Understaning nlsModel in the base R nls() function

Arkajyoti Bhattacharjee, Indian Institute of Technology, Kanpur
John C. Nash, University of Ottawa, Canada

05/06/2021

nlsModel()

- Current understanding (JN): nlsModel (and nlsModel.plinear too probably) creates an R object that it labels as class "nlsModel." This object contains functions that are called from nls.c::nls_iter to run the interation and estimate the parameters in the model. There seem to be some extraneous functions, and we can hopefully learn enough to remove the extras.
- The current structure is to particularize the functions in "m" so the (essentially) external nls.c code acts on these. As a first goal, and part of learning how things work, we will want to replace the nls.c::nls_iter with all-R equivalent.

A script to examine the output of nlsModel()

We will use the Hobbs weed infestation problem (Nash (1979), page 120) again. ?? AB: we should build a set of test problems that are easy to try out. Let us discuss the examples that are in the help for nls. (?nls in R will show them.)

```
# Data for Hobbs problem
ydat  <-  c(5.308, 7.24, 9.638, 12.866, 17.069, 23.192, 31.443,
            38.558, 50.156, 62.948, 75.995, 91.972) # for testing
tdat  <-  seq_along(ydat) # for testing

# A simple starting vector -- must have named parameters for nlxb, nls, wrapnlsr.
start1  <-  c(b1=1, b2=1, b3=1)
eunsc  <-   y ~ b1/(1+b2*exp(-b3*tt))
str(eunsc)
```

```
## Class 'formula'  language y ~ b1/(1 + b2 * exp(-b3 * tt))
##   ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
```

```
# Can we convert a string form of this "model" to a formula
ceunsc <- " y ~ b1/(1+b2*exp(-b3*tt))"
str(ceunsc)
```

```
##  chr " y ~ b1/(1+b2*exp(-b3*tt))"
```

```
weeddata1  <-  data.frame(y=ydat, tt=tdat)

## Now ready to try things out.
library(nlspkg) # ?? needed because base R does not export nlsModel()
```

```
## Registered S3 methods overwritten by 'nlspkg':
##   method           from
```

```
##   anova.nls         stats
##   coef.nls          stats
##   confint.nls       stats
##   deviance.nls      stats
##   df.residual.nls   stats
##   fitted.nls        stats
##   formula.nls       stats
##   logLik.nls        stats
##   nobs.nls          stats
##   plot.profile.nls  stats
##   predict.nls       stats
##   print.nls         stats
##   print.summary.nls stats
##   profile.nls       stats
##   residuals.nls     stats
##   summary.nls       stats
##   vcov.nls          stats
##   weights.nls       stats
```

```
##
## Attaching package: 'nlspkg'
```

```
## The following objects are masked from 'package:stats':
##
##     asOneSidedFormula, getInitial, nlminb, nls, nls.control,
##     NLSstAsymptotic, NLSstClosestX, NLSstLfAsymptote, NLSstRtAsymptote,
##     numericDeriv, selfStart, setNames, sortedXyData
```

```
nmod1<-nlsModel(form=eunsc, data=weeddata1, start=start1, wts=NULL, upper=NULL, scaleOffset = 0, nDcent
str(nmod1)
```

```
## List of 16
##  $ resid     :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 325 15 325 30 22 37 973 973
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ fitted    :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 326 16 326 29 23 36 974 974
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ formula   :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 327 17 327 31 24 38 975 975
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ deviance  :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 328 18 328 31 25 38 976 976
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ lhs       :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 329 13 329 26 20 33 977 977
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ gradient  :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 330 18 330 57 25 64 978 978
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ conv      :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 331 14 331 34 21 41 979 979
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ incr      :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 332 14 332 42 21 49 980 980
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
```

```
##  $ setVarying:function (vary = rep_len(TRUE, np))
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 333 20 356 7 27 14 981 1004
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ setPars   :function (newPars)
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 357 17 364 7 24 14 1005 1012
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ getPars   :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 365 17 365 36 24 43 1013 1013
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ getAllPars:function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 366 20 366 39 27 46 1014 1014
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ getEnv    :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 367 16 367 29 23 36 1015 1015
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ trace     :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 368 15 374 7 22 14 1016 1022
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ Rmat      :function ()
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 375 14 375 32 21 39 1023 1023
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  $ predict   :function (newdata = list(), qr = FALSE)
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 376 17 377 56 24 56 1024 1025
##   .. ..- attr(*, "srcfile")=Classes 'srcfilealias', 'srcfile' <environment: 0x556ffc840928>
##  - attr(*, "class")= chr "nlsModel"
```

```
ls.str(nmod1)
```

```
## conv : function ()
## deviance : function ()
## fitted : function ()
## formula : function ()
## getAllPars : function ()
## getEnv : function ()
## getPars : function ()
## gradient : function ()
## incr : function ()
## lhs : function ()
## predict : function (newdata = list(), qr = FALSE)
## resid : function ()
## Rmat : function ()
## setPars : function (newPars)
## setVarying : function (vary = rep_len(TRUE, np))
## trace : function ()
```

```
print(nmod1)
```

```
## $resid
## function() resid
## <bytecode: 0x556ffd1da6e8>
## <environment: 0x556ffd217a10>
##
## $fitted
## function() rhs
## <bytecode: 0x556ffd1d96f0>
```

```
## <environment: 0x556ffd217a10>
##
## $formula
## function() form
## <bytecode: 0x556ffd1dc528>
## <environment: 0x556ffd217a10>
##
## $deviance
## function() dev
## <bytecode: 0x556ffd1db530>
## <environment: 0x556ffd217a10>
##
## $lhs
## function() lhs
## <bytecode: 0x556ffd1de368>
## <environment: 0x556ffd217a10>
##
## $gradient
## function() .swts * attr(rhs, "gradient")
## <bytecode: 0x556ffd1dd370>
## <environment: 0x556ffd217a10>
##
## $conv
## function() convCrit()
## <bytecode: 0x556ffd1e0608>
## <environment: 0x556ffd217a10>
##
## $incr
## function() qr.coef(QR, resid)
## <bytecode: 0x556ffd1dfb88>
## <environment: 0x556ffd217a10>
##
## $setVarying
## function(vary = rep_len(TRUE, np)) {
##                  np <- length(useParams)
##       useParams <<- useP <-
##                       if(is.character(vary)) {
##                           temp <- logical(np)
##                           temp[unlist(ind[vary])] <- TRUE
##                           temp
##                       } else if(is.logical(vary) && length(vary) != np)
##                           stop("setVarying : 'vary' length must match length of parameters")
##                       else
##                           vary # envir = thisEnv
##       gradCall[[length(gradCall) - 1L]] <<- useP
##       if(all(useP)) {
##           setPars <<- setPars.noVarying
##           getPars <<- getPars.noVarying
##           getRHS  <<-  getRHS.noVarying
##           npar    <<- length(useP)
##       } else {
##           setPars <<- setPars.varying
##           getPars <<- getPars.varying
##           getRHS  <<-  getRHS.varying
```

```
##           npar    <<- sum(useP)
##       }
##       }
## <bytecode: 0x556ffd1e22f8>
## <environment: 0x556ffd217a10>
##
## $setPars
## function(newPars) {
##       setPars(newPars)
##       resid <<- .swts * (lhs - (rhs <<- getRHS())) # envir = thisEnv {2 x}
##       dev   <<- sum(resid^2) # envir = thisEnv
##       if(length(gr <- attr(rhs, "gradient")) == 1L) gr <- c(gr)
##       QR <<- qr(.swts * gr) # envir = thisEnv
##       (QR$rank < min(dim(QR$qr))) # to catch the singular gradient matrix
##       }
## <bytecode: 0x556ffd1fbd98>
## <environment: 0x556ffd217a10>
##
## $getPars
## function() getPars()
## <bytecode: 0x556ffd202cf8>
## <environment: 0x556ffd217a10>
##
## $getAllPars
## function() getPars()
## <bytecode: 0x556ffd202240>
## <environment: 0x556ffd217a10>
##
## $getEnv
## function() env
## <bytecode: 0x556ffd2055f0>
## <environment: 0x556ffd217a10>
##
## $trace
## function() {
##       d <- getOption("digits")
##       cat(sprintf("%-*s (%.2e): par = (%s)\n", d+4L+2L*(scaleOffset > 0),
##               formatC(dev, digits=d, flag="#"),
##               convCrit(),
##               paste(vapply(getPars(), format, ""), collapse=" ")))
##       }
## <bytecode: 0x556ffd2045f8>
## <environment: 0x556ffd217a10>
##
## $Rmat
## function() qr.R(QR)
## <bytecode: 0x556ffd20c680>
## <environment: 0x556ffd217a10>
##
## $predict
## function(newdata = list(), qr = FALSE)
##                 eval(form[[3L]], as.list(newdata), env)
## <bytecode: 0x556ffd20f3d8>
## <environment: 0x556ffd217a10>
```

```
##
## attr(,"class")
## [1] "nlsModel"
```

How contents of the "m" object are used

This section refers to `nls()` in baseR. Replacement functions will likely be different.

References

Nash, John C. 1979. *Compact Numerical Methods for Computers : Linear Algebra and Function Minimisation.* Book. Hilger: Bristol.