HW02\_MoreRBasics

Nolan Bentley

9/4/2021

Table of Contents

[Instructions 1](#_Toc94626873)

[Object indexing and extraction 1](#_Toc94626874)

[Q1 2](#_Toc94626875)

[Q2-Q5 2](#_Toc94626876)

[Q6 3](#_Toc94626877)

[Q7 3](#_Toc94626878)

[Q8 4](#_Toc94626879)

[Q9 5](#_Toc94626880)

[Loops 5](#_Toc94626881)

[Q10 5](#_Toc94626882)

[EC Part 1 6](#_Toc94626883)

[EC Part 2 7](#_Toc94626884)

# Instructions

You will submit responses like last time via the Canvas quiz interface.

# Object indexing and extraction

**Run the following to create the qList object:**

qList <- list(first = 3:1,  
 second = "hello",  
 third = c(1:3, "hello"),  
 fourth = data.frame("4.1" = 1:4,  
 Four.2 = 5:8,  
 Four\_Three = 9:12),  
 fifth = matrix(12:1, nrow = 4, ncol = 3))  
#Print to visualize  
qList

## $first  
## [1] 3 2 1  
##   
## $second  
## [1] "hello"  
##   
## $third  
## [1] "1" "2" "3" "hello"  
##   
## $fourth  
## X4.1 Four.2 Four\_Three  
## 1 1 5 9  
## 2 2 6 10  
## 3 3 7 11  
## 4 4 8 12  
##   
## $fifth  
## [,1] [,2] [,3]  
## [1,] 12 8 4  
## [2,] 11 7 3  
## [3,] 10 6 2  
## [4,] 9 5 1

## Q1

Show code to find the class of the third element in qList. The output should be “character”.

## [1] "character"

> class(qList$third)

## Q2-Q5

Write the code for extracting the second column of the fourth element in qList. using the following methods. All of these methods should produce the EXACT same output. **Hint: review old examples of extraction methods**

Fill in the blanks. Make sure you include quotation marks where needed.

**Q2**: Use a dollar signs (No square brackets):  
qList$\_\_\_\_\_$\_\_\_\_\_

> qList$fourth$Four.2

**Q3**: Using only [[]] and [,] with index positions:  
qList[[ \_\_\_\_\_ ]][ , \_\_\_\_\_ ]

> qList[[4]][,2]

**Q4**: Using only [[]] and [,] with names:  
qList[[ \_\_\_\_\_ ]][ , \_\_\_\_\_ ]

> qList[["fourth"]][, "Four.2"]

**Q5**: Extract the fourth element however you want, but extract the second column with a logical vector:  
qList ##### [ , \_\_\_\_\_ ]

> qList$fourth[, c(FALSE, TRUE, FALSE)]

The output of each should be:

## [1] 5 6 7 8

## Q6

The following output was created by printing an object called qList2. What is the length of qList3?

print(qList2)

## $first  
## [1] 3 2 1  
##   
## $second  
## [1] "hello"  
##   
## $third  
## [1] "1" "2" "3" "hello"  
##   
## $fourth  
## X4.1 Four.2 Four\_Three  
## 1 1 5 9  
## 2 2 6 10  
## 3 3 7 11  
## 4 4 8 12  
##   
## $fifth  
## [,1] [,2] [,3]  
## [1,] 12 8 4  
## [2,] 11 7 3  
## [3,] 10 6 2  
## [4,] 9 5 1  
##   
## [[6]]  
## [1] 5  
##   
## [[7]]  
## [1] 6  
##   
## [[8]]  
## [1] 7  
##   
## [[9]]  
## [1] 8

Length=9

## Q7

The following output was created by printing an object called qList3. What is the length of qList3?

print(qList3)

## [[1]]  
## [[1]]$first  
## [1] 3 2 1  
##   
## [[1]]$second  
## [1] "hello"  
##   
## [[1]]$third  
## [1] "1" "2" "3" "hello"  
##   
## [[1]]$fourth  
## X4.1 Four.2 Four\_Three  
## 1 1 5 9  
## 2 2 6 10  
## 3 3 7 11  
## 4 4 8 12  
##   
## [[1]]$fifth  
## [,1] [,2] [,3]  
## [1,] 12 8 4  
## [2,] 11 7 3  
## [3,] 10 6 2  
## [4,] 9 5 1  
##   
## [[1]][[6]]  
## [1] 5  
##   
## [[1]][[7]]  
## [1] 6  
##   
## [[1]][[8]]  
## [1] 7  
##   
## [[1]][[9]]  
## [1] 8  
##   
##   
## [[2]]  
## [1] "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o"

Length=2

## Q8

* Experiment with the c and list functions and figure out how to make qList2 and qList3 from qList1.
* Respond with code that reproduces them when copy and pasted into an R environment that is empty except for qList
* Hint / challenge: I did this using only c, list, [], and : over 2 lines of code; one for each object.

> qList[[6]] <- 5

> qList[[7]] <- 6

> qList[[8]] <- 7

> qList[[9]] <- 8

> list(qList, letters[5:15])

## Q9

Respond with code that reproduces them when copy and pasted into an R environment that is empty except for qList:

Write code to get the result below from adding the following vectors.

* The first three values of the first column of the fourth element in qList
* The fourth row of the fifth element in qList
* The the first element of qList

vectorA + vectorB + vectorC

## [1] 13 9 5

> vectorA <- as.vector(x = qList$fourth$X4.1[1:3])

> vectorB <- as.vector(x = qList$fifth[4,])

> vectorC <- as.vector(x = qList$first)

# Loops

## Q10

The following code simulates the production of all sequences of DNA of a certain size (kmersize). k-mers (<https://en.wikipedia.org/wiki/K-mer>) are small sequences of a certain length (k) commonly used in various bioinformatic processes.

Read through the following line of code. Try and understand each line. Add comment notes.

Once you have tried this, go on Canvas and match the line of code with its description.

Hint: If you are not sure what a line does, beyond looking up information on the functions, try running the script without that line to see what happens.

kmerSize <- 6 #Numbers >>6 may cause memory issues  
nucVector <- c("A", "C", "G", "T")  
nucList <- rep(list(nucVector), kmerSize)  
kmerDf <- expand.grid(nucList)  
kmerDf$Pasted <- NA  
kmerNrow <- nrow(kmerDf)  
for (i in 1:kmerNrow) {  
 currRow <- kmerDf[i, 1:kmerSize]  
 currRow <- unlist(currRow) # Needed to prevent paste0 converting the factors to their number values  
 kmerDf$Pasted[i] <- paste0(currRow, collapse = "")  
 if (i %% 1000 == 1) {  
 iterPerc <- round(i / kmerNrow, 4) \* 100  
 print(paste0(i, "/", kmerNrow, " (", iterPerc, "%)"))  
 }  
}

## [1] "1/4096 (0.02%)"  
## [1] "1001/4096 (24.44%)"  
## [1] "2001/4096 (48.85%)"  
## [1] "3001/4096 (73.27%)"  
## [1] "4001/4096 (97.68%)"

head(kmerDf$Pasted)

## [1] "AAAAAA" "CAAAAA" "GAAAAA" "TAAAAA" "ACAAAA" "CCAAAA"

tail(kmerDf$Pasted)

## [1] "GGTTTT" "TGTTTT" "ATTTTT" "CTTTTT" "GTTTTT" "TTTTTT"

## EC Part 1

Modify the loop code to only produce the 6-mers starting with “AT”. Answer the following:

* How many rows is the resulting data.frame?
* What is the 200th k-mer?

Example head of result:

head(kmerDf.EC$Pasted,100)

## [1] "ATAAAA" "ATCAAA" "ATGAAA" "ATTAAA" "ATACAA" "ATCCAA" "ATGCAA" "ATTCAA"  
## [9] "ATAGAA" "ATCGAA" "ATGGAA" "ATTGAA" "ATATAA" "ATCTAA" "ATGTAA" "ATTTAA"  
## [17] "ATAACA" "ATCACA" "ATGACA" "ATTACA" "ATACCA" "ATCCCA" "ATGCCA" "ATTCCA"  
## [25] "ATAGCA" "ATCGCA" "ATGGCA" "ATTGCA" "ATATCA" "ATCTCA" "ATGTCA" "ATTTCA"  
## [33] "ATAAGA" "ATCAGA" "ATGAGA" "ATTAGA" "ATACGA" "ATCCGA" "ATGCGA" "ATTCGA"  
## [41] "ATAGGA" "ATCGGA" "ATGGGA" "ATTGGA" "ATATGA" "ATCTGA" "ATGTGA" "ATTTGA"  
## [49] "ATAATA" "ATCATA" "ATGATA" "ATTATA" "ATACTA" "ATCCTA" "ATGCTA" "ATTCTA"  
## [57] "ATAGTA" "ATCGTA" "ATGGTA" "ATTGTA" "ATATTA" "ATCTTA" "ATGTTA" "ATTTTA"  
## [65] "ATAAAC" "ATCAAC" "ATGAAC" "ATTAAC" "ATACAC" "ATCCAC" "ATGCAC" "ATTCAC"  
## [73] "ATAGAC" "ATCGAC" "ATGGAC" "ATTGAC" "ATATAC" "ATCTAC" "ATGTAC" "ATTTAC"  
## [81] "ATAACC" "ATCACC" "ATGACC" "ATTACC" "ATACCC" "ATCCCC" "ATGCCC" "ATTCCC"  
## [89] "ATAGCC" "ATCGCC" "ATGGCC" "ATTGCC" "ATATCC" "ATCTCC" "ATGTCC" "ATTTCC"  
## [97] "ATAAGC" "ATCAGC" "ATGAGC" "ATTAGC"

## EC Part 2

Write code to make **aa\_error** exactly the same as **aa** without using any of the whole strings contained within aa. TAs will only be copying and pasting your code to make sure it works.

Ex of **unacceptable** answers:

aa\_error[1] <- aa[1]  
aa\_error[1] <- "All that is gold does not glitter."

aa\_error <- c(  
 "All that is''gold does not glitter.",  
 "The olld that is strong does not wither",  
 "Not all those whoGOwander are lost.",  
 "Deep roots\" are not reached by the Frost."  
)  
  
aa <- c(  
 "All that is gold does not glitter.",  
 "Not all those who wander are lost.",  
 "The old that is strong does not wither.",  
 "Deep roots are not reached by the frost."  
)