Intro to R Questions

Module 2 – Introduction

02-01 - Why R?

1. What is a leading factor that has led to the popularity of R?
2. It is a subscription-based software.
3. **It is free and open-source.**
4. It was designed by two leading statisticians.
5. It is based upon the S programming language.

02-02 - R for Data Science

1. Which of the following is a benefit for data scientists to use R?
2. Great packages for analyzing data
3. Great online community
4. Easy to communicate with others
5. **All of the above**

02-03 - Preparing your workspace

1. When you install RStudio, you automatically have R installed.
2. True
3. **False**

02-04 - Guide to RStudio

1. Which line of code denotes one way to install a package in RStudio?
2. **install.packages(“ggplot2”)**
3. install.packages(ggplot2)
4. library(ggplot2)
5. library(“ggplot2”)
6. install.ggplot2

Module 3 - Hello World! - Basics of R programming

03-01 - Operations

1. What is the first thing that you should do when beginning a new project in R?
2. **Create a new project at the top right of RStudio**
3. Create a new RScript
4. Create a new variable
5. Just begin typing code into the console

03-02 - Variables

1. What is the proper symbol for assigning a variable?
2. %>%
3. =
4. **<-**
5. ->
   1. - Data Types in R
6. Which of the following is the most common data type?
7. Raw
8. Logical
9. Integer
10. **Numeric**
11. Character

03-04 - Coding Style

1. Which of the lines of code below follow R Coding Style?
2. 34+2
3. 34 +2
4. 34+ 2
5. **34 + 2**
   1. - Comments
6. What symbol does R recognize as the beginning of a comment?
7. ^
8. &
9. **#**
10. $

Module 4 – Vectors

04-01 - Vector Creation

1. What happens when you try to create a vector with different data types?
2. It causes an error.
3. **R coerces the components into a single data type.**
4. It creates a vector with different data types.
5. None of the above

04-02 - Selecting Vector Components

1. How would you return “Veery” from the Birds vector below?

birds <- c(“Blue Jay”, “Common Nightjar”, “Eastern Kingbird”, “Veery”)

1. birds[-1]
2. birds[1]
3. **birds[4]**
4. birds[3]
5. birds[“Veery”]

04-03 - Labeling Vector Components

1. Which function will allow you to name the components of a vector after you have already created the vector?
2. labels()
3. **names()**
4. new\_names()
5. vector\_labels()

04-04 - Calculations with Vectors

1. Multiplying the following 2 vectors will result in which of the following?

a <- c(2, 4)

b <- c(4, 2, 3, 1)

a \* b = ?

1. 16, 8
2. 8, 8
3. 6, 4, 8, 8
4. **8, 8, 6, 4**
5. This is not possible; vectors must be the same length in order to multiply them together

04-05 - Base R Functions to Use with Vectors

1. Which line of code would you use to sort the vector “vec1” in descending order?
2. sort(vec1, descending = TRUE)
3. sort(vec1)
4. vec1[order(vec1)]
5. vec1[order(-vec1)]
6. **Both a and d**

04-06 - Comparing Two Vectors

1. Which of the following operators can be used to compare two vectors?
2. **%in%**
3. %>%
4. %
5. \*

04-07 - Modifying Vector Components

1. How would you change the following vector to have a “v” in the 3rd index instead of “c”?

letters <- c(“a”, “b”, “c”, “d”, “e”)

1. letters[1] <- “v”
2. letters[“c”] <- “v”
3. **letters[3] <- “v”**
4. letters[2] <- “v”

Module 5: Matrices

05-01 - Matrix Introduction and Creation

1. Which matrix function will result in a matrix with 5 rows and 3 columns with the letters of the alphabet going sequentially through each row?
2. matrix(“a”:”p”, 5, 3, byrow = TRUE)
3. matrix(letters[1]:letters[15], 5, 3)
4. **matrix(letters[1]:letters[15], 5, 3, byrow = TRUE)**
5. matrix(letters[1]:letters[15], 3, 5, byrow = TRUE)

05-02 - Matrix Naming and Metrics

1. Which metric will allow you to figure out the number of columns in a matrix?
2. ncol
3. nrow
4. dim
5. **Both a and c**

05-03 - Selecting Elements

1. Which option will return a submatrix of the last row in the original 2x3 matrix (org\_matrix) where the last row is named “c”?
2. **org\_matrix[3, ]**
3. org\_matrix[, 3]
4. org\_matrix[, “c”]
5. org\_matrix[“c”]

05-04 - Matrix Arithmetic

1. Which operator performs the dot product between 2 matrices?
2. \*
3. +
4. %o%
5. **%\*%**

05-05 – Matrix Operations

1. The following function will flip the rows and columns of a matrix.
2. flip()
3. **t()**
4. transpose()
5. opposite()

05-06 – Matrix Modification

1. How can you replace all missing values with 0 in the matrix “example\_matrix”?
2. example\_matrix[replace(example\_matrix)] <- 0
3. example\_matrix[is.finite(example\_matrix)] <- 0
4. **example\_matrix[!is.finite(example\_matrix)] <- 0**
5. replace(example\_matrix, NA, 0)

Module 6: Arrays

06-01 - Array Introduction and Creation

1. Under what condition would an array also be considered a matrix?
2. **An array with only 2 dimensions.**
3. An array is an array, it never is a matrix.
4. An array with only 1 dimension.
5. An array with greater than 3 dimensions.

06-02 – Array Similarities to Matrices

1. Given an array with 2 dimensions (meaning it can be an array or a matrix), how would you select the 3rd item in the 2nd row from the 2-dimensional array called “two\_d”?
2. two\_d[3, 2]
3. two\_d[2, 3]
4. **two\_d[2, 3, 1]**
5. two\_d[2, 3, 2]

06-03 - Other Array Operations

1. Which line of code correctly appends the vector to an array after position 9?

v <- array(24:48, dim = c(4,3,2))

1. **append(v, c(1300, 1400, 1500), after = 9)**
2. append(v, c(1300, 1400, 1500, 1600), after = 9)
3. append(v, c(1300, 1400, 1500))
4. append(v, c(1300, 1400), after = 9)

Module 7: Lists

07-01 - List Introduction and Creation

1. How is a list different from a vector?
2. **A list can have multiple different data types.**
3. A list must be all the same data type.
4. A list is a 1-dimensional data structure.
5. A list is a 2-dimensional data structure.

07-02 - List Naming

1. Which of the following is NOT true about naming items in a list?
2. You can rename a single item in a list without naming the others**.**
3. The names() function will allow you to rename items in a list**.**
4. **Naming items in a list is rarely useful and will only just take up time.**
5. After naming an item in a list, you can use that name to call the item without having to know the index of that item**.**

07-03 - Selecting List Elements

1. Given the freedom vector below:

**freedom <- list("greeting" = "hello", "logical" = FALSE, "counts" = c(2,3,1),**

**"second\_logical" = TRUE, "large\_number" = 34, "double" = 3.2)**

How do you select the 3rd item in the “counts” vector?

1. It is not possible to select individual components of a vector within a list.
2. **freedom[[3]][[3]]**
3. freedom[[3]][[“counts”]]
4. freedom[“counts”][[2]]

07-04 - List Manipulation

1. How can you delete an item from a list?
2. del(list[1])
3. **list[1] <- NULL**
4. list[1] <- 0
5. list[1] <- delete

07-05 - List Operations

1. What happens to the names of a list when you convert the list into a vector?
2. **They are converted to the names of each component in the vector.**
3. They become individual components themselves within the vector**.**
4. It is impossible to convert a list into a vector**.**
5. They become the column names in the resulting matrix**.**

Module 8: Factors

08-01 - Factor Introduction and Creation

1. A factor can be created from:
2. **Integers and character strings**
3. Integers only
4. Characters only
5. Logical only

08-02 - Setting Factor Levels

1. John wants to add an additional level to his factor (“j\_factor”). Which of the following would allow him to do so?
2. Labels cannot be made after creating a factor so he would need to create a new factor with a new name.
3. labels(j\_factor) <- c(“new\_level”)
4. j\_factor <- factor(j\_vector, labels = c(“old\_level1”, “old\_level2”, “new\_level”)
5. **j\_factor <- factor(j\_vector, levels = c(“old\_level1”, “old\_level2”, “new\_level”))**

08-03 – Ordering Factors

1. What does ordering factors allow you to do?
2. Add in additional levels to the factor**.**
3. **Compare whether one level is greater than or less than another.**
4. Remove additional levels from the factor**.**
5. It is for aesthetic uses only and doesn’t allow you to do anything further**.**

08-04 - Converting Factors

1. Choose the correct way(s) to convert a factor into a character vector.
2. as.character(char\_factor)
3. labels(char\_factor)
4. levels(char\_factor)
5. **Both a and c**

08-05 – Other Considerations

1. Which combination of functions will allow you to create a function with nicer default labels?
2. **cut() and pretty()**
3. Just pretty()
4. Just cut()
5. split()

Module 9: Loops

09-01 – Loop Introduction and Creation

1. If you are copying and pasting code more than 3 times, what should you consider using?
2. A list
3. A vector
4. **A loop**
5. Magic

09-02 – If-else Statements

1. If-else statements are useful because:
2. They allow you to only execute portions of your code if a condition is met.
3. They are extremely intuitive as we think this way every day.
4. They allow you to perform a check to make sure what you think is happening in your code, is actually happening.
5. **All of the above**

09-03 – For Loops

1. For loops are best used when you:
2. Need to wait for a condition to become FALSE before ending the loop
3. **Have a set number of times you need to execute code**
4. Need to wait for a condition to become TRUE before ending the loop
5. Need to check to make sure your code is performing the right way

09-04 - While Loops

1. While loops are best used when you:
2. **Need to wait for a condition to become FALSE before ending the loop**
3. Have a set number of times you need to execute code
4. Need to wait for a condition to become TRUE before ending the loop
5. Need to check to make sure your code is performing the right way

09-05 – Repeat Loops

1. Repeat loops are best used when you:
2. Need to wait for a condition to become FALSE before ending the loop
3. Have a set number of times you need to execute code
4. **Need to wait for a condition to become TRUE before ending the loop**
5. Need to check to make sure your code is performing the right way

09-06 – Loop Comparison

1. How is a while loop different from a repeat loop?
2. **While loops have the condition at the beginning, while repeat loops have the condition at the end.**
3. While loops run the risk of repeating forever.
4. Repeat loops repeat for only a set number of times.
5. While loops repeat for only a set number of times.

Module 10: Functions

10-01 - Function Introduction and Creation

1. Which of the following is a function in R?
2. func()
3. tidyverse()
4. lubridate()
5. **factor()**

10-02 – Function Arguments

1. What is a default argument in a function?
2. An argument that can take any optional other argument for a function**.**
3. The rule that you must always have at least one argument in a function**.**
4. **An argument that already has a value that doesn’t have to be provided during the function call.**
5. None of the above

10-03 – Nested Functions

1. You can provide a function within the arguments of another function.
2. **True**
3. False

10-04 - Global vs. Local Variables

1. Which option would allow you to change the local and global variable “z”?
2. z <- 1
3. z = 1
4. **z <<- 1**
5. z -< 1

Module 11: Dataframes

11-01 – Dataframe Introduction and Creation

1. What is one similarity and one difference between a dataframe and a matrix?
2. Both can handle multiple different data types; a matrix is only 1-dimensional.
3. **Both are 2-dimensional data structures; a dataframe can handle multiple different data types**.
4. Both are great for data manipulation, a dataframe has set dimensions.
5. Both are made up of lists, the rows and columns in a matrix cannot be named.

11-02 – Tidyverse

1. Which package is not part of the tidyverse?
2. **MASS**
3. dplyr
4. ggplot2
5. purrr

11-03 – Tibbles

1. What is one change from data.frame() in base r to the tibble?
2. Tibbles only print the first 10 rows.
3. data.frame() will change the column names
4. Tibbles do not import character vectors as factors.
5. **All of the above are changes from data.frame() to tibble()**.

11-04 – Tidy Data

1. What are the 3 requirements for tidy data?
2. Each column must be a value, each row an observation, and each cell a variable.
3. **Each column must be a variable, each row an observation, and each cell a value**.
4. Each column must be an observation, each row a variable, and each cell a value.
5. Each column must be a value, each row a variable, and each cell an observation.

11-05 – dplyr and data transformation

1. If you are looking to filter rows, you will use the \_\_\_\_\_\_\_\_\_ function, and if you are selecting columns, you will use the \_\_\_\_\_\_\_\_ function from the dplyr package.
2. **filter; select**
3. select; filter
4. filter; rename
5. rename; select

11-06 – Summarizing Dataframes

1. Which function from the dplyr package is used to summarize data and which makes this function more powerful?
2. **summarize; group by**
3. summarize; tibble
4. summarize; tibble
5. group by; summarize

Module 12: Mini-project

12-01 - Introduction to Mini-project

1. Which of the following tasks would help you the MOST to become the best R programmer you could be?
2. Memorizing R packages and what they do
3. Watching R programming videos
4. Reading someone else’s code
5. **Conducting your own R programming project**

12-02 – Importing Data

1. Joey has a dataset that shows the transaction time of a purchase, the item purchased, and the quantity of that item purchased. What is not a possible question Joey could ask about the dataset?
2. Is there a difference in the number of items purchased in January versus December?
3. **Does Larry sell more products than Joanne?**
4. What is the total number of items sold in the month of August?
5. Is there an item that is always purchased with 2 or more of the items in the cart during the same order?

12-03 - Comprehending the Dataset

1. Which of the following functions is NOT used when trying to comprehend the data?
2. **summarize()**
3. str()
4. head()
5. tail()

12-04 - Tidying Data

1. How many variables belong in a column for the data to be tidy?
2. 0
3. **1**
4. 2
5. 3+

12-05 - Grouping Time Series Data

1. Mallory is a huge fan of Regex (Regular Expressions) and wants to use this, to group and compare time series data. What would you suggest to Mallory to consider instead?
2. Consider converting the time series data into factors.
3. Consider converting the time series data into the logical data type by just comparing them all to a single day.
4. **Consider converting the time series data into datetime objects because they have functions like “month()”, “day()”, “year()”, “hour()”, etc. which makes it really easy to group time series data**.
5. Support Mallory’s decision to use Regex for time series analysis as you would do the same thing.

12-06 – Data Visualization

1. Data visualization through graphs is great for:
2. Understanding trends in data
3. Seeing correlations between variables
4. Summarizing the findings of the data
5. **All of the above**

12-07 – Statistical Analysis

1. Harold is ready to present his findings to the board, but you notice that he has not done any statistical analysis on his data. What would you do?
2. Ask to see how his figures look because they will be good enough.
3. **Tell Harold that he needs to conduct a statistical analysis to see whether these results are actually significant and did not just happen by chance.**
4. Just take a look at the tables, they hold all the information.
5. Wish Harold the best of luck during his presentation.