Current quirks in R (and S)

Peter Dalgaard
Department of Biostatistics
Faculty of Health Sciences
University of Copenhagen
p.dalgaard@biostat.ku.dk

Workshop on Distributed Statistical Computing
Vienna, March 19–23, 1999

Points for discussion

(in mixed order)

- Things that don't work
- Things to make it easier on the beginner
- Improving internal structure/logic of code
- Improving structure of system
- Dreams...

Model formulas

- (Why aren't these coded in the language itself?)
- Anomalies: `~1:x`, `~x^2`, also other specials
- Contrasts (and display thereof)
- Non-full rank matrices (mixed-effects!)

Consistency

- `t.test(x,y)`
- `kruskal.test(z,g)`
- `lm(z~g)`
- `coplot(y~x|g)`
- `boxplot(x,y)` but not `stripplot(x,y)`
- `coxph(...~x+strata(g))`
- `prop.test(succ,total)`
- `chisq.test(cbind(succ,fail))`

S(-plus) compatibility?

Odds and ends

- Silly inefficiencies, e.g. `deparse(object[[i]])[1]` in `data.frame`.
- Problem with return value in browser
- Wanted: proper debugger with breakpoints (in code and data)?
- Autoloadable packages
- Namespaces
- Constants (`c`, `t`, `pi`, `pt`, `df`, ...)
- `NA` handling (esp. strings)

Unprintable functions

- Typing `ls`, `help`, `demo`, etc. and forgetting the `()`
- Make them class `system.function` with a `name` attribute:

```
system.function<-function (obj) {
    name <- deparse(substitute(obj))
    class(obj) <- "system.function" ; attr(obj, "name") <- name
    assign(name, obj, envir = sys.frame(sys.parent())) }
print.system.function<-function (x, ...) {
    name <- attr(x, "name")
    cat("'", name, "' is a syste.......") }
```

```
> system.function(ls)
> ls
'ls' is a system function.
If you *really* want to see its definition, use print.default(ls)
To call ls with no arguments, use ls()
```

Simplifying R code

- Recognize frequently used idioms
- Create low-level building blocks
- Already done some (`quote`, `evalq`)
- Some obvious candidates: `sys.frame(sys.parent())`, `eval(e,sys.frame(sys.parent()))`, `deparse(substitute(x))`
- Do we understand the idioms well enough?

Evaluation model

- Getting it documented!!
- — and scrutinized
- Strange mix of S's search path, lexical scoping and closures
- Undesired closures when porting S functions (`family`, e.g.)
- Top frame attachments
- Unnamed recursives (`Recall` not working)

Error trapping

We're not really good at this:

```
> lm(igf1~testvol+menarche,data=juul2)
Error: names attribute must be the same length as the vector
```

(OK, that one is probably fixable. . .)

Generic problem with reusable modules:

- checking "..." arguments
- passing them to a partially specified component

Preemptive checking leads to clumsy code. Can we do better?

Object-oriented error trapping?

Parse/deparse

- Disappearing comments (BUG!)
- Parenthesis handling: R keeps '(' in parsed expressions and cannot deparse expressions without them

```
> e[[1]]<-as.name("*")
> e
3 + 4 * 5
> eval(e)
[1] 35
> e<-quote((3+4)*5)
> e[[2]]
(3 + 4)
> e[[2]][[1]]
(
```

Code review?

Glancing at `[<-.data.frame`:

```
...
    if ((nA <- nargs()) == 4) {
        has.i <- !missing(i)
        has.j <- !missing(j)
    }
    else if (nA == 3) {
        ## really ambiguous, but follow common use as if list
        if (is.matrix(i))
            stop("matrix subscripts not allowed in replacement")
        j <- i
        i <- NULL
        has.i <- FALSE
        has.j <- TRUE
    }
    else if (nA == 2) {
        value <- i
        i <- j <- NULL
        has.i <- has.j <- FALSE
    }
 ...
```

(Why `value<-i` in the 2-arg case, but not `value<-j` when 3 args?? Splus has more elaborate code but same oddity.)

Dreaming on...

- 3-d scenes to be fed to raytracing programs. Stereograms?
- Parallel processing (`parapply`?)
- What can the language do? "Scheme with a friendly face." More primitives for general purpose programming, e.g. file handling.
- Multithreaded coding, GUI interfaces. The language base could be very good for this kind of application (may avoid the multi-stack problem that leads to GUI-centric programming – "1 Infinite Loop" – with C++ type object-oriented languages?)