Gaussian EM Algorithm

Greg Johnson

```
EM_normal<-function(data,mu0=c(0,0,0),E0=diag(c(1,1,1)),tol=1e-6){
  muhat<-mu0
  Ehat<-E0

  X<-data
  n<-nrow(X)

  mre<-10000 #modified relative error - something large/arbitrary to enter the while loop
  while(mre>tol){
    #predict

    T1<-0
    T2<-0

    for(j in 1:n){
      if(sum(is.na(X[j,]))==0){next}
      mis<-is.na(X[j,])
      x1<-X[j,mis] #missing components
      x2<-X[j,!mis] #not missing
      #we estimate x1 with the conditional expectation of x1 given x2
      muhat1<-muhat[mis]
      muhat2<-muhat[!mis]
      Ehat11<-Ehat[mis,mis]
      Ehat12<-Ehat[mis,!mis]
      Ehat21<-Ehat[!mis,mis]
      Ehat22<-Ehat[!mis,!mis]

      x1est<-muhat1+Ehat12%*%solve(Ehat22)%*%(x2-muhat2)

      x<-X[j,]
      x[mis]<-x1est
      T1<-T1+x   #estimated complete data contribution to the sufficient statistic T1

      x1x1Test<-Ehat11-Ehat12%*%solve(Ehat22)%*%Ehat21+x1est%*%t(x1est)
      xxT<-X[j,]%*%t(X[j,])
      xxT[mis,mis]<-x1x1Test
      xxT[!mis,mis]<-x2%*%t(x1est)
      xxT[mis,!mis]<-x1est%*%t(x2)

      T2<-T2+xxT   #estimated complete data contribution to the sufficient statistic T2

    }
    #save initial estimate to compute mre
    muhat_old<-muhat
    Ehat_old<-Ehat

    #estimate the new
    muhat<- 1/n*T1
```

```
    Ehat<-1/n*T2-muhat%*%t(muhat)

    #calculate mre
    par_old<-c(muhat_old,Ehat_old[lower.tri(Ehat_old,diag=TRUE)])
    par_new<-c(muhat,Ehat[lower.tri(Ehat,diag=TRUE)])
    mre<-sqrt(sum((par_old-par_new)^2))/max(1,sqrt(sum(par_new^2)))
  }
  return(list(muhat,Ehat))
}
```

Now let's test our EM-algorithm on some sample data:

```
X<-matrix(c(3,6,0,4,4,3,NA,8,3,5,NA,NA),4,byrow=TRUE)
```

For our initial estimate of μ, we'll compute sample estimates with the data points we have. For our initial estimate of Σ, we'll impute the missing data with the respective initial estimate of μ.

```
mu0<-apply(X,2,mean,na.rm=TRUE)
Xtemp<-X
Xtemp[3,1]<-mu0[1]
Xtemp[4,2]<-mu0[2]
Xtemp[4,3]<-mu0[3]
E0<-cov(Xtemp)
EM_normal(X,mu0,E0)
```

```
## [[1]]
## [1] 2.489376 4.017141 1.506429
##
## [[2]]
##           [,1]      [,2]     [,3]
## [1,]  6.197220 10.000536 3.750204
## [2,] 10.000536 16.138010 6.051758
## [3,]  3.750204  6.051758 2.269411
```