RWorksheet_Corvera#1

2024-09-17

#1.Set up a vector named age age <- c(34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 37, 34, 19, 20, 57, 49, 50, 37, 46, 25, 17, 37, 42, 53, 41, 51, 35, 24, 33, 41) age

#2.Find the reciprocal of the values for age. reciprocal_age <- 1/age reciprocal_age

#3.Assign also new_age <- c(age, 0, age). new_age <- c(age, 0, age) new_age

#4.Sort the values for age. sorted_age <- sort(age) sorted_age

#5.Find the minimum and maximum value for age. min_age <- min(age) max_age <- max(age) min_age max_age

#6.Set up a vector named data 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) data

#7.Generates a new vector for data where you double every value of the data. new_data <- data * 2 new_data

#8.Generate a sequence for the following scenario #8.1Integers from 1 to 100 seq_1_to_100 <- seq(1, 100) seq_1_to_100

#8.2Numbers from 20 to 60 seq_20_to_60 <- seq(20, 60, 1) seq_20_to_60

#8.3Mean of numbers from 20 to 60 mean_20_to_60 <- mean(seq_20_to_60) mean_20_to_60

#8.4Sum of numbers from 51 to 91 sum_51_to_91 <- sum(seq(51, 91)) sum_51_to_91

#8.5Integers from 1 to 1,000 seq_1_to_1000 <- seq(1,1000) seq_1_to_1000

#c.For 8.5 find only maximum data points until 10. seq_1_to_1000 <- seq(1,1000) seq_1_to_10 <- seq_1_to_1000[1:10] seq_1_to_10

#9.*Print a vector with the integers between 1 and 100 that are not divisible by 3, 5 and 7 using filter option. filter_num <- Filter(function(i) { all(i %% c(3,5,7) != 0) }, seq(100)) filter_num

#10.Generate a sequence backwards of the integers from 1 to 100. seq100_to_1 <- seq(100, 1, by = -1) seq100_to_1

#11.List all the natural numbers below 25 that are multiples of 3 or 5. multiples <- Filter(function(i) { i %% 3 == 0 | i %% 5 == 0 }, seq(1, 24)) multiples

#Find the sum of these multiples. sum_multiples <- sum(sort_multiples) sum_multiples

#12.Statements can be grouped together using braces '{' and '}'. A group of statements is sometimes called a block. Single statements are evaluated when a new line is typed at the end of the syntactically complete statement. Blocks are not evaluated until a new line is entered after the closing brac. x <- {0 + x + 5 + }

#13.Set up a vector named score, consisting of 72, 86, 92, 63, 88, 89, 91, 92, 75, 75 and 77. score <- c(72, 86, 92, 63, 88, 89, 91, 92, 75, 75 77) score

#14.To access individual elements of an atomic vector, one generally uses the x[i] construction. Find x[2] and x[3]. x2 <- score[2] x3 <- score[3]

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

#Change the NA to 999 using the codes print(a,na.print="-999"). print(a,na.print="-999")

#15.A special type of function calls can appear on the left hand side of the assignment operator as in > class(x) <- “foo” name = readline(prompt=“Input your name:”) age = readline(prompt=“Input your age:”) print(paste(“My name is”,name, “and I am”,age ,“years old.”)) print(R.version.string)