Assignment 1

Nina Dyatchenko

September 3d, 2018

## Exercise set 1

1. Create an object called “myAge” that holds your age *in months*. Hint: the multiplication operator in R is “\*“.

myAge <- 29 \* 12 + 8  
myAge

## [1] 356

1. Using the object “myAge”, approximate the number of days you’ve been alive. Use 30.44 days per month. Check this against your age in days on Wolfram Alpha. How many days off was your estimate?

myAge \* 30.44

## [1] 10836.64

# from Wolfram Alpha, I am 10827 days old today.

1. Using the object “myAge”, find your decimal age. Hint: the division operator is “/”.

myAge / 12

## [1] 29.66667

1. Copy and Paste MyAge into the console and press ENTER. What happens? Why?

Console returened error “Error: object ‘MyAge’ not found”. Because R is case sencitive, it sees MyAge and myAge differently. myAge was defined in the code previously, but MyAge was not. Thus R gives the error.

## Exercise set 2

1. Five people start the newest diet craze. Their weights (in kg) before and after were [78 72 78 79 105] and [67 65 79 70 93], respectively. Store these two vectors as “wtBefore” and “wtAfter”, respectively. Find the total weight lost, the average weight lost, and the weight lost per person.

wtBefore <- c(78, 72, 78, 79, 105)  
wtAfter <- c(67, 65, 79, 70, 93)  
  
# Total weight lost:  
sum(wtBefore - wtAfter)

## [1] 38

# Average weight lost:  
mean(wtBefore - wtAfter)

## [1] 7.6

# Weight lost per person:  
wtBefore - wtAfter

## [1] 11 7 -1 9 12

1. Create the following vectors using the rep() and seq() functions:
2. start with the first non-negative integer powers of 2

2^(seq(from=0, to=8, by=1))

## [1] 1 2 4 8 16 32 64 128 256

2^(0:8)

## [1] 1 2 4 8 16 32 64 128 256

1. 1, 1.5, 2, 2.5, …, 12

seq( from =1, to = 12 , by = 0.5)

## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5  
## [15] 8.0 8.5 9.0 9.5 10.0 10.5 11.0 11.5 12.0

1. 1, 8, 27, 64, …, 1000

seq(from = 1, to = 10, by = 1)^3

## [1] 1 8 27 64 125 216 343 512 729 1000

1. 1, -1/2, 1/3, -1/4, …, -1/100

library(MASS)  
as.fractions(-(seq(from = -1, to = 1, by = 2)/seq(from = 1, to = 100, by = 1)))

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

# or  
  
-(seq(from = -1, to = 1, by = 2)/seq(from = 1, to = 100, by = 1))

## [1] 1.00000000 -0.50000000 0.33333333 -0.25000000 0.20000000  
## [6] -0.16666667 0.14285714 -0.12500000 0.11111111 -0.10000000  
## [11] 0.09090909 -0.08333333 0.07692308 -0.07142857 0.06666667  
## [16] -0.06250000 0.05882353 -0.05555556 0.05263158 -0.05000000  
## [21] 0.04761905 -0.04545455 0.04347826 -0.04166667 0.04000000  
## [26] -0.03846154 0.03703704 -0.03571429 0.03448276 -0.03333333  
## [31] 0.03225806 -0.03125000 0.03030303 -0.02941176 0.02857143  
## [36] -0.02777778 0.02702703 -0.02631579 0.02564103 -0.02500000  
## [41] 0.02439024 -0.02380952 0.02325581 -0.02272727 0.02222222  
## [46] -0.02173913 0.02127660 -0.02083333 0.02040816 -0.02000000  
## [51] 0.01960784 -0.01923077 0.01886792 -0.01851852 0.01818182  
## [56] -0.01785714 0.01754386 -0.01724138 0.01694915 -0.01666667  
## [61] 0.01639344 -0.01612903 0.01587302 -0.01562500 0.01538462  
## [66] -0.01515152 0.01492537 -0.01470588 0.01449275 -0.01428571  
## [71] 0.01408451 -0.01388889 0.01369863 -0.01351351 0.01333333  
## [76] -0.01315789 0.01298701 -0.01282051 0.01265823 -0.01250000  
## [81] 0.01234568 -0.01219512 0.01204819 -0.01190476 0.01176471  
## [86] -0.01162791 0.01149425 -0.01136364 0.01123596 -0.01111111  
## [91] 0.01098901 -0.01086957 0.01075269 -0.01063830 0.01052632  
## [96] -0.01041667 0.01030928 -0.01020408 0.01010101 -0.01000000

1. 1, 0, 3, 0, 5, 0, 7, 0, …, 0, 49

rep(c(1, 0))\*(c(1:49, by=2))

##   
## 1 0 3 0 5 0 7 0 9 0 11 0 13 0 15 0 17 0 19 0 21 0 23 0 25   
## by   
## 0 27 0 29 0 31 0 33 0 35 0 37 0 39 0 41 0 43 0 45 0 47 0 49 0

1. 1, 3, 6, 10, 15, …, 210 (Hint: ?cumsum)

cumsum(1:20)

## [1] 1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 153  
## [18] 171 190 210

1. 1, 2, 2, 3, 3, 3, …, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10 (Hint: ?rep)

x=1:10  
rep(x, times = c(1:10), length.out = NA, each = 1)

## [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 7 7  
## [24] 7 7 7 7 7 8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 10  
## [47] 10 10 10 10 10 10 10 10 10

1. (ADVANCED) The jth term of the Taylor-MacLaurin series for the natural logarithm is
2. Find the residuals of this approximation for n = 5, 10, 50, 100 around x = 0.25

x <- 0.25  
n <- c(5, 10, 50, 100)  
  
# n = 5  
est\_n5 <- x ^ (1:n[1]) / (1:n[1])  
-sum(est\_n5) # estimate at n = 5

## [1] -0.2876302

log(1-x) # truth

## [1] -0.2876821

log(1-x) - (-sum(est\_n5)) #residual

## [1] -5.186412e-05

# n = 10  
est\_n10 <- x ^ (1:n[2]) / (1:n[2])  
-sum(est\_n10) # estimate at n = 10

## [1] -0.287682

log(1-x) # truth

## [1] -0.2876821

log(1-x) - (-sum(est\_n10)) #residual

## [1] -2.813493e-08

# n = 50  
est\_n50 <- x ^ (1:n[3]) / (1:n[3])  
-sum(est\_n50) # estimate at n = 50

## [1] -0.2876821

log(1-x) # truth

## [1] -0.2876821

log(1-x) - (-sum(est\_n50)) #residual

## [1] 0

# n = 100  
est\_n100 <- x ^ (1:n[4]) / (1:n[4])  
-sum(est\_n100) # estimate at n = 100

## [1] -0.2876821

log(1-x) # truth

## [1] -0.2876821

log(1-x) - (-sum(est\_n100)) #residual

## [1] 0

## Exercise set 3

1. Create a vector of the months of the year and store it in an object called “mons”.

mons<- month.name  
mons

## [1] "January" "February" "March" "April" "May"   
## [6] "June" "July" "August" "September" "October"   
## [11] "November" "December"

1. You would like to discard the first three months, so you type “mons[-1:3]”. What happened? Why should you type instead?

# Error in mons[-1:3] : only 0's may be mixed with negative subscripts. R doesn't allow to #"knit" with the error  
  
mons[1:3]

## [1] "January" "February" "March"

mons[4:12]

## [1] "April" "May" "June" "July" "August" "September"  
## [7] "October" "November" "December"

# or  
mons[-(1:3)]

## [1] "April" "May" "June" "July" "August" "September"  
## [7] "October" "November" "December"

1. Find a method to use the “length()” function to find the last four entries of *any* atomic vector. Apply this to find the last four months of the year.

mons[length(mons)-(3:0)]

## [1] "September" "October" "November" "December"

1. Logically subset the uppercase vector of English letters in the following ways:
2. Type LETTERS and press ENTER. What is your guess to find the lower case letters?

LETTERS

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

letters

## [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"  
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"

1. Create a vector of the first 12 upper-case English letters

LETT<- c(LETTERS[1:12])  
LETT

## [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L"

1. Create a vector of the 1st, 3rd, 5th, 7th, …, 25th English letters

LETTnum<- c(LETTERS[seq(from = 1, to = 26, by = 2)])  
LETTnum

## [1] "A" "C" "E" "G" "I" "K" "M" "O" "Q" "S" "U" "W" "Y"

1. Create a vector of the consonants. (Hint: it may be easier to find the letters that are *not* one of the vowels)

vowels <- c("A", "E", "I", "O", "U")  
"A" %in% vowels # test the "in" operator

## [1] TRUE

areVowels <- LETTERS %in% vowels #will give TRUE for vowles and FALSE for consonants.  
areVowels

## [1] TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE  
## [12] FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [23] FALSE FALSE FALSE FALSE

areConsonants <- !(LETTERS %in% vowels) # will give opposite of above.  
areConsonants

## [1] FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE  
## [12] TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE  
## [23] TRUE TRUE TRUE TRUE

LETTERS[!(LETTERS %in% vowels)]

## [1] "B" "C" "D" "F" "G" "H" "J" "K" "L" "M" "N" "P" "Q" "R" "S" "T" "V"  
## [18] "W" "X" "Y" "Z"

LETTERS[areConsonants]

## [1] "B" "C" "D" "F" "G" "H" "J" "K" "L" "M" "N" "P" "Q" "R" "S" "T" "V"  
## [18] "W" "X" "Y" "Z"