Lec 1(1) Basic of R I

Xinan Wang

2023-01-18

Install Required Packages

install.packages('installr')

install.packages('tidyverse')

library(tidyverse)

install.packages('rmarkdown')

if (!requireNamespace('devtools'))

install.packages('devtools')

devtools::install_github('rstudio/rmarkdown')

install.packages('tinytex')

tinytex::install_tinytex()

install.packages('ggplot2')

install.packages('nycflights13')

install.packages('gapminder')

libs <- c('ggplot2','nycflights13','gapminder')

x <- sapply(libs, function(x)

if (!require(x, character.only = T, warn.conflicts = F, quietly = T)) install.packages(x))

Basic Math

```
1 + 1
```

```
## [1] 2
```

```
1/(1+1)
```

```
## [1] 0.5
```

Function in R

```r
if (sample(10,1)%in%c(1:9)) {
  print('My students love me')
} else {
  'They hate me'
}
```

```
## [1] "My students love me"
```

Variable

```r
x <- 13
x
```

```
## [1] 13
```

```r
x = 13
x
```

```
## [1] 13
```

```r
assign('x',13)
x
```

```
## [1] 13
```

```r
# Remove variables
x <- 13
rm(x)
```

Data Types

```r
x <- 13
class(x)
```

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

```r
x <- 13
is.numeric(x)
```

```
## [1] TRUE
```

```
x <- 3
is.integer(x)
```

```
## [1] FALSE
```

```
x <- 3L
is.integer(x)
```

```
## [1] TRUE
```

```
is.numeric(x)
```

```
## [1] TRUE
```

Character

R has 2 primary ways of handling character data: character and factor

```
# Store 'RStudio' to x as character data
x <- 'RStudio'
x
```

```
## [1] "RStudio"
```

```
# Store 'RStudio' to x as factor data by using factor function
x <- factor('RStudio')
x <- factor(c('a','c','D'))
x <- factor(c(1,2,3))
x
```

```
## [1] 1 2 3
## Levels: 1 2 3
```

```
# Length
x <- 'RStudio'
nchar(x)  ## nchar can only be used for character and numeric data
```

```
## [1] 7
```

```
y <- 365
nchar(y)
```

```
## [1] 3
```

Date

```
date <- as.Date('2022-01-17')
date
```

```
## [1] "2022-01-17"
```

```
class(date)
```

```
## [1] "Date"
```

```
as.numeric(date)
```

```
## [1] 19009
```

```
as.numeric(2022-01-17)
```

```
## [1] 2004
```

Data Wrangling

install.packages('stringr') # For string install.packages('lubridate') # For date install.packages('forcats') # For factor

Logical

```
x <- 13
is.numeric(x)
```

```
## [1] TRUE
```

```
x1 <- 'I am awesome'
is.numeric(x1)
```

```
## [1] FALSE
```

```
is.character(x1)
```

```
## [1] TRUE
```

```
is.na(x1)
```

```
## [1] FALSE
```

```
x <- TRUE
class(x)
```

```
## [1] "logical"
```

```
is.logical(x)
```

```
## [1] TRUE
```

```
TRUE * 5
```

```
## [1] 5
```

```
c(TRUE, TRUE, TRUE)*1
```

```
## [1] 1 1 1
```

```
FALSE * 5
```

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

```
# Comparison

1 == 1
```

```
## [1] TRUE
```

```
1 != 1
```

```
## [1] FALSE
```

```
1 < 1
```

```
## [1] FALSE
```

```
1 <= 2
```

```
## [1] TRUE
```

```
'i' == 'IE6600'
```

```
## [1] FALSE
```

```
'i' < 'IE6600'
```

```
## [1] TRUE
```

Vector

A vector is a collection of elements, all of the same type. A vector cannot be of mixed type.

Vectors don't have dimensions

Column or row vectors can be represented as one-dimensional matrices.

```
x <- c('IE6600', 'Data', 'Visulization')

class(x)
```

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

```
y <- c(2,3)

class(y)
```

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

```
x <- c(1,2,3,4,5)
x
```

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

```
y <- c(1:5)
y
```

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

```
x * 3
```

```
## [1]  3  6  9 12 15
```

```
y^2
```

```
## [1]  1  4  9 16 25
```

```
sqrt(x)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
length(x)
```

```
## [1] 5
```

```
length(y)
```

```
## [1] 5
```

```
x + y
```

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

```
x * y
```

```
## [1]  1  4  9 16 25
```

```
x / y
```

```
## [1] 1 1 1 1 1
```

```
x^y
```

```
## [1]    1    4   27  256 3125
```

```
x <- c(1:5)
x + c(1,2) ## 1+1 2+2 3+1 4+2 5+1
```

```
## Warning in x + c(1, 2): longer object length is not a multiple of shorter object
## length
```

```
## [1] 2 4 4 6 6
```

```
x <= 3
```

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

```
nchar(x)
```

```
## [1] 1 1 1 1 1
```

```
c('I', 'am', 'beautiful')
```

```
## [1] "I"         "am"        "beautiful"
```

```
c(One='I', Two='am', Three='beautiful')
```

```
##         One         Two       Three
##         "I"        "am" "beautiful"
```

```
x <- c(5:10)

x[2]
```

```
## [1] 6
```

```r
x[c(1,3)]
```

```
## [1] 5 7
```

```r
x[1:3]
```

```
## [1] 5 6 7
```

Factor

```r
x <- c('I','am','awesome')
x
```

```
## [1] "I"       "am"      "awesome"
```

```r
x <- as.factor(x)
x
```

```
## [1] I       am      awesome
## Levels: am awesome I
```

```r
x <- factor(x, levels=c('I','awesome','am'))
x
```

```
## [1] I       am      awesome
## Levels: I awesome am
```

Missing Data

R has 2 types of missing data: NA and NULL. Similar but behave differently

NA

NA will often be seen as just another element of a vector. is.na tests each element of a vector for missingness

```r
x <- c(1,2,3,NA,5)
x
```

```
## [1]  1  2  3 NA  5
```

```r
length(x)
```

```
## [1] 5
```

```
is.na(x)
```

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

NULL

NULL is nothingness. Functions can sometimes return NULL and their arguments can be NULL

NULL is often returned by expressions and functions whose value is undefined

NULL is atomical and cannot exist within a vector.

```
x <- c(1,2,3,NULL,5)

length(x)
```

```
## [1] 4
```

```
is.null(x)
```

```
## [1] FALSE
```

Call Functions

```
mean(c(1,2,3))
```

```
## [1] 2
```

```
sum(c(1,2,3))
```

```
## [1] 6
```

Pipe

```
library(magrittr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
y <- c(1,2,3)
mean(y)
```

```
## [1] 2
```

```
c(1,2,3) %>% mean
```

```
## [1] 2
```

Advanced Data Structures

The most common are data frame, matrix, and list.

Data Frames

R organized data.frame, each column is actually a vector, each of which has the same length. Within a column each element must be of the same type

```
x <- 1:5
y <- -1:3
a <- c('I','am','an','awesome','student')

xya.df <- data.frame(x,y,a)
xya.df
```

```
##   x  y       a
## 1 1 -1       I
## 2 2  0      am
## 3 3  1      an
## 4 4  2 awesome
## 5 5  3 student
```

```
xya.df <- data.frame(1:5, -1:3, a)
xya.df
```

```
##   X1.5 X.1.3       a
## 1    1    -1       I
## 2    2     0      am
## 3    3     1      an
## 4    4     2 awesome
## 5    5     3 student
```

```
xya.df <- data.frame(first = x, second = y, third = a)
xya.df
```

```
##   first second   third
## 1     1     -1       I
## 2     2      0      am
## 3     3      1      an
## 4     4      2 awesome
## 5     5      3 student
```

```
names(xya.df) <- c(x='first', y='second', a='third')
xya.df
```

```
##   first second    third
## 1     1     -1        I
## 2     2      0       am
## 3     3      1       an
## 4     4      2 awesome
## 5     5      3 student
```

```
nrow(xya.df)
```

```
## [1] 5
```

```
ncol(xya.df)
```

```
## [1] 3
```

```
dim(xya.df)
```

```
## [1] 5 3
```

```
names(xya.df)
```

```
## [1] "first"  "second" "third"
```

```
head(xya.df)
```

```
##   first second    third
## 1     1     -1        I
## 2     2      0       am
## 3     3      1       an
## 4     4      2 awesome
## 5     5      3 student
```

```
head(xya.df, n=3)
```

```
##   first second third
## 1     1     -1     I
## 2     2      0    am
## 3     3      1    an
```

```
xya.df$second
```

```
## [1] -1  0  1  2  3
```

```
xya.df$newColumn <- c(1:5)
xya.df
```

```
##   first second   third newColumn
## 1     1     -1       I         1
## 2     2      0      am         2
## 3     3      1      an         3
## 4     4      2 awesome         4
## 5     5      3 student         5
```

```
xya.df[5,2] # 5th row, 2th col
```

```
## [1] 3
```

```
xya.df[5,] # 5th row
```

```
##   first second   third newColumn
## 5     5      3 student         5
```

```
xya.df[c(1,2),3] # 1st & 2nd row, 3rd col
```

```
## [1] "I"  "am"
```

```
class(xya.df[, 'third'])
```

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

```
xya.df[, 'third', drop=FALSE]
```

```
##     third
## 1       I
## 2      am
## 3      an
## 4 awesome
## 5 student
```

```
class(xya.df[, 'third', drop=FALSE])
```

```
## [1] "data.frame"
```

```
xya.df[xya.df$second > 1, ]
```

```
##   first second   third newColumn
## 4     4      2 awesome         4
## 5     5      3 student         5
```

```
xya.df[xya.df$second > 1, 2:3]
```

```
##   second   third
## 4      2 awesome
## 5      3 student
```

List

A list can contain all numeric or characters or a mix of the 2 or data.frames or recursively other lists.

Lists are created with the list function where each argument to the function becomes an element of the list

```
list(1,2,3) # Create a 3 element list
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
```

```
list(c(1,2,3)) # Create a single element list
```

```
## [[1]]
## [1] 1 2 3
```

```
list(c(1,2,3), 1:4) # Create a 2 element list
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 1 2 3 4
```

```
b <- c('master','phD','undergraduate','others','master')
list(c(1:3),b)
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "master"        "phD"           "undergraduate" "others"
## [5] "master"
```

```
d <- list(c(1:3),b)

names(d) <- c('number','degree')
d
```

```
## $number
## [1] 1 2 3
##
## $degree
## [1] "master"        "phD"           "undergraduate" "others"
## [5] "master"
```

```
list1 <- list(number=1:3, degree=b)
list1
```

```
## $number
## [1] 1 2 3
##
## $degree
## [1] "master"        "phD"           "undergraduate" "others"
## [5] "master"
```

```
list1 <- list(number=1:3,degree=b)

list1[[1]]
```

```
## [1] 1 2 3
```

```
list1[1]
```

```
## $number
## [1] 1 2 3
```

```
d <- list(number = 1:3, degree = b)
length(d)
```

```
## [1] 2
```

```
d[[3]] <- c(7,7,7,7)
d[['student']] <- c('John','Peter','Tome','Jerry')
d
```

```
## $number
## [1] 1 2 3
##
## $degree
## [1] "master"        "phD"           "undergraduate" "others"
## [5] "master"
##
## [[3]]
## [1] 7 7 7 7
##
## $student
## [1] "John"  "Peter" "Tome"  "Jerry"
```

Matrix

```
A <- matrix(1:6, nrow = 3)
A
```

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

```
nrow(A)
```

```
## [1] 3
```

```
ncol(A)
```

```
## [1] 2
```

```
B <- matrix(2:7, nrow = 3)
B
```

```
##      [,1] [,2]
## [1,]    2    5
## [2,]    3    6
## [3,]    4    7
```

```
nrow(B)
```

```
## [1] 3
```

```
ncol(B)
```

```
## [1] 2
```

```
A == B
```

```
##       [,1]  [,2]
## [1,] FALSE FALSE
## [2,] FALSE FALSE
## [3,] FALSE FALSE
```

```
A * B
```

```
##      [,1] [,2]
## [1,]    2   20
## [2,]    6   30
## [3,]   12   42
```

Data Input to R

```r
readurl <- 'http://www.jaredlander.com/data/Tomato%20First.csv'
tomato <- read.csv(readurl)

tomato
```

```
##    Round                   Tomato Price           Source Sweet Acid Color
## 1      1               Simpson SM  3.99      Whole Foods   2.8  2.8   3.7
## 2      1        Tuttorosso (blue)  2.99          Pioneer   3.3  2.8   3.4
## 3      1       Tuttorosso (green)  0.99          Pioneer   2.8  2.6   3.3
## 4      1           La Fede SM DOP  3.99        Shop Rite   2.6  2.8   3.0
## 5      2             Cento SM DOP  5.49       D Agostino   3.3  3.1   2.9
## 6      2            Cento Organic  4.99       D Agostino   3.2  2.9   2.9
## 7      2              La Valle SM  3.99        Shop Rite   2.6  2.8   3.6
## 8      2          La Valle SM DOP  3.99           Faicos   2.1  2.7   3.1
## 9      3   Stanislaus Alta Cucina  4.53 Restaurant Depot   3.4  3.3   4.1
## 10     3                     Ciao    NA            Other   2.6  2.9   3.4
## 11     3       Scotts Backyard SM  0.00       Home Grown   1.6  2.9   3.1
## 12     3 Di Casa Barone (organic) 12.80           Eataly   1.7  3.6   3.8
## 13     4         Trader Joes Plum  1.49      Trader Joes   3.4  3.3   4.0
## 14     4          365 Whole Foods  1.49      Whole Foods   2.8  2.7   3.4
## 15     4        Muir Glen Organic  3.19      Whole Foods   2.9  2.8   2.7
## 16     4        Bionature Organic  3.39      Whole Foods   2.4  3.3   3.4
##    Texture Overall Avg.of.Totals Total.of.Avg
## 1      3.4     3.4          16.1         16.1
## 2      3.0     2.9          15.3         15.3
## 3      2.8     2.9          14.3         14.3
## 4      2.3     2.8          13.4         13.4
## 5      2.8     3.1          14.4         15.2
## 6      3.1     2.9          15.5         15.1
## 7      3.4     2.6          14.7         14.9
## 8      2.4     2.2          12.6         12.5
## 9      3.2     3.7          17.8         17.7
## 10     3.3     2.9          15.3         15.2
## 11     2.4     1.9          11.9         11.9
## 12     2.3     1.4          12.7         12.7
## 13     3.6     3.9          17.8         18.2
## 14     3.1     3.1          14.8         15.2
## 15     3.2     3.1          14.8         14.7
## 16     3.2     2.8          15.1         15.2
```

CSV data

library(tidyverse)

library(readr)

tomato <- read_csv('tomato.csv')

Excel data

install.packages('readxl')

library(readxl)

tomato <- read_excel(‘excel location’)

Other types of data

read.spss => SPSS

read.dta => Stata

read.ssd => SAS

read.octave => Octave

read.mtp => Minitab

read.systat => Systat