BloodDiagnostics

Kevin O'Brien

Tuesday, July 07, 2015

```
#using package NLME
library(nlme)
source("BloodData.R")
```

Working on the JS comparison

```
blood = groupedData( BP ~ method | subject ,
  data = data.frame( BP = c(Blood),
    subject = rep(seq(nrow(Blood)), ncol(Blood)),
        method = rep(c("J","R","S"), rep(nrow(Blood)*3, 3)),
        repl = rep(rep(c(1:3), rep(nrow(Blood), 3)), 3) ),
    labels = list(BP = "Systolic Blood Pressure", method = "Measurement Device"),
    )
```

Working on the JS comparison

```
# consider on methods "J" and "S"
dat = subset(blood, subset = method != "R") # fixed-effects X_i with(subset(dat, subset = subject
== "1"), model.matrix(BP ~ method)) # random-effects Z_i with(subset(dat, subset = subject ==
"1"), model.matrix( ~ method -1))
```

Fitting LME Models

```
JS.roy1 = lme(BP ~ method-1, data = dat,random = list(subject=pdSymm(~ method-1)), weights=varIden
t(form=~1|method),correlation = corSymm(form=~1 | subject/repl), method="ML")

JS.roy2 = lme(BP ~ method-1, data = dat,random = list(subject=pdCompSymm(~ method-1)), correlation
= corSymm(form=~1 | subject/repl), method="ML")

JS.roy3 = lme(BP ~ method-1, data = dat,random = list(subject=pdSymm(~ method-1)),weights=varIden
t(form=~1|method), correlation = corCompSymm(form=~1 | subject/repl), method="ML")

JS.roy4 = lme(BP ~ method-1, data = dat,random = list(subject=pdCompSymm(~ method-1)), correlation
= corCompSymm(form=~1 | subject/repl), method="ML")
```

LOO updating

```
subject.c <- levels(dat$subject)

subject.c
```

```
##  [1] "74" "36" "3"  "62" "31" "42" "11" "41" "55" "17" "45" "52" "57" "18"
## [15] "7"  "9"  "12" "50" "10" "14" "83" "15" "37" "1"  "2"  "35" "85" "13"
## [29] "32" "33" "53" "34" "56" "79" "4"  "8"  "54" "69" "84" "5"  "19" "47"
## [43] "6"  "16" "70" "26" "43" "39" "82" "28" "46" "40" "27" "20" "64" "22"
## [57] "63" "25" "81" "49" "73" "65" "77" "59" "60" "23" "24" "21" "75" "67"
## [71] "51" "44" "72" "58" "66" "76" "48" "30" "80" "68" "38" "78" "29" "61"
## [85] "71"
```

```
lmeU <- function(cx) {
    dfU <- subset(dat, subject != cx)          # LOO data
    update(JS.roy1, data = dfU)                # LOO fit
}


lmeUall        <- lapply(subject.c, lmeU)       # List with LOO fits
names(lmeUall) <- subject.c                     # Names assigned
```

```
class(lmeUall)
```

```
## [1] "list"
```

```
length(lmeUall)
```

```
## [1] 85
```

```
library(nlmeU)
```

```
## Warning: package 'nlmeU' was built under R version 3.2.1
```

```
lLik.i <- by(dat, dat$subject,
   FUN = function(dfi) logLik1(JS.roy1, dfi))
lLik.i <- as.vector(lLik.i)   # Coerse array to vector
lLik.i[1:5]               # logLik_i for the first five subjects
```

```
## [1] -23.80285 -22.28637 -22.77093 -25.01049 -22.79563
```

```
sum(lLik.i)               # Sum logLik_i; compare to Panel 20.6a
```

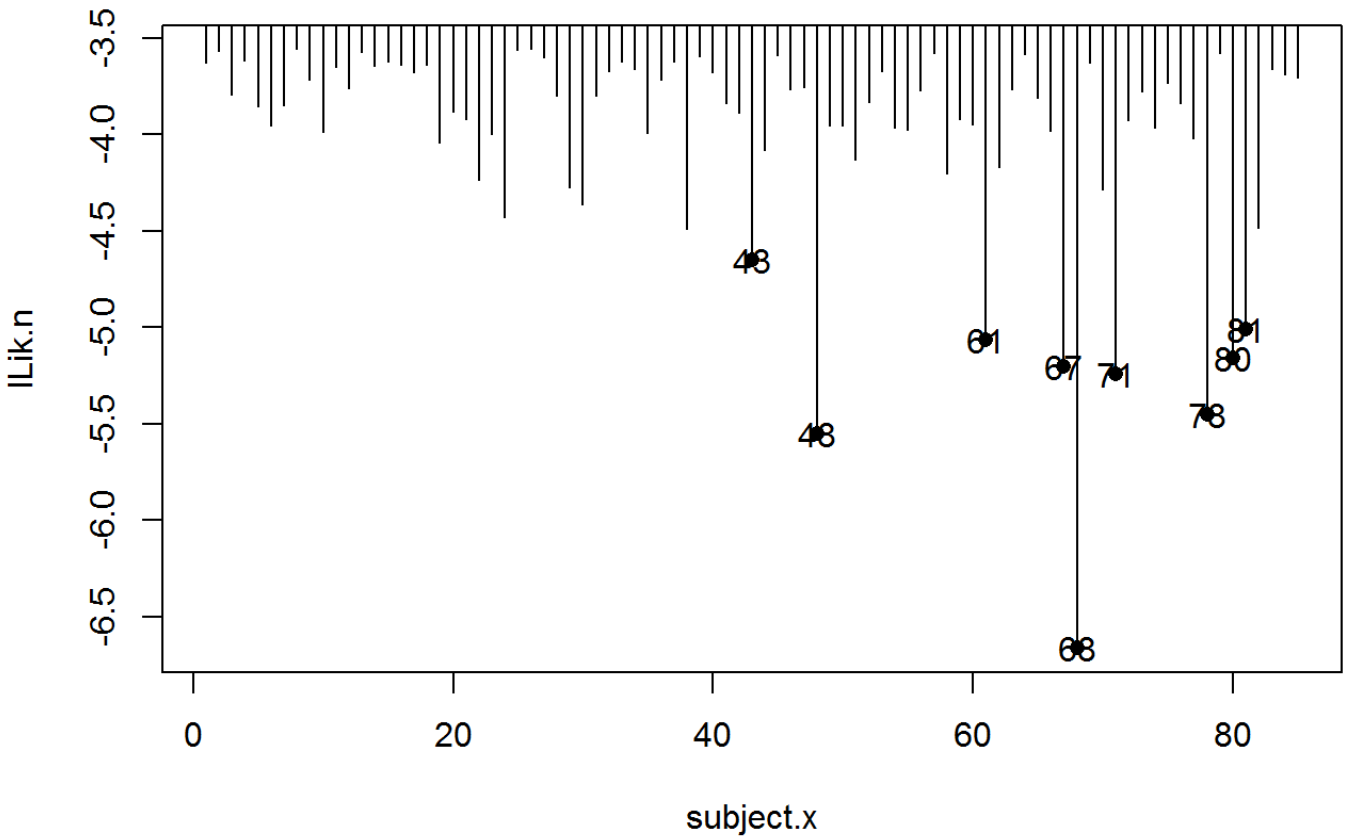
```
## [1] -2038.116
```

```
nx <- by(dat, dat$subject, nrow)             # ni
lLik.n <- lLik.i/as.vector(nx)                 # logLiki
outL <- lLik.n < -4.5                            # TRUE for values < -4.5
lLik.n[outL]                                   # logLiki/ni <  -6
```

```
## [1] -4.649704 -5.011806 -5.202401 -5.551252 -5.156625 -6.662282 -5.451200
## [8] -5.066321 -5.241152
```

```
subject.c <- levels(dat$subject)
subject.x <- as.numeric(subject.c)
```

```
plot(lLik.n ~ subject.x, type = "h")           # Fig. 20.1
points(subject.x[outL], lLik.n[outL], type = "p", pch = 16)
text(subject.x[outL], lLik.n[outL], subject.c[outL])
```



code chunk: R20.9a

```
lLik <- function(cx){
    lmeU   <- lmeUall[[cx]]           # LOO fit extracted
    lLikU  <- logLik(lmeU, REML = FALSE)  # LOO log-likelihood
    df.s   <-                         # Data for subject cx...
       subset(dat, subject == cx)
    lLik.s <- logLik1(lmeU, df.s)       # ...and log-likelihood.
    return(lLikU + lLik.s)            # "Displaced" log-likelihood...
}
lLikUall <- sapply(subject.c, lLik)     # ...for all subjects.
```

```
dif.2Lik <- 2*(logLik(JS.roy1) - lLikUall) # Vector of LDi
summary(dif.2Lik)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -6.9360  0.1484  0.3016  0.3386  0.7129  4.0010
```

```
beta0  <- fixef(JS.roy1)                # beta
names(beta0)                            # Long names
```

```
## [1] "methodJ" "methodS"
```

```
names(beta0) <- abbreviate(names(beta0), minlength = 7) # Short names
beta0                                    # beta printed.
```

```
##  methodJ  methodS
## 127.4078 143.0275
```

```
vcovb  <- vcov(JS.roy1)                 # vcovb
colnames(vcovb) <- names(beta0)          # Short names
vcovb                                    # vcovb printed.
```

```
##          methodJ  methodS
## methodJ 11.01701  9.30105
## methodS  9.30105 11.75301
```

```
betaUall <- sapply(lmeUall, fixef)          # Matrix with beta(-i)
betaUall
```

```
##               74       36        3       62       31       42       11
## methodJ 127.9246 127.8373 127.9643 127.8294 127.8770 127.7897 127.6944
## methodS 143.6627 143.5913 143.5913 143.6706 143.5079 143.4960 143.4167
##               41       55       17       45       52       57       18
## methodJ 127.7976 127.9881 127.7183 127.7262 127.7103 127.6389 127.5119
## methodS 143.4524 143.4167 143.4008 143.3651 143.3651 143.3413 143.3730
##                7        9       12       50       10       14       83
## methodJ 127.6310 127.6548 127.5675 127.8373 127.7341 127.6706 127.6071
## methodS 143.3571 143.2897 143.2500 143.2976 143.3413 143.2897 143.3095
##               15       37        1        2       35       85       13
## methodJ 127.6706 127.6468 127.6825 127.6310 127.5040 127.5516 127.5992
## methodS 143.2540 143.2857 143.2460 143.2381 143.3016 143.2540 143.2143
##               32       33       53       34       56       79        4
## methodJ 127.5198 127.4643 127.6865 127.4167 127.6706 127.6389 127.6706
## methodS 143.1984 143.2302 143.2460 143.2222 143.2619 143.1984 143.1865
##                8       54       69       84        5       19       47
## methodJ 127.5913 127.7500 127.5357 127.7183 127.5437 127.2897 127.5595
## methodS 143.1667 143.2817 143.1389 143.1190 143.1627 143.2262 143.0675
##                6       16       70       26       43       39       82
## methodJ 127.3929 127.5040 127.4881 127.3690 127.4881 127.3849 127.6468
## methodS 143.0754 143.0952 143.1627 143.0476 143.2063 143.0635 143.1032
##               28       46       40       27       20       64       22
## methodJ 127.1548 127.4008 127.3294 127.2421 127.1706 127.2500 127.1865
## methodS 143.0913 142.9722 143.0079 142.9167 143.0198 142.9167 142.9643
##               63       25       81       49       73       65       77
## methodJ 127.4008 127.2262 127.2659 127.1071 127.0516 127.2897 127.5357
## methodS 142.9405 142.8889 143.0913 143.0079 142.8452 142.8849 142.8413
##               59       60       23       24       21       75       67
## methodJ 127.0595 127.2579 126.8929 126.9802 127.0437 126.8611 127.3373
## methodS 142.7937 142.7937 142.7579 142.8056 142.6786 142.5873 142.8492
##               51       44       72       58       66       76       48
## methodJ 127.0595 126.9960 127.3294 126.8056 127.0595 126.7897 127.3373
## methodS 142.6825 142.5833 142.4960 142.4246 142.5119 142.3651 142.5079
##               30       80       68       38       78       29       61
## methodJ 126.5437 127.6151 127.5833 126.4325 127.5040 126.3135 126.5833
## methodS 142.5198 142.3294 142.5159 142.2579 142.0595 142.0238 142.2103
##               71
## methodJ 126.3452
## methodS 142.1508
```

```
vb.inv <- solve(vcovb)
vb.inv
```

```
##            methodJ    methodS
## methodJ  0.2734946 -0.2164370
## methodS -0.2164370  0.2563676
```

```
CookDfun <- function(betaU){
   dbetaU <- betaU - beta0                  # beta(-i) - beta
   CookD.value <- t(dbetaU) %*% vb.inv %*% dbetaU
}
```

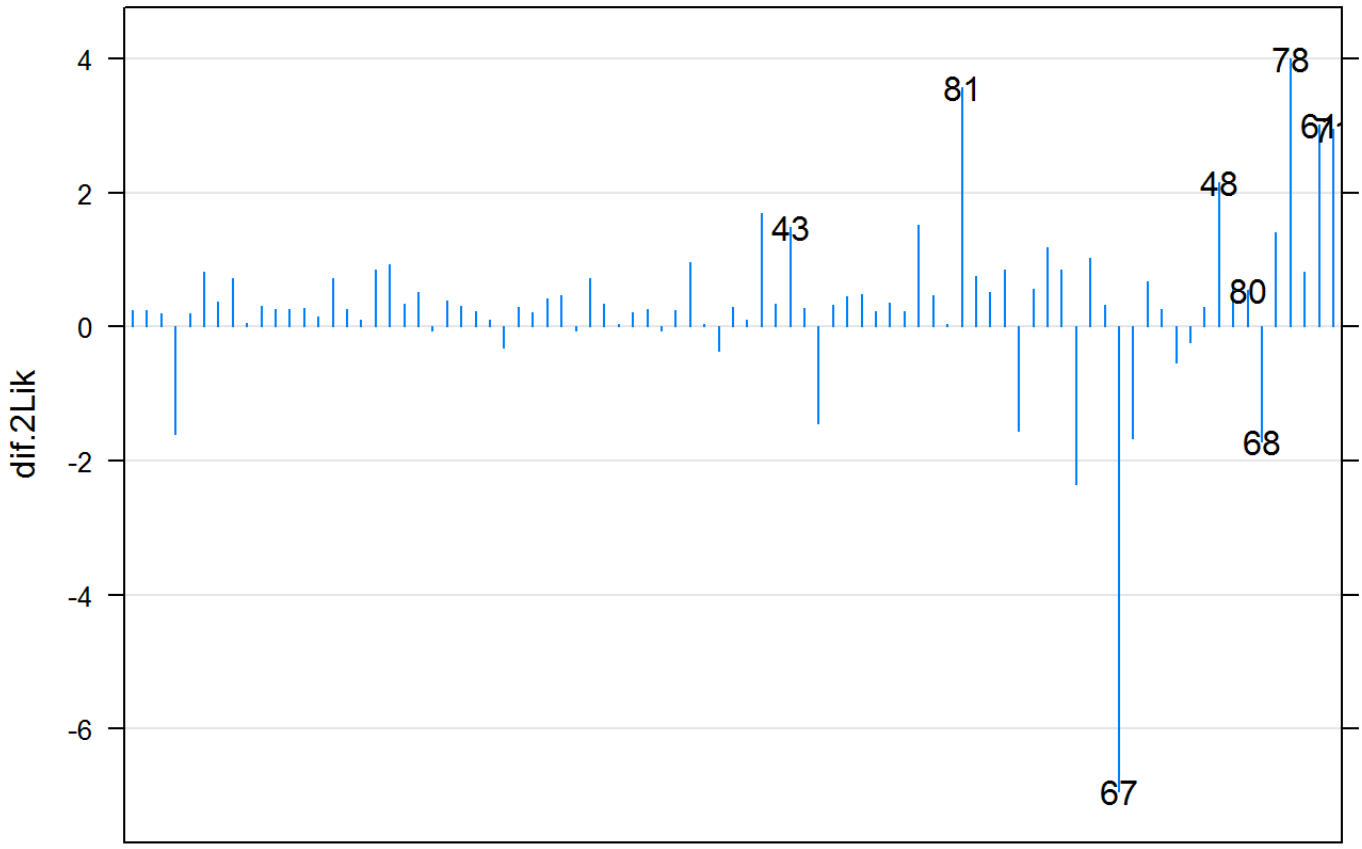
```
CookD.num <- apply(betaUall, 2, CookDfun)
(n.fixeff <- length(beta0))                 # Number of fixed effects
```

```
## [1] 2
```

```
rankX <- n.fixeff                           # Rank of matrix X
CookD <- CookD.num/rankX                    # Cook's distance Di
```

```
subject.f <- factor(subject.c, levels = subject.c)
myPanel <- function(x, y, ...){
   x1 <- as.numeric(x)
   panel.xyplot(x1, y, ...)
   ltext(x1[outL], y[outL], subject.c[outL])  # Label outlying LDi
}
```

```
library(lattice)
dtp <-                                        # Fig. 20.2
   dotplot(dif.2Lik ~ subject.f, panel = myPanel, type = "h")
lxlims <- length(dtp$x.limits)
update(dtp, xlim = rep("", lxlims), grid = "h")
```


dif.2Lik
4
2
0
-2
-4
-6
78
81
71
48
43
80
68
67