Homework 3 pt. 2

Jonathan Brinkerhoff

2023-02-19

a. Read in functions and generate data

```
library(matrixcalc)

sqrtm <- function (A) {
  # Obtain matrix square root of a matrix A
  a = eigen(A)
  sqm = a$vectors %*% diag(sqrt(a$values)) %*% t(a$vectors)
  sqm = (sqm+t(sqm))/2
}
# a
gen <- function(n,p,mu,sig,seed = 2023){
  #---- Generate data from a p-variate normal with mean mu and covariance sigma
  # mu should be a p by 1 vector
  # sigma should be a positive definite p by p matrix
  # Seed can be optionally set for the random number generator
  set.seed(seed)
  # generate data from normal mu sigma
  z = matrix(rnorm(n*p),n,p)
  datan = z %*% sqrtm(sig) + matrix(mu,n,p, byrow = TRUE)
  datan
}

log_likelihood <- function(data,mu,sig,n,p){
  siginv = solve(sig)
  C= matrix(0,p,p); # initializing sum of (xi-mu)(xi-mu)^T
  for (i in 1:n){
    xm = data[i,] - mu
    C = C + xm %*% t(xm)
  }
  if(is.null(siginv)){
    log_det_sig <- log(det(sig))
  } else {
    #-- in this case siginv is input so we use the fact that det(sig)=1/det(siginv)
    log_det_sig <-log(1/det(siginv))
  }
  l = -(n*p*log(2*pi)+n*log_det_sig + sum(siginv * C ))/2
}

gradient <- function(data, mu, sig){
  p = dim(data)[2]
```

```r
  siginv = solve(sig)

  C= matrix(0,p,p); # initializing sum of (xi-mu)(xi-mu)^T
  sxm = matrix(0,p,1) # initializing sum of xi-mu
  gradmu = sxm; # initializing this sum is used for the gradient w.r.t. mu
  for (i in 1:n){
    xm = data[i,] - mu
    sxm = sxm + xm
    C = C + xm %*% t(xm)
  }
  gradmu = siginv %*% sxm
  gradsig = n*siginv - siginv %*% C %*% siginv

    for (i in 1:p){
      for (j in 1:p){
        if(i == j){gradsig[i,j] = -1/2*gradsig[i,j]}else{gradsig[i,j] = -1*gradsig[i,j]}
      }
    }
  #ensure symmetry of sigma
  gradsig = vec2mat(mat2vec(gradmu, gradsig, p),p)$sigma

  gradlist <- list("gradmu" = gradmu, "gradsig" = gradsig)
  return(gradlist)
}

mat2vec <- function(mu, sig, p){
  #vectorize mu
  out = c(mu)

  #vectorize lower diagonals of sigma
  sigvec = c(sig[1:1])
  for (i in 2:p){
    sigvec = append(sigvec, sig[i, 1:i])
  }

  out = append(out, sigvec)
  return(out)
}

vec2mat <- function(thetvec, p){
  mumat = matrix(thetvec[1:p])
  sigmat = matrix(0, nrow = p, ncol = p)

  pos = p+1
  for (i in 1:p){
    sigmat[i, 1:i] = thetvec[c(pos:(pos+i-1))]
    pos = pos + i
  }
  #return upper diagonals
  for(i in 1:p){
    for(j in i : p){
      if(j > i){sigmat[i,j] = sigmat[j,i]}
    }
```

```
  }

  matlist <- list("mu" = mumat, "sigma" = sigmat)
  return(matlist)
}

hessian <- function(data, mu, sig){
  p = dim(data)[2]
  n = dim(data)[1]

  siginv = solve(sig)
  I = diag(p)

  s= matrix(0,p,p); # initializing sum of (xi-mu)(xi-mu)^T
  sxm = matrix(0,p,1) # initializing sum of xi-mu

  for (i in 1:n){
    xm = data[i,] - mu
    sxm = sxm + xm
    s = s + (xm %*% t(xm))
  }
  c = siginv %*% sxm
  z = (-n*I + 2*siginv %*% s) %*% siginv

  mumu = -n*siginv

  columnindex = 1
  rowindex = 1
  sigsig = matrix(0, p*(p+1)/2, p*(p+1)/2)
  for(i in 1:p){
    for(j in 1:i){
      for(k in 1:p){
        for(l in 1:k){
          if(i == j & l == k){sigsig[rowindex, columnindex] = -1/2*(z[k,i]*siginv[i,k])}
          if(i != j & k != l){sigsig[rowindex, columnindex] = -1/2*(z[k,i]*siginv[j,l] + z[l,j]*siginv[
          if(i != j & k == l){sigsig[rowindex, columnindex] = -1/2*(z[k,i]*siginv[j,k] + z[k,j]*siginv[
          if(i == j & k != l){sigsig[rowindex, columnindex] = -1/2*(z[l,i]*siginv[i,k] + z[k,i]*siginv[
          if(rowindex < p*(p+1)/2){rowindex = rowindex + 1}else{rowindex = 1}
          }
      }
      if(columnindex < p*(p+1)/2){columnindex = columnindex + 1}else{columnindex = 1}
    }
  }


  sigmu = matrix(0, p, p*(p+1)/2)
  columnindex = 1
  #new matrix for mu sigma section
  for(l in 1:p){
    for(k in 1:l){
      for(i in 1:p){
        if(l == k){sigmu[i,columnindex] = -siginv[i,k]*c[k]}else{sigmu[i,columnindex] = -siginv[i,k]*c[
      }
```

```
      if(columnindex < p*(p+1)/2){columnindex = columnindex + 1}else{columnindex = 1}
    }
  }


  sigmuleft = t(sigmu)

  combination = cbind(mumu, sigmu)
  combination2 = cbind(sigmuleft, sigsig)
  finalcombo = rbind(combination, combination2)

  return(finalcombo)
}

n <- 200
p <- 3
sig <- matrix(c(1,.7,.7,.7,1,.7,.7,.7,1),3,3) # known sigma Note <<- makes it global
mu <- matrix(c(-1,1,2))
datan = gen(n,p,mu,sig)
datan[1:3,]
```

```
##           [,1]       [,2]      [,3]
## [1,] -1.898781 -0.9176151 0.8583400
## [2,] -1.214523  2.1586398 2.5588271
## [3,] -2.745287  0.4606282 0.9513419
```

b. Steepest Ascent

```
SA <- function (mu, datan, sig, maxit, tolerr, tolgrad) {
  n <- nrow(datan)
  p = dim(datan)[2]

  for (it in 1:maxit){
    a <- log_likelihood(datan, mu, sig,n,p)
    agrad <- gradient(datan,mu,sig)

    mu1 = mu + agrad$gradmu
    sig1 = sig + agrad$gradsig
    halve = 0;

    #L2 norm of gradient
    gradnorm = norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),typ
    if(it == 1 || it == 2 || it == 425 || it == 426){print("iteration  halving    log-likelihood   ||gr

    #check if sig1 is positive definite before sending to log likelihood
    while(!is.positive.definite(sig1) & halve <= 20){
      if(it == 1 || it == 2 || it == 425 || it == 426){print(sprintf('%2.0f          %2.0f          %2.4
      halve = halve + 1
      mu1 = mu + (agrad$gradmu)/2^halve  # Steepest Ascent
      sig1 = sig + (agrad$gradsig)/2^halve
    }
```

```r
    if (halve >= 20) print('Step-halving failed after 20 halvings')

    atmp = log_likelihood(datan, mu1, sig1,n,p)

    #continue halving if new value smaller than prior
    while (atmp < a & halve <= 20){
      #L2 norm of gradient
      gradnorm <- norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),t
      if(it == 1 || it == 2 || it == 425 || it == 426){print(sprintf('%2.0f          %2.0f         %2.4f
      halve = halve+1
      mu1 = mu + (agrad$gradmu)/2^halve  # Steepest Ascent
      sig1 = sig + (agrad$gradsig)/2^halve
      atmp = log_likelihood(datan, mu1, sig1,n,p)
    }
    if (halve >= 20) print('Step-halving failed after 20 halvings')

    #modified relative error
    mre = max(((mat2vec(mu1, sig1, p)) - (mat2vec(mu, sig, p)))/pmax(1,abs(mat2vec(mu1, sig1, p))))
    #L2 norm of gradient
    gradnorm = norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),type

    if(it == 1 || it == 2){print(sprintf('%2.0f          %2.0f         %2.4f         %0.1e',it, halve,at
    if(it == 1 || it == 2 || it == 425 || it == 426){print('-----------------------------')}
    if(mre < tolerr & gradnorm < tolgrad){break}

    mu = mu1
    sig = sig1

  }
  paramlist <- list("mu" = mu, "sigma" = sig)
  return(paramlist)
}

mu0 <- matrix(c(0,0,0))
sig0 <- matrix(c(1,0,0,0,1,0,0,0,1),3,3)
SA(mu0, datan, sig0, 500, 1e-6, 1e-5)
```

```
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 1           0         -1383.9388        1.1e+03"
## [1] " 1           1         -1383.9388        1.1e+03"
## [1] " 1           2         -1383.9388        1.1e+03"
## [1] " 1           3         -1383.9388        1.1e+03"
## [1] " 1           4         -1383.9388        1.1e+03"
## [1] " 1           5         -1383.9388        1.1e+03"
## [1] " 1           6         -1383.9388        1.1e+03"
## [1] " 1           7         -1383.9388        1.1e+03"
## [1] " 1           8         -1383.9388        1.1e+03"
## [1] " 1           9         -890.8302        1.8e+02"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 2           0         -890.8302        4.5e+03"
## [1] " 2           1         -890.8302        4.5e+03"
## [1] " 2           2         -890.8302        4.5e+03"
```

```
## [1] " 2          3         -890.8302         4.5e+03"
## [1] " 2          4         -890.8302         4.5e+03"
## [1] " 2          5         -890.8302         4.5e+03"
## [1] " 2          6         -890.8302         4.5e+03"
## [1] " 2          7         -890.8302         4.5e+03"
## [1] " 2          8         -889.3361         1.2e+03"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] "425         0          -684.7842         1.8e-02"
## [1] "425         1          -684.7842         9.2e-03"
## [1] "425         2          -684.7842         4.6e-03"
## [1] "425         3          -684.7842         2.3e-03"
## [1] "425         4          -684.7842         1.1e-03"
## [1] "425         5          -684.7842         5.7e-04"
## [1] "425         6          -684.7842         2.8e-04"
## [1] "425         7          -684.7842         1.3e-04"
## [1] "425         8          -684.7842         6.3e-05"
## [1] "425         9          -684.7842         2.7e-05"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] "426         0          -684.7842         1.7e-02"
## [1] "426         1          -684.7842         8.3e-03"
## [1] "426         2          -684.7842         4.1e-03"
## [1] "426         3          -684.7842         2.1e-03"
## [1] "426         4          -684.7842         1.0e-03"
## [1] "426         5          -684.7842         5.1e-04"
## [1] "426         6          -684.7842         2.5e-04"
## [1] "426         7          -684.7842         1.2e-04"
## [1] "426         8          -684.7842         5.6e-05"
## [1] "426         9          -684.7842         2.4e-05"
## [1] "-----------------------------"
```

```
## $mu
##            [,1]
## [1,] -0.9223115
## [2,]  0.9609005
## [3,]  1.9714278
##
## $sigma
##           [,1]      [,2]      [,3]
## [1,] 0.8292774 0.5622944 0.5511324
## [2,] 0.5622944 0.9279104 0.6348468
## [3,] 0.5511324 0.6348468 0.9080536
```

c. Newton's Method

```
newton <-function(mu, datan, sig, maxit, tolerr, tolgrad){

   n = dim(datan)[1]
   p = dim(datan)[2]

   for(it in 1:maxit){
```

```
    a <- log_likelihood(datan, mu, sig,n,p)
    grad = gradient(datan, mu, sig)
    gradvec = mat2vec(grad$gradmu, grad$gradsig, p)
    dir = -1*solve(hessian(datan, mu, sig)) %*% gradvec
    dirmat = vec2mat(dir, p)

    mu1 = mu + dirmat$mu
    sig1 = sig + dirmat$sigma

    halve = 0;

    #L2 norm of gradient
    gradnorm = norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),typ
    print("iteration  halving    log-likelihood   ||gradient||")
    #check if sig1 is positive definite before sending to log likelihood
    while(!is.positive.definite(sig1) & halve <= 20){
      print(sprintf('%2.0f          %2.0f         %2.4f         %0.1e',it, halve,a,gradnorm))
      halve = halve + 1
      mu1 = mu + (dirmat$mu)/2^halve
      sig1 = sig + (dirmat$sigma)/2^halve
    }

    atmp = log_likelihood(datan, mu1, sig1,n,p)

    #continue halving if new value smaller than prior
    while (atmp < a & halve <= 20){
      #L2 norm of gradient
      gradnorm <- norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),
      print(sprintf('%2.0f          %2.0f         %2.4f         %0.1e',it, halve,atmp,gradnorm))
      halve = halve+1
      mu1 = mu + (dirmat$mu)/2^halve
      sig1 = sig + (dirmat$sigma)/2^halve
      atmp = log_likelihood(datan, mu1, sig1,n,p)
    }
    if (halve >= 20) print('Step-halving failed after 20 halvings')

    #modified relative error
    mre = max(((mat2vec(mu1, sig1, p)) - (mat2vec(mu, sig, p)))/pmax(1,abs(mat2vec(mu1, sig1, p))))
    #L2 norm of gradient
    gradnorm = norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),typ
    if(mre < tolerr & gradnorm < tolgrad){break}

    print(sprintf('%2.0f          %2.0f         %2.4f         %0.1e',it, halve,atmp,gradnorm))
    print('-----------------------------')


    mu = mu1
    sig = sig1

   }
   paramlist <- list("mu" = mu, "sigma" = sig)
   return(paramlist)
}
```

```r
sig0 <- matrix(c(1,.5,.5,.5,1,.5,.5,.5,1),3,3)
mu0 <- matrix(c(-1.5,1.5,2.3))

newton(mu0, datan, sig0, 500, 1e-7, 1e-7)
```

```
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 1           0         -849.2651         6.7e+02"
## [1] " 1           1         -849.2651         6.7e+02"
## [1] " 1           2         -849.2651         6.7e+02"
## [1] " 1           3         -804.2871         2.9e+03"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 2           0         -733.8443         1.2e+03"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 3           0         -702.1274         4.7e+02"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 4           0         -689.0608         1.6e+02"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 5           0         -685.3022         4.5e+01"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 6           0         -684.7982         6.6e+00"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 7           0         -684.7842         2.1e-01"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 8           0         -684.7842         2.2e-04"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 9           0         -684.7842         2.6e-10"
## [1] "-----------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] "10           0         -684.7842         2.3e-13"
## [1] "10           1         -684.7842         1.3e-10"
## [1] "10           2         -684.7842         2.0e-10"
```

```
## $mu
##            [,1]
## [1,] -0.9223115
## [2,]  0.9609005
## [3,]  1.9714278
##
## $sigma
##           [,1]      [,2]      [,3]
## [1,] 0.8292774 0.5622943 0.5511324
## [2,] 0.5622943 0.9279103 0.6348467
## [3,] 0.5511324 0.6348467 0.9080535
```

d. Fisher Scoring

```
fisher_info <-function(data, sig){
  n = dim(data)[1]
  siginv = solve(sig)
  topleft = -n*siginv

  topright = matrix(0,3,6)
  botleft = t(topright)

  columnindex = 1
  rowindex = 1
  botright = matrix(0, p*(p+1)/2, p*(p+1)/2)
  for(i in 1:p){
    for(j in 1:i){
      for(k in 1:p){
        for(l in 1:k){
          if(i == j & l == k){botright[rowindex, columnindex] = -n/2*(siginv[k,i]*siginv[i,k])}
          if(i != j & k != l){botright[rowindex, columnindex] = -n/2*(siginv[k,i]*siginv[j,l] + siginv[l
          if(i != j & k == l){botright[rowindex, columnindex] = -n/2*(siginv[k,i]*siginv[j,k] + siginv[l
          if(i == j & k != l){botright[rowindex, columnindex] = -n/2*(siginv[l,i]*siginv[i,k] + siginv[l
          if(rowindex < p*(p+1)/2){rowindex = rowindex + 1}else{rowindex = 1}
        }
      }
      if(columnindex < p*(p+1)/2){columnindex = columnindex + 1}else{columnindex = 1}
    }
  }

  top = cbind(topleft, topright)
  bottom = cbind(botleft, botright)
  finalcombo = rbind(top, bottom)
  return(finalcombo)

}

fisher <-function(mu, datan, sig, maxit, tolerr, tolgrad){
  n = nrow(datan)
  p = dim(datan)[2]

  for(it in 1:maxit){
    a <- log_likelihood(datan, mu, sig,n,p)
    grad = gradient(datan, mu, sig)
    gradvec = mat2vec(grad$gradmu, grad$gradsig, p)
    info = fisher_info(datan, sig)
    dir = -solve(info) %*% gradvec
    dirmat = vec2mat(dir, p)

    mu1 = mu + dirmat$mu
    sig1 = sig + dirmat$sigma

    halve = 0;
    gradnorm = norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),typ
    print("iteration  halving    log-likelihood   ||gradient||")
```

```
    print(sprintf('%2.0f          %2.0f         %2.4f         %0.1e',it, halve,a,gradnorm))
    #check if sig1 is positive definite before sending to log likelihood
    while(!is.positive.definite(sig1) & halve <= 20){
      print(sprintf('%2.0f          %2.0f         %2.4f         %0.1e',it, halve,a,gradnorm))
      halve = halve + 1
      mu1 = mu + (dirmat$mu)/2^halve
      sig1 = sig + (dirmat$sigma)/2^halve
    }

    atmp = log_likelihood(datan, mu1, sig1,n,p)

    #continue halving if new value smaller than prior
    while (atmp < a & halve <= 20){
      #L2 norm of gradient
      gradnorm <- norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),
      print(sprintf('%2.0f          %2.0f         %2.4f         %0.1e',it, halve,atmp,gradnorm))
      halve = halve+1
      mu1 = mu + (dirmat$mu)/2^halve
      sig1 = sig + (dirmat$sigma)/2^halve
      atmp = log_likelihood(datan, mu1, sig1,n,p)
    }
    if (halve >= 20) print('Step-halving failed after 20 halvings')

    #modified relative error
    mre = max(((mat2vec(mu1, sig1, p)) - (mat2vec(mu, sig, p)))/pmax(1,abs(mat2vec(mu1, sig1, p))))
    #L2 norm of gradient
    gradnorm = norm(mat2vec(gradient(datan, mu1, sig1)$gradmu,gradient(datan, mu1, sig1)$gradsig,p),typ
    if(mre < tolerr & gradnorm < tolgrad){break}

    print('------------------------------')


    mu = mu1
    sig = sig1

  }
  paramlist <- list("mu" = mu, "sigma" = sig)
  return(paramlist)
}

sig0 <- matrix(c(1,.5,.5,.5,1,.5,.5,.5,1),3,3)
mu0 <- matrix(c(-1.5,1.5,2.3))

fisher(mu0, datan, sig0, 500, 1e-7, 1e-7)
```

```
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 1           0         -849.2651         8.1e+01"
## [1] "------------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 2           0         -731.4733         6.5e-13"
## [1] "------------------------------"
## [1] "iteration  halving    log-likelihood   ||gradient||"
## [1] " 3           0         -684.7842         2.1e-13"
```

```
## $mu
##            [,1]
## [1,] -0.9223115
## [2,]  0.9609005
## [3,]  1.9714278
##
## $sigma
##           [,1]      [,2]      [,3]
## [1,] 0.8292774 0.5622943 0.5511324
## [2,] 0.5622943 0.9279103 0.6348467
## [3,] 0.5511324 0.6348467 0.9080535
```