R Basics Exercises

RICC International Young Investigator Training

11/13/2023

These exercises will give you some introductory experience with the R programming basics. Please complete the following:

1. What is the sum of the first 100 positive integers? The formula for the sum of integers 1 through n is $n(n+1)/2$. Define $n = 100$ and then use R to compute the sum of 1 through 100 using the formula. What is the sum?

2. Now use the same formula to compute the sum of the integers from 1 through 1,000.

3. Look at the result of typing the following code into R:

```
n <- 1000
x <- seq(1, n)
sum(x)
```

Based on the result, what do you think the functions `seq` and `sum` do? You can use `help`.

a. `sum` creates a list of numbers and `seq` adds them up.
b. `seq` creates a list of numbers and `sum` adds them up.
c. `seq` creates a random list and `sum` computes the sum of 1 through 1,000.
d. `sum` always returns the same number.

4. In math and programming, we say that we evaluate a function when we replace the argument with a given number. So if we type `sqrt(4)`, we evaluate the `sqrt` function. In R, you can evaluate a function inside another function. The evaluations happen from the inside out. Use one line of code to compute the log, in base 10, of the square root of 100.

5. Which of the following will always return the numeric value stored in `x`? You can try out examples and use the help system if you want.

a. `log(10^x)`
b. `log10(x^10)`
c. `log(exp(x))`
d. `exp(log(x, base = 2))`

6. Load the US murders dataset.

```
library(dslabs)
data(murders)
```

Use the function `str` to examine the structure of the `murders` object. Which of the following best describes the variables represented in this data frame?

a. The 51 states.
b. The murder rates for all 50 states and DC.
c. The state name, the abbreviation of the state name, the state's region, and the state's population and total number of murders for 2010.
d. `str` shows no relevant information.

7. What are the column names used by the data frame for these five variables?

8. Use the accessor `$` to extract the state abbreviations and assign them to the object `a`. What is the class of this object?

9. Now use the square brackets to extract the state abbreviations and assign them to the object `b`. Use the `identical` function to determine if `a` and `b` are the same.

10. We saw that the `region` column stores a factor. You can corroborate this by typing:

```
class(murders$region)
```

With one line of code, use the function `levels` and `length` to determine the number of regions defined by this dataset.

11. The function `table` takes a vector and returns the frequency of each element. You can quickly see how many states are in each region by applying this function. Use this function in one line of code to create a table of states per region.

12. Use the function `c` to create a vector with the average high temperatures in January for Beijing, Lagos, Paris, Rio de Janeiro, San Juan, and Toronto, which are 35, 88, 42, 84, 81, and 30 degrees Fahrenheit. Call the object `temp`.

13. Now create a vector with the city names and call the object `city`.

14. Use the `names` function and the objects defined in the previous exercises to associate the temperature data with its corresponding city.

15. Use the `[` and `:` operators to access the temperature of the first three cities on the list.

16. Use the `[` operator to access the temperature of Paris and San Juan.

17. Use the `:` operator to create a sequence of numbers $12, 13, 14, \ldots, 73$.

18. Create a vector containing all the positive odd numbers smaller than 100.

19. Create a vector of numbers that starts at 6, does not pass 55, and adds numbers in increments of 4/7: 6, $6 + 4/7$, $6 + 8/7$, and so on. How many numbers does the list have? Hint: use `seq` and `length`.

20. What is the class of the following object `a <- seq(1, 10, 0.5)`?

21. What is the class of the following object `a <- seq(1, 10)`?

22. The class of `class(a<-1)` is numeric, not integer. R defaults to numeric and to force an integer, you need to add the letter `L`. Confirm that the class of `1L` is integer.

23. Define the following vector:

```
x <- c("1", "3", "5")
```

and coerce it to get integers.

24. Use the `$` operator to access the population size data and store it as the object `pop`. Then use the `sort` function to redefine `pop` so that it is sorted. Finally, use the `[` operator to report the smallest population size.

25. Now instead of the smallest population size, find the index of the entry with the smallest population size. Hint: use `order` instead of `sort`.

26. We can actually perform the same operation as in the previous exercise using the function `which.min`. Write one line of code that does this.

27. Now we know how small the smallest state is and we know which row represents it. Which state is it? Define a variable `states` to be the state names from the `murders` data frame. Report the name of the state with the smallest population.

28. You can create a data frame using the `data.frame` function. Here is a quick example:

```
temp <- c(35, 88, 42, 84, 81, 30)
city <- c("Beijing", "Lagos", "Paris", "Rio de Janeiro",
          "San Juan", "Toronto")
city_temps <- data.frame(name = city, temperature = temp)
```

Use the `rank` function to determine the population rank of each state from smallest population size to biggest. Save these ranks in an object called `ranks`, then create a data frame with the state name and its rank. Call the data frame `my_df`.

29. Repeat the previous exercise, but this time order `my_df` so that the states are ordered from least populous to most populous. Hint: create an object `ind` that stores the indexes needed to order the population values. Then use the bracket operator `[` to re-order each column in the data frame.

30. The `na_example` vector represents a series of counts. You can quickly examine the object using:

```
data("na_example")
str(na_example)
```

```
##  int [1:1000] 2 1 3 2 1 3 1 4 3 2 ...
```

However, when we compute the average with the function `mean`, we obtain an `NA`:

```
mean(na_example)
```

```
## [1] NA
```

The `is.na` function returns a logical vector that tells us which entries are `NA`. Assign this logical vector to an object called `ind` and determine how many `NA`s does `na_example` have.

31. Now compute the average again, but only for the entries that are not `NA`. Hint: remember the `!` operator.

32. Previously we created this data frame:

```
temp <- c(35, 88, 42, 84, 81, 30)
city <- c("Beijing", "Lagos", "Paris", "Rio de Janeiro",
          "San Juan", "Toronto")
city_temps <- data.frame(name = city, temperature = temp)
```

Remake the data frame using the code above, but add a line that converts the temperature from Fahrenheit to Celsius. The conversion is $C = \frac{5}{9} \times (F - 32)$.

33. What is the following sum $1 + 1/2^2 + 1/3^2 + \dots 1/100^2$? Hint: thanks to Euler, we know it should be close to $\pi^2/6$.

34. Compute the per 100,000 murder rate for each state and store it in the object `murder_rate`. Then compute the average murder rate for the US using the function `mean`. What is the average?

35. Compute the per 100,000 murder rate for each state and store it in an object called `murder_rate`. Then use logical operators to create a logical vector named `low` that tells us which entries of `murder_rate` are lower than 1.

36. Now use the results from the previous exercise and the function `which` to determine the indices of `murder_rate` associated with values lower than 1.

37. Use the results from the previous exercise to report the names of the states with murder rates lower than 1.

38. Now extend the code from exercises 2 and 3 to report the states in the Northeast with murder rates lower than 1. Hint: use the previously defined logical vector `low` and the logical operator `&`.

39. In a previous exercise we computed the murder rate for each state and the average of these numbers. How many states are below the average?

40. Use the match function to identify the states with abbreviations AK, MI, and IA. Hint: start by defining an index of the entries of `murders$abb` that match the three abbreviations, then use the `[` operator to extract the states.

41. Use the `%in%` operator to create a logical vector that answers the question: which of the following are actual abbreviations: MA, ME, MI, MO, MU?

42. Extend the code you used in exercise 7 to report the one entry that is **not** an actual abbreviation. Hint: use the `!` operator, which turns `FALSE` into `TRUE` and vice versa, then `which` to obtain an index.

43. We made a plot of total murders versus population and noted a strong relationship. Not surprisingly, states with larger populations had more murders.

```
library(dslabs)
data(murders)
population_in_millions <- murders$population/10^6
total_gun_murders <- murders$total
plot(population_in_millions, total_gun_murders)
```

Keep in mind that many states have populations below 5 million and are bunched up. We may gain further insights from making this plot in the log scale. Transform the variables using the `log10` transformation and then plot them.

44. Create a histogram of the state populations.

45. Generate boxplots of the state populations by region.