Win-Vector Blog

The Win-Vector LLC data science blog

Bend or break: strings in R

March 10, 2016 John Mount Programming R

A common complaint from new users of R is: the string processing notation is ugly.

- Using `paste(,,sep='')` to concatenate strings seems clumsy.
- You are never sure which regular expression dialect `grep()/gsub()` are really using.
- Remembering the difference between `length()` and `nchar()` is initially difficult.

As always things can be improved by using additional libraries (for example: stringr). But this always evokes Python's "There should be one– and preferably only one –obvious way to do it" or what I call the "rule 42" problem: "if it is the right way, why isn't it the first way?"

From "Alice's Adventures in Wonderland":

Alice's Adventures in Wonderland, drawn by John Tenniel.

> *At this moment the King, who had been for some time busily writing in his note-book, cackled out `Silence!' and read out from his book, `Rule Forty-two. All persons more than a mile high to leave the court.'*
>
> *Everybody looked at Alice.*
>
> *`I'm not a mile high,' said Alice.*
>
> *`You are,' said the King.*
>
> *`Nearly two miles high,' added the Queen.*
>
> *`Well, I shan't go, at any rate,' said Alice: `besides, that's not a regular rule: you invented it just now.'*
>
> *`It's the oldest rule in the book,' said the King.*

> *`Then it ought to be Number One,' said Alice.*

We will write a bit on evil ways that you should never actually use to try and weasel around the string concatenation notation issue in R.

If you read enough R documentation and resources you would think you could use one of the object oriented system (S3 or S4) to override the behavior of plus for strings. A bit more digging shows you can't override infix methods quite the same way you override named methods (R is a language where even the exceptions have exceptions), but it can be done using "Ops groups".

The below code re-purposes most of the binary operators for a user defined class called "`cats`" to be aliases for "`paste(,,sep="")`." This lets us concatenate objects of class "`cats`" using the "+" operator:

```
a = "a"
class(a) = "cats"
Ops.cats <- function(e1,e2) { paste(e1,e2,sep="")}
a + "b"
# [1] "ab"
```

The above code isn't that great because it doesn't work on the "`character`" class directly and doesn't inspect the magic "`.Generic`" variable to override only "+" (try `print(Ops.factor)` to see what a proper Ops method override looks like).

We don't pursue this approach much further as it doesn't work on strings/characters without the extra user supplied class declaration. Below is similar code failing to have any effect on R character types:

```
Ops.character <- function(e1,e2) { paste(e1,e2,sep="")}
"a" + "b"
# Error in "a" + "b" : non-numeric argument to binary operator
```

We can in fact re-define "+" to perform string concatenation by replacing the definition of "+" directly. This is shown below:

```
`+` <- function(e1,e2) {
  if (is.character(e1)) {
    paste(e1,as.character(e2),sep = "")
  } else {
```

```
    .Primitive("+")(e1,e2)
  }
}
"a"+"b"
# [1] "ab"
```

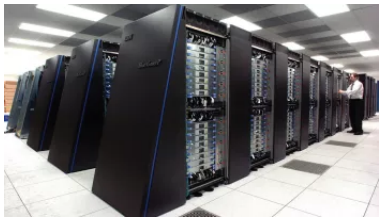
However, I do not recommend including this code in any serious R project. This code could slow down the interpreter greatly (as it is intercepting all references to "+") and could have unintended consequences if we haven't picked the exact right test in deciding when to trigger string concatenation. If we could have gotten the effect through the S3 object system we might have used it, but re-writing the "+" operator directly seems too dicey and could break things (especially if some other clever piece of code depends on re-defining "+"). All joking aside- this is why you don't commonly see R customized to allow string concatenation through "+": it could be done- but none of the polite ways can be used to accomplish it.

Our advice in using R is: learn to bend. If you are going to use R the effort to learn to work with it idiomatically is well rewarded (it is in fact quite powerful and expressive). But understand it is in fact different than other programming languages (such as Python).

Bonus question: what technique does `ggplot2` use to override "+"? It isn't obviously user visible through `methods("Ops")` (unlike our "`cats`" example) or through the operator definition (unlike magrittr's definition of "`%>%`" which is visible through `print(`%>%`)`).

RELATED

R has some sharp corners
R is definitely our first choice go-to analysis system. In our opinion you really shouldn't use something else until you have an articulated reason (be it a
In "Opinion"



A gentle introduction to parallel computing in R
In "Coding"

What can be in an R data.frame column?
As an R programmer have you every wondered what can be in a data.frame column? The documentation is a bit vague, help(data.frame) returns some
In "Opinion"

March 10, 2016 John Mount Programming R

3 thoughts on “Bend or break: strings in R”

Jeff Breiwick

March 10, 2016 at 7:45 pm

I find the R string tutorial at http://www.gastonsanchez.com to be very good and is useful for learning about the various R string-related functions.

Michael

March 11, 2016 at 10:49 am

The obvious thing to do is to check the codebase –

https://github.com/hadley/ggplot2/blob/master/R/plot-construction.r#L64

John Mount

March 11, 2016 at 11:14 am

Thanks!

Or we could also try more of R’s method inspectors:

```
library('ggplot2')
methods('+')
# [1] +.Date   +.gg*    +.POSIXt
```

(Notice the “gg”, one of the classes ggplot2 objects declare.)

That being said, it shows 3rd way to override operators (presumably through S3 classes), that unfortunately still does not apply to “`character`“.

```
`+.character` <- function(e1,e2) {paste(e1,e2)}
"a"+"b"
# Error in "a" + "b" : non-numeric argument to binary operator
```

Comments are closed.

Proudly powered by WordPress