Problem 1: Using the given data, we will attempt to improve forecasts of hurricane intensity (VMAX, the maximum wind velocity), using other state variables that are collected, including HWRF. We will use linear regression to model VMAX, with a random effect, θ_{b_i} for the ocean basin the hurricane occurs in, where

$$Y_i \sim \mathcal{N}(\alpha + \sum_{j=1}^{p} X_{ij}\beta_j + \theta_{b_i}, \sigma^2)$$

It is reasonable to assume that the response, VMAX, is Gaussian, so linear regression is an appropriate choice for a model here. Since I have no prior information about the effects of any of the covariates, we will use an uninformative prior for each of the linear regression coefficients, $\beta_j \sim \mathcal{N}(0, 1/\tau_e^2)$ where $\tau_e \sim \text{Gamma}(0.1, 0.1)$, and $\alpha \sim \mathcal{N}(0, 100^2)$. We will also use an uninformative prior for the random effects, as I do not have prior information on whether the ocean basis affects the prediction, so $\theta_{b_i} \sim \mathcal{N}(0, \tau)$ and $\tau \sim \text{Gamma}(0.1, 0.1)$. For the variance, σ^2, we will also use an uninformative prior, $\sigma^2 = \frac{1}{\tau_e^2}$, where $\tau_e \sim \text{Gamma}(0.1, 0.1)$. The model will then use all quantitative covariates ($p = 18$), columns 5 through 22 in the original data set, as predictors. After running the model, we will check the Gelman-Rubin statistic for each parameter to ensure convergence. Each β_i has a Gelman-Rubin statistic of 1. α and θ_{b_i} have Gelman-Rubin statistics of 1.09, so we can be reasonably confident in the convergence, since all values are below 1.1.

Problem 2: We will compare the linear regression model with a random effect for the ocean basins ($\mathcal{M}_1$) with a linear regression model without a random effect,

$$\mathcal{M}_2 : Y_i \sim \mathcal{N}(\alpha + \sum_{j=1}^{p} X_{ij}\beta_j, \sigma^2)$$

$\mathcal{M}_2$ has the same priors as $\mathcal{M}_1$, with the absence of a random effect. Since we are ultimately concerned with prediction, we will use cross-validation, splitting the data into five groups to compare $\mathcal{M}_1$ and $\mathcal{M}_2$. $\mathcal{M}_1$ has a slightly lower mean absolute deviation of 9.05 and a mean-squared error of 152.01, compared to $\mathcal{M}_2$ which has a mean absolute deviation of 9.18 and a mean squared error of 152.48. While the difference between the mean absolute deviation and the mean squared error between the models is minimal, we are looking to improve the predictions even if it only improves slightly, so we will use $\mathcal{M}_1$, the model with a random effect to make our predictions, but it would also be reasonable to choose $\mathcal{M}_2$, as there is not a major difference in errors between the two models.

Problem 3: To verify that this model fits the data, we will use posterior predictive checks. We are assuming that the response is Gaussian, so we are most concerned with the skew of the data. Therefore, we will check the range of the predictions. The range of the data falls within the range of the model, as shown in Figure 1. The p-value is 0.87, which is not close to 0 or 1, so we will accept that it is reasonable to model the response using a Gaussian model.

Problem 4: To determine variable importance, we will use stochastic search variable selection (SSVS) to determine the selection probability of each covariate in the model. There are

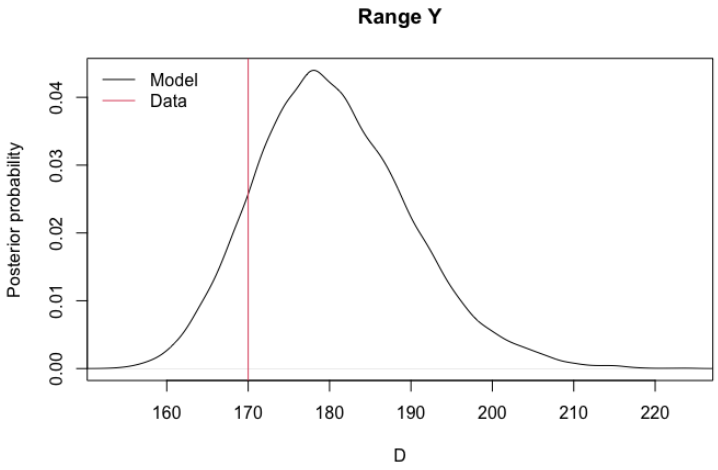

Figure 1: Posterior Predictive Check for Range of Data and Range of Model

three parameters that have inclusion probabilities of 1, β_9, β_{16} and β_{18}. β_9, the regression coefficient for CAPE3, represents convective available potential energy and has a 95% credible interval of (1.6,3.9). Therefore, as CAPE3 increases, so does VMAX. β_{16} is the regression coefficient for HWFI, which is a maximum one minute wind speed, and has a 95% credible interval of (12.7,16.1). Therefore, as HWFI increases, so does VMAX. β_{18}, the regression coefficient for HWRF, has a 95% credible interval of (9.02,12.8), so as HWRF increases, so does VMAX.

Problem 5: To make predictions for the missing VMAX values, I used the posterior predictive distribution and calculated the mean for each missing value. To estimate the error in these predictions, I used cross validation for the data that did not have missing values. I split the data into five groups for cross validation. The estimate for the mean absolute error of the predicted VMAX is 9.05 and the coverage of the 95% credible intervals is 100%.

Appendix: R Code

Problem 1

```
data <- read.csv("E2_data.csv",header=TRUE,stringsAsFactors = TRUE)

basin <-as.numeric(data$basin)

Y<-data$VMAX
X<-scale(data[,5:22])

library(rjags)

## remove observations with missing Vmax

junk        <- is.na(Y)
Yr           <- Y[!junk]
Xr           <- X[!junk,]
basin_r <- basin[!junk]
n <- length(Yr)
p <- ncol(Xr)

dat    <- list(Xr=Xr,Yr=Yr,n=n,p=p,basin_r=basin_r)



model <- "model{

   # Likelihood
    for(i in 1:n){
      Yr[i] ~ dnorm(alpha+inprod(Xr[i,],beta[])+theta[basin_r[i]],taue)
    }
   # Priors
    for(j in 1:p){
      beta[j] ~ dnorm(0,0.001)
    }

    for(j in 1:2){
    theta[j]~dnorm(0,tau)
    }

    alpha ~ dnorm(0,0.01)
    tau ~ dgamma(0.1,0.1)
    taue  ~ dgamma(0.1, 0.1)


  }"


library(rjags)
iters  <- 5000
chains <- 2
model  <- jags.model(textConnection(model),n.chains=chains,data = dat,quiet=TRUE)
```

```
update(model, 10000, progress.bar="none")

samp   <- coda.samples(model,
                       variable.names=c("beta","alpha","theta"),
                       n.iter=20000, progress.bar="none")

sum <- summary(samp)

gelman.diag(samp)
```

Problem 2

Cross Validation for Model 1

```
data <- read.csv("E2_data.csv",header=TRUE,stringsAsFactors = TRUE)

basin <-as.numeric(data$basin)

Y<-data$VMAX
X<-scale(data[,5:22])

library(rjags)

## remove observations with missing Vmax

junk        <- is.na(Y)
Yr           <- Y[!junk]
Xr           <- X[!junk,]

## Prepare Data for JAGS
basin_r <- basin[!junk]
n <- length(Yr)
p <- ncol(Xr)



## Break Data into Groups
fold <- rep(1:5,341)
fold <- sample(fold)
length(fold)

## Vectors to store statistics
Y_mean   <- c()
Y_median <- c()
Y_low    <- c()
Y_high   <- c()
error<- c()
abs_error <- c()

## Cross-validation
for(f in 1:5){

  # Select training data with fold not equal to f
  Yp<-Yr[fold==f]
```

```
  Xp<-Xr[fold==f,]
  basin_p<-basin_r[fold==f]
  data   <- list(Yr=Yr[fold!=f],Xr=Xr[fold!=f,],n=sum(fold!=f),basin_r=basin_r[fold!=f],p=p)



  # Fit model 1
  m1 <- textConnection("model{
     # Likelihood
    for(i in 1:n){
      Yr[i] ~ dnorm(alpha+inprod(Xr[i,],beta[])+theta[basin_r[i]],taue)
    }

   # Priors
    for(j in 1:p){
      beta[j] ~ dnorm(0,taue)
    }
    for(j in 1:2){
    theta[j]~dnorm(0,tau)
    }

    alpha ~ dnorm(0,0.01)
    tau ~ dgamma(0.1,0.1)
    taue  ~ dgamma(0.1, 0.1)
   }")

model1 <- jags.model(m1,data = data, n.chains=1,quiet=TRUE)
update(model1, 10000, progress.bar="none")
samp     <- coda.samples(model1,
                         variable.names=c("alpha","beta","theta","taue"),
                         thin=5, n.iter=20000, progress.bar="none")
b1       <- samp[[1]]

## Make Predictions
for (i in 1:length(Yr)){if(fold[i]==f){
Y_mod<- rnorm(nrow(b1),b1[,1]+Xr[i,1]*b1[,2]+Xr[i,2]*b1[,3]+Xr[i,3]*b1[,4]+
                Xr[i,4]*b1[,5]+
                Xr[i,5]*b1[,6]+Xr[i,6]*b1[,7]+Xr[i,7]*b1[,8]+
                Xr[i,8]*b1[,9]+Xr[i,9]*b1[,10]+
                Xr[i,10]*b1[,11]+Xr[i,10]*b1[,11]+Xr[i,11]*b1[,12]+
                Xr[i,12]*b1[,13]+Xr[i,13]*b1[,14]+Xr[i,14]*b1[,15]+
                Xr[i,15]*b1[,16]+Xr[i,16]*b1[,17]+Xr[i,17]*b1[,18]+Xr[i,18]*b1[,19]+
                b1[,20+basin_r[i]],1/b1[,20])
Y_mean[i]   <- mean(Y_mod)
Y_low[i]    <- quantile(Y_mod,0.025)
Y_high[i]   <- quantile(Y_mod,0.975)
error[i]<-abs(Y_mean[i]-Yr[i])
}}
rm(model1)
}

y     <- cbind(Yr,Yr) # Make data the same size/format as predictions
BIAS  <- colMeans(Y_mean-y)
```

```
MSE   <- colMeans((Y_mean-y)^2)
MAD   <- colMeans(abs(Y_mean-y))
COV   <- colMeans( (Y_low <= y) & (y <= Y_high))
```

Cross Validation for Model 2

```
data <- read.csv("E2_data.csv",header=TRUE,stringsAsFactors = TRUE)

basin <-as.numeric(data$basin)

Y<-data$VMAX
X<-scale(data[,5:22])

library(rjags)

## remove observations with missing Vmax

junk        <- is.na(Y)
Yr           <- Y[!junk]
Xr           <- X[!junk,]

## Prepare Data for JAGS
n <- length(Yr)
p <- ncol(Xr)



## Split data into groups
fold <- rep(1:5,341)
fold <- sample(fold)
length(fold)

## Vectors to store statistics
Y_mean   <- c()
Y_median <- c()
Y_low    <- c()
Y_high   <- c()
error<- c()
abs_error <- c()

for(f in 1:5){

  # Select training data with fold not equal to f
  Yp<-Yr[fold==f]
  Xp<-Xr[fold==f,]
  basin_p<-basin_r[fold==f]
  data   <- list(Yr=Yr[fold!=f],Xr=Xr[fold!=f,],n=sum(fold!=f),p=p)



  # Fit model 1
  m1 <- textConnection("model{
     # Likelihood
    for(i in 1:n){
```

```
      Yr[i] ~ dnorm(alpha+inprod(Xr[i,],beta[]),taue)
    }

   # Priors
    for(j in 1:p){
      beta[j] ~ dnorm(0,0.001)
    }

    alpha ~ dnorm(0,0.01)
    taue  ~ dgamma(0.1, 0.1)

   }")

  model1 <- jags.model(m1,data = data, n.chains=1,quiet=TRUE)
  update(model1, 10000, progress.bar="none")
  samp     <- coda.samples(model1,
                           variable.names=c("alpha","beta","taue"),
                           thin=5, n.iter=20000, progress.bar="none")
  b1       <- samp[[1]]

## Make Predictions
  for (i in 1:length(Yr)){if(fold[i]==f){
    Y_mod<- rnorm(nrow(b1),b1[,1]+Xr[i,1]*b1[,2]+Xr[i,2]*b1[,3]+Xr[i,3]*b1[,4]+Xr[i,4]*b1[,5]+
                    Xr[i,5]*b1[,6]+
                    Xr[i,6]*b1[,7]+Xr[i,7]*b1[,8]+Xr[i,8]*b1[,9]+
                    Xr[i,9]*b1[,10]+Xr[i,10]*b1[,11]+Xr[i,10]*b1[,11]+
                    Xr[i,11]*b1[,12]+
                    Xr[i,12]*b1[,13]+Xr[i,13]*b1[,14]+Xr[i,14]*b1[,15]+
                    Xr[i,15]*b1[,16]+Xr[i,16]*b1[,17]+Xr[i,17]*b1[,18]+
                    Xr[i,18]*b1[,19],
                  1/b1[,20])
    Y_mean[i]   <- mean(Y_mod)
    Y_low[i]    <- quantile(Y_mod,0.025)
    Y_high[i]   <- quantile(Y_mod,0.975)
    error[i]<-abs(Y_mean[i]-Yr[i])
  }}
  rm(model1)
}


y     <- cbind(Yr,Yr) # Make data the same size/format as predictions
BIAS  <- colMeans(Y_mean-y)
MSE   <- colMeans((Y_mean-y)^2)
MAD   <- colMeans(abs(Y_mean-y))
COV   <- colMeans( (Y_low <= y) & (y <= Y_high))
```

Problem 3

```
data <- read.csv("E2_data.csv",header=TRUE,stringsAsFactors = TRUE)

basin <-as.numeric(data$basin)

Y<-data$VMAX
```

```r
X<-scale(data[,5:22])

library(rjags)

## remove observations with missing Vmax

junk        <- is.na(Y)
Yr           <- Y[!junk]
Xr           <- X[!junk,]
basin_r <- basin[!junk]
n <- length(Yr)
p <- ncol(Xr)

dat    <- list(Xr=Xr,Yr=Yr,n=n,p=p,basin_r=basin_r)



model <- "model{

   # Likelihood
    for(i in 1:n){
      Yr[i] ~ dnorm(alpha+inprod(Xr[i,],beta[])+theta[basin_r[i]],taue)
    }
   # Priors
    for(j in 1:p){
      beta[j] ~ dnorm(0,0.001)
    }

    for(j in 1:2){
    theta[j]~dnorm(0,tau)
    }

    alpha ~ dnorm(0,0.01)
    tau ~ dgamma(0.1,0.1)
    taue  ~ dgamma(0.1, 0.1)


   # Posterior preditive checks
  for(i in 1:n){
   Y2[i] ~ dnorm(alpha+inprod(Xr[i,],beta[])+theta[basin_r[i]],taue)
  }
  D[1] <- max(Y2[])-min(Y2[])
  }"


library(rjags)
iters  <- 5000
chains <- 2
model  <- jags.model(textConnection(model),
                     n.chains=chains,data = dat,quiet=TRUE)

update(model, 10000, progress.bar="none")
```

```
samp   <- coda.samples(model,
                       variable.names=c("D","beta"),
                       n.iter=20000, progress.bar="none")

sum <- summary(samp)

D1  <- samp[[1]]


# Compute the test stats for the data
D0   <- c(max(Yr)-min(Yr))
Dnames <- c("Range Y")

# Compute the test stats for the models

pval1 <- 0
names(pval1)<-Dnames
plot(density(D1[,1]), xlim=range(c(D0[1],D1[,1])),
       xlab="D",ylab="Posterior probability",
       main=Dnames[1])
  abline(v=D0[1],col=2)
  legend("topleft",c("Model","Data"),lty=1,col=1:2,bty="n")
  pval1[1] <- mean(D1[,1]>D0[1])
```

Problem 4

```
data <- read.csv("E2_data.csv",header=TRUE,stringsAsFactors = TRUE)

basin <-as.numeric(data$basin)

Y<-data$VMAX
X<-scale(data[,5:22])

library(rjags)

## remove observations with missing Vmax

junk        <- is.na(Y)
Yr           <- Y[!junk]
Xr           <- X[!junk,]
basin_r <- basin[!junk]
n <- length(Yr)
p <- ncol(Xr)

dat    <- list(Xr=Xr,Yr=Yr,n=n,p=p,basin_r=basin_r)


model <- "model{

   # Likelihood
    for(i in 1:n){
      Yr[i] ~ dnorm(alpha+inprod(Xr[i,],beta[])+
      theta[basin_r[i]],taue)
```

```
    }
   # Priors
    for(j in 1:p){
        beta[j] <- gamma[j]*delta[j]
        gamma[j] ~ dbern(0.5)
        delta[j] ~ dnorm(0,tau)

    }
    for(j in 1:2){
    theta[j]~dnorm(0,tau)
    }
    alpha ~ dnorm(0,0.01)
    tau ~ dgamma(0.1,0.1)
    taue  ~ dgamma(0.1, 0.1)

}"



library(rjags)
iters  <- 5000
chains <- 2
model  <- jags.model(textConnection(model),n.chains=chains,data = dat,quiet=TRUE)

update(model, 10000, progress.bar="none")

samp   <- coda.samples(model,
                       variable.names=c("beta"),
                       n.iter=20000, progress.bar="none")

sum <- summary(samp)

beta    <- NULL
for(l in 1:chains){
  beta <- rbind(beta,samp[[l]])
}

for(j in 1:p){
  hist(beta[,j],xlab=expression(beta[j]),ylab="Posterior density",
       breaks=100)
}

Inc_Prob <- apply(beta!=0,2,mean)
Q        <- t(apply(beta,2,quantile,c(0.5,0.05,0.95)))
out      <- cbind(Inc_Prob,Q)
```

Problem 5

(code for cross-validation is under Problem 3)

```
data <- read.csv("E2_data.csv",header=TRUE,stringsAsFactors = TRUE)

basin <-as.numeric(data$basin)
```

```
Y<-data$VMAX
X<-scale(data[,5:22])

nt<-length(Y)

library(rjags)

## remove observations with missing Vmax

junk        <- is.na(Y)
Yr           <- Y[!junk]
Xr           <- X[!junk,]
basin_r <- basin[!junk]
Xp <-X[junk,]
Yp <-Y[junk]
basin_p<-basin[junk]
n <- length(Yr)
p <- ncol(Xr)
np <- length(Yp)

dat    <- list(Xr=Xr,Yr=Yr,n=n,p=p,np=np,Xp=Xp,basin_r=basin_r,basin_p=basin_p)



model <- "model{

   # Likelihood
    for(i in 1:n){
      Yr[i] ~ dnorm(alpha+inprod(Xr[i,],beta[])+
      theta[basin_r[i]],taue)
    }
   # Priors
    for(j in 1:p){
      beta[j] ~ dnorm(0,0.001)
    }
    for(j in 1:2){
    theta[j]~dnorm(0,tau)
    }
    alpha ~ dnorm(0,0.01)
    tau ~ dgamma(0.1,0.1)
    taue  ~ dgamma(0.1, 0.1)

  # Prediction

  for(i in 1:np){
  Yp[i] ~ dnorm(alpha+inprod(Xp[i,],beta[])+
  theta[basin_p[i]],taue)

  }
}"


library(rjags)
```

```
iters  <- 5000
chains <- 2
model  <- jags.model(textConnection(model),n.chains=chains,data = dat,quiet=TRUE)

update(model, 10000, progress.bar="none")

samp   <- coda.samples(model,
                       variable.names=c("Yp","beta"),
                       n.iter=20000, progress.bar="none")

sum <- summary(samp)

D1  <- samp[[1]]
Yp.samps    <- D1[,1:np]
Y.predict <-colMeans(Yp.samps)
low_v   <- apply(Yp.samps,2,quantile,0.025)
high_v  <- apply(Yp.samps,2,quantile,0.975)
L<-rep(NA,nt)
U<-rep(NA,nt)

j<-1

for(i in 1:nt){if (junk[i]){

Y[i]<-Y.predict[j]
L[i]<-low_v[j]
U[i]<-high_v[j]
j<-j+1
}
}

predict<-cbind(data$HWRF,Y,L,U)
write.csv(predict,"EbbyLouisa.csv")
```