ex\_lineaRegression\_postcheck

In the original Mauna Loa CO data analysis we visualized the posterior predictive distribution of the expected CO with respect to the month and compared it to the observed data points. This can be seen as one method for visual posterior predictive check. However, let’s continue model checking a bit more and then improve the model based on our findings.

Conduct posterior predictive check for the Mauna Loa CO data in similar manner as in the Speed of Light example in BDA3. Sample 20 replicates of the and do the following:

To sample a replicate data set you must sample values from , where and are drawn from the posterior. For example, pick 20 random triplets of from the Markov chain and for each of them sample . Then plot histogram of each .\

Next, revise the model so that . Find the posterior of the parameters of the new model and do the following:

Note! Since you are not asked about the parameter inference, you don’t need to worry about how to scale back to even if you standardize your and .

The danger with sequantial model refinements is that we conduct it so long that our model overfits the data. Hence,

Conduct the posterior predictive comparison using the point-wise log predictive density

and the root mean squared error

where are the training and test data and are the test data points. Which of the models has better posterior predictive performance. Based on this results, does it seem that model M2 has overfitted the data?

Note, where is a sample from the posterior distribution of the parameters ( in the original model), that is .

Each of the above four tasks provides 5 points from correct implementation and answer. Each task gives 2 points if it is done towards right direction and partially correct.

Load the needed libraries.

library(ggplot2)  
library(StanHeaders)  
library(rstan)

##   
## rstan version 2.26.13 (Stan version 2.26.1)

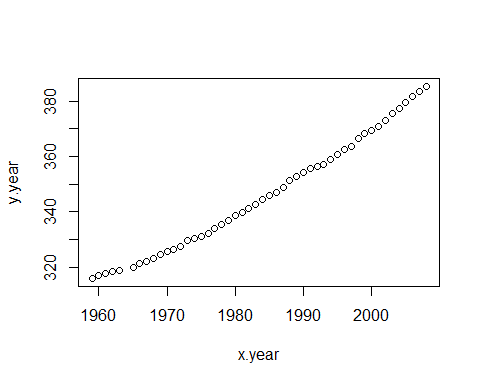
## For execution on a local, multicore CPU with excess RAM we recommend calling  
## options(mc.cores = parallel::detectCores()).  
## To avoid recompilation of unchanged Stan programs, we recommend calling  
## rstan\_options(auto\_write = TRUE)  
## For within-chain threading using `reduce\_sum()` or `map\_rect()` Stan functions,  
## change `threads\_per\_chain` option:  
## rstan\_options(threads\_per\_chain = 1)

## Do not specify '-march=native' in 'LOCAL\_CPPFLAGS' or a Makevars file

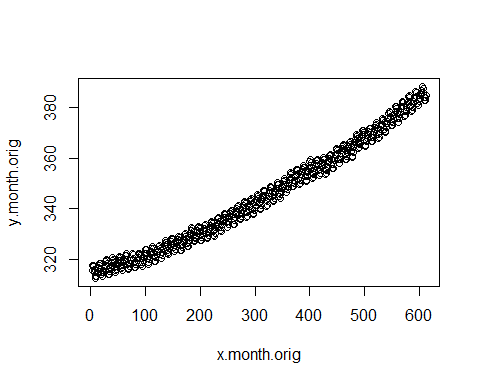
set.seed(123)  
  
options(mc.cores = parallel::detectCores())  
rstan\_options(auto\_write = TRUE)

Load the data and explore its properties

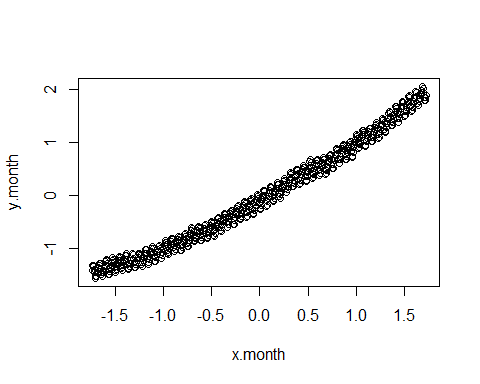
# Load the data and explore it visually  
maunaloa.dat = read.table("maunaloa\_data.txt", header=FALSE, sep="\t")  
# The columns are   
# Year January February ... December Annual average  
  
# Notice! values -99.99 denote NA  
  
# Let's take the yearly averages and plot them  
x.year = as.vector(t(maunaloa.dat[,1]))  
y.year = as.vector(t(maunaloa.dat[,14]))  
# remove NA rows  
x.year = x.year[y.year>0]  
y.year = y.year[y.year>0]  
plot(x.year,y.year)



# Let's take the monthy values and construct a "running month" vector  
y.month.orig = as.vector(t(maunaloa.dat[,2:13]))  
x.month.orig = as.vector(seq(1,length(y.month.orig),1))  
  
# remove NA rows  
x.month.orig = x.month.orig[y.month.orig>0]  
y.month.orig = y.month.orig[y.month.orig>0]  
plot(x.month.orig,y.month.orig)



# standardize y and x  
my = mean(y.month.orig)  
stdy = sd(y.month.orig)  
y.month = (y.month.orig-my)/stdy  
  
mx = mean(x.month.orig)  
stdx = sd(x.month.orig)  
x.month = (x.month.orig-mx)/stdx  
  
plot(x.month,y.month)



# data list  
data <- list (N=length(x.month), y=y.month, x=x.month)

## Posterior predictive check

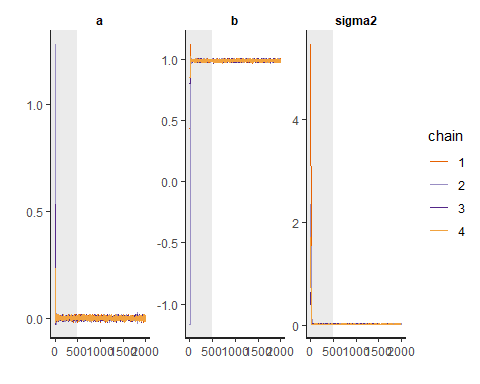
Analysis with the original model

mauna\_loa\_c02\_model = "  
data{  
 int<lower=0> N; // number of observations   
 real y[N]; // observed CO2 values   
 real x[N]; // observed times   
}  
parameters{  
 real a;  
 real b;  
 real<lower=0> sigma2;   
}  
transformed parameters{  
 real<lower=0> sigma;  
 real mu[N];  
   
 sigma=sqrt(sigma2);  
   
 for( i in 1 : N ) {  
 mu[i] = a + b \* x[i];  
 }  
}  
model{  
 a ~ normal( 0, sqrt(1e6));  
 b ~ normal( 0, sqrt(1e6));  
 sigma2 ~ inv\_gamma(0.001,0.001);  
   
 for( i in 1 : N ) {  
 y[i] ~ normal(mu[i],sigma);  
 }  
}"

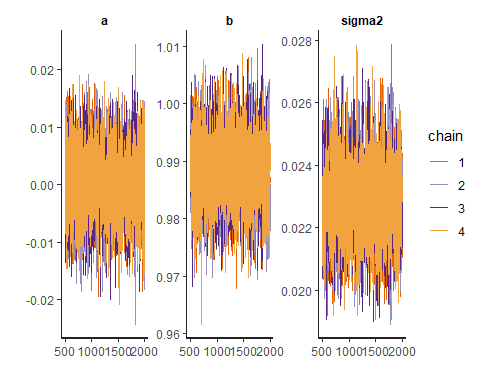
set.seed(123)  
post=stan(model\_code=mauna\_loa\_c02\_model,data=data,warmup=500,iter=2000,chains=4,thin=1,control = list(adapt\_delta = 0.8,max\_treedepth = 10))  
  
# Check for convergence, see PSRF (Rhat in Stan)  
print(post,pars=c("a","b","sigma2"))

## Inference for Stan model: anon\_model.  
## 4 chains, each with iter=2000; warmup=500; thin=1;   
## post-warmup draws per chain=1500, total post-warmup draws=6000.  
##   
## mean se\_mean sd 2.5% 25% 50% 75% 97.5% n\_eff Rhat  
## a 0.00 0 0.01 -0.01 0.00 0.00 0.00 0.01 6239 1  
## b 0.99 0 0.01 0.98 0.98 0.99 0.99 1.00 7006 1  
## sigma2 0.02 0 0.00 0.02 0.02 0.02 0.02 0.03 3775 1  
##   
## Samples were drawn using NUTS(diag\_e) at Mon Dec 12 16:42:10 2022.  
## For each parameter, n\_eff is a crude measure of effective sample size,  
## and Rhat is the potential scale reduction factor on split chains (at   
## convergence, Rhat=1).

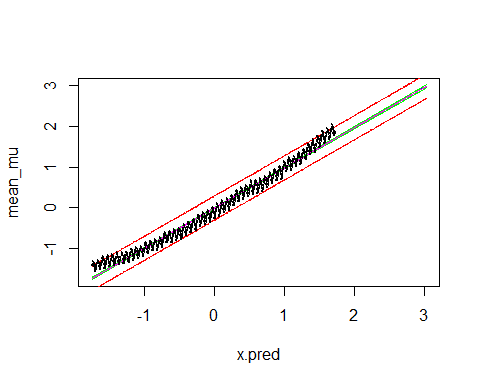
#print(post)  
plot(post, pars=c("a","b","sigma2"),plotfun= "trace", inc\_warmup = TRUE)



plot(post, pars=c("a","b","sigma2"), plotfun= "trace", inc\_warmup = FALSE)

 The Rhat-values of 1 and the convergence seem good.

set.seed(123)  
# extract the samples into matrix  
post\_sample <- as.matrix(post, pars =c("a","b","sigma2")) # combine all chains into one matrix in R workspace  
a\_dot=post\_sample[,1]  
b\_dot=post\_sample[,2]  
sigma2\_dot=post\_sample[,3]  
  
  
# This means that the prediction inputs have to be standardized as well  
  
x.pred= (seq(1,70\*12,length=70\*12)-mx)/stdx  
mu = matrix(NA,length(x.pred),length(b\_dot))  
y.tilde = matrix(NA,length(x.pred),length(b\_dot))  
  
mean\_mu=rep(NA, length(x.pred))  
int\_mu = matrix(NA,length(x.pred),2)  
  
mean\_y=rep(NA, length(x.pred))  
int\_y = matrix(NA,length(x.pred),2)  
  
for (i in 1:length(x.pred)) {  
 #mu[i,] = (a + b\*x.pred[i])\*stdy + my  
 mu[i,] = a\_dot + b\_dot\*x.pred[i]  
 mean\_mu[i]=mean(mu[i,])  
 int\_mu[i,] = quantile(mu[i,],probs=c(0.025,0.975))  
 #y\_i = mu\_i + e\_i and e\_i ~ N(0,sigma2)  
 y.tilde[i,] = mu[i,] + rnorm(length(mu[i,]), 0, sqrt(sigma2\_dot))  
 mean\_y[i]=mean(y.tilde[i,])  
 int\_y[i,] = quantile(y.tilde[i,],probs=c(0.025,0.975))  
   
}  
  
plot(x.pred,mean\_mu, type="l",col="blue") #posterior mean for mu(x)  
lines(x.pred,int\_mu[,1],col="green")  
lines(x.pred,int\_mu[,2],col="green") # 95% interval of mu(x)  
lines(x.pred,mean\_y, type="l",col="magenta") #posterior mean for y.tilde  
lines(x.pred,int\_y[,1],col="red")  
lines(x.pred,int\_y[,2],col="red") # 95% interval of y.tilde  
lines(x.month,y.month)  
points(x.month,y.month, cex=0.2)



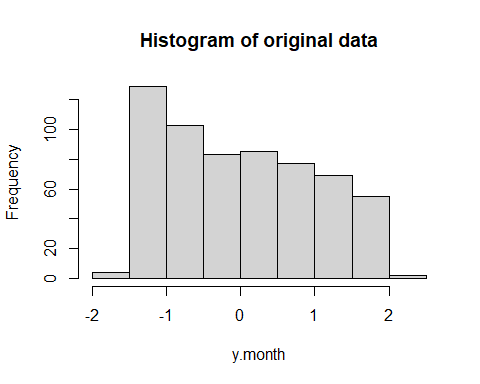
Now we are ready to conduct the posterior predictive check

First, sample 20 random draws from the Markov chain

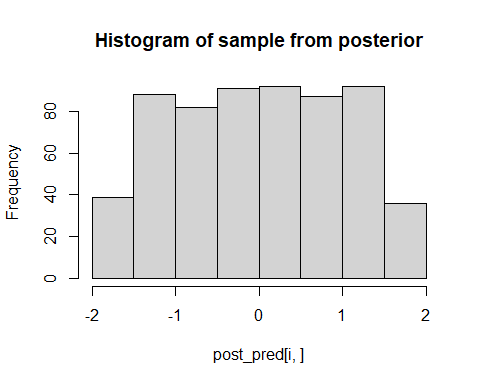
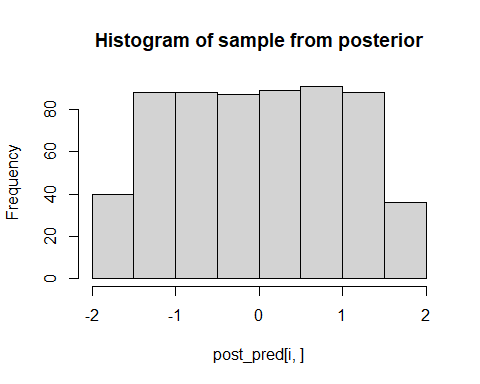
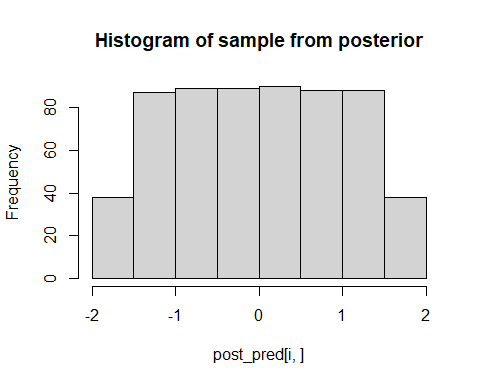
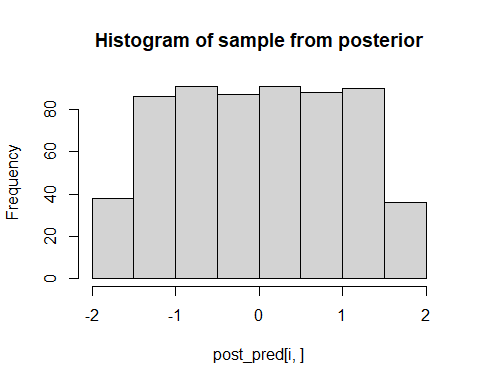
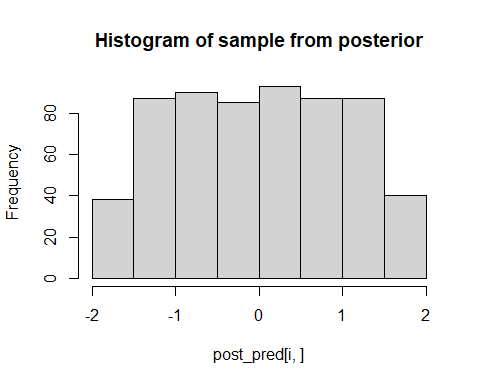
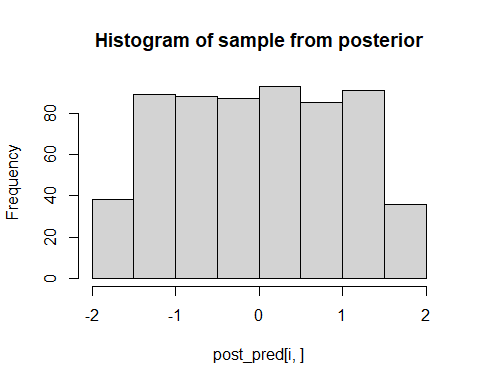
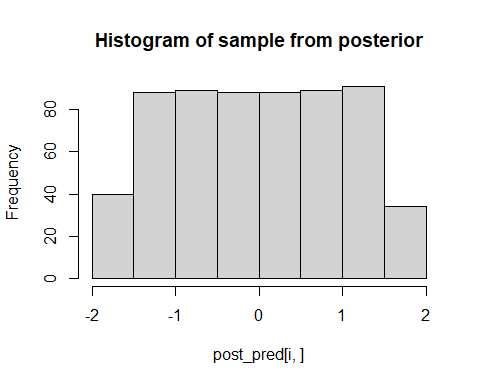
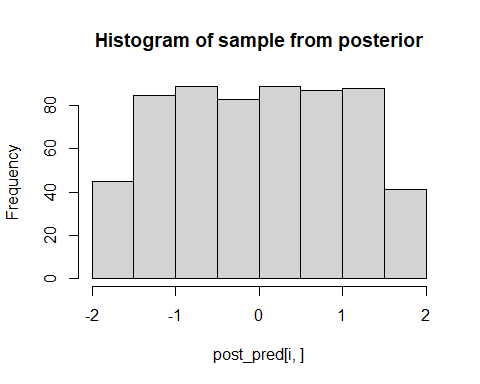
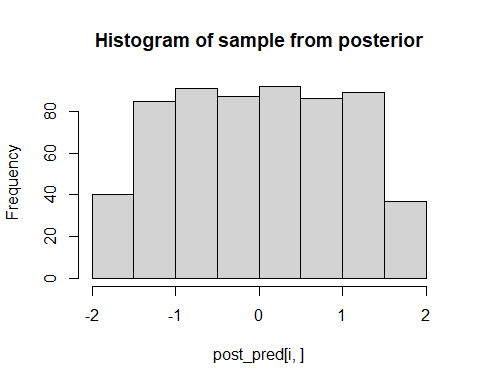
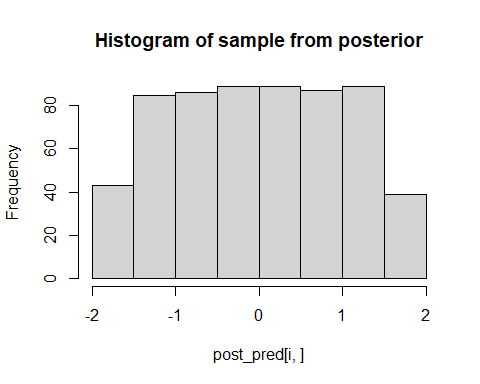
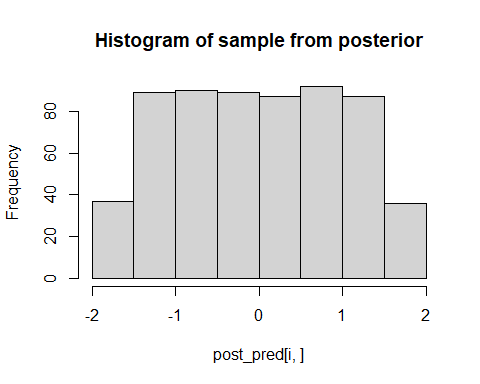
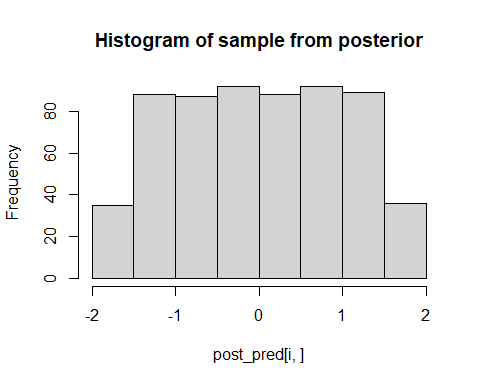
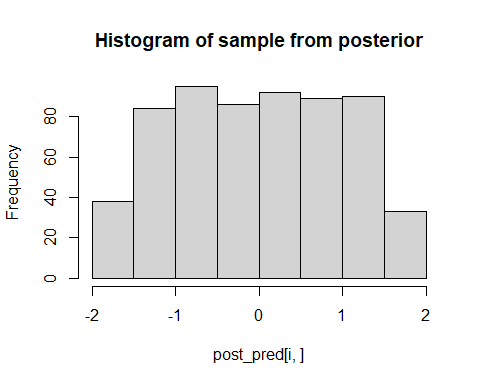
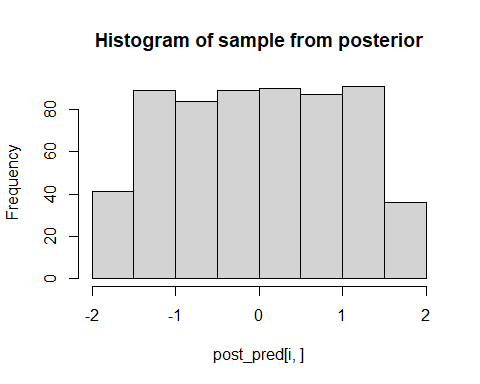
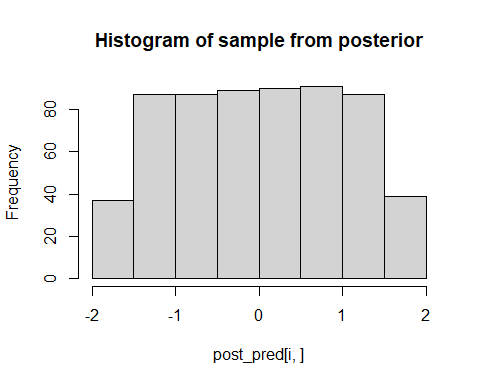
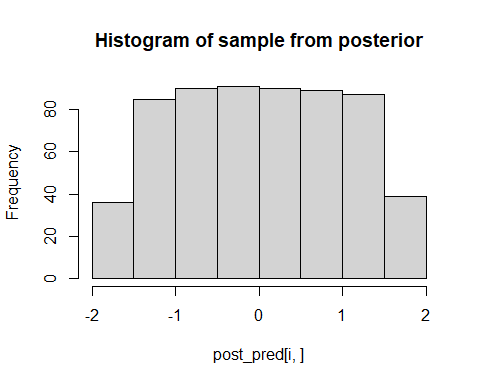
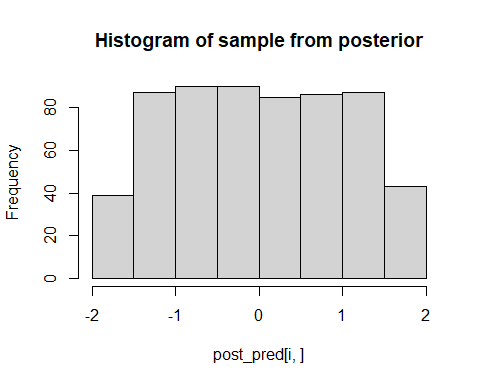
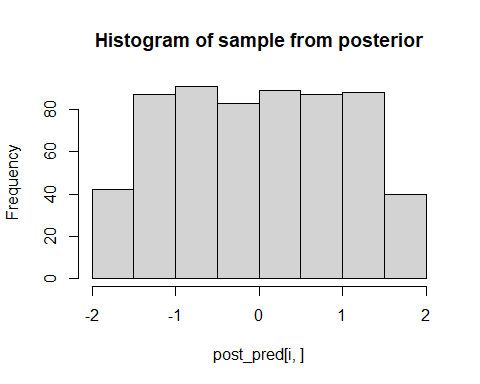
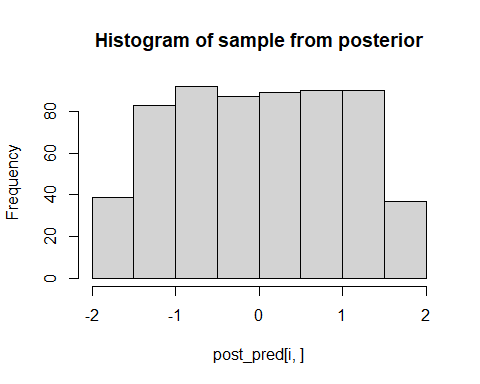
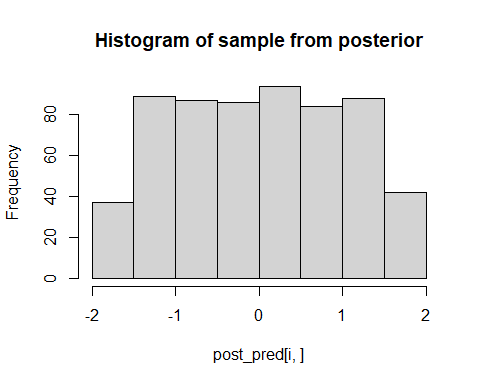
set.seed(123)  
  
pred\_sample <- post\_sample[sample(nrow(post\_sample),size=20,replace=TRUE),]

post\_pred <- array(dim=c(20, length(x.month)))  
for (i in 1:20){  
 for (j in 1:length(x.month)){  
 post\_pred[i,j] <- rnorm(1, (pred\_sample[i, 'a'] + pred\_sample[i, 'b'] \* x.month[j]), pred\_sample[i, 'sigma2'])  
 }  
}  
#for (i in 1:20){plot(x.month, post\_pred[i,])}

hist(y.month, main = "Histogram of original data")



for (i in 1:20){hist(post\_pred[i,], main="Histogram of sample from posterior")}

 **Analysis:**

From the histograms we see that the posterior predictive samples are not very convincing. They are very evenly distributed over the range of x, while the original data was more heavy on the lower values. It seems like this model can not replicate this feature in the real data, and we probably ought to look for another, more correct model.

Let’s refine the model so that

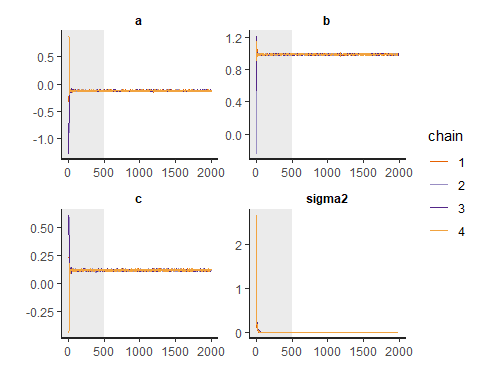
and rerun the analysis

mauna\_loa\_c02\_model2 = "  
data{  
 int<lower=0> N; // number of observations   
 real y[N]; // observed CO2 values   
 real x[N]; // observed times   
}  
parameters{  
 real a;  
 real b;  
 real c;  
 real<lower=0> sigma2;   
}  
transformed parameters{  
 real<lower=0> sigma;  
 real mu[N];  
   
 sigma=sqrt(sigma2);  
   
 for( i in 1 : N ) {  
 mu[i] = a + b \* x[i] + c \* square(x[i]);  
 }  
}  
model{  
 a ~ normal( 0, sqrt(1e6));  
 b ~ normal( 0, sqrt(1e6));  
 c ~ normal( 0, sqrt(1e6));  
 sigma2 ~ inv\_gamma(0.001,0.001);  
   
 for( i in 1 : N ) {  
 y[i] ~ normal(mu[i],sigma);  
 }  
}  
"

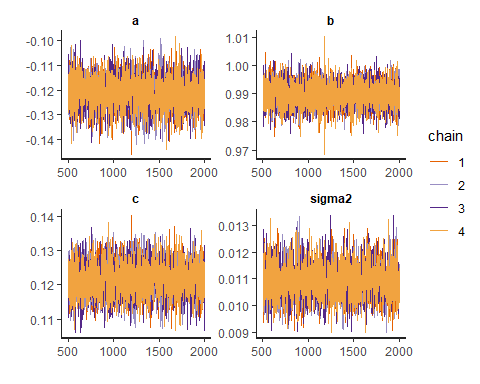
post2=stan(model\_code=mauna\_loa\_c02\_model2,data=data,warmup=500,iter=2000,chains=4,thin=1,control = list(adapt\_delta = 0.8,max\_treedepth = 10))  
  
# Check for convergence, see PSRF (Rhat in Stan)  
print(post2,pars=c("a","b","c", "sigma2"))

## Inference for Stan model: anon\_model.  
## 4 chains, each with iter=2000; warmup=500; thin=1;   
## post-warmup draws per chain=1500, total post-warmup draws=6000.  
##   
## mean se\_mean sd 2.5% 25% 50% 75% 97.5% n\_eff Rhat  
## a -0.12 0 0.01 -0.13 -0.13 -0.12 -0.12 -0.11 3724 1  
## b 0.99 0 0.00 0.98 0.99 0.99 0.99 1.00 6807 1  
## c 0.12 0 0.00 0.11 0.12 0.12 0.12 0.13 4280 1  
## sigma2 0.01 0 0.00 0.01 0.01 0.01 0.01 0.01 3097 1  
##   
## Samples were drawn using NUTS(diag\_e) at Mon Dec 12 16:43:59 2022.  
## For each parameter, n\_eff is a crude measure of effective sample size,  
## and Rhat is the potential scale reduction factor on split chains (at   
## convergence, Rhat=1).

#print(post)  
plot(post2, pars=c("a","b", "c","sigma2"),plotfun= "trace", inc\_warmup = TRUE)



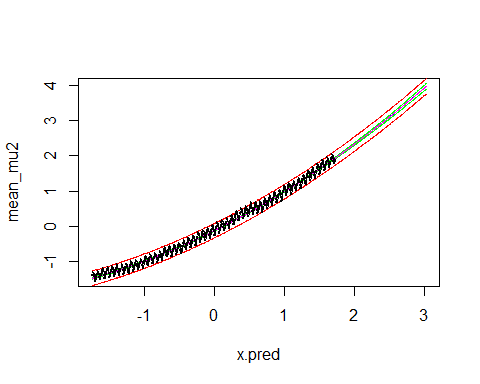
plot(post2, pars=c("a","b", "c","sigma2"), plotfun= "trace", inc\_warmup = FALSE)

 The Rhat-values of 1 and the convergence and autocorrelation seem good.

Let’s then examine the prediction along months

set.seed(123)  
# extract the samples into matrix  
post\_sample2 <- as.matrix(post2, pars =c("a","b", "c","sigma2")) # combine all chains into one matrix in R workspace  
a\_dot2=post\_sample2[,1]  
b\_dot2=post\_sample2[,2]  
c\_dot2=post\_sample2[,3]  
sigma2\_dot2=post\_sample2[,4]

mu2 = matrix(NA,length(x.pred),length(b\_dot))  
y.tilde2 = matrix(NA,length(x.pred),length(b\_dot))  
  
mean\_mu2=rep(NA, length(x.pred))  
int\_mu2 = matrix(NA,length(x.pred),2)  
  
mean\_y2=rep(NA, length(x.pred))  
int\_y2 = matrix(NA,length(x.pred),2)  
  
for (i in 1:length(x.pred)) {  
 #mu[i,] = (a + b\*x.pred[i])\*stdy + my  
 mu2[i,] = a\_dot2 + b\_dot2\*x.pred[i] + c\_dot2\*(x.pred[i]^2)  
 mean\_mu2[i]=mean(mu2[i,])  
 int\_mu2[i,] = quantile(mu2[i,],probs=c(0.025,0.975))  
 #y\_i = mu\_i + e\_i and e\_i ~ N(0,sigma2)  
 y.tilde2[i,] = mu2[i,] + rnorm(length(mu2[i,]), 0, sqrt(sigma2\_dot2))  
 mean\_y2[i]=mean(y.tilde2[i,])  
 int\_y2[i,] = quantile(y.tilde2[i,],probs=c(0.025,0.975))  
   
}  
  
plot(x.pred,mean\_mu2, type="l",col="blue") #posterior mean for mu(x)  
lines(x.pred,int\_mu2[,1],col="green")  
lines(x.pred,int\_mu2[,2],col="green") # 95% interval of mu(x)  
lines(x.pred,mean\_y2, type="l",col="magenta") #posterior mean for y.tilde  
lines(x.pred,int\_y2[,1],col="red")  
lines(x.pred,int\_y2[,2],col="red") # 95% interval of y.tilde  
lines(x.month,y.month)  
points(x.month,y.month, cex=0.2)

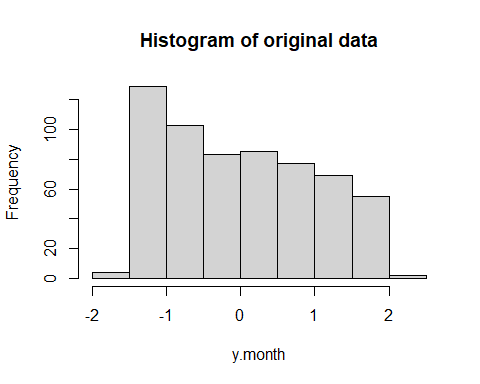
 **Analysis of linear graph**

From the graph we can see that the prediction by the model is closer to the curved shape that we can observe in the original data. A curve like this in a linear regression task usually is a sign that it would be good to check a model using the squares or higher powers of the features in the data.

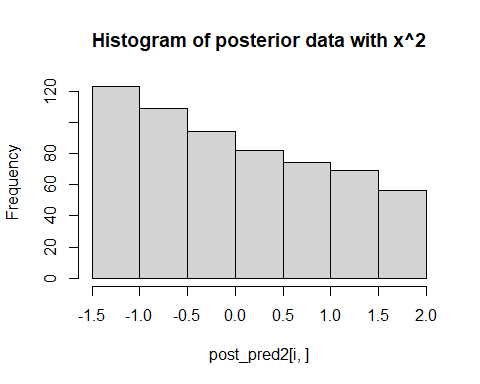
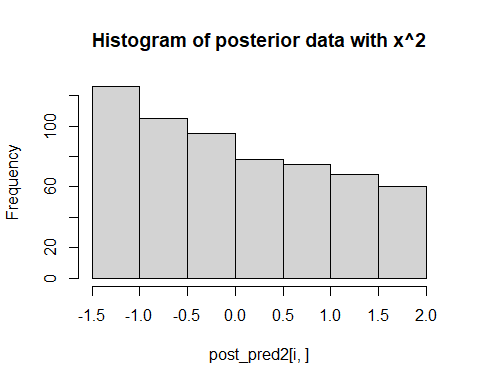
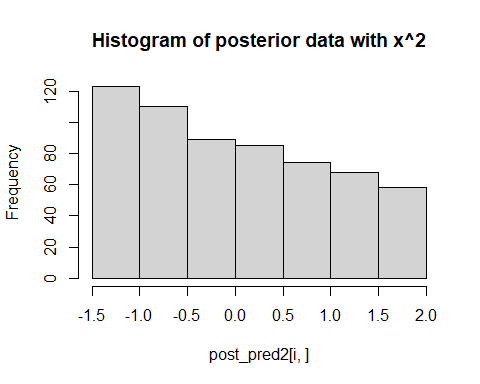
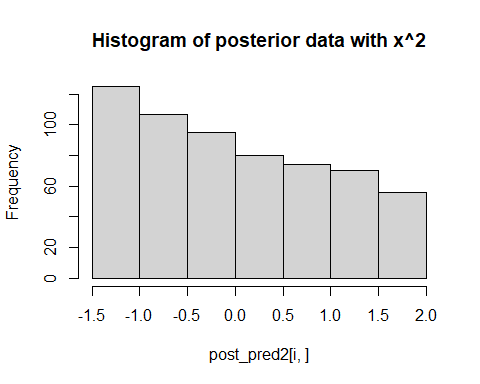
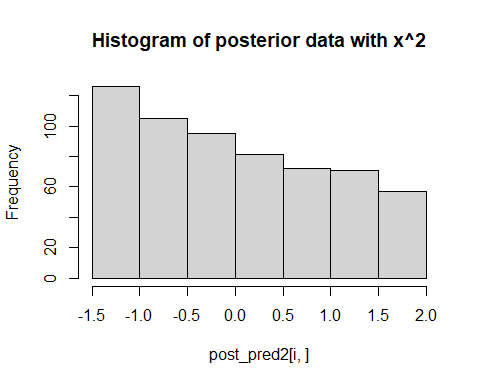
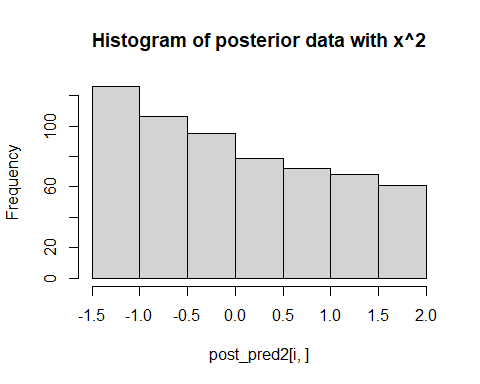
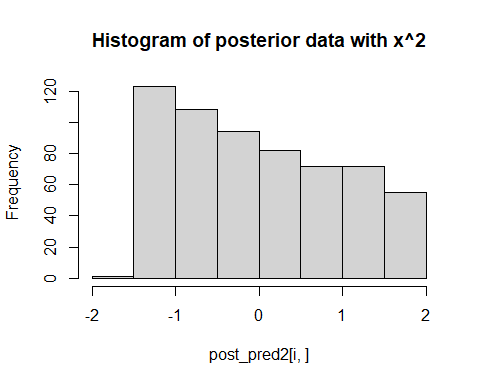
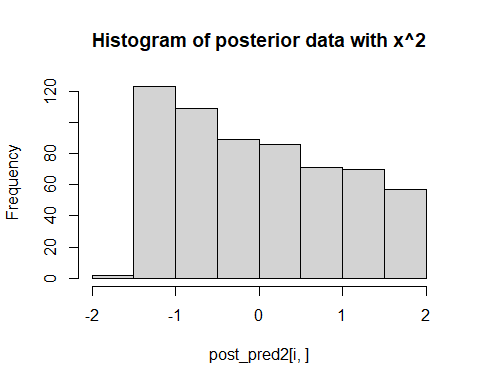
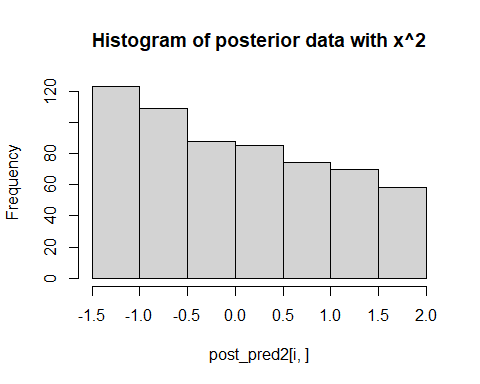
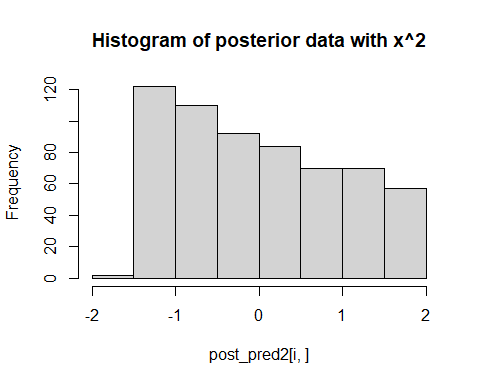
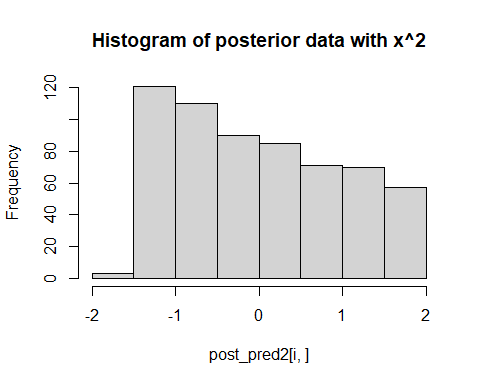
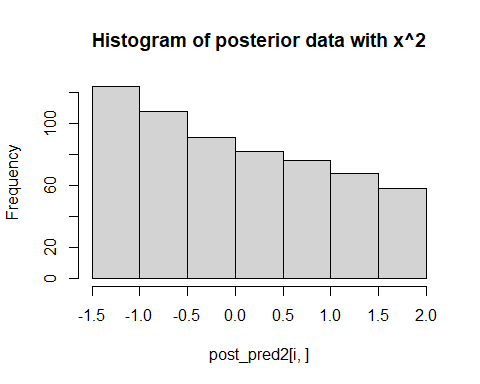
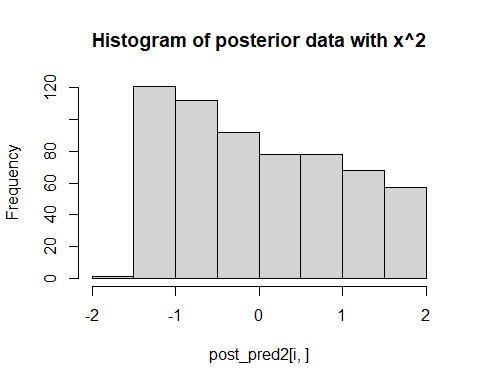
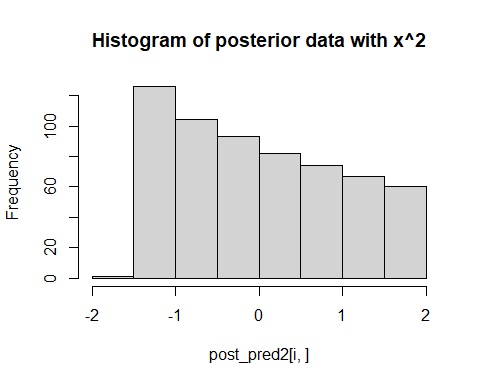
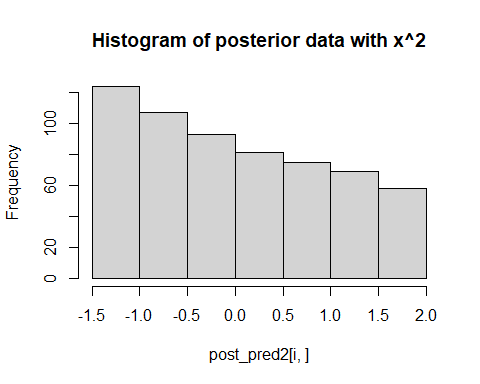
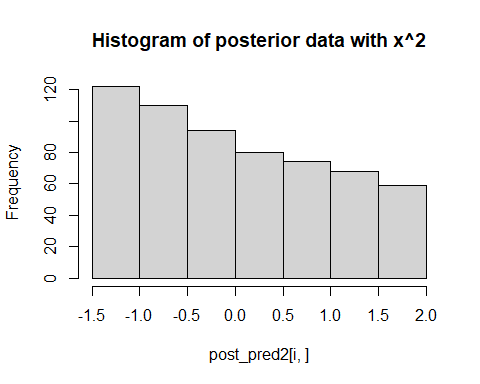
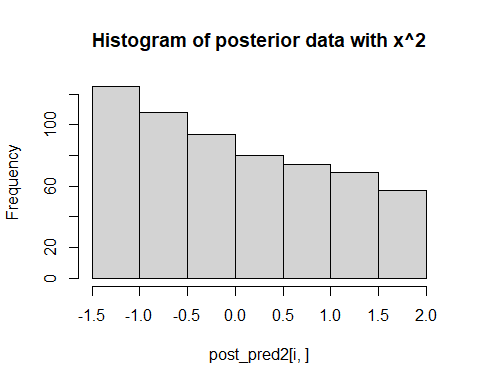
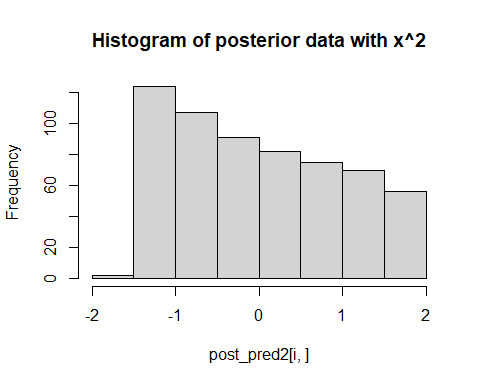
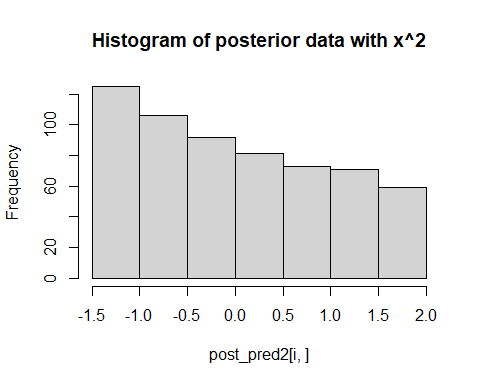
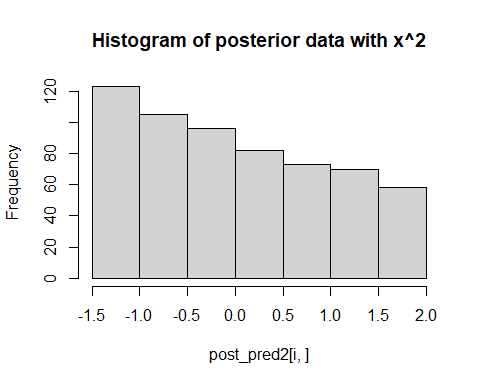
And then we can do the asked posterior predictive check

pred\_sample2 <- post\_sample2[sample(nrow(post\_sample2),size=20,replace=TRUE),]  
post\_pred2 <- array(dim=c(20, length(x.month)))  
for (i in 1:20){  
 for (j in 1:length(x.month)){  
 post\_pred2[i,j] <- rnorm(1, (pred\_sample2[i, 'a'] + pred\_sample2[i, 'b'] \* x.month[j] + pred\_sample2[i, 'c'] \* (x.month[j]^2)), pred\_sample[i, 'sigma2'])  
 }  
}  
#for (i in 1:20){plot(x.month, post\_pred2[i,])}

hist(y.month, main="Histogram of original data")



for (i in 1:20){hist(post\_pred2[i,], main="Histogram of posterior data with x^2")}

 **Analysis**

We see that with this model, the posterior predictive samples are much more similar to the original data. The samples are more concentrated at lower values, exactly like in the original data, and as can be expected with a linear model of with the square of a feature.

## Posterior predictive comparison

Let’s next compare the models’ cababilities to predict unseen data. For this we divide the data into training and test sets, infer the model parameters with the former and evaluate models performance in predicting the latter

Split the data into training and test sets so that you put every other data point into training and every other into test (we could do random split as well).

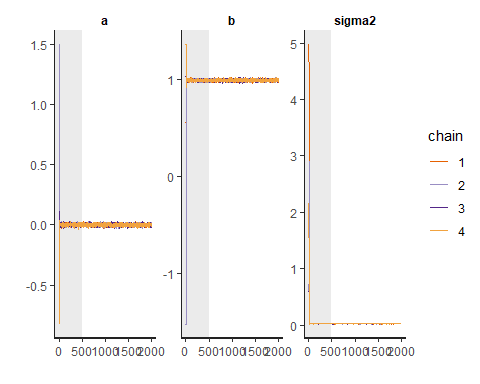
x.month.train <- x.month[seq(1, length(x.month), 2)]  
x.month.test <- x.month[seq(2, length(x.month), 2)]  
y.month.train <- y.month[seq(1, length(y.month), 2)]  
y.month.test <- y.month[seq(2, length(y.month), 2)]

Sample from the posterior distribution of the first model () conditional on the training data and check for convergence

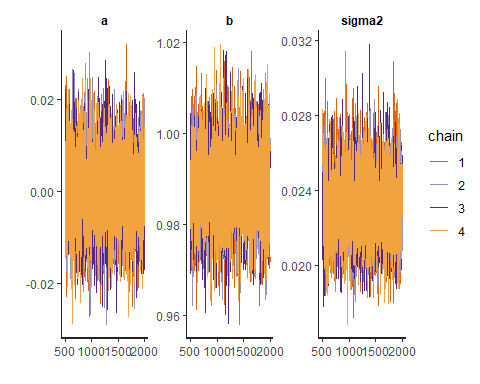
set.seed(123)  
data <- list (N=length(x.month.train), y=y.month.train, x=x.month.train)  
post3=stan(model\_code=mauna\_loa\_c02\_model,data=data,warmup=500,iter=2000,chains=4,thin=1,control = list(adapt\_delta = 0.8,max\_treedepth = 10))  
  
# Check for convergence, see PSRF (Rhat in Stan)  
print(post3,pars=c("a","b","sigma2"))

## Inference for Stan model: anon\_model.  
## 4 chains, each with iter=2000; warmup=500; thin=1;   
## post-warmup draws per chain=1500, total post-warmup draws=6000.  
##   
## mean se\_mean sd 2.5% 25% 50% 75% 97.5% n\_eff Rhat  
## a 0.00 0 0.01 -0.02 -0.01 0.00 0.01 0.02 6211 1  
## b 0.99 0 0.01 0.97 0.98 0.99 0.99 1.01 7153 1  
## sigma2 0.02 0 0.00 0.02 0.02 0.02 0.02 0.03 4318 1  
##   
## Samples were drawn using NUTS(diag\_e) at Mon Dec 12 16:44:43 2022.  
## For each parameter, n\_eff is a crude measure of effective sample size,  
## and Rhat is the potential scale reduction factor on split chains (at   
## convergence, Rhat=1).

#print(post)  
plot(post3, pars=c("a","b","sigma2"),plotfun= "trace", inc\_warmup = TRUE)



plot(post3, pars=c("a","b","sigma2"), plotfun= "trace", inc\_warmup = FALSE)



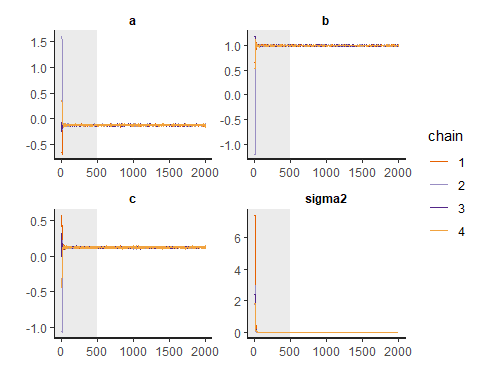
The Rhats are good at 1 and visually one can see that the chains have converged.

Sample from the posterior distribution of the second model () conditional on the training data and check for convergence

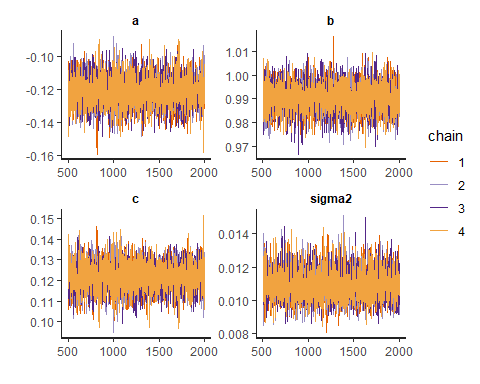
set.seed(123)  
post4=stan(model\_code=mauna\_loa\_c02\_model2,data=data,warmup=500,iter=2000,chains=4,thin=1,control = list(adapt\_delta = 0.8,max\_treedepth = 10))  
  
# Check for convergence, see PSRF (Rhat in Stan)  
print(post4,pars=c("a","b","c", "sigma2"))

## Inference for Stan model: anon\_model.  
## 4 chains, each with iter=2000; warmup=500; thin=1;   
## post-warmup draws per chain=1500, total post-warmup draws=6000.  
##   
## mean se\_mean sd 2.5% 25% 50% 75% 97.5% n\_eff Rhat  
## a -0.12 0 0.01 -0.14 -0.13 -0.12 -0.12 -0.10 3544 1  
## b 0.99 0 0.01 0.98 0.99 0.99 0.99 1.00 5258 1  
## c 0.12 0 0.01 0.11 0.12 0.12 0.13 0.13 3735 1  
## sigma2 0.01 0 0.00 0.01 0.01 0.01 0.01 0.01 3826 1  
##   
## Samples were drawn using NUTS(diag\_e) at Mon Dec 12 16:45:17 2022.  
## For each parameter, n\_eff is a crude measure of effective sample size,  
## and Rhat is the potential scale reduction factor on split chains (at   
## convergence, Rhat=1).

#print(post)  
plot(post4, pars=c("a","b","c", "sigma2"),plotfun= "trace", inc\_warmup = TRUE)



plot(post4, pars=c("a","b","c", "sigma2"), plotfun= "trace", inc\_warmup = FALSE)



Again, Rhats and convergence looks fine.

Now we can calculate point-wise posterior predictive log density and RMSE at test locations

Starting with RMSE:

post\_sample3 <- as.matrix(post3, pars =c("a","b","sigma2"))  
a3 <- post\_sample3[, 'a']  
b3 <- post\_sample3[, 'b']  
sigma23 <- post\_sample3[, 'sigma2']

mu3 <- array(dim = c(length(x.month.test)))  
for (i in 1:length(x.month.test)){  
 mu3[i] <- mean(a3 + b3\*x.month.test[i])  
}

RMSE <- sqrt(sum((y.month.test - mu3 )^2))  
RMSE

## [1] 2.616689

post\_sample4 <- as.matrix(post4, pars =c("a","b","c", "sigma2"))  
a4 <- post\_sample4[, 'a']  
b4 <- post\_sample4[, 'b']  
c4 <- post\_sample4[, 'c']  
sigma24 <- post\_sample4[, 'sigma2']  
  
mu4 <- array(dim = c(length(x.month.test)))  
for (i in 1:length(x.month.test)){  
 mu4[i] <- mean(a4 + b4\*x.month.test[i] + c4\*(x.month.test[i]^2))  
}  
  
RMSE2 <- sqrt(sum((y.month.test - mu4 )^2))  
RMSE2

## [1] 1.808179

**Analysis of RMSE** The RMSE of the squared, second model is at lower than the RMSE for the first linear model at , which indicates that the second model () produces predictions that were closer to the measured values. The sum of the squares of differences between the predictions and the actual values is smaller.

Then let’s have a look at the log predictive density:

loglik1 <- array(dim = c(length(x.month.test)))  
for (i in 1:length(x.month.test)){  
 loglik1[i] <- log(mean(dnorm(y.month.test[i], a3 + b3\*x.month.test[i], sigma23)))  
}  
sum(loglik1)

## [1] -3186.7

loglik2 <- array(dim = c(length(x.month.test)))  
for (i in 1:length(x.month.test)){  
 loglik2[i] <- log(mean(dnorm(y.month.test[i], a4 + b4\*x.month.test[i] + c4\*(x.month.test[i]^2), sigma24)))  
}  
sum(loglik2)

## [1] -5926.257

#y.month.test

**Analysis of predictive log likelihood**

The result of these calculations are a bit unclear to me. As I understand it, a lower log likelihood would imply a worse model, so in that case, this test would show that the model with a squared feature would not be the better model. I therefore draw the conclusion that either there is something wrong with the calculation (which would be strange, as I attended the exercise session specifically to ask for help for this, so it would be confusing if I wouldn’t have received correct guidance), or then I am incorrect in interpreting the predictive log likelihood.