Spring 2022 courses

DONE ds quiz 6

Settings

This quiz covers the `apply` family of functions, and some of our lab practice session material. Rules of the game:

- A question has only ONE right answer unless otherwise noted
- After the first play, the quiz will be opened for unlimited play
- Some questions were inspired by the DataCamp presentation
- Get the DataCamp mobile app and complete daily practice runs!
- Let me know if you have any comments or corrections

This is the last quiz before the next test.

Which key sequence runs an Org-mode code block?

/ More than one answer is correct/

TRUE

- C-c C-c
- M-x org-babel-execute-src-block

FALSE

- C-c C-o
- M-x indent-region

Feedback: C-c C-o runs org-open-at-point, to open links (to image files, URLs etc.). M-x indent-region indents each nonblank line in the region - bound to C-M-\ - inside code blocks, it establishes indentation according to the syntax.

The `lapply` function replaces a `for` loop by running a function over all elements of a list

TRUE

Feedback: There are several related functions in R which allow you to apply some function to a series of objects (eg. vectors, matrices, dataframes, lists or files). They include: lapply sapply, tapply, aggregate, mapply, apply. (Source)

Complete the code!

`cities` is a `character` vector of city names. Complete the code parts [1] [2] below to obtain the number of characters of each city in `cities` using `sapply`, a user-friendly version of `lapply`.

```
cities <- c("Jonesboro", "Batesville", "Conway", "Searcy")

sapply(cities, nchar)
```

```
Jonesboro Batesville     Conway     Searcy
        9         10          6          6
```

More than one answer is correct.

True:

- [1] cities [2] nchar
- [1] cities[1:4] [2] nchar
- [1] X=cities [2] FUN=nchar

  False:

  - [1] cities() [2] nchar()

Feedback: `sapply` is a simple version of `lapply` for vectors. It requires the format (check `help(sapply)`): `sapply(X = v, FUN= fun)` for a vector `v` and a function `fun`. The names `X` and `FUN` can be omitted. `cities[1:4]` is an index version of `cities`. The function `cities()` is not known, and `nchar()` is missing an argument, and cannot be used like that inside `sapply`.

```
cities <- c("Jonesboro", "Batesville", "Conway", "Searcy")

sapply(cities, nchar)
sapply(cities[1:4], nchar)
sapply(X=cities, FUN=nchar)
```

```
Jonesboro Batesville     Conway     Searcy
        9         10          6          6
Jonesboro Batesville     Conway     Searcy
        9         10          6          6
Jonesboro Batesville     Conway     Searcy
        9         10          6          6
```

Which function turns a list `l` into a vector?

TRUE:

- unlist(l)

  FALSE:

  - as.vector(l)
  - is.vector(l)
  - vector(l)

  Feedback: `as.vector()` converts a matrix into a vector. `is.vector` checks if its argument is a vector or not. The function `vector()` generates a (by default logical) empty vector.

  ```
  cities <- c("Jonesboro", "Batesville", "Conway", "Searcy")
  l <- lapply(cities,nchar)
  u <- unlist(l)
  v <- as.vector(l)
  ```

```
identical(v,1)
identical(u,1)
```

```
[1] TRUE
[1] FALSE
```

```
li <- list(1, "foo")
unlist(li)
is.matrix(as.matrix(li))
class(as.matrix(li))
vector(li)
as.vector(matrix(1:4))
vector()
```

```
[1] "1"   "foo"
[1] TRUE
[1] "matrix" "array"
Error in vector(li) : invalid 'mode' argument
[1] 1 2 3 4
logical(0)
```

Complete [1] [2] and [3] in the code below to get the output.

- `addr` is a function that adds its two arguments.
- `foo` is a numeric vector

Identify [1] [2] [3] to obtain the output below!

```
addr <- function(x, y) x + y      # define function
foo <- c(1.45, 4.4, 2.33, 5.0)    # define list

bar <- lapply([1], [2], y = [3])  # run function
unlist(bar)                       # print as vector
```

```
Error: unexpected '[' in "bar <- lapply(["
Error in unlist(bar) : object 'bar' not found
```

Output:

```
: [1]  6.45  9.40  7.33 10.00
```

TRUE:

- [1] foo [2] addr [3] 5

FALSE:

- [1] bar [2] foo [3] x
- [1] foo [2] TRUE [3] bar
- [1] foo [2] addr [3] TRUE

Feedback: `lapply` has the arguments `(X, FUN, FUN.VALUE)` where `FUN.VALUE` is a vector of return values from `FUN`, which is applied to the list `X`. In this case, `FUN.VALUE = y = 5`.

```
addr <- function(x, y) x + y       # define function
foo <- list(1.45, 4.4, 2.33, 5.0)  # define list

bar <- lapply(foo, addr, y = 5)    # run function
unlist(bar)                        # print as vector
```

```
[1]  6.45  9.40  7.33 10.00
```

What is the output of this code?

Tip:

- `strsplit` splits its argument vector according to its `split` attribute.
- `length` returns the number of elements of a vector
- `sum` sums the numeric elements of a vector

```
cities <- c("Jonesboro:large", "Batesville:small", "Searcy:medium")

citySize <- strsplit(x=cities, split=":")
len <- sapply(citySize,length)
sum(len)
```

```
[1] 6
```

TRUE:

- 6

FALSE:

- TRUE
- FALSE
- 3

Feedback: After the split, `citySize` is a list of three elements - each element is a character vector like `"Jonesboro" "large"`. `len` is the vector `2 2 2`, with the `length` of each list element, and `sum` sums these values to obtain `6`.

```
cities <- c("Jonesboro:large", "Batesville:small", "Searcy:medium")

citySize <- strsplit(x=cities, split=":")
len <- sapply(citySize,length)
sum(len)
citySize
```

```
[1] 6
[[1]]
[1] "Jonesboro" "large"
```

```
[[2]]
[1] "Batesville" "small"

[[3]]
[1] "Searcy" "medium"
```

What is an "anonymous" function?

TRUE:

- An unnamed function

FALSE:

- A system function
- A function that has not been loaded into R
- An incomplete function

Feedback: "In R, functions are objects in their own right. They aren't automatically bound to a name. Unlike many languages (e.g., C, C++, Python, and Ruby), R doesn't have a special syntax for creating a named function: when you create a function, you use the regular assignment operator to give it a name. If you choose not to give the function a name, you get an anonymous function." (Wickham, Advanced R, 2019). Unnamed functions are also called "lambda expressions".

User-defined functions can be save and re-used later

TRUE

Feedback: the corresponding functions are `save(function, file='path')` and `load(file='PATH')`.

Be the interpreter - fix the code!

The code block below generates a scatter plot of one-hundred random numbers. I want to save the graphic output in the file `plot.png`, but when I open the link in the `#+Results:` below, I cannot find the file. Where is it?

```
#+name: plot
#+begin_src R :session *R* :results output graphics file :file plot.png
  plot(rnorm(100))
#+end_src

#+Results: plot
file:plot.png
```

More than one answer is correct.

TRUE:

- The Org-mode file is in a different directory from the one used by the `*R*` session.
- The header argument does not contain the path to the file

FALSE:

- The graphics device has to be opened with `dev.new()` first

- PNG is the wrong file type

Feedback: You decide the filetype. R can do anything. `dev.new()` is a graphic device control command, but Org-mode takes care of that. You can check this if you leave out the `file :file plot.png` part - then the `graphics` result will be opened in a new device (which you can see in the `dev.list()`.) An absolute or relative path to the file would solve the problem (it would show up in the link). The current working directory (`getwd()`) used by the R session could indeed be different from the one in which the Org-mode file is - the default link opens in the current directory.

The `sapply` function is a simplified version of `lapply` for arrays

Tip: vectors and matrices are one and two dimensional arrays, respectively.

TRUE

Author: Marcus Birkenkrahe
Created: 2022-03-11 Fri 13:12
Validate