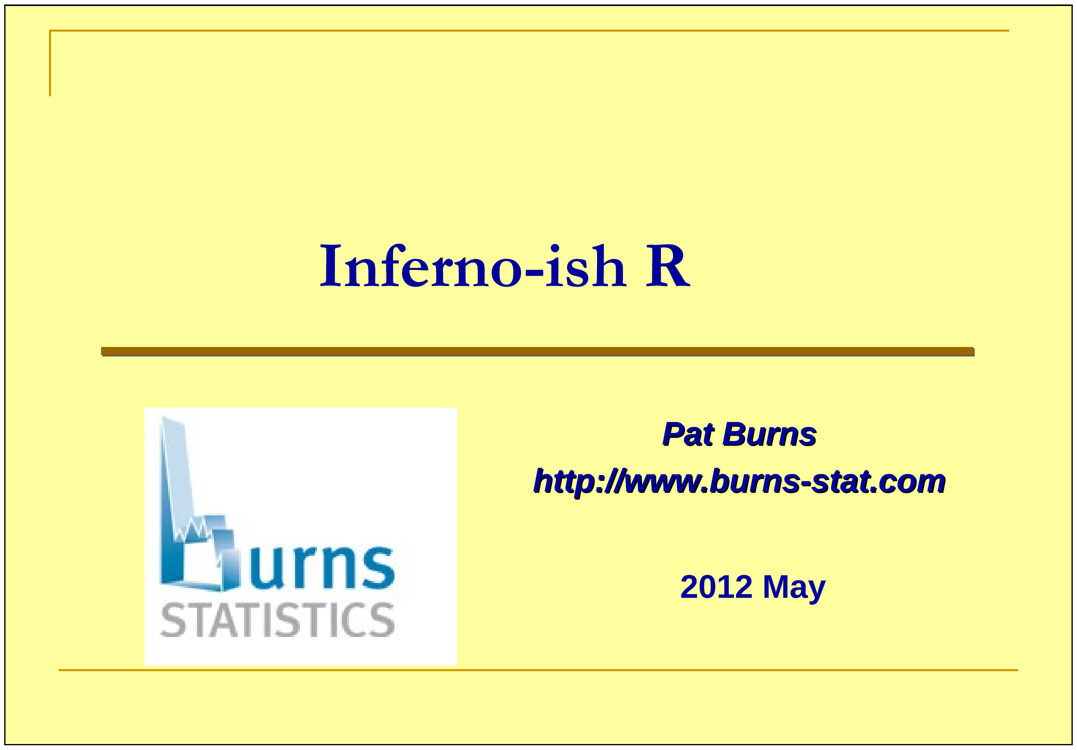


Inferno-ish R

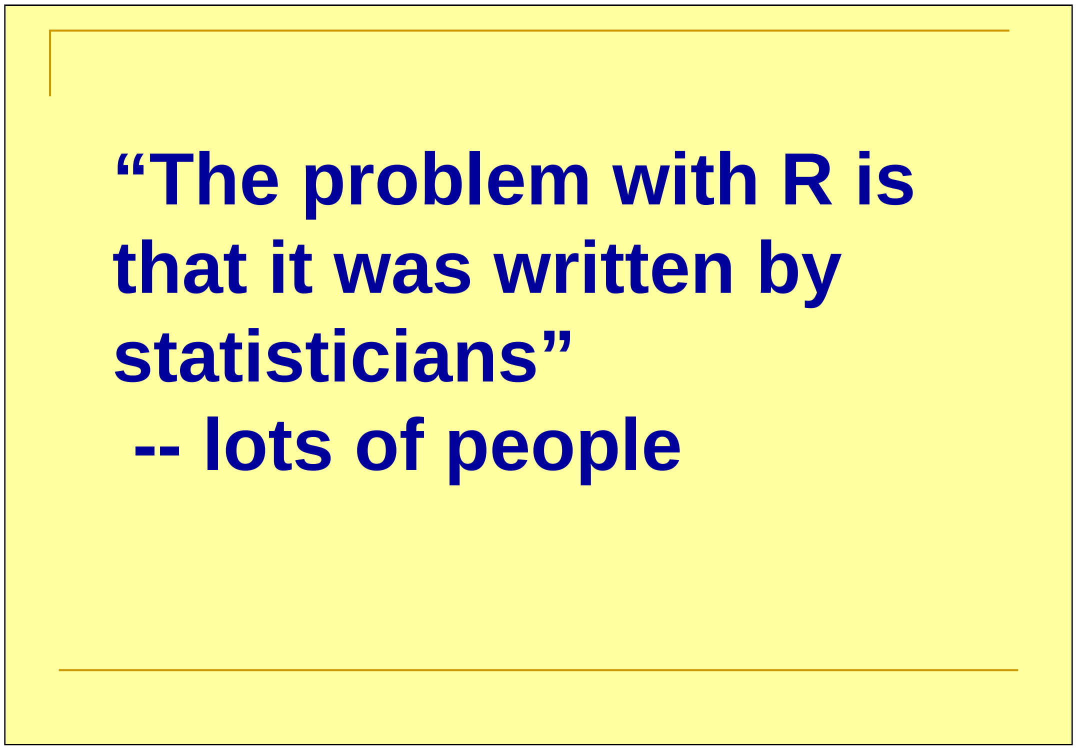
Pat Burns
http://www.burns-stat.com

2012 May

Talk given 2012 May 29 at CambR in Cambridge UK.

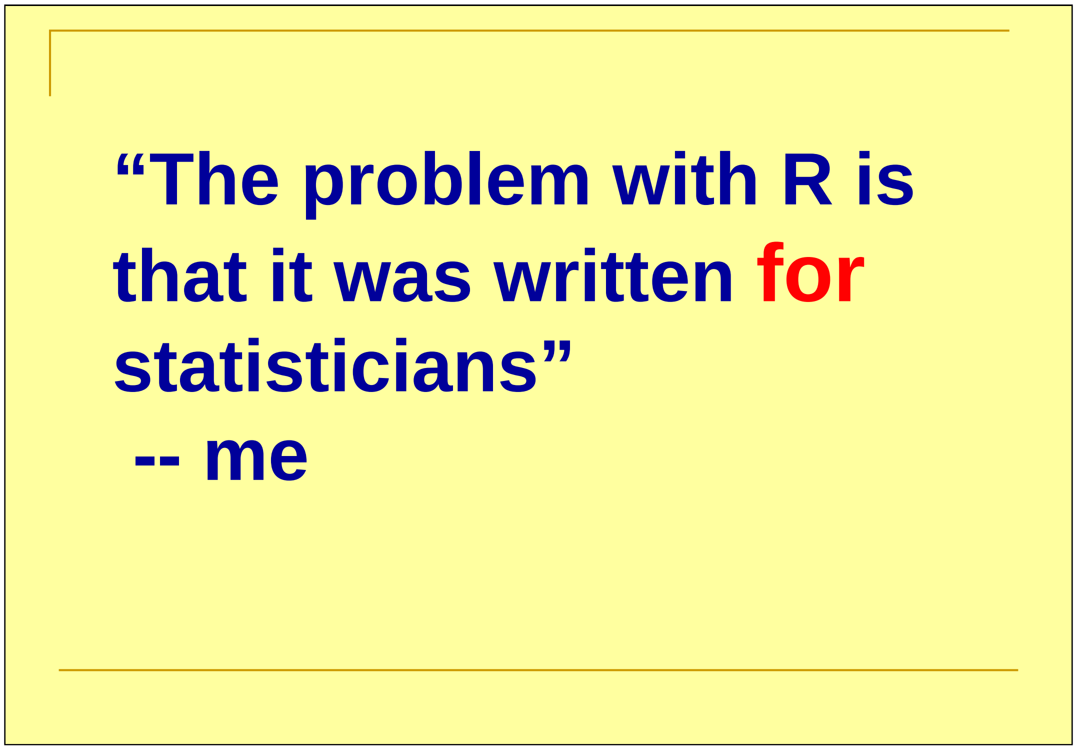
The final scene (that’s a pun) from the movie “Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb”.

Image taken from: en.wikipedia.org/wiki/File:Dr._Strangelove_-_Riding_the_Bomb.png


“The problem with R is that it was written by statisticians”
-- lots of people

This is an R user’s group, so the party line is that R is the most wonderful thing ever created. If that is true, then there’s a lot wrong in the world.

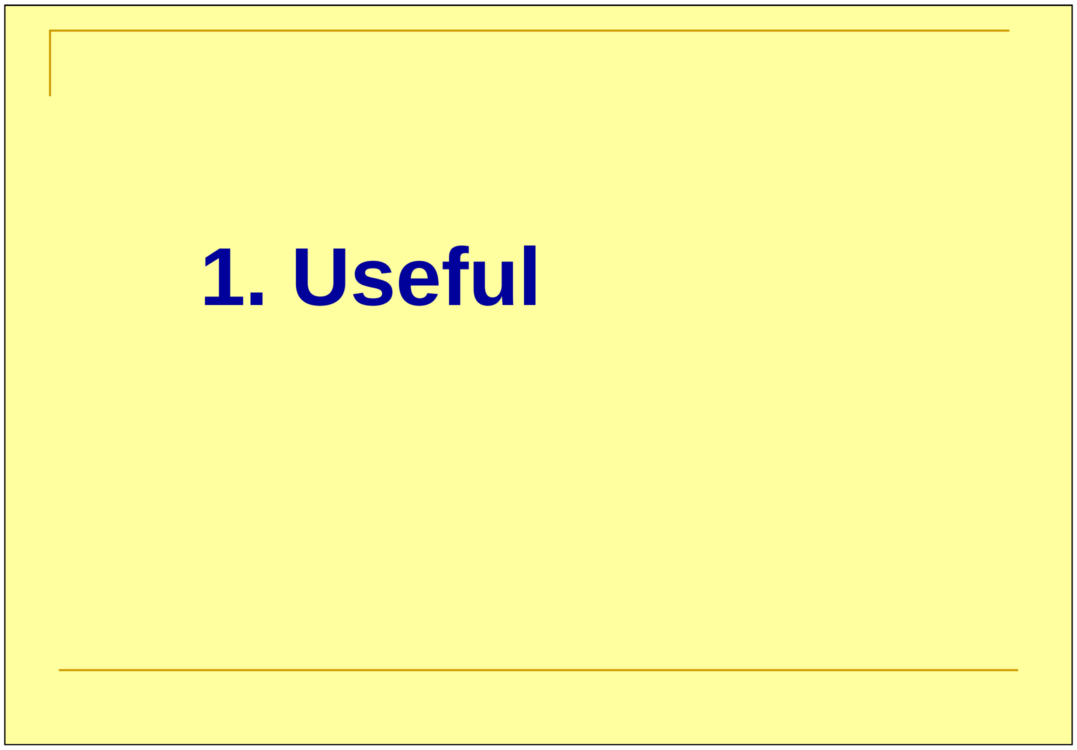
A lot of people have stated that the problem is that the language was created by statisticians. I agree there is a problem, but I don’t think that properly identifies the cause.



“The problem with R is that it was written for statisticians”

-- me

The aim of the talk is to make sense of this statement.



1. Useful

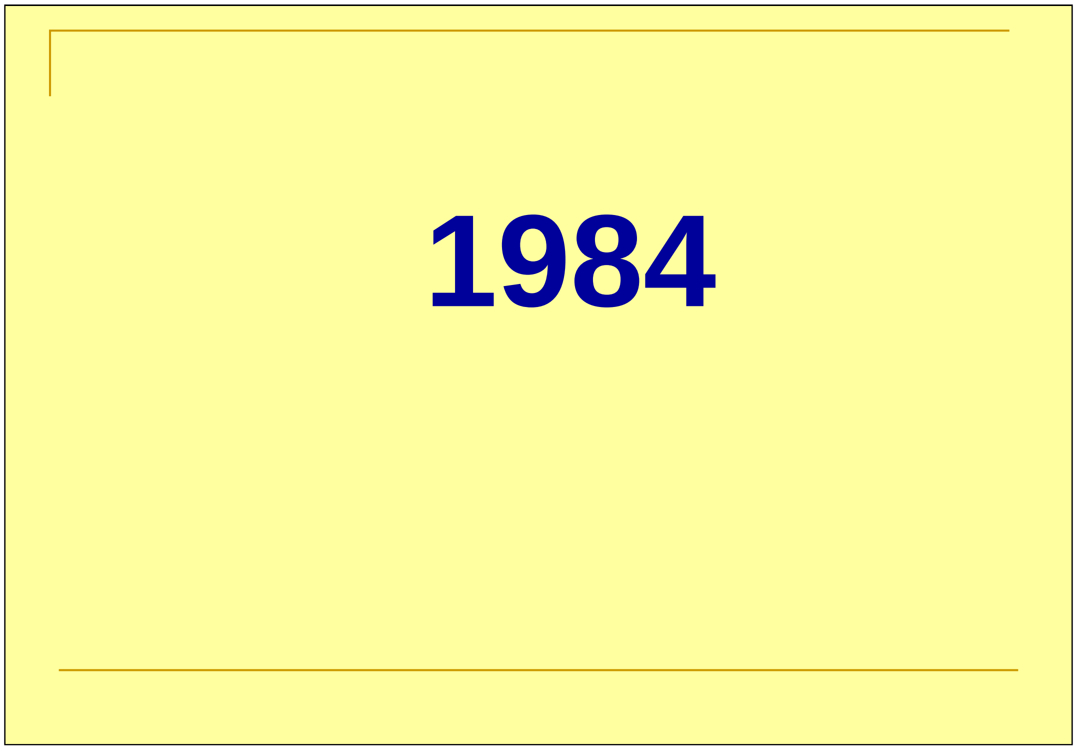
I see three drivers of problems with R.

The first is that R is useful.

This might seem like a strange thing to be problematic. I assure you it is, and I’ll tell you a story to try to convince you.

The story starts at Bell Labs with a research group. The project that the group had was to investigate the best computing environment for doing data analysis. The group had an experiment. The experiment eventually came to be called “S”.

Then something momentous happened.



1984

S escaped into the world.

Bell Labs sold tapes of the source code. It was a nominal fee for universities, and a non-exorbitant fee for commercial entities.

When you got the tape, it was up to you to try to figure out how to make it go on your hardware.

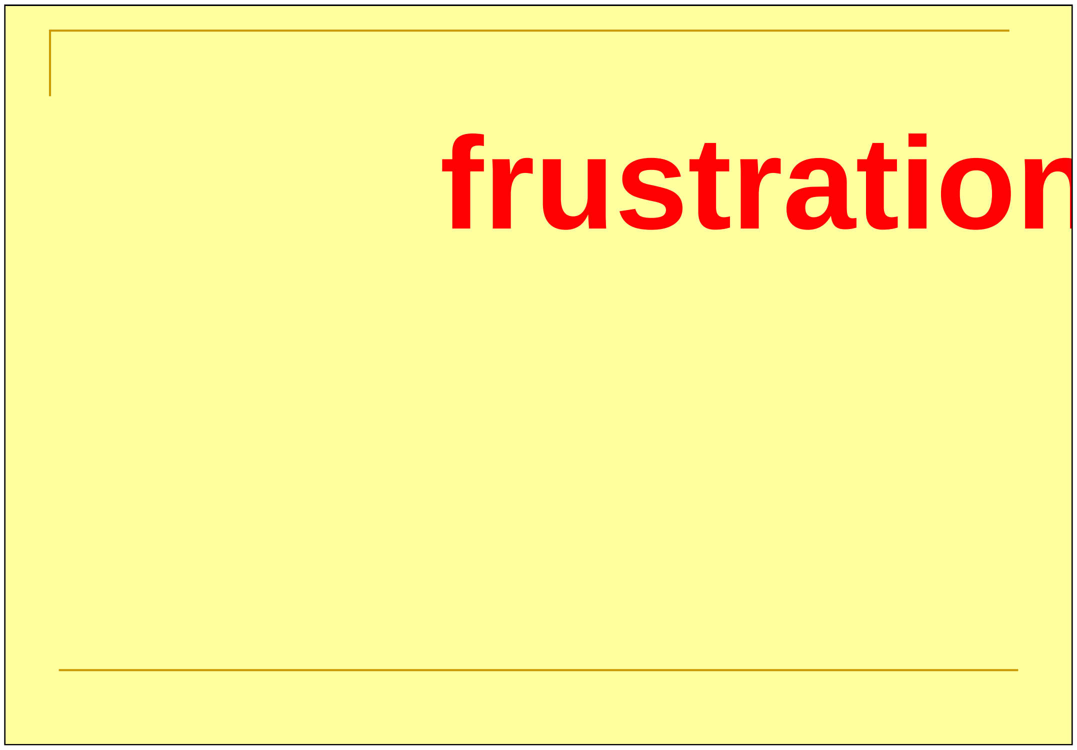
Photo by cliff1066 via everystockphoto.com

(In case you can't tell, this is a wax statue and not the real guy.)

This was the year that a mediocre actor who already had Alzheimer's would be re-elected to preside over a big pile of nuclear weapons. A very Strangelovian event.

It was also the year that I started graduate school in one of those universities that bought the S tape.

A few weeks before that election I was introduced to S, and my first reaction to it was …



frustration

I had preconceived ideas about how S should operate. S on the other hand was very insistent that it would operate how it operated.

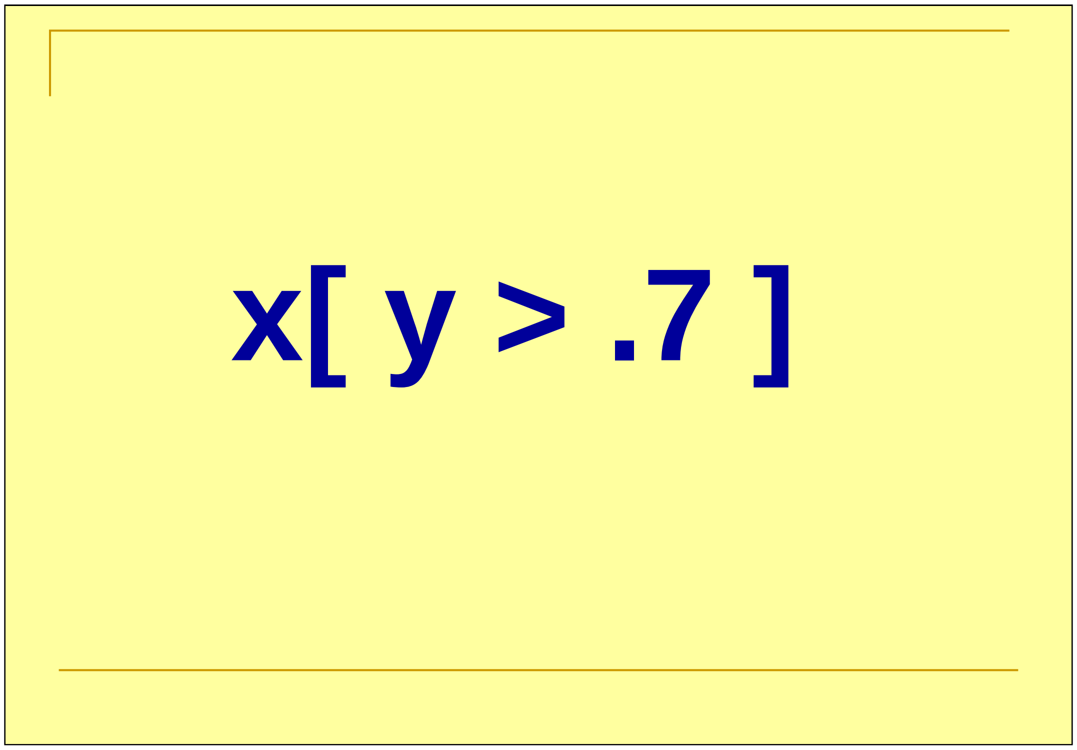
That is probably a very common reaction, both to S back then and to R now.

If you run into such frustration, my prescription is:

1. Start breathing again
2. Think to yourself: given that I'm frustrated, some of my ideas must be wrong; so let me start dismantling my assumptions

My preconceptions came from having used ISP, which was similar to S and had a fairly similar level of functionality at that point.

A little about ISP is in arxiv.org/pdf/0808.0777.pdf



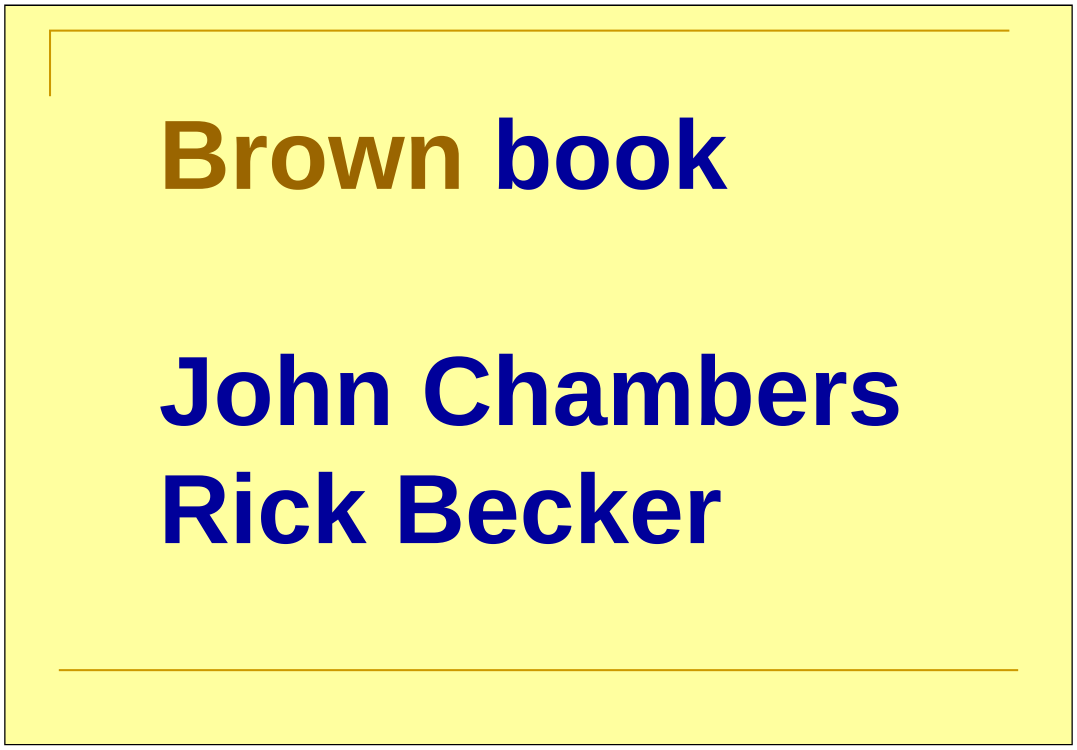
```
x[ y > .7 ]
```

I did eventually fall in love with S. I was sitting in the second or third chair in the statistics computer room (Padelford C-301). The command that made me fall in love was akin to this statement. There were two things about it that struck me.

I didn't need to do this in two lines – I could combine operations in one go.

Even more striking: y was not x. I could subscript x based on stuff other than x.

For some reason this opened up my mind to possibilities.



Brown book

John Chambers
Rick Becker

S was not only software, it was also a book.

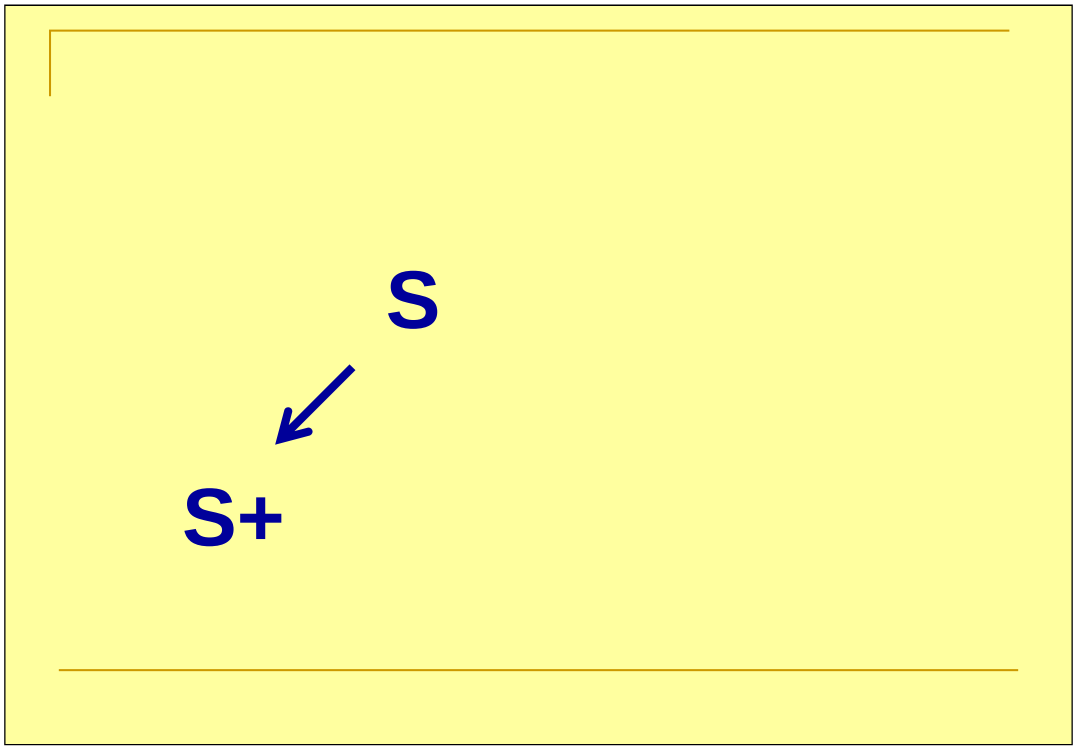
No one knew it at the time, but S would come to be color coded by the covers of the books about it.

S code from this era would be recognizable as a relative of R, but is clearly not R. In particular there were macros. They had the role that functions now perform. Macros were really, really ugly – lots of question marks and capital T's.

It was also possible to add S functions, the closest analogy is probably writing code that uses '.Call'. Except that the underlying code was Fortran, and the success of compiling had something to do with the height of the tide and phases of the moon.

In terms of research, there were two groups of people in the computer room. There was the group that were yelling in frustration and bloodying their fists as they tried to create S functions. There was the group that were laughing and having fun playing on the Lisp machines (they were Symbolics machines).

I had two officemates through most of graduate school – Jeff Banfield and Robert Gentleman. All three of us saw fit to join the playing-with-Lisp crowd.



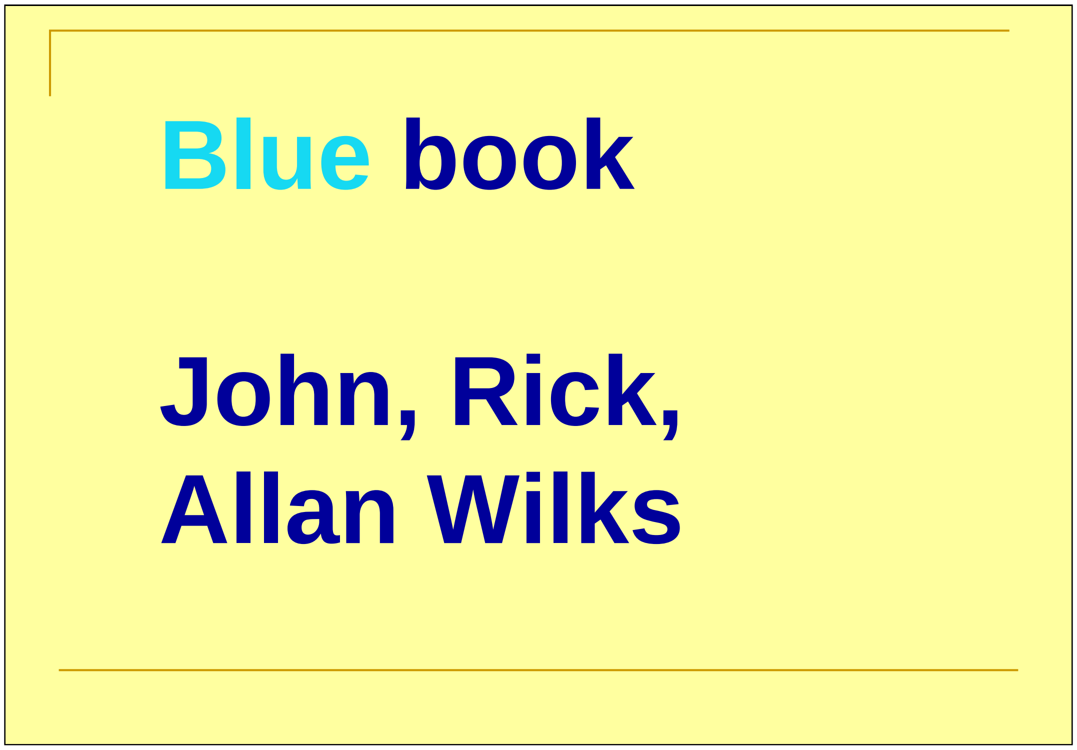
In about 1987 S-Plus was born. This was a commercial product.

Instead of buying a tape of the source code of S, you could buy a tape of S-Plus that was compiled for your machine. Assuming you had the right machine. The initial right machine was a Sun 3.

Perhaps you see the storm clouds starting to gather – we have a commercial product that is underpinned by an experiment.

1988

Another US presidential election year.

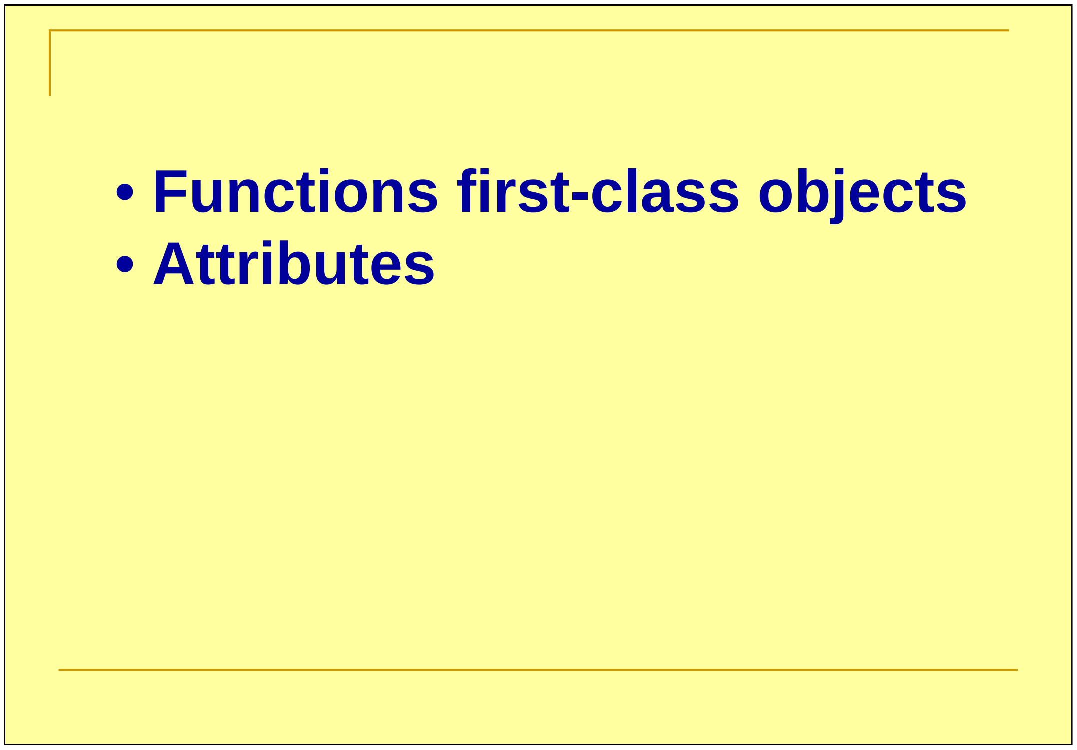


Blue book

John, Rick,
Allan Wilks

Another book about S.

But we still hadn't learned about color coding. It was "old S" and "new S".



- Functions first-class objects
- Attributes

In this version functions become first-class objects, just as they are now in R. No more macros.

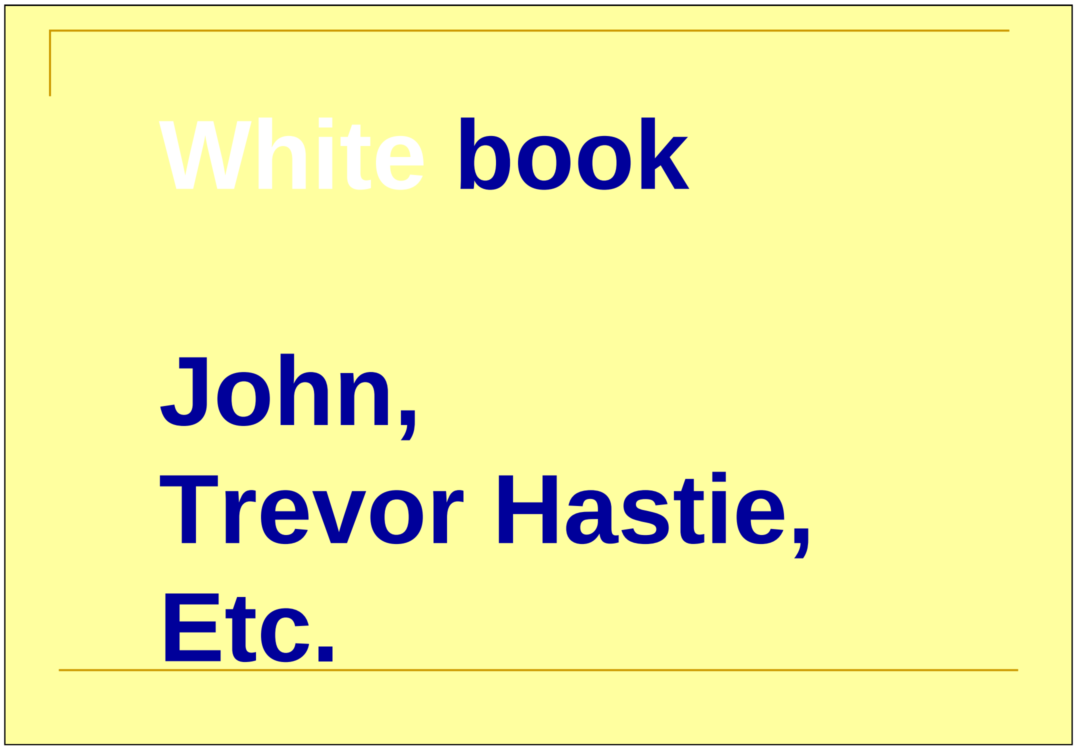
Also attributes were born – I think this was a stroke of genius.

No one complained about giving up old S for new S. Show new S to someone who had old S, and they switched in a heartbeat.

This was heaven.

1992

A US election year.

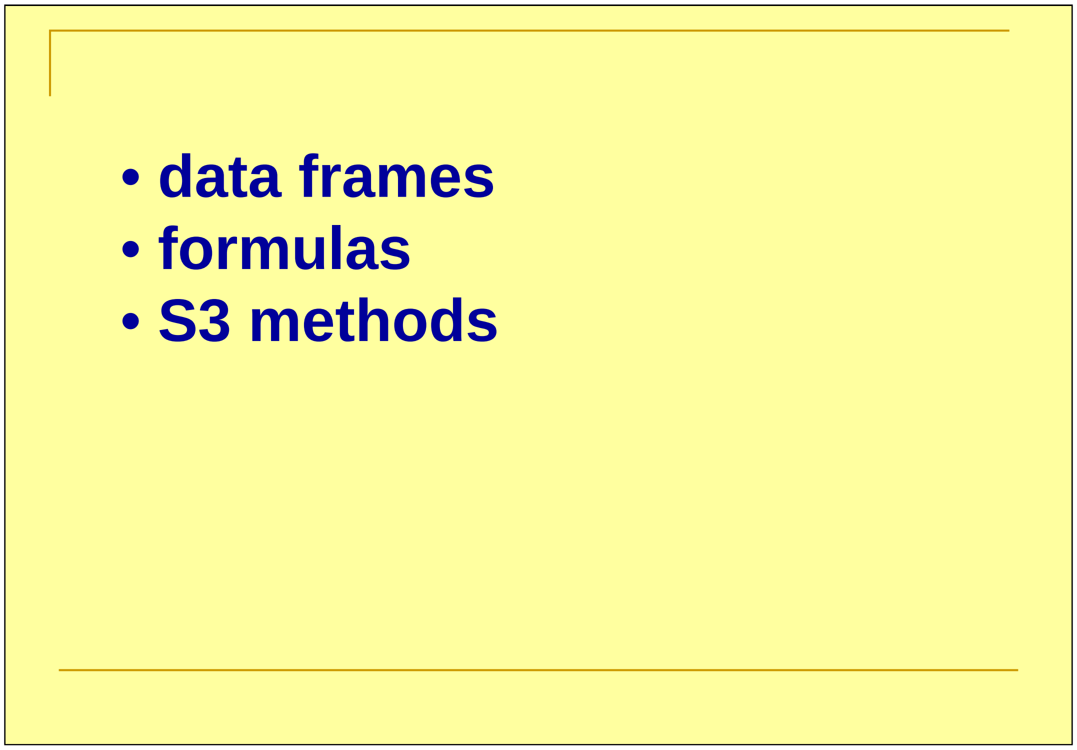


White book

John,
Trevor Hastie,
Etc.

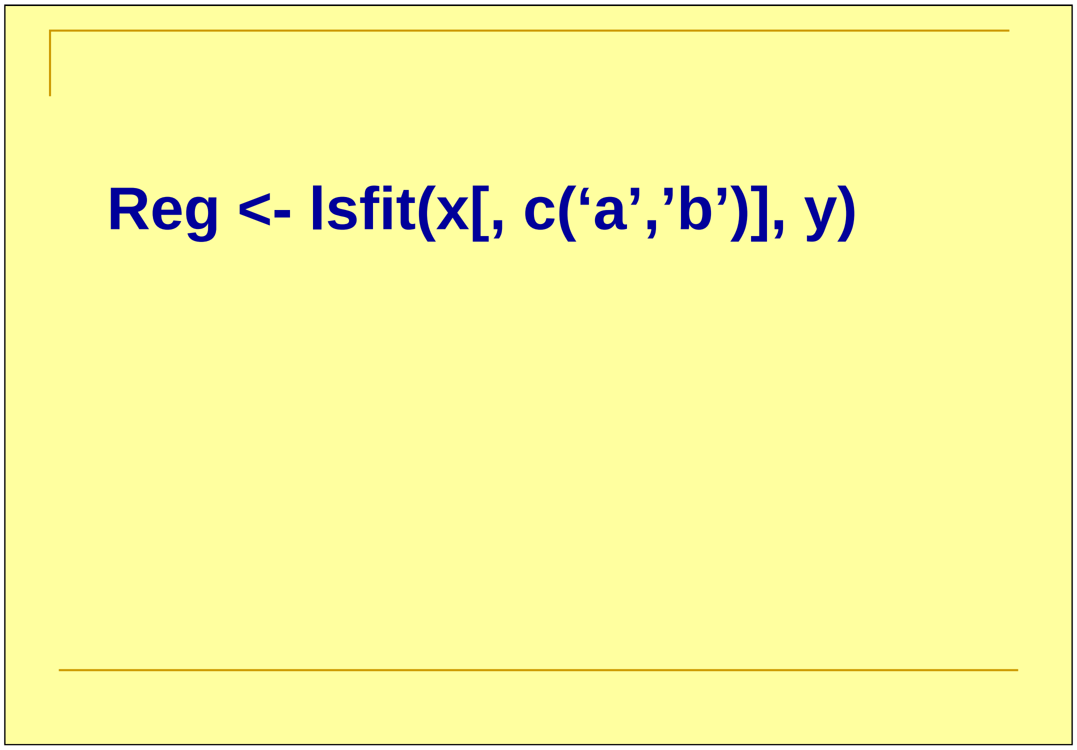
A book about S.

Now we have color coding. And we have trouble.



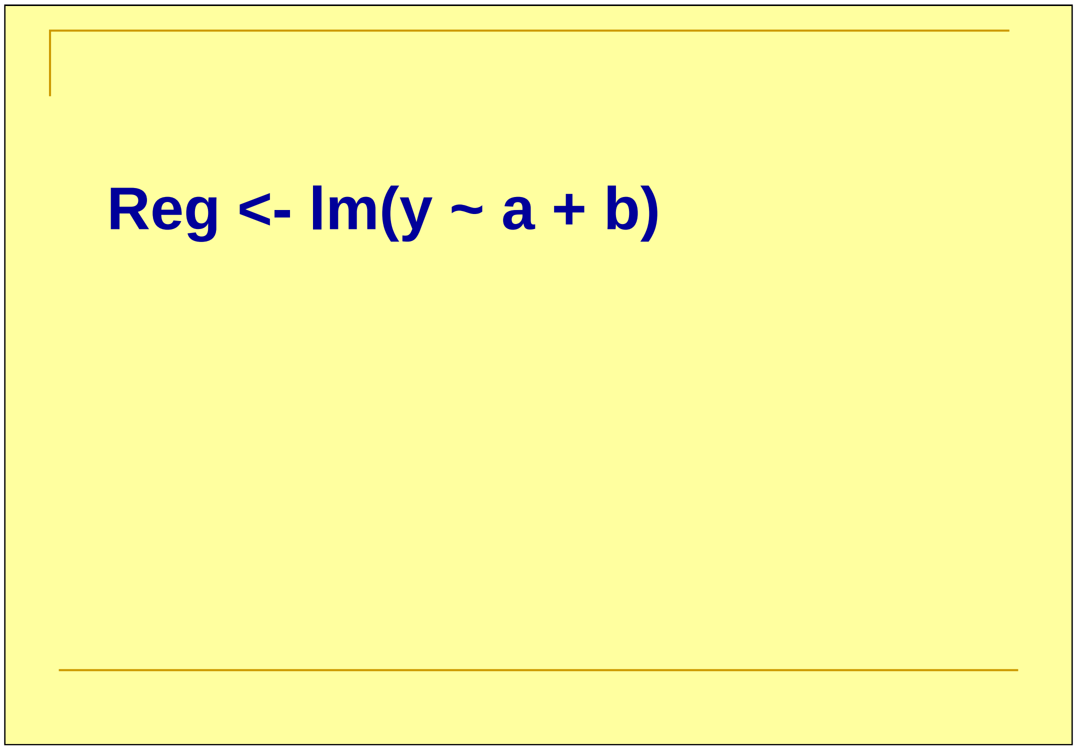
- data frames
- formulas
- S3 methods

In this version we gain: data frames, formulas and what is now known as S3 methods. They were just “methods” at the time.



```
Reg <- lsfit(x[, c(‘a’,’b’)], y)
```

The way to do linear regression in the Brown and Blue books was to use ‘lsfit’ and give it a matrix of explanatory variables.



```
Reg <- lm(y ~ a + b)
```

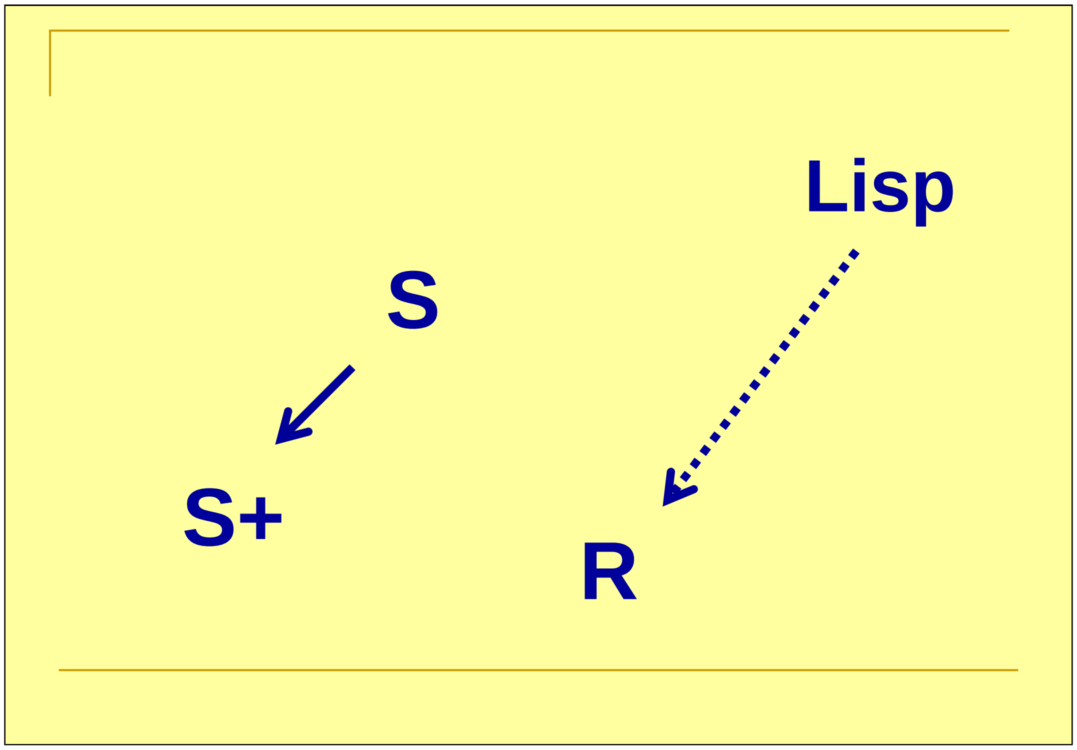
The way to do linear regression in the White book is to use ‘lm’ and give it a formula.

So we have two completely different ways to do exactly the same thing. What an excellent way to confuse the hell out of users.

People were not going to throw away the code that they had built over the prior four years just so they could say ‘lm’ instead of ‘lsfit’. (Besides the new stuff was not without bugs at the time.) So the two methodologies continued to live side by side.

Now, I’m supposed to be telling you why R is in such a state. What does this have to do with R?

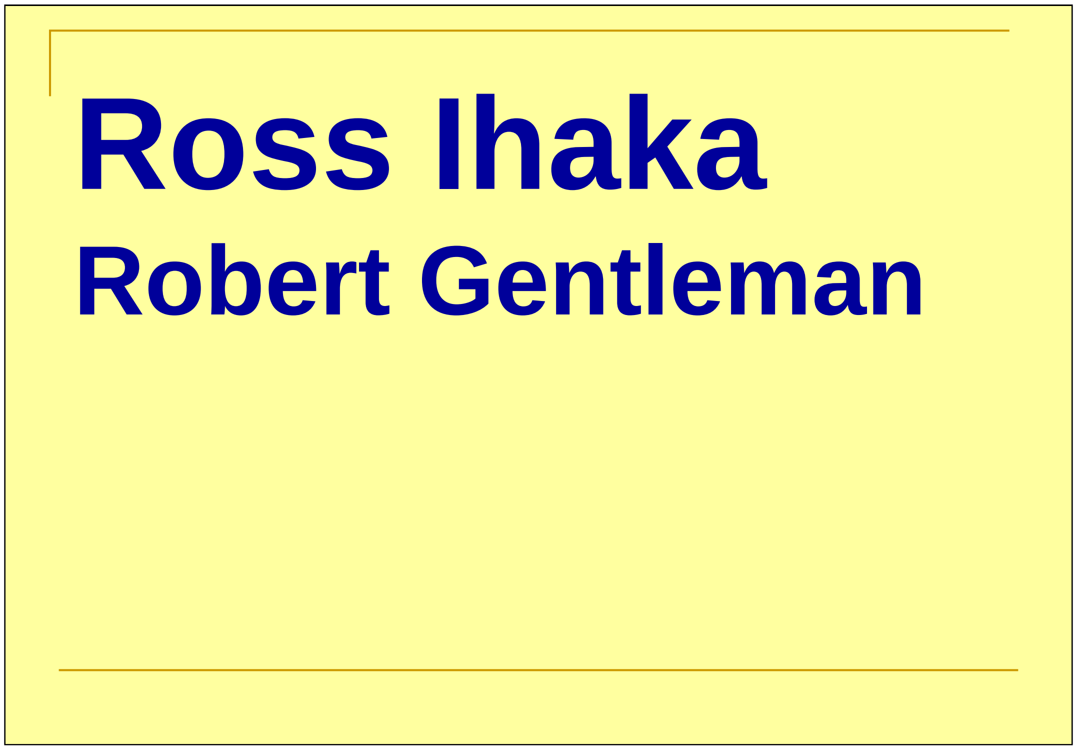
Nothing. Yet.



R was born as Lisp with some syntactic sugar on top so that it would look like S.

It didn't have to be precisely S. In fact, part of the point was to be different from S. S was an experiment. An experiment means that you don't know what you're doing. S had made mistakes.

R could avoid at least some of the mistakes that S made.



Ross Ihaka

Robert Gentleman

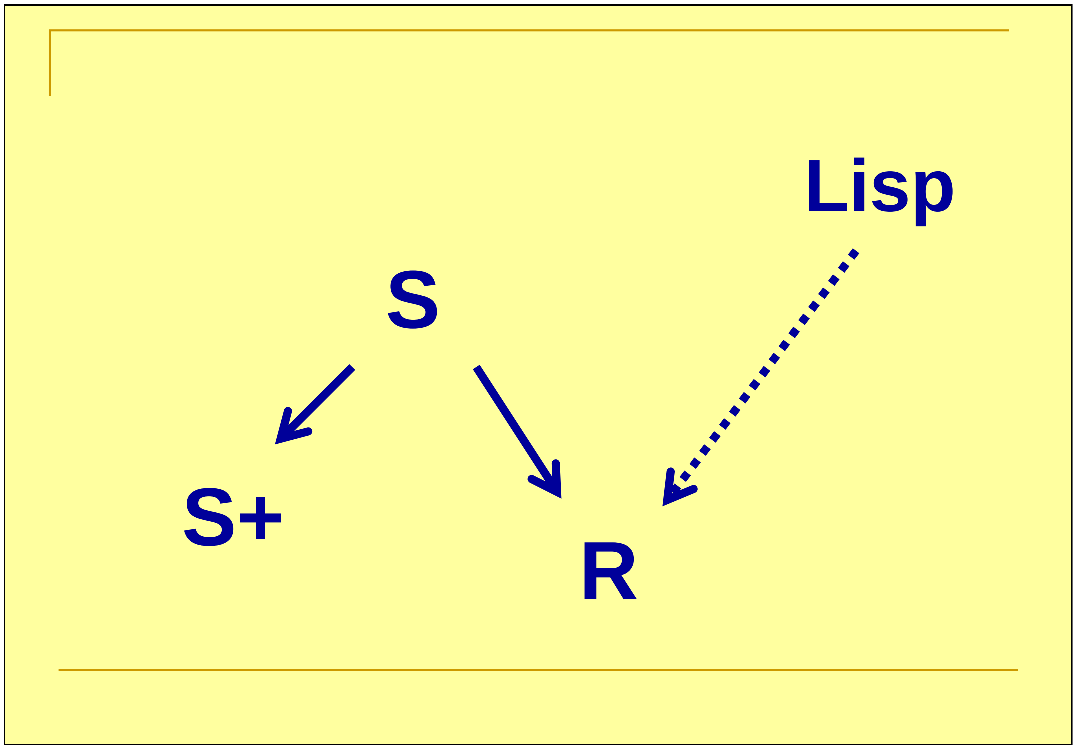
R was made by Ross Ihaka and Robert Gentleman.

And here they are as they were then: Robert on the left, Ross on the right.

As long as R stayed in Ross and Robert’s computer lab, R could be perfectly clean and beautiful.

But R started to escape out into the world, as S had before it.

Photo courtesy of Ross Ihaka.

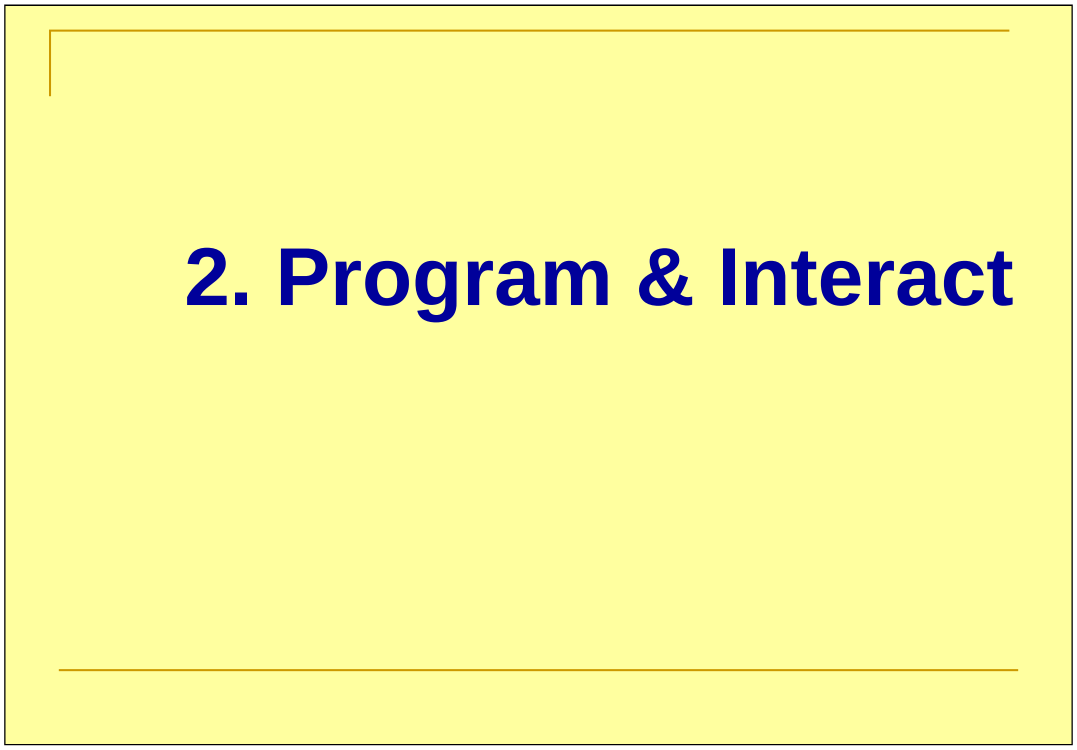


For the most part the new users gathering around R were not people with a clean slate. They were people who didn't want to pay for S-Plus any more. If you know anything about academic budgets, you will know who a whole lot of those initial users were.

These were users with S code that wanted their S code to work in R.

R got sucked into the vortex of S.

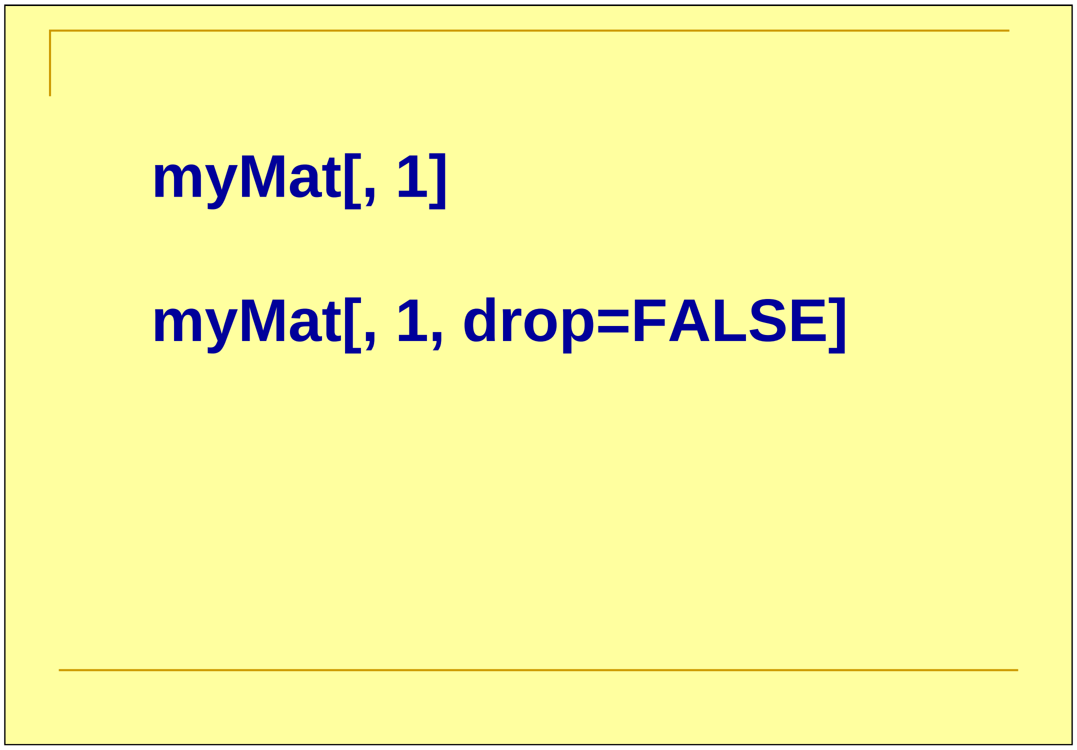
The moral of the story is that if you want to create a beautiful language, for god's sake don't make it useful.



2. Program & Interact

The second driver of problems with R is that it is both a programming language and an interactive language.

There is a tension there that is unavoidable.



```
myMat[, 1]

myMat[, 1, drop=FALSE]
```

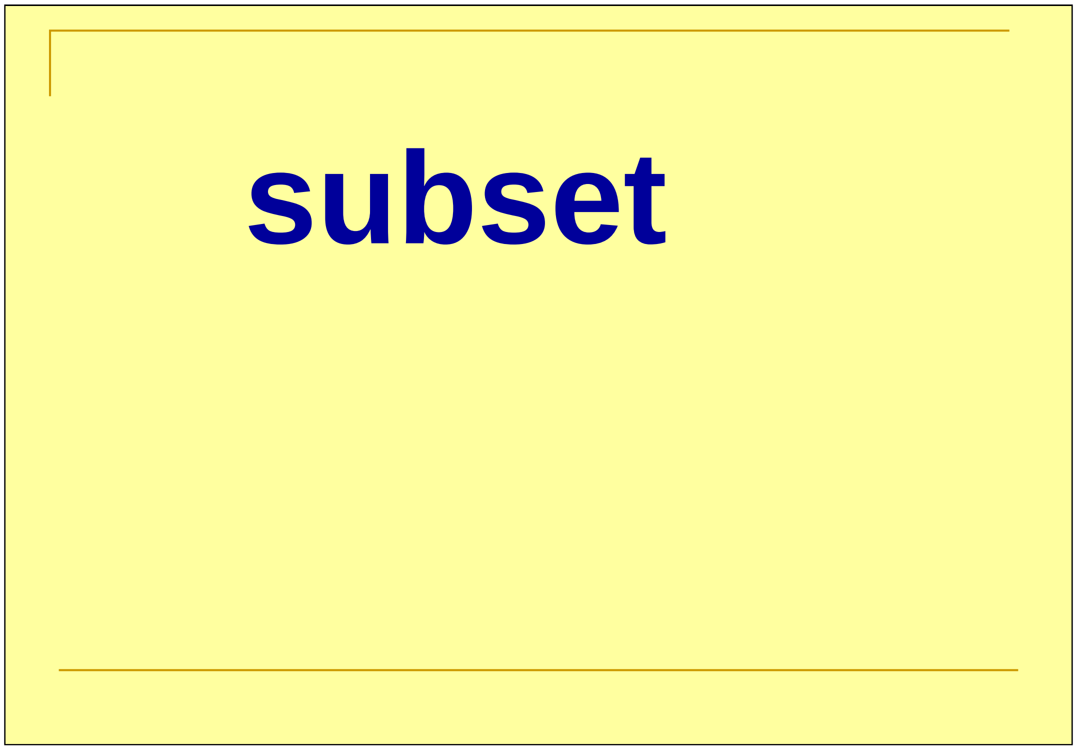
If you are working interactively and you want the first column of a matrix, then almost always the right thing is that the result is a plain vector – the matrixness is lost.

If the exact same command is inside a function, then that is fine also. But more likely it will be a variable and not a number if the command is inside a function.

If that variable is always of length one, there is still no problem.

If the variable is generally of length greater than one so a matrix is expected, then there is trouble. If the length of the variable is one, then the result will be a vector and not a matrix, and the function is likely to fail.

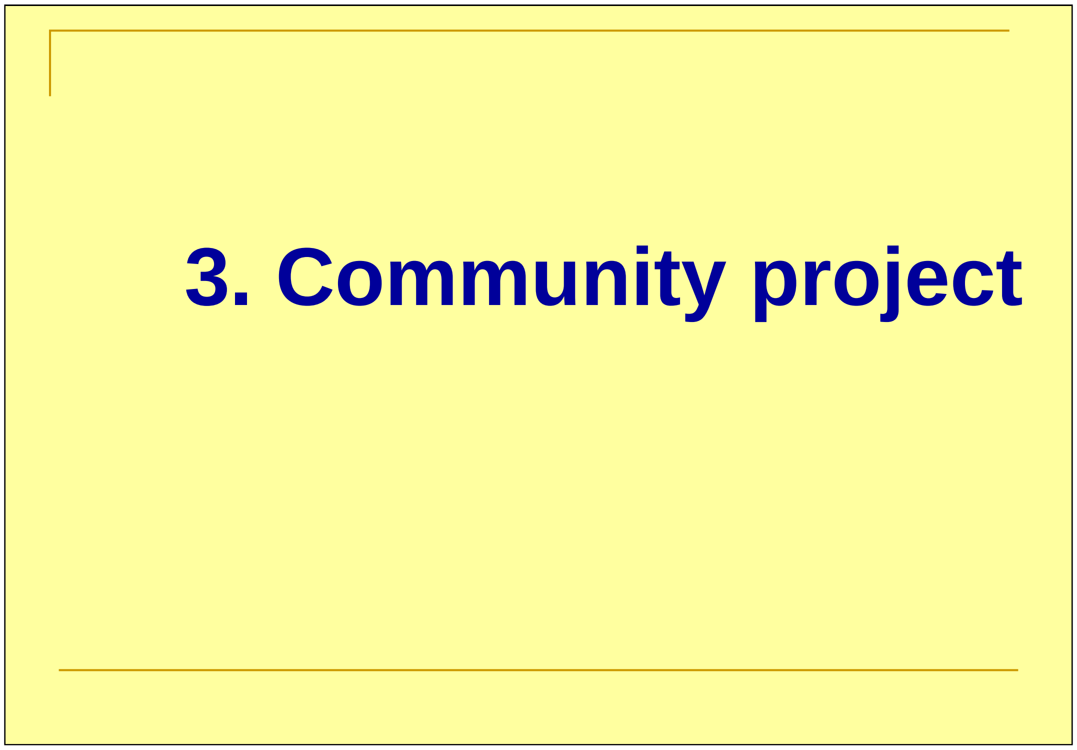
The right default for interactive use is ‘drop=TRUE’, the right default for programming is ‘drop=FALSE’.



subset

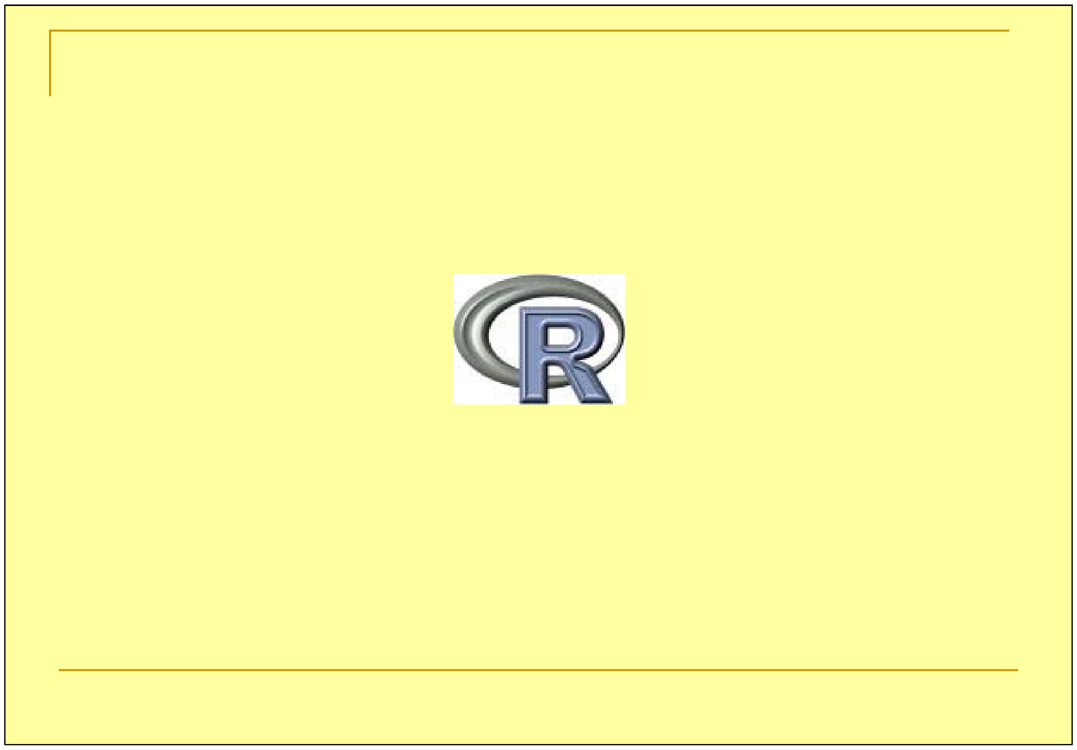
There are a number of functions that break the usual syntax rules to be more friendly for interactive use. The ‘subset’ function is one example.

These functions can be very nice for interactive use, but they will generally break when used inside a function.

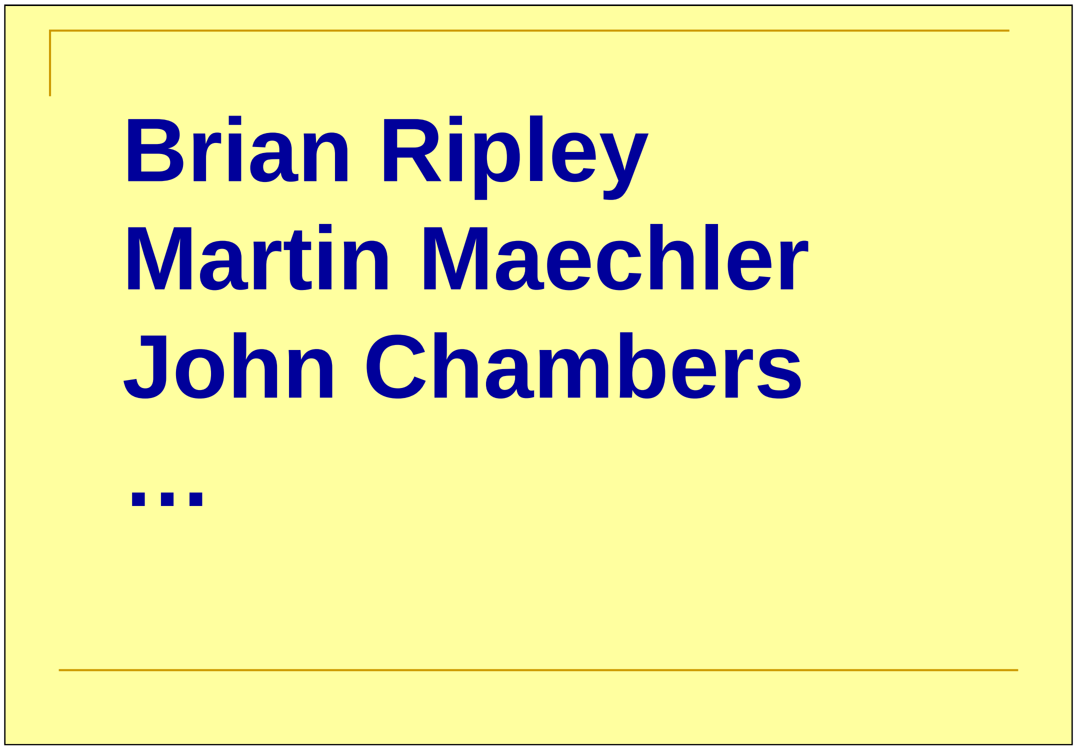


3. Community project

The third big driver of problems is that R is not software, it is a community.



There is R in the small sense of R Core.


Brian Ripley
Martin Maechler
John Chambers
…

People like these:

Brian Ripley has worked an incredible amount on R. See, for instance: www.r-bloggers.com/celebrating-r-commit-50000/

If Ross and Robert were the parents of R, then Martin Maechler was the midwife. As far as I know he was very influential in R being born to the world.

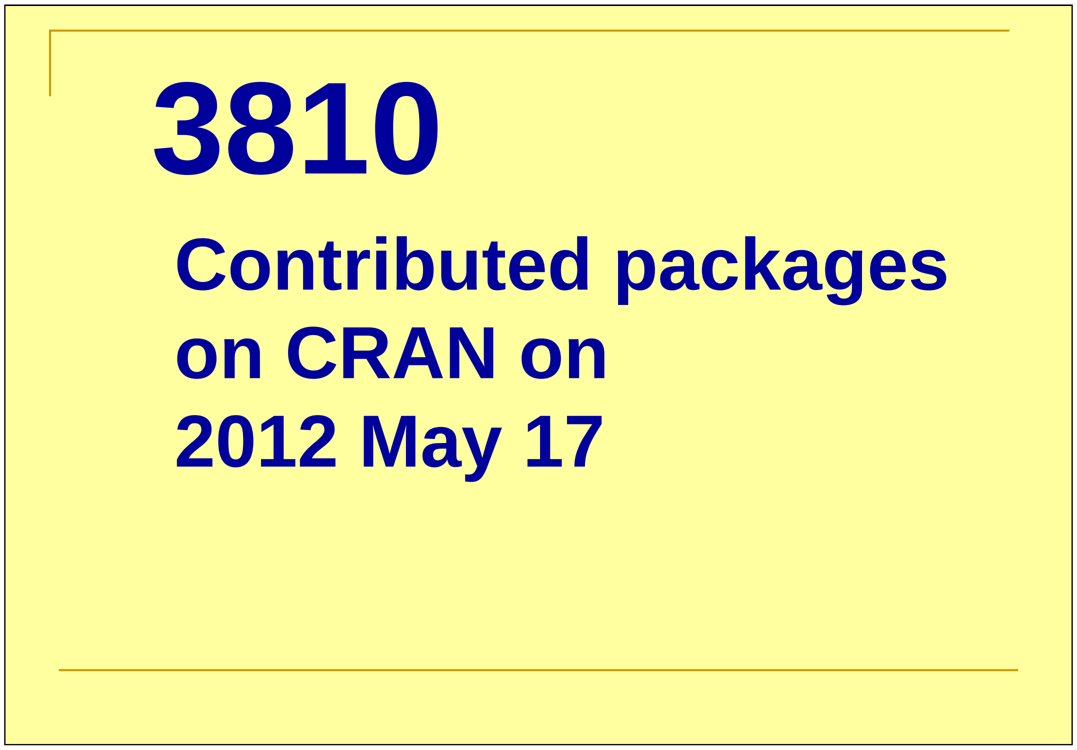
There are a lot of people in the world named John Chambers. But this is in fact the same John Chambers whose name is on all of the S books. That John is a member of R core is one of the most interesting statements about R that I know of.

This selection is by no means meant to denigrate the other members of R Core. I failed in my attempt to find a list of R core members.

When you download a new version of R, you are downloading code that is under the control of R Core.

That code is problematic because of the things we’ve seen, but it is exceptionally clean code. I think the quality of the core R packages is extremely high. And I’m picky.

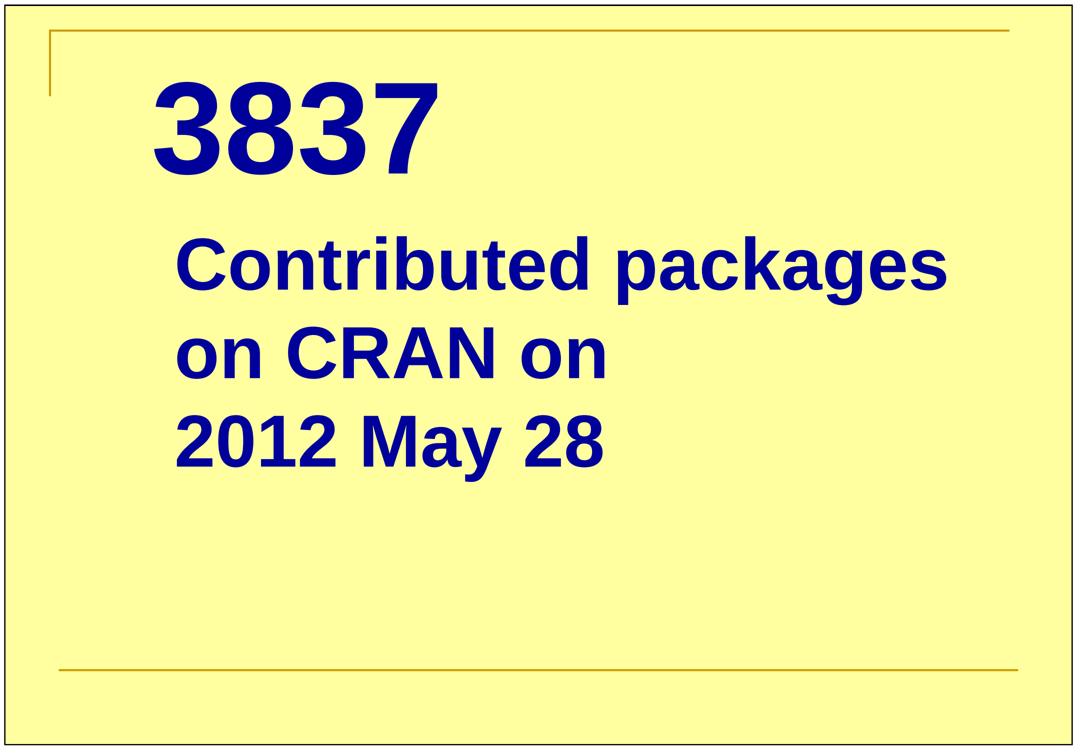
But there is also R in the bigger sense of the R community.



3810

Contributed packages on CRAN on 2012 May 17

One of the great advances of R over S was the package system.



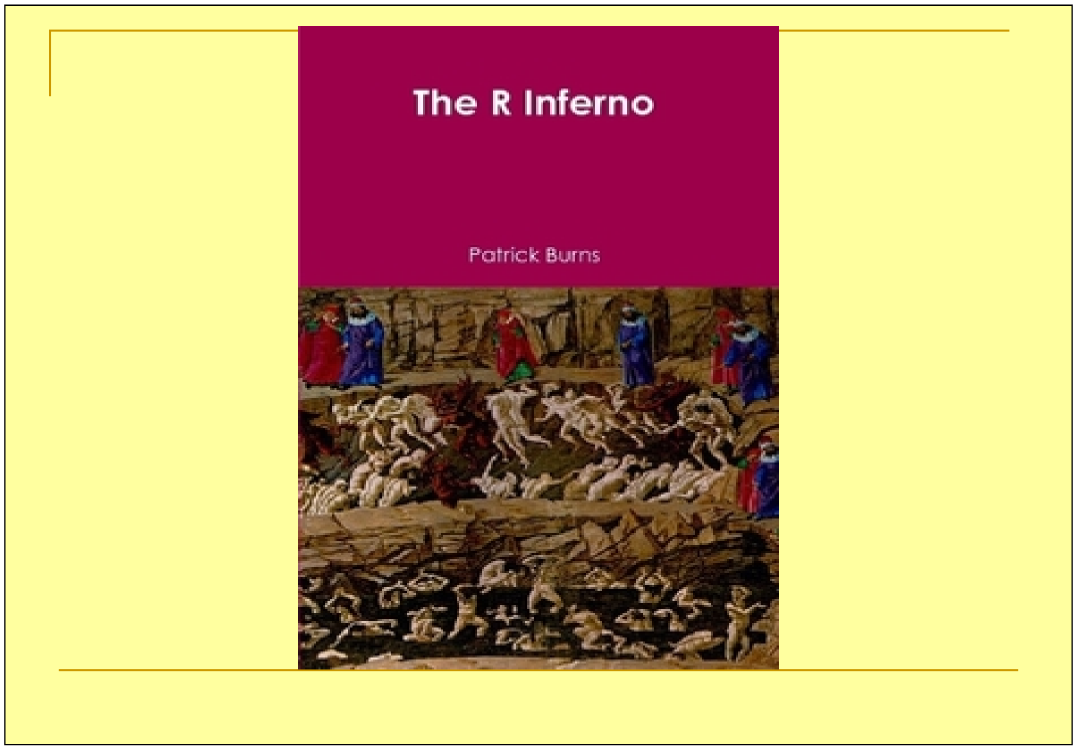
3837

Contributed packages on CRAN on 2012 May 28

A count of packages that was 12 days before the talk was, of course, out-of-date.

If you think you can learn all of R, you are wrong. For the foreseeable future you will not even be able to keep up with the new additions.

There are some absolutely wonderful packages on CRAN. There are undoubtedly some horrendously bad packages. But most of all there is massive redundancy – lots of ways of doing pretty much the same thing. It is the linear regression problem on steriods.



If you are going to enjoy the bliss of R, you're going to have to suffer the pain of R.

But you don't have to suffer alone. There is 'The R Inferno'. You can amuse yourself by seeing how others have suffered before you.

You can spend an extravagant amount on a physical copy, or you can be rational and get the free pdf from http://www.burns-stat.com

I have both and I find the pdf to be more useful. (Especially because the index isn't very good.)

There was a question from the audience that I rather stumbled through.

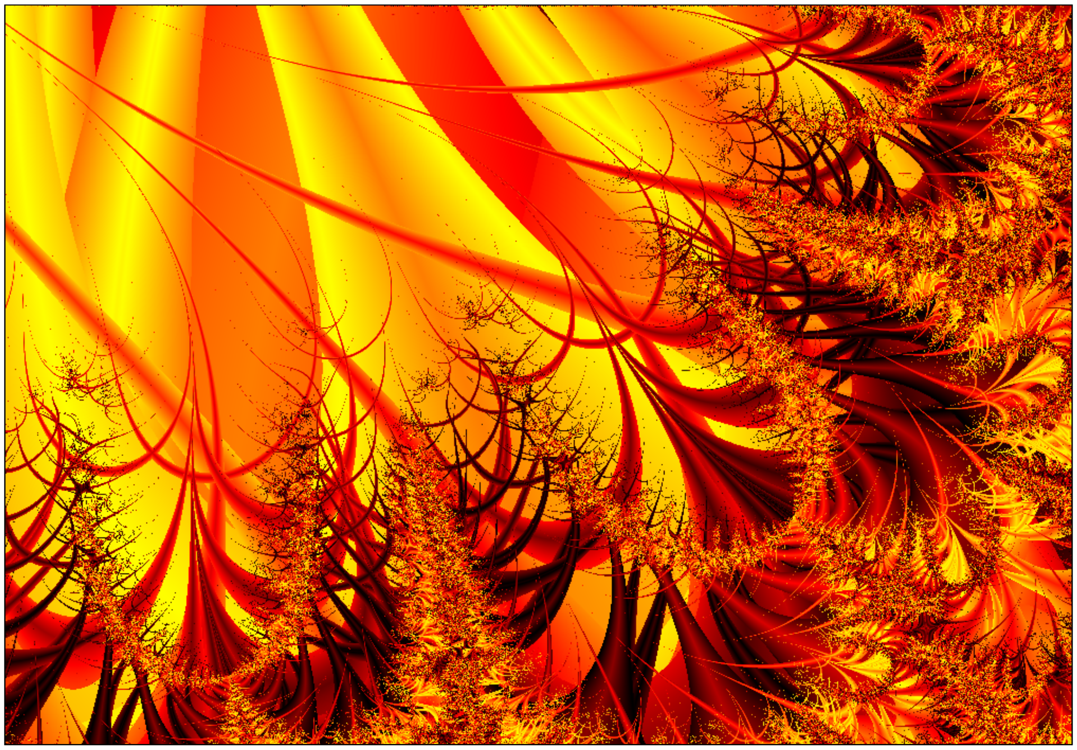
The question was: What about the future?

I correctly hedged by stating that my first prediction is that I'm not good at prediction.

R will eventually die in favor of something else. I think it will need to be something clearly superior to do that. Blue S was clearly superior to Brown S, and it completely replaced it immediately. White S was incrementally better than Blue S – there was accommodation rather than replacement.

In the meantime, R has a lot of momentum. It is likely to continue to absorb users from SAS, SPSS, Matlab and other platforms. It should, and hopefully will, replace lots of the data analysis that is currently done in spreadsheets.

My final piece of advice is:



Relax

and

Embrace

the

Chaos

that is R

“Fractal Friday 3” courtesy of MoneyScience

or: How I Learned to Stop Worrying and Love the Bomb