
```

```
> class(aa)
```

```
[1] "character"
```

```
> str(aa)
```

```
chr [1:3] "a" "b" "c"
```

# Vectors – Miscellaneous

- Invoking a function on a vector

  - > shapiro.test(x)

    - Shapiro–Wilk normality test

    - data: x

    - W = 0.958, p-value = 0.8043

- Random number generation

  - x2<-rnorm(10,mean=5, sd=2)

  - > x2

    - [1] 5.778666 2.143754 7.489185 5.508185 2.708508  
9.040165 3.875893 3.246373 6.384725 3.515293

  - > sum(x2)

    - [1] 49.69075

# Operations on Vectors

- There are ways of expressing:
  - Greater than ( $>$ )
  - Less than ( $<$ )
  - Greater than or equal to ( $>=$ )
  - Less than or equal to ( $<=$ )
  - Not equal to ( $!=$ )

```
> x <- c(3,6,5,4)
```

```
> x > 4
```

```
[1] FALSE TRUE TRUE FALSE
```

```
> x >= 4
```

```
[1] FALSE TRUE TRUE TRUE
```

# Indexing

```
> x2[1] # x2 is the random variable created earlier  
[1] 5.778666
```

```
> sum(x2[1:3])  
[1] 15.41161
```

```
> sort(x2)  
[1] 2.143754 2.708508 3.246373 3.515293 3.875893  
5.508185 5.778666 6.384725 7.489185 9.040165
```

```
> rev(sort(x2))[1:2]  
[1] 9.040165 7.489185
```



Getting fancy!

# Indexing with & and |

- We can use & (and) and | (or, note that this is a vertical bar not upper case i)

```
> x<-c(3,6,5,4)
```

```
> x[x>3]
```

```
[1] 6 5 4
```

```
> x[x>3 & x<5]
```

```
[1] 4
```

```
> x[x<4 | x>5]
```

```
[1] 3 6
```



# More on Vectors

```
> vec<-3:12
```

```
> vec
```

```
[1] 3 4 5 6 7 8 9 10 11 12
```

```
> vec[5]
```

```
[1] 7
```

```
> vec[-5]
```

```
[1] 3 4 5 6 8 9 10 11 12
```



vec without the  
5<sup>th</sup> case

# More on Vectors

```
> vec<7
```

```
[1] TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
> any(vec>12)
```

```
[1] FALSE
```

```
> which (vec>11)
```

```
[1] 12
```

```
> which (vec>3 & vec<9)
```

```
[1] 2 3 4 5 6
```



Locations in the  
vector

# Ranks, Bootstrapping/Resampling

```
> x3<-rnorm(10)
```

```
> x3
```

```
[1] -0.7791141  0.2138522 -0.1641621  
[4] -0.4108616 -0.3146857  1.6288796  
[7]  0.3486638 -0.4073264  0.3174018  
[10]  0.3474381
```

```
> rank(x3)
```

```
[1]  1  6  5  2  4 10  9  3  7  8
```

```
> sample(x=x3,size=3)
```

```
[1] -0.1641621  0.2138522 -0.4073264
```

```
> sample(x=x3,replace=TRUE)
```

```
[1]  0.3474381 -0.4073264 -0.1641621  
[4] -0.7791141 -0.7791141 -0.1641621  
[7]  0.3486638  0.3474381  0.3174018  
[10] -0.4073264
```

# More Operations on Vectors

- In some situations we need to perform mathematical operations on vectors
  - For example, we might need to add an amount to all elements of a vector, etc.

```
> aa<-c(4,6,5,3,6)
```

```
> aa+2
```

```
[1] 6 8 7 5 8
```

```
> aa-4
```

```
[1] 0 2 1 -1 2
```

```
> aa*2
```

```
[1] 8 12 10 6 12
```

# Missing Data

- The symbol 'NA' (not available) is used as a placeholder for missing values

```
x<-c(3,5,3,7,NA,1,4,6)
```

```
> x
```

```
[1] 3 5 3 7 NA 1 4 6
```

```
> mean(x)
```

```
[1] NA # mean can't be computed with NAs
```

```
> mean(x, na.rm=TRUE)
```

```
[1] 4.142857
```

Different functions have different defaults for handling missing data

# Missing Data

```
> var(x[-5])  
[1] 4.142857
```

```
y<-x[!is.na(x)]  
> y  
[1] 3 5 3 7 1 4 6
```

```
> y<-x[complete.cases(x)]  
> y  
[1] 3 5 3 7 1 4 6
```

# Missing Data

- NaN (not a number)

```
> x4<-sqrt(-1)
```

```
Warning message: In sqrt(-1) : NaNs produced
```

```
> x4
```

```
[1] NaN
```

- NaN is a mathematical property (e.g., division by 0) where NA is for missing cases

# Demonstration

- Calculator
- Vectors