Jacobian Calculations for nls()

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

26/05/2021

Jacobians in nls()

`nls()` needs Jacobians calculated at the current set of trial nonlinear model parameters to set up the Gauss-Newton equations. Unfortunately, `nls()` calls the Jacobian the "gradient", and uses function `numericDerivs()` to compute them. This document is an attempt to describe different ways to compute the Jacobian for use in nls() and related software, and to evaluate the performance of these approaches.

In evaluating performance, we need to know the conditions under which the evaluation was conducted. Thus the computations included in this document, which is built using `Rmarkdown`, are specific to the computer in which the document is processed. We will add tables that give the results for different computing environments at the bottom.

An example problem

We will use the Hobbs weed infestation problem (@jncnm79, page 120).

```
# 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))"
```

```
# Will be TRUE if we have made the conversion
print(as.formula(ceunsc)==eunsc)
```

```
## [1] TRUE
```

```
## LOCAL DATA IN DATA FRAMES
weeddata1  <-  data.frame(y=ydat, tt=tdat)
```

```
## Put data in an Environment
weedenv <- list2env(weeddata1)
weedenv$b1 <- start1[[1]]
weedenv$b2 <- start1[[2]]
weedenv$b3 <- start1[[3]]
# Display content of the Environment
## Note that may need to do further commands to get everything
ls.str(weedenv)
```

```
## b1 :  num 1
## b2 :  num 1
## b3 :  num 1
## tt :  int [1:12] 1 2 3 4 5 6 7 8 9 10 ...
## y :  num [1:12] 5.31 7.24 9.64 12.87 17.07 ...
```

```
# Generate the residual "call"
rexpr<-call("-",eunsc[[3]], eunsc[[2]])
# Get the residuals
r0<-eval(rexpr, weedenv)
print(r0)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
```

```
cat("Sumsquares at 1,1,1 is ",sum(r0^2),"\n")
```

```
## Sumsquares at 1,1,1 is  23520.58
```

```
## Another way
ldata<-list2env(as.list(start1),envir=weedenv)
ldata
```

```
## <environment: 0x55ccd6a03d20>
```

```
ls.str(ldata)
```

```
## b1 :  num 1
## b2 :  num 1
## b3 :  num 1
## tt :  int [1:12] 1 2 3 4 5 6 7 8 9 10 ...
## y :  num [1:12] 5.31 7.24 9.64 12.87 17.07 ...
```

```
eval(rexpr,envir=ldata)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
```

```
## Do we need to get a model frame? How? and How to use it?

## Now ready to try things out.
```

Tools for Jacobians

numericDeriv()

`numericDeriv` is the R function used by `nls()` to evaluate Jacobians for its Gauss-Newton equations. The R source code is in the file `nls.R`. It calls a C function numeric_deriv in `nls.c`.

```r
## seems to work -- Note file ExDerivs.R has many "failures"
theta <- c("b1", "b2", "b3")
ndeunsc<-numericDeriv(rexpr, theta, rho=weedenv)
print(ndeunsc)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
## attr(,"gradient")
##            [,1]          [,2]         [,3]
##  [1,] 0.7310585 -1.966119e-01 0.1966118813
##  [2,] 0.8807971 -1.049936e-01 0.2099871635
##  [3,] 0.9525741 -4.517674e-02 0.1355299950
##  [4,] 0.9820137 -1.766276e-02 0.0706508160
##  [5,] 0.9933071 -6.648064e-03 0.0332403183
##  [6,] 0.9975274 -2.466440e-03 0.0147991180
##  [7,] 0.9990890 -9.102821e-04 0.0063714981
##  [8,] 0.9996643 -3.356934e-04 0.0026817322
##  [9,] 0.9998765 -1.235008e-04 0.0011105537
## [10,] 0.9999547 -4.529953e-05 0.0004539490
## [11,] 0.9999828 -1.716614e-05 0.0001831055
## [12,] 0.9999943 -5.722046e-06 0.0000743866
```

```r
print(sum(ndeunsc^2))
```

```
## [1] 23520.58
```

```r
tndeunsc<-microbenchmark(ndeunsc<-numericDeriv(rexpr, theta, rho=weedenv))
print(tndeunsc)
```

```
## Unit: microseconds
##                                                       expr   min     lq    mean
##  ndeunsc <- numericDeriv(rexpr, theta, rho = weedenv) 9.152 9.3465 9.95542
##  median   uq    max neval
##  9.5465 9.87 38.283   100
```

```r
## numericDeriv also has central difference option, as well as choice of eps parameter
## Central diff
ndeunsc2<-numericDeriv(rexpr, theta, rho=weedenv, central=TRUE)
print(ndeunsc2)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
## attr(,"gradient")
##            [,1]          [,2]         [,3]
##  [1,] 0.7310586 -1.966119e-01 1.966119e-01
##  [2,] 0.8807971 -1.049936e-01 2.099872e-01
##  [3,] 0.9525741 -4.517666e-02 1.355300e-01
##  [4,] 0.9820138 -1.766271e-02 7.065082e-02
##  [5,] 0.9933071 -6.648057e-03 3.324028e-02
##  [6,] 0.9975274 -2.466509e-03 1.479906e-02
##  [7,] 0.9990889 -9.102211e-04 6.371548e-03
##  [8,] 0.9996647 -3.352378e-04 2.681902e-03
##  [9,] 0.9998766 -1.233799e-04 1.110414e-03
## [10,] 0.9999546 -4.539623e-05 4.539581e-04
## [11,] 0.9999833 -1.670090e-05 1.837134e-04
## [12,] 0.9999939 -6.143885e-06 7.372897e-05
```

```
print(sum(ndeunsc2^2))
```

```
## [1] 23520.58
```

```
tndeunsc2<-microbenchmark(ndeunsc2<-numericDeriv(rexpr, theta, rho=weedenv, central=TRUE))
print(tndeunsc2)
```

```
## Unit: microseconds
##                                                                         expr    min
##  ndeunsc2 <- numericDeriv(rexpr, theta, rho = weedenv, central = TRUE) 11.798
##      lq     mean  median      uq    max neval
##  12.109 12.88357 12.4465 12.7925 49.672   100
```

```
## Forward diff with smaller eps
ndeunscx<-numericDeriv(rexpr, theta, rho=weedenv, eps=1e-10)
print(ndeunscx)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
## attr(,"gradient")
##            [,1]          [,2]         [,3]
##  [1,] 0.7310597 -0.1966160568 0.1966071750
##  [2,] 0.8807977 -0.1049915710 0.2099920238
##  [3,] 0.9525714 -0.0451905180 0.1355182633
##  [4,] 0.9820056 -0.0176747506 0.0706457115
##  [5,] 0.9933032 -0.0066435746 0.0332534000
##  [6,] 0.9975309 -0.0024513724 0.0148148160
##  [7,] 0.9990941 -0.0009237056 0.0063593575
##  [8,] 0.9996626 -0.0003552714 0.0026290081
##  [9,] 0.9998757 -0.0001421085 0.0011368684
## [10,] 0.9999468  0.0000000000 0.0004973799
## [11,] 0.9998757 -0.0001421085 0.0001421085
## [12,] 1.0000178  0.0000000000 0.0001421085
```

```
print(sum(ndeunscx^2))
```

```
## [1] 23520.58
```

```
tndeunscx<-microbenchmark(ndeunscx2<-numericDeriv(rexpr, theta, rho=weedenv, eps=1e-10))
print(tndeunscx)
```

```
## Unit: microseconds
##                                                                       expr   min
##  ndeunscx2 <- numericDeriv(rexpr, theta, rho = weedenv, eps = 1e-10) 8.018
##      lq    mean median     uq   max neval
##  8.2245 8.68317 8.3175 8.4175 31.78   100
```

```
## Central diff with smaller eps
ndeunscx2<-numericDeriv(rexpr, theta, rho=weedenv, central=TRUE, eps=1e-10)
print(ndeunscx2)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
## attr(,"gradient")
##            [,1]          [,2]         [,3]
##  [1,] 0.7310597 -1.966116e-01 1.966116e-01
##  [2,] 0.8807977 -1.049916e-01 2.099876e-01
```

```
##  [3,] 0.9525714 -4.518164e-02 1.355271e-01
##  [4,] 0.9820145 -1.766587e-02 7.065459e-02
##  [5,] 0.9933032 -6.643575e-03 3.325340e-02
##  [6,] 0.9975309 -2.451372e-03 1.481482e-02
##  [7,] 0.9990941 -9.059420e-04 6.359357e-03
##  [8,] 0.9996981 -3.197442e-04 2.664535e-03
##  [9,] 0.9998757 -1.421085e-04 1.136868e-03
## [10,] 0.9999468 -3.552714e-05 4.618528e-04
## [11,] 0.9999468 -7.105427e-05 2.131628e-04
## [12,] 1.0000178  0.000000e+00 7.105427e-05
```

```
print(sum(ndeunscx2^2))
```

```
## [1] 23520.58
```

```
tndeunscx2<-microbenchmark(ndeunscx2<-numericDeriv(rexpr, theta, rho=weedenv, central=TRUE, eps=1e-10))
print(tndeunscx2)
```

```
## Unit: microseconds
##                                                                                          expr
##  ndeunscx2 <- numericDeriv(rexpr, theta, rho = weedenv, central = TRUE,      eps = 1e-10)
##     min      lq    mean  median      uq   max neval
##  10.987 11.2295 12.4357 11.4965 11.8545 47.38   100
```

We have not tried the dir parameter (probably allows backward differences)

Symbolic methods from nlsr

The package `nlsr` has a function `model2rjfun()` that converts an expression describing how the residual functions are computed into an R function that computes the residuals at a particular set of parameters and sets the **attribute** "gradient" of the vector of residual values to the Jacobian at the particular set of parameters.

```
# nlsr has function model2rjfun. We can evaluate just the residuals
res0<-model2rjfun(eunsc, start1, data=weeddata1, jacobian=FALSE)
res0(start1)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
```

```
# or the residuals and jacobian
## nlsr::model2rjfun forms a function with gradient (jacobian) attribute
funsc <- model2rjfun(eunsc, start1, data=weeddata1) # from nlsr: creates a function
tmodel2rjfun <- microbenchmark(model2rjfun(eunsc, start1, data=weeddata1))
print(tmodel2rjfun)
```

```
## Unit: microseconds
##                                           expr    min      lq     mean  median
##  model2rjfun(eunsc, start1, data = weeddata1) 82.329 84.7615 94.10805 86.1485
##       uq     max neval
##  90.4305 190.579   100
```

```
print(funsc)
```

```
## function (prm)
## {
##     if (is.null(names(prm)))
##         names(prm) <- names(pvec)
##     localdata <- list2env(as.list(prm), parent = data)
```

```
##     eval(residexpr, envir = localdata)
## }
## <bytecode: 0x55ccd4bea628>
## <environment: 0x55ccd5938700>
```

```
print(funsc(start1))
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
## attr(,"gradient")
##              b1            b2           b3
##  [1,] 0.7310586 -1.966119e-01 1.966119e-01
##  [2,] 0.8807971 -1.049936e-01 2.099872e-01
##  [3,] 0.9525741 -4.517666e-02 1.355300e-01
##  [4,] 0.9820138 -1.766271e-02 7.065082e-02
##  [5,] 0.9933071 -6.648057e-03 3.324028e-02
##  [6,] 0.9975274 -2.466509e-03 1.479906e-02
##  [7,] 0.9990889 -9.102212e-04 6.371548e-03
##  [8,] 0.9996646 -3.352377e-04 2.681901e-03
##  [9,] 0.9998766 -1.233793e-04 1.110414e-03
## [10,] 0.9999546 -4.539581e-05 4.539581e-04
## [11,] 0.9999833 -1.670114e-05 1.837126e-04
## [12,] 0.9999939 -6.144137e-06 7.372964e-05
```

```
print(environment(funsc))
```

```
## <environment: 0x55ccd5938700>
```

```
print(ls.str(environment(funsc)))
```

```
## data : <environment: 0x55ccd5928f50>
## jacobian :  logi TRUE
## modelformula : Class 'formula'  language y ~ b1/(1 + b2 * exp(-b3 * tt))
## pvec :  Named num [1:3] 1 1 1
## residexpr :   expression({  .expr3 <- exp(-b3 * tt)  .expr5 <- 1 + b2 * .expr3  .expr10 <- .expr5^2
## rjfun : function (prm)
## testresult :  logi TRUE
```

```
print(ls(environment(funsc)$data))
```

```
## [1] "tt" "y"
```

```
eval(eunsc, environment(funsc))
```

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

```
vfunsc<-funsc(start1)
print(vfunsc)
```

```
##  [1]  -4.576941  -6.359203  -8.685426 -11.883986 -16.075693 -22.194473
##  [7] -30.443911 -37.558335 -49.156123 -61.948045 -74.995017 -90.972006
## attr(,"gradient")
##              b1            b2           b3
##  [1,] 0.7310586 -1.966119e-01 1.966119e-01
##  [2,] 0.8807971 -1.049936e-01 2.099872e-01
##  [3,] 0.9525741 -4.517666e-02 1.355300e-01
##  [4,] 0.9820138 -1.766271e-02 7.065082e-02
##  [5,] 0.9933071 -6.648057e-03 3.324028e-02
```

```
##  [6,] 0.9975274 -2.466509e-03 1.479906e-02
##  [7,] 0.9990889 -9.102212e-04 6.371548e-03
##  [8,] 0.9996646 -3.352377e-04 2.681901e-03
##  [9,] 0.9998766 -1.233793e-04 1.110414e-03
## [10,] 0.9999546 -4.539581e-05 4.539581e-04
## [11,] 0.9999833 -1.670114e-05 1.837126e-04
## [12,] 0.9999939 -6.144137e-06 7.372964e-05
```

```
tfunsc<-microbenchmark(funsc(start1))
print(tfunsc)
```

```
## Unit: microseconds
##           expr   min      lq     mean  median    uq    max neval
##  funsc(start1) 13.13 13.3575 14.69758 13.5475 13.87 46.605   100
```

`numDeriv` package

The package `numDeriv` includes a function `jacobian()` that acts on a user function `resid()` to produce the Jacobian at a set of parameters by several choices of approximation.

```
# We use the residual function (without gradient attribute) from nlsr
jeunsc<-jacobian(res0, start1)
jeunsc
```

```
##            [,1]          [,2]         [,3]
##  [1,] 0.7310586 -1.966119e-01 1.966119e-01
##  [2,] 0.8807971 -1.049936e-01 2.099872e-01
##  [3,] 0.9525741 -4.517666e-02 1.355300e-01
##  [4,] 0.9820138 -1.766271e-02 7.065082e-02
##  [5,] 0.9933071 -6.648057e-03 3.324028e-02
##  [6,] 0.9975274 -2.466509e-03 1.479906e-02
##  [7,] 0.9990889 -9.102212e-04 6.371548e-03
##  [8,] 0.9996647 -3.352378e-04 2.681902e-03
##  [9,] 0.9998766 -1.233791e-04 1.110414e-03
## [10,] 0.9999546 -4.539572e-05 4.539580e-04
## [11,] 0.9999833 -1.670116e-05 1.837129e-04
## [12,] 0.9999939 -6.144205e-06 7.373002e-05
```

```
tjeunsc<-microbenchmark(jeunsc<-jacobian(res0, start1))
print(tjeunsc)
```

```
## Unit: microseconds
##                               expr    min      lq     mean  median      uq
##  jeunsc <- jacobian(res0, start1) 345.94 365.555 411.5162 382.754 429.985
##       max neval
##  1517.889   100
```

Note that the manual pages for `numDeriv` offer many options for the functions in the package. At 2021-5-27 we have yet to explore these.

Comparisons

In the following, we are comparing to `vfunsc`, which is the evaluated residual vector at `start1=c(1,1,1)` with "gradient" attribute (jacobian) included, as developed using package `nlsr`. This is taken as the "correct" result.

`numericDeriv` computes a similar structure (residuals with "gradient" attribute): `ndeunsc`: the forward difference result with default `eps` (1e-07 according to manual) `ndeunsc2`: Central difference with default `eps` `ndeunscx`: Forward difference with smaller eps=1e-10 `ndeunscx2`: Central difference with smaller eps=1e-10

`jeunsc`: numDeriv::jacobian() result with default settings.

```
## Matrix comparisons
attr(ndeunsc, "gradient")-attr(vfunsc,"gradient")
```

```
##                 b1            b2            b3
##  [1,] -4.066995e-08 -7.619266e-09 -5.198538e-08
##  [2,]  1.016833e-08  3.631656e-09 -7.263312e-09
##  [3,]  7.050552e-09 -8.473015e-08  1.577186e-08
##  [4,] -8.764533e-08 -5.738229e-08 -8.889419e-09
##  [5,] -3.542825e-08 -6.988878e-09  3.494439e-08
##  [6,] -1.592723e-08  6.909055e-08  6.229383e-08
##  [7,]  5.380365e-08 -6.095489e-08 -5.015294e-08
##  [8,] -3.432289e-07 -4.556886e-07 -1.691883e-07
##  [9,] -1.062480e-07 -1.214742e-07  1.395936e-07
## [10,]  9.833867e-08  9.627771e-08 -9.102750e-09
## [11,] -4.647158e-07 -4.649948e-07 -6.071033e-07
## [12,]  4.221287e-07  4.220910e-07  6.569545e-07
```

```
attr(ndeunsc2, "gradient")-attr(vfunsc,"gradient")
```

```
##                 b1            b2            b3
##  [1,] -5.513268e-11  1.371020e-11  5.962686e-11
##  [2,]  6.850076e-13 -3.831403e-11  3.291006e-12
##  [3,] -2.144829e-11 -1.208666e-10  6.925160e-11
##  [4,] -2.665634e-11  4.123477e-11 -1.826495e-11
##  [5,]  2.175706e-10 -7.825720e-11  9.793778e-11
##  [6,] -2.416068e-10 -1.666460e-10 -1.735172e-10
##  [7,] -8.350343e-11  5.415257e-11 -8.571976e-11
##  [8,]  3.255913e-10 -9.864836e-11  2.024904e-10
##  [9,]  1.379652e-11 -5.685668e-10 -1.631673e-10
## [10,] -9.296786e-11 -4.173983e-10  6.710748e-11
## [11,] -4.266865e-11  2.415372e-10  8.632699e-10
## [12,] -2.738492e-10  2.515432e-10 -6.717320e-10
```

```
attr(ndeunscx, "gradient")-attr(vfunsc,"gradient")
```

```
##                 b1            b2            b3
##  [1,]  1.078625e-06 -4.123528e-06 -4.758257e-06
##  [2,]  5.790545e-07  2.014411e-06  4.852963e-06
##  [3,] -2.771694e-06 -1.385826e-05 -1.171591e-05
##  [4,] -8.202081e-06 -1.204434e-05 -5.113350e-06
##  [5,] -3.931620e-06  4.482091e-06  1.311668e-05
##  [6,]  3.569890e-06  1.513685e-05  1.576029e-05
##  [7,]  5.191947e-06 -1.348438e-05 -1.219078e-05
##  [8,] -2.074929e-06 -2.003370e-05 -5.289324e-05
##  [9,] -8.676624e-07 -1.872920e-05  2.645423e-05
## [10,] -7.810096e-06  4.539581e-05  4.342184e-05
## [11,] -1.075608e-04 -1.254074e-04 -4.160402e-05
## [12,]  2.399048e-05  6.144137e-06  6.837890e-05
```

```
attr(ndeunscx2, "gradient")-attr(vfunsc,"gradient")
```

```
##                  b1            b2            b3
##  [1,]  1.078625e-06  3.173646e-07 -3.173646e-07
##  [2,]  5.790545e-07  2.014411e-06  4.120706e-07
##  [3,] -2.771694e-06 -4.976479e-06 -2.834131e-06
##  [4,]  6.797035e-07 -3.162555e-06  3.768434e-06
##  [5,] -3.931620e-06  4.482091e-06  1.311668e-05
##  [6,]  3.569890e-06  1.513685e-05  1.576029e-05
##  [7,]  5.191947e-06  4.279192e-06 -1.219078e-05
##  [8,]  3.345221e-05  1.549344e-05 -1.736611e-05
##  [9,] -8.676624e-07 -1.872920e-05  2.645423e-05
## [10,] -7.810096e-06  9.868671e-06  7.894701e-06
## [11,] -3.650654e-05 -5.435313e-05  2.945025e-05
## [12,]  2.399048e-05  6.144137e-06 -2.675369e-06
```

```
jeunsc-attr(vfunsc,"gradient")
```

```
##                  b1            b2            b3
##  [1,] -2.239464e-11  7.806283e-12 -1.156686e-12
##  [2,] -2.267631e-12  2.974312e-11  2.957756e-11
##  [3,] -8.948509e-12  3.630193e-11 -1.256267e-11
##  [4,] -1.649125e-12  1.182179e-13  6.369841e-11
##  [5,] -4.272493e-11 -1.109757e-11  3.501116e-11
##  [6,]  1.867381e-10  1.793287e-11  3.319552e-11
##  [7,]  1.090728e-11  1.840947e-11  9.946462e-12
##  [8,]  2.035664e-10 -1.520996e-10  1.911593e-10
##  [9,] -3.582228e-10  2.039028e-10 -1.905254e-10
## [10,]  3.202474e-10  8.291927e-11 -6.263366e-11
## [11,]  5.931922e-12 -1.779933e-11  3.682277e-10
## [12,]  4.132902e-10 -6.831839e-11  3.817154e-10
```

```
## Summary comparisons
max(abs(attr(ndeunsc, "gradient")-attr(vfunsc,"gradient")))
```

```
## [1] 6.569545e-07
```

```
max(abs(attr(ndeunsc2, "gradient")-attr(vfunsc,"gradient")))
```

```
## [1] 8.632699e-10
```

```
max(abs(attr(ndeunscx, "gradient")-attr(vfunsc,"gradient")))
```

```
## [1] 0.0001254074
```

```
max(abs(attr(ndeunscx2, "gradient")-attr(vfunsc,"gradient")))
```

```
## [1] 5.435313e-05
```

```
max(abs(jeunsc-attr(vfunsc,"gradient")))
```

```
## [1] 4.132902e-10
```

Performance results for different computing environments

Here we present tables of the results, preceded by identified descriptions of the machines we used.

M21-LM20.1

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Linux Mint 20.1
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
##  [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
##  [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
##  [7] LC_PAPER=en_CA.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] microbenchmark_1.4-7 numDeriv_2016.8-1.1  nlsr_2019.9.7
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.0    magrittr_2.0.1    tools_4.1.0       htmltools_0.5.1.1
##  [5] yaml_2.2.1        stringi_1.6.2     rmarkdown_2.8     knitr_1.33
##  [9] stringr_1.4.0     xfun_0.23         digest_0.6.27     rlang_0.4.11
## [13] evaluate_0.14
```

To get a good picture of the physical and logical machine that is M21-LM20.1, we can run

```
inxi -F >tlinux.txt
```

in a command line terminal in the host machine.

While it may be tempting to run either

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

or

```
inxi -F >t2.txt
```

it turns out that the encoding of the files is different. Indeed the files are different sizes!

```
ls -al t*.txt
```

```
## -rw-rw-r-- 1 john john 2409 May 27 11:57 t2.txt
## -rw-rw-r-- 1 john john 2413 May 27 10:17 tlinux2.txt
## -rw-rw-r-- 1 john john 1917 May 27 09:48 tlinuxOS.txt
## -rw-rw-r-- 1 john john 2400 May 27 11:57 t.txt
```

Here is the result from running the `inxi` command in the native Linux terminal.

```
cat(readLines('tlinuxOS.txt'), sep = '\n')
```

```
## System:
##   Host: M21 Kernel: 5.4.0-73-generic x86_64 bits: 64 Desktop: MATE 1.24.0
##   Distro: Linux Mint 20.1 Ulyssa
## Machine:
##   Type: Desktop System: ASUS product: N/A v: N/A
```

```
##   serial: <superuser/root required>
##   Mobo: ASUSTeK model: PRIME Z490-A v: Rev 1.xx
##   serial: <superuser/root required> UEFI: American Megatrends v: 0607
##   date: 05/29/2020
## CPU:
##   Topology: 6-Core model: Intel Core i5-10400 bits: 64 type: MT MCP
##   L2 cache: 12.0 MiB
##   Speed: 800 MHz min/max: 800/4300 MHz Core speeds (MHz): 1: 800 2: 800
##   3: 800 4: 800 5: 800 6: 800 7: 800 8: 800 9: 800 10: 800 11: 800 12: 800
## Graphics:
##   Device-1: Intel driver: i915 v: kernel
##   Display: x11 server: X.Org 1.20.9 driver: modesetting unloaded: fbdev,vesa
##   resolution: 1920x1080~60Hz, 1920x1080~60Hz
##   OpenGL: renderer: Mesa Intel UHD Graphics 630 (CML GT2) v: 4.6 Mesa 20.2.6
## Audio:
##   Device-1: Intel Comet Lake PCH cAVS driver: snd_hda_intel
##   Sound Server: ALSA v: k5.4.0-73-generic
## Network:
##   Device-1: Intel driver: igc
##   IF: enp3s0 state: down mac: 3c:7c:3f:2d:e8:8b
##   Device-2: ASIX AX88179 Gigabit Ethernet type: USB driver: ax88179_178a
##   IF: enx00249b1580ea state: up speed: 100 Mbps duplex: full
##   mac: 00:24:9b:15:80:ea
## Drives:
##   Local Storage: total: 4.55 TiB used: 1.09 TiB (24.0%)
##   ID-1: /dev/nvme0n1 vendor: Western Digital model: WDS100T3X0C-00SJG0
##   size: 931.51 GiB
##   ID-2: /dev/sda vendor: Seagate model: ST2000DM008-2FR102 size: 1.82 TiB
##   ID-3: /dev/sdb vendor: Seagate model: ST2000DM008-2FR102 size: 1.82 TiB
## RAID:
##   Device-1: m21z type: zfs status: ONLINE size: 1.81 TiB free: 950.00 GiB
##   Components: online: sda sdb
## Partition:
##   ID-1: / size: 915.40 GiB used: 212.99 GiB (23.3%) fs: ext4
##   dev: /dev/nvme0n1p2
## Sensors:
##   System Temperatures: cpu: 27.8 C mobo: N/A
##   Fan Speeds (RPM): N/A
## Info:
##   Processes: 408 Uptime: 1h 25m Memory: 31.19 GiB used: 4.54 GiB (14.6%)
##   Shell: bash inxi: 3.0.38
```