Appendix

Link to our Git Repository

https://github.com/kkung111/MA578_Project.git

Multivariate Normal Model Code

```
library(MCMCpack)
library(mvtnorm)

#hold our values for each area
output_th <- list(rep(0,9))
output_s2 <- list(rep(0,9))
output_yt <- list(rep(0,9))

#helper functions from class
rmv<-function(n,mu,Sigma){    # samples Y~MVN(mu,Sigma)
  cm<-chol(Sigma);d<-dim(Sigma)[1]
  Y0<-matrix(rnorm(n*d),nrow=d)
  t(cm)%*%Y0 + mu
}
riw<-function(n,nu0,Sm){ # Sigma~IW(nu0,Sigma^(-1)); requires rmv
  m<- solve(Sm)
  sapply(1:n,function(i)
    solve(crossprod(t(rmv(nu0*n,0,m))[(i-1)*nu0+1:nu0,])), simplify = 'array')
}

for (k in seq(1,9)) {
precipData<-as.matrix(read.table(paste("Area",k,"MinMaxDataV2.tsv"), sep = "\t"))

precipDataCov<-cov(precipData)

#find the prior means
precipDataMinMu0<-170
precipDataMaxMu0<-260

#store them into a matrix
precipDataMu0<-matrix(c(precipDataMinMu0, precipDataMaxMu0), nrow = 2)


#prior standard deviations of theta
precipDataMins20<-30
precipDataMaxs20<-30

#prior Sigma0
precipDatas20<-matrix(c(precipDataMins20, 0, 0, precipDataMaxs20), nrow = 2)


#Gibbs Sampler
#Starting values
ybW<-apply(precipData, 2, mean)
nW<-dim(precipData)[1]
nu0<-2 #note: had to change this because was giving me errors with 1
nun<-nu0 + nW

set.seed(1234+k)
nSim<-20000

SigmaW<-riwish(nu0, precipDataCov)
thetaW<-rmvnorm(1, precipDataMu0, precipDatas20)


#initiate the matrices to hold our values
THW<-S2W<-YtW<-NULL

for(i in 1:(nSim+1000)){
  #sample Sigma
  LnW<-precipDataCov + crossprod(precipData - outer(rep(1, nrow(precipData)), c(thetaW)))
  SigmaW<-riw(1, nun, LnW)[,,1]


  #sample theta
  LN<-solve(solve(precipDatas20) + nW *solve(SigmaW))
  munW<-LN%*%(solve(precipDatas20)%*%precipDataMu0 + nW*solve(SigmaW)%*%ybW)
  thetaW<-rmv(1, munW,LN)


  #prediction
  #we didn't take into account of the first 1000 simulations during the "warm up" period"
  if(i > 1000) {
  yPred<-rmv(1, thetaW, SigmaW)
  #restrict predictions to be between 0 and 365
  while(yPred[1]<0 | yPred[2]>365){yPred<-rmv(1, thetaW, SigmaW)}
  ytW<-yPred
  THW<-cbind(THW, thetaW)
  S2W<-cbind(S2W, c(SigmaW))
  YtW<-cbind(YtW, ytW)
  }

}

rowMeans(THW)
rowMeans(S2W)
rowMeans(YtW)

output_th[[k]] <- t(THW)
output_s2[[k]] <- t(S2W)
output_yt[[k]] <- t(YtW)
}

#assumption checking:
#plot Seattle Data
```

```r
plot(density(as.matrix(read.table(paste("Area",4,"MinMaxDataV2.tsv"), sep = "\t"))[,1],
     kernel = "epanechnikov"),lty = 1,xlim=c(1,365),main = "Region 4 Data Distribution",
     xlab = "Day of Year",ylim = c(0,1/50))
points(density(as.matrix(read.table(paste("Area",4,"MinMaxDataV2.tsv"), sep = "\t"))[,2],
     kernel = "epanechnikov"),lty = 2,type="l")
legend("topleft", c("Dry Start","Wet Start"),lty = c(1,2))

#analysis
#thin out the data and keep copies of the original data
seqLength<-seq(1, dim(output_yt[[1]])[1], 10)
tempoutput_yt<-output_yt
tempoutput_th<-output_th
tempoutput_s2<-output_s2

output_yt<-lapply(output_yt, function(x){x[seqLength,]})
output_th<-lapply(output_th, function(x){x[seqLength,]})
output_s2<-lapply(output_s2, function(x){x[seqLength,]})

#find the effectice Sizes and acf
#effective size calculation
effSizeth<-matrix(unlist(lapply(tempoutput_th, function(x){effectiveSize(x)})), ncol = 2, byrow = T)
effSizes2<-matrix(unlist(lapply(tempoutput_s2, function(x){effectiveSize(x)})), ncol = 2, byrow = T)

colMeans(effSizeth)
colMeans(effSizes2)


#plot some of the acf plots
par(mfrow=(c(2,2)))
acf(tempoutput_th[[3]][,1], main = expression(paste("ACF for ", theta, " for Seattle Area")))
acf(output_th[[3]][,1], main = expression(paste("ACF for Thinned ", theta, " for Seattle Area")))
acf(tempoutput_s2[[3]][,1], main = expression(paste("ACF for ", Sigma, " for Seattle Area")))
acf(output_s2[[3]][,1], main = expression(paste("ACF for Thinned ", Sigma, " for Seattle Area")))

#start analysis

# Theta Means and Variance
round(matrix(unlist(lapply(output_th,function(x) {apply(x,2,mean)})),ncol=2,byrow=T))
round(matrix(unlist(lapply(output_th,function(x) {apply(x,2,var)})),ncol=2,byrow=T))

# First/Last Dates for each area
start_dates <- matrix( unlist(lapply(output_yt,function(x) {x[,1]} )),
              nrow = length(seqLength),ncol=9,byrow=F)
table(apply(start_dates,1,which.min))/length(seqLength)

end_dates <- matrix( unlist(lapply(output_yt,function(x) {x[,2]} )),
             nrow = length(seqLength),ncol=9,byrow=F)
table(apply(start_dates,1,which.max))/length(seqLength)

# Length of summer
summer_length <- matrix( unlist(lapply(output_yt,function(x) {x[,2]-x[,1]} )),
                 nrow = length(seqLength),ncol=9,byrow=F)
round(apply(summer_length,2,mean))
round(sqrt(apply(summer_length,2,var)))
```

Daily Precipitation Chance Model

```
library(coda)
# pi-wet (beta), pi-dry (Beta), dry-start (normal mu_s), dry-end (normal)
#tf_weather_data2 <- weatherDat >= .5
# see code in create_9part_dataset.R to manipulate this into a list of lists.
# this uses an intermediate result - i.e. a list of 9 [ list of 46 [vector of 275 T/F]]
# Parameterize it by mindate,minval, a, b
drop_days <-90
mindate_all <- list(rep(0,9))
minval_all<-list(rep(0,9))
a_all <- list(rep(0,9))
b_all <- list(rep(0,9))
obsrain_all <- list(rep(0,9))
prain_all <- list(rep(0,9))
posterior_means <- matrix(nrow=9,ncol=4)



for (l in seq(1,9)){
  data_met <- tf_data2[[l]]
  year_aggregates <-  apply(matrix(as.numeric(unlist(data_met)),ncol=365,byrow=T),
          2,sum)[(drop_days+1):365]
#priors
mindate_mean <- mindate <- 210-drop_days
mindate_sd <- 30
minval_a  <- 5
minval_b <- 20
a_mean <-  -.6/150
a_sd <- abs(a_mean)/2
b_mean <- b <- .6/150
b_sd <- abs(b_mean)/2
minval <- minval_a / (minval_a + minval_b)

S <- 10000
MINDATE <- MINVAL <- A <- B <- rep(0,S)
OBSRAIN <- PRAIN <- matrix(ncol=365-drop_days,nrow=S)
accept_probs <- rep(0,S+500)

delta <- .095
llike <- function(y,theta_min,theta_val,a,b){
  # takes in the list of 365 data points
  after_min <- c(rep(0,floor(theta_min)),rep(1,365-drop_days-floor(theta_min)))
  j <- seq(1,(365-drop_days))
  prob <- theta_val + (j-theta_min)*(a*(1- after_min)+ b*after_min)
  prob <- sapply(prob,function(x) {max(min(x,1),0)})
  llike <- sum(dbinom(y,46,prob,log=T))
  return(llike)
}
```

```r
prob_by_day <- function(theta_min,theta_val,a,b)
{
  after_min <- c(rep(0,floor(theta_min)),rep(1,365-drop_days-floor(theta_min)))
  j <- seq(1,(365-drop_days))
  prob <- theta_val + (j-theta_min)*(a*(1- after_min)+ b*after_min)
  prob <- sapply(prob,function(x) {max(min(x,1),0)})
  return(prob)
}

accept <- 0
for( i in seq(1,(S+500))){
 mindate.star <- rnorm(1,mindate,mindate_sd*delta)
 minval.star <- rnorm(1,minval,sqrt(minval_a*minval_b/((minval_a+minval_b)^2*
                (minval_a+minval_b+1)))*delta)
 a.star <- rnorm(1,a,delta*a_sd)
 b.star <- rnorm(1,b,delta*b_sd)

  log.r <- llike(year_aggregates,mindate.star,minval.star,a.star,b.star)
    -llike(year_aggregates,mindate,minval,a,b) +
    dnorm(a.star,a_mean,a_sd,log=T) + dbeta(minval.star,minval_a,minval_b,log=T) +
    dnorm(a.star,a_mean,a_sd,log=T) + dnorm(b.star,b_mean,b_sd,log=T) -
    dnorm(a,a_mean,a_sd,log=T) - dbeta(minval,minval_a,minval_b,log=T) -
    dnorm(a,a_mean,a_sd,log=T) - dnorm(b,b_mean,b_sd,log=T)

  accept_probs[i] <- exp(log.r)

  if(log(runif(1))<log.r){
    if(i > 500) accept <- accept+1
    #print("Accept")
    a <- a.star
    b <- b.star
    minval <- minval.star
    mindate <- mindate.star
  }

  output_filter <- 50
  if((i > 500) ) {
      A[i-500] <- a
      B[i-500] <- b
      MINDATE[i-500] <- mindate
      MINVAL[i-500] <- minval
      PRAIN[i-500,] <- prob_by_day(mindate,minval,a,b)
      OBSRAIN[i-500,] <- rbinom(365-drop_days,46,prob_by_day(mindate,minval,a,b))/46
  }
}
MINDATE <- MINDATE[seq(1,S,by=50)]
MINVAL <- MINVAL[seq(1,S,by=50)]
A <- A[seq(1,S,by=50)]
B <- B[seq(1,S,by=50)]
OBSRAIN <- OBSRAIN[seq(1,S,by=50),]
PRAIN <- PRAIN[seq(1,S,by=50),]

sum(accept)/S
```

```r
c(effectiveSize(MINDATE),effectiveSize(MINVAL),effectiveSize(A),effectiveSize(B))
c(mean(MINDATE),mean(MINVAL),mean(A),mean(B))
c(var(MINDATE),var(MINVAL),var(A),var(B))

plot(density(MINDATE+drop_days,adjust = 1.5),xlab = "Day of Year",
     main = "Distribution of Nicest day of the Year")
plot(seq(drop_days+1,365),year_aggregates/46,ylab = "Probability of Rain",
     xlab = "Day of Year",main = "Probability of Rain by Day\nWith 95% Credible Interval")
points(seq(drop_days+1,365),prob_by_day(mean(MINDATE),mean(MINVAL),mean(A),mean(B)),type="l")
points(seq(drop_days+1,365),apply(OBSRAIN,2,quantile,.975),type="l",col="red")
points(seq(drop_days+1,365),apply(OBSRAIN,2,quantile,.025),type="l",col="red")

mindate_all[[l]] <- MINDATE
minval_all[[l]] <- MINVAL
a_all[[l]] <- A
b_all[[l]] <- B
obsrain_all[[l]] <- OBSRAIN
prain_all[[l]] <- PRAIN
posterior_means[l,] <- c(mean(MINDATE),mean(MINVAL),mean(A),mean(B))
}


#Analysis Ideas:
# Biggest Change

#P escape rain by day
p_escape_rain <- rep(0,365-drop_days)
for(k in 1:(365-drop_days)){
  p_escape_rain[k] <- mean(sapply(prain3[,k],rbinom,n=1,size=1)&
                      sapply(1-prain6[,k],rbinom,n=1,size=1))
}
plot(seq(drop_days+1,365) , p_escape_rain , xlab = "day of year",
     ylab = "Probability of Escaping Rain",main =
    "Daily Chance that Area 3 rains and Area 6 is Dry")


#plot dist of all nicest days
library(ggplot2)
plt_df <- data.frame(region = rep("1",512),DayOfYear =
          density(mindate_all[[1]]+drop_days,adjust = 1.5)$x,Density =
          density(mindate_all[[1]]+drop_days,adjust = 1.5)$y)
for(l in seq(2,9)){
  plt_df <- rbind(plt_df,data.frame(region = rep(paste(l),512),DayOfYear =
            density(mindate_all[[l]]+drop_days,adjust = 1.5)$x,Density =
            density(mindate_all[[l]]+drop_days,adjust = 1.5)$y))
}

ggplot(data=plt_df, aes(x=DayOfYear, y=Density, group=region) )
      +geom_line(aes(color=region)) + ggtitle("Posterior Distributions
      of the Nicest Day of the Year")


par(mfrow = c(3,3))
```

```r
for (l in 1:9) {
  plot(seq(drop_days+1,365),apply(matrix(as.numeric(unlist(tf_data2[[l]])),ncol=365,
      byrow=T),2,sum)[(drop_days+1):365]/46,ylab = "Probability of Rain",xlab =
      "Day of Year",main = paste("Region",l))
  points(seq(drop_days+1,365),prob_by_day(mean(mindate_all[[l]]),mean(minval_all[[l]]),
      mean(a_all[[l]]),mean(b_all[[l]])),type="l")
  points(seq(drop_days+1,365),apply(obsrain_all[[l]],2,quantile,.975),type="l",col="red")
  points(seq(drop_days+1,365),apply(obsrain_all[[l]],2,quantile,.025),type="l",col="red")
}
par(mfrow = c(1,1))

#Escape Seattle Rain
escape_p <- seq(1,365-drop_days)
num_escape <- seq(1,365-drop_days)
for( i in seq(1,365-drop_days)){
  searain <- rbinom(length(prain_all[[3]][,i]),1,prain_all[[3]][,i])
  eastdry <- rbinom(length(prain_all[[3]][,i]),1,1-prain_all[[6]][,i])
  escape_p[i] <-  sum(searain*eastdry)/sum(searain)
  num_escape[i] <- sum(searain*eastdry)/length(searain*eastdry)
}
plot(seq(drop_days+1,365),escape_p,ylim = c(0,1),col="blue",cex=.5,
     xlab = "Day Of Year",ylab = "Probability",main = "Escaping Seattle Rain",pch=16)
points(seq(drop_days+1,365),num_escape,col="red",cex=.5,pch=16)
legend("topright",legend = c("Given Rain in Seattle","Unconditional Probability"),
       col = c("blue","red"),cex = .5,pch=c(16,16))
```