post01-amy-zhu

Amy Zhu

October 31, 2017

Introduction

I was very fascinated with the power of vectors in R since learning about it. The convenience of manipulating vectors with single operations is exciting. Compared to other languages like Java, R does not require the user to hard code a loop to go through each element in order to perform a uniform action on each element. In this post write-up, I will talk about the basic features of vector and go into some deeper characteristics that we did not cover in class.

What is Vector?

There are two types of vectors in R, atomic and lists. The distinction between the two is that lists can contain other vectors, whereas atomic vectors are single layer. A vector in R is essentially like arrays in other languages. The elements in the atomic vector must have the same data type. However the lists contained within lists may have different data types. This is because when performing functions such as arithmetic operations and character manipulations on the individual elements of an atomic vector, the functions take in specific value types as arguments. This characteristic is the same across languages. Vectors in R can take a total of 5 different data types, which are numerical, logical, complex, raw, and character. Logical values are what we normally call boolean values, which can only either take a TRUE value or FALSE value. Numerical are numbers, not limited to integer or doubles, which other languages distinguish. Characters are values that take in strings and are enclosed by quotation marks. Unlike the three I just mentioned, complex and raw data types are normally not used in data analysis.

Atomic Vectors

Atomic vectors are the type of vectors we have been handling in class most of the time. Depending on the data type of the vector, we can perform various operations on the vector. If the vector is of numeric type, we can for example take the square root of every element in the vector. This ability is extremely convenient for manipulating data and is not common to all languages.

Lists

Lists are essentially vectors that can contain other vectors. Unlike atomic vectors, lists can contain different data types. This can be done because the list contains atomic vectors which are single data type only. This is useful when we need to represent hierarchical structures. For example, a list representing the scores five students received on their midterm and final may look like this:

[[88, 70], [80, 61], [90, 87], [53, 36], [68, 71]]

The first elements of the sublists are the midterm scores and the second element of the sublists are the final scores (i.e. student one received 88 on midterm and 70 on final). This can be created like this:

```
x <- list(c(88, 70), c(80, 61), c(90,87), c(53,36), c(68,71))
```

The depth of the lists can go further, meaning there can be a list of lists of lists.

Index

Indexing in R is unconventional compared to other programming languages. Most languages use zero indexing whereas R starts indices at one. What this means is that the first element in the vector can be accessed by passing in 1 as i in v[i] operation. I often get confused because I am used to zero indexing.

Attributes

Vectors can contain additional metadata in the form of attributes. Attributes are essentially external information/description about a vector that is not represented in the vector. Attributes can be assigned like the following:

```
attr(x, "persona") <- "students"
attr(x, "tests") <- "midterm and final"
attributes(x)
```

```
## $persona
## [1] "students"
##
## $tests
## [1] "midterm and final"
```

Memory

Memory usage is important when user is dealing with limited memory and when user want to write code that will run faster on large data sets, which is extremely useful and efficient. I will mainly talk about memory management specific to vectors. In R, every length zero vector, regardless of data type uses us 40 bytes of memory. The distribution of those 40 bytes are as follows: . 4 bytes - object metadata . 16 bytes - two pointers (8 bytes each) one pointing to the next object in memory and one pointing to the previous object. This kind of creates a linked list scenario in both ways allowing easy loops through the elements. * 8 bytes - pointer to the attributes * 4 bytes - length of vector; this is very interesting because every byte is 8 bits so with 4 bytes as length of vector, we would expect the maximum number of elements a vector can hold is 2^{31}, however in R 3.0.0 and later, vectors can store up to 2^{52} elements. * 4 bytes - “true” length of vector, normally not used besides to represent the space allocated when using a hash table, which is a type of data structure * 0 bytes initially - actual data

Summary

It was interesting researching about vectors and the specific properties of vectors in R. The most fascinating fact I learned was that vectors can store up to 2^{52} elements with only 4 bytes. Vectors make R a very useful language for data manipulation and there are a lot of advanced concepts we did not cover in class.

References

- http://r4ds.had.co.nz/vectors.html
- http://adv-r.had.co.nz/memory.html
- http://www.philender.com/courses/multivariate/notes/vecr.html
- http://www.r-tutor.com/r-introduction/vector
- https://www.rdocumentation.org/packages/base/versions/3.4.1/topics/vector
- http://www.cyclismo.org/tutorial/R/basicOps.html
- t/datatypes.html