Lesson 1: Intro to R and Python Programming

Console Programming and Basic Scripts

Suresh Paul, C.Phil, M.S., B.Engg

2024-10-21

Console R Programming

arithmetic operations (use R like a calculator)

```
219 + 774 #addition
```

```
## [1] 993
```

```
912 - 1004 #subtraction
```

```
## [1] -92
```

```
23 * 14 # multiplication
```

```
## [1] 322
```

```
21 / 4 # division
```

```
## [1] 5.25
```

```
2 ^ 6 # power
```

```
## [1] 64
```

```
(2187 + 1144) * (12 - 4) # combination
```

```
## [1] 26648
```

```
log(60) # example of a function: log
```

```
## [1] 4.094345
```

```r
exp(3) # example of a function: exponent
```

```
## [1] 20.08554
```

```r
abs(-364) # example of a function: absolute
```

```
## [1] 364
```

```r
350 %% 17 # Mod, remainder after division of 2 numbers.
```

```
## [1] 10
```

variable assignments

```r
# variable assignment

addition1 <- 219 + 774 ;

# print output to screen

print(addition1)
```

```
## [1] 993
```

```r
print(paste("Sum =", addition1))
```

```
## [1] "Sum = 993"
```

```r
# variable assignment

total_weight <- 45
total_area <- 10

density_measure <- total_weight / total_area

# print output to screen
density_measure
```

```
## [1] 4.5
```

remember, variable assignments are case-sensitive

```r
amsterdam <- 100

Amsterdam <- 250

AMSTERDAM <- 700

amsterdam + Amsterdam - AMSTERDAM
```

```
## [1] -350
```

know your variable types

numeric

```
# numeric

var1 <- 117.2; var1
```

```
## [1] 117.2
```

```
class(var1)
```

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

```
typeof(var1)
```

```
## [1] "double"
```

```
length(var1)
```

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

```
# print and paste (also, paste0)

print(
  paste(
    "The variable", var1, "is a", class(var1),
    "variable with a type", typeof(var1),
    "and a length of", length(var1)
    )
  )
```

```
## [1] "The variable 117.2 is a numeric variable with a type double and a length of 1"
```

numeric: integer

```
# numeric: integer

var2 <- 77L; var2
```

```
## [1] 77
```

```
class(var2)
```

```
## [1] "integer"
```

```r
typeof(var2)
```

```
## [1] "integer"
```

```r
length(var2)
```

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

character

```r
# character

var3 <- "A"; var3
```

```
## [1] "A"
```

```r
class(var3)
```

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

```r
typeof(var3)
```

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

```r
length(var3)
```

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

string

```r
# character

var4 <- "BUS 622: Development Tooks in Business Analytics"; var4
```

```
## [1] "BUS 622: Development Tooks in Business Analytics"
```

```r
class(var4)
```

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

```r
typeof(var4)
```

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

```
length(var4)
```

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

boolean

```
# boolean

var5 <- TRUE; var5
```

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

```
class(var5)
```

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

```
typeof(var5)
```

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

```
length(var5)
```

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

```
# boolean abbreviations

var5a <- T; var5a
```

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

```
class(var5a)
```

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

```
typeof(var5a)
```

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

```
length(var5a)
```

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

vectors

series of numbers: numeric vector

```r
# vector

var6 <- c(1, 2, 3, 5, 8, 13, 21, 34, 55); var6
```

```
## [1]  1  2  3  5  8 13 21 34 55
```

```r
class(var6)
```

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

```r
typeof(var6)
```

```
## [1] "double"
```

```r
length(var6)
```

```
## [1] 9
```

series of strings: character vector

```r
# vector

var7 <- c(
  "BUS 622: Development Tooks in Business Analytics",
  "R Programming",
  "Python Programming",
  "MBA",
  "Baldwin Wallace University"
  ); var7
```

```
## [1] "BUS 622: Development Tooks in Business Analytics"
## [2] "R Programming"
## [3] "Python Programming"
## [4] "MBA"
## [5] "Baldwin Wallace University"
```

```r
class(var7)
```

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

```r
typeof(var7)
```

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

```r
length(var7)
```

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

series of boolean values: boolean vector

```r
# vector

var8 <- c(TRUE, FALSE, TRUE, FALSE, TRUE); var8
```

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

```r
class(var8)
```

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

```r
typeof(var8)
```

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

```r
length(var8)
```

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

more on vectors

```r

# list first 100 numbers - ascending
numList1 <- c(1:100); numList1
```

vector operations

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

```r
# perform operations on the vector
numList1 + 100
```

```
##   [1] 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
##  [19] 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
##  [37] 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154
##  [55] 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172
##  [73] 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190
##  [91] 191 192 193 194 195 196 197 198 199 200
```

```r
numList1 > 50
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [97]  TRUE  TRUE  TRUE  TRUE
```

```r
# list first 100 numbers - descending
numList2 <- c(100:1); numList2
```

```
##   [1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83
##  [19]  82  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66  65
##  [37]  64  63  62  61  60  59  58  57  56  55  54  53  52  51  50  49  48  47
##  [55]  46  45  44  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29
##  [73]  28  27  26  25  24  23  22  21  20  19  18  17  16  15  14  13  12  11
##  [91]  10   9   8   7   6   5   4   3   2   1
```

```r
# element-wise addition / multiplication etc.

vec1 <- c(1:10); vec1
```

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

```r
vec2 <- c(10:1); vec2
```

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

```r
vec1 + vec2 # addition
```

```
##  [1] 11 11 11 11 11 11 11 11 11 11
```

```r
vec1 * vec2 # multiplication
```

```
##  [1] 10 18 24 28 30 30 28 24 18 10
```

```r
vec2 / vec1 # division
```

```
##  [1] 10.0000000  4.5000000  2.6666667  1.7500000  1.2000000  0.8333333
##  [7]  0.5714286  0.3750000  0.2222222  0.1000000
```

```r
vec1 ^ vec2 # power
```

```
##  [1]     1   512  6561 16384 15625  7776  2401   512    81    10
```

```
# vector of letters
letterList1 <- LETTERS ; letterList1
```

indexing

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
## [20] "T" "U" "V" "W" "X" "Y" "Z"
```

```
letterList2 <- LETTERS[1:5] ; letterList2
```

```
## [1] "A" "B" "C" "D" "E"
```

```
# indexing
homework_scores <- c(28, 22, 21, 32, 13, 22, 40, 12, 17, 22, 18, 26, 38, 40, 40, 39, 37)

project_scores <- c(33, 28, 19, 29, 25, 17, 16, 15, 24, 12, 26, 27, 37, 40, 35, 23, 29)
```

```
# What is first element of the Homework vector?
homework_scores[1]
```

```
## [1] 28
```

```
# What is last element of the Project Writen Report vector?
last_index <- length(project_scores)
project_scores[last_index]
```

```
## [1] 29
```

```
# Everything except for the first element of the Project Presentation vector?
project_scores[-1]
```

```
##  [1] 28 19 29 25 17 16 15 24 12 26 27 37 40 35 23 29
```

```
# Second and Third Homework Scores...
homework_scores[2:3]
```

```
## [1] 22 21
```

```
# Assign student names to te above score vectors #1
names(homework_scores) <- c("Alan Beaney", "Luke Driscoll", "Jesse Franklin", "Roula Gemelas", "John Hal

names(project_scores) <- c("Alan Beaney", "Luke Driscoll", "Jesse Franklin", "Roula Gemelas", "John Hab
```

```
# What is Oliver Kelleher's Homework Score?
homework_scores["Oliver Kelleher"]
```

```
## Oliver Kelleher
##              12
```

```r
# What is Zach Ottenweller's Total Score?
homework_scores["Zach Ottenweller"] + project_scores["Zach Ottenweller"]
```

```
## Zach Ottenweller
##               53
```

```r
# What is Alan Beaney's, Juliet Penrod's and Marley Wilder's Project scores?
project_scores[c("Alan Beaney", "Juliet Penrod", "Marley Wilder")]
```

```
##   Alan Beaney Juliet Penrod Marley Wilder
##            33            40            29
```

```r
# Who earned full Homework score?
full_hw_score <- homework_scores == 40
homework_scores[full_hw_score]
```

```
##  Seth Johnson Juliet Penrod  Kade Swisher
##            40            40            40
```

```r
# Assign student names to te above score vectors #2
full_names <- c("Alan Beaney", "Luke Driscoll", "Jesse Franklin", "Roula Gemelas", "John Haberman", "Mi

names(homework_scores) <- full_names

names(project_scores) <- full_names
```

```r
# What is Project score for John Haberman?
project_scores["John Haberman"]
```

```
## John Haberman
##            25
```

```r
# What is total Project score for the class?
sum(project_scores)
```

```
## [1] 435
```

```r
# ... same as
project_scores[1] + project_scores[2] + project_scores[3] + project_scores[4] + project_scores[5] + pro
```

```
## Alan Beaney
##         308
```

```r
# ... same as
project_scores["Alan Beaney"] + project_scores["Luke Driscoll"] + project_scores["Jesse Franklin"] + pr
```

```
## Alan Beaney
##         435
```

```r
# What is the mean, min, and max Homework Score for the class?
mean(homework_scores)
```

```
## [1] 27.47059
```

```r
min(homework_scores)
```

```
## [1] 12
```

```r
max(homework_scores)
```

```
## [1] 40
```

```r
sd(homework_scores)
```

```
## [1] 10.0257
```

```r
# Did the class score in the Homework or in the Project?
sum(homework_scores) > sum(project_scores)
```

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

```r
# Did the class score in the Homework or in the Project?
# assign and fancy it
check <- sum(homework_scores) > sum(project_scores)

ifelse(
  check == TRUE,
  "Class scored more in the Homeworks than in the Project.",
  "Class scored more in the Project than in the Homeworks."
  )
```

```
## [1] "Class scored more in the Homeworks than in the Project."
```

relational and logical operators:

$>$, $<$, $==$, $<=$, $>=$, &, |, $!=$

```r
num1 <- 100
num2 <- 200
num3 <- 100
```

```r
# greater than
num1 > num2
```

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

```r
# less than
num1 < num2
```

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

```r
# greater than or equal to
num1 >= num3
```

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

```r
# less than or equal to
num3 <= num2
```

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

```r
# equal to
num1 == num3
```

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

```r
# not equal to
num1 != num2 # (or) !(num1 == num2)
```

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

```r
# AND operator
num1 < num2 & num1 <= num3
```

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

```r
# OR operator
num1 == num2 | num1 == num3
```

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

```r
# compound operations
num1 + num3 == num2
```

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

factor operators

```r
# educational classification
edu_class <- c("HS", "UG", "MBA", "HS", "MBA", "UG",
               "HS", "HS", "UG", "MA", "MBA", "HS",
               "UG", "MA", "UG", "MA", "MA", "MBA")
```

```r
# convert edu_class to a factor
factor_edu_class <- factor(edu_class)
factor_edu_class
```

```
##  [1] HS  UG  MBA HS  MBA UG  HS  HS  UG  MA  MBA HS  UG  MA  UG  MA  MA  MBA
## Levels: HS MA MBA UG
```

```r
# convert edu_class to a factor with levels
factor_edu_class1 <- factor(edu_class, order = TRUE, levels = c("HS", "UG", "MA", "MBA"))
factor_edu_class1
```

```
##  [1] HS  UG  MBA HS  MBA UG  HS  HS  UG  MA  MBA HS  UG  MA  UG  MA  MA  MBA
## Levels: HS < UG < MA < MBA
```

```r
# summary
summary(factor_edu_class1)
```

```
##  HS  UG  MA MBA
##   5   5   4   4
```

```r
# is someone more qualified than another?
factor_edu_class1[5]
```

```
## [1] MBA
## Levels: HS < UG < MA < MBA
```

```r
factor_edu_class1[9]
```

```
## [1] UG
## Levels: HS < UG < MA < MBA
```

```r
factor_edu_class1[5] > factor_edu_class1[9]
```

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

simple computations

e.g., Celsius to Fahrenheit conversion

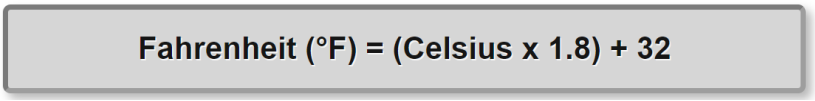

Figure 1: Celsius to Fahrenheit formula

```r
# Celsius to Fahrenheit conversion

tempCelsius <- 40

tempFahrenheit <- (tempCelsius * 1.8) + 32

print(tempFahrenheit)
```

```
## [1] 104
```

e.g., Fahrenheit to Celsius conversion

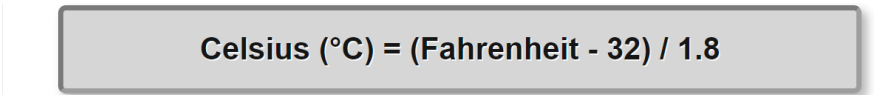

Figure 2: Fahrenheit to Celsius formula

```r
# Fahrenheit to Celsius conversion

tempFahrenheit <- c(55:70)

tempCelsius <- (tempFahrenheit - 32) * 5/9

print(tempCelsius)
```

```
##  [1] 12.77778 13.33333 13.88889 14.44444 15.00000 15.55556 16.11111 16.66667
##  [9] 17.22222 17.77778 18.33333 18.88889 19.44444 20.00000 20.55556 21.11111
```

Rounding

```r
# rounding to the nearest 2 decimal places

round(tempCelsius, digits = 2)
```

```
##  [1] 12.78 13.33 13.89 14.44 15.00 15.56 16.11 16.67 17.22 17.78 18.33 18.89
## [13] 19.44 20.00 20.56 21.11
```

Floor

```r
# rounding to the lowest integer

floor(tempCelsius)
```

```
##  [1] 12 13 13 14 15 15 16 16 17 17 18 18 19 20 20 21
```

Ceiling

```
# rounding to the highest integer

ceiling(tempCelsius)
```

```
##  [1] 13 14 14 15 15 16 17 17 18 18 19 19 20 20 21 22
```

e.g., compute Future Value

$$\text{Future Value (FV)} = PV \times (1 + r)^n$$

- PV = Present Value
- r = Interest Rate (%)
- n = Number of Compounding Periods

Figure 3: Future Value formula

```
int <- 0.08 # 8% annual rate

nterms <- 7 # number of years

amount <- 1000 # present value
```

```
# compute future value
fv <- amount * ((1 + int)^nterms); fv
```

```
## [1] 1713.824
```

```
# modular calculation

fv1 <- (1 + int)^nterms;

fv1 <- fv1 * amount; fv1
```

```
## [1] 1713.824
```

e.g., compute Future Value for a vector of interest rates

```
int1 <- c(0.06, 0.08, 0.10, 0.12); int1
```

```
## [1] 0.06 0.08 0.10 0.12
```

```r
# compute future value
fv2 <- amount * ((1 + int1)^nterms); fv2
```

```
## [1] 1503.630 1713.824 1948.717 2210.681
```
