R Coding for Fun and Profit - 1

Code

Data Types, Variables, Functions

AUTHOR
James R. Hunter, D.Sc.

AFFILIATION
Laboratório de Retrovirologia, UNIFESP

PUBLISHED
October 13, 2024

Let's do some coding!

I will generally not execute the code in the code blocks below so you can have that pleasure. However, if we do these in class, you will see the effect of these commands there.

Hello, World!

The universal first attempt at code in any language.

```
print("Hello, World")
```

Type this text in your Console and hit `<Enter>`.

Let's make it a bit more personal. We'll add your name. The result we want is "Hello, Jim" or whatever your name is.

To do that, we have to capture your name, which we do with `readline().`

```
readline("What's your name? ")
```

Bug

What's the problem with this attempt?

What are we going to do with the result? How will we feed it into the print?

We need to save the name in an R object like a string (character vector) and then we can use it.

```
name <-  readline("What's your name? ")
name
print("Hello,", name)
```

Bug

We want `print()` to print both elements, the word "Hello," and the name.

To combine elements together, we use the command `c()`, with the elements separated by commas.

```
name <-  readline("What's your name? ")
print(c("Hello,", name))
```

> **Not So Bad Bug**
>
> Program ran correctly but did not produce the nice output we want. It printed what we told it to print, the separate elements of the vector we created with c() .

The function `paste()` will combine elements into a single string, as the name implies.

```
name  <-  readline("What's your name? ")
print(paste("Hello,", name))
```

New Commands

- print()
- readline()
- <- (attribution symbol)
- c()
- paste()

Strings

We can do a lot with strings. We have a data base with various data entered about a number of patients in a study. The people who entered the data were a bit careless. We need to fix these data to make them useful for our analysis. We will take them 1 variable at a time.

Here are the first names of the five patients in the study.

```
first_names <- c("Jorge", "susana", "  José", "Maria aparacida", "  RoBERto  ")
```

We have errors of capitalization and extra white space.

Tidyverse Functions

We will use the string functions of the tidyverse, which sit in the `stringr` package of functions. This package is also automatically called if you load the `tidyverse` package into memory, which is what we will do with the `library()` command. As you would expect, `library()` loads the library you called. We will check that by executing `sessioninfo().`

```
library(tidyverse)
sessionInfo()
```

```
R version 4.4.1 (2024-06-14)
Platform: aarch64-apple-darwin20
Running under: macOS 15.0.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-
arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-
arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3.12.0

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Sao_Paulo
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1   dplyr_1.1.4
 [5] purrr_1.0.2     readr_2.1.5     tidyr_1.3.1     tibble_3.2.1
 [9] ggplot2_3.5.1   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.5      jsonlite_1.8.9    compiler_4.4.1    tidyselect_1.2.1
 [5] scales_1.3.0      yaml_2.3.10       fastmap_1.2.0     R6_2.5.1
 [9] generics_0.1.3    knitr_1.48        htmlwidgets_1.6.4 munsell_0.5.1
[13] pillar_1.9.0      tzdb_0.4.0        rlang_1.1.4       utf8_1.2.4
[17] stringi_1.8.4     xfun_0.48         timechange_0.3.0  cli_3.6.3
[21] withr_3.0.1       magrittr_2.0.3    digest_0.6.37     grid_4.4.1
[25] rstudioapi_0.16.0 hms_1.1.3         lifecycle_1.0.4   vctrs_0.6.5
[29] evaluate_1.0.1    glue_1.8.0        fansi_1.0.6       colorspace_2.1-1
[33] rmarkdown_2.28    tools_4.4.1       pkgconfig_2.0.3   htmltools_0.5.8.1
```

You can see that the `stringr` package we want is listed as loaded under the heading of “other attached packages”.

Capitalization

The first problem appears in the second case: “susana”. The first S needs to be upper case. We can use the str_to_title() function in stringr to accomplish this.

```
fname <- first_names[2]
# just assigning a value to a variable will not print it to the screen
fname <- str_to_title(fname)
# an external parenthesis will force a result to be printed.
(fname <- str_to_title(fname))
```

```
[1] "Susana"
```

We also have “Maria aparacida” as the fourth name. It also needs capitalization. Can this also be done with `str_to_title()` ? Yes, but what about our the capitalization in our last entry: ” RoBERto “?

```
str_to_title(first_names[4:5])
```

We can apply this function to the entire variable and take care of fixing the capitalization of all the names in one command.

```
(first_names <- str_to_title(first_names))
```

White Space

Now, to resolve the extra white space in the third and fifth entries. The tidyverse has a command str_trim() that takes care of this.

```
str_trim(first_names[c(3, 5)])
```

Combine the Operations

We can successively fix these two conditions in the same block of code as follows:

```
first_names <- str_to_title(first_names)
first_names # only to show the intermediate result
first_names <- str_trim(first_names)
first_names
```

Next week, we will learn a way that we can make this block more efficient both to type and to execute.

New Commands and Packages

- library
- sessionInfo (watch for the capital I)
- tidyverse
- stringr
- str_to_title()
- str_trim