Traps and flaws of the R language

Examples from

The R Inferno

by Patrick Burns

Guillaume Devailly

2015-11-18

The R inferno is:

- A free eBook: http://www.burns-stat.com/pages/Tutor/R_inferno.pdf
- Fun (if you are into R)
- Useful to avoid R common pitfalls

Here are some examples adapted from it.

Space matters

Problem: Code is hard to read

```
x<-3
x<--3
x<  -3
```

Space matters

Problem: Code is hard to read

```
x<-3
x<--3
x<  -3
```

Solution: Spaces, a lot of.

```
x  <-  -3
```

= or -> ?

Problem: = and -> are mostly synonymous,
unless when they are not.

= or -> ?

Problem: = and -> are mostly synonymous,
unless when they are not.

Solution: Use them wisely.

Falling into the Floating Point Trap

Problem: Computers are really bad at managing non-integer numbers.

Falling into the Floating Point Trap

Problem: Computers are really bad at managing non-integer numbers.

Solution: `all.equal?`

Failing to vectorize

Problem: Any problem here?

```
max(-1, 5, 118)
min(-1, 5, 118)
mean(-1, 5, 118)
```

Failing to vectorize

Problem: Any problem here?

```
max(-1, 5, 118)
min(-1, 5, 118)
mean(-1, 5, 118)
```

Solution:

```
max(c(-1, 5, 118))
min(c(-1, 5, 118))
mean(c(-1, 5, 118))
```

if else or ifelse?

Problem:

```
x <- seq(-1, 2, by = 1)
if(x < 1){
     y <- -1
} else{
     y <- 1
}
```

if else or ifelse?

Problem:

```
x <- seq(-1, 2, by = 1)
if(x < 1){
     y <- -1
} else{
     y <- 1
}
```

Solution:

```
y <- ifelse(
      x < 1,
      -1,
      1
)
```

Unexpected else in else

```
Error: unexpected 'else' in
"else“
```

While you may think that R is ludicrous for giving you such an error message, R thinks you are even more ludicrous for expecting what you did to work.

Not enough memory

`Error: cannot allocate vector of size 79,8 Mb`

This is often misinterpreted along the mines of: *"I have xxx gigabytes of memory, why can't R even allocate 80 megabytes?"* It is because R has already allocated a lot of memory successfully. The error message is about how much memory R was going after at the point where it failed.

The user who has seen this message logically asks, *"What can I do about it?"* There are some easy answers:

1- Don't be a glutton by using bad programming constructs.

2- Get a bigger computer.

3- Reduce the problem size.

read.table output is not a table

The `read.table` function does not create a table, it creates a data frame. You don't become a book just because you read a book. The `table` function returns a table.

sort.list not for lists

Do not be thinking that `sort.list` is to sort lists. You silly fool.

In fact sorting doesn't work on lists:

```
> sort(as.list(1:20))
Error in sort.int(x, na.last =
na.last, ...) :
'x' must be atomic
> sort.list(as.list(1:20))
Error                          in
sort.list(as.list(1:20)) : 'x'
must be atomic
Have you called 'sort' on a
list?
```

If you have lists that you want sorted in some way, you'll probably need to write your own function to do it.

Believing it does at intended

Problem: Precedence order may not be what you think it should.

Believing it does at intended

Problem: Precedence order may not be what you think it should.

Solution: Make precedence order (more) explicit.

== NA does not work

(the way I wanted it to work)

Problem: == NA is not helpful in finding NAs.

`==` NA does not work

(the way I wanted it to work)

Problem: `==` NA is not helpful in finding NAs.

Solution: `is.na()`

Multiple testing

Problem:

```
myVect <- 1:5
myVect == 4|6
```

Multiple testing

Problem:

```
myVect <- 1:5
myVect == 4|6
```

You're drunk R,
everything is not TRUE.

Multiple testing

Problem:

```
myVect <- 1:5
myVect == 4|6
```

You're drunk R,
everything is not TRUE .

Solution: Don't code when you're wasted.

Numeric to factor, accidentally

When using `read.table` or its friends, it is all too common for a column of data that is meant to be numeric to be read as a factor. This happens if `na.strings` is not properly set, if there is a bogus entry in the column, and probably many other circumstances.

This is dynamite.

Numeric to factor, accidentally

When using `read.table` or its friends, it is all too common for a column of data that is meant to be numeric to be read as a factor. This happens if `na.strings` is not properly set, if there is a bogus entry in the column, and probably many other circumstances.

This is dynamite.

Solution:

```
options(stringsAsFactors = FALSE)
```

At the beginning of EVERY scripts.

… or deals with it.

List subseting

Problem: Many ways to subset a list.

List subseting

Problem: Many ways to subset a list.

Solution: Do it the right way.

Combining lists

Problem: c( )works on lists!

Combining lists

Problem: c( )works on lists!

Solution: Be aware of the unexpected twist!

Coercion

Problem:

```
50 < “7”
```

Coercion

Problem:

```
50 < “7”
```

Solution:

```
50 < as.numeric(“7”)
```

`seq()` and `sample()`

Problem: They may return unexpected results.

```
seq(0:10)
sample(c(5.2), 9,
       replace = TRUE)
```

`seq()` and `sample()`

Problem: They may return unexpected results.

```
seq(0:10)
sample(c(5.2), 9,
       replace = TRUE)
```

Solution: Be aware of the unexpected twist!

Matrices in data frames

Matrices in data frames

You will surely think that allowing a data frame to have components with more than one column is an abomination.

That will be your thinking unless, of course, you've had occasion to see it being useful. The feature is worth the possible confusion, but perhaps a change to printing could reduce confusion.

Failing to use `drop = FALSE`

Problem: Writing functions is hard because of all those nasty special cases.

Failing to use `drop = FALSE`

Problem: Writing functions is hard because of all those nasty special cases.

Solution: Learn good coding practice.

Unreserved

Problem:

I screwed up my R environment.
Nothing works anymore.

Solution:

Do not create a T or a F variable.
Use TRUE and FLASE.
Do not create a c(), a t() ore a return() function.
Rescue your session with rm()

Quotes

Negative something is something

Subseting when names are not unique

Triping on Object Orientation

Once upon a time a new user was appropriately inquisitive and wanted to know how the median function worked. So, logically, the new user types the function name to see it:

```
> median
function (x, na.rm = FALSE)
UseMethod("median")
<environment: namespace:stats>
```

The new user then asks, *"How can I find the code for median?"*

The answer is, *"You have found the code for median."* median is a generic function as evidenced by the appearance of `UseMethod`.

What the new user meant to ask was, *"How can I find the default method for median?"*

The most sure-re way of getting the method is to use `getS3method("median", "default")`