final

uni:xw2598

2019/5/11

Question1

```
#data manipulation
dt= read.csv("./hurrican356.csv")
dt$date = substr(x = as.character(dt$time),start = 2,stop = 10)
dt$day = yday(dt$date)
dt$hour = substr(x = as.character(dt$time),start = 2,stop = 18) %>%
  str_c("19",.) %>% as_datetime()
dt$hours = dt$Latitude
cur = dt$hour[1]
for(i in 2:nrow(dt)){
  if(dt$ID[i] == dt$ID[i-1]){
    dt$hours[i] <- difftime(dt$hour[i],cur,units = "hours") %>% as.numeric()}
  else{dt$hours[i]= 0
  cur = dt$hour[i]}
}
dt$hours[1]=0
data = dplyr::select(dt,ID,year = Season,type = Nature,Latitude,Longitude,wind = Wind.kt,day,hours) %>%
cur_lat = data$Latitude[1]
cur_long = data$Longitude[1]
cur_wind = data$wind[1]
data$delta_lat=data$delta_long=data$delta_wind=data$Latitude
for (i in 2:nrow(data)) {
  if(dt$ID[i] == dt$ID[i-1]){
    data$delta_lat[i] = data$Latitude[i]-data$Latitude[i-1]
    data$delta_long[i] = data$Longitude[i]-data$Longitude[i-1]
    data$delta_wind[i] = data$wind[i]-data$wind[i-1]
  }
  else{
    cur_lat = data$Latitude[i]
    cur_long = data$Longitude[i]
    cur_wind = data$wind[i]
    data$delta_lat[i] =0
    data$delta_long[i] = 0
    data$delta_wind[i] = 0
  }
}
data$delta_lat[1]=data$delta_long[1]=data$delta_wind[1]=0
head(data,5)
```

```
##             ID year type Latitude Longitude wind day hours delta_wind
## 1 ALLISON.1989 1989   TS     27.0     -96.0   30 175     0          0
## 2 ALLISON.1989 1989   TS     27.0     -96.0   30 176     6          0
## 3 ALLISON.1989 1989   TS     27.2     -96.0   30 176    12          0
## 4 ALLISON.1989 1989   TS     27.4     -95.9   30 176    18          0
## 5 ALLISON.1989 1989   TS     27.6     -95.8   30 176    24          0
```

```
##   delta_long delta_lat
## 1        0.0       0.0
## 2        0.0       0.0
## 3        0.0       0.2
## 4        0.1       0.2
## 5        0.1       0.2
```

(1)Randomly select 80% hurricanes

```
data$type <- as.numeric(data$type)
set.seed(1234)
training.samples = sample(c(1:356),size = 285,replace = FALSE)
dat.train = data[as.numeric(data$ID) %in% training.samples, ]
`%not_in%` <- purrr::negate(`%in%`)
dat.test <- data[as.numeric(data$ID) %not_in% training.samples,]
```

(2)develop an MCMC algorithm to estiamte the posterior mean of the model parameters.

$$Y_{ij}(t)|Y_{ij}(t-6) \sim N(\mu_{ij}(t-6) + \rho_j Y_{ij}(t-6), \Sigma)$$

$$P(Y_{ij}(t)|Y_{ij}(t-6)) \propto (\frac{1}{\sqrt{|\Sigma|}})^m exp - \frac{1}{2}((Y_{ij} - \mu_{ij}(t-6) - \rho_j Y_{ij}(t-6))\Sigma^{-1}(Y_{ij} - \mu_{ij}(t-6) - \rho_j Y_{ij}(t-6)))$$

So the likelihood is:

$$L(Y_{ij}) = P(Y_{ij}(t)|Y_{ij}(t-6)) * P(Y_{ij}(t-6)|Y_{ij}(t-12)...)$$

loglikelihood is:

$$L(Y_{ij}) \propto \sum K\Sigma^{-1} * K^T$$

K is a 1*3 matrix

$$K = ((Y_{i1} - \mu_{i1}(t-6) - \rho_j Y_{i1}(t-6) \quad Y_{i2} - \mu_{i2}(t-6) - \rho_j Y_{i2}(t-6) \quad Y_{i3} - \mu_{i3}(t-6) - \rho_j Y_{i3}(t-6))$$

The prior function is :

$$\pi(\beta_1, ..., \beta_m|Y_1, Y_2, ...Y_n) * \pi(\rho_1|Y_1, Y_2, ...Y_n)\pi(\rho_2|Y_1, Y_2, ...Y_n)\pi(\rho_3|Y_1, Y_2, ...Y_n)\pi(\Sigma^{-1})$$

```
x.train = dplyr::select(dat.train,day,year,type,starts_with("delta"))
Y.train = dplyr::select(dat.train,Latitude,Longitude,wind)
beta = matrix(nrow = 7 ,ncol = 3)
id = distinct(dat.train,ID)

#loglikelihood function

logp = matrix(nrow = nrow(distinct(dat.train,ID)),ncol = 1)

loglike = function(Y,X,rho,cov,beta){
  for (i in 1:nrow(distinct(dat.train,ID))) {
      y = Y[which(dat.train$ID == id[i,]),]
      x = X[which(dat.train$ID == id[i,]),]
```

```
      logp[i,]=0
      for(m in length(Y):2){
      u = beta[1,]+x[m,1]*beta[2,]+x[m,2]*beta[3,]+x[m,3]*beta[4,]+x[m,4]*beta[5,]+x[m,5]*beta[6,]+x[m,
      a =as.matrix(y[m,]-u-rho*y[m-1,])
      logp[i,] = logp[i,] - 1/2*log(det(cov))-1/2*a%*%solve(cov)%*%t(a)
      }
  }
  return(sum(logp[,1],na.rm = TRUE))
}

#set current value for test

beta = matrix(rep(1,21),nrow = 7,ncol = 3)
cov=rWishart(3,diag(0.1,3),inverse=TRUE)
rho = c(0.1,0.2,0.3)
a = loglike(Y.train,x.train,rho,cov,beta)

#prior function
logprior = function(beta,rho,cov){
   return(dWishart(solve(cov),diag(0.1,3),3)*dtruncnorm(rho[1],a=0,b=1,mean=0.5,sd = 1/5)*dtruncnorm(rh

}
#test
#a = logprior(beta,rho,cov)
#posterio function
logpost = function(X,Y,rho,cov,beta){
  return(loglike(Y,X,rho,cov,beta)+logprior(beta,rho,cov))
}
#test
#a = logpost(x.train,Y.train,rho,cov,beta)
```

```
MHstep = function(pars,avec,Y,X){
  res = pars
  npars = length(pars)
  for (i in 1:npars) {
    prop = res
    prop[i] = res[i]+2*avec[i]*(runif(1)-0.5)
    a = matrix(prop[4:12],ncol=3)
    b= matrix(prop[13:33],ncol=3)
    c=matrix(res[4:12],ncol=3)
    d=matrix(res[13:33],ncol=3)
    if(log(runif(1))<(logpost(X,Y,prop[1:3],a,b)-logpost(X,Y,res[1:3],c,d)))
    res[i]=prop[i]
  }
  return(res)
}
#test
pars = c(rho,as.vector(cov),as.vector(beta))
avec = c(rep(0.1,3),rep(1,9),rep(1,21))
```

```
nCores <- 4
registerDoParallel(nCores)
#nrep = 1000
nrep = 3
```

```
avec = c(rep(0.1,3),rep(1,9),rep(1,21))
mchain = matrix(NA,nrow = nrep, ncol = 33)
mchain[1,] = c(rho,as.vector(cov),as.vector(beta))
for(i in 2:nrep){
  mchain[i,]=MHstep(mchain[i-1,],avec,Y.train,x.train)
}
mchain <- foreach(i = 2:nrep, .combine = rbind) %dopar% {
  mchain[i,]=MHstep(mchain[i-1,],avec,Y.train,x.train)
}
```

It took hundreds of year to run out if we choose a mcmc chain with length 10000. So for the following steps, I just set the length of mcmc chain as 3, which may be inaccurate for estimation due to small sample size,but it provides a correct thought about the whole MCMC algorithm.

```
res = colMeans(mchain,na.rm = TRUE)
rho_hat = res[1:3]
cov_hat = matrix(res[4:12],ncol=3)
beta_hat =matrix(res[13:33],ncol=3)
print(rho_hat)
```

```
## [1] 0.04538301 0.25404835 0.25869149
```

```
print(cov_hat)
```

```
##          [,1]      [,2]       [,3]
## [1,] 7.597283  2.393415  0.5222042
## [2,] 2.757443 11.453848 -5.2677612
## [3,] 1.087405 -5.692469  7.6749997
```

```
print(beta_hat)
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.6306504 0.5389167 0.2065795
## [2,] 0.7026216 1.0000000 0.7462638
## [3,] 1.0000000 0.5884699 0.5593908
## [4,] 0.5034655 0.5401081 0.6986511
## [5,] 0.6649451 0.3479323 0.0640423
## [6,] 1.6546269 1.8987827 1.0000000
## [7,] 0.8705581 0.2092637 0.5451426
```

The estimated posterior mean of parameters(recorded by once set nrep = 4)

$$\rho_1 = 0.0984\rho_2 = 0.23436918\rho_3 = 0.29836776$$

Σ= 10.699668 4.556651 3.047737 4.668151 7.217416 4.595182 3.455998 4.694572 8.920684

β_{ij} = [1,] 0.6308118 1.1485184 0.8532505 [2,] 0.5516681 1.1815708 1.0000000 [3,] 0.7783336 0.5053999 0.8885836 [4,] 0.7720843 1.4161843 1.0000000 [5,] 0.9741097 1.3487143 0.3051520 [6,] 0.8998162 0.4097745 1.9334911 [7,] 0.6070781 1.1090746 0.5406332

apply the model

```
x.test = dplyr::select(dat.test,day,year,type,starts_with("delta"))
y.test = dplyr::select(dat.test,Latitude,Longitude,wind)
```