Quiz questions

Section 3. The building blocks of R

1. Which command creates an object called seq that stores the sequence 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17?
    a. seq <- (7-17)
    b. **seq <- 7:17**
    c. 7:17 -> "seq"
    d. "seq" <- 7;17

2. What is the notation for a comment in R?
    a. //
    b. <!>
    c. **#**
    d. /**/

3. Choose the incorrect definition of a vector
    a. Named data structure that stores data
    b. A sequence of data elements that are of the same type
    c. **Both are correct**
    d. Neither is correct

4. If you are using the combine function to create an integer vector, which command would you use?
    a. object.name <- c(4, 8, 15, 16, 23, 42)
    b. object.name <- combine(4, 8, 15, 16, 23, 42)
    c. object.name <- combine(4l, 8l, 15l, 16l, 23l, 42l)
    d. **object.name <- c(4L, 8L, 15L, 16L, 23L, 42L)**

5. Which command would create a character object?
    a. my.character <- c("I've", "got", "R's", "coercion", "rules", "down", "to", "a", "T")
    b. my.character <- c("I've", "got", "R's", "coercion", "rules", "down", 2, "a", T)
    c. my.character <- c("I've", "got", "R's", "coercion", "rules", "down", 2, "a", "T")
    d. **answer <- c("All", 4, "answers", "are", TRUE)**

6. If I wanted to create a function that simulates flipping a coin 100 times and prints the result, what would be the best way to do it?
    a. flip <- function(){
       coin <- c("heads", "tails")
       throw <- sample(size = 100, replace = TRUE)
       }
    b. **flip <- function(){**
       coin <- c("heads", "tails")

throw <- sample(coin, size = 100, replace = TRUE)
print(throw)}

c. flip <- function{
coin <- c("heads", "tails")
throw <- sample(coin, size = 100, replace = TRUE)
print(throw)}
d. None of the above

7. Choose the incorrect statement.
   a. We use functions to create objects
   b. Functions take arguments
   c. When functions are nested (e.g., round(mean(data))), execution happens from left to right
   d. You can omit the name of an argument if you follow the order in which the arguments of a function are defined

Section 4. Vectors and vector operations

8. Which of the following is true?
   a. Vectors are a sequence of data elements that are of the same type
   b. Multiplying two vectors returns a matrix
   c. R returns an error message if you try to do operations on vectors of different lengths
   d. All of the above

9. What would the following commands return?

```
> my.vec <- 1:6
  a <- c("Patrick", "Squidward", "Plankton", "Mr. Krabs", Sandy Cheeks, "Gary")
  names(a) <- my.vec
  print(a)
```

   a. It would print the object a
   b. It would print the object a, and each element of a will be named with a number from 1 to 6
   c. It would print the object a and each of the numbers from 1 to 6 will have the name of a SpongeBob character
   d. It would prompt an error message

10. Typing ??ls will
    a. Open the help page on the ls() function
    b. Search the CRAN resources for help on the ls() function
    c. Open your browser and search the CRAN resources online
    d. Do nothing. "??" is not a valid command in R

11. What can you find in an R Help page?

a. Examples
b. A Usage section documenting how you're intended to use the function
c. A list of the arguments the function takes
d. A short description of the function's application
e. A technical Details section
f. A Value section explaining what the function returns
g. A See also section or related functions
h. **All of the above (Help pages are super useful!)**

12. **Indexing and slicing are both methods to extract elements from an object. What is R's notation for this?**
    a. **object.name[index]**
    b. object.name<index>
    c. object.name{index}
    d. index(object.name) = index

13. **In R, you reference an element of a vector by:**
    a. Passing in square brackets [] the number indexing the position of the value within the vector
    b. Passing in square brackets [] the element value
    c. Passing in parentheses () the element name
    d. Passing in square brackets [] the element name in quotation marks
    e. **A and D**
    f. A, B, and D

14. **The three most common attributes to give a vector are:**
    a. Names, arguments, and type
    b. **Class, names, and dimensions**
    c. Type, class, and names
    d. Arguments, class, and type

Section 5. Matrices

15. **Given there is an object x, storing the values from 1 to 39 with a step of 2, the following data structure can be the result of which function?**

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    9   17   25   33
[2,]    3   11   19   27   35
[3,]    5   13   21   29   37
[4,]    7   15   23   31   39
```

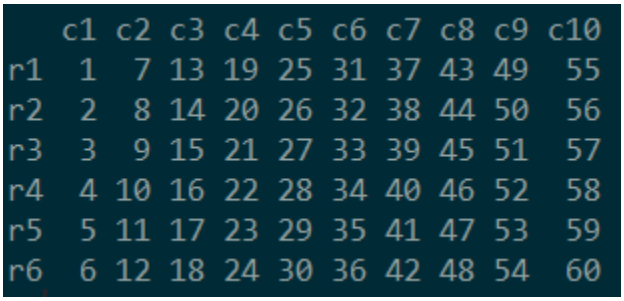
    a. x <- matrix(x, nrow = 4, byrow = T)
    b. dim(x) <- c(4, 5)
    c. x <- matrix(x, ncol = 5, byrow = F)

d. dim(x) <- c(5, 4)
e. A and C
f. B and C
g. C and D

16. Which of the following is correct?
    a. A matrix is a two- or more-dimensional data array
    b. Matrices have a fixed number of rows and columns
    c. The utility of matrices comes from the fact that they can store data of more than one type
    d. None of the above is true

17. The names() function is to a vector as
    a. Colnames() is to the columns of a matrix
    b. Rownames() is to the rows of a matrix
    c. Rbind() and cbind() are to each other
    d. Dimnames = is to the matrix() function
    e. A and B
    f. A, B, and D
    g. C and D

18. You have a 6x10 matrix called a.mat, which looks like this; which command would you use to slice extract the information held in columns 7, 8, and 9?



	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
r1	1	7	13	19	25	31	37	43	49	55
r2	2	8	14	20	26	32	38	44	50	56
r3	3	9	15	21	27	33	39	45	51	57
r4	4	10	16	22	28	34	40	46	52	58
r5	5	11	17	23	29	35	41	47	53	59
r6	6	12	18	24	30	36	42	48	54	60

    a. a.mat[7:9]
    b. a.mat[c(7:9),]
    c. a.mat[,c(7:9)]
    d. a.mat[c("c7", "c8", "c9"),]

19. Scaling a matrix means
    a. To standardize a matrix by multiplying, dividing, etc. the entire object by a single number
    b. To extract a subset of the matrix that spans several columns
    c. To extract a subset of the matrix that spans several rows
    d. To extract a smaller matrix from a larger one

20. Multiplying two matrixes (mat.a*mat.b) returns
    a. A double or a complex matrix product
    b. An error: you must transpose one of the matrices first

c. **A matrix storing the product of the of the two matrices, with the cells being filled in an element-wise fashion**
d. A vector storing the product of each element of mat.a multiplied by the corresponding element of mat.b

21. **Matrix recycling happens when**
    a. You multiply a matrix by a vector
    b. You multiply a matrix by a matrix of different size
    c. You create a matrix from a vector that has fewer elements than specified in the matrix() call
    d. **All of the above**

Section 6. The fundamentals of programming

22. **Select all relational operators**
    a. <
    b. >
    c. =<
    d. <=
    e. >=
    f. =>
    g. =
    h. ==
    i. !==
    j. !=
    k. a, b, c, f, h, i
    l. **a, b, d, e, h, j**
    m. a, b, d, e, g, j

23. **What would the test mode > mope return?**
    a. TRUE
    b. FALSE
    c. An error: you cannot compare character data
       Explanation: This is not character data; these are object names. Unless defined already, R will lash an error message telling you it cannot find the objects
    d. **An error: these are objects and unless previously defined, R won't be able to find them to compare them**

24. **What would the test !("mode" > "mope") return?**
    a. **TRUE**
    b. FALSE

25. **You have a vector s storing the values 3, 1, 2, 5. What would the following operation return: (s < 4) & (s > 2)?**

a. TRUE
b. FALSE
c. TRUE FALSE
d. FALSE TRUE
e. FALSE TRUE TRUE TRUE
f. **TRUE FALSE FALSE FALSE**

26. You have a vector s storing the values 3, 1, 2, 5. What would the following operation return: (s < 4) && (s > 2)?
    a. **TRUE**
    b. FALSE
    c. TRUE FALSE
    d. FALSE TRUE
    e. FALSE TRUE TRUE TRUE
    f. TRUE FALSE FALSE FALSE

27. You have the following for loop. What values will the loop assign to i?
    a. The string "I is i" twenty-two times
    b. Each value in the sequence from 22 to 43
    c. **The string "I is [the corresponding value of i taken from the sequence from 22 to 43]"**
    d. An error message stating that the object i cannot be found in the string from 22 to 43

28. What is a for loop
    a. An expression designed to iterate through a block of code as many times as there are elements in a set of input
    b. An expression that allows you to connect a program with each element in a set
    c. A program that repeats a sequence of instructions under some conditions
    d. **All of the above are correct**

29. Which schema corresponds to a while loop?
    a. A
    b. **B**
    c. C
    d. Neither, the while loop needs to have a while element where there is condition in the schema

30. Choose the incorrect statement
    a. While loops and for loops are programs that repeat a sequence of instructions under certain conditions
    b. While loops have a logical test in their condition statement
    c. **While loops, unlike for loops, return an output**
       Explanation: Unless you specifically ask R to return or to print the output of a loop, the output of the loop will remain behind the proverbial curtain in all three types of loops in R

d. A while loop runs for as long as a condition evaluates to TRUE

31. Choose the incorrect continuation of the sentence "Repeat loops…"
    a. repeat the chunk of code until they encounter a break command
    b. repeat the chunk of code until you tell them to stop
    c. repeat the chunk of code until the condition evaluates to TRUE
    d. repeat the chunk of code until the condition evaluates to FALSE

Section 7. Data frames

32. What makes data frames different from matrices?
    a. They are two-dimensional extensions of a vector
    b. The columns of a data frame can hold elements of different types
    c. Data frames are cumbersome structures difficult to manipulate but because they can store elements of different type, they are the most widely used data structure
    d. The rows of a data frame can hold elements of different types
    e. The cells of a data frame can store multiple objects of different types
    f. All of the above are correct

33. Which of the following is TRUE?
    a. To create a data frame, you need to use rbind() to bind together rows that are of the same length

       Explanation: Rbind() and cbind() actually create a matrix. Use the data.frame() function from the {base} package to bind together columns of equal length, not rows. As you know, the rows of a data frame can store elements of different kinds, whereas vectors cannot – they are limited to storing only one basic type. Therefore, to take advantage of the functionality of a data frame, you ought to bind together columns; each column can store different data (col.a can be a character vector, col.b can be numeric, etc.), resulting in an information-rich structure. This also won't violate the vector "only one basic type" rule 😊

    b. To create a data frame, you need to use cbind() to bind together columns that are of the same length

       Explanation: Rbind() and cbind() actually create a matrix. Use the data.frame() function from the {base} package to bind together columns of equal length. As you know, the rows of a data frame can store elements of different kinds, whereas vectors cannot – they are limited to storing only one basic type. Therefore, to take advantage of the functionality of a data frame, you ought to bind together columns; each column can store different data (col.a can be a character vector, col.b can be numeric, etc.), resulting in an information-rich structure. This also won't violate the vector "only one basic type" rule 😊

c. To create a data frame, you need to use the str() function, passing vectors of the same length as arguments

   Explanation: The str() function returns information about the structure of the object you pass as an argument. To create a data frame, use the data.frame() function and pass vectors of equal length as arguments. They will become the data frame's columns.

d. To create a data frame, you need to use the data.frame() function, passing vectors of equal length to be bound as the rows of the data frame

   Explanation: As you know, the rows of a data frame can store elements of different kinds, whereas vectors cannot – they are limited to storing only one basic type. Therefore, to take advantage of the functionality of a data frame, the vectors you pass will be bound together as the columns of the data frame; each column can store different data (col.a can be a character vector, col.b can be numeric, etc.), resulting in an information-rich structure. This also won't violate the vector "only one basic type" rule 😊

e. None of the above is correct

   Explanation. Indeed. As you know, the rows of a data frame can store elements of different kinds, whereas vectors cannot – they are limited to storing only one basic type. Therefore, to take advantage of the functionality of a data frame, the vectors you pass will be bound together as the columns of the data frame; each column can store different data (col.a can be a character vector, col.b can be numeric, etc.), resulting in an information-rich structure. This also won't violate the vector "only one basic type" rule 😊
   Rbind() and cbind() actually create a matrix.

34. The read.table() family of functions include?
    a. Read.table(), read.data(), read.csv. read.tab()
    b. Read.table(), read.csv(), read.csv2(), read.tab()
    c. Read.table(), read.delim(), read.csv(), read.csv2(), read.delim2()
    d. Read.data(), read.delim(), read.csv(), read.structure()

35. The following is what R returns when the str() function has been called on a certain data set. When importing the data set, what should you have done to store the Species column as character data?

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
>
```

a. Set the header = argument to FALSE
b. Set the sep = “” argument to char
c. Use the read.csv2() or the read.delim2() function, depending on how your text file is organised
d. **Set the stringsAsFactors = argument to FALSE**

36. **If you have a data structure that is saved as something else other than a data frame but you want to convert it into a data frame so you can use all the traditional functionalities of the structure, which command would you use?**
    a. **as.data.frame()**
    b. is.data.frame()
    c. as.df()
    d. as.table()
    e. is.table()

37. **To export your data as a text file, the best function to use would be?**
    a. write.csv(data = data.frame, file = “filepath”, stringsAsFactors = FALSE)
    b. **write.table(data.frame, file = “filepath”, row.names = FALSE, sep = “,”)**
    c. write.csv2(data.frame, file = “filepath”, stringsAsFactors = FALSE)
    d. write.csv(data.frame, file = “filepath”, row.names = TRUE)

38. **Imagine you have a data frame with a lot of variables storing character data. Your row and column names as also characters. Which, if any, would be the function that you wouldn’t really benefit from using if you want to get a general sense of your data frame?**
    a. colnames()
       Explanation: This will generally be helpful to get an idea what variables your data frame has
    b. ncol()
       Explanation: Knowing how many variables you have is generally useful
    c. nrow()
       Explanation: nrow() gives you the number of observations in your data – you get a sense of its size; that’s generally useful
    d. **summary()**
       Explanation: Granted, summary works best with numerical data as it gives you means, medians, minimum, and maximum values, as well 1st and 3rd quartile information, and number of NA values per numeric variable

e. str()
   Explanation: This gives you both variable and observation counts, as well as information about variables stored as factors, and the type of data stored in each variable; overall, useful
f. rownames()
   Explanation: This is a function that isn't particularly useful in any situation involving a data frame with unnamed rows (i.e., a large portion of the time), but in this case, because the data frame has row names, this can give you a sense of the observations in your data.
g. All of these are useful in their own way, even with character data
   Explanation: Most of the time, this would be a correct statement. However, here, the summary() function is pointedly less useful than the rest.

39. To only get the top six rows of a data frame when R returns the result of an operation, you'd use which function?
    a. top()
    b. tip()
    c. head()
    d. tail()
    e. upper()
    f. summary()
40. Look at the following data. Select all the ways in which you could extract the mpg column and get a data frame as your output

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

```
> 
```

a. mtcars$mpg
   Explanation: This returns a vector
b. mtcars[[“mpg”]]
   Explanation: This returns a vector
c. mtcars[1]
d. Explanation: This returns the value in the first cell of the data frame
e. **mtcars[“mpg”]**
f. mtcars[,1]
   Explanation: This returns a vector

41. You have data frame that looks like this:

	Months old	Size	Weight	Breed
Flipper	53	medium	21	dog
Bromley	19	small	8	dog
Nox	34	medium	NA	cat
Orion	41	large	6	cat
Dagger	84	small	7	dog
Zizi	140	extra small	2	cat
Carrie	109	large	NA	dog

Which R statement will replace all the missing values in the Weight variable with the average for the variable?

a. df$Weight[is.na(df$Weight)] <- mean(df$Weight, na.rm = TRUE)
b. pets$Weight[pets$Weight == NA] <- mean(pets$Weight)
c. pets$Weight[pets$Weight == 11] <- NA
d. All of the above are different ways of doing the same thing

Section 8. Manipulating data

42. Have a look at the data frame and the subset of it I have extracted. Which command did I use to do that?

```
# A tibble: 53,940 x 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1 0.230 Ideal     E     SI2      61.5  55.0   326  3.95  3.98  2.43
 2 0.210 Premium   E     SI1      59.8  61.0   326  3.89  3.84  2.31
 3 0.230 Good      E     VS1      56.9  65.0   327  4.05  4.07  2.31
 4 0.290 Premium   I     VS2      62.4  58.0   334  4.20  4.23  2.63
 5 0.310 Good      J     SI2      63.3  58.0   335  4.34  4.35  2.75
 6 0.240 Very Good J     VVS2     62.8  57.0   336  3.94  3.96  2.48
 7 0.240 Very Good I     VVS1     62.3  57.0   336  3.95  3.98  2.47
 8 0.260 Very Good H     SI1      61.9  55.0   337  4.07  4.11  2.53
 9 0.220 Fair      E     VS2      65.1  61.0   337  3.87  3.78  2.49
10 0.230 Very Good H     VS1      59.4  61.0   338  4.00  4.05  2.39
# ... with 53,930 more rows
> 
```

```
# A tibble: 40 x 10
   carat cut     color clarity depth table price     x     y     z
   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1 0.550 Premium D     VVS1     60.3  59.0  3006  5.34  5.30  3.21
 2 0.540 Premium D     VVS1     61.4  57.5  3194  5.25  5.28  3.23
 3 0.710 Premium D     VVS1     58.8  58.0  3952  5.89  5.81  3.44
 4 0.710 Premium D     VVS1     59.2  59.0  4234  5.88  5.84  3.47
 5 0.710 Premium D     VVS1     59.7  59.0  4482  5.81  5.88  3.49
 6 0.900 Premium D     VVS1     61.3  59.0  7711  6.18  6.12  3.77
 7 1.00  Premium D     VVS1     62.9  58.0 10752  6.34  6.28  3.97
 8 1.01  Premium D     VVS1     59.3  59.0 11480  6.56  6.53  3.88
 9 1.11  Premium D     VVS1     62.0  54.0 13405  6.67  6.63  4.16
10 1.20  Premium D     VVS1     62.1  59.0 15686  0     0     0
# ... with 30 more rows
> |
```

a. filter(diamonds, cut = “Premium”, clarity = “VVS1”, depth > 60)
b. filter(diamonds, cut == “Premium”, color == “D”, clarity == “VVS1”, depth >> 60)
c. select(diamonds, cut == “Premium”, color == “D”, clarity == “VVS1”)
d. filter(diamonds, cut == “Premium”, color == “D”, clarity == “VVS1”)
e. arrange(select(cut:clarity), cut)

43. Imagine you are an aspiring botanist and you are going through the data you have on the iris plant in your collection. You have gathered observations about three species – setosa, virginica, and versicolor, and your data includes information about their sepal length and width, and petal length and width. Which combination of functions would you use to get the average values on these variables per species?
    a. select() & summarize()
    b. group_by() & summarize()
    c. filter() & summarize()
    d. arrange_by() & summarize()

44. If you are using the iris data frame (the structure of which is shown below), and you want to sample 1000 observations per group, what would be the best way to do it?
    a. result <- iris %>% group_by(Species) %>% sample_frac(6.68, replace = TRUE)
    b. result <- iris %>% group_by(Species) %>% sample_frac(1000, replace = TRUE)
    c. result <- iris %>% select(Species) %>% sample_n(1000, replace = TRUE)
    d. result <- iris %>% group_by(Species) %>% sample_n(1000, replace = TRUE)

45. The %>% operator from the previous question is called?
    a. A line operator
    b. A pike operator
    c. A pipe operator
    d. A direct operator

46. Consider the following line of code

```
averages <- iris %>%
  group_by(Species) %>%
  summarize(Sepal.Length = mean(Sepal.Length), Sepal.Width = mean(Sepal.Width),
            Petal.Length = mean(Petal.Length), Petal.Width = mean(Petal.Width))
```

The best way to make sense of the pipe operator is to read it in your mind as…?

a. "and"
b. **"then"**
   ***Explanation**: Indeed – the pipe allows us to translate the way we think about operations being executed, the sequences and the continuity, into working code*
c. "before that"
d. "percentage arrow percentage"

47. Look at following data structures; which one is the organised according to the rules of tidy data?

	name	caseA	caseB
1	John Smith	*NA*	18
2	Jane Doe	4	1
3	Agnes Forrest	6	7

	case	john.smith	jane.doe	agnes.forrest
1	a	*NA*	4	6
2	b	18	1	7

	name	case	n
1	Agnes Forrest	A	6
2	Agnes Forrest	B	7
3	Jane Doe	A	4
4	Jane Doe	B	1
5	John Smith	A	*NA*
6	John Smith	B	18

a. The first data frame
b. The second data frame
c. **The third data frame**
d. Neither of these are truly tidy

48. What are the three rules of tidy data?

a. **Column per variable; row per observation, cell per value**
b. Column per observation, row per variable, cell per value
c. Cell per observation, column per variable, row per value
d. Neither. Tidy data is subjective and depends on the analysis
   ***Explanation**: This is a solid argument, however there are universal rules for a tidy data set, and following them makes the data easier to handle, even by people who are unfamiliar with how it has been collected and what the intended analyses are. These rules are as follows: column per variable; row per observation, cell per value*

49. The data you have has column names which are, for all intents and purposes, values. How do you fix that?
    a. By using the spread() function from the tidyr package
    b. By using unite()
    c. **By using the gather() function**
    d. By separating the columns with separate()

50. You have the following data frame, and you are not happy with its general state of tidiness. Which command would produce the best end result?

	name	caseA	caseB
1	John Smith	*NA*	18
2	Jane Doe	4	1
3	Agnes Forrest	6	7

    a. **tidied.data <- r.object %>% gather(case, n, caseA:caseB) %>% mutate(case = gsub("case", "", case)) %>% arrange(name, case)**
    Explanation: *Even though you do not know yet what the gsub function does yet, you ought to be able to guess from the code that it further cleans your data. In this case, it takes the values in the case column created by the gather function, and replaces the "case" in "caseA" and "caseB" with nothing, effectively keeping only the A and the B.* 😊
    b. tidied.data <- r.object %>% separate(caseA, intro = c("case", "A")) %>% mutate(case = gsub("case", "", case))
    c. tidied.data <- t(r.object)
    d. tidied.data <- r.object %>% arrange(name, case)

51. What can be considered the opposite of the gather() function in R?
    a. mutate()
    b. expand()
    c. separate()
    d. **spread()**
    e. unite()
    f. scatter()