Working Document for the Improvement to nls() GSOC project

Arkajyoti Bhattacharjee, Indian Institute of Technology, Kanpur
John C. Nash, University of Ottawa, Canada
Heather Turner, University of Warwick, UK

10 August, 2021

TODOS

- Show how `dir` works in nlsalt::numericDeriv(); need some examples/tests Currently is used when approaching upper bounds. ?? Need to put this in other codes and document with examples.
- Can we find a way to use analytic derivatives where they are available? nlsderivchk.R shows deriv() not terribly smart. [partly DONE]: the revision to rjfun now tries nlsdericchk() and then uses numericDeriv() if analytic deriv() fails. Temporary changes to nls.control() in nlsj.control(), but these choices need some rationalization, cleanup and documentation [unfinished]: have not added numDeriv choices (of which there are several); nor has nlfb been upgraded to use such choices.
- At some point, it will be important to know what forms of residual and jacobian evaluation forms work best. That means measuring timing (which we know how to do with microbenchmark) and memory, which is a little more difficult because of R's "garbage collection" and other policies. ?? Not even begun, but eventually performance will become important.
- Does the presence of p1 and p2 objects in .GlobalEnv cause trouble for nls() (it does seem to for nlsj)??
- Testing structure to enable us to compare different routines on all the different tests.
- nlsModel needs better attention to WHICH routines from the `m` object are used EXTERNALLY to nls(). `nlsj` with NOT have all of these. And `m` is now (2021-7-12) built at the end of the computation rather than the beginning.
- When we evaluate derivatives, should we ignore bounded/masked parameters to avoid stepping into inadmissible area?
- ### mimic what model.frame.default does

  wts <- if (mWeights) rep_len(1, n) else eval(substitute(weights), data, environment(formula))
- Can we handle indexed parameters??

Possibly working through model frame. Note original nls code varIndex.

Abstract

nls() is the primary nonlinear modeling tool in base R. It has a great many features, but it is about two decades old and has a number of weaknesses, as well as some gaps in documentation. This document is an ongoing record of work under the Google Summer of Code 2021 of the first author. As such it is NOT meant to be a finished academic report, but a form of extended diary of activity, issues and results.

Project repository

https://gitlab.com/nashjc/improvenls

———— MAIN OUTCOMES ————

Here is an annotated listing of product of the project.

- RefactoringNLS.Rms: the document which will become the main report of this project.
- DerivsNLS.Rmd: a document to explain different ways in which Jacobian information is supplied to nonlinear least squares computation in R. File `ExDerivs.R` is intended to provide examples.?? needs documentation??
- ImproveNLS.bib: a consolidated BibTex bibliography for all documents in this project, possibly with wider application to nonlinear least squares in general
- MachineSummary.Rmd: an informal investigation of ways to report the characteristics and identity of machines running tests. `MachID.R` offers a concise summary function ?? should revisit??
- nlsalt: attempt to mirror nls() behaviour in R. UNFINISHED?? The effort showed that the structure of the programs was difficult to follow, undocumented, and unsuited to adding improvements like Marquardt. Possible that we will abandon this. `nls-flowchart.txt` is a partial documentation of the structure.

‘ nls-changes-for-small-residuals-in-nls-R-4.0.2.zip: collected material for the JN fix to the relative offset convergence criterion failure when there are small residuals in problems sent to nls().

- nlsj: A refactoring of the nls() functionality. `redesign2107.txt` gives some notes.

‘ nlsModelDoc.Rmd

‘ nlspkg: a packaged version of the nls() code from R-base. Thanks to Duncan Murdoch

`nlsralt: a modified version of Nash and Murdoch package`nlsr‘ with improvements discovered as a result of this project. Need to add subset correctly and document changes e.g., to numerical derivatives. This is for JN rather than AB.

`PkgFromRbase.Rmd: an explanation of the construction of the`nlspkg‘ from the code in R-base.

‘ TestsDoc.Rmd: a survey of testing tools in R. ?? need to incorporate MachID and the many tests.

‘ VarietyInNonlinearLeastSquaresCodes.Rmd: a review of the different algorithms and the many choices in their implementation for nonlinear least squares. ??unfinished but getting there.

- WorkingDocument4ImproveNLS.Rmd: this document

———— DIARY PORTION ————

- In reverse date order *

[2021-8-7] - [2021-8-10] - JN fix and test nlsj bounds – needed to move marquardt stabilization AFTER the bounds check. - AB worked on tester, modularizing code, checking and cleaning code. - both: meetings on Sunday Aug 8 and Tues Aug 10

Heather’s comment: For the internal data sets like “problems.csv” you might use sysdata.rda: https://r-pkgs.org/data.html#data-sysdata

[2021-8-6] AB: Worked on creating a package version of the runner folder called “nlscompare.” Still messy. [To be discussed in the meeting.] Created a corresponding article using The R Journal template from “rticles” package in RStudio. Need suggestions on that. [To be discussed in the meeting.]

[2021-8-5] AB: Did error handling on Testing part of the “run_JN.R” file. Made changes in how “NLSErrorLog” outputs - improved presentation and clarity of results.

JN: look into the possibility of a generic `getStart(formula, data)` function. Test if SSxyzw functions show a class "selfStart" easily. No! Note that one can try to have formulas of type

```
y ~ SSab(x, p1, p2, p4) + SScd(x, p2, p3)
```

and `nls()` seems not to decode their selfstarts. This is another INCONSISTENCY. There is also the inconsistency in indexed parameter output. The output is numbered, but is NOT in the form prmA[index] but prmAindex in output.

[2021-8-4] Looked into "RefactoringNLS" adn suggested minor fixes.

[3-8-2021] Discussed problems with runner files and testthat files - 1. NLSlower, NLSupper, NLSsubset, NLSweights not used - later 2. gradient in run_JN causing problems - solved 3. How to set up nlsr in methods? "algorithm=" needs to be handled! Idea: Before output_nls and output, define NLSrunline if solver==nlsr::nlxb; algorithm=default in output_nls; how to setup control for nlsr::nlxb - setup runline – FILTER IT 4. commented out gradient from "Passed" condition and changed 6–>5 – SOLVED 5. minpack convInfo()?? – SOLVED 6. where to put try statement? problems like gradient may occur? SOLUTION - put try statements and write a function check2() that outputs "ERROR" in passed? 7. Note - Tetra gives singular grad in nls but not in nlsj – BETTER?? 8. Compare "port" in nls with "default" in nlsj and "marquardt" in nlsj for bounds. Do also (for bounds) nlsLM, nlxb.

Latest test and runner folders attached in a single zip file.

SUGGESTIONS -

In GitLab repo - improvenls, inside "Tests," you can

DELETE: "singular Gradient Problems," "test_better.R," "test-Sulfi_1.R," "test-Sacch2_1.R," "test-Tetra_1.R"

KEEP: "test-Isom_1.R"

Note that both "Tests" and inside "nlsj"-"tests" are similar. I think putting the attached (inside zip file) "tests" in the nlsj-tests would be better. One file in "Tests" does have a comment - "better stub" - so maybe you are using those files somewhere? If not, you may consider to delete them as well.

[1-8-2021 to 2-8-2021] AB. Edited run file in runner directory to create "nlsErrorLog.csv" hat contains which solver, algorithm, control, filename, date and machine that causes an error while running run.R. Added edited Croucher_base.R in testhat. run.R is currently running without any error. Maybe need to add more test files in runner??

[28-7-2021 to 30-7-2021] Fixed runner directory contents and edited testthat files. Succeeded with "control" using JN's suggestion.

[2021-7-28] to [2021-7-30]

JN: worked on nlsj and on a stripped down version nlsgn (only the Gauss Newton and no bounds). Some issues with haveQRJ NOT being reset when a successful line search complete. Also the tactical rules for setting the stepsize `fac` had to be adjusted as AB pointed out that the "Leaves" test was not getting answers equivalent to those from `nls()`. Also some cleanup of several documents.

Both: meeting of July 30 found glitch in Gillespie's benchmarkme::get_ram(). Sorted out some of the issues in tests (for packages) and the run.R program, in particular the string to be added for different "control" sections of the calls.

AB: more work on the tests.

[2021-07-25 - 2021-07-26]

- AB: Got a working workaround function to resolve RAM issue for Windows (probably; Machine ID is now machine dependent. Need to make it robust??). Started "runner_v2" to compare various NLS

methods. Observed that "plinear" created an error `"Error in nls(formula = NLSformula, data = NLSdata, start = NLSstart, algorithm = NLSmethods$algorithm[j]) : step factor 0.000488281 reduced below 'minFactor' of 0.000976562"`.

There is an issue with "control" in "methods.csv" file. Trying to resolve it. Spreadsheet is running successfully, but with default options only. Should column names be extended??

[2021-7-25]

- JN: continued work on nlsj to check features working. Some rewrite of sections of the Refactoring article.

[2021-7-24]

- gradual working through the code to add Marquardt yet not lose Gauss Newton. Looking to Hobbs to prepare problems to test different situations, in particular,
  - bounds
  - singular gradient
  - bad start
  - good start

  and match `nls()` output.

[2021-7-23]

- nlsj more or less running OK with base Gauss Newton. Putting in Marquardt started but messy. Issues of object*m*resid() (which is WEIGHTED) vs. resid(object) (which is NOT) partly resolved, but definitely needs documenting better.
- meeting resolved some ideas. Runner is progressing.

[2021-7-22]

- JN did some cleanup of nlsj and made some modifications to the Croucher example which we have been using to provide some initial checks of the code.
- AB provided some test files and infrastructure.
- JN suggested a function rmNLS() to remove test routine elements to keep the workspace tidy. We need to use `rm(list=objectname, envir=.GlobalEnv)` Note the "list=."

To discuss July 23:

- Should we create "base" files for families of test problems?
- Should we put file / problem name into a variable?
- Ditto for the description?
- How should be structure / build the problem list?
- In the "runner" program, how should the file location be specified? This is a cross-platform issue, and we would like a standard program that needs minimal change between platforms / users.
- Should we modify rmNLS() or have rm() statements at end of runs?
- How to specify which program is to be run?
- How to provide lists of problems to be run?
- Further discussion of the output.

[2021-7-21]

[20-7-21] Discussed problems with test file outputs over GMeet. `tolerance` needs to be modified in certain expectations. Rather than comparing entire lists, was suggested to test particular list items. Problems with `testthat` while executing R CMD check, saying: "Error in library(testthat) : there is no package called 'testthat'."

[21-7-21] Fixed `testthat` issue. Apparantly, the test file names in the `testthat` sub-directory inside `tests` directory needs to be prefixed with "test-filename.R." Realized this when a template test file was created using `usethis::use_test("nlspkg")` named "test-nlspkg.R." Still, tests require some work; errors maybe due to `nlsj`??

[24-07-21] Cleaned test files and ran R CMD check after moving the tests directory inside nlsj. Seems to work! Inside "run.R," benchmarkme::get_cpu does not seem to work on Windows 10. Reference Issue: https://github.com/csgillespie/benchmarkme/issues/25 Workaround: CPU<-as.numeric(system('wmic MemoryChip get Capacity,'intern=TRUE))[2]/1024^3 Plans to improve structure of runner discussed over GMeet.

- JN looked into minpack.lm handling of bounds constraints, and also downloaded the original **netlib** version of `minpack`. This does NOT handle bounds (at least no mention of "bounds," "upper" or "lower" were found w.r.t. parameters). Inside `minpack.lm` package, it seems that bounds are handled by the code

```
if(par[i] < NUMERIC_POINTER(OS->lower)[i])
  par[i] = NUMERIC_POINTER(OS->lower)[i];
if(par[i] > NUMERIC_POINTER(OS->upper)[i])
  par[i] = NUMERIC_POINTER(OS->upper)[i];
```

This is simply moving the parameters to the bounds.

Have I missed something?

[2021-7-20]

Meeting online. Worked through some issues relating to testing.

Outside of meeting, JN got nlsj to do a "first run" with bounds. Much to be checked and cleaned up.

AB working on tests, both for the "within package" and "across packages" roles.

[2021-7-19]

Look at indexed parameters example from `nls()`. This does NOT work at moment for nlsr[alt], even with wrapnlsr.

```
## The muscle dataset in MASS is from an experiment on muscle
## contraction on 21 animals.  The observed variables are Strip
## (identifier of muscle), Conc (Cacl concentration) and Length
## (resulting length of muscle section).
utils::data(muscle, package = "MASS")

## The non linear model considered is
##       Length = alpha + beta*exp(-Conc/theta) + error
## where theta is constant but alpha and beta may vary with Strip.

with(muscle, table(Strip)) # 2, 3 or 4 obs per strip

## We first use the plinear algorithm to fit an overall model,
## ignoring that alpha and beta might vary with Strip.
```

```
musc.1 <- nls(Length ~ cbind(1, exp(-Conc/th)), muscle,
              start = list(th = 1), algorithm = "plinear")
## IGNORE_RDIFF_BEGIN
summary(musc.1)
## IGNORE_RDIFF_END

## Then we use nls' indexing feature for parameters in non-linear
## models to use the conventional algorithm to fit a model in which
## alpha and beta vary with Strip.  The starting values are provided
## by the previously fitted model.
## Note that with indexed parameters, the starting values must be
## given in a list (with names):
b <- coef(musc.1)
musc.2 <- nls(Length ~ a[Strip] + b[Strip]*exp(-Conc/th), muscle,
              start = list(a = rep(b[2], 21), b = rep(b[3], 21), th = b[1]))
## IGNORE_RDIFF_BEGIN
summary(musc.2)
## IGNORE_RDIFF_END
library(nlsr)
## THIS FAILS -- we don't handle indexed parameters yet
musc.2a <- try(nlxb(Length ~ a[Strip] + b[Strip]*exp(-Conc/th), muscle,
              start = list(a = rep(b[2], 21), b = rep(b[3], 21), th = b[1])))
if (inherits(musc.2a, "try-error")) cat("nlxb indexed does not work")
# But
musc.2b <- try(wrapnlsr(Length ~ a[Strip] + b[Strip]*exp(-Conc/th), muscle,
              start = list(a = rep(b[2], 21), b = rep(b[3], 21), th = b[1])))
if (inherits(musc.2b, "try-error")) cat("wrapnlsr indexed does not work")
```

[**2021-7-19**]

- JN got first run bounds and masks working on Croucher example. Many questions and issues remain, but it is getting seemingly correct answers compared to nlxb.

Test alternative coding for bounds. Note that for loop seems twice as fast in microbenchmark.

```
## tests of bounds setting - J Nash 210719
# setup
rm(list=ls())
bdmsk <- c(1,1,0,-1,-3,1)
set.seed(123)
res <- runif(10)*10
J <- matrix(1:60, nrow=10)
pnum<-1:6
npar<-6
gjty<-t(J)%*%res # raw gradient
tr1 <- function(){
    for (i in 1:npar){
       bmi<-bdmsk[i]
       if (bmi==0) {
           gjty[i]<-0 # masked
           J[,i]<-0
       }
       if (bmi<0) {
           if((2+bmi)*gjty[i] > 0) { # free parameter
              bdmsk[i]<-1
```

```
#             if (watch) cat("freeing parameter ",i," now at ",pnum[i],"\n")
           } else {
              gjty[i]<-0 # active bound
              J[,i]<-0
#             if (watch) cat("active bound ",i," at ",pnum[i],"\n")
           }
       } # bmi
    } # end for loop
    ans<-list(J=J, gjty<-gjty, bdmsk=bdmsk)
} # end tr1

tr2 <- function(){
#    btmp<-rep(1,npar)
    gjty<-t(J)%*%res # raw gradient
    idx0 <- which(bdmsk==0)
    gjty[idx0]<-0 # masked
    J[,idx0]<-0
    idxb <- which((bdmsk < 0) & ((2+bdmsk)*gjty <= 0.0))
    bdmsk[-idxb] <- 1 # change to free
    gjty[idxb]<-0 # active bound
    J[,idxb]<-0
    ans<-list(J=J, gjty<-gjty, bdmsk=bdmsk)
}
bdmsk
J
gjty
print(tr1())
bdmsk
gjty
J
print(tr2())$bdmsk
print(tr1())$bdmsk
library(microbenchmark)
microbenchmark(tr1())
microbenchmark(tr2())
```

2021-7-18 and 19 AB

Working on creating test files based on `Examples for tests`. Initial draft based on BOD2 data sent to Dr.Nash. It contains nls vs nlsj for now; Will have to work on including nls vs minpack.lm::nlsLM and nls vs nlsr::nlxb.

[2021-7-16] + [2021-7-17]

- JN working on trying to put bounds into nlsj.R, but very early days.
- Started project report whith background and goals. RefactoringNLS.Rmd

[2021-07-14 to 2021-7-16] AB

Worked on `Examples for Tests` file. Includes those data from NRAIA package whose help files have models explicitly defined. Hobbs weed problem is also included in it. Will have to look into more examples??

[2021-07-16] Via Gmeet with JN did a code-walk through nlsj.R and discussed on how to proceed with tests.

[2021-7-15]

- JN changed NAMESPACE of nlspkg to remove the exportPattern line and explicitly export `nls` and `numericDeriv`. ?? More may be needed to be exported. Now R CMD check clears OK, but "test" fails.

[2021-7-14] **[2021-7-15]**

- some work on nlsj to purge errors raised by R CMD check. One WARNING was raised by generic functions e.g., fitted.nlsj being "undocumented." It turns out the same error is in `nlspkg`, and JN emailed Duncan Murdoch for advice.
- added asOneSidedFormula to nlsjextra.R but commented out. Not sure this is needed or useful, but put in collection so it is available to us.
- JN created a `Performance` directory and `timessloop.R`. Found sum(x^2) by far the fastest way to compute sum of squares.
- To check derivative computation in `nlsj`, JN created `testrj.R` (in `Performance` though not about timing at this moment, but about values.)
- AB working on SelfStart functions and tests.

[2021-7-13]

- JN some work on nlsj – more or less running with both analytic and numeric derivs, but need to check sign of residuals and jacobian for equivalence, and also for nlsr.
- AB edits on Variety paper incorporated by JN. (Excellent attention to detail by AB.)
- HT, AB, JN meeting. Discussed directions. AB will focus on tests to check equivalence of new codes with nls(). JN will concentrate on new code. Both to work on checking each other. WIll meet more frequently to do code walks.
- JN posted on r-devel asking about "subset." Note some replies pointing to "model.frame" as the structure through which subsetting is applied, along with some other things. JN to dig!
- HT suggestion in email led to notes on how we might structure test output. This is included as follows.

```
TestingIdeas.txt

2021-7-13

Following up Heather's email about the need to rigorously test to see if
our "new" code gets equivalent output to nls(), I think we should discuss
the "how" of testing and build it into the TestsDoc.

1) To be able to check how we are doing, we need to have a set of tests and
know what they do. You have been looking into this, but we should try to see
how we can structure things so they can be run easily.

Can we put each test in a separate file of R code, with a filename having
what we can call the testID in the name. Hence testID.R.nltst as a filename.

As a check on whether this is feasible, let me try with Croucher, since it
is small

# TCroucher.R.nltst
# Implements tests of nonlinear least squares code for the Croucher example
# Reference: https://walkingrandomly.com/?p=5254
```

```
xdata = c(-2,-1.64,-1.33,-0.7,0,0.45,1.2,1.64,2.32,2.9)
ydata = c(0.699369,0.700462,0.695354,1.03905,1.97389,2.41143,1.91091,0.919576,-0.730975,-1.42001)
Croucherdata<-data.frame(xdata, ydata)
Croucherform <- ydata ~ p1*cos(p2*xdata) + p2*sin(p1*xdata)
# ?? Do we want an array of starts? Some problems have > 1
Croucherstart<-list(p1=1,p2=0.2)
Crouchersubset<-1:8 # Possibly have subsets -- multiple tests.
# ?? do we want separate files for each test. More storage, but then each test
# has a separate ID and can be run independently
## Example use
# library(nlsj)
# fitj = nlsjx(ydata ~ p1*cos(p2*xdata) + p2*sin(p1*xdata), start=list(p1=1,p2=.2), subset=1:8, trace=T
# summarise
# summary(fitj)
callstring<-"(Croucherform, start=Croucherstart, data=Croucherdata, subset=1:8)"
#?? can we embed this in different calls e.g.,
#  runline <- paste("result<-nlxb(",callstring,")")
```

We can build a "database" or else simply documentation files (?? .Rd format like in R, or something similar and simple) to explain the origin and purpose of each test file. This collection will get large, but I think it will be best to keep things simple for now.

2) My feeling is that EACH variant of a test gets its own file that builds the problem, so that our output is indexed on machineID, testID, programID, and relevant time/date and results information.

While we could (and likely will eventually) use a database, for the moment I think a comma separated values (csv) file is more straightforward. As long as we agree the structure in advance, we can simply concatenate such files and draw them into data frames or databases easily.

3) Output fields from tests

machineID: the ID of the machine running the tests. We may need to set up a list of these. For ourselves it won't be large, but if others join us we may need to appoint a "naming authority". Just thinking ahead.

userID: the ID of the person who ran the test. Again we may eventually need a "naming authority". JN would be nashjc, for example.

testID: the particular test and variation (subset, start, etc.) I suggest that we think of test names and put a 2 or 3 digit suffix so we can expand over time.

UTCtime: A date and timestamp at UTC (GMT) in form yyyymmddhhmmss (maybe could leave off seconds, but .... UTC so we don't get timezone confusion.

programID: the program used. I suggest names like nlsr::nlxb, nlspkg::nls, minpack.lm::nlsLM so we identify as completely as possible. These can be the names used with the "runline" above.

tResult: a 1 character output. 0 = success, and anything else will need documenting.

notes: a field allowing us to add extra information

?? Can you think of anything else?

Files like this will be plain text and can simply be concatenated. The structure
is independent of the testing program(s) that we decide to use, but hopefully can
be output by them.

It seems that given my experience in the nonlinear codes, I should probably continue
with working on nlsj and the other packages, and you (Arkajyoti) should work on the
tests. That way we won't overlap too much and tread on each other's toes. Are you OK
with that.

On the other hand, I think that we should probably schedule extra meetings to do
code walks where one of us explains each line of code in a program to the other. I've
found that sort of exercise shows up loose ends and forces the code to be tightened.
It's how I did the programming with Mary Walker-Smith in the mid-80s and we made a
lot of progress quite quickly.

Talk later, but I wanted to get some of these ideas down.

Best, JN

tryrunline.R test

```
xdata = c(-2,-1.64,-1.33,-0.7,0,0.45,1.2,1.64,2.32,2.9)
ydata = c(0.699369,0.700462,0.695354,1.03905,1.97389,2.41143,1.91091,0.919576,-0.730975,-1.42001)
Croucherdata<-data.frame(xdata, ydata)
Croucherform <- ydata ~ p1*cos(p2*xdata) + p2*sin(p1*xdata)
# ?? Do we want an array of starts? Some problems have > 1
Croucherstart<-list(p1=1,p2=0.2)
Crouchersubset<-1:8 # Possibly have subsets -- multiple tests.
# ?? do we want separate files for each test. More storage, but then each test
# has a separate ID and can be run independently
## Example use
# library(nlsj)
# fitj = nlsjx(ydata ~ p1*cos(p2*xdata) + p2*sin(p1*xdata), start=list(p1=1,p2=.2), subset=1:8, trace=Tl
# summarise
# summary(fitj)
callstring<-"(Croucherform, start=Croucherstart, data=Croucherdata)"
#?? can we embed this in different calls e.g.,
runline <- paste("result<-nlsr::nlxb",callstring)
runline
eval(parse(text=runline))
result
```

Heather Turner12:00 -- in chat of meeting
So we can produce something like this: https://cran.r-project.org/web/checks/check_results_gnm.html
(Which is the CRAN package test results page for gnm.)

[2021-7-12]

- JN and AB completed evaluations for GSoC.
- JN working on nlsj package in last couple of days. Refactored into new file nlsjx.R with NO nlsjModel() function. Got this "running" on Croucher example (which is changed slightly to reflect new name.) Notes:
  - It seems nls() counts "iterations" whereas JN prefers jacobians and residuals. As structured, the algorithm computes residuals an extra time, since the internal backtracking does NOT compute jacobian. Possibly we should. Timing will be needed to see what is most efficient, though we will need a number of different sizes and types of problem. Can possibly think of ways to short-circuit the computation.
  - The particular choice of strategy for the backtracking always uses fac=0.5 on first iteration (as far as JN can determine.) This may or may not be a good choice, but for the moment it gives the same results as nls().
  - The "m" object is NOT built until we finish nlsj(). And some of the functions are missing / wrong. We need to fix what we do need and eliminate those that are not needed.
  - note that by using zero weights, the `subset` argument is handled OK.
  - Some functions for print and summary needed changes to allow them to run, since they try to use the functions from nlsModel.
  - Output is currently untidy. Need to fixup.

[2021-7-10]

- AB noted error in PkgFromRbase document that was fixed.
- JN working on `nlsj` and issues raised, along with some documentation of ideas in the "Variety" article.
- Serious issues of concern in the STRUCTURE of the `nlsj` package are mostly related to the placement of functions and features. For example, if we put functions in the nlsjModel() function and return them to nlsj() by the model object `m`, then there are some awkward communication issues of the convergence criterion information, then updated parameters, etc.

  Consider, for example, if we were to change from the nls() step-halving method of seeking a "better" set of parameters. The original method at each iteration has a set of parameters x_0 and forms $x = x_0 + s * \delta$ where s is stored in the variable `fac`. We start with this at 1.0 and halve it until the sum of squared weighted residuals is reduced in size. Then we double `fac` for the next major iteration (i.e., with x_0 replacing x and a newly computed δ.

  However, if we use a parabolic inverse interpolation, we need to store 3 sets of parameters and their deviances e.g., x_0, x and x_{half}, where we use a step of `0.5*fac` for the step s. These have to be shared with the objects and environment that nlsjModel() can access.

  ?? Arkajyoti: Give these issues some thought. I don't think there's a true "answer," just some good and some not-so-good choices. I'll do some trials to see what seems to work and how easy to understand and use.

  By creating a small function that builds m <- nlsjModel(....) and trying out some ideas, it appears the easiest way to work is to run the nlsjModel() then ee <- m$getEnv() to get the environment and put items in it. This is NOT necessarily how R gurus would do it, but likely will be safer (and maybe faster – need to check.)

[2021-7-7] - [2021-7-9]

- JN had computer troubles (failed machine and disks)

- AB participating in UseR!

[2021-7-5] [2021-7-6]

- some editing and extension of Variety document. ?? still needs some tidying and rationalization.
- redesign2107.txt is a set of notes on some redesign issues.
- Note new function nlsderivchk.R – checks if derivs can be found analytically. So far it seems we cannot do some partial derivatives analytically.
- JN checked nlsr will work with data in current workspace
- note

```
# A function to evaluate f in environment e
with_env <- function(f, e=parent.frame()) {
  stopifnot(is.function(f))
  environment(f) <- e
  f
}
```

- JN: Looking at ORIGINAL `nls.R` in `nlspkg`, it appeared that argument `subset` is NOT used. However, Croucher-example1.R modified to check using `subset=1:8` showed it did work. ?? Documentation could be improved! The input to subset should resolve to a set of indices in the range of the data rows. Trying to use similar call to nlsralt did NOT work – subset was ignored even when put in the calling args. Difficult to see how nls() manages this feature.
- BELIEVE(??) that the subsetting is done via model.frame using match.call() in nls() (nlspkg::nls.R#497), but mechanism does not seem clear. However, we can do a much simpler job by doing the subset within nlfb(x)
- Note that `subset` does NOT seems to be explicit in the code of `nlspkg` nor in the R-devel source tarball in a way that seems obvious. Where is it?? IDEAS: We could incorporate subset into weights and ALWAYS use weights. This may be inefficient in general, but easier and possibly safer to implement. That is, weights (and square roots) are initialized to 1, then set 0 for not-in-subset. Another, and possibly parallel, approach is to build a wrap-subset. Part of the difficulty in subsetting is that users and programmers can get confused. wrap-subset would be useful if the active problem is much smaller than the original data.
- nls has generic "residuals" function for nls() objects. residuals(myfit). Note that myfit*m*resid() returns WEIGHTED residuals (weighted by square roots of the weights) Note that getAllPars() add the plinear parameters in nlsModel.plinear, but otherwise is the same as getPars()

[2021-7-1] to [2021-7-4] nlsj and tentative program for rest of formal project

July 4: discussed tentative steps. JN made some progress with nlsj, but still some issues with the m$functions not finding the correct inputs, particularly the residual function does NOT seem to use the correct parameters. Seems that parameters are in both a vector `prm` and individual named variables, e.g., for Croucher `p1` and `p2`.

Suggested directions for latter half of project. Should agree approximate timeline for finishing items.

- Continue diary so we have a good record of what has been done. [Aug 31]
- AB: continue "testing" review document TestsDoc [July 16] [Added July 4:] JN realizing that building tests for different aspects of nls() (and, by natural extension, nlsr, minpack.lm etc.) is a good way for AB to learn about nonlinear modeling. But the current tests largely are "just there," so it would be useful to add comments that tell what the test aims to check. For example, we could be checking simple bounds, or a formula that includes a special function (this will defeat analytic derivatives in nlsr and future nls()).

- AB: review DerivsNLS, VarietyinNonlinearLeastSquaresCodes MachineSummary and PkgFromRbase documents and report any unfinished / erroneous / clumsy parts [July 16]
- JN: review TestsDoc [July 21]
- Both: List tests for nls() and its substitutes with explanation of why they are included. [Aug 1]
- JN: continue nlsj package. This is a "from zero" attempt to mimic nls(), but using a different set of routines and model object "m" [July 21 to first version]
- Both: Try to clear all "??" issues and check working of nlsj. Prepare: "nlsj – a substitute for nls()" [July 28]
- Both: Try to document where issues remain in nlsalt package, which is an attempt to put nls() (i.e., nlspkg) entirely in R and to document and clean up. [Aug 7?? Probably won't be fully finished]
- Clean up repo, removing incomplete or unhelpful elements AB: Look into how Google wants things put away. https://github.com/rstats-gsoc/gsoc2020/wiki/table-of-proposed-coding-projects [Aug 15??]

The following is from an email to Patrice Kiener, a co-mentor on the Google Summer of code NNBenchmark projects for 2019 and 2020.

```
The nls() stuff is going much more slowly than I'd like. I knew when we started that
the student is willing and generally capable but not very experienced in R internals.
But then, my own experience is strong on nonlinear optimization, but not so much on
the quirkiness of R + C interaction. Heather and I decided to try to segment the work
so there are some sub-projects that can be completed, though they might not be
directly on the original topic. I'm leading the code digging, and getting Arkajyoti
to try to follow and ask questions to force us to keep honest. This is working quite
well, and between us we are learning and moving the frontier forward. However, as my
doctoral supervisor (C. A. Coulson FRS) said "pi is the ratio of how long a task
takes over how long you thought it would take".

The big obstacle at the moment is a key routine created by nls() in its "model"
object "m" called "setPars". Here is the output of fit <- nls([a simple problem])


fit$m$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: 0x55f3e6c05cb0>
<environment: 0x55f3e6c27360>
>

You'll note:
 1) it is recursive
 2) it creates a bunch of functions and objects, but only returns a single LOGICAL
 3) if you run nls() finding QR and other objects just doesn't seem possible easily
   (I haven't been able to find them! But they are the key solution item.)
 4) "global" assignments are used "<<-"
 5) "hidden" objects are used (.swts)
 6) if the problem has nonlinear parameters in the lhs, then this fails, because it
```

```
    only builds the jacobian for the rhs of the model. Problems like
    that aren't traditional in statistics, but can arise in optimization.

So a real mess. I've been in contact with Doug Bates, who is supposedly the author and who
said he would help if he could. He switched to Julia a few years ago. He can't recall why
the recursion is needed. But if you run the function outside of nls(), it blows the C
language stack! Sigh.

I plan to refactor, eliminating the recursion and globals and use a proper list() return
so all objects are in known locations. However, that will certainly cause us some pain when
it comes to profile.nls() and possibly some other features. And truthfully, I'd like to
understand the code, even if we replace it. My intention is to continue for a few more
days to run calls of the function under different initial conditions to create a
"reproducible example" (I think reprex is becoming the abbreviation) so I can post on
r-devel and see if there is anyone who can help to decode this. I strongly suspect that
the R-core maintainers are praying that no changes in the R and C infrastructure will
break the current code, as I'm pretty sure there's no developer documentation to provide
a guide.

On the bright side, I have got numericDeriv() function all in R. It's a bit slower than the
mixed R+C, but much shorter and more easily maintained and adapted -- the plan is to offer
the analytic derivatives as an option. Also to change the name to "jacobian" or "jacobian.nls".
And we've drafts of several short articles on some issues tangential to the project,
including one I'm preparing on the variety of ways nonlinear least squares can be
structured and implemented. Also one on a neat idea from Duncan Murdoch on how to extract
base-R code into a package that can be modified, thereby avoiding rebuilds of full R.
```

In that JN has NOT been able to satisfactorily understand the structure of setPars() and setVarying() (the latter produces a huge set of functions, and it is not clear how it has been defined), I decided to do a zero-base build of a new set of functions in a package called `nlsj`. This will hopefully have fully documented functions and features. However, I expect that it will need quite a lot of work to add the equivalent of `profile.nls()` and similar features. We note, from the `nls()` documentation, that

```
An nls object is a type of fitted model object. It has methods for the generic
functions anova, coef, confint, deviance, df.residual, fitted, formula, logLik,
predict, print, profile, residuals, summary, vcov and weights.
```

Thus we will have to build equivalent capability, but we can possibly avoid a lot of the extraneous material in the current infrastructure and also document what we do provide.

[2021-6-29] and [2021-6-30] Struggling with nls() structure

- JN: worked on the Croucher-expandednlsnoc.R file. A number of issues related to the setPars() function. This was expanded to try to work around the issue that it is recursively defined (why?). Then troubles with getRHS(), which does not seem to be exposed for access in the top level code. Moreover, issues with .swts, which is hidden (why? again).
- JN: discussion with Paul Gilbert about recursive calls and global assignment in nlsModel and setPars. Email to Doug Bates got rather discouraging reply that Doug unable to recall or figure out why setPars is recursive.
- setPars(): search shows it is in nls-profile() as well as nls(). The calls to it are unusual, first with no argument, then with an argument. We need to understand how it works, though nls-profile is a tool that may not be sensible to keep in R. However, “profile” is a tool in R, and “profile.nls” is documented, with an example.
- note also that setPars() works with gradient in rhs only. This is NOT OK when lhs and rhs have

parameters, which can be the case when we are not modeling a single variable. Should use `resid`.

[2021-6-28] continuing Croucher-expanded

- JN: found that minpack.lm version of conv could be put into code and appears to function correctly. It is much simpler than code in nlspkg::nlsModel(). Still need to sort out setPars.
- Online meeting – some issues with git, Google Meet. Discussed diary portion of this document, some issues related to testing, and plans for this week. Agreed to meet Sunday July 4 to avoid collision with UseR!2021.

[2021-6-26] and [2021-6-27]

- expanded VarietyInNonlinearLeastSquaresCodes substantially
- started unrolling nls code in Croucher examples. Some issues with if ... else and brackets, possibly due to last "}" in function definition of nls().
- Note the strange "%||%" – comes from `utils` package via base-internals.R in `nlspkg` or `nlsalt` BUT looks like a language construct. ??Should we replace with something inline so this does not trip up other programmers. In `minpack.lm` the structure used is

```
env <- environment(formula)
if (is.null(env)) env <- parent.frame()
```

- AB: looking up suitable examples in NRAIA package[23 June]
- AB: looking up various testing methods in R [21-28 June]
- AB: working on an Rmd file, documenting what testing strategies might be taken up for our project [26-27 June] (Made a branch, but I don't see a new branch created in the repo. May have forgotten how to do this. Will look into it.

[2021-6-23] [2021-6-25] Tidying and documenting nlsModel

- JN cleaned up DerivsNLS. Extracted ssasymptot model and tested in nlsalt. numericDeriv must be working, but not explicitly tested. Note that the encapsulated version in the examples of nls() seem to fail.??
- used file inport of nls.R and nls.c parts (which are now separated out in improvnls)
- nlsModelDoc proceeding slowly. There are many undocumented features and pieces of code in nlsModel() that need careful understanding, particularly the scoping assignment "«-" in some lines of the code. In `minpack.lm` there is a version of nlsModel() that uses `assign` rather than `<<-`, and some documentation points out that this may be a more explicit mechanism (and therefore easier to understand and keep correct). Note the R documentation `?assignOps` as well as the `assign()` and `get()` functions.
- created nlsModelx.R file with nlsModelx(), and then added print statements to allow understanding of the function and build developer documentation.
- AB: looking up some tutorials on NLS with the aim of writing an article/tutorial on NLS, with the use of nls_new[24-25 June]

[2021-6-22] Answering some of the TODOS / nls-noc.R / nlsModelDoc.Rmd

- Do we need to update nlsalt and nlspkg separately from improvenls on a machine when there are changes imported via git? Answer is that git within improvenls directory seems sufficient.
- response from Doug Bates. Added him to gitlab project.
- Expanded document nlsModelDoc.Rmd to show what different objects created actually do. There are still plenty of questions, especially about `setPars()`.

- AB: in our documents, I am currently copying and pasting material. However, it would be sensible to do an "include" of the material from the nlsalt package, but to do that cleanly, we will want to separate the functions into different files. This will mean we have to be careful that the package still builds OK.

https://towardsdatascience.com/rtest-pretty-testing-of-r-packages-50f50b135650

https://towardsdatascience.com/unit-testing-in-r-68ab9cc8d211

https://r-pkgs.org/tests.html

JN contacted Doug Bates who indicated he may be able to provide some input on the code. There does not appear to be any developer documentations. Ongoing work with nlsModel() seems to indicate that the "m" object is set up to provide all the infrastructure for the rest of the code to work with for a particular nonlinear model.

- AB: reading up parts of chapter 2 and 3 of Bates and co's NRAIA book[21-22 June]

Week of [2021-6-21]

Online meeting on June 21 talked about `nls-noc.R` as well as testing schemes. AB will try to develop a short appraisal of testing schemes and suggest a path for us. Some discussion of tools.

JN started `nlsModelDoc.Rmd`. There are many functions, seemingly without documentation. Also quite complex codes and reuse of names. In some codes the "«-" assignment is used. We will need to be very careful to document and understand this.

[2021-6-21] Meeting etc.

Met online.

- discussed testing schemes. AB will see if he can develop a short appraisal and suggestions for our use.
- talked about `nls-noc.R` process. This is ongoing by JN
- later in day, JN contacted Doug Bates to ask if there are developer notes or other information, as there seem to be extra features in the nlsModel() output and the nls_iter code in nls.c

[2021-6-15] - [2021-6-20]: nlsalt issues

1) Need to ensure the name "nlsalt" is fixed DESCRIPTION, NAMESPACE (dynload line), .gitignore (.so line can be *.so), init.c (dynload line, R_init line)
2) appears new all-R numericDeriv not quite right for all nls examples. Probably (to be determined??) whether example uses a function or expression in the definition of the model.

During this time did some work on splitting nls.R into separate files where major functions were in their own file. In particular, nlsModel() was separated out so that nlsModel and nlsModel.plinear did not get confused in editing, since they are largely the same.

The file `nls-noc.R` has the ongoing work to try to build an all-R nls_iter function. This is gradually proceeding, but not functional at the time of our June 21 online meeting.

JN took a brief look at 2010 UseR! tutorial on nonlinear models. This may give us some ideas for documentation and tests.

[2021-6-12] –> [2021-6-14]

Document PkgFromRbase written to document the packaging effort and report on the Makevars.win issue.

JN diverted to update optimx package.

[2021-6-15] chunk code from file

The **R Markdown Cookbook** is quite useful in giving hints on some features of R Markdown. In particular, sections 16.1 to 16.3 of

https://bookdown.org/yihui/rmarkdown-cookbook/

show how to use R scripts saved in files within chunks of R Markdown documents. This is important, as we may wish to list such files, but copying the script into the text of the document means that this copy and the original file can become unsynchronized.

From work on the Gramps genealogy software (and some local setup issues), JN found that the following chunk will allow listings in a nice format. Note the "comment=NA" to avoid line prefixing with hash symbols. The code which follows needs surrounding by the usual three ticks before and after.

```
{r lscript, comment=NA, echo=FALSE}
cat(readLines('/home/gramps/GrampsSetup/FrohnGramps'), sep = '\n')
```

[2021-6-6] –> [2021-6-11]

Had online meeting and got Rstudio more or less set up to deal with version control and package building, BUT ...

hit a big problem with 32 bit Windows architecture NOT linking to BLAS. A posting to R-package-devel list got a suggestion from Dirk Eddelbeutel to check sessionInfo() and this shows that on the Windows R side, BLAS and LAPACK are not listed, while they are in Linux. Duncan Murdoch pointed to makevars.win, but severeal tries with some possible PKG_LIBS settings did not give any change. Noted that command line CMD.exe use of "R CMD build" and "R CMD INSTALL" showed the issue to be at the INSTALL stage, and it is clear that the 32 bit architecture is NOT finding the BLAS routines, but

Then found https://stackoverflow.com/questions/42118561/error-in-r-cmd-shlib-compiling-c-code and this shows what to use for makevars.win, which is the single line file

PKG_LIBS = $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS)

[2021-6-6]

Wanted package benchmarkme.

install failed. Needed dependencies.

So in linux, install libssl.dev libcurl4-openssl-dev

Then benchmarkme installed OK.

We can use this within our benchmarking in addition to sessionInfo().

[2021-6-5]

- created script `tnlsModel.R` as first step to create and produce an "m" object
- 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.
- Started document nlsModel.Rmd to document this function. ?? AB: in our documents, I am currently copying and pasting material. However, it would be sensible to do an "include" of the material from

the nlsalt package, but to do that cleanly, we will want to separate the functions into different files. This will mean we have to be careful that the package still builds OK.

From `nlsr` vignette `nlsr-devdoc.Rmd` [2021-5-23 onwards]

[2021-6-4]

- Code for `numericDeriv()` in nls.R was extracted to `numericDeriv20210603CommentedWithC.txt`.
- The code for numeric_deriv was removed from nls.c
- The line in init.c that mentions numeric_deriv was commented out so it would not be linked into the shared library for nlsalt (nlsalt.so)
- By launching Rstudio from the nlsalt.Rproj file, the package `nlsalt` could be rebuilt using "clean and rebuild." nlsalt.Rproj can be initiated from the "Files" menu of Rstudio. Note that rebuild is probably necessary on each machine on which nlsalt is used. (?? Do we need to check this is the case?)

[2021-6-2] Converting codes to All-R – numericDeriv

In the `ExDerivs.R` script, JN tested a forward difference code in R against a call to numericDeriv(). On a single example, it tests OK, so next step is to modify nls.R in package `nlsalt`.

Note that `ExDerivs.R` is in flux. "Old" tests will be commented out. Look for the double question mark (??) for things that may need to be checked.

Some concerns:

- the code requires us to get() and assign() values in the environment `rho` that is used to evaluate the nonlinear residual functions. There may or may not be ways to make this more efficient.
- the C code uses explicit loops. A very preliminarly use of R's vectorization in the difference between two residual vectors (i.e., without subscripts or looping) worked fine, but there are almost certainly further optimizations. By contrast, we MAY want to leave either explicit loops or commented code that contains such loops as a way to provide explanation of what is happening.

The code from the trials in ExDerivs.R was ported to nls.R in nlsalt package and tested with TestnumericDeriv1.R (added to repo). Things seem to work OK. I have NOT tried changing `dir`. ??

[2021-6-1] nlsalt package mods started

numericDeriv first cut at pure R. JN incorporated C code into the R code of the numericDeriv function as comments, then tried to provide R statements to do what it appears the C is doing. Some steps and checks likely left out.

[2021-6-1] nlspkg package created

Thanks to Duncan Murdoch who created nlspkg from the R source files. This is a package version of the files in the source tree that will let us work with the code and change it. When we issue

```
library(nlspkg)
```

once we have installed it, the objects in the package will replace (mask) those in base-R.

To build nlspkg, I needed the inconsolata.sty font that is in the `texlive-fonts-extra` package of Linux Mint. In other systems, this font may reside elsewhere.

As of June 1, the package builds and mostly checks. There are two warnings:

- a number of function objects are NOT documented. We may or may not wish to add `man` files (of type `.Rd`). The list of objects is quite long, but they seem not to be needed by the user. Note that nlminb is present, likely for the PORT option of the "algorithm" parameter of `nls()`.

- There is a non-standard license specification. Checking in the Writing R Extensions manual (file r-exts.pdf), suggests dropping one of the two licenses listed (GPL-2 and GPL-3). Dropping the latter removed one warning.

The package `nlspkg` is to provide the EXISTING nls() functionality, while the package `nlsalt` is for our replacement. I plan to make this all-R. If we want to use C (or something else) later, then we will create another "alternate" package.

Note that each of these packages needs its own "project." In Rstudio we choose to create a project using an existing directory (copy of that for `nlspkg`). This sets up the control files for the project and allows the package to be built and checked with the Rstudio tools. The umbrella ImproveNLS project handles the **git** version control.

[2021-5-31] Notes on investigations

1) AB managed to get R built from an R-devel expanded tarball. JN had managed this earlier. There appear to be some transient glitches (HT has noted also). These are likely due to R-core members making trial changes. MASS package in particular noted.

2) Investigations of using subversion or git repo of the code so we can track any changes in files relevant to nls(). JN tried from https://github.com/wch/r-source/wiki which has some instructions. These seemed to work, but we note the need for texinfo and git-svn packages to be installed. Also standard "git pull" cannot be used. "git sv rebase" followed by "git svn fetch" tried and seem to work, but JN does not feel he understands what is going on.

3) Started to look into development of replacement code for parts of nls(). Created ReplaceNLS.R and RestoreNLS.R with the intention that source() of these files would replace the functions. However, first attempt, which simply added a cat() output to numericDeriv() function got

```
Error in numericDeriv(rexpr, theta, rho = weedenv) :
  object 'C_numeric_deriv' not found
```

[JN: 2021-5-19] Some thoughts on how nls() code might be modified

- R-core will veto any changes that do not leave existing tests and examples as they are. Therefore we will be always thinking of ways to leave the default behaviour as it is now. In the relative offset convergence test that was causing small-residual problems to fail, JN suggested a small change to the test. In very simplified terms, the test is a ratio of `(new sum of squares)/(baseline sum of squares)` When both these quantities are small, we risk zero/zero situation. Therefore, a new control parameter `convTeestAdd` that has a default of 0 was introduced. This can be set in the `nls.control()` function. It is added to the denominator and allows the `nls()` function to complete satisfactorily when we have problems where the fit is nearly exact. Thus we are seeking similar modifications that are invoked by setting new control parameters to non-default settings to get new behaviour.

- In the material below on implementing nonlinear least squares, it should be possible to have a zero valued default for the Marquardt parameter λ. As long as this zero value is preserved, the legacy behaviour is used. Our decisions need to be careful in situations where we really do need the Marquardt stabilization. That is, should we simply fail (legacy behavior), should we override the legacy behaviour, since the user almost certainly does not want an avoidable failure, or should we seek an additional control that allows for more nuanced outcomes. Clearly thought is needed, and we will need to find some examples to illustrate our arguments.

- Note that nls() is a mix of several codes, using different solution tools. We will need to decide where our modifications apply and document carefully. JN opinion: we may want to postpone consideration of "port" and "plinear" algorithms, but it would be useful to document what they do. The current documentation suggests that bounds on parameters can only be used with "port."

- The derivative code to generate the Jacobian in nls() is (JN believes??) a simple forward-difference code. This may be in C rather than R. It is well-known that central differences, while they cost twice the computing effort, do a better job. Can we find a way to allow that easily?
- If there are bounds on parameters, the "step" taken in estimating differences may violate bounds. This is an issue well-known to workers in optimization, but not on the radar of most users. Clearly a "nasty." Can we do anything to at least warn users in a way that is useful?

2021-5-18

By email, a Google Meet was set up for May 19 at noon Ottawa time. JN downloaded the latest version of `R-devel.tar.gz` and unpacked it into a directory `~/vmshare/R-devel` which is shared with a VirtualBox VM of Kubuntu 20.04 (Focal Fossa), a long-term support version of Linux. Opening a terminal inside this VM, it was possible to build and run this version of R.

```
cd /media/sf_vmshare/R-devel/
./configure
make
sudo make install
[admin password]
R
```

This launched the development version of R correctly.

JN also set up this document.

Agenda for May 19

- check Meet is working
- introductions
- start linux VM install if an iso is available. May want to check that VirtualBox Guest additions is available and that the shared directory works, as these sometimes require some attention to permissions and ownership etc.
- Consider early goals to for possible nls() changes. See below.
- Set objectives for next two weeks
- Set next online meeting

Possible early goals

- Get a VM running under VirtualBox and install build tools. See https://support.rstudio.com/hc/en-us/articles/218004217-Building-R-from-source I did NOT need more than ./configure in my build, as I am not building a server version.
- Try the build. (Cheer loudly when it works!)
- Explore and document the R source for nls-related code. In particular, we want
  - to list the files that have such code or calls to it
  - to note, if possible, what each does
  - to note, in particular, where nls() solves the Gauss-Newton equations, as we will want to modify these sections to augment them to allow a Marquardt stabilization.
  - to note, in particular, where nls() computes the Jacobian and/or Hessian for the nonlinear least squares problem. In package `nlsr`, there are tools that allow an expression for the model to be parsed and processed to compute the Jacobian using symbolic or automatic derivatives. This may or may not be feasible for `nls()`. However, we may be able to improve the approximation used.

[2021-5-27] first draft DerivsNLS.Rmd. Note discovered "central" control for numericDeriv() function. It does NOT appear that the 'dir' control is actually used. This would likely change from forward to backward differences for SOME components of the step to compute first order finite difference approximation to the Jacobian.

- Augment this document to record what has been done and results or problems.
- Add to the bibliography file specified. This is taken from another work, but as Rmarkdown only uses the references it needs, it will serve as a template.
- AB can ask for pointers to references to add to this document or to subsidiary documents we will create as necessary to describe parts of the work if required.

[2021-5-25] Meet session. AB got Linux Mint 20.1 VM going and verified. SOme questions sorted out. JN added some material re: linear least squares solutions to the Variety document.

==========end diary only portion ==========

5. Implementation of nonlinear least squares methods

This section is a review of approaches to solving the nonlinear least squares problem that underlies nonlinear regression modelling. In particular, we look at using a QR decomposition for the Levenberg-Marquardt stabilization of the solution of the Gauss-Newton equations.

Gauss-Newton variants

Nonlinear least squares methods are mostly founded on some or other variant of the Gauss-Newton algorithm. The function we wish to minimize is the sum of squares of the (nonlinear) residuals $r(x)$ where there are m observations (elements of r) and n parameters x. Hence the function is

$$f(x) = \sum_i ({r_i}^2)$$

Newton's method starts with an original set of parameters $x[0]$. At a given iteration, which could be the first, we want to solve

$$x[k+1] = x[k] - H^{-1}g$$

where H is the Hessian and g is the gradient at $x[k]$. We can rewrite this as a solution, at each iteration, of

$$H\delta = -g$$

with

$$x[k+1] = x[k] + \delta$$

For the particular sum of squares, the gradient is

$$g(x) = 2 * r[k]$$

and

$$H(x) = 2(J'J + \sum_i (r_i * Z_i))$$

where J is the Jacobian (first derivatives of r w.r.t. x) and Z_i is the tensor of second derivatives of r_i w.r.t. x). Note that J' is the transpose of J.

The primary simplification of the Gauss-Newton method is to assume that the second term above is negligible. As there is a common factor of 2 on each side of the Newton iteration after the simplification of the Hessian, the Gauss-Newton iteration equation is

$$(J'J)\delta = -J'r$$

This iteration frequently fails. The approximation of the Hessian by the Jacobian inner-product is one reason, but there is also the possibility that the sum of squares function is not "quadratic" enough that the unit step reduces it. Hartley (1961) introduced a line search along delta, while Marquardt (1963) suggested replacing $J'J$ with $(J'J + \lambda * D)$ where D is a diagonal matrix intended to partially approximate the omitted portion of the Hessian.

Marquardt suggested $D = I$ (a unit matrix) or $D =$ (diagonal part of $J'J$). The former approach, when λ is large enough that the iteration is essentially

$$\delta = -g/\lambda$$

gives a version of the steepest descents algorithm. Using the diagonal of $J'J$, we have a scaled version of this (see https://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm; Levenberg (1944) predated Marquardt, but the latter seems to have done the practical work that brought the approach to general attention.)

Nash (1977) found that on low precision machines, it was common for diagonal elements of $J'J$ to underflow. A very small modification to solve

$$(J'J + \lambda(D + \phi I))\delta = -g$$

where ϕ is a small number avoids most of these conditions. $\phi = 1$ seems to work quite well. We note that this modification would likely not have been recognized had I not been working on machines with mediochre floating-point arithmetic – a little more than 6 decimal digits of precision and no extended precision.

Choices

Both `nlsr::nlxb()` and `minpack.lm::nlsLM` use a Levenberg-Marquardt stabilization of the iteration described above, with `nlsr` using the modification involving the ϕ control parameter. The complexities of the code in `minpack.lm` are such that I have relied largely on the documentation to judge how the iteration is accomplished. `nls()` uses a straightforward Gauss-Newton iteration, but rather than form the sum of squares and cross-products, uses a QR decomposition of the matrix J that has been found by a forward difference approximation.

(Nash (1979)), solving

$$(J^T J + \lambda D)\delta = -J^T r$$

where D is some diagonal matrix and lambda is a number of modest size initially. Clearly for $\lambda = 0$ we have a Gauss-Newton method. Typically, the sum of squares of the residuals calculated at the "new" set of parameters is used as a criterion for keeping those parameter values. If so, the size of λ is reduced. If not, we increase the size of λ and compute a new δ. Note that a new J, the expensive step in each iteration, is NOT required.

As for Gauss-Newton methods, the details of how to start, adjust and terminate the iteration lead to many variants, increased by different possibilities for specifying D. See Nash (1979).

Using matrix decompositions [2021-5-23 onwards]

In Nash (1979), the iteration equation was solved as stated. However, this involves forming the sum of squares and cross products of J, a process that loses some numerical precision. A better way to solve the linear equations is to apply the QR decomposition to the matrix J itself. However, we still need to incorporate the λI or λD adjustments. This is done by adding rows to J that are the square roots of the "pieces." We add 1 row for each diagonal element of I and each diagonal element of D.

In each iteration, we reduce the λ parameter before solution. If the resulting sum of squares is not reduced, λ is increased, otherwise we move to the next iteration. Various authors (including the present one) have suggested different strategies for this. My current opinion is that a "quick" increase, say a factor of 10, and a "slow" decrease, say a factor of 0.2, work quite well. However, it is important to check that λ has not got too small or underflowed before applying the increase factor. On the other hand, it is useful to be able to set lambda = 0 in the code so that a pure Gauss-Newton method can be evaluated with the program(s). The current code `nlfb()` uses the line

```
if (lamda<1000*.Machine$double.eps) lamda<-1000*.Machine$double.eps
```

to ensure we get an increase. To force a Gauss-Newton algorithm, the controls `laminc` and `lamdec` are set to 0.

The Levenberg-Marquardt adjustment to the Gauss-Newton approach is the second major improvement of `nlsr` (and also its predecessor `nlmrt` and the package `minpack-lm`) over `nls()`.

`nls()` attempts to minimize a sum of squared residuals by a Gauss-Newton method. If we compute a Jacobian matrix J and a vector of residuals r from a vector of parameters x, then we can define a linearized problem

$$J^T J\delta = -J^T r$$

This leads to an iteration where, from a set of starting parameters x_0, we compute

$$x_{i+1} = x_i + \delta$$

This is commonly modified to use a step factor `step`

$$x_{i+1} = x_i + step * \delta$$

It is in the mechanisms to choose the size of *step* and to decide when to terminate the iteration that Gauss-Newton methods differ. Indeed, though I have tried several times, I find the very convoluted code behind `nls()` very difficult to decipher. Unfortunately, its authors have now (as far as I am aware) all ceased to maintain the code.

We could implement the methods using the equations above. However, the accumulation of inner products in $J^T J$ occasions some numerical error, and it is generally both safer and more efficient to use matrix decompositions. In particular, if we form the QR decomposition of J

$$QR = J$$

where Q is an orthonormal matrix and R is Right or Upper triangular, we can easily solve

$$R\delta = -Q^T r$$

for which the solution is also the solution of the Gauss-Newton equations. But how do we get the Marquardt stabilization?

If we augment J with a square matrix $\sqrt{\lambda D}$ whose diagonal elements are the square roots of λ times the diagonal elements of D, and augment the vector r with n zeros where n is the column dimension of J and D, we achieve our goal.

Typically we can use $D = I_n$ (the identity of order n), but Marquardt (1963) showed that using the diagonal elements of $J^T J$ for D results in a useful implicit scaling of the parameters. Nash (1977) pointed out that on computers with limited arithmetic (which now are rare since the IEEE 754 standard appeared in 1985), underflow might cause a problem of very small elements in D and proposed adding ϕI_n to the diagonals of $J^T J$ before multiplying by λ in the Marquardt stabilization. This avoids some awkward cases with very little extra overhead. It is accomplished with the QR approach by appending $\sqrt{\phi\lambda}$ times a unit matrix I_n to the matrix already augmented matrix. We must also append a further n zeros to the augmented r.

[2021-5-27] JN added some notes on LU and Cholesky to "Variety" document.

======== End of extract on approaches to solving Gauss Newton equations =========

Early explorations and setup of project [2021-5-24]

This is essentially a diary entry. (?? AB you may want to add some notes too.)

- AB working to secure his computer, set up VirtualBox and prepare a linux guest VM
- AB has been looking at documents JN posted. Gitlab repo seems to be working OK. On comment from HT, JN discovered (very overdue!) that Rstudio support for git is very useful.
- JN set up nls-flowchart.txt for use by team. This is looking at R-devel.tar.gz downloaded May 18, 2021. It is likely we can use any recent version, though at some point we may need to be more precise. The purpose of this is to describe what is going on in the nls() operations.
- JN added document **VarietyInNonlinearLeastSquaresCodes.Rmd**. This is tangential to the project, but is an attempt to provide a panorama of the different ways in which nonlinear least squares software has been set up over the years. JN will respond to questions from other team members to try to make this explanatory document helpful in understanding the existing nls() code and proposed changes.
- Section added above "From `nlsr` vignette `nlsr-devdoc.Rmd`"
- In the "VarietyInNonlinearLeastSquaresCodes.Rmd" document, JN added a section on different approaches to solving LINEAR least squares problems, since the inner iteration of almost all nonlinear least squares methods involves such a sub-problem. AB has been looking at this. It is currently a messy DRAFT section at 2021-5-25.
- Meet scheduled for May 25;

Attempts to understand numericDeriv() [2021-5-24 to 27]

JN has found it quite difficult to get a good understanding of the `numericDeriv()` function that appears in `nls.R` and calls material in `nls.c`. That this has proved difficult to use suggests the documentation is less than wonderful. However, the main reason is to set up tests and timings and compare to other approaches to computing the Jacobian in nonlinear least squares. ??Gut feeling: numericDeriv is really only set up to compute a Jacobian. ?? Note: there is a "central" option. Can this be used with nls()? Why or why not?

A file ExDerivs.R has been added to the repo.

[2021-5-26] JN sorted out one approach that allows numricDeriv, nlsr::model2rjfun and numDeriv::jacobian to be compared. [2021-5-27] DerivsNLS.Rmd first draft done. Problems with system() and system2() and calling bash from rmarkdown document led to email sent to Yihui Xie, as the output of a Linux command (inxi -F) to get machine properties was returned with extra characters.

Tangential issues raised during the project

Output from `bash` scripts in rmarkdown files

In trying to capture the output of the Linux command-line program `inxi` (which reports hardware and software information), JN found rmarkdown failed to knit correctly.

```
system('inxi -F >t.txt')
```

and

```
system2("inxi -F >tlinux.txt")
```

both failed, as did specifying the chunk as `{bash inxi2}` and trying `inxi -F >t2.txt`.

```
The outputs had strange characters that were indecipherable by the LaTEX processor
and caused a crash. Indeed files of substantially differing sizes were generated.

We also tried
```

system2("inxi," "-F," stdout="tlinux2.txt")

```
The issue was reported on community.rstudio.

\url{https://community.rstudio.com/t/bash-output-causes-trouble-for-latex-and-hence-no-pdf/106502}

where a workaround is reported. This puts the inxi command in a script which uses the preamble
`/#!/bin/sh`. That is, we use the `sh` shell. For example, let us call it myscript.sh and it
contains
```

#!/bin/sh # myscript.sh – push hw info to thw.txt one directory above inxi -F >../thw.txt echo "done"

```
This command script is then run from R using
```

system("mate-terminal -e ./myscript.sh") "'

JN tested this with mate-terminal, xterm and konsole as the terminal programs.

References

Hartley, H. O. 1961. "The Modified Gauss-Newton Method for Fitting of Nonlinear Regression Functions by Least Squares." *Technometrics* 3: 269–80.

Levenberg, Kenneth. 1944. "A Method for the Solution of Certain Non-Linear Problems in Least Squares." *Quarterly of Applied Mathematics* 2: 164--168.

Marquardt, Donald W. 1963. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." *SIAM Journal on Applied Mathematics* 11 (2): 431–41.

Nash, John C. 1977. "Minimizing a Nonlinear Sum of Squares Function on a Small Computer." *Journal of the Institute for Mathematics and Its Applications* 19: 231–37.

———. 1979. *Compact Numerical Methods for Computers : Linear Algebra and Function Minimisation.* Book. Hilger: Bristol.