



# Object Oriented Programming Systems (oops? - or just right?)

---

Robert Gentleman

# Outline

---

- Historical Comments
- Statistician Programmers
- Software Engineering vs use
- OOPS
- Comments

# JMC – philosophy

---

- users should be drawn in, and they should find a system that is
  1. expressive: easy to do what they want
  2. programmable: they will want to make small changes
  3. extensible: they will consider making larger changes

# JMC – Philosophy

---

- the S language meets these criteria very well
- there is now a very large user base
- S is widely used in many scientific disciplines (a recent poll on Bioinformatics.org showed that more than 25% of respondents use it – the largest proportion for any language)
- many hundreds of add-on packages (CRAN is over 300, Bioconductor over 100)

# R

---

- around 1992 Ross Ihaka and I started to work on a language that was Lisp/Scheme based
- a hybrid, of sorts, between Lisp-stat (who likes some many brackets anyways?) and S (array-bounds checking, anyone?)
- it was really just the most simple Scheme interpreter on which we grafted an S-like syntax
- people started to use it - some of them became developers

# R

---

- Lesson 1: you cannot be similar to another language - you must be the same, or you must be different
- so - we needed to get JMC to change his mind about some things - and we did too
- this was easy - since like all of us, John is open to criticism - easily sees the errors of his ways, and is quick to change them
- or at least - he put up with us, and our questions - and I think we all learned a lot

# A big idea

- where does S lead

---

- interactive data analysis
- report generation, with graphics
- report generation with integrated software (literate data analysis)
- reproducible research

# Software Engineering

---

- the number of programmers working on/in the language has grown enormously
- large projects are being carried out (pieces of code larger than S itself written in S)
- more and better tools are needed to support these initiatives
- one tool/paradigm is object oriented programming

# Software Design

---

- there is tension - easy to use often contravenes some of the primary goals of good design
- working at the command line is not the same as writing a large body of interconnected software components

# OOP

---

- basically it has been observed that writing code, so that it contains components that represent real things, and the actions that you might perform on them, leads to better programs that are more easily understood and that interoperate
- hence OOP - objects, represent things and methods (generic functions) represent the actions that can be performed on these things

# OOP

---

- there are many styles - in recent years many seem to think that their favorite language invented the idea
- and that the way their favorite language does this is the right way
- but unless their favorite language is Lisp or Smalltalk - they are probably wrong about the first point - and possibly wrong about the second

# OOP

---

- the R implementation has 2 built in OOP systems and 3 others available as add-ons
- S3 and S4 are builtin
- some Java-style add-ons R.oop, OOP, etc
- some rationalization is needed - the technical part of this talk is about mechanisms that might lead to convergence

# OOP S3 Classes

---

- S3 - members of a class are not guaranteed to conform
- so testing of instances is constantly required (and inefficient)
- inheritance does not work

# OOP: Generic Functions

---

- generic functions - polymorphism
  - plot does something useful for many different data types
- methods - implement the actual mechanisms for different classes of objects
- the link between generic and method is very weak - things with the right names are methods
- this make it possible, likely, for unintended interactions

# OOP- Methods

---

- Josephine User, writes a function
  - `foo.bar`, that does something useful
- Pauline Programmer, has a generic function named `foo`, in her software
- And Joe Schmuck tries to use both - no good thing happens

# S4

---

- real classes
- methods are registered with generics

# Comments

---

- we will get more programmers by giving them a better and richer environment in which to work
- it is remarkably easy to experiment
  - most users simply don't care - that is not what they want
  - but we need to try and encourage that sort of behaviour - read -> experiment -> write