Stat 33B - Lecture Notes 1

August 23, 2020

R Basics

R has a Read-Eval-Print Loop (REPL):

1. Type an expression at the R prompt and hit the enter key.
2. R reads the expression.
3. R evaluates the expression to compute a result.
4. R prints the result in the console.
5. R loops back to waiting for you to enter an expression.

This is similar to Python, Julia, Lisp, etc.

R has many built-in functions for doing math and stats.

Getting Help

R has built-in documentation.

You can use the `?` command to get help with a specific function:

```
?sin
```

You can use the `??` command to search the documentation:

```
??graphics
```

Strings use single or double quotes (there's no difference).

```
"hi"
```

```
## [1] "hi"
```

```
'hi'
```

```
## [1] "hi"
```

The help commands work with strings or unquoted names.

```
?"+"
```

The `sessionInfo()` function prints info about your R session:

```
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19041)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] ggplot2_3.3.2
##
## loaded via a namespace (and not attached):
##  [1] knitr_1.29       magrittr_1.5     munsell_0.5.0    colorspace_1.4-1
##  [5] R6_2.4.1         rlang_0.4.7      stringr_1.4.0    tools_4.0.2
##  [9] grid_4.0.2       gtable_0.3.0     xfun_0.16        tinytex_0.25
## [13] withr_2.2.0      htmltools_0.5.0  ellipsis_0.3.1   yaml_2.2.1
## [17] digest_0.6.25    tibble_3.0.3     lifecycle_0.2.0  crayon_1.3.4
## [21] vctrs_0.3.2      glue_1.4.1       evaluate_0.14    rmarkdown_2.3
## [25] stringi_1.4.6    compiler_4.0.2   pillar_1.4.6     scales_1.1.1
## [29] pkgconfig_2.0.3
```

Order of Operations

The order of operations in R is similar to math and most programming languages.

To see the entire order:

```
?Syntax
```

Functions & Calls

Recall that:

- **Parameters** are the inputs a function accepts.
- **Arguments** are the values assigned to parameters in a call.

You can set arguments by position or by name:

```
log(10)
```

```
## [1] 2.302585
```

```
?log

log(10, 2)
```

```
## [1] 3.321928
```

```
log(base = 2, x = 10)
```

```
## [1] 3.321928
```

```
log(base = 2, 10)
```

```
## [1] 3.321928
```

Copy-on-write, Part 1

In R, most objects are **copy-on-write**.

That is, if we assign `x` to `y`:

```r
x = 3
y = x
```

And then change `x`:

```r
x = 5
```

Then `y` remains unchanged:

```r
y
```

```
## [1] 3
```

Originally, `x` and `y` referred to the same value in memory.

When we changed `x` (a "write"), R automatically copied the original value so that `y` remained the same.

Packages & Notebooks

A **package** is collection of functions and/or data for use in R.

The Comprehensive R Archive Network (**CRAN**) stores most user-contributed packages.

You can install packages from CRAN with `install.packages()`.

For example:

```r
#install.packages("ggplot2")
```

A package only needs to be installed once.

For maintaining your packages, there are also the functions:

- `installed.packages()` to list installed packages
- `remove.packages()` to remove a package
- `update.packages()` to update ALL packages

Loading Packages

The `library()` function loads an installed package:

```r
library(ggplot2)
```

Only load the packages you actually need.

You'll have to reload the packages each time you restart R.

Notebooks

Two typical ways to save R code:

- R script (.R file)
- R notebook (.Rmd file)

R scripts are simpler:

- No extra packages required
- Ideal for developing software

R notebooks are richer:

- Can store formatted text and code

- Can be converted to HTML, DOCX, and PDF
- Ideal for data analyses and presentations

R notebooks require the `rmarkdown` package:

```r
install.packages("rmarkdown")
```

Generating a report from an R notebook is called **knitting**.

TinyTeX

If you want to knit PDFs from R notebooks, you also need LaTeX.

LaTeX is programming language for typesetting books.

The `tinytex` package aims to make installing LaTeX easy.

First, install `tinytex`:

```r
install.packages("tinytex")
```

Second, tell `tinytex` to install LaTeX:

```r
library(tinytex)
install_tinytex()
```

This may take a while, and you may need administrator permissions.

Finally, restart R and try knitting an R notebook.

Remember that the output type must be `pdf_document`.

Vectors

R has no concept of scalars or arrays.

R's atomic data type is the **vector**, an ordered container for 0 or more elements.

Vector elements must all have the same data type.

The `c()` function combines vectors:

```r
x = c(5, 7, 1)
x
```

```
## [1] 5 7 1
```

```r
c(x, 1)
```

```
## [1] 5 7 1 1
```

```r
c("hi", "hello")
```

```
## [1] "hi"    "hello"
```

```r
"hi"
```

```
## [1] "hi"
```

```r
c("hi", 1)
```

```
## [1] "hi" "1"
```

Vectorization

A **vectorized** function is one that is applied element-by-element when passed a vector argument.

Many R functions are vectorized:

```r
c(sin(0), sin(1), sin(2))
```

```
## [1] 0.0000000 0.8414710 0.9092974
```

```r
x = c(0, 1, 2)
sin(x)
```

```
## [1] 0.0000000 0.8414710 0.9092974
```

```r
# NOT VECTORIZED:
mean(x)
```

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

Vectorization is the fastest kind of iteration in R.

Indexing

In R, indexes start at 1.

Use the square bracket `[` to access elements of a vector:

```r
x = c(1, 3, 7)
x[2]
```

```
## [1] 3
```

```r
x[6]
```

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

You can use a vector as an index:

```r
x[c(1, 1, 2)]
```

```
## [1] 1 1 3
```

Copy-on-write, Part 2

The copy-on-write rule applies to vectors.

For example:

```r
x = c(10, 20, 30)
y = x
x
```

```
## [1] 10 20 30
```

```r
y
```

```
## [1] 10 20 30
```

```r
x[1] = 15
x
```

```
## [1] 15 20 30
```

```r
y
```

```
## [1] 10 20 30
```

This is different from languages like C and Python.