ndexr: A Hanbook on R

Freddy Ray Drennan

2021-11-15

Contents

Chapter 1

Prerequisites

- Install R
- Install R Studio
- Windows Only: Install RTools
  - When installed, run in the RStudio Console: `write('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"', file = "~/.Renviron", append = TRUE)`
- Windows Only: Install WSL2
  - Computer should be completely updated before install.
- Install Git
- Create Github Account
- Fork r-handbook
- Install Docker and Docker Compose
- Create AWS Account
  - Billing will be discussed in the course, but don't expect to pay much - i.e., 10-20 dollars a month for high course activity.
  - Remember to `stop` EC2 servers when we begin using them. AWS is polite about your first few refund requests.
- Create Reddit Account
  - Follow Instructions here

Make sure you install the `tidyverse` packages.

```
install.packages(c('tidyverse'))
```

Chapter 2

Functions

2.1 Indexes

For example, consider the values $1, 2, ...n$. Programmers will often talk about indexing. An index in R starts at `1` and extends through the integers until `n` equals the number of items in the list, or vector, or columns of a table, etc.. So suppose we have the vector `(2, 4, 5)`, we can take the value at location or index `2`, and get `4` back. Indexes are useful, because we often subset or filter our data based on the task at hand. The function `seq` generates a sequence of integers between the `from` and `to` argument.

```
seq(from = 1, to = 10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

What is special about starting the sequence at `1` is that it creates a vector of integers that also match their location in the vector. When you think about vectors, remember that whether they be logical (`TRUE`/`FALSE`), numeric (`1.1`, `2`), integer (`1`, `4`), or character (`"a"`, `"cat"`), there is *always* an integer associated with it's position in the object.

For example, R has a built-in `constant` called `letters`. This means that no matter where you are writing R, `letters` will be available to you without opening up a library with the value. We see that the letters vector contains 26 cells, which we can access with the integer vector we mentioned above.

```
message('Full letters vector')
```

```
## Full letters vector
```

```
print(letters)
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```
message('Selecting the first 10 values')
```

```
## Selecting the first 10 values
```

```
letters[seq(from = 1, to = 10)]
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

2.2 What are Functions?

Functions are containers where anything or nothing can happen. They allow for generalization of complicated ideas and routines that we wish to repeat many times. A function may have an input, but no output. It may have an output, but no input, both or none. If it's something you need to do repeatedly, or containing code makes your program easier to read, then write a function.

We can assign this **integer vector** to a **variable** called `data` . When we assign something to a variable, we can take it with us to use in other parts of our program. We will use a built in constant that R provides called `LETTERS` which is a **character vector** containing the 26 letters of the alphabet capitalized.

```
data = seq(1, 10)
data <- setNames(data, LETTERS[data])
print(data)
```

```
##  A  B  C  D  E  F  G  H  I  J
##  1  2  3  4  5  6  7  8  9 10
```

Maybe we are interested in how long the vector is, or what its names are.

```
length(data)
```

```
## [1] 10
```

```
class(data)
```

```
## [1] "integer"
```

```
names(data)
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

2.3 Sequences

Suppose we have the function $g(x) = x^2$, and want to use it. We can define the function below, and call it `square`. The function can be named whatever we like, `square` could be `platypus` for all I care. It doesn't really matter, except to whoever is reading your program - likely you at a later, more forgetful date. Generally, we want to name the things we create something useful, so that when we are reading our program, we understand what is happening. Naming functions is a

```
square = function(x) x^2
square(3) #
```

```
## [1] 9
```

Chapter 3

Literature

Here is a review of existing methods.

Chapter 4

Methods

We describe our methods in this chapter.

Chapter 5

Applications

Some *significant* applications are demonstrated in this chapter.

5.1 Example one

5.2 Example two

Chapter 6

Final Words

We have finished a nice book.