RWorksheet_Pineda

Epiphany Louise O. Pineda

2025-10-13

```
#(1.)
#Set up a vector named age, consisting of 34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 22, 37, 3
age <- c(34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 22, 37,
         34, 19, 20, 57, 49, 50, 37, 46, 25, 17, 37, 42, 53, 41, 51,
         35, 24, 33, 41)
#How many data points?
length(age)
#Output 1[34]

#(2.)
#Find the reciprocal of the values for age.
1 / age
#Output [1] 0.02941176 0.03571429 0.04545455 0.02777778 0.03703704 0.05555556
#Output [7] 0.01923077 0.02564103 0.02380952 0.03448276 0.02857143 0.03225806
#Output [13] 0.03703704 0.04545455 0.02702703 0.02941176 0.05263158 0.05000000
#Output [19] 0.01754386 0.02040816 0.02000000 0.02702703 0.02173913 0.04000000
#Output [25] 0.05882353 0.02702703 0.02380952 0.01886792 0.02439024 0.01960784
#Output [31] 0.02857143 0.04166667 0.03030303 0.02439024

#(3.)
#Assign also new_age <- c(age, 0, age).
new_age <- c(age, 0, age)
new_age
#What happen to the new_age?
# - A zero (0) is inserted in the middle of the original age vector.
# - The new vector has 35 + 34 = 69 elements.

#(4.)
#Sort the values for age.
sort(age)
#Output [1] 17 18 19 20 22 22 24 25 27 27 28 29 31 33 34 34 35 35 36 37 37 37
#Output [23] 39 41 41 42 42 46 49 50 51 52 53 57

#(5.)
#Find the minimum and maximum value for age.
min(age)
max(age)
#Output [1] 17
#Output [1] 57

#(6.)
#Set up a vector named data, consisting of 2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, and 2
```

```r
data <- c(2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, 2.7)
#a. How many data points?
length(data)
#Output [1] 12

#(7.)
#Generates a new vector for data where you double every value of the data.
data2 <- data * 2
data2
#What happen to the data?
# -Every value is multiplied by 2, creating a new vector.

#(8.)
#Generate a sequence for the following scenario:
#8.1 Integers from 1 to 100.
#8.2 Numbers from 20 to 60
#8.3 Mean of numbers from 20 to 60
#8.4 Sum of numbers from 51 to 91
#8.5 Integers from 1 to 1,000

#a. Number of data points from 8.1 to 8.4
#8.1 (1:100)    Data points: 100
#8.2 (20:60)    Data points: 41
#8.3 (mean) Data Points: 1
#8.4 (sum)  Data Points: 1
#Total data points = 100 + 41 + 1 + 1 = 143

#b. Write the R code and its output from 8.1 to 8.4.
seq1 <- 1:100 #8.1
seq2 <- 20:60 #8.2
mean(seq2) #8.3
sum(51:91) #8.4
#Output # 8.1 Integers 1 to 100
#[1] 1 2 3 ... 100
# 8.2 Numbers 20 to 60
#[1] 20 21 22 ... 60
# 8.3 Mean of numbers from 20 to 60
#[1] 40
# 8.4 Sum of numbers from 51 to 91
#[1] 2911

#c. For 8.5 find only maximum data points until 10.
seq5 <- 1:1000
max(seq5[1:10])
#Output [1] 10

#(9.)
#*Print a vector with the integers between 1 and 100 that are not divisible by 3, 5 and 7 using filter
result <- Filter(function(i) { all(i %% c(3,5,7) != 0) }, 1:100)
result
#Output [1] 1 2 4 8 11 13 16 17 19 22 23 26 29 31 32 34 37 38 41 43 44 46 47 52 53 56 58 59 61 64 67 68

#(10.)
```

```r
#Generate a sequence backwards of the integers from 1 to 100.
rev(1:100)
#Output [1] 100  99  98 ... 3  2  1

#(11.)
#List all the natural numbers below 25 that are multiples of 3 or 5. Find the sum of these multiples.
nums <- 1:24
multiples <- nums[nums %% 3 == 0 | nums %% 5 == 0]
multiples
sum_multiples <- sum(multiples)
sum_multiples

#a. How many data points from 10 to 11?
#10 Data points: 100
#11 Data Points: 11

#b. Write the R code and its output from 10 and 11.
# Step 10
#[1] 100 99 98 ... 3 2 1
# Step 11
#[1] 3 5 6 9 10 12 15 18 20 21 24
#[1] 143

#(12.)

#(13.)
#Find x[2] and x[3]. Write the R code and its output.
score <- c(72, 86, 92, 63, 88, 89, 91, 92, 75, 75, 77)
score[2]
score[3]
#Output [1] 86
#Output [1] 92

#(14.)
#Create a vector a = c(1,2,NA,4,NA,6,7).
a <- c(1, 2, NA, 4, NA, 6, 7)

#a. Change the NA to 999 using the codes print(a,na.print="-999").
print(a, na.print = "-999")
#Output [1]    1    2 -999    4 -999    6    7
# - NA represents missing values in R.
# - Using print(a, na.print = "-999") does not change the vector, it only displays NAs as -999 when pri
# - The actual values in a remain NA.

#(15.)
#Create a vector x = (2,3,4). Check for the class(x). What is the class type?
x <- c(2, 3, 4)
class(x)
# Class type: "numeric"

#Change the class into foo. What will now be the class type?
class(x) <- "foo"
class(x)
```

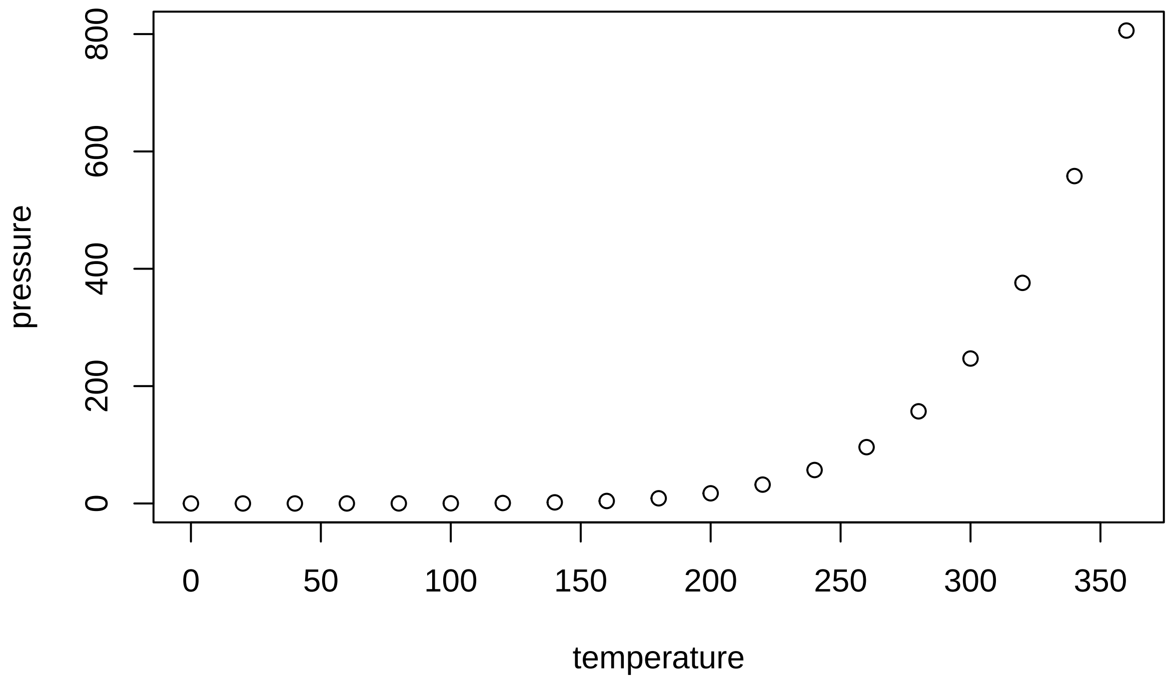
```
# Class type: "foo"

#Output: [1] "numeric"   # original class
#Output: [1] "foo"       # class after change
```

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.