Foundations of Data Analysis

The University of Texas at Austin

R Tutorials: Week 1

Vectors

In this R tutorial, we're going to learn how to assign data to a vector. So assigning specific values to objects is sometimes useful. However, most of the time it's going to be a lot more useful if we want to assign a vector or a set of numbers to a specific object. To do that we're going to need to learn a new function called the concatenate function. So if I have a set of data that I want to store in say, the object called myvariable, I can assign the word myvariable to a vector by using the concatenate function, which is just the letter "c".

Again, I have a function so I'm going to be using parentheses. And for the concatenate function, I'm going to give it just a list of numbers that I want to basically combine into a single object, which is a vector. So say I have some data that is the following numbers, and I want to store these into the object called myvariable.

```
myvariable <- c(72, 28, -9, 12, 11)
```

Well, if I submit this line– again, a couple things are going to happen. It's going to submit it down ... on the console, but now I'm going to see this new item in my workspace, or my environment, that is actually a vector of numbers.

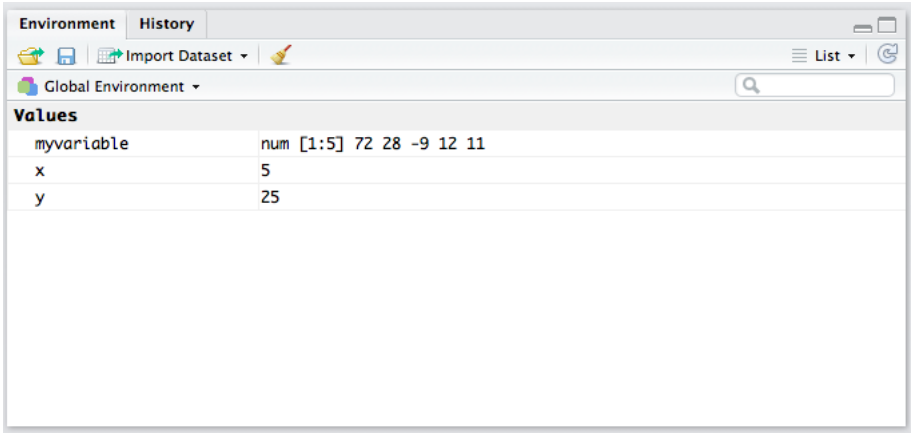

Figure 1: RStudio environment window.

So depending on the version of R that you have, this might say vector, it might say num and then actually give you the values, but you'll see that this is different from just the simple numbers that x and y are holding. "myvariable" is holding this set of numbers.

So now that we have a vector of numbers, we can actually do a lot of fun things with it. We can do some multiplication, division, addition, and subtraction on this vector numbers all at once because now they're contained in a single object. So say I wanted to take "myvariable", and I wanted to take every single value and multiply it by 3. I can just take the object name times 3, and now I get those five numbers displayed all multiplied by 3.

```
myvariable*3
```

```
## [1] 216  84 -27  36  33
```

I can also include "myvariable" in a function. So let's use our square root function again, ask for the square root of "myvariable", and again this function is going to be applied to every single item in that vector.

```
sqrt(myvariable)
```

```
## Warning in sqrt(myvariable): NaNs produced
```

```
## [1] 8.485281 5.291503      NaN 3.464102 3.316625
```

Now notice one thing here. Contained in "myvariable" vector is a negative number, and you can't take the square root of a negative. So what it got me was the square root of the other numbers that it was able to perform, and then this NaN came up for the negative 9. And that stands for not a number, meaning it can't do that function on a negative. And you also get a nice red warning basically alerting you to the fact that there was some number in here that the function was not able to compute.

Something that we can also deal with this object called "myvariable" is we can assign a modification of it to some other object. So just like we did here when we assigned x squared to y, we can assign maybe a new vector, "myvariable2", to be "myvariable" times 3.

```
myvariable2 <- myvariable*3
```

Again, if we submit it, we're not going to see the values of "myvariable2" because we didn't ask for R to show them. All we did was say assign this to the label "myvariable2." If we wanted to see what was in "myvariable2", we could either look over ... in our environment or we could just ask R to display it for us.

```
myvariable2
```

```
## [1] 216  84 -27  36  33
```

Some other really useful functions that we would want to use on vectors like this are the "**mean()**". So if I want to take the mean of "myvariable", I'm going to get a single value, which is literally the mean of all five of those values.

```
mean(myvariable)
```

```
## [1] 22.8
```

Another helpful function is if I want to know how many cases are contained in a vector. Here myvariable is pretty small. I can see that there are five. But if I had a vector that was very long, I might want to ask for the length of "myvariable."

```
length(myvariable)
```

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

And that'll just give me the number of cases that it contains. And finally, we've got the "**sum()**" function, which is just going to add up every number in that object. And I get 114.

```
sum(myvariable)
```

```
## [1] 114
```