R practice exercises

Statistics 243
UC Berkeley

August 28, 2018

Please work through this set of practice problems to make sure you have a basic working knowledge of R. This is not a graded assignment, so feel free to work through these problems in the way which works best for you.

Most of the information you will need is covered in modules 1-4 and 6 of the R bootcamp. If not, it will be noted (you are also expected to be able to pick up new functions when needed).

Creating datastructures

1. Create the following vectors as tersely as possible:
   (a) `1, 2, 3, ..., 49, 50`
   (b) a logical vector that is `TRUE` exactly when the corresponding element of the above vector is even
   (c) `50, 49, ..., 3, 2, 1`
   (d) `1, 2, 3, ..., 49, 50, 49, ..., 3, 2, 1`
   (e) `-10, -9, -8, ..., 8, 9, 10`
   (f) `3, 6, 9, ..., 45, 48`
   (g) `"3", "6", "9", ..., "45", "48"` (Hint: use the previous vector)
   (h) `"a", "a", "a", "a", "b", "b", "b", "c", "c", "d"` (Hint: use `rep`)
   (i) turn the above character vector into a factor vector
   (j) 200 evenly spaced numbers between -1 and 1 (inclusive)
2. Create a data frame through the following steps:
   (a) Create a vector `1, 2, 3, ..., 49, 50` and call it `x`
   (b) Create a vector by taking the cosine of x and call it `y`
   (c) Create a vector by taking the tagent of y and call it `z`
   (d) Create a vector by multiplying the elements of y and z and call it `w`
   (e) Create a logical vector that is TRUE exactly when `x` is between 10 and 29 inclusively and call it `f`
   (f) Create a data frame with column names `x, y, z, w, f` in that order with the obvious content and call it `df1`
   (g) Change the names of `df1` to uppercase letters.
   (h) What would you have done differently if you wanted to use `x` as the row names instead of making it a column? Would you have needed to use `x`?

Subsetting datastructures

1. Create the following:
   (a) A matrix with only the numeric elements of `df1` and call it `m1`
   (b) A matrix `m2` with only the rows where `df1$f` is `TRUE`
   (c) A data frame `df2` with only the rows where `z` is non-negative and has all columns but `z`
   (d) A data frame `df3` without the 3rd and 17th rows of `df1`
   (e) A data frame `df4` with only the even rows of `df1`

Vectorized calculations

1. Create a vector of values $e^{2x}x^{\sqrt{x}}$ for $x = 1, 1.1, 1.2, ..., 2.9, 3.0$.
2. Create the following:
   (a) A 5×5 matrix of zeros called `x`
   (b) See what `row(x)` and `col(x)` return
   (c) Create the following matrix (you could use part (c) for this):

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

   (d) Using `row(x)` and `col(x)`, create the following matrix:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix}$$

3. Create the following matrices:
   (a) (Hint: `outer` works well here)

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

   (b) (Hint: Modify what you did above)

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 3 & 4 & 0 & 1 & 2 \\ 4 & 0 & 1 & 2 & 3 \end{pmatrix}$$

4. Normalize rows and columns
   (a) Create a 5×6 matrix of numbers uniformly drawn from the interval (1, 100) call it `m1`.
   (b) Normalize the rows of `m1` so that they sum to 1. Call this matrix `m2`
   (c) Use `apply` on `m2` to verify that the rows sum to 1. Now use `rowSums` to do the same thing. Which is better? `rowSums` or `apply`?.
   (d) Repeat the last two steps but normalize the columns.
5. Linear model
   (a) Create a vector `x` containing `1, 1.1, 1.2, ..., 9.9, 10.0`
   (b) Create a vector y that is twice `x` but with standard Gaussian noise added
   (c) Create a scatterplot
   (d) Create an `lm` object where `y` depends on `x` called `my_lm`
   (e) Use `lapply` to find the classes of the elements of `my_lm`
   (f) Use `sapply` to find the classes of the elements of `my_lm`
   (g) Do they differ? Why or why not? When would they differ and when would they be the same?

Functions

1. Write a function which returns the sum of the absolute deviations from the median of an input vector x. Add the following:
   (a) Make sure the input vector x is numeric.
   (b) An additional argument `na.rm` which is a logical. If it is `TRUE`, the function removes all the NAs from the computation of the return value. Give it a default value of `FALSE`.
2. Simulate a coin toss
   (a) Use the `sample` function to sample with replacement a vector of 0s and 1s with 100 elements. Call this `x`. Do it again and call it `y`. Would it make sense to call `set.seed` before calling `sample`? Why?
   (b) Write a function `sum_heads` that takes as input the number of desired coin flips and returns the number of heads (assume heads are coded by 1). Would it make sense to call `set.seed` in the body of your function? Why?
   (c) Create a new vector `sums` by calling `sum_heads(200)` 10,000 times. (Hint: use `replicate`)
   (d) Plot a histogram of `sums`
3. Write a function that takes two numeric vectors `x` and `y` as well as a variable `operation` with a default value of `"add"`.
   (a) If `operation` is `"add"`, return `x+y`
   (b) If `operation` is `"subtract"`, return `x-y`
   (c) If `operation` is `"multiply"`, return `x*y`
   (d) If `operation` is `"divide"`, return `x/y`
   (e) If `operation` isn't one of the above, return an error that `operation` is unknown.
4. Write a function that takes a vector `x` and returns a vector containing the cumulative sum vector. Note that R provides a builtin function `cumsum` that you can use to verify that your function works. You should implement this function using a `for` loop to make sure you understand how it works.