Home

An intro to R for new programmers

This is an introduction to R. I promise this will be fun. Since you have never used a programming language before, or any for that matter, you won't be tainted by real progamming languages like `Python` or `Java`. This is good - we can teach you the R way of doing things.

jsforcats?

Yep, this is a total rip off of JSforcats.com.

What will we do?

- Using the R console - let's dig our claws in
- vector's - the basic R data structure
- data.frame's - weird but familiar
- lists
- functions
- Using packages
- Open data from the web! Cat's love open data
- Reading

R console

Writing code is fun. Since you're a cat, not having opposable thumbs may be a bit of an issue, but surely you're clever enough to find a way around that.

So open up R, and you'll see something like this:

You can do math:

```
1 + 1
```

```
## [1] 2
```

Type a set of letters together (also known as a *word*) within quotes and the console will print it back to you

```
"Hello Mr Tickles"
```

```
## [1] "Hello Mr Tickles"
```

Another thing you'll want to do as a cat using R is assign things to a name so that you can use it later. This is as if you were a chipmunk and you burried a nut in the ground to dig up later. You can assign anything in R to a name, then use it later (in the current R session of course :)).

Assign the number 5 to the name `mynumber`

```
mynumber <- 5
```

Later you can use `mynumber`, like adding it to another number

```
mynumber + 1
```

```
## [1] 6
```

Sweet!

Vectors

Vectors are one of the simplest and common objects in R. Think of a vector like a cat's tail. Some are short. Some are long. But they are are pretty much the same width - that is, they can only contain a single data type. So a vector can only have all `numeric`, all `character`, all `factor`, etc. But wait, how do we make a vector? The easiest way is to use a function simply called `c`. So `c(5,6,7)` will create a vector of numbers 5, 6, and 7: 5, 6, 7. Let's try to put a `character` and a `numeric` together.

```
c("hello", 5)
```

```
## [1] "hello" "5"
```

Notice how the output of the above converted the 5 to a character type with quotes around the `5` to make it `"5"`. But we can happily make a vector of the same type of information, like

```
c(5, 8, 200, 1, 1.5, 0.9)
```

```
## [1]   5.0   8.0 200.0   1.0   1.5   0.9
```

Data.frame's

A `data.frame` is one of the most commonly used objects in R. Just think of a `data.frame` like a table, or a spreadsheet, with rows and columns and numbers, text, etc. in the cells. A very special thing about the `data.frame` in R is that it can handle multiple types of data - that is, each column can have a different type. Like in the below table the first column is of `numeric` type, the second a `factor`, and the third `character`.

```
df <- data.frame(hey=c(5,6,7),
           there=as.factor(c("a","b","c")),
           fella=c("blue","brown","green"))

df
```

```
##   hey there fella
## 1   5     a  blue
## 2   6     b brown
## 3   7     c green
```

Notice that the first *column* of numbers are actually row names, and are not part of the `data.frame` *per se*, though are part of the *metadata* for the `data.frame`.

We can quickly get a sense for the type of data in the `df` object by using the function `str`, which gives inforomation on the types of data in each column.

```r
str(df)
```

```
## 'data.frame':    3 obs. of  3 variables:
##  $ hey  : num  5 6 7
##  $ there: Factor w/ 3 levels "a","b","c": 1 2 3
##  $ fella: Factor w/ 3 levels "blue","brown",..: 1 2 3
```

Lists

Lists are sorta crazy. They are kinda like vectors, but not. Using our cat tail anology again, lists are like cat tails in that they can be short or long, but they can also vary in width. That is, they can hold any type of object. Whereas vectors can only hold one type of object (only `character` for example), lists can hold for example, a data.frame and a number, or a data.frame and another list! The way we make a list is via the function `list`

```r
list(1, "a")
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
```

Functions

XXXX

Using packages

XXXX

Data from the web

Install `cowsay`

```r
install.packages("devtools")
library("devtools")
install_github("sckott/cowsay")
```

Now let's get a cat fact!

```r
library("cowsay")
say("catfact", "cat")
```

```
##
##
##  -----
##  Cats' hearing stops at 65 khz (kilohertz); humans' hearing stops at 20 khz.
```

```
##  ------
##     \
##      \
##                 \`*-.
##                  )  _`-.
##                 .  : `. .
##                 : _   '
##                 ; *` _.   `*-._
##                 `-.-'          `-.
##                   ;       `       `.
##                   :.       .       \
##                   .\  .   :   .-'   .
##                   '  `+.;  ;  '      :
##                   :  '  |    ;       ;-.
##                   ; '   : :`-:     _.`* ;
##                .*' /  .*' ; .*`- +'  `*'
##                `*-*   `*-*  `*-*'
##
```

A little explanation is in order me thinks. There are a few things going on in the last thing we just did. The `say` function looks like sorta like this:

```
say <- function(what, by, type){
  <== some ascii art ==>
  url <- "http://catfacts-api.appspot.com/api/facts?number=1"
  what <- fromJSON(url)$facts
  message(sprintf(by, what))
}
```

The first line is a bunch of ascii characters to make a cat. The second line defines the url for the cat facts API. The third line retrieves one cat fact from the cat facts API. And the fourth line prints the messages with a cat.

> But what is an API? I'm a cat, I only drink water or milk (preferably milk) - but at least I've heard of an IPA. What the rat's ass (yum) is an API.

Okay, here goes. An API stands for Application Programming Interface. It's just a set of instructions for two computers to talk to each other. It's sorta like if you run into another cat and if you both knew beforehand a lot about each other, you would have a sense for how to behave - if you don't know each other, then best of luck to you Mr. Tickles.

Reading

XXXX

Cat's love R

License