SCS 2211 - Laboratory II

Ms Ashmari Pramodya - ash@ucsc.cmb.ac.lk

University of Colombo School of Computing

Welcome to SCS 2211

Intended Learning Outcomes

- ✓ Perform basic mathematics and simple manipulations with Octave and R
- ✓ Perform basic statistical operations with R
- ✓ Use Octave and R for basic plotting
- ✓ Write and execute scripts in Octave and R
- ✓ Use different data analysis techniques in Octave and R
- ✓ Understand how to explore and describe various real-life data sets, using R

Welcome to SCS 2211

- 2L + 1P
- Evaluation Criteria
    - Final Examination – 70%
    - Assignments – 30%
        - 2 in-class tests (R and Octave)
        - 1 take home assignments (R and Octave)

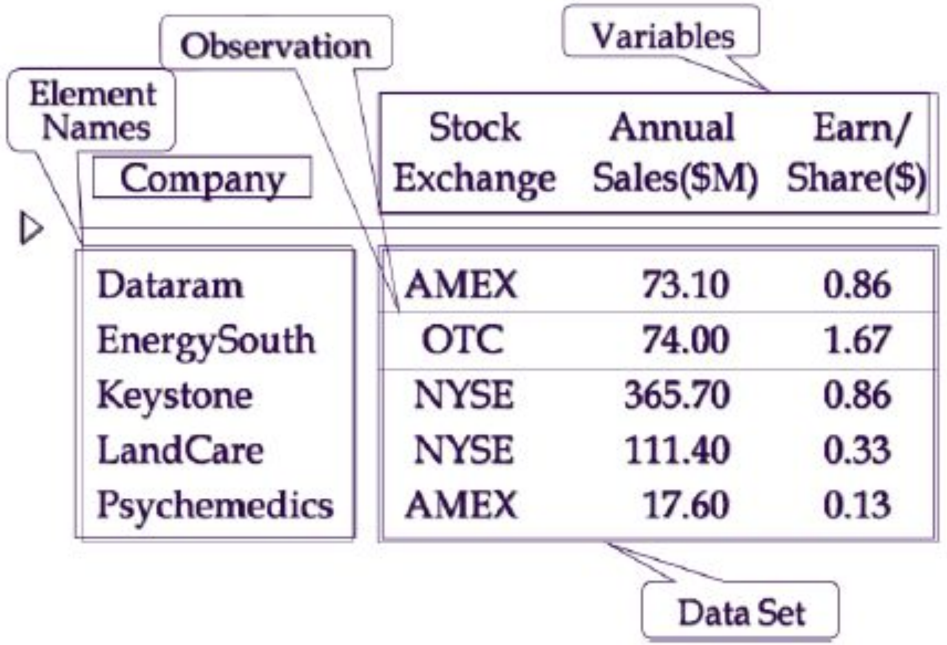
Getting Familiar With The Terminologies

- ❑ **Data**
  - ❑ Facts and figures
- ❑ **Data Set**
  - ❑ All the data collected in a particular study
- ❑ **Elements/Individuals**
  - ❑ Entities on which data are collected
  - ❑ The objects described by a set of data
  - ❑ May be people, animals or things

Getting Familiar With The Terminologies

- **Variable**
  - A characteristic of interest for the elements.
  - Can take different values for different individuals
- **Observation**
  - Set of measurements collected for a particular element

Make and model	Vehicle type	Transmission type	Number of cylinders	City mpg	Highway mpg	Carbon footprint (tons)
⋮						
Aston Martin Vantage	Two-seater	Manual	8	12	19	13.1
Honda Civic	Subcompact	Automatic	4	25	36	6.3
Toyota Prius	Midsize	Automatic	4	51	48	3.7
Chevrolet Impala	Large	Automatic	6	18	29	8.3
⋮						


Observation
Variables
Element Names
Data Set

Company	Stock Exchange	Annual Sales($M)	Earn/ Share($)
Dataram	AMEX	73.10	0.86
EnergySouth	OTC	74.00	1.67
Keystone	NYSE	365.70	0.86
LandCare	NYSE	111.40	0.33
Psychemedics	AMEX	17.60	0.13

Scales of Measurements

4 types

- Nominal
- Ordinal
- Interval
- Ratio

Scales of Measurements cont

- **Nominal**
  - Nominal **labels** or **names** used to identify an attribute of the element
  - Can be recorded using **numeric code**
  - Ex Subject Codes, Gender, ID No
- **Ordinal**
  - Displays the properties of nominal data yet, the ranking or the ordering is meaningful
  - A nonnumeric label or numeric code may be used.
  - Ex Rating of an app, Ranking of students in a class, quality of a service

Scales of Measurements

- **Interval**
  - The interval between observations is expressed in terms of a fixed unit of measure
  - already established, constant, and measurable
  - Ex Temperature, Year
- **Ratio scale**
  - The ratio of two values is meaningful.
  - This scale must contain a zero value.
  - Ex Height, Weight

TASK

1. A course is being rated by students on a scale of 1 star to 5 stars. What is the level of measurement being used in this case?
2. List of popular video games.What is the level of measurement being used in this case?
3. Students' scores on a biology test" is an example of which scale of measurement?

   a. Ratio b. Interval c. Ordinal d. Nominal

TASK

1. A course is being rated by students on a scale of 1 star to 5 stars. What is the level of measurement being used in this case? **c. Ordinal**

2. List of popular video games.What is the level of measurement being used in this case? **d. Nominal**

3. Students' scores on a biology test" is an example of which scale of measurement? a. **Ratio**

a. Ratio b. Interval c. Ordinal d. Nominal

Data - Quantitative Data

- Indicate **how many** or **how much**.
  - Discrete - if measuring how many
  - Continuous - if measuring how much
- Quantitative data are always numeric.
- Ordinary arithmetic operations are meaningful for quantitative data.
- Arithmetic operations make sense

Data - Qualitative Data

- Labels or names used to identify an attribute of each element.
- These are often referred to as **categorical data**
- Use either the **nominal or ordinal scale** of measurements.
- Qualitative data can be either **numeric or nonnumeric** and appropriate statistical analyses for them are rather limited.

Task

1. Identify the kind of variable in this situation: number of paper clips in mason jar on Mr. Rhoades' desk

    a. Discrete b. Continuous

2. Which of the following variables is most likely to be used as categorical?

    a. length of a garden hose **c. recorded rainfall for a day**

    b. flower color **d. weight of books on a shelf**

3. Which of the following variables is most likely to be used as quantitative?

    a. **ZIP code** **b. Google Drive capacity** **c. country of birth**

Task

1. Identify the kind of variable in this situation: number of paper clips in mason jar on Mr. Rhoades' desk
   a. **Discrete** **b. Continuous**
2. Which of the following variables is most likely to be used as categorical?
   a. **length of a garden hose** **c. recorded rainfall for a day**
   b. **flower color** **d. weight of books on a shelf**

3. Which of the following variables is most likely to be used as quantitative?

a. **ZIP code** **b. Google Drive capacity** **c. country of birth**

Introduction to R

What is R

- R is a free software environment for data manipulation, data analysis, statistical computing and graphics display
- A programming language
- Natural to use, complete data analyses in just a few lines

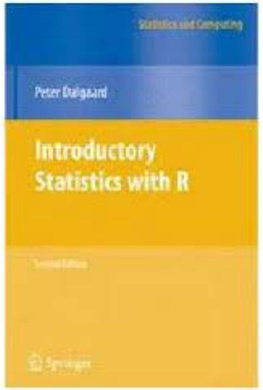
History of R

- Originated from **S**
    - A statistical programming language developed by John Chambers at Bell Labs in the 1970 s
- The first version of R was developed by Robert Gentleman and Ross Ihaka at the University of Auckland in the mid 1990's
- Freely available under the GNU General Public License

Why R

- Free and Open source
- Statistical programming
- Graphics
- Operating system agnostic
- Linear and nonlinear model
- Built in testing and help
- Many add-on

Learning Resources

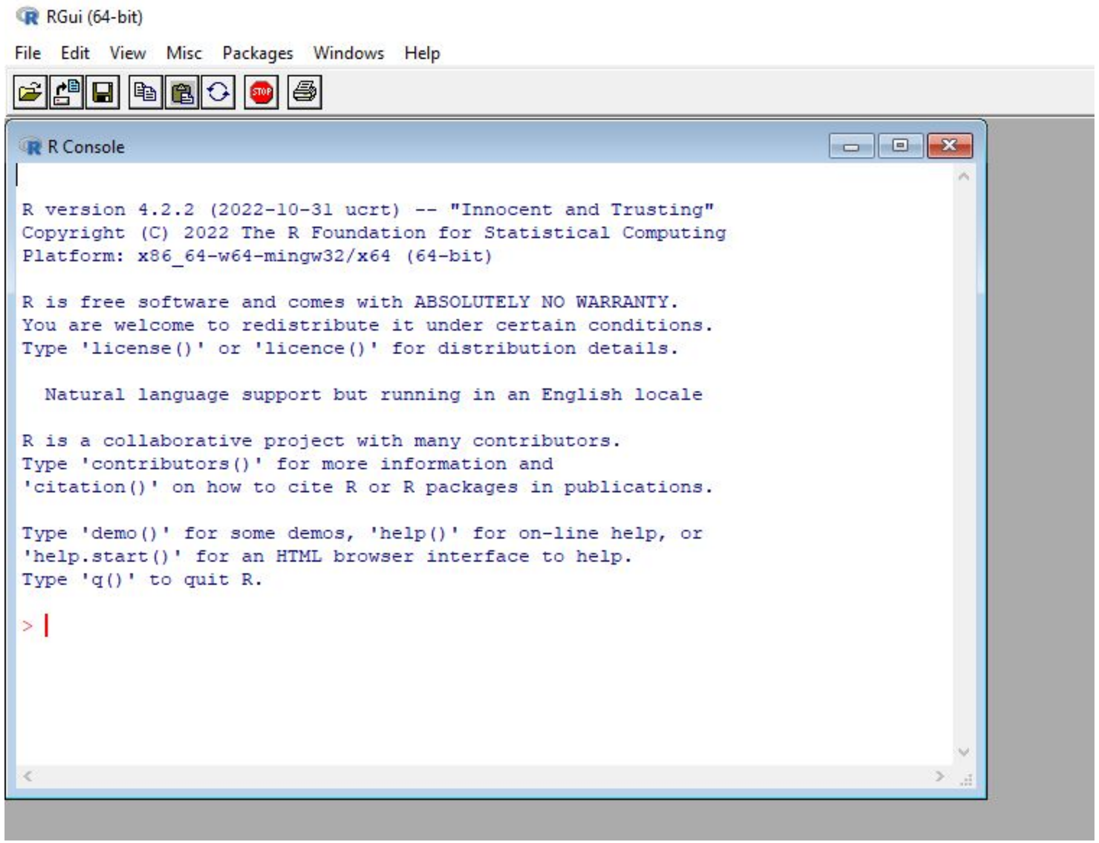


- “Introductory Statistics with R“, by Peter Dalgaard,Springer (2002)
- An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics, by W. N. Venables, D. M. Smith.
- An Introduction R: Introduction and examples, by Deepayan Sarkar
  - https://www.isid.ac.in/~deepayan/R-tutorials/
  - https://www.isid.ac.in/~deepayan/R-tutorials/labs/

Installing R

- Download relevant binary from
  - https://cran.r-project.org/mirrors.html and Install.
- Official page https://www.r-project.org/

R console



```
R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 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.

>
```

R help

R has an inbuilt help facility

- Functions
  - **`help(<keyword>)`**
  - **`?<keyword>`**

R searching

- Searching
  - **`?help.search`**
- HTML format
  - **`help.start()`**
- For examples
  - **`example (<topic>)`**

Objects

R works on objects.

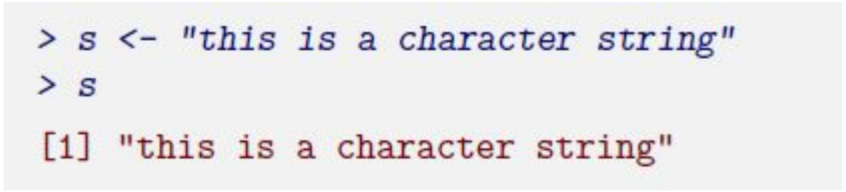
Objects have following properties

- mode : type of object
- length : no: of components of the object
- Objects store information.
- Data sets are stored in “data frame object”s.
- R has several important kinds of objects; for example: functions, vectors (numeric, character, logical), matrices, lists, and data frames.

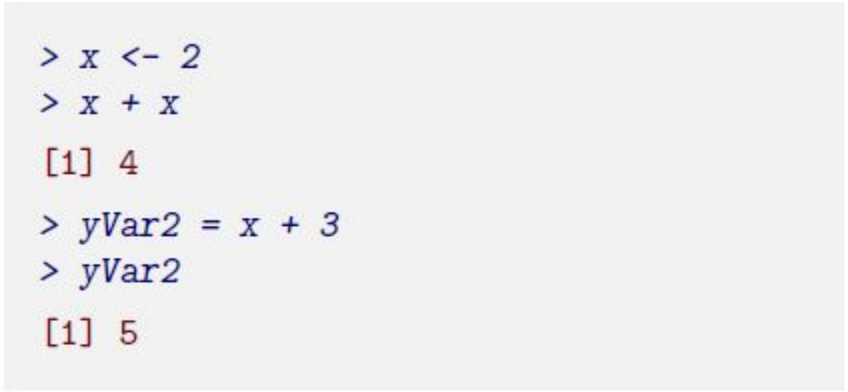
R Objects

Results of calculations can be stored in objects using the assignment operators:

- An arrow (<-) formed by a smaller than character and a hyphen without a space!
- The equal character (=).



```
> s <- "this is a character string"
> s
[1] "this is a character string"
```



```
> x <- 2
> x + x
[1] 4
> yVar2 = x + 3
> yVar2
[1] 5
```

Identifiers

Identifier is used to name your objects

Rules for writing Identifiers in R

- Identifiers can be a combination of letters, digits, period (.) and underscore (_).
- It must start with a letter or a period. If it starts with a period, it cannot be followed by a digit.
- Reserved words in R cannot be used as identifiers.
- R is case sensitive, X and x are two different objects, as well as temp and temP.

Unique name given to variable (function and objects as well) is identifier

Identifiers

- Valid identifiers in R

  total, Sum, .fine.with.dot, this_is_acceptable, Number5

- Invalid identifiers in R

  tot@l, 5um, _fine, TRUE, .0ne

Best Practices

For example, **a.variable.name** is preferred **over a_variable_name** or alternatively we could use camel case as **aVariableName**

R is a case sensitive language.

FOO, Foo, and foo are three different objects

R Functions, Arguments and Return Values

R works by calling functions

- A function takes a collection of inputs and maps them to a single output
  - Inputs - Arguments
  - Output - Return Value
- User can define their own functions
- They can be assigned to variables,
- They can be used as arguments in other function calls.

Everything that exists is an object.

Everything that happens is a function call.

Computations in R

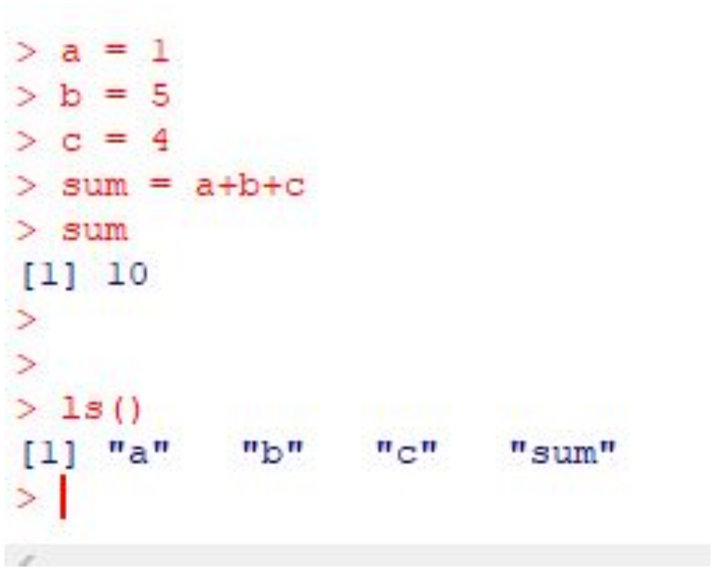
R session Management

- R has the ability to save objects, to be loaded again later.
- Whenever exiting, R asks to save all the variables created by the user, and restores them when starting up the next time (in the same directory).
- To exit the R session, type quit()

```
> quit()
```

R sessions

- Listing the objects in the current R session
- We can list the names of the objects in the current R session by ls() command. For example, start R session fresh and proceed as follows:

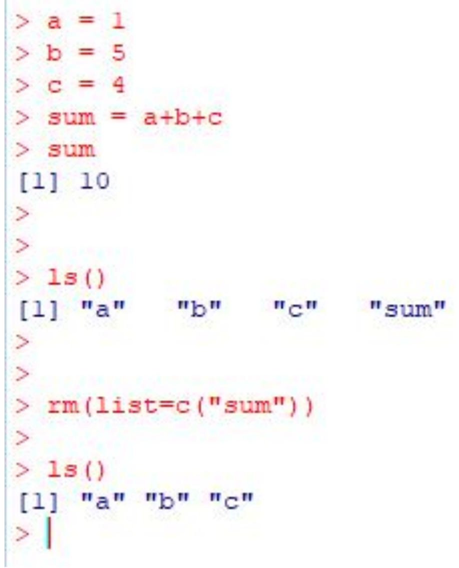


```
> a = 1
> b = 5
> c = 4
> sum = a+b+c
> sum
[1] 10
>
>
> ls()
[1] "a"   "b"   "c"   "sum"
> 
```

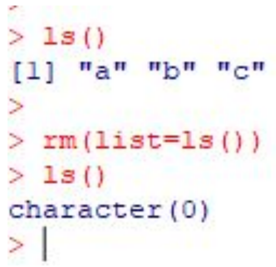
Removing objects from the current R session

Specific objects created in the current session can be removed using **rm()** command. If we specify the name of an object, it will be removed.

If we just say **rm(list = ls())** , all objects created so far will be removed.



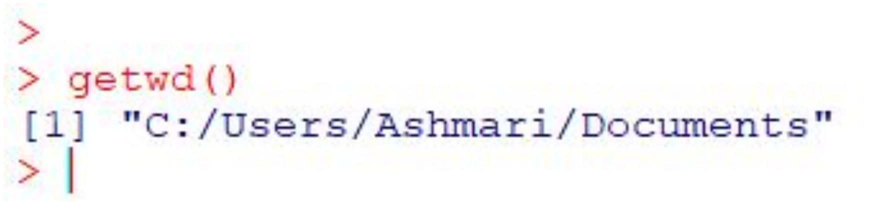
```
> a = 1
> b = 5
> c = 4
> sum = a+b+c
> sum
[1] 10
>
>
> ls()
[1] "a"    "b"    "c"    "sum"
>
>
> rm(list=c("sum"))
>
> ls()
[1] "a" "b" "c"
> |
```



```
> ls()
[1] "a" "b" "c"
>
> rm(list=ls())
> ls()
character(0)
> |
```

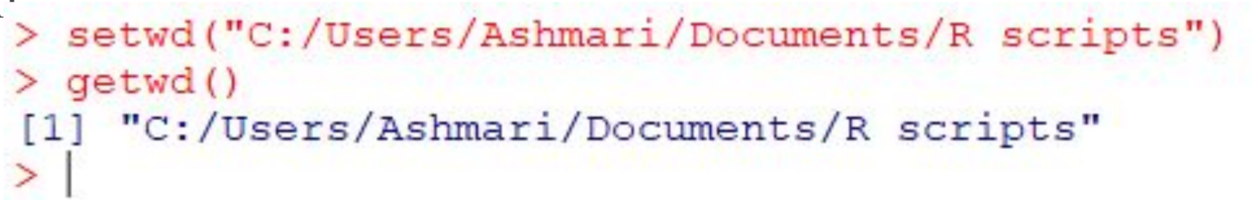
Getting and setting the current working directories

- From R prompt, we can get information about the current working directory using **getwd()** command:



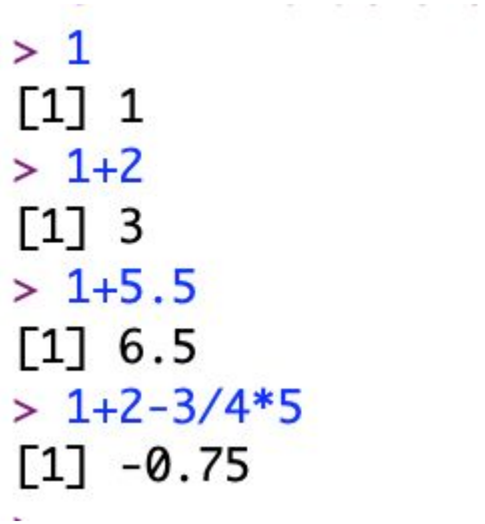
```
>
> getwd()
[1] "C:/Users/Ashmari/Documents"
> |
```

- Similarly, we can set the current working directory by calling **setwd()** function:



```
> setwd("C:/Users/Ashmari/Documents/R scripts")
> getwd()
[1] "C:/Users/Ashmari/Documents/R scripts"
> |
```

R as a Simple Calculator



```
> 1
[1] 1
> 1+2
[1] 3
> 1+5.5
[1] 6.5
> 1+2-3/4*5
[1] -0.75
```

R as a Simple Calculator

Build in character constants

- LETTERS
    - The 2 upper-case letters of the roman alphabet
- letters
    - 26 lower case letters of the roman alphabet
- month.abb
    - The three letter abbreviations for english months names
- month.name
    - The english names for months of the year

R as a Simple Calculator

- Imaginary numbers
  - sqrt(-1) = NaN   sqrt(-1 + 0i) = 0 + 1i
- Trigonometric functions
  - sin(),cos(), tan()
- Logarithms
  - log(5)
  - log10(5)
  - exp

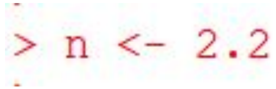
Order of precedence

Operator precedence

- Exponential and roots
- Multiplication and division
- Addition and subtraction

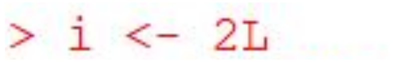
Data Types in R

Numeric - Decimal (floating point values) are part of the numeric class in R



```
> n <- 2.2
```

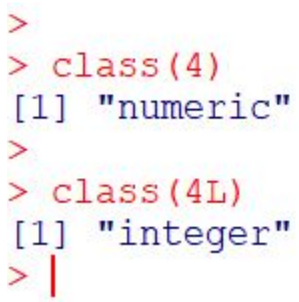
Integer - Natural (whole) numbers are known as integers and are also part of the numeric class



```
> i <- 2L
```

Data Types in R

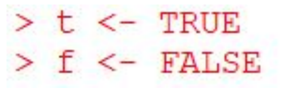
- When you write numbers like 4 and 3, they are interpreted as floating-point numbers.
- To explicitly get an integer, you must write 4L and 3L



```
>
> class(4)
[1] "numeric"
>
> class(4L)
[1] "integer"
>
```

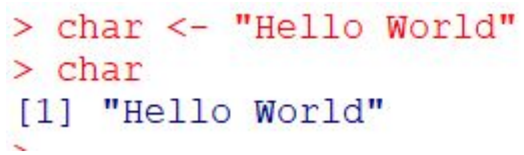
Data Types in R cont.

Logical - Boolean values (True and False) are part of the logical class. In R these are written in All Caps.



```
> t <- TRUE
> f <- FALSE
```

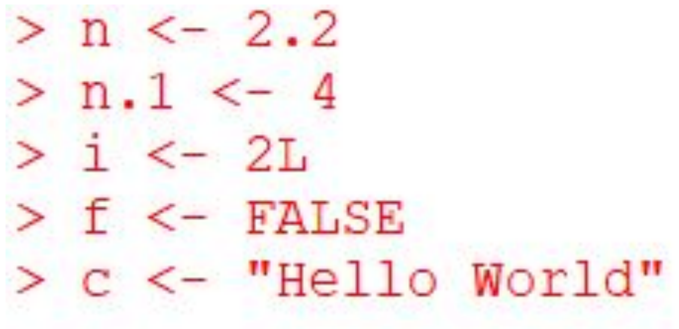
Characters - Text/string values are known as characters in R. You use quotation marks to create a text character string:



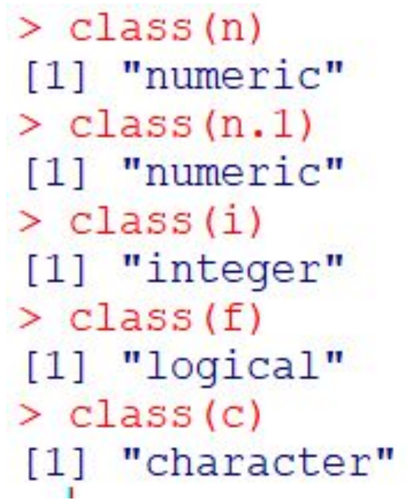
```
> char <- "Hello World"
> char
[1] "Hello World"
```

Data Types in R cont.

You can use the class() function to check the data type of a variable:



```
> n <- 2.2
> n.1 <- 4
> i <- 2L
> f <- FALSE
> c <- "Hello World"
```



```
> class(n)
[1] "numeric"
> class(n.1)
[1] "numeric"
> class(i)
[1] "integer"
> class(f)
[1] "logical"
> class(c)
[1] "character"
```

Special Types

In addition, there are four special constants,

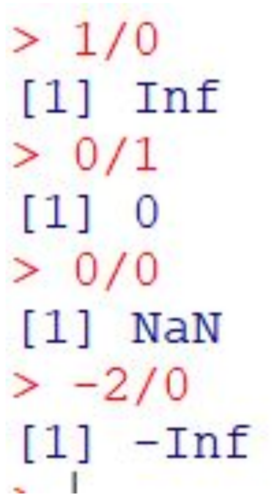
– NULL : used to indicate the empty object

– NA : for absent (“Not Available”) data values

– Inf : denotes infinity

– NaN: is not-a-number

E.x.: Determine 1/0, 0/1, 0/0, -2/0



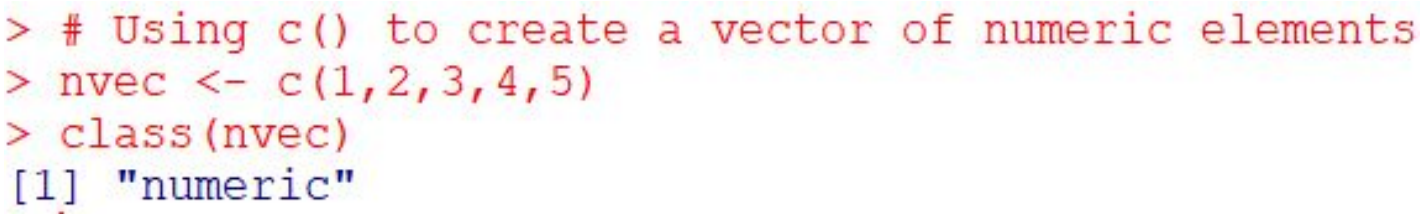
```
> 1/0
[1] Inf
> 0/1
[1] 0
> 0/0
[1] NaN
> -2/0
[1] -Inf
```

R-Objects

- The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable.
- There are many types of R- objects. The frequently used ones are
  - Vectors
  - Lists
  - Matrices
  - Arrays
  - Factors
  - Data Frames

Vectors

- A vector is the most common and basic data structure in R and is pretty much the workhorse of R.
- Create an empty vector with vector()
  - The general pattern is vector(class of object, length).
- The function c() is used to create vectors:



```
> # Using c() to create a vector of numeric elements
> nvec <- c(1,2,3,4,5)
> class(nvec)
[1] "numeric"
```

Vectors cont.

- x is a numeric vector
    - `x <- c(1, 2, 3)`
- Then if we want to add 2 to everything in this vector, or to square each entry:
- You can also have logical vectors.
    - `y <- c(TRUE, TRUE, FALSE, FALSE)`
- Finally you can have character vectors:
    - `z <- c("Alec", "Dan", "Rob", "Rich")`

Vectors cont.

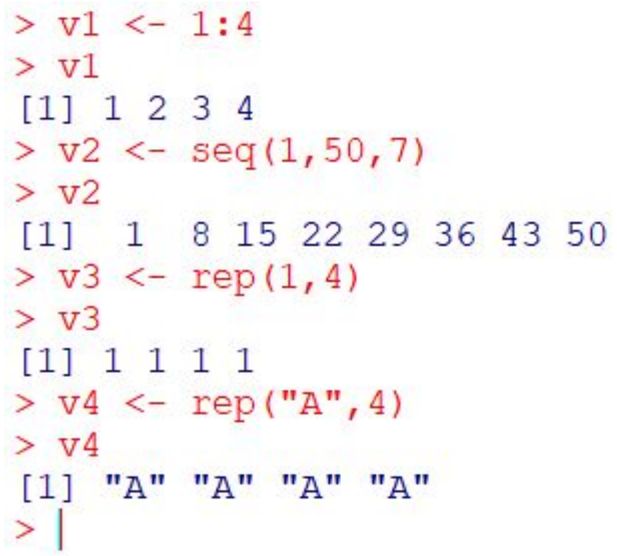
- Examine your vector
    - `typeof(z)`
    - `length(z)`
    - `class(z)`
    - `str(z)`

Vectors cont.

---

seq() creates a sequence of equidistant numbers

```
> seq(from=1, to=50, by=7)
```

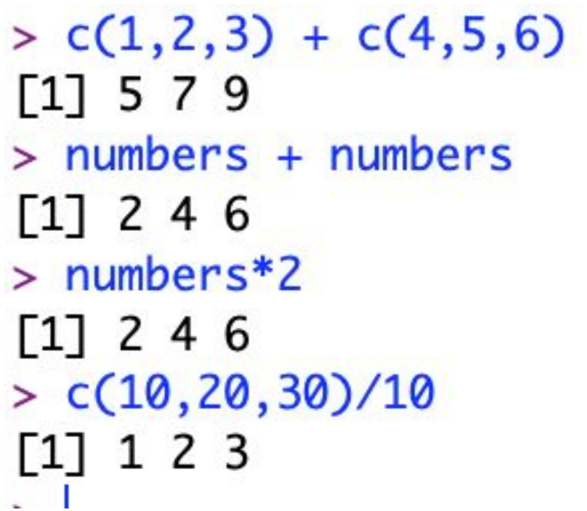


```
> v1 <- 1:4
> v1
[1] 1 2 3 4
> v2 <- seq(1,50,7)
> v2
[1]  1  8 15 22 29 36 43 50
> v3 <- rep(1,4)
> v3
[1] 1 1 1 1
> v4 <- rep("A",4)
> v4
[1] "A" "A" "A" "A"
> |
```

Sometimes it's necessary to have repeated values, for which we use rep()

Vectors cont.

Simple arithmetic operations can be performed on vectors



```
> c(1,2,3) + c(4,5,6)
[1] 5 7 9
> numbers + numbers
[1] 2 4 6
> numbers*2
[1] 2 4 6
> c(10,20,30)/10
[1] 1 2 3
```

Home Work

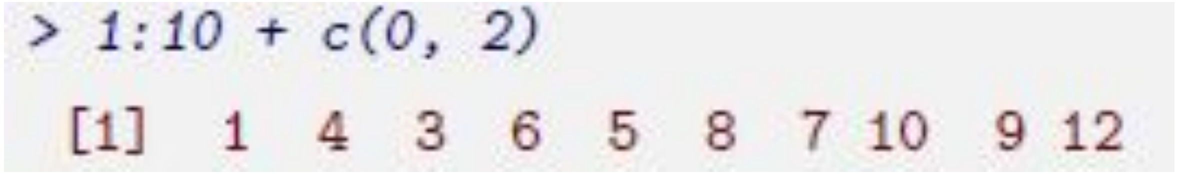
Find out what happens if we mixed the data types in a vector.

Add vectors of different lengths?

When two vectors are not of equal length, the shorter one is recycled.

– E.g.: The following adds 0 to all the odd elements and 2 to all

the even elements of 1:10

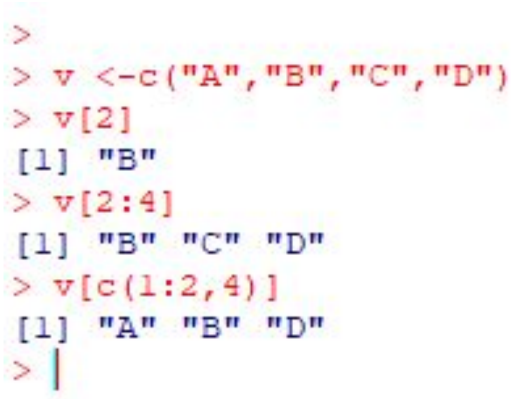


```
> 1:10 + c(0, 2)
 [1]  1  4  3  6  5  8  7 10  9 12
```

Indexing vectors

We use the indexing to reach certain elements of the vector

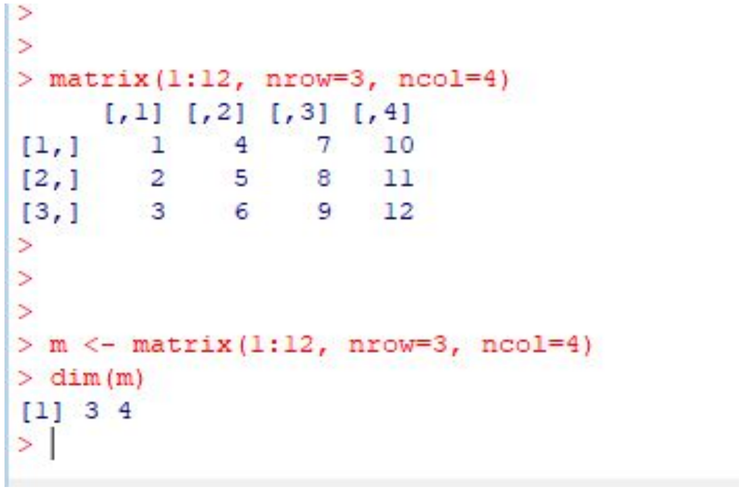
- R indices start at 1



```
>
> v <-c("A","B","C","D")
> v[2]
[1] "B"
> v[2:4]
[1] "B" "C" "D"
> v[c(1:2,4)]
[1] "A" "B" "D"
> 
```

Matrices

- Matrices are a special vector in R.
- They are not a separate class of object but simply a vector but now with dimensions added on to it.
- Matrices have rows and columns
- Matrices are constructed **column wise.**



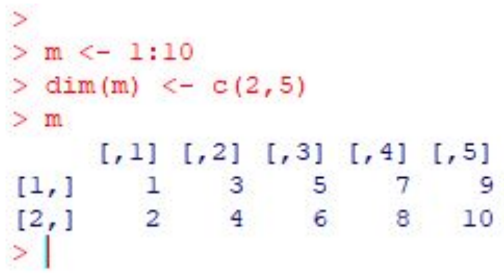
```
>
>
> matrix(1:12, nrow=3, ncol=4)
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
>
>
>
> m <- matrix(1:12, nrow=3, ncol=4)
> dim(m)
[1] 3 4
> |
```

Matrices

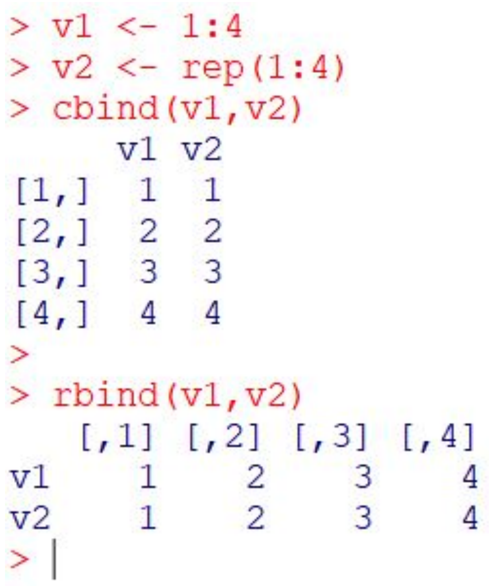
Ways to construct a matrix

```
m <- 1:10
dim(m) <- c(2,5)
```

- Another way is to bind columns or rows using **cbind()** and **rbind()**.



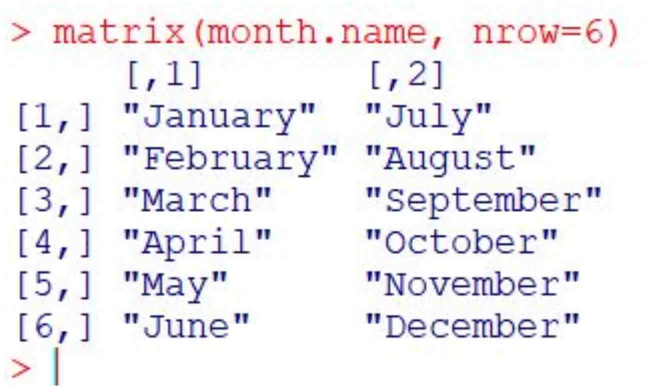
```
>
> m <- 1:10
> dim(m) <- c(2,5)
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> |
```



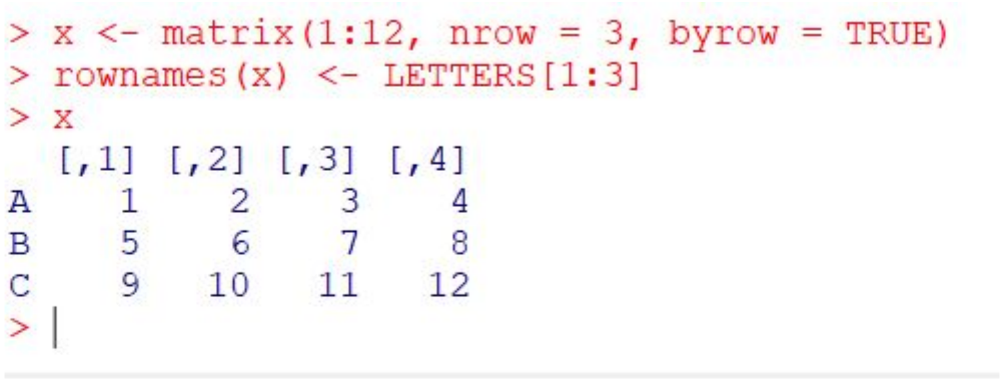
```
> v1 <- 1:4
> v2 <- rep(1:4)
> cbind(v1,v2)
     v1 v2
[1,]  1  1
[2,]  2  2
[3,]  3  3
[4,]  4  4
>
> rbind(v1,v2)
   [,1] [,2] [,3] [,4]
v1    1    2    3    4
v2    1    2    3    4
> |
```

Matrices

Matrices do not need to be numeric, there can be character or logical matrices as well:



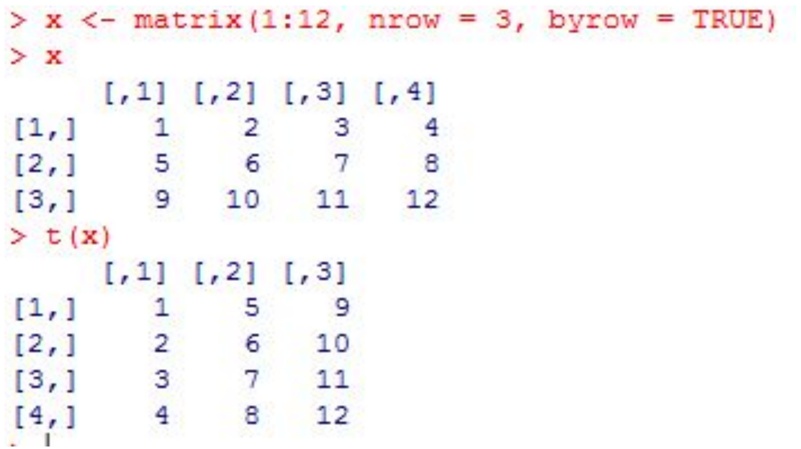
```
> matrix(month.name, nrow=6)
     [,1]        [,2]
[1,] "January"   "July"
[2,] "February"  "August"
[3,] "March"     "September"
[4,] "April"     "October"
[5,] "May"       "November"
[6,] "June"      "December"
> 
```



```
> x <- matrix(1:12, nrow = 3, byrow = TRUE)
> rownames(x) <- LETTERS[1:3]
> x
  [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
> 
```

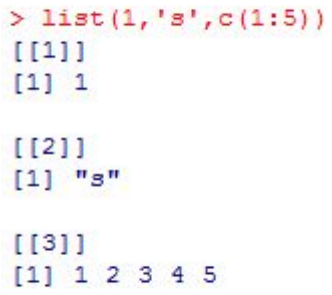
Matrices

Transpose : t() function



```
> x <- matrix(1:12, nrow = 3, byrow = TRUE)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> t(x)
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

List



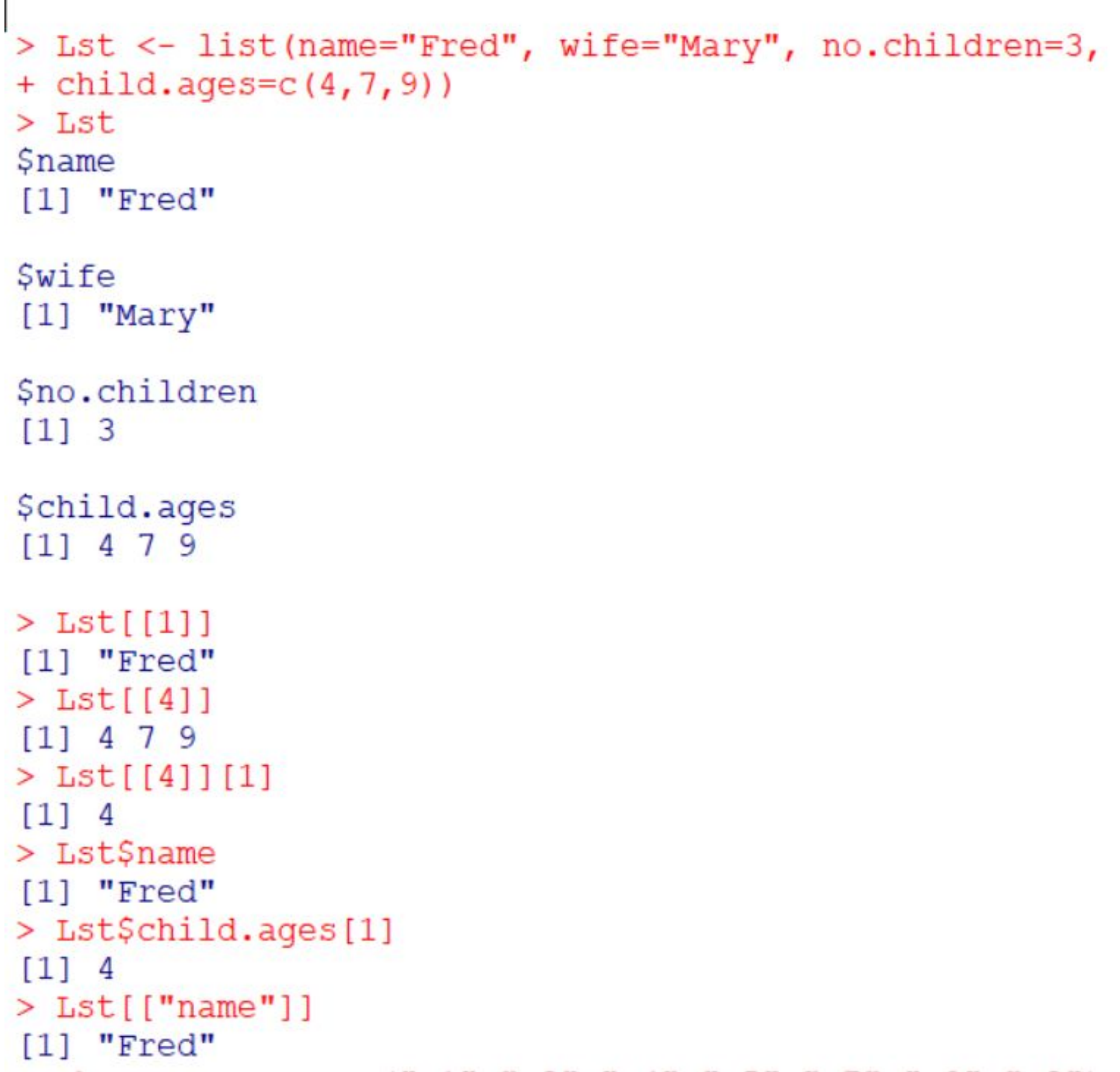
```
> list(1,'s',c(1:5))
[[1]]
[1] 1

[[2]]
[1] "s"

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

- Lists are very flexible data structures used extensively in R.
- A list is a vector, but the elements of a list do not need to be of the same type.
    - Each element of a list can be any R object, including another list
- Lists could consist components with different mode

List



```
> Lst <- list(name="Fred", wife="Mary", no.children=3,
+ child.ages=c(4,7,9))
> Lst
$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9

> Lst[[1]]
[1] "Fred"
> Lst[[4]]
[1] 4 7 9
> Lst[[4]][1]
[1] 4
> Lst$name
[1] "Fred"
> Lst$child.ages[1]
[1] 4
> Lst[["name"]]
[1] "Fred"
```

Thank You

Feedback / Comments

https://forms.gle/dwboWCsoCzpjMdFc7