**R**

* Language for Statistical Computing & Data Science
* Millions of users around the world
* 100% free and open source!
* Best in-class data visualization
* Thousands of R packages (extensions to R)
* Vibrant and helpful community
* Real programming language

**Variables**

Store a variable to reuse later. Use “->” sign to create a variable.

height -> 2 #This will not print anything just save it for reuse

height #[1] 2…… R returns 2

**Workspace**

ls() #This will show all the variables/objects

**The Basic Data Types**

class() **to reveal type**

* **Logical:** TRUE/T, FALSE/F, NA
* Numeric: 2, 2.5 (to specify a number is integer use: 2L)

To see a variable is numeric: as.numeric(2) / as.numeric(2L)

Integer is numeric but numeric not always integer.

* Character: “I Love Data Science!”
* Double: higher precession
* Complex: complex number
* Raw: store raw bytes
* Coercion: converting data type but it is not always possible

As.numeric(TRUE) = 1

As.numeric(FALSE) = 0

As.character(4) = “4”

As.numeric(“4.5”) = 4.5

As.integer(4.5) = 4

As.numeric(“Hello”) = NA, warning message: NAs introduced by coercion

**Create and Name Vectors**

* Vector:
  + Sequence of data elements
  + Same basic type
  + Character, numeric, logical
* Create a vector: c()
  + C(“a”, ”bb”, “ja12”)
  + X <- c(“a”, ”bb”, “ja12”)
  + is.vector(x)
  + c(6, 11, 20, 54)
* Name a vector:
  + Remain <- (11, 12, 11, 13)
  + Suits <- (“spades”, “hearts”, “diamonds”, “clubs”)
  + Names(remain) <- suits
  + Remain = Now it will print :
  + Spades hearts diamonds clubs
    - 11 12 11 13
* Another way to Name a vector
  + Remain <- c(spades=11, hearts=12, diamonds=11, clubs=13)
  + OR remain <- c(“spades”=11, “hearts”=12, “diamonds”=11, “clubs”=13)
  + Str(remain)
* Single value = vector:
  + My\_apples <- 5
  + My\_oranges <- “six”
  + Is.vector(my\_apples)
  + Length(apples) = 1
  + Length(drawn\_suits) = 5
* Vectors are homogenous
  + Only element of the same type
  + Atomic vectors <> lists
  + Automatic coercion if necessary(Can hold element of different types)
* Coercion for vectors
  + Drawn\_ranks <- c(7, 4, “A”, 10, “K”, 3, 2, “Q”)
  + Drawn\_ranks = (output): “7”, “4”, “A”, “10”, “K”, “3”, “2”, “Q”
  + Class(drawn\_ranks) = “character”
* Vector Calculus
  + My\_apples <- 5
  + My\_oranges <- 6
  + My\_apples + my\_oranges : output(11)
  + Computations are performed **element-wise**
  + Earnings <- c(50, 10, 30)
  + Earnings \* 3 : output(150, 30, 90)
  + Earning / 3, earnings – 20, earnings + 100, earnings ^ 2
  + Expenses <- c(30, 40, 80)
  + Earnings – expenses, Earnings + expenses, Earnings \* expenses, Earnings / expenses : all will be done element-wise operation
  + Sum(expense) : output will be = 150
  + Earnings > expenses : output will be = TRUE FALSE FALSE

**Subsetting vectors in R**

* Subset by index

Remain <- c(spades = 11, hearts = 12, diamonds = 10, clubs = 13)

Remain[1] output would be: spades

11

Remain[3] outoput would be: diamonds

10

[3] = Takes element at index 3, result is a (named) vector too!

* Subset by name

Remain[“spades”] output would be: spades

11

* Subset multiple elements

Remain\_black <- remain[c(1, 4)]

Output would be: spades clubs

11 13

Order in selection vector matters!

Remain\_black <- remain[c(4,1)]

Output would be: clubs spades

13 11

* Remain[c(“spades”, “clubs”)]

Output would be:

Spades clubs

11 13

* Subset all but some

Remain[-1]

Output would be: All but index 1 are returned

Hearts diamonds clubs

12 10 13

Remain[-c(1, 2)]

Output would be:

Diamonds clubs

10 13

BUT

Remain[-“spades”]

Output would be:

Error in –“spades”: Invalid argument to unary operator

* Subset using logical vector

Remain[c(FALSE, TRUE, FALSE, TRUE)]

Output would be:

Hearts clubs

12 13

Selection\_vector <- c(FALSE, TRUE, FALSE, TRUE)

Remain[selection\_vector]

Output would be:

Hearts clubs

12 13

Remain[c(TRUE, FALSE)] will be converted into remain[c(TRUE, FALSE, TRUE, FALSE)]

Reman[c(TRUE, FALSE, TRUE)] will be converted to remain[c(TRUE, FALSE, TRUE, FALSE)]

**How to Create and Name Matrices in R**

* Vector: 1D Array of data elements
* Matrix: 2D Array of data elements
* Rows and columns
* One atomic vector type
* Create a Matrix:

Matrix(1:6, nrow=2)

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

Matrix(1:6, ncol=3)

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

Matrix(1:6, nrow=2, byrow=TRUE)

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 4 5 6

* Create a Matrix: recycling

Matrix(1:3, nrow=2, ncol=3)

[,1] [,2] [,3]

[1,] 1 3 2

[2,] 2 1 3

Matrix(1:4, nrow=2, ncol=3)

[,1] [,2] [,3]

[1,] 1 3 1

[2,] 2 4 2

Warning Message:

In matrix(1:4, nrow = 2, ncol = 3):

Data length [4] is not a sub-multiple or multiple of the number of columns [3]

* Rbind(), cbind()

Cbind(1:3, 1:3)

[,1] [,2]

[1,] 1 1

[2,] 2 2

[3,] 3 3

Rbind(1:3, 1:3)

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 1 2 3

M <- matrix(1:6, byrow=TRUE, nrow=2)

Rbind(m, 7:9)

[,1] [,2] [,3]

[1,] 1 2 3

[2,] 4 5 6

[3,] 7 8 9

Cbind(m, 10:11)

[,1] [,2] [,3] [,4]

[1,] 1 2 3 10

[2,] 4 5 6 11

* Naming a Matrix: rownames(), colnames()

M <- matrix( 1:6, byrow=TRUE, nrow=2)

Rownames(m) <- c(“row1”, “row2”)

M

Output:

[,1] [,2] [,3]

Row1 1 2 3

Row2 4 5 6

Colnames(m) <- c(“col1”, “col2”, “col3”)

M

Output:

Col1 col2 col3

Row1 1 2 3

Row2 4 5 6

M<- matrix(1:6, byrow=TRUE, nrow = 2, dimnames = list( c(“row1”, “row2”),

C(“col1”, “col2”, “col3”)) )

M

Output:

Col1 col2 col3

Row1 1 2 3

Row2 4 5 6

* Coercion

Char <- matrix(LETTERS[1:6], nrow=4, ncol=3)

Char

[,1] [,2] [,3]

[1,] “A” “E” “C”

[2,] “B” “F” “D”

[3,] “C” “A” “E”

[4,] “D” “B” “F”

Num<- matrix(1:8, ncol=2)

[,1] [,2]

[1,] 1 5

[2,] 2 6

[3,] 3 7

[4,] 4 8

Cbind(num,char)

[,1] [,2] [,3] [,4] [,5]

[1,] “1” “5” “A” “E” “C”

[2,] “2” “6” “B” “F” “D”

[3,] “3” “7” “C” “A” “E”

[4,] “4” “8” “D” “B” “F”

**Subset Matrix**

* Subset Element

M <- matrix(sample(1:15, 12) nrow = 3)

M

[,1] [,2] [,3] [,4]

[1,] 5 11 15 3

[2,] 12 14 8 9

[3,] 6 1 4 2

M[1, 3] = 15

M[3, 2] = 1

* Subset column or row:

M[3,] = 6 1 4 2 (it becomes a vector)

M[,3] = 15 8 4

M[4] = 11 (matrix goes to the column to column from left to rights)

M[9] = 4

* Subset Multiple elements

M[2, c(2, 3)] = 14 8

M[c(1,2), c(2,3) ] Now the output would be:

[,1] [,2]

[1,] 11 15

[2,] 14 8

M[c(1, 3), c(1, 3, 4)] Now the output would be:

[,1] [,2] [,3]

[1,] 5 15 3

[2,] 6 4 2

* Subset by name

Rownames(m) <- c(“r1”, “r2”, “r3”)

Colnames(m) <- c(“a”, “b”, “c”, “d”)

M

A b c d

R1 5 11 15 3

R2 12 14 8 9

R3 6 1 4 2

M[2,3] = 8

M[“r2”, “c”] = 8

M[2, “c”] = 8

M[3, c(“c”, “d”)]

C d

4 2

* Subset with logical vector

M[c(FALSE, FALSE, TRUE),

C(FALSE, FALSE, TRUE, TRUE)]

C d

4 2

M[c(FALSE, FALSE, TRUE),

C(FALSE, TRUE)]

B d

1 2

M[c(FALSE, FALSE, TRUE),

C(FALSE, TRUE, FALSE, TRUE)]

B d

1 2

**Matrix Arithmatic**

* colSums(), rowSums()
* Standard arithmetic possible
* Element-wise computation
* Lotr\_matrix

the\_fellowship <- c(316, 556)

two\_towers <- c(343, 584)

return\_king <- c(378, 742)

lotr\_matrix <-rbind(the\_fellowship, two\_towers, return\_king)

colnames(lotr\_matrix) <- c(“US”, “non-US”)

rownames(lotr\_matrix) <- c(“Fellowship”, “Two towers”, “Return king”)

lotr\_matrix

the output would be:

US non-US

Fellowship 316 556

Two towers 343 584

Return king 378 742

* Matrix Scalar:

Convert dollars to Euros:

Lotr\_matrix / 1.12

* Matrix

# Definition of theatre\_cut omitted

Theatre\_cut

[,1] [2,]

[1,] 50 50

[2,] 80 80

[3,] 100 100

Lotr\_matrix – theatre\_cut

Output would be:

US non-US

Fellowship 266 506

Two towers 263 504

Return king 278 642

* Recycling

Lotr\_matrix – c(50,80, 100)

US non-US

Fellowship 266 506

Two towers 263 504

Return king 278 642

* Matrix Multiplication

Multiplication simply perform element-wise

#Definition of rates omitted

Rates

[,1] [,2]

[1,] 1.11 1.11

[2,] 0.99 0.99

[3,] 0.82 0.82

Lotr\_matrix \* rates

Oytput would be:

US non-US

Fellowship 350.76 617.16

Two towers 339.57 578.16

Return king 309.96 608.44

* Matrices and Vectors
  + Very similar
  + Vector = 1D, Matrix = 2D
  + Coercion if necessary
  + Recycling if necessary
  + Element wise calculations

**Factors**

* Categorical Variables
  + Limited number of different values
  + Belong to category
  + In R: factor
* Create Factor

Blood <- c(“B”, “AB”, “O”, “A”, “O”, “O”, “A”, “B”)

Blood

Output would be:

“B”, “AB”, “O”, “A”, “O”, “O”, “A”, “B”

Blood\_factor <- factor(blood)

Blood\_factor

Output would be:

[1] B AB O A O O A B

Levels: A AB B O (R sorts levels alphabetically!)

Str(blood\_factor)

Output would be:

Factor w/4 levels “A”, “AB”, “B”, “O”: 3 2 4 1 4 4 1 3

Why this conversion?

In categories it can be very long character strings. Each time repeating this string per observation can take a lot of memory. By using this simple encoding much less space necessary. Factors are actually integer vectors, for each integer corresponds to category or a level.

* Order levels differently

Blood\_factor2 <- factor(blood, levels=c(“O”, “A”, “B”, “AB”)

Blood\_factor2

Output would be:

[1] B AB O A O O A B

Levels: O A B AB

Str(blood\_factor2)

Output would be:

Factor w/4 levels “O” “A” “B” “AB”: 3 4 1 2 1 1 2 3

Str(blood\_factor)

Output would be:

Factor w/4 levels “A” “AB” “B” “O”: 3 2 4 1 4 4 1 3

* Rename factor levels

Blood <- c(“B”, “AB”, “O”, “A”, “O”, “O”, “A”, “B”)

Blood\_factor <- factor(blood)

Levels(blood\_factor) <- c(“BT\_A”, “BT\_AB”, “BT\_B”, “BT\_O”)

Blood\_factor

Output would be:

BT\_B BT\_AB BT\_O BT\_A BT\_O BT\_O BT\_A BT\_B

Levels: BT\_A BT\_AB BT\_B BT\_O

Factor(blood, lebels = c (“BT\_A”, “BT\_AB”, “BT\_B”, “BT\_O”))

Output would be:

BT\_B BT\_AB BT\_O BT\_A BT\_O BT\_O BT\_A BT\_B

Levels: BT\_A BT\_AB BT\_B BT\_O

Blood <- c(“B”, “AB”, “O”, “A”, “O”, “O”, “A”, “B”)

Blood\_factor <- factor(blood)

Factor(blood,

Levels = c(“O”, “A”, “B”, “AB”),

Levels = c(“BT\_O”, “BT\_A”, “BT\_B”, “BT\_AB”) )

* Nominal Vs Original

Blood\_factor[1] < blood\_factor[2]

[1] NA

Warning message:

In Ops.factor(blood\_factor[1], blood\_factor[2]) :

‘<’ not meaningful for factors

Tshirt <- c( “M”, “L”, “S”, “S”, “L”, “M”, “L”, “M”)

Tshirt\_factor <- factor(tshirt, ordered = TRUE, levels=c(“S”, “M”, “L”)

Tshirt\_factor

Output would be:

[1] M L S S L M L M

Levels: S < M < L

* Wrap-up
  + Factors are categorical variables
  + Factors are integer vectors
  + Change factor levels:

Levels() function or labels argument

* + Ordered factors: ordered = TRUE