INTRODUCTION TO R

Manpreet S. Katari

Video 1

LEARNING OBJECTIVES

After completing this module, you will be able to:

- Install R and RStudio on Windows and Mac OS
- Become familiar with the R environment
- Define and recognize basic data structures in R such as vector, factor, matrix, data.frame and list
- Execute commands using R functions
- Import and export data in R

What is R ?

- R is a free software environment for statistical computing and graphics.
- It is similar to the S language and environment which was developed at Bell Laboratories
- One of R's strengths is the ease with which well-designed publication-quality plots can be produced,
- Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

http://www.r-project.org/about.html

The R environment

- An effective data handling and storage facility
- a suite of operators for calculations on arrays, in particular matrices
- a large, coherent, integrated collection of intermediate tools for data analysis
- graphical facilities for data analysis and display either directly at the computer or on hard copy
- programming language (called “S”): includes conditionals, loops, user defined recursive functions and input and output facilities.

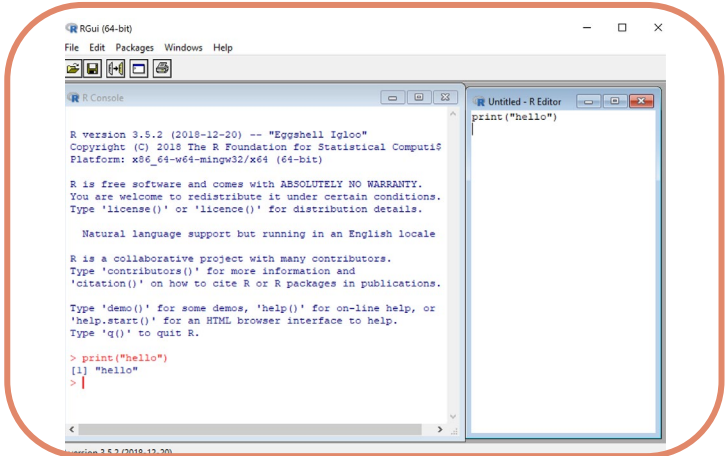
http://www.r -project.org/about.html

Install R

- www.r-project.org
  - Main website for R project
- http://lib.stat.cmu.edu/R/CRAN/
  - One of the mirrors where you can download R

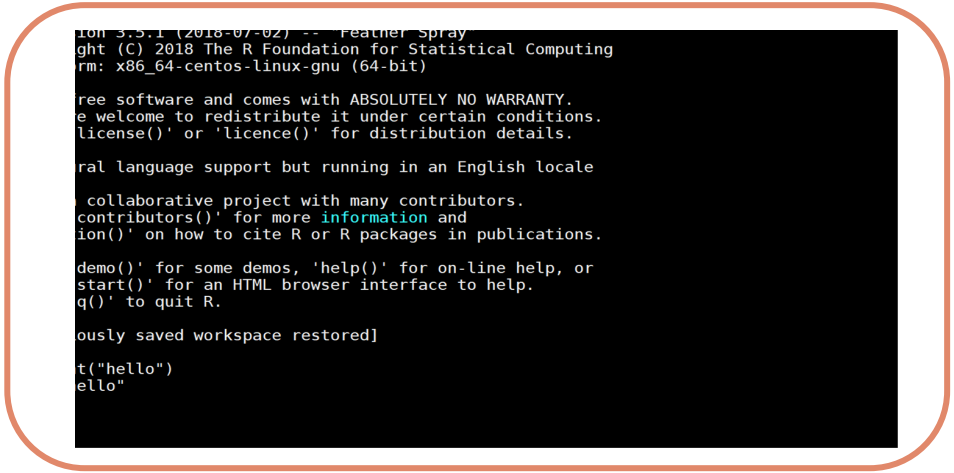
Using R

- The Window System (Rgui)
  - The most convenient way to use R is at a graphics workstation running a windowing system.
  - R uses the X window system bundled with R software.



Using R

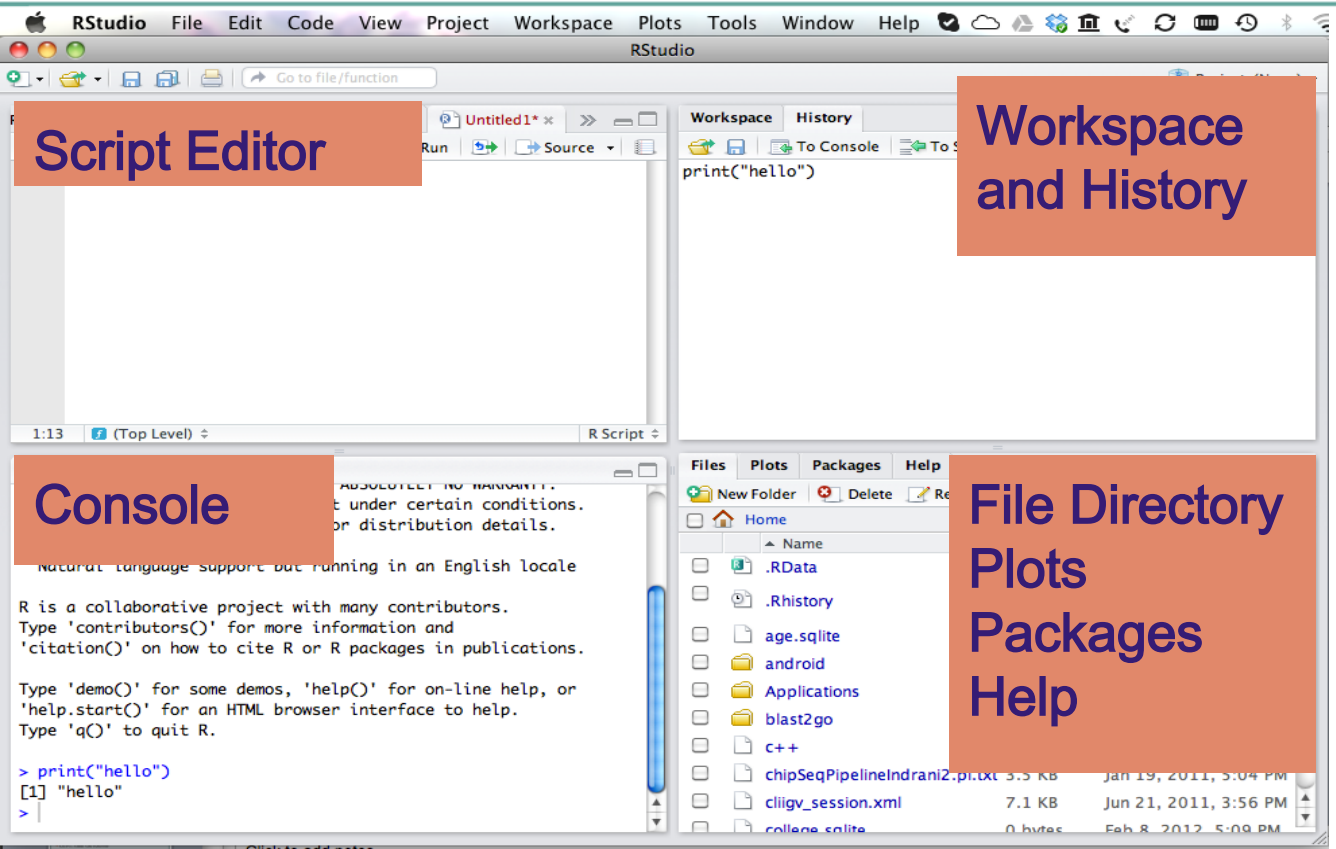
- Using R on command line
  - Requires knowledge of command-line operations.
  - Will use less computer resources because window system not required.



Using R

- RStudio (www.rstudio.org)
  - Much more powerful and user-friendly interface compared to Rgui.
  - Requires that R be already installed on the system.

RStudio


Script Editor
Workspace and History
Console
File Directory
Plots
Packages
Help
RStudio
print("hello")
> print("hello")
[1] "hello"

Video 2

R editor and Scripts

- A history of your commands is saved and it can be accessed by using the up and down keys.
- Your history is saved as .Rhistory in your working directory.
- It is a good idea to save your successful commands in a separate file because your history will also contain your mistakes.

R editor and Scripts

- Open editor by selecting “New Script” from the File Menu.
- Similar to Notepad, it will allow you to type and save code as text.
- MS Word is not a good choice for this because when you paste it can insert funny characters.
- You can execute an entire R script by using the “Source R code” from File menu or the source() function.

Familiar commands that work in R script editor

- Ctrl-c : copy
- Ctrl-v : paste
- Ctrl-L : clear the console
- Esc : stop

R Console Prompt

- R is used by typing in a list of commands
- Commands are entered after the prompt “>”
- After you type a command and its arguments, simply press the Return or Enter Key
    - Separate commands using “ ;” or “newline” (enter)

```
print(“hello”)
[1] “hello”
```

Basic Syntax

- In order to see the contents of an object you can simply type the name of the object.
- If you type a word that is not an object you will get an error

```
hello
Error: object "hello" not found
```

- Names of objects are case sensitive so “Print” is not the same as “print”

More syntax

- You can add comment to your code without it being computed by preceding it with #.

```
x<-1 #everything on the right side is
a comment.
```

- In a case when not all the code can fit in one line, or you want to make the command more readable, you can press “Return” and R will simply start the prompt with +

```
x<-1
print(
x)
[1] 1
```

Video 4

R session

- Default Workspace
  - Workspace contains the different R objects and function only (not the commands)
  - The name of the default workspace is saved as .RData
  - To load .RData, set the directory where .RData is located as current directory and then select to “load Default workspace”
- Working Directory
  - It is a good idea to have separate workspace and history for different projects saved in different directories (folders)

Working Directory

- Getting you working directory

```
getwd()
```

- Changing your working directory

```
setwd("R_project_folder")
```

- Save a workspace

```
save.image("Name_of_workspace.RData")
```

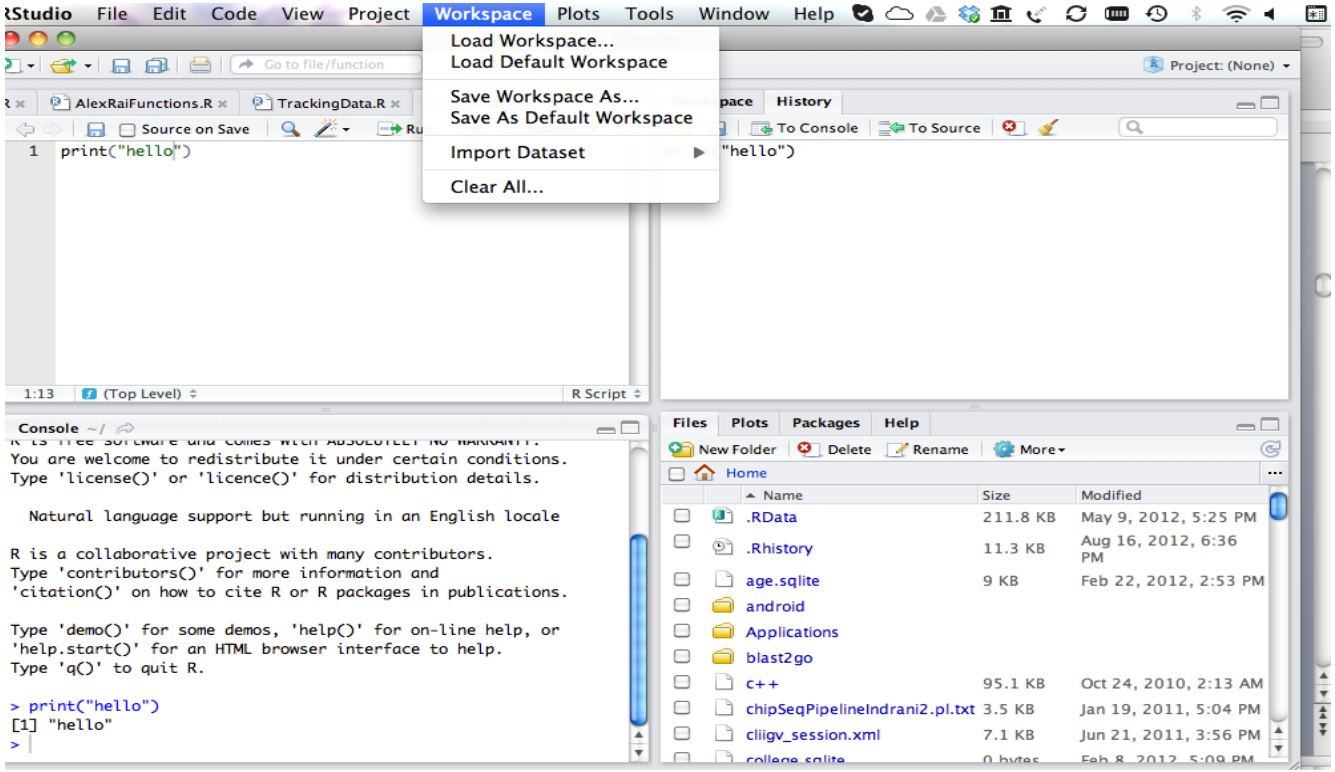
- Load a workspace

```
load("Name_of_workspace.RData")
```

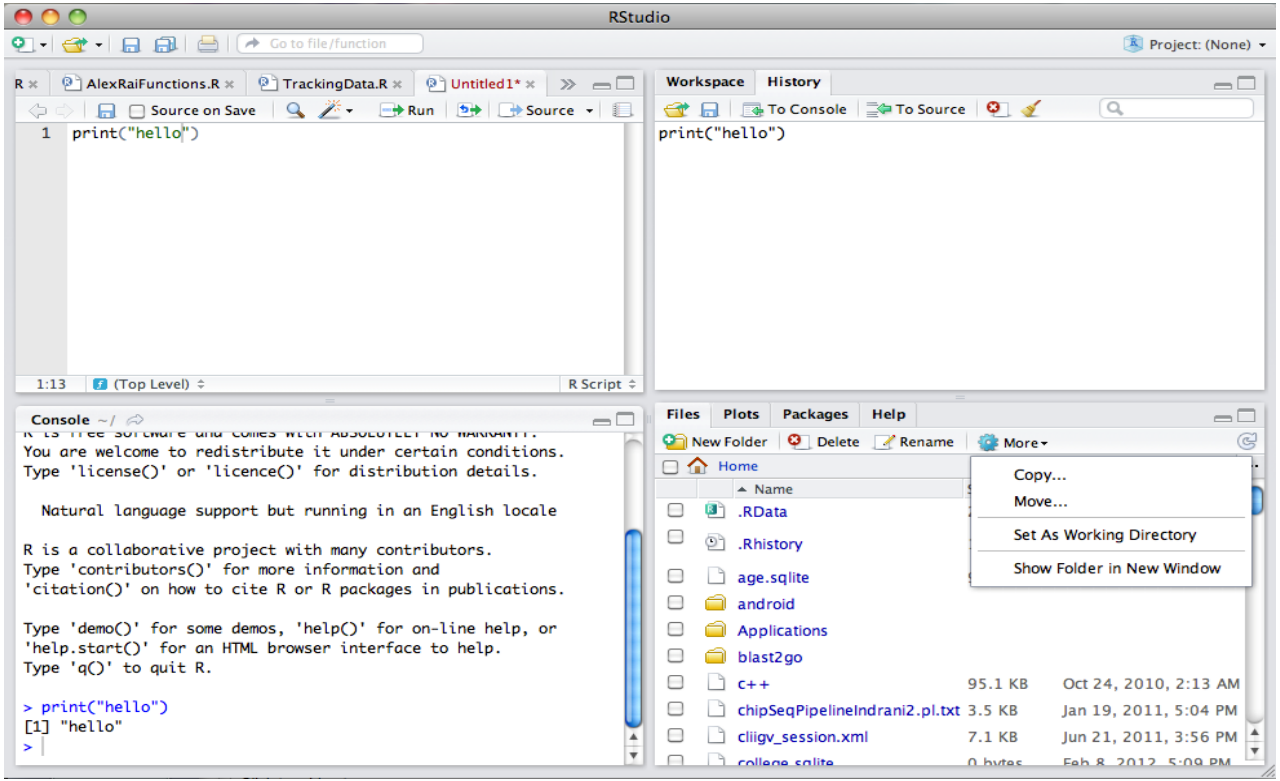
- List the contents of your working directory

```
dir()
```

Workspaces and Working directories in R Studio

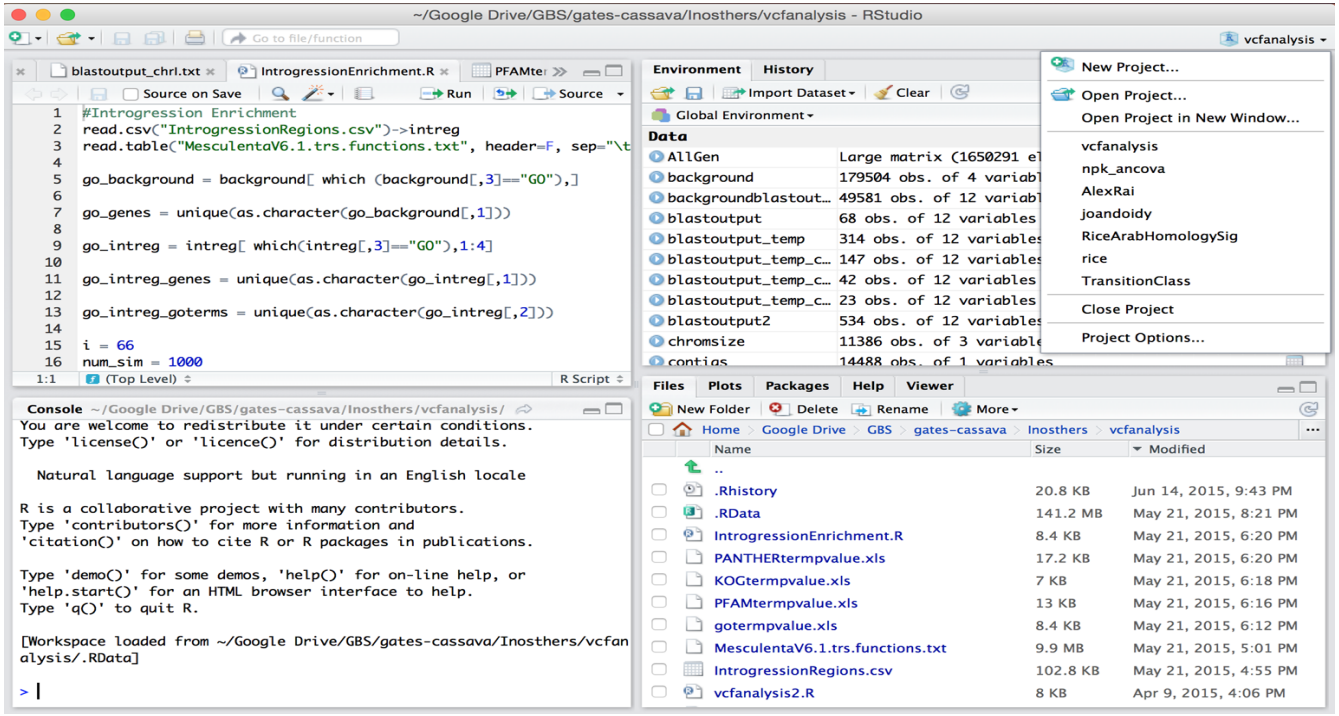


Workspaces and Working directories in R Studio



Rstudio Projects

- manage workspaces and working directories for you.



Packages

- Packages such as Bioconductor are available on CRAN. They contain specialized functions and data that can be used for your analysis.
- To view the names of the packages installed

```
library()
```

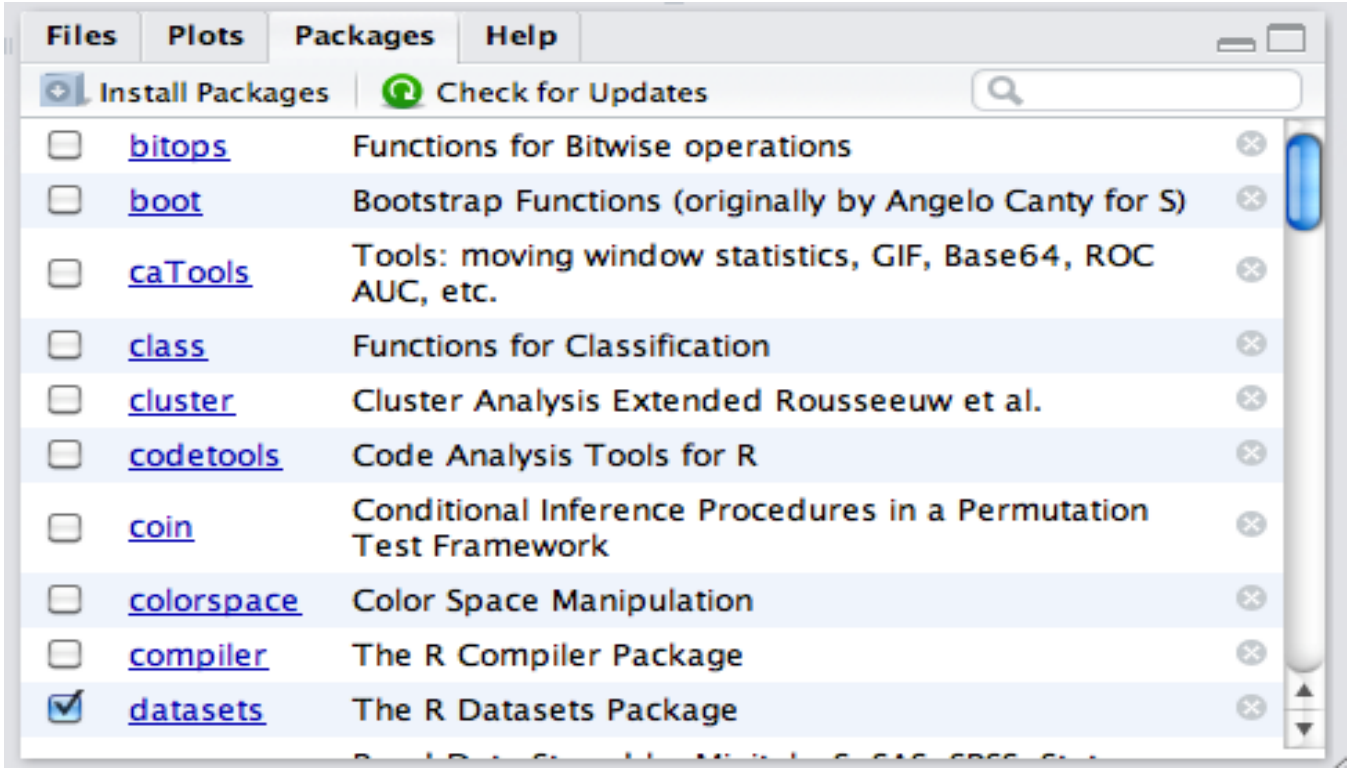
- To load a library.

```
library(cluster)
```

- To install packages

```
>install.packages()
```

Rstudio interface to packages



Getting Help with functions and features

- Different commands to get help for a command

```
help(sum)
?sum
```

- In case you don't know the full name of the command

```
apropos("mean")
```

- To only get an example of how to use the command

```
example("mean")
```

- To start the HTML version of help simply type:

```
help.start()
```

R object

- Container for a piece of data or lines of code
- Objects can be named so they can be accessed at any point.
- Three ways to assign data to a named object:

```
x <- 1
assign("x", 1)
x = 1
```

R functions

- Functions contain lines of pre-written code that performs some task.
  - Gather information about R environment
  - Change properties of an environment
  - Perform task on one or more data structures
  - Below is an example of the function sum()

```
a<-1
b<-4
c<-10
d<-sum(a,b,c)
d
[1] 15
```

R objects: Modes

- The "type" of the components
  - Numeric : numbers
  - Complex : complex numbers
  - Logical : True/False
  - Character : alphanumeric values
  - Raw : bytes

The class of an object

- Class of a vector is the same as a mode
- Other classes: "matrix", "array", "factor" and "data.frame”
- These classes help R act like an object-oriented language.
  - Plot function for an object of class matrix is different than plot for numeric vector.
- Can use “unclass()” if you do not want to treat the object as its a class

R objects: Vectors

- Most basic data structure
- Sequence of data:
  - can be numbers, characters and also logical.
- Scalar is a vector of length 1.
- Vector of more than one element can be created using c() function.
- Elements of vector must be same type and mode.
  - Characters must be enclosed in either single or double quotes
- Missing data can be represented as NA

R objects: Character Vectors

- Denoted by double quotes
  - "x-values", "New iteration results”
- Escape characters \
  - \\ to use \ in string
  - \” to use “ in string
  - \n new line character
  - \t tab character
  - \b backspace character

R object: Logical Vectors

- A vector with three possible values:
  - TRUE, FALSE, NA (Not available)
- Are generated by conditions.
  - temp<- x > 13
- List of Logical operators
  - <, <=, >, >=,
  - == for exact equality
  - != for inequality
  - & (and)
  - | (or)

Arithmetic Operators and Functions

Video 8

- The usual operators
  - + - * / ^
- The usual functions
  - log, exp, sin, cos, tan, sqrt, ...
  - min(x) max(x) length(x) sum(x) prod(x)
  - mean(x) = sum(x)/length(x)
  - var(x) = sum((x-mean(x))^2)/(length(x)-1)
  - sort(x)
- Vector Arithmetic
  - **v <- 2*x + y + 1**

R objects: Factors

- A type of vector that allows you to group together values in a different vector.
- Simplest way to create a factor is to first create a character vector using identical names at the positions of the vector you want to group.
- Then use factor() function to create a factor.

```
mygroupnames<-c("groupA", "groupB", "groupA", "groupC", "groupC", "groupB", "groupA","groupA")

myfactor<-factor(mygroupnames)
```

Video 9

R objects: Arrays and Matrices

- Array: A multiply subscripted collection of data entries
- Matrix: is a two-dimensional array.
- Matrix can be created by using the matrix() function or the array() function.
    - The first argument for both functions is a data vector. Matrix then requires nrow and ncol arguments where as array requires a vector defining the dim property of the array
- The dim() function can be used to convert a vector to a matrix.

R objects: Creating matrices using cbind() and rbind()

- Arguments to cbind() must be either vectors of same length, or matrices with the same column size, that is the same number of rows.
- rbind() is the same as cbind but combines elements as rows.
- For vectors that are shorter than the matrix, the values are cyclically added to the matrix

Different ways to reference

- A vector of positive integral quantities.

```
x[1:10]
mat[c(1,3,4),]
```

- A logical vector

```
mat[c(TRUE,TRUE,FALSE),]
v[v<5]
```

- Using names or vector of character strings

```
fruit <- c(5, 10, 1, 20)
names(fruit) <- c("orange", "banana", "apple", "peach”)

lunch <- fruit[c("apple","orange")]
```

R objects: Data Frames

- A drawback to matrices is that all the values have to be the same mode.
- A dataframe is composed of vectors of the same length but can be of different modes.
  - This makes it perfect structure for mixed-type biomedical data

R objects: Data Frames

- Header of the dataframe can be obtained/set using names() function.
- Specific columns can be accessed using the $ or traditional way for matrix.
  - Dataframe$column
  - Dataframe[,1]
- Row labels can be modified using the rownames() function and similarly column labels can be modified using colnames() function

R objects: Lists

- List is a collection of objects.
- It can contain vectors, matrices, and dataframes of different lengths.
- Great way to collate different information
- To access elements of a list use double square brackets [[]] or names (if they have one)

Some useful commands for objects

- The mode() and typeof() functions provide mode and type of the object.
- The attributes() function provides useful information such as dimensions and names.
- The as() function can be used to coerce one object type to another.

Some functions

- sample() - Get a random sample of numbers
- order() – Returns a numeric vector of the element position in ascending order
- sort() – Returns the values in ascending order
- paste() – Create a character vector by concatenating two other vectors
- print() – Prints content of an object to screen
- range() – Returns minimum and maximum value of a vector
- t() – Transpose a matrix or dataframe