Interfaces, Efficiency and Big Data

John M. Chambers

useR!2014

Thesis:

- Big data and/or highly iterative computations are important challenges, sometimes.
- Other languages, other computing models or specialized hardware will help, sometimes.
- Well-designed interfaces can provide such computations effectively to R users.
- In fact, interfaces are central to R and always have been.

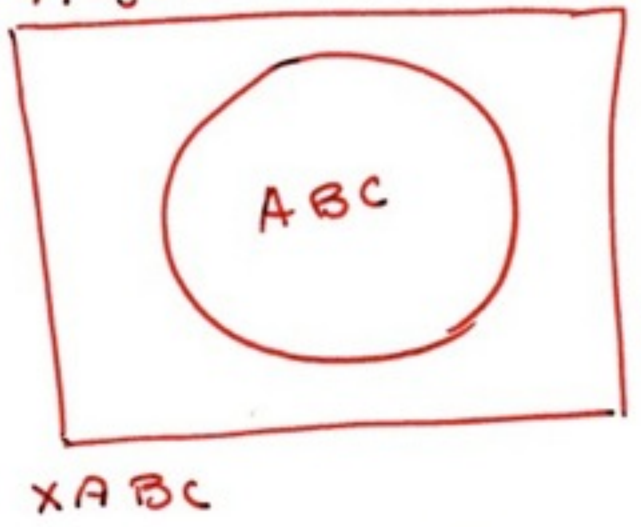
Bell Labs,
Murray Hill NJ

May 5, 1976

JTM
1

Algorithm Interface

5/5/76

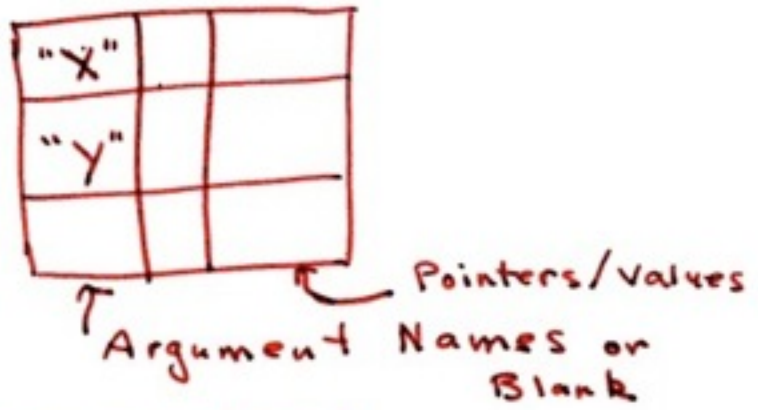


ABC: general (FORTRAN) algorithm

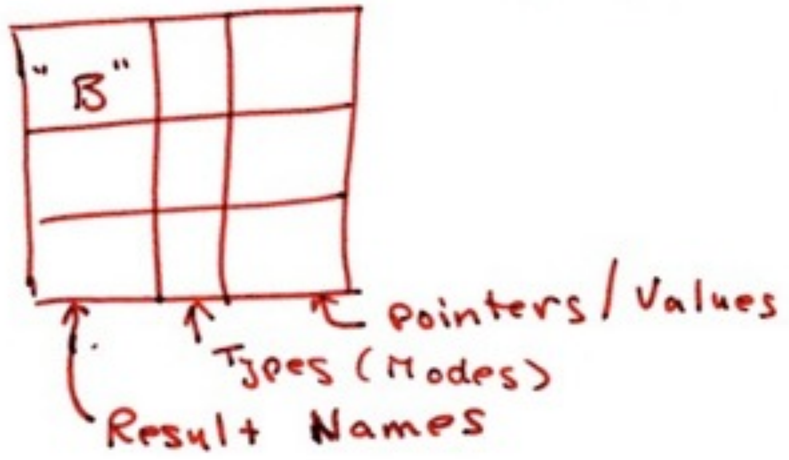
XABC: FORTRAN subroutine to provide interface between ABC & Language and/or utility programs

XABC (INSTR, OUTSTR)

Input INSTR →



OUTSTR →

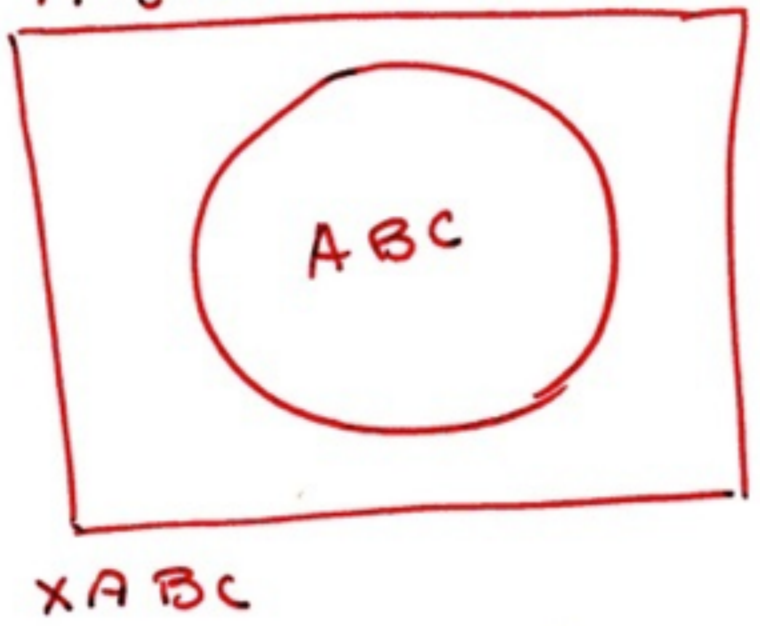


Note: Names are meaningful to Algorithm, not neccesarily to Language

JHC
①

Algorithm Interface

5/5/76



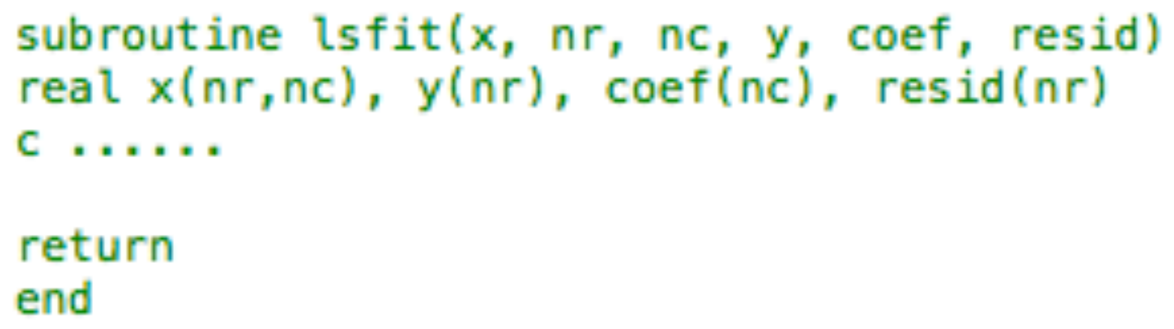
ABC: general (FORTRAN) algorithm

XABC: FORTRAN subroutine to provide interface between ABC & language and/or utility programs

XABC (INSTR, OUTSTR)

Implementation

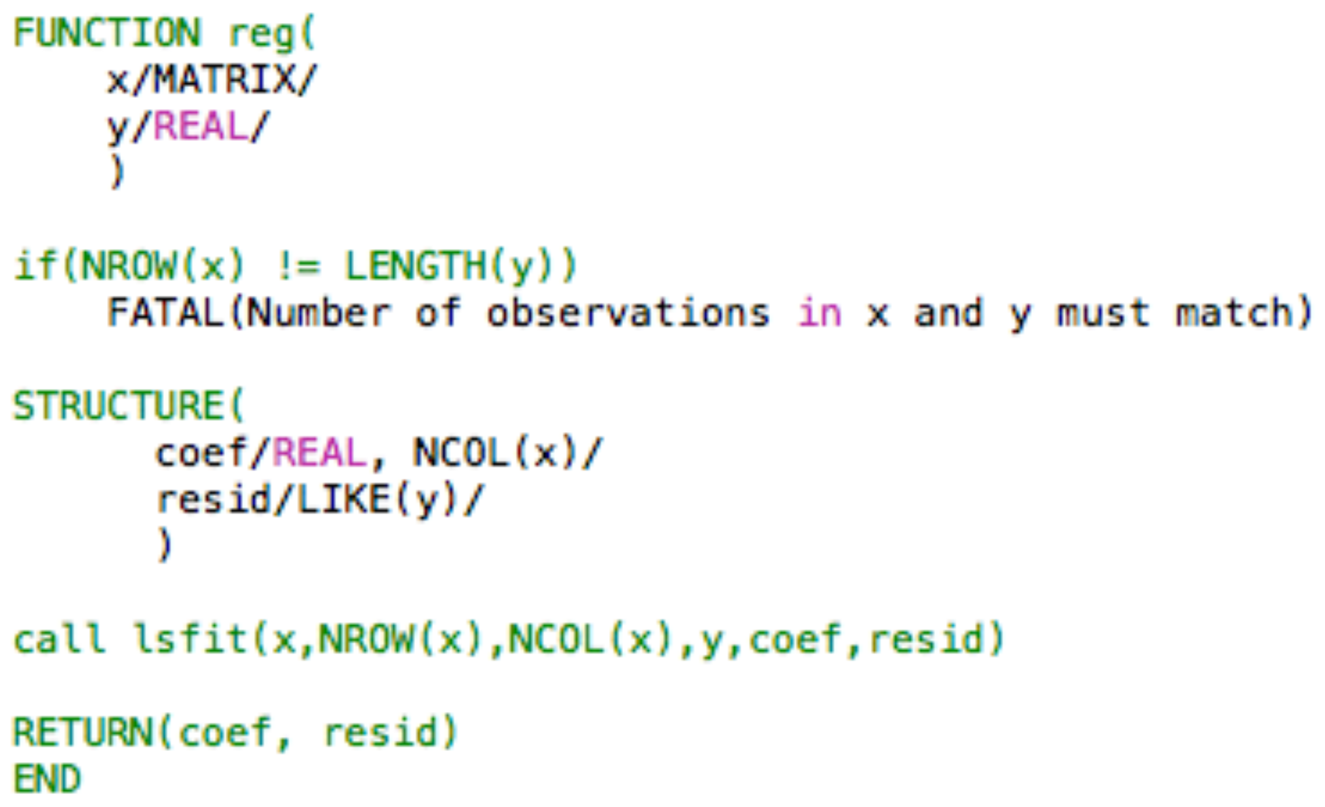
“Algorithm” (Fortran)



```
subroutine lsfit(x, nr, nc, y, coef, resid)
real x(nr,nc), y(nr), coef(nc), resid(nr)
c ......

return
end
```

Interface Language



```
FUNCTION reg(
    x/MATRIX/
    y/REAL/
    )

if(NROW(x) != LENGTH(y))
    FATAL(Number of observations in x and y must match)

STRUCTURE(
      coef/REAL, NCOL(x)/
      resid/LIKE(y)/
      )

call lsfit(x,NROW(x),NCOL(x),y,coef,resid)

RETURN(coef, resid)
END
```

The First Versions of S

- S was designed as an interactive interface to the best "algorithms" (Fortran-callable library, ca 1976);
- It had versions of (most of) the functions in R's base package that reference *Becker, Chambers & Wilks (1988)*
- S was NOT designed as a ground-up programming language.

Two Comments on early S

"At last, a computer language that speaks *my* language."

(faculty, Carnegie Mellon Stat. Department)

"S is great, but *serious* data analysis will always have to be done in Fortran."

(my Bell Labs management)

"Ideas into software" & serious applications:
Our goal is still to do both.

3 Key Principles

Object	Everything that exists is an object.
Function	Everything that happens is a function call.
Interface	R is built on interfaces to many “algorithms”.

Implications

- R evaluation consists of uniform, high-level function calls.
- Function call is O(10^3) ops (my rough estimate).
- Lots of dynamic memory management and copying; all objects are kept in main memory.
- Interfacing to a variety of non-R "algorithms".
- Not originally designed to program low-level computations.

Usually, computer speed and memory size make all this irrelevant.
When it's not, we can and should augment R, keeping in mind the key principles.

Interfaces, Efficiency and Big Data

- Unlike 38 years ago, there are now many possible "interfaces" (languages, computing models, hardware,…)
- Our challenging applications and our circumstances are too diverse for one solution to fit them all.
- Without losing the R that works in "ordinary" circumstances, we need to explore a variety of interfaces to alternatives.

Some Promising Projects

1. Rcpp, Rcpp11: Interface to C++ and programming in C++ [CRAN, for Rcpp]
2. LLVM for R: Compiling toolkit for R [omegahat.org for RLLVM, RLLVMCompile]
3. h2o: Interface and Java-based computations for big data [CRAN]

	Approach to Interfaces
Rcpp, Rcpp11	• Generate interface to call C++ with less programming than `.Call` • “Interface language” based on C++ *Interfaces*
RLLVM, RLLVMCompile	Compile R-language code into specialized forms for efficiency or other goals. *Efficiency*
h2o	• Compressed, efficient external version of data frame objects. • Interface from R functions to fast computations on big data. *Big Data*

Summary

- Interfaces are an integral part of R.
- For big data and intense computations, explore widely.
- Useful new interfaces exist; more work on them will be valuable.
- The best future is one of variety, not uniformity.