Testing differences among variances for repeated measures data

```
> # Testing differences among variances for repeated measures data
> # Simulate some MVN data
>
> Sigma <- rbind( c(10,      5,          5.049752),
+                 c( 5,      10,         5.049752),
+                 c(5.049752, 5.049752, 10.2)  )
> # Correlations are all 0.50
>
> spec <- eigen(Sigma) ; spec
$values
[1] 20.133923  5.066077  5.000000

$vectors
           [,1]       [,2]          [,3]
[1,] -0.5741427 -0.4127472  7.071068e-01
[2,] -0.5741427 -0.4127472 -7.071068e-01
[3,] -0.5837126  0.8119603  1.799357e-15

> P <- spec$vectors
> Lambda <- diag(spec$values) ; Lambda
         [,1]     [,2] [,3]
[1,] 20.13392 0.000000    0
[2,]  0.00000 5.066077    0
[3,]  0.00000 0.000000    5
> SQR <- P %*% sqrt(Lambda) %*% t(P)  # Square root matrix
>
> n <- 1000
> set.seed(999)
> Z <- rnorm(n*3) ; dim(Z) <- c(3,n)
> X <- t(SQR%*%Z)
> # Mu1=10, Mu2=15, Mu3=20, not that it matters
> X[,1] <- X[,1]+10 ; X[,2] <- X[,2]+15 ; X[,3] <- X[,3]+20
>
> cor(X)
          [,1]      [,2]      [,3]
[1,] 1.0000000 0.5544458 0.5057988
[2,] 0.5544458 1.0000000 0.4904126
[3,] 0.5057988 0.4904126 1.0000000
> var(X) ; Sigma
          [,1]     [,2]      [,3]
[1,] 10.632235 5.566524  5.324287
[2,]  5.566524 9.480372  4.874675
[3,]  5.324287 4.874675 10.421800
          [,1]      [,2]      [,3]
[1,] 10.000000  5.000000  5.049752
[2,]  5.000000 10.000000  5.049752
[3,]  5.049752  5.049752 10.200000
>
> Sighat <- (n-1)*var(X)/n  # MLE, not the unbiased version
```

```
> pi # Built-in
[1] 3.141593
> det(Sigma)
[1] 510
> prod(spec$values) # Determinant is product of eigenvalues
[1] 510
>
> # Minimize this over all Sig
> # n/2 * ( p*log(2*pi) + log(det(Sig)) + sum(diag(Sighat%*%solve(Sig))) )
>
> unrest <- function(theta) # - log likelihood for unrestricted model (6-D)
+                         # Assumes existence of Sighat and n
+       {
+       pp <- dim(Sighat)[1]
+       # Load matrix Sig with theta values
+       Sig <- rbind( c(theta[1],theta[2],theta[3]),
+                     c(theta[2],theta[4],theta[5]),
+                     c(theta[3],theta[5],theta[6])      )
+       unrest <- n/2 * ( pp*log(2*pi) + log(det(Sig)) +
+                         sum(diag(Sighat%*%solve(Sig))) )
+       unrest
+       } # End of function unrest
>
>
> # It is not really necessary to fit the unrestricted model. We will
> # do it to test our code and to illustrate the importance of good
> # starting values in finding numerical MLEs.
>
> # The right MLE is contained in Sighat
> Sighat
          [,1]     [,2]      [,3]
[1,] 10.621603 5.560957  5.318963
[2,]  5.560957 9.470891  4.869800
[3,]  5.318963 4.869800 10.411378
> # And the right - log likelihood at the MLE is
> right <- n/2 * ( 3*log(2*pi) + log(det(Sighat)) + 3 ) ; right
[1] 7357.706
>
>
> badstart <- c(1,0,0,1,0,1)  # Identity matrix
> full1 <- nlm(unrest,badstart) ; full1
$minimum
[1] 10866.45

$estimate
[1] 690.2371 796.7104 762.0400 607.8066 697.6891 675.1778

$gradient
[1] -3.1853419 -0.9174104  8.0776012 -4.8839326 11.0467687 -9.4603981

$code
[1] 5

$iterations
[1] 1

> # help(nlm)
```

```
> # Code=5 means "maximum step size stepmax exceeded five consecutive times.
> # Either the function is unbounded below, becomes asymptotic to a finite
> # value from above in some direction or stepmax is too small."
>
> badstart2 <- 10*badstart
> full2 <- nlm(unrest,badstart2) ; full2
There were 50 or more warnings (use warnings() to see the first 50)
$minimum
[1] -293864975

$estimate
[1]   18.01661  161.12471 -102.81476   51.64468 -191.07915  204.83213

$gradient
[1] -5.124603e+12           Inf           Inf -1.264528e+12           Inf
[6]           Inf

$code
[1] 2

$iterations
[1] 34

> # Code=2 means "successive iterates within tolerance, current iterate
> # is probably solution."
>
> # I wonder what's wrong with the answer
> bad <- full2$estimate
> wonder <- rbind(    c(bad[1], bad[2],  bad[3]),
+                     c(bad[2], bad[4],  bad[5]),
+                     c(bad[3], bad[5],  bad[6])   )
> wonder
           [,1]       [,2]      [,3]
[1,]   18.01661  161.12471 -102.8148
[2,]  161.12471   51.64468 -191.0792
[3,] -102.81476 -191.07915  204.8321
> det(wonder)
[1] 1.954324
> # What's the (estimated) correlation between X1 and X2?
> bad[2]/sqrt(bad[1]*bad[4])
[1] 5.282176
> # Outside the parameter space. Try taking smaller steps
>
```

```
> full3 <- nlm(unrest,badstart2,stepmax=1) ; full3
$minimum
[1] 7357.706

$estimate
[1] 10.621509  5.560928  5.318897  9.470917  4.869788 10.411327

$gradient
[1] -3.031218e-04  6.656521e-05 -2.074146e-04  3.872901e-04  2.022640e-04
[6] -8.491030e-05

$code
[1] 1

$iterations
[1] 19

> right
[1] 7357.706
> Sighat
          [,1]     [,2]      [,3]
[1,] 10.621603 5.560957  5.318963
[2,]  5.560957 9.470891  4.869800
[3,]  5.318963 4.869800 10.411378
>
> # Define function rest
>
> rest <- function(theta) # - log likelihood for restricted model
+                         # Assumes existence of Sighat and n
+                         # Theta has 4 elements this time
+       {
+       pp <- dim(Sighat)[1]
+       # Load matrix Sig with theta values
+       Sig <- rbind( c(theta[1],theta[2],theta[3]),
+                     c(theta[2],theta[1],theta[4]),
+                     c(theta[3],theta[4],theta[1])      )
+       rest <- n/2 * ( pp*log(2*pi) + log(det(Sig)) +
+                       sum(diag(Sighat%*%solve(Sig))) )
+       rest
+       } # End of function rest
>
> # Use some common sense to get good starting values.
>
> vstart <- mean(diag(Sighat)) # Average observed sample variances
> # And choose starting covariances to reproduce observed sample correlations
> # Covariance = Correlation * SD1 * SD1
> Kor <- cor(X) ; Kor
          [,1]      [,2]      [,3]
[1,] 1.0000000 0.5544458 0.5057988
[2,] 0.5544458 1.0000000 0.4904126
[3,] 0.5057988 0.4904126 1.0000000
> Cov12 <- Kor[1,2] * vstart
> Cov13 <- Kor[1,3] * vstart
> Cov23 <- Kor[2,3] * vstart
>
> rstart <- c(vstart,Cov12,Cov13,Cov23) ; rstart
[1] 10.167958  5.637582  5.142941  4.986494
>
```

```
> red <- nlm(rest,rstart) ; red
$minimum
[1] 7360.342

$estimate
[1] 10.172146  5.656430  5.095395  5.014849

$gradient
[1]  0.0003498625  0.0003680472 -0.0002507833 -0.0005377334

$code
[1] 1

$iterations
[1] 6

>
> G <- 2 * (red$minimum-right) ; G
[1] 5.2728
> 1-pchisq(G,2)
[1] 0.07161865
>
```

Now do it more compactly. Here is the program `eqvar.R.txt`

```
# eqvar.R.txt
# Testing differences among variances for repeated measures data
# Define functions

LLMVN <- function(Sig,Sighat,nn)
# Minus Log Likelihood for MVN. A function of Sigma, with mu=Xbar
# This is general. Usually call it with another function.
    {
    pp <- dim(Sig)[1]
    LLMVN <- n/2 * ( pp*log(2*pi) + log(det(Sig)) +
                      sum(diag(Sighat%*%solve(Sig))) )
    LLMVN
    } # End of function LLMVN

rest <- function(theta,MLE,n) # Restricted (null) model
    {
      # Load matrix sigma with theta values
      sigma <- rbind( c(theta[1],theta[2],theta[3]),
                      c(theta[2],theta[1],theta[4]),
                      c(theta[3],theta[4],theta[1])      )
      rest <- LLMVN(sigma,MLE,n)
    rest
    } # End of function rest

# Simulate some MVN data

Sigma <- rbind( c(10,      5,          5.049752),
                c( 5,      10,         5.049752),
                c(5.049752, 5.049752, 10.2)  )
# Correlations are all 0.50

spec <- eigen(Sigma) ; P <- spec$vectors ; Lambda <- diag(spec$values)
```

```
SQR <- P %*% sqrt(Lambda) %*% t(P)  # Square root matrix

n <- 1000
set.seed(999)
Z <- rnorm(n*3) ; dim(Z) <- c(3,n)
X <- t(SQR%*%Z)
# Mu1=10, Mu2=15, Mu3=20, not that it matters
X[,1] <- X[,1]+10 ; X[,2] <- X[,2]+15 ; X[,3] <- X[,3]+20
Sigmahat <- (n-1)*var(X)/n  # MLE, not the unbiased version

right <- n/2 * ( 3*log(2*pi) + log(det(Sigmahat)) + 3 )

# Use common sense to get good starting values for the restricte model.
vstart <- mean(diag(Sigmahat)) # Average observed sample variances
# And choose starting covariances to reproduce observed sample correlations
# Covariance = Correlation * SD1 * SD1
Kor <- cor(X)
Cov12 <- Kor[1,2] * vstart
Cov13 <- Kor[1,3] * vstart
Cov23 <- Kor[2,3] * vstart
rstart <- c(vstart,Cov12,Cov13,Cov23)

red <- nlm(rest,rstart,MLE=Sigmahat,n)
G <- 2 * (red$minimum-right) ; pval1 <- 1-pchisq(G,2)
cat("LR Test:  G = ", G, ", p = ", pval1, "\n")

# Try a Wald Test: Need to fit the unrestricted model numerically

unrest <- function(theta,MLE,nn) # - log likelihood for unrestricted model (6-D)
      {
      pp <- dim(MLE)[1]
      # Load matrix Sig with theta values
      Sig <- rbind( c(theta[1],theta[2],theta[3]),
                    c(theta[2],theta[4],theta[5]),
                    c(theta[3],theta[5],theta[6])      )
      unrest <- LLMVN(Sig,MLE,nn)
      unrest
      } # End of function unrest

badstart2 <- c(10,0,0,10,0,10)  # Identity matrix
full <- nlm(unrest,badstart2,hessian=T,stepmax=1,MLE=Sigmahat,nn=n)
asymp <- solve(full$hessian) ; thetahat <- full$estimate
# H0: C theta = 0 is that theta1=theta4=theta6
CC <- rbind(  c(1,0,0,-1,0,0),
              c(0,0,0,1,0,-1) )
W <- t(CC%*%thetahat) %*% solve(CC%*%asymp%*%t(CC)) %*% CC%*%thetahat
W <- as.numeric(W) # it was a 1x1 matrix
pval2 <- 1-pchisq(W,2)
cat("Wald Test:  W = ", W, ", p = ", pval2, "\n")
cat("\n")
```

Input and output

```
> source("eqvar.R.txt")
LR Test:  G =  5.2728 , p =  0.07161865
Wald Test:  W =  5.431555 , p =  0.0661535
>
```