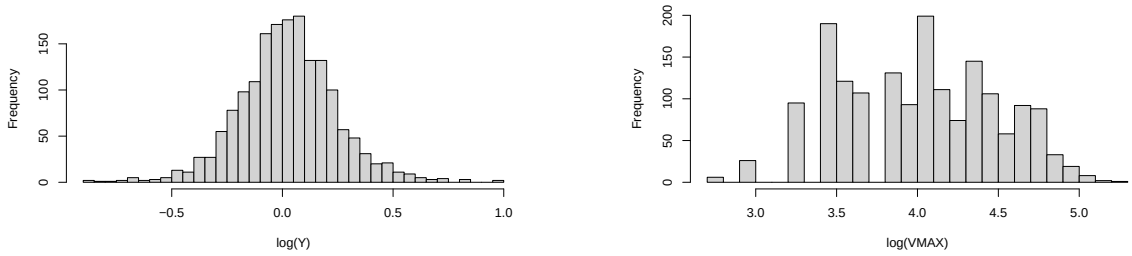
1. Let Y_i be the ratio between (`VMAX`) and (`HWRF`), *i.e.*, $Y_i = \text{VMAX}_i/\text{HWRF}_i$. This variable represents a correction factor for the mathematical model and could be used to improve the prediction of `HWRF`. The idea is to used the other state variables as indicators of the a possible underestimation ($Y_i > 1$) or overestimation ($Y_i < 1$) of the mathematical model. This approach is preferred because the value of `HWRF` is expected to be relatively close to `VMAX`. Moreover, the histogram of $\log(Y_i)$ (Fig. 1a) more closely resembles a normal distribution (bell-shape) than that of log(VMAX) (Fig. 1b).



(a) (b)

Figure 1. a) Histogram of the observed $\log(Y)$ and, b) log(VMAX)

Y_i was regressed onto 18 potential covariates (X_j): `LAT`, `LON`, `MINSLP`, `SHR_MAG`, `STM_SPD`, `SST`, `RHLO`, `CAPE1`, `CAPE3`, `SHTFL2`, `TCOND7002`, `INST2`, `CP1`, `TCONDSYM2`, `COUPSYM3`, `HWFI`, `VMAX_OP_TO`, and `dum_basin`. The variables `lead_time` and `Date` were excluded from the regression because they did not provide relevant information (`lead_time` had the same value for all the observations) or were considered as a potential source of collinearity. The following three models were compared:

- **Model 1 - Linear regression**. $\log(Y_i) = \alpha + \sum_{j=1}^{p} \beta_j X_{ij} + \varepsilon_i$, were $\varepsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$.
- **Model 2 - Linear Mixed Model**. $\log(Y_i) = \alpha + \sum_{j=1}^{p} \beta_j X_{ij} + \theta_{\text{storm}_k} + \varepsilon_{ij}$, were $\theta_{\text{storm}_k} \sim \mathcal{N}(0, \tau^2)$ represents the random effect for storm k, and $\varepsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$.
- **Model 3 - Linear mixed model with second-order polynomial**. $\log(Y_i) = \alpha + \sum_{j=1}^{p} \beta_j X_{ij} + \gamma X_{i,16}^2 + \theta_{\text{storm}_k} + \varepsilon_{ij}$, were $X_{i,16}$ is the covariate associated with `HWFI`, $\theta_{\text{storm}_k} \sim \mathcal{N}(0, \tau^2)$ and $\varepsilon \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$.

The variable `StormID` was considered as a possible random effect in the second and third models since the predictions for the same storm are expected to be clustered to some extent. A second-order polynomial was proposed in the third model to account for possible misfitting (see section 3) to the data. For the three models, all the variables were centered and scaled. Uninformative priors $\alpha, \beta_j, \gamma \sim \mathcal{N}(0, 1000)$ and $\sigma^2, \tau^2 \sim \text{InvGamma}(0.1, 0.1)$ were selected due to the lack of previous information and to facilitate convergence (conjugate priors).

2. The three models were compared using DIC and WAIC. To monitor convergence, the effective sample size and the Gelman-Rubin static were computed. For the three models, two chains of length 50000 (after 10000 burn in) iterations were simulated. In all cases, the effective sample size were larger than 1000 and the Gelman-Rubin statistic was smaller than 1.01. Thus, it is concluded that the chains converged and mixed sufficiently. Table 1 summaries the information criteria of each model. Model 2 has the smallest DIC and WAIC,

however its difference with Model 3 is not substantial (less than 10). Considering that Model 2 is simpler, Model 2 was used for the remainder of the analysis.

Table 1. Information criteria for each model

Model	DIC			WAIC	
	Mean deviance	Penalty	DIC	p_W	WAIC
Model 1	-518.1	20.05	-498.1	20.83	-496.94
Model 2	-753.6	88.6	-665.1	96.21	-650.46
Model 3	-753.2	89.44	-663.7	97.07	-648.87

3. To verify that the models fitted the data, posterior predictive checks using the minimum, maximum, range and standard deviation were computed. Table 2 summaries the Bayesian p-values for Models 2 and 3. Originally, only Models 1 and 2 were analysed. However, since two of the four Bayesian p-values of Model 2 were near zero, it was concluded that the model does not fit the data sufficiently well. Thus, Model 3 was proposed as an option to improve the goodness-of-fit. However, as seen in Table 2, the p-values for this model are also near zero. To address this issue, an option might be fitting another model with higher-order polynomial or using a second-order polynomial with other important predictors.

Table 2. Bayesian p-values for different test statistic

Model	Max(Y)	Min(Y)	Range(Y)	SD(Y)
Model 2	0.029	0.882	0.020	0735
Model 3	0.030	0.890	0.030	0.738

4. Table 3 summaries the quantiles of the regression coefficient of each of the 18 covariates considered in Model 2. Since the covariates were standarized prior to the regression analysis, it is possible to use their coefficients to determine the importance of each predictor. The covariates with the greatest coefficients (in absolute value) are `HWFI`, `COUPSYM3`, and `SHR_MAG`. The coefficients represent the change in the mean of Y due to an increase of one standard deviation unit of the predictor, all other factors equal.

Table 3. Quantiles of the regression coefficients in Model 2

X_j	2.5%	50%	97.5%	X_j	2.5%	50%	97.5%
LAT	-0.045	-0.017	0.011	LONG	-0.024	0.017	0.059
MINSLP	-0.054	0.012	0.076	SHR_MAG	-0.055	-0.039	-0.024
STM_SPD	-0.027	0.006	0.038	SST	-0.021	-0.003	0.015
RHLO	0.018	0.034	0.051	CAPE1	-0.062	-0.037	-0.011
CAPE3	-0.002	0.022	0.046	SHTFL2	-0.044	-0.007	0.032
TCOND7002	-0.122	-0.056	0.01	INST2	-0.027	0.009	0.045
CP1	-0.01	0.007	0.025	TCONDSYM2	0.001	0.024	0.047
COUPSYM3	0.027	0.054	0.081	HWFI	0.08	0.111	0.143
VMAX_OP_TO	-0.043	-0.017	0.009	dum_basin	-0.06	-0.013	0.035

5. To performed the Cross-Validation, the data was split in five equally-sized groups. Predictions were made by setting aside one group at a time and fitting the model with the observations of the remaining groups. For Model 2, a mean absolute error of 5.939 and a coverage of 0.999 was estimated .

Annotated code

Model 1

```
library (rjags)
set.seed(2120)

#### Load data ####
db <- read.csv("E2_data.csv")
summary(db)

# Creating dummy variable for Basin
db$dum_basin <- ifelse(db$basin == "atlantic",1,0)


####  Check for NA and define variables #####
junk <- is.na(db$VMAX)
data <- db[!junk,-grep(c("Date|^basin$|lead_time"),names(db))]

X <- data[,-grep(c("StormID|^VMAX$|HWRF"),names(data))]
Y <- log(data$VMAX/data$HWRF)

n <- length(Y)
p <- ncol(X)

# Scale the variables
Xs <- scale(X)

#### Model comparison using DIC/WAIC ####

# Model 1 - Linear regression
m1 <- textConnection("model{

 # Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(mu[i],taue)
    mu[i] <- alpha + inprod(X[i,],beta[])
 }

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

```
 taue ~ dgamma(0.1,0.1)

 # WAIC calculations
 for(i in 1:n){
    like[i] <- dnorm(Y[i],mu[i],taue)
 }
}")

# JAGS
data <- list(Y=Y,X=Xs,n=n,p=p)
burn     <- 10000
n.iter   <- 50000
n.chains <- 2
params <- c("alpha","beta")

model1 <- jags.model(m1, data = data, n.chains=n.chains,quiet=TRUE)
update(model1, burn, progress.bar="none")

samples1 <- coda.samples(model1, variable.names=params,
                     n.iter=n.iter, progress.bar="none")

# Check convergence for Model 1
ESS1 <- round(effectiveSize(samples1),1)
out1 <- summary(samples1)$quantiles

gel1<-gelman.diag(samples3)
max(gel1$psrf[,1])

# Compute DIC
DIC3 <- dic.samples(model1, n.iter=n.iter,progress.bar="none")

# Compute WAIC
waic1   <- coda.samples(model1,
                    variable.names=c("like"),
                    n.iter=n.iter, progress.bar="none")
like1 <- rbind(waic1[[1]],waic1[[2]])
fbar1 <- colMeans(like1)
Pw1     <- sum(apply(log(like1),2,var))
WAIC1   <- -2*sum(log(fbar1))+2*Pw1

DIC1
WAIC1;Pw1

#### Posterior Predictive Checks ####
```

```
model_string <- textConnection("model{

 # Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(mu[i],taue)
    mu[i] <- alpha + inprod(X[i,],beta[])
 }

 # Priors
 for(j in 1:p){
    beta[j]~dnorm(0,0.001)
 }
 alpha ~ dnorm(0,0.001)
 taue ~ dgamma(0.1,0.1)

# Posterior preditive checks
for(i in 1:n){
  Y2[i] ~ dnorm(mu[i],taue)
}

D[1] <- max(Y2[])
D[2] <- min(Y2[])
D[3] <- D[1] - D[2]
D[4] <- sd(Y2[])
}")

# JAGS
data <- list(Y=Y,X=Xs,n=n,p=p)
burn     <- 10000
n.iter   <- 50000
n.chains <- 2
params <- c("D")

model <- jags.model(model_string, data = data, n.chains=n.chains,quiet=TRUE)
update(model, burn, progress.bar="none")

samples <- coda.samples(model, variable.names=params,
                    n.iter=n.iter, progress.bar="none")

D <- rbind(samples[[1]],samples[[2]])
D0 <- c(max(Y), min(Y), max(Y)-min(Y), sd(Y))

Dnames <- c("Max", "Min", "Range", "SD")
pval <- rep(0,4)
```

```
names(pval)<-Dnames

for(j in 1:4){
  plot(density(D[,j]), xlim=range(c(D0[j],D[,j])),
   main='' ,xlab=Dnames[j],ylab="Posterior probability")
  abline(v=D0[j],col=2)
  legend("topright",c("Model","Data"),lty=1,col=1:2,bty="n")

  pval[j] <- mean(D[,j]>D0[j])
}
```

Model 2

```
library (rjags)
set.seed(2120)

#### Load data ####
db <- read.csv("E2_data.csv")
summary(db)

# Creating dummy variable for Basin
db$dum_basin <- ifelse(db$basin == "atlantic",1,0)


####  Check for NA and define variables #####
junk <- is.na(db$VMAX)
dat <- db[!junk,-grep(c("Date|^basin$|lead_time"),names(db))]

X <- dat[,-grep(c("StormID|^VMAX$|HWRF"),names(dat))]
Y <- log(dat$VMAX/dat$HWRF)
storms <- as.factor(dat$StormID)
storm <- as.numeric(storms)
n_storm <- max(storm)

n <- length(Y)
p <- ncol(X)

# Scale the variables
Xs <- scale(X)

#### Model comparison using DIC/WAIC ####

# Model 2 - Random linear model
```

```
m2 <- textConnection("model{

 # Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(mu[i],taue)
    mu[i] <- alpha + inprod(X[i,],beta[])+theta[storm[i]]
 }

 # Priors
 for(j in 1:p){
    beta[j]~dnorm(0,0.001)
 }
 for(j in 1:ns){
    theta[j]~dnorm(0,tau)
 }
 alpha ~ dnorm(0,0.001)
 taue ~ dgamma(0.1,0.1)
 tau ~ dgamma(0.1,0.1)

 # WAIC calculations
 for(i in 1:n){
    like[i] <- dnorm(Y[i],mu[i],taue)
 }
}")

# JAGS
data <- list(Y=Y,X=Xs,n=n,storm=storm,ns=n_storm,p=p)
burn     <- 10000
n.iter   <- 50000
n.chains <- 2
params <- c("alpha","beta","theta")

model2 <- jags.model(m2, data = data, n.chains=n.chains,quiet=TRUE)
update(model2, burn, progress.bar="none")

samples2 <- coda.samples(model2, variable.names=params,
                     n.iter=n.iter, progress.bar="none")

# Check convergence for Model 2
ESS2 <- round(effectiveSize(samples2),1)
out2 <- summary(samples2)$quantiles
gel2 <-gelman.diag(samples2)
max(gel2$psrf[,1])
```

```
# Compute DIC
DIC2 <- dic.samples(model2, n.iter=n.iter,progress.bar="none")

# Compute WAIC
waic2   <- coda.samples(model2,
                    variable.names=c("like"),
                    n.iter=n.iter, progress.bar="none")
like2 <- rbind(waic2[[1]],waic2[[2]])
fbar2 <- colMeans(like2)
Pw2   <- sum(apply(log(like2),2,var))
WAIC2 <- -2*sum(log(fbar2))+2*Pw2

DIC2
WAIC2;Pw2

#### Posterior Predictive Checks ####

model_string <- textConnection("model{

 # Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(mu[i],taue)
    mu[i] <- alpha + inprod(X[i,],beta[])+theta[storm[i]]
 }

 # Priors
 for(j in 1:p){
    beta[j]~dnorm(0,0.001)
 }
 for(j in 1:ns){
    theta[j]~dnorm(0,tau)
 }
 alpha ~ dnorm(0,0.001)
 taue ~ dgamma(0.1,0.1)
 tau ~ dgamma(0.1,0.1)

# Posterior preditive checks
for(i in 1:n){
  Y2[i] ~ dnorm(mu[i],taue)
}

D[1] <- max(Y2[])
D[2] <- min(Y2[])
D[3] <- D[1] - D[2]
```

```
D[4] <- sd(Y2[])
}")

# JAGS
data <- list(Y=Y,X=Xs,n=n,storm=storm,ns=n_storm,p=p)
burn     <- 10000
n.iter   <- 50000
n.chains <- 2
params <- c("D")

model <- jags.model(model_string, data = data, n.chains=n.chains,quiet=TRUE)
update(model, burn, progress.bar="none")

samples <- coda.samples(model, variable.names=params,
                    n.iter=n.iter, progress.bar="none")

D <- rbind(samples[[1]],samples[[2]])
D0 <- c(max(Y), min(Y), max(Y)-min(Y), sd(Y))

Dnames <- c("Max", "Min", "Range", "SD")
pval <- rep(0,4)
names(pval)<-Dnames

for(j in 1:4){
  plot(density(D[,j]), xlim=range(c(D0[j],D[,j])),
   main='' ,xlab=Dnames[j],ylab="Posterior probability")
  abline(v=D0[j],col=2)
  legend("topright",c("Model","Data"),lty=1,col=1:2,bty="n")

  pval[j] <- mean(D[,j]>D0[j])
}
```

Model 3

```
library (rjags)
set.seed(2120)

#### Load data ####
db <- read.csv("E2_data.csv")
summary(db)

# Creating dummy variable for Basin
db$dum_basin <- ifelse(db$basin == "atlantic",1,0)
```

```
####  Check for NA and define variables #####
junk <- is.na(db$VMAX)
dat <- db[!junk,-grep(c("Date|^basin$|lead_time"),names(db))]

X <- dat[,-grep(c("StormID|^VMAX$|HWRF"),names(dat))]
Y <- log(dat$VMAX/dat$HWRF)
storms <- as.factor(dat$StormID)
storm <- as.numeric(storms)
n_storm <- max(storm)

n <- length(Y)
p <- ncol(X)

# Scale the variables
Xs <- scale(X)

#### Model comparison using DIC/WAIC ####

# Model 3 - Linear Mixed Model with quadratic term
m3 <- textConnection("model{

 # Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(mu[i],taue)
    mu[i] <- alpha + inprod(X[i,],beta[])+theta[storm[i]] + gamma*X[i,16]*X[i,16]
 }

 # Priors
 for(j in 1:p){
    beta[j]~dnorm(0,0.001)
 }
 for(j in 1:ns){
    theta[j]~dnorm(0,tau)
 }
 alpha ~ dnorm(0,0.001)
 gamma ~ dnorm(0,0.001)
 taue ~ dgamma(0.1,0.1)
 tau ~ dgamma(0.1,0.1)

 # WAIC calculations
 for(i in 1:n){
    like[i] <- dnorm(Y[i],mu[i],taue)
 }
```

```
}")

# JAGS
data <- list(Y=Y,X=Xs,n=n,storm=storm,ns=n_storm,p=p)
burn     <- 10000
n.iter   <- 50000
n.chains <- 2
params <- c("alpha","beta","gamma","theta")

model3 <- jags.model(m3, data = data, n.chains=n.chains,quiet=TRUE)
update(model3, burn, progress.bar="none")

samples3 <- coda.samples(model3, variable.names=params,
                     n.iter=n.iter, progress.bar="none")

# Check convergence for Model 3
ESS3 <- round(effectiveSize(samples3),1)
out3 <- summary(samples3)$quantiles
gel3 <-gelman.diag(samples3)
max(gel3$psrf[,1])

# Compute DIC
DIC3 <- dic.samples(model3, n.iter=n.iter,progress.bar="none")

# Compute WAIC
waic3   <- coda.samples(model3,
                    variable.names=c("like"),
                    n.iter=n.iter, progress.bar="none")
like3 <- rbind(waic3[[1]],waic3[[2]])
fbar3 <- colMeans(like3)
Pw3   <- sum(apply(log(like3),2,var))
WAIC3 <- -2*sum(log(fbar3))+2*Pw3

DIC3
WAIC3;Pw3

#### Posterior Predictive Checks ####

model_string <- textConnection("model{

 # Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(mu[i],taue)
    mu[i] <- alpha + inprod(X[i,],beta[])+theta[storm[i]]+ gamma*X[i,16]*X[i,16]
```

```
 }

 # Priors
 for(j in 1:p){
    beta[j]~dnorm(0,0.001)
 }
 for(j in 1:ns){
    theta[j]~dnorm(0,tau)
 }
 alpha ~ dnorm(0,0.001)
 gamma ~ dnorm(0,0.001)
 taue ~ dgamma(0.1,0.1)
 tau ~ dgamma(0.1,0.1)

# Posterior preditive checks
for(i in 1:n){
  Y2[i] ~ dnorm(mu[i],taue)
}

D[1] <- max(Y2[])
D[2] <- min(Y2[])
D[3] <- D[1] - D[2]
D[4] <- sd(Y2[])
}")

# JAGS
data <- list(Y=Y,X=Xs,n=n,storm=storm,ns=n_storm,p=p)
burn     <- 10000
n.iter   <- 50000
n.chains <- 2
params <- c("D")

model <- jags.model(model_string, data = data, n.chains=n.chains,quiet=TRUE)
update(model, burn, progress.bar="none")

samples <- coda.samples(model, variable.names=params,
                    n.iter=n.iter, progress.bar="none")

D <- rbind(samples[[1]],samples[[2]])
D0 <- c(max(Y), min(Y), max(Y)-min(Y), sd(Y))

Dnames <- c("Max", "Min", "Range", "SD")
pval <- rep(0,4)
names(pval)<-Dnames
```

```
for(j in 1:4){
  plot(density(D[,j]), xlim=range(c(D0[j],D[,j])),
   main='' ,xlab=Dnames[j],ylab="Posterior probability")
  abline(v=D0[j],col=2)
  legend("topright",c("Model","Data"),lty=1,col=1:2,bty="n")

  pval[j] <- mean(D[,j]>D0[j])
}
```

Predictions

```
library (rjags)
set.seed(2120)

#### Load data ####
db <- read.csv("E2_data.csv")
summary(db)

# Creating dummy variable for Basin
db$dum_basin <- ifelse(db$basin == "atlantic",1,0)

####  Check for NA and define variables #####
test <- is.na(db$VMAX)
dat <- db[,-grep(c("Date|^basin$|lead_time"),names(db))]

X <- dat[,-grep(c("StormID|^VMAX$|HWRF"),names(dat))]
X <- scale(X)     # scale predictors

storms <- as.factor(dat$StormID)
storm <- as.numeric(storms)
n_storm <- max(storm)

# Observed data
Yo <- log(dat$VMAX[!test]/dat$HWRF[!test])
Xo <- X[!test,]
stormo <- storm[!test]

# Prediction data
Xp <- X[test,]
stormp <- storm[test]

# General
no <- length(Yo)
```

```
np <- nrow(Xp)
p <- ncol(Xo)

# Model - Random effects with log transformation
m4 <- textConnection("model{

 # Likelihood
 for(i in 1:no){
    Yo[i] ~ dnorm(muo[i],taue)
    muo[i] <- alpha + inprod(Xo[i,],beta[])+theta[stormo[i]]
 }

 # Prediction
 for(i in 1:np){
    Yp[i] ~ dnorm(mup[i],taue)
    mup[i] <- alpha + inprod(Xp[i,],beta[])+theta[stormp[i]]
 }

 # Priors
 for(j in 1:p){
    beta[j]~dnorm(0,0.001)
 }
 for(j in 1:ns){
    theta[j]~dnorm(0,tau)
 }
 alpha ~ dnorm(0,0.001)
 taue ~ dgamma(0.1,0.1)
 tau ~ dgamma(0.1,0.1)
}")

# JAGS
data <- list(Yo=Yo,Xo=Xo,no=no,stormo=stormo,ns=n_storm,p=p,
          Xp=Xp,np=np,stormp=stormp)
burn     <- 10000
n.iter   <- 50000
n.chains <- 1
params <- c("alpha","beta","theta","Yp")

model_pred <- jags.model(m4, data = data, n.chains=n.chains,quiet=TRUE)
update(model_pred, burn, progress.bar="none")
samples_pred <- coda.samples(model_pred, variable.names=params,
                    n.iter=n.iter, progress.bar="none")

# Extract samples for predictions
```

```
samps       <- samples_pred[[1]]
Yp.samps    <- samps[,1:np]

# PPD 95% intervals and mean
mu_pred <- apply(Yp.samps,2,mean)
low_pred   <- apply(Yp.samps,2,quantile,0.025)
high_pred  <- apply(Yp.samps,2,quantile,0.975)

# Exporting results
res <- matrix(rep(NA,nrow(db)*4),nrow=nrow(db),ncol=4)
res[,1] <- db$HWRF
res[,2] <- db$VMAX
j <-1
for(i in 1:nrow(db)){
  if(test[i]==TRUE){
res[i,2] <- res[i,1]*exp(mu_pred[j])
res[i,3] <- res[i,1]*exp(low_pred[j])
res[i,4] <- res[i,1]*exp(high_pred[j])
j <- j+1
  }
}

write.csv(res,"BarbaDavid.csv")
```

Cross-Validation

```
library (rjags)
set.seed(2120)

#### Load data ####
db <- read.csv("E2_data.csv")

# Creating dummy variable for Basin
db$dum_basin <- ifelse(db$basin == "atlantic",1,0)

####  Check for NA and define variables #####
junk <- is.na(db$VMAX)
dat <- db[!junk,-grep(c("Date|^basin$|lead_time"),names(db))]

X <- dat[,-grep(c("StormID|^VMAX$|HWRF"),names(dat))]
X <- scale(X)     # scale predictors
Y <- log(dat$VMAX/dat$HWRF)
storms <- as.factor(db$StormID)
```

```
storm <- as.numeric(storms)
n_storm <- max(storm)

n <- length(Y)
p <- ncol(X)

### Assign observations to K=5 folds ###

set.seed(0820)
fold <- rep(1:5,n/5)
fold <- sample(fold)
fold

### Fitting model and making predictions ###

VMAX_mean <- matrix(NA,n,1)
VMAX_low <- matrix(NA,n,1)
VMAX_high <- matrix(NA,n,1)

for(f in 1:5){

  # Select the training data with fold not equal to f
  data <- list(Y=Y[fold!=f], X=X[fold!=f,],n=sum(fold!=f),
               storm=storm[fold!=f], ns=n_storm,p=p)
  params <- c("alpha","beta","sigmae","theta")

  # Fit model
  model_string <- textConnection("model{

 # Likelihood
 for(i in 1:n){
    Y[i] ~ dnorm(mu[i],taue)
    mu[i] <- alpha + inprod(X[i,],beta[])+theta[storm[i]]
 }

 # Priors
 for(j in 1:p){
    beta[j]~dnorm(0,0.001)
 }
 for(j in 1:ns){
    theta[j]~dnorm(0,tau)
 }
 alpha ~ dnorm(0,0.001)
 taue ~ dgamma(0.1,0.1)
```

```
 tau ~ dgamma(0.1,0.1)

 sigmae <- 1/sqrt(taue)
  }")
  model <- jags.model(model_string,data = data, n.chains=1,quiet=TRUE)
  update(model, 10000, progress.bar="none")
  samples <- coda.samples(model, variable.names=params, n.iter=30000,
                          progress.bar="none")

  # Extract the samples
  samps <- samples[[1]]
  alpha <- samps[,1]
  bet <- samps[,1+1:p]
  sigmae <- samps[,p+2]
  theta <- samps[,(p+3):ncol(samps)]

  # Predictions
  for(i in 1:n){if(fold[i]==f){
mu_mod <- alpha + bet[] %*% X[i,]+theta[,storm[i]]
Y_mod <- rnorm(nrow(bet),mu_mod,sigmae)
VMAX_mod <- dat$VMAX[i]*exp(Y_mod)
VMAX_mean[i,1] <- mean(VMAX_mod)
VMAX_low[i,1] <- quantile(VMAX_mod,0.025)
VMAX_high[i,1] <- quantile(VMAX_mod,0.975)
}}

  rm(model)
}

## Results summary
MAE <- colMeans(abs(dat$VMAX-VMAX_mean))
COV <- colMeans( (VMAX_low <= dat$VMAX) & (dat$VMAX <= VMAX_high))
```