STAT5310 Project

Hao Xu T00732492

2024-04-18

```
IELTS <- read.csv("IELTS.csv")
```

```
pdf("pic/IELTS.pdf")
par(mar=c(7,4.5,1,1),mgp=c(1.88,0.50,0))
plot(0, type = "n", xlim = c(1, ncol(IELTS)), ylim = range(IELTS[, -1]),
     xlab = "", ylab = "Scores", main = "Scores by Nationality")

# Add lines for each nationality
for (i in 1:nrow(IELTS)) {
  lines(1:(ncol(IELTS)-1), IELTS[i, -1], type = "o", pch = i%%10, col = i)
}

# Add legend
legend("topright", legend = IELTS[,1], col = 1:nrow(IELTS), pch = (1:nrow(IELTS))%%10,
       bty = "n", cex = 0.8, y.intersp = 0.75)

text(x = 1:5,
     y = par("usr")[3] - 0.45,
     labels = colnames(IELTS)[-1],
     xpd = NA,
     srt = 45,
     cex = 1.2,
     adj = 1)
dev.off()
```

```
## pdf
##   2
```

IELTS scores Missing Data Imputation By Gibbs sampler

Utility functions

```
#### Simulate multivariate normal vector
rmvnorm<-
function(n,mu,Sigma) {
  p<-length(mu)
  res<-matrix(0,nrow=n,ncol=p)
  if( n>0 & p>0 ) {
  E<-matrix(rnorm(n*p),n,p)
  res<-t(  t(E%*%chol(Sigma)) +c(mu))
                   }
  res
```

```r
                        }
#### Simulate from the Wishart distribution
rwish<-function(n,nu0,S0)
{
  sS0 <- chol(S0)
  S<-array( dim=c( dim(S0),n ) )
  for(i in 1:n)
  {
     Z <- matrix(rnorm(nu0 * dim(S0)[1]), nu0, dim(S0)[1]) %*% sS0
     S[,,i]<- t(Z)%*%Z
  }
  S[,,1:n]
}
```

```r
Y0<-IELTS[,2:6]
Y<-Y0
n<-dim(Y)[1]
p<-dim(Y)[2]

set.seed(1)
# Create some NA values!
O<-matrix(rbinom(n*p,1,.7),n,p)
Y[O==0]<-NA

pdf("pic/stat.pdf",family="Times", height=6,width=6)
par(mar=c(1,1,.5,.5)*1.75,mfrow=c(p,p),mgp=c(1.75,.75,0))
for(j1 in 1:p) {
for(j2 in 1:p) {
 if(j1==j2){hist(Y[,j1],main="");mtext(colnames(Y)[j1],side=3,line=-.1,cex=.7)}
  if(j1!=j2) { plot(Y[,j1],Y[,j2],xlab="",ylab="",pch=16,cex=.7)}
                  }}
dev.off()
```

```
## pdf
##   2
```

```r
## prior parameters
p<-dim(Y)[2]
# Take the population mean as the prior
mu0<-apply(Y[,c(1:5)],MARGIN=2,FUN=mean,na.rm=TRUE)
# Standard deivation: 4.0~8.5 -> 95% sigma_0 = 1.1538
sd0<-rep(1.1538, 5)
L0<-matrix(.5,p,p) ; diag(L0)<-1 ; L0<-L0*outer(sd0,sd0)
nu0<-p+2 ; S0<-L0
###

### starting values
Sigma<-S0
Y.full<-Y
for(j in 1:p)
{
  Y.full[is.na(Y.full[,j]),j]<-mean(Y.full[,j],na.rm=TRUE)
}
###
```

```
### Gibbs sampler
THETA<-SIGMA<-Y.MISS<-NULL
set.seed(1)
GIBBS_start_time<- Sys.time()
for(s in 1:1000)
{

  ###update theta
  ybar<-apply(Y.full,2,mean)
  Ln<-solve( solve(L0) + n*solve(Sigma) )
  mun<-Ln%*%( solve(L0)%*%mu0 + n*solve(Sigma)%*%ybar )
  theta<-rmvnorm(1,mun,Ln)
  ###

  ###update Sigma
  Sn<- S0 + ( t(Y.full)-c(theta) )%*%t( t(Y.full)-c(theta) )
#  Sigma<-rinvwish(1,nu0+n,solve(Sn))
  Sigma<-solve( rwish(1, nu0+n, solve(Sn)) )
  ###

  ###update missing data
  for(i in 1:n)
  {
    b <- ( O[i,]==0 )
    a <- ( O[i,]==1 )
    iSa<- solve(Sigma[a,a])
    beta.j <- Sigma[b,a]%*%iSa
    s2.j   <- Sigma[b,b] - Sigma[b,a]%*%iSa%*%Sigma[a,b]
    theta.j<- theta[b] + beta.j%*%(t(Y.full[i,a])-theta[a])
    Y.full[i,b] <- rmvnorm(1,theta.j,s2.j )
  }

  ### save results
  THETA<-rbind(THETA,theta) ; SIGMA<-rbind(SIGMA,c(Sigma))
  Y.MISS<-rbind(Y.MISS, Y.full[O==0] )
  ###

  #cat(s,theta,"\n")
}
GIBBS_end_time<- Sys.time()

#### Posterior mean and correlation matrix
apply(THETA,2,mean)
```

```
## [1] 6.325467 6.457498 5.705664 6.245985 6.233816
```

```
COR <- array( dim=c(p,p,1000) )
for(s in 1:1000)
{
  Sig<-matrix( SIGMA[s,] ,nrow=p,ncol=p)
  COR[,,s] <- Sig/sqrt( outer( diag(Sig),diag(Sig) ) )
}

apply(COR,c(1,2),mean)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0000000 0.8834623 0.7915787 0.8035453 0.8782880
## [2,] 0.8834623 1.0000000 0.8206582 0.8060882 0.8836628
## [3,] 0.7915787 0.8206582 1.0000000 0.7376627 0.8145011
## [4,] 0.8035453 0.8060882 0.7376627 1.0000000 0.8238962
## [5,] 0.8782880 0.8836628 0.8145011 0.8238962 1.0000000
```

```
return
```

```
## .Primitive("return")
```

```
##### Plot the Predicted vs True value
pdf("pic/Gibbs_PredvsTrue.pdf",family="Times", height=14,width=4)
Y.true<-Y0
V<-matrix(1:p,nrow=n,ncol=p,byrow=TRUE)

v.miss<-V[O==0]
y.pred<-apply(Y.MISS,2,mean)
y.true<-Y.true[O==0]
par(mfrow=c(5,1),mar=c(3,3,1,1),mgp=c(1.75,.75,0))
for(j in 1:p){ plot(y.true[v.miss==j], y.pred[v.miss==j],
          xlab=paste("true", colnames(Y.true)[j]),
          ylab=paste("predictied", colnames(Y.true)[j]),pch=16 )
          abline(0,1)
mse <- round(mean((y.true[v.miss==j]- y.pred[v.miss==j])^2),4)
text(x=min(y.true[v.miss==j]),y=max(y.pred[v.miss==j]),paste("MSE:", mse, sep=""),
     col="BLUE", cex=1.5, adj=0,pos=4)
#cat(j, mean( (y.true[v.miss==j]- y.pred[v.miss==j])^2),"\n")
}
MSE_GIBBS <- mean((y.true- y.pred)^2)
dev.off()
```

```
## pdf
##   2
```

```
#### Convert SIGMA to an array of correlation parameters
COR<-array(dim=c(p,p,dim(SIGMA)[1]) )
for(s in 1:dim(SIGMA)[1]) {
Sig<-matrix( SIGMA[s,] ,nrow=p,ncol=p)
COR[,,s] <- Sig/sqrt(outer( diag(Sig),diag(Sig)))
                           }
colnames(COR)<-rownames(COR)<-colnames(Y)




#### Function for posterior quantile intervals for matrices
#### From the `sbgcop` package
plotci.sA<-function(sA, ylabs = colnames(sA[, , 1]), mgp = c(1.75, 0.75,
    0))
{
    qA <- qM.sM(sA)
    p <- dim(qA)[1]
    tmp <- c(qA)
    tmp <- tmp[tmp != 1]
    par(mgp = mgp)
    for (j in 1:p) {
        plot(0, 0, type = "n", ylim = range(c(tmp), na.rm = TRUE),
            xlim = c(1, p), ylab = ylabs[j], xaxt = "n", xlab = "")
        points((1:p)[-j], qA[j, -j, 2], pch = 16, cex = 0.6)
        segments(x0 = (1:p)[-j], y0 = qA[j, -j, 1], x1 = (1:p)[-j],
            y1 = qA[j, -j, 3])
        abline(h = 0, col = "gray")
        abline(v = j, col = "gray")
    }
    axis(side = 1, at = 1:p, labels = colnames(qA[, , 1]), las = 2)
}

sR.sC<-function(sC)
{
    p <- dim(sC)[1]
    s <- dim(sC)[3]
    sR <- array(dim = c(p, p, s))
    dimnames(sR) <- dimnames(sC)
    for (l in 1:s) {
        C <- sC[, , l]
        R <- C * NA
        for (j in 1:p) {
            R[j, -j] <- C[j, -j] %*% solve(C[-j, -j])
        }
        sR[, , l] <- R
    }
    sR
}

qM.sM<-function (sM, quantiles = c(0.025, 0.5, 0.975))
{
    p1 <- dim(sM)[1]
    p2 <- dim(sM)[2]
    s <- dim(sM)[3]
    qM <- array(dim = c(p1, p2, length(quantiles)))
    dimnames(qM) <- list(dimnames(sM)[[1]], dimnames(sM)[[2]],
        paste(quantiles * 100, rep("% quantile", length(quantiles)),
            sep = ""))
    for (l in 1:length(quantiles)) {
        qM[, , l] <- apply(sM, c(1, 2), quantile, prob = quantiles[l],
            na.rm = TRUE)
    }
    qM
}




#### Figure 7.4
pdf("pic/Gibbs_CI.pdf",height=6,width=6,family="Times")

par(mfcol=c(5,2),mar=c(3,2.75,1,1),mgp=c(1.75,.75,0),oma=c(1.5,0,0,0))
plotci.sA(COR)
```

```r
REG<-sR.sC(COR)
plotci.sA(REG)
dev.off()
```

```
## pdf
##   2
```

```r
CQ<-apply(COR, c(1,2), quantile,prob=c(.025,.5,.975) )

round(CQ[1,,],2)
```

```
##           Reading Listening Writing Speaking Overall
## Reading      1.00      0.78    0.63     0.65    0.77
## Listening    0.78      1.00    0.67     0.65    0.79
## Writing      0.63      0.67    1.00     0.55    0.66
## Speaking     0.65      0.65    0.55     1.00    0.67
## Overall      0.77      0.79    0.66     0.67    1.00
```

```r
round(CQ[2,,],2)
```

```
##           Reading Listening Writing Speaking Overall
## Reading      1.00      0.89    0.80     0.81    0.89
## Listening    0.89      1.00    0.83     0.82    0.89
## Writing      0.80      0.83    1.00     0.75    0.82
## Speaking     0.81      0.82    0.75     1.00    0.83
## Overall      0.89      0.89    0.82     0.83    1.00
```

```r
round(CQ[3,,],2)
```

```
##           Reading Listening Writing Speaking Overall
## Reading      1.00      0.94    0.89     0.90    0.94
## Listening    0.94      1.00    0.91     0.90    0.95
## Writing      0.89      0.91    1.00     0.86    0.91
## Speaking     0.90      0.90    0.86     1.00    0.91
## Overall      0.94      0.95    0.91     0.91    1.00
```

```r
round(apply(COR,c(1,2),mean),2)
```

```
##           Reading Listening Writing Speaking Overall
## Reading      1.00      0.88    0.79     0.80    0.88
## Listening    0.88      1.00    0.82     0.81    0.88
## Writing      0.79      0.82    1.00     0.74    0.81
## Speaking     0.80      0.81    0.74     1.00    0.82
## Overall      0.88      0.88    0.81     0.82    1.00
```

```r
pdf("pic/Posterior_vs_Prior.pdf")
par(mfrow = c(5,2), mar=c(3,3,1,1),mgp=c(1.75,.75,0))

S0_prior <- rwish(n=1000, nu0=nu0, S0)

plot(density(SIGMA[,1]),
     main=expression(paste("Posterior density of ", sigma[11],sep="")),
     xlab=expression(Sigma[11]), xlim=c(0,1.5))
lines(density(S0_prior[1,1,]),col="gray", lwd=2)
abline(v=S0[1], lty=2, col="GRAY", lwd=2)

plot(density(THETA[,1]),
     main=expression(paste("Posterior density of ", theta[1],sep="")),
     xlab=expression(theta[1]))
x_theta <- seq(4, 9, length.out=100)
lines(x_theta, dnorm(x_theta, mean=mu0[1], sd=sqrt(S0[1])),
      col="GRAY", lwd=2)
abline(v=mu0[1], lty=2, col="GRAY", lwd=2)

plot(density(SIGMA[,5*1+2]),
     main=expression(paste("Posterior density of ", sigma[22],sep="")),
     xlab=expression(Sigma[22]), xlim=c(0,1.5))
lines(density(S0_prior[2,2,]),col="gray", lwd=2)
abline(v=S0[5*1+2], lty=2, col="GRAY", lwd=2)

plot(density(THETA[,2]),
     main=expression(paste("Posterior density of ", theta[2],sep="")),
     xlab=expression(theta[2]))
x_theta <- seq(4, 9, length.out=100)
lines(x_theta, dnorm(x_theta, mean=mu0[2], sd=sqrt(S0[5*1+2])),
      col="GRAY", lwd=2)
abline(v=mu0[2], lty=2, col="GRAY", lwd=2)

plot(density(SIGMA[,5*2+3]),
     main=expression(paste("Posterior density of ", sigma[33],sep="")),
     xlab=expression(Sigma[33]), xlim=c(0,1.5))
lines(density(S0_prior[3,3,]),col="gray", lwd=2)
abline(v=S0[5*2+3], lty=2, col="GRAY", lwd=2)

plot(density(THETA[,3]),
     main=expression(paste("Posterior density of ", theta[3],sep="")),
     xlab=expression(theta[3]))
x_theta <- seq(4, 9, length.out=100)
lines(x_theta, dnorm(x_theta, mean=mu0[3], sd=sqrt(S0[5*2+3])),
      col="GRAY", lwd=2)
abline(v=mu0[3], lty=2, col="GRAY", lwd=2)

plot(density(SIGMA[,5*3+4]),
     main=expression(paste("Posterior density of ", sigma[44],sep="")),
     xlab=expression(Sigma[44]), xlim=c(0,1.5))
lines(density(S0_prior[4,4,]),col="gray", lwd=2)
abline(v=S0[5*3+4], lty=2, col="GRAY", lwd=2)

plot(density(THETA[,4]),
     main=expression(paste("Posterior density of ", theta[4],sep="")),
     xlab=expression(theta[4]))
x_theta <- seq(4, 9, length.out=100)
lines(x_theta, dnorm(x_theta, mean=mu0[4], sd=sqrt(S0[5*3+4])),
      col="GRAY", lwd=2)
abline(v=mu0[4], lty=2, col="GRAY", lwd=2)

plot(density(SIGMA[,5*4+5]),
     main=expression(paste("Posterior density of ", sigma[55],sep="")),
     xlab=expression(Sigma[55]), xlim=c(0,1.5))
lines(density(S0_prior[5,5,]),col="gray")
```

```
abline(v=S0[5*4+5], lty=2, col="GRAY", lwd=2)

plot(density(THETA[,5]),
     main=expression(paste("Posterior density of ", theta[5],sep="")),
     xlab=expression(theta[5]))
x_theta <- seq(4, 9, length.out=100)
lines(x_theta, dnorm(x_theta, mean=mu0[5], sd=sqrt(S0[5*4+5])),
      col="GRAY", lwd=2)
abline(v=mu0[5], lty=2, col="GRAY", lwd=2)
dev.off()
```

```
## pdf
##   2
```

IELTS scores Missing Data Imputation By Gibbs sampler

```
em_algorithm <- function(data, max_iter = 100, tol = 1e-6) {
    # Initial estimates
    means <- apply(data, 2, function(x) mean(x, na.rm = TRUE))
    sigma <- cov(data, use = "complete.obs")

    # Iteration control
    iter <- 1
    conv <- FALSE

    while (iter <= max_iter && !conv) {
        # E-step: Estimate missing values
    #print(iter)
        data_imputed <- data
        for (i in 1:nrow(data)) {
            missing <- is.na(data[i,])
            if (any(missing)) {
                obs <- !missing
                mu_obs <- means[obs]
                mu_miss <- means[missing]
                Sigma_oo <- sigma[obs, obs]
                Sigma_om <- sigma[obs, missing]
                Sigma_mm <- sigma[missing, missing]
                Sigma_mo <- t(Sigma_om)
            s2.j   <- Sigma_mm - Sigma_mo%*%solve(Sigma_oo)%*%Sigma_om

                # Conditional mean and covariance
                cond_mean <- mu_miss + (Sigma_mo %*% solve(Sigma_oo)) %*% (t(data[i, obs]) - mu_obs)
                data_imputed[i, missing] <- rmvnorm(1, cond_mean, s2.j)
            }
        }

        # M-step: Re-estimate parameters
        old_means <- means
        old_sigma <- sigma

        means <- colMeans(data_imputed, na.rm = TRUE)
        sigma <- cov(data_imputed, use = "pairwise.complete.obs")
```

```r
        # Check for convergence
        conv <- max(abs(old_means - means), abs(sigma - old_sigma)) < tol
        iter <- iter + 1
    }

    return(list(means = means, sigma = sigma, data_imputed = data_imputed))
}
```

```r
EM_start_time<- Sys.time()
em_result <- em_algorithm(Y)
EM_end_time<- Sys.time()
```

```r
##### Plot the Predicted vs True value
pdf("pic/EM_PredvsTrue.pdf",family="Times", height=14,width=4)
Y.true<-Y0
V<-matrix(1:p,nrow=n,ncol=p,byrow=TRUE)

v.miss<-V[O==0]
y.pred <- em_result$data_imputed
y.pred <- y.pred[O==0]
y.true<-Y.true[O==0]
MSE_EM<- mean((y.true- y.pred)^2)
par(mfrow=c(5,1),mar=c(3,3,1,1),mgp=c(1.75,.75,0))
for(j in 1:p){ plot(y.true[v.miss==j], y.pred[v.miss==j],
          xlab=paste("true", colnames(Y.true)[j]),
          ylab=paste("predictied", colnames(Y.true)[j]),pch=16 )
          abline(0,1)
mse <- round(mean((y.true[v.miss==j]- y.pred[v.miss==j])^2),4)
text(x=min(y.true[v.miss==j]),y=max(y.pred[v.miss==j]),paste("MSE:", mse, sep=""),
     col="BLUE", cex=1.5, adj=0,pos=4)
  cat(j, mean( (y.true[v.miss==j]- y.pred[v.miss==j])^2),"\n") }
```

```
## 1 0.1027275

## 2 0.03862703

## 3 0.03020887

## 4 0.249199

## 5 0.02529312
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
print(MSE_GIBBS)
```

```
## [1] 0.03895665
```

```r
print(GIBBS_end_time - GIBBS_start_time)
```

```
## Time difference of 10.48454 secs
```

```r
print(MSE_EM)
```

```
## [1] 0.06930736
```

```r
print(EM_end_time - EM_start_time)
```

```
## Time difference of 1.251888 secs
```