Week 6

Exercise 1

In this exercise, we continue the analysis of the white fish larval areas (week 2, exercise 3 and week4, exercise 2). This time we extend the model to include regression along two continues covariates in addition to the vegetation cover status. The additional covariates that we are interested in are the distance to sandy shore and the length of ice cover during winter. Many fishermen have observed that white fish are caught more easily from sandy shores than elsewhere during their spawning season. Moreover, white fish spawn their eggs during fall but the larvae hatch only in the spring. Hence, it has been suggested that longer ice cover period works as a shelter for the eggs. Hence, let's take a look whether there is statistical signal to these covariates.

Let's load the data and construct covariate matrix X (a matrix where the i'th row contains the covariates for the i'th sampling site), vector of area indexes a (areas in the code) and vector of white fish presence-absence observations y.

```
# Read the data
data = read.csv("white_fishes_data.csv")

# Let's then take the covariates to matrix X and standardize them
X = data[,c("DIS_SAND","ICELAST09","BOTTOMCOV")]

# And for last let's take the presence-absence observations of white fish
larvae into Y
y = data$WHIBIN
```

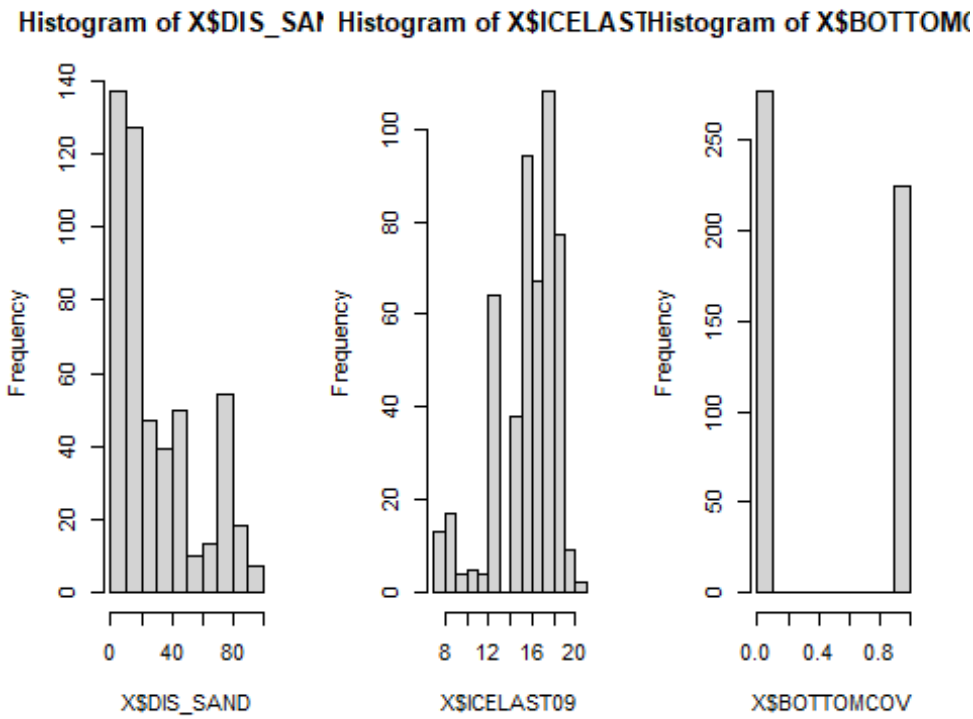
Unlike in our previous analyses of this data we treat each sampling site as one observation and consider the triplets $\{y_i, a_i, X_i\}$ (X_i is the i'th row of X) exchangeable.

We will first build the following model to analyze the data. Hence, we assume that the prior expectation of the probability to observe white fish larvae ($E[y_i] = \theta_i$) follows logit linear model where α is the intercept and β is a 3×1 vector of (fixed) effects of covariates. Note that the matrix notation $X\beta$ is the same as writing

$$X\beta = \beta_1 \times \text{DISSAND} + \beta_2 \times \text{ICELAST09} + \beta_3 \times \text{BOTTOMCOV}$$

Note also that the DIS_SAND and ICELAST09 are continuous covariates whereas BOTTOMCOV is a categorical covariate getting value 1 if the bottom is covered by vegetation and 0 if the bottom is not covered by vegetation.

```
par(mfrow=c(1,3))
hist(X$DIS_SAND)
hist(X$ICELAST09)
hist(X$BOTTOMCOV)
```



Hence, the parameter β_3 corresponds to the effect of vegetation to the observation probability of white fish larvae.

Before starting the analysis we standardize the continues covariates but not the categorical BOTTOMCOV covariate. If we standardized the categorical variable the interpretation of β_3 parameter would change.

```
mx = colMeans(X[,1:2])
stdx = apply(X[,1:2],2,sd)
X[,1:2] = (X[,1:2]-t(replicate(dim(X)[1],mx)))/t(replicate(dim(X)[1],stdx))
```

Task 1.

Implement the model in Stan and sample from the posterior for the parameters α and β. Check for convergence of the MCMC chain and examine the autocorrelation of the samples. Visualize the posterior for α and β and discuss the results.

First we define the model:

```
logit_fish_model = "
data{
   int<lower = 0> n;
   real DIS_SAND[n];
   real ICELAST09[n];
   int BOTTOMCOV[n];
   int<lower=0,upper=1> y[n];
}
```

```
parameters{
   real alpha;
   real beta_1;
   real beta_2;
   real beta_3;
}

model{
   //priors:
   alpha ~ normal(0, sqrt(10));
   beta_1 ~ normal(0, sqrt(10));
   beta_2 ~ normal(0, sqrt(10));
   beta_3 ~ normal(0, sqrt(10));
   for (i in 1:n){
      y[i] ~ bernoulli_logit(alpha + beta_1*DIS_SAND[i] + beta_2*ICELAST09[i]
+ beta_3*BOTTOMCOV[i]);
   }
}

"
```

Then prepare the data:

```
data <- list (n=length(y), DIS_SAND=X$DIS_SAND, ICELAST09=X$ICELAST09,
BOTTOMCOV=X$BOTTOMCOV, y=y)
```

Then running the model:

```
##
## 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

## Warning: package 'coda' was built under R version 4.2.2

##
## Attaching package: 'coda'

## The following object is masked from 'package:rstan':
##
##     traceplot
```

```
post=stan(model_code=logit_fish_model,data=data,warmup=500,iter=2000,chains=4
,thin=1)

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000687 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would
take 6.87 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  501 / 2000 [ 25%]  (Sampling)
## Chain 1: Iteration:  700 / 2000 [ 35%]  (Sampling)
## Chain 1: Iteration:  900 / 2000 [ 45%]  (Sampling)
## Chain 1: Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1: Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1: Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1: Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1: Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.299 seconds (Warm-up)
## Chain 1:                3.404 seconds (Sampling)
## Chain 1:                4.703 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000292 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would
take 2.92 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  501 / 2000 [ 25%]  (Sampling)
## Chain 2: Iteration:  700 / 2000 [ 35%]  (Sampling)
## Chain 2: Iteration:  900 / 2000 [ 45%]  (Sampling)
## Chain 2: Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2: Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2: Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2: Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2: Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
## Chain 2:
## Chain 2:  Elapsed Time: 1.276 seconds (Warm-up)
## Chain 2:                3.888 seconds (Sampling)
## Chain 2:                5.164 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000299 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would
take 2.99 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  501 / 2000 [ 25%]  (Sampling)
## Chain 3: Iteration:  700 / 2000 [ 35%]  (Sampling)
## Chain 3: Iteration:  900 / 2000 [ 45%]  (Sampling)
## Chain 3: Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3: Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3: Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3: Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3: Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1.409 seconds (Warm-up)
## Chain 3:                4.08 seconds (Sampling)
## Chain 3:                5.489 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000321 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would
take 3.21 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  501 / 2000 [ 25%]  (Sampling)
## Chain 4: Iteration:  700 / 2000 [ 35%]  (Sampling)
## Chain 4: Iteration:  900 / 2000 [ 45%]  (Sampling)
## Chain 4: Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4: Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4: Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4: Iteration: 1700 / 2000 [ 85%]  (Sampling)
```

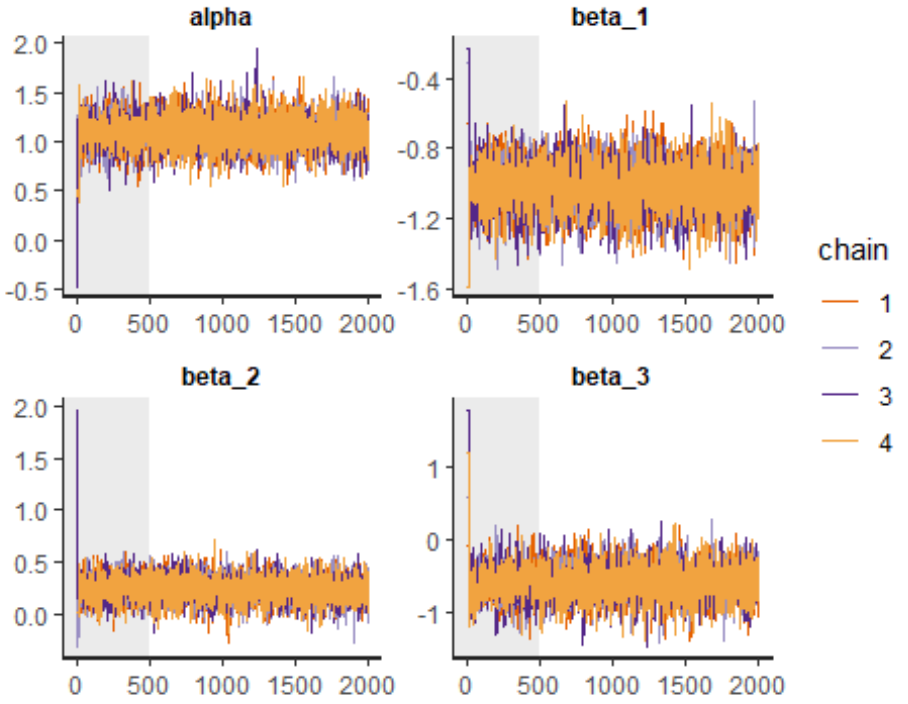
```
## Chain 4: Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1.334 seconds (Warm-up)
## Chain 4:                4.701 seconds (Sampling)
## Chain 4:                6.035 seconds (Total)
## Chain 4:
```

Convergence etc.:

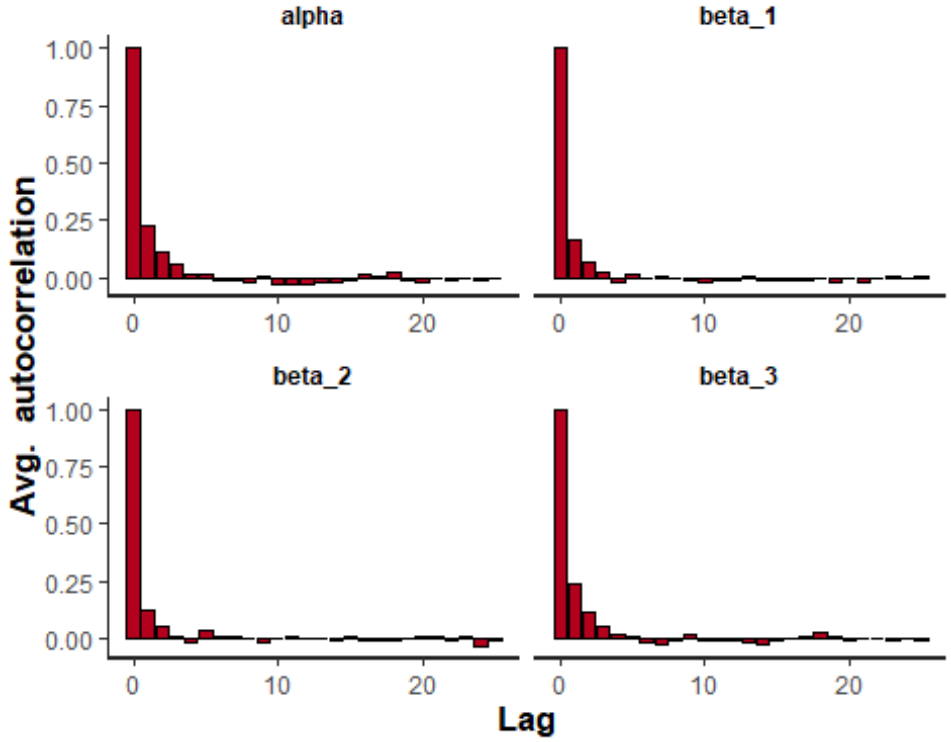
```
print(post)
```

```
## 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
## alpha     1.09    0.00 0.17    0.77    0.98    1.09    1.20    1.42  3244
1
## beta_1   -1.02    0.00 0.13   -1.28   -1.11   -1.02   -0.93   -0.76  3987
1
## beta_2    0.24    0.00 0.12    0.00    0.16    0.24    0.32    0.47  4162
1
## beta_3   -0.58    0.00 0.24   -1.04   -0.74   -0.58   -0.43   -0.12  3195
1
## lp__   -250.71    0.03 1.46 -254.36 -251.38 -250.37 -249.65 -248.92  2618
1
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 12 18:18:46 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).
```

```
plot(post,plotfun= "trace", inc_warmup = TRUE)
```

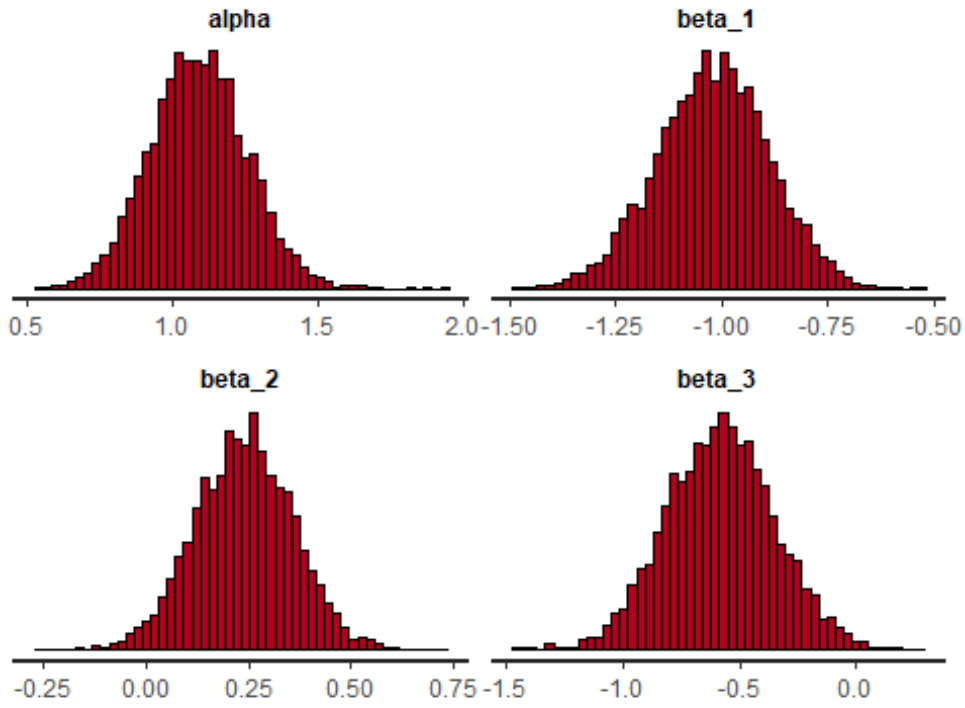



```
stan_ac(post,inc_warmup = FALSE, lags = 25)
```



As we can conclude from above, the convergence and autocorrelation of the chains were in order. Rhats at 1, visually we can see that the chains have converged, and the autocorrelation is around zero at all higher lags.

```
Nsamp=as.matrix(post)
plot(post, plotfun = "hist", pars = list('alpha', 'beta_1', 'beta_2',
'beta_3') ,bins=50)
```



As we can see from the histograms above, alpha, i.e., the intercept is around 1.2, while distance to sandy shores have a negative impact on the prevalence of white fish larvae, a longer icecover has a positive effect on white fish larvae, and (as we have seen in earler exercises) bottom cover vegetation has a negative effect of white fish larvae. The distribution of the parameters has retained a gaussian spread.

Task 2.

Calculate the posterior correlation between α and β_3. How does this differ from the prior correlation and why?

```
corr <- cor(Nsamp[,'alpha'], Nsamp[,'beta_3'])
corr

## [1] -0.7262299
```

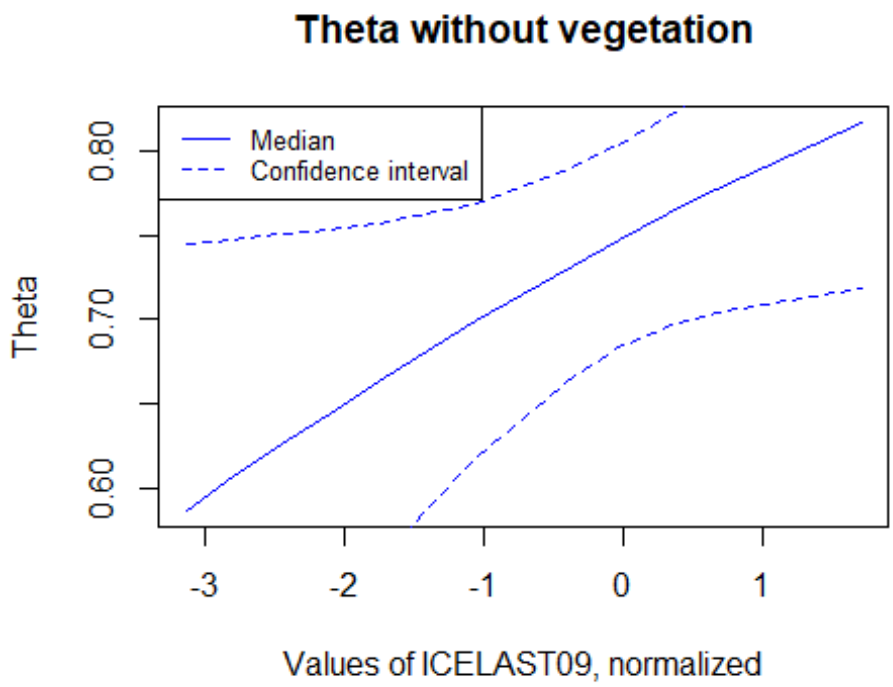
The prior did not still include any information on the target values y, and that is why the only thing that affected the values were the prior, uninformed and identical distribution. After training the model on the target data, the distribution of α and β_3 have changed to

adapt to their role in the linearly logistic model. The prior correlation, as both α and β_3 was drawn from a N(0,10) distribution, should have been close to 0. In the posterior, the correlation is -0.716, which shows, that with a higher intercept or α, β_3 needs to be lower, in order to reach the same result for $\alpha + \beta X$.

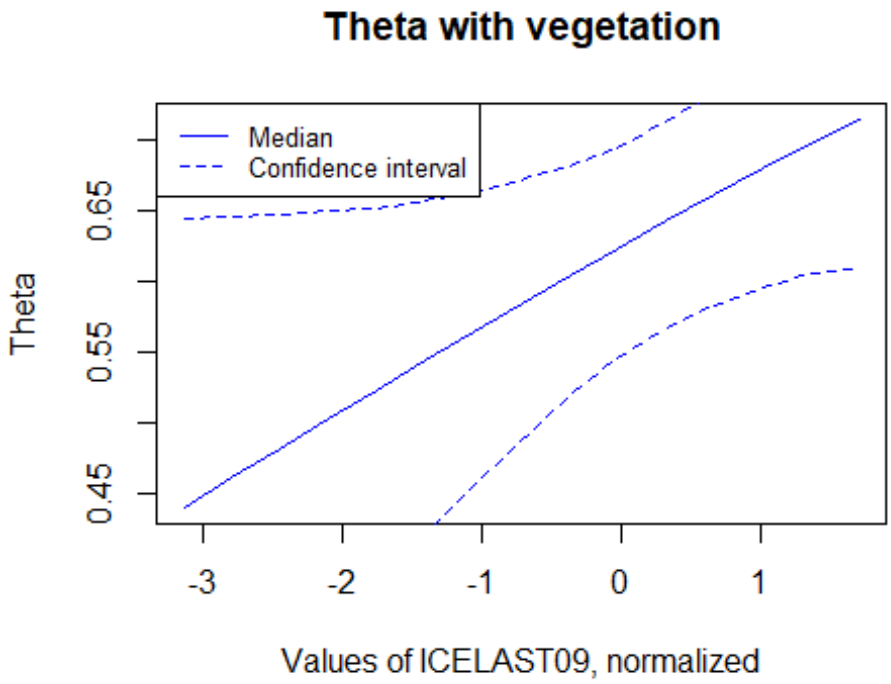
Task 3.

Visualize the posterior of θ as a function of ICELAST09 when DISSAND is set to its mean value and in both cases when BOTTOMCOV=0 and BOTTOMCOV=1. That is, draw the median and 95% credible interval of the prediction function within the range from minimum to maximum value of ICELAST09 in the data.

```
mean_sand = mean(X$DIS_SAND)
sigmoid = function(x) {
   1 / (1 + exp(-x))
}

#beta_2s = Nsamp[, 'beta_2']*X$ICELAST09#calculate this individually for all
values of icelast and plot median and confidence interval of result.
thetas_cov0 = array(dim=c(length(X$ICELAST09),3))
icelast <- X$ICELAST09
icelast <- sort(icelast)

for (i in 1:502){
  bx <- Nsamp[, 'alpha'] + Nsamp[,'beta_1']*mean_sand
+Nsamp[,'beta_2']*icelast[i] + Nsamp[,'beta_3'] * 0;
  theta <- sigmoid(bx)
  thetas_cov0[i,1] <- median(theta)
  thetas_cov0[i,2] <- quantile(theta,probs=c(0.025))
  thetas_cov0[i,3] <- quantile(theta,probs=c(0.975))
}

plot(icelast,thetas_cov0[,1], col="blue", main="Theta without vegetation",
type="l", xlab="Values of ICELAST09, normalized", ylab="Theta")
lines(icelast,thetas_cov0[,2], col="blue", lty=2, type='l')
lines(icelast,thetas_cov0[,3], col="blue", lty=2, type="l")
legend(x="topleft", legend=c("Median", "Confidence interval"),
       col=c("blue", "blue"), lty=1:2, cex=0.8)
```




```
thetas_cov1 = array(dim=c(length(X$ICELAST09),3))
for (i in 1:502){
  bx2 <- Nsamp[, 'alpha'] + Nsamp[,'beta_1']*mean_sand
+Nsamp[,'beta_2']*icelast[i] + Nsamp[,'beta_3'] * 1;
  #theta2 <- sigmoid(bx2)
  thetas_cov1[i,1] <- sigmoid(median(bx2))
  thetas_cov1[i,2] <- sigmoid(quantile(bx2,probs=c(0.025)))
  thetas_cov1[i,3] <- sigmoid(quantile(bx2,probs=c(0.975)))
}

plot(icelast,thetas_cov1[,1], col="blue", main="Theta with vegetation",
type="l", xlab="Values of ICELAST09, normalized", ylab="Theta")
lines(icelast,thetas_cov1[,2], col="blue", lty=2, type='l')
lines(icelast,thetas_cov1[,3], col="blue", lty=2, type="l")
legend(x="topleft", legend=c("Median", "Confidence interval"),
       col=c("blue", "blue"), lty=1:2, cex=0.8)
```



Task 4.

Visualize the posterior distribution of θ at location where DIS_SAND is 60 and ICELAST is 18 for both vegetated and non-vegetated bottom types as well as their difference.

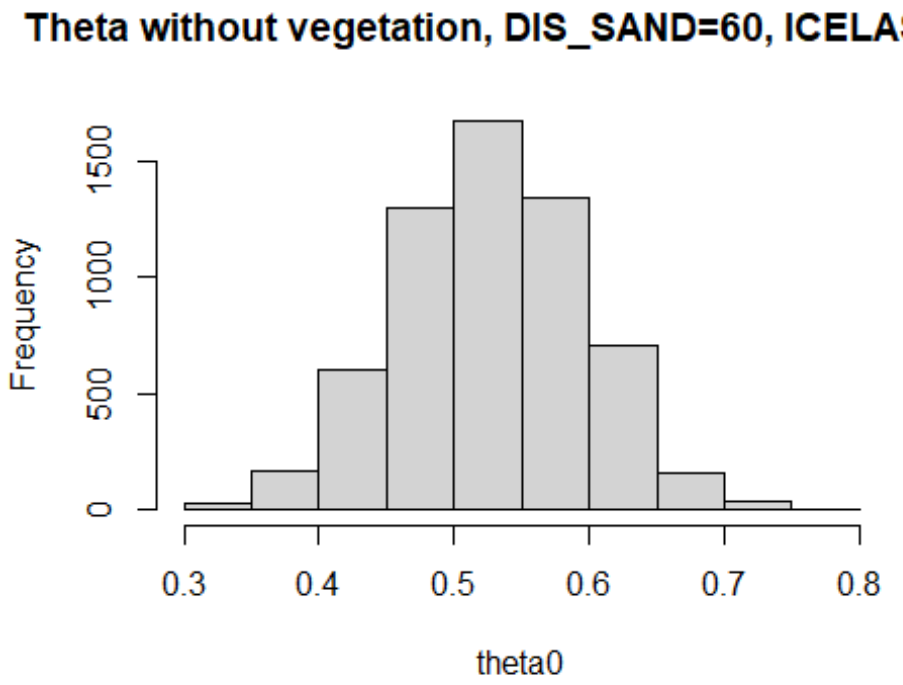
First normalizing the values given:

```
sand <- (60-mx['DIS_SAND'])/stdx['DIS_SAND']
ice <- (18-mx['ICELAST09'])/stdx['ICELAST09']
print(sand)

## DIS_SAND
##  1.11729

print(ice)

## ICELAST09
## 0.6707792
```

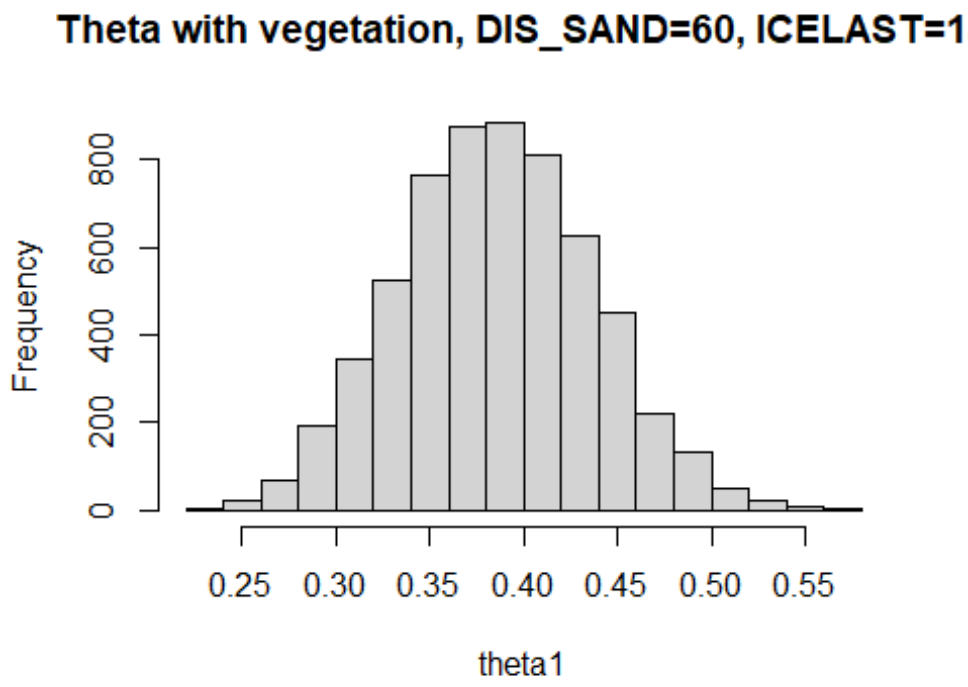
Then calculate thetas:

```
bx0 <- Nsamp[, 'alpha'] + Nsamp[,'beta_1']*sand +Nsamp[,'beta_2']*ice +
Nsamp[,'beta_3'] * 0;
theta0 <- sigmoid(bx0)

bx1 <- Nsamp[, 'alpha'] + Nsamp[,'beta_1']*sand +Nsamp[,'beta_2']*ice +
Nsamp[,'beta_3'] * 1;
theta1 <- sigmoid(bx1)
```

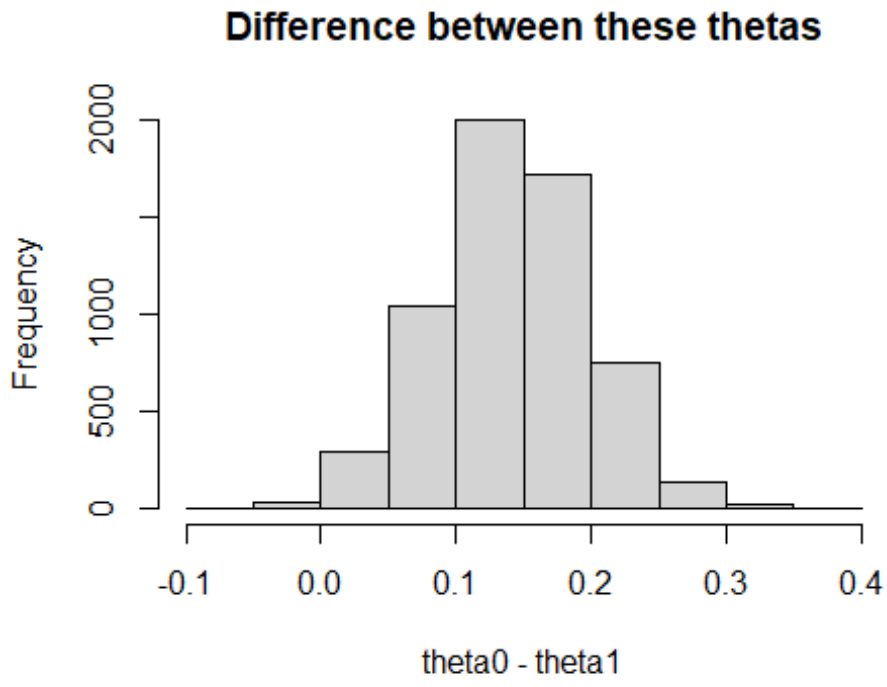
```
hist(theta0, main="Theta without vegetation, DIS_SAND=60, ICELAST=18")
```




```
hist(theta1, main="Theta with vegetation, DIS_SAND=60, ICELAST=18")
```




```
hist(theta0-theta1, main="Difference between these thetas")
```



Task 5.

How does the difference in θ for vegetated and non-vegetated bottom differ from $\phi = \Delta\theta = \theta_0 - \theta_1$ in exercise 3 of week 2 and $\delta\mu$ in exercise 2 of week 4? Would you say that the result concerning the effect of vegetation is consistent in all these different analyses? Which analysis would you prefer?

Compared to the difference of θ in week 2, the difference found here is somewhat smaller. The calculation in week 2 was a somewhat more naive approach, that did not take into considerations the difference between different areas. The difference found in week 4, again, is closer to this difference. As the calculation in week 4 took into consideration the possible differences in the sites, the logic of that θ is somewhat closer to this one, albeit the difference calculated above is for one specific site only.

The result is still somewhat consistent, a site without vegetation tends to have more white fish larvae than sites with vegetation.

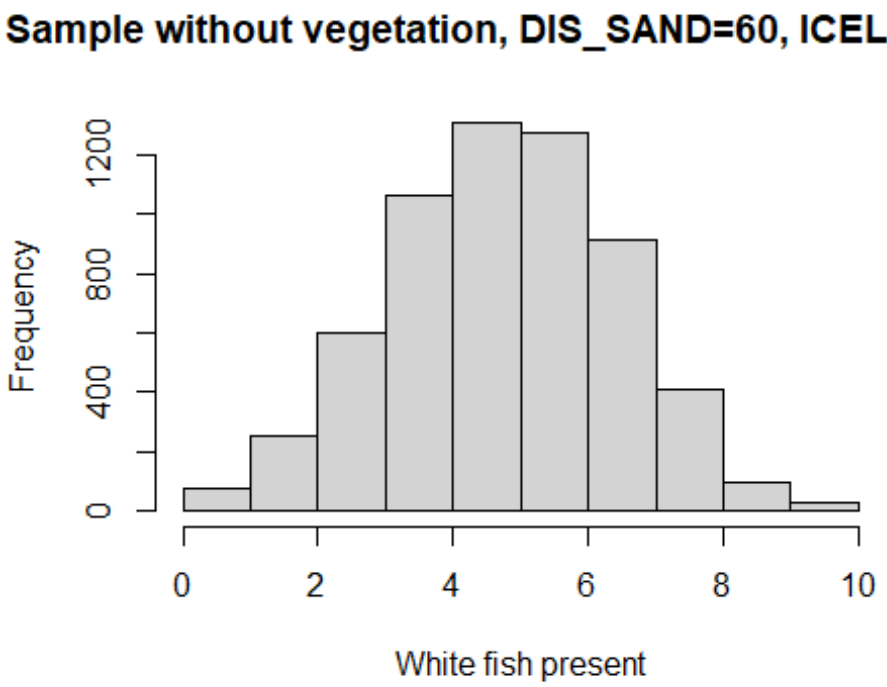
In my own opinion, this last one seems the most compelling model, as here we are also trying to understand the factors that affect the occurence of white fish larvae in different sites, and therefore it is also more informative. By building this model, we can actually make some kind of inference for a new site without calculation, just by looking at those parameters that we have found to be important.

Task 6.

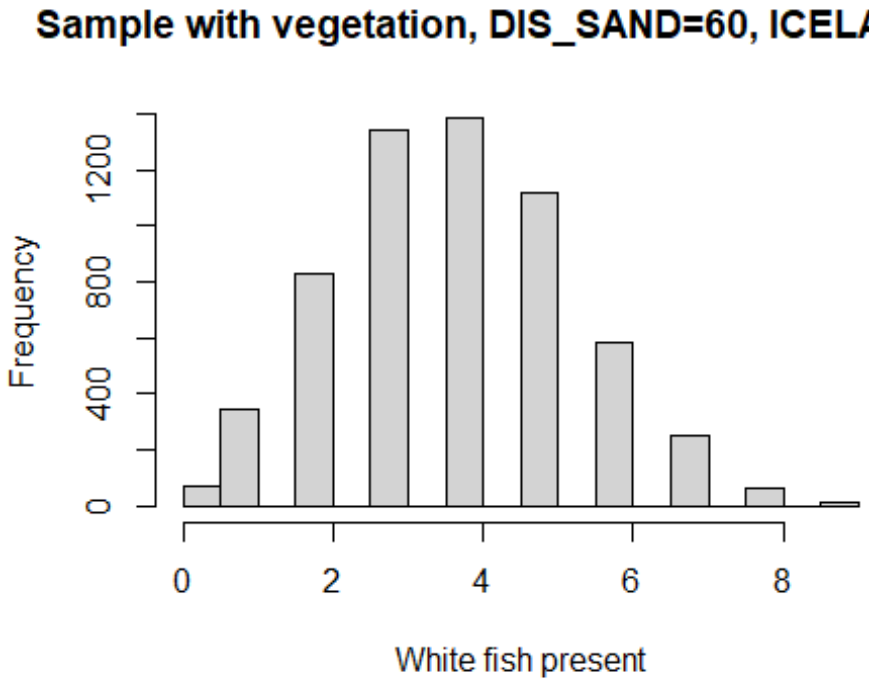
Visualize the posterior distribution of $\tilde{y}$ corresponding to the number sampling occasions where white fish is present out of a total 10 repeated sampling occasions at location where DIS_SAND is 60 and ICELAST is 18 for both vegetated and non-vegetated bottom types.

Drawing random samples:

```
sample_0 <- rbinom(theta0, 10, theta0)
sample_1 <- rbinom(theta1, 10, theta1)
hist(sample_0, main="Sample without vegetation, DIS_SAND=60, ICELAST=18",
xlab="White fish present")
```




```
hist(sample_1, main="Sample with vegetation, DIS_SAND=60, ICELAST=18",
xlab="White fish present")
```



Exercise 2

In the original Mauna Loa CO_2 data analysis we visualized the posterior predictive distribution of the expected CO_2 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_2 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 $\tilde{y} = [\tilde{y}_1, \dots, \tilde{y}_n]$ values from $\tilde{y}_i \sim N(\mu_i, \sigma^2)$, where $\mu_i = a + bx_i$ and a, b, σ^2 are drawn from the posterior. For example, pick 20 random triplets of a, b, σ^2 from the Markov chain and for each of them sample $\tilde{y}$. Then plot histogram of each $\tilde{y}$.\

Next, revise the model so that $\mu_i = a + bx_i + cx_i^2$. 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 $\dot{c}$ back to c even if you standardize your y and x.

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

$$\text{lpd} = \sum_{i=1}^{n_{\text{test}}} \log p\left(\tilde{y}_i | \tilde{x}_i, y_{\text{training}}, x_{\text{training}}\right)$$

and the root mean squared error

$$\text{RMSE} = \sqrt{\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \left(E\left[\tilde{y}_i | \tilde{x}_i, y_{\text{training}}, x_{\text{training}}\right] - \tilde{y}_i\right)^2}$$

where $y_{\text{training}}, x_{\text{training}}$ are the training and test data and $\tilde{y}_i, \tilde{x}_i$ 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 $\theta^{(s)}$ is a sample from the posterior distribution of the parameters ($\theta = \{a, b, \sigma^2\}$ in the original model), that is $\theta^{(s)} \sim p\left(\theta | y_{\text{training}}, x_{\text{training}}\right)$.

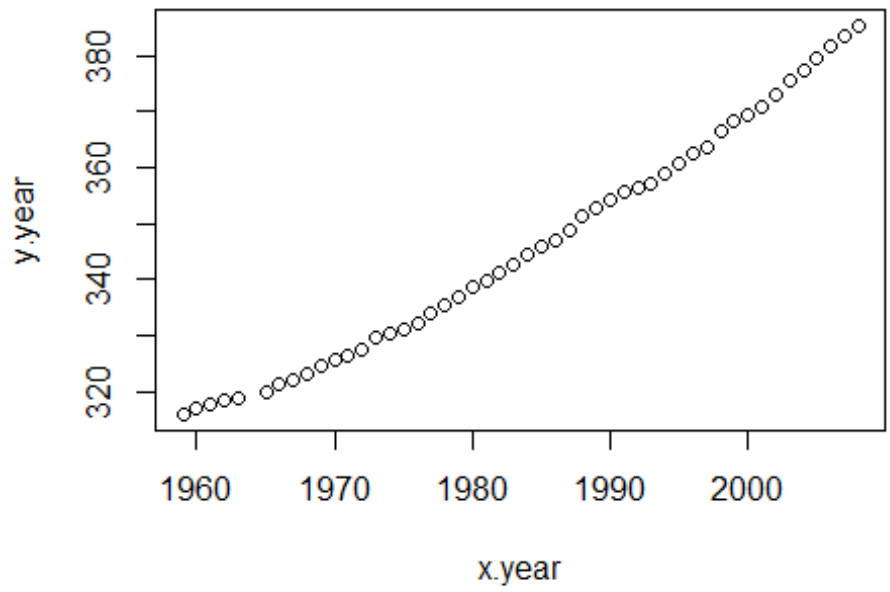
Each of the above four tasks provides 5 points from correct implementation and answer. Each task gives 2 points if it is done towards right direction and partially correct.

Load the needed libraries.

```
library(ggplot2)
library(StanHeaders)
library(rstan)
set.seed(123)

options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
```

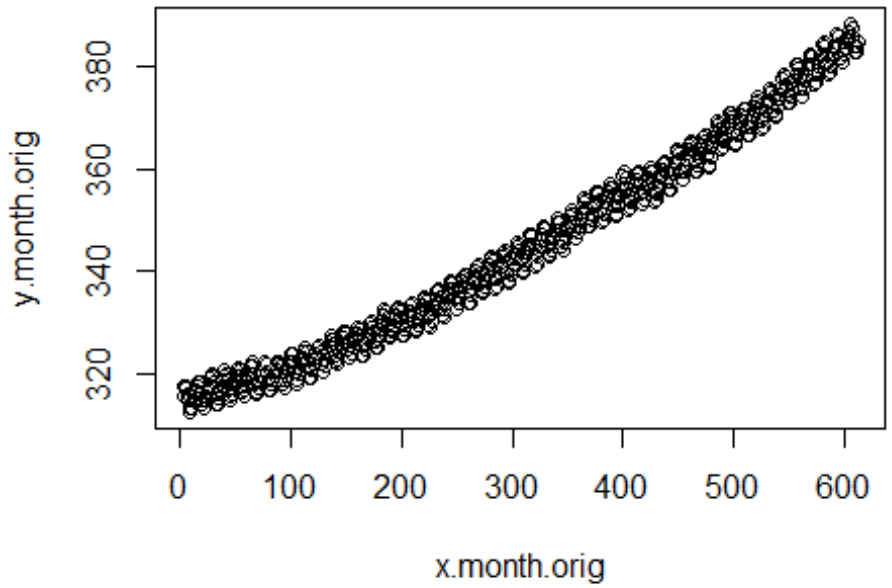
Load the data and explore its properties

```
# 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)
```

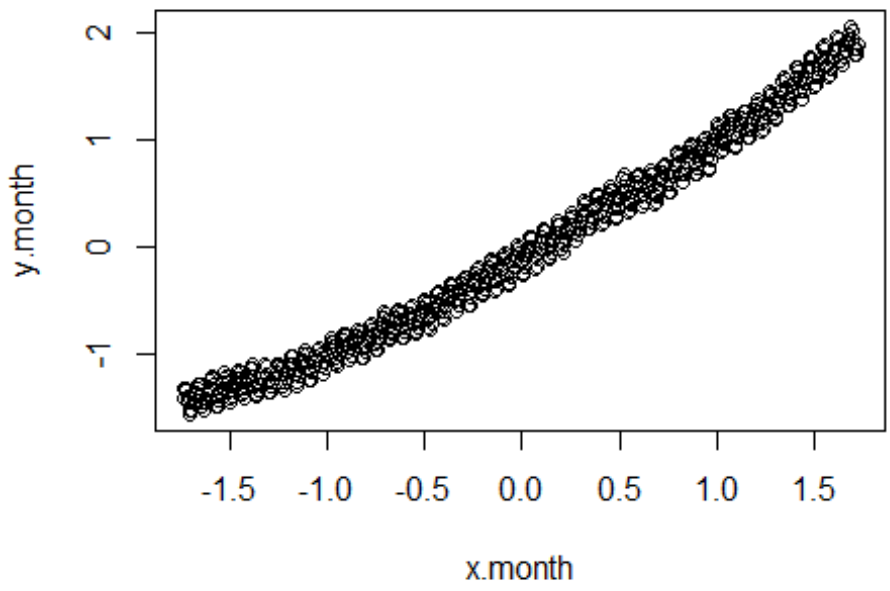



```
# Let's take the monthy values and construct a "running month" vector
y.month.orig = as.vector(t(maunaloa.dat[,2:13]))
```

```r
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)
```




```r
# 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)
```

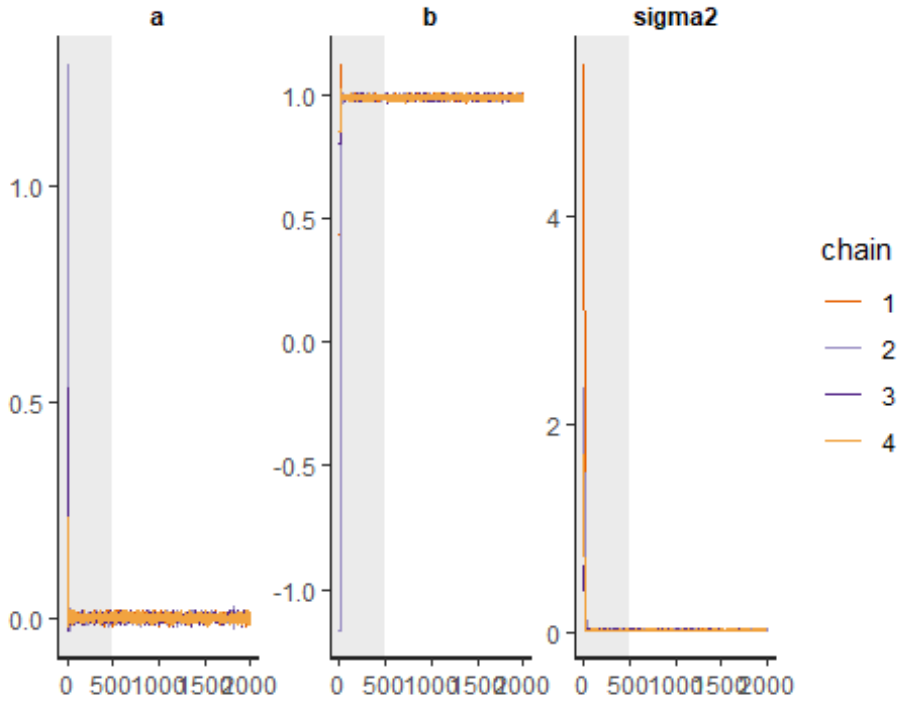
Posterior predictive check

Analysis with the original model

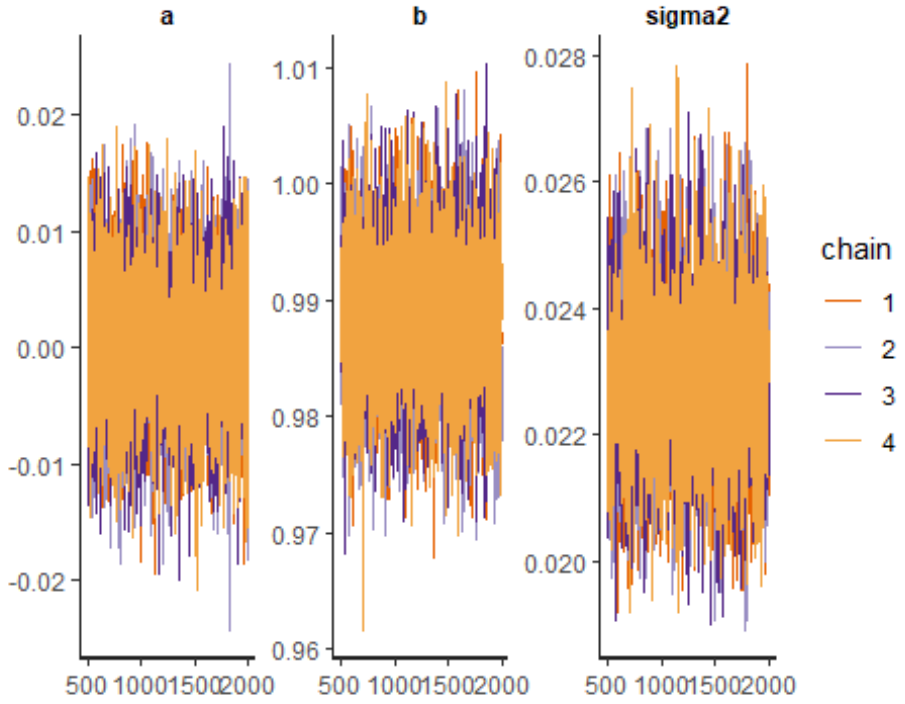
$$y_i = a + bx_i + \epsilon_i$$

```
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,chain
s=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 18:20:29 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)
```

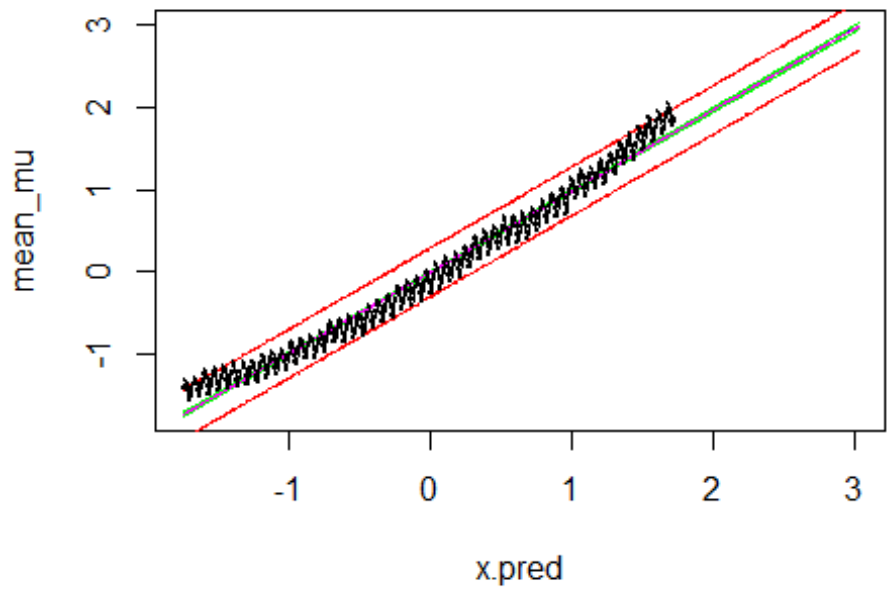



```
plot(post, pars=c("a","b","sigma2"), plotfun= "trace", inc_warmup = FALSE)
```



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

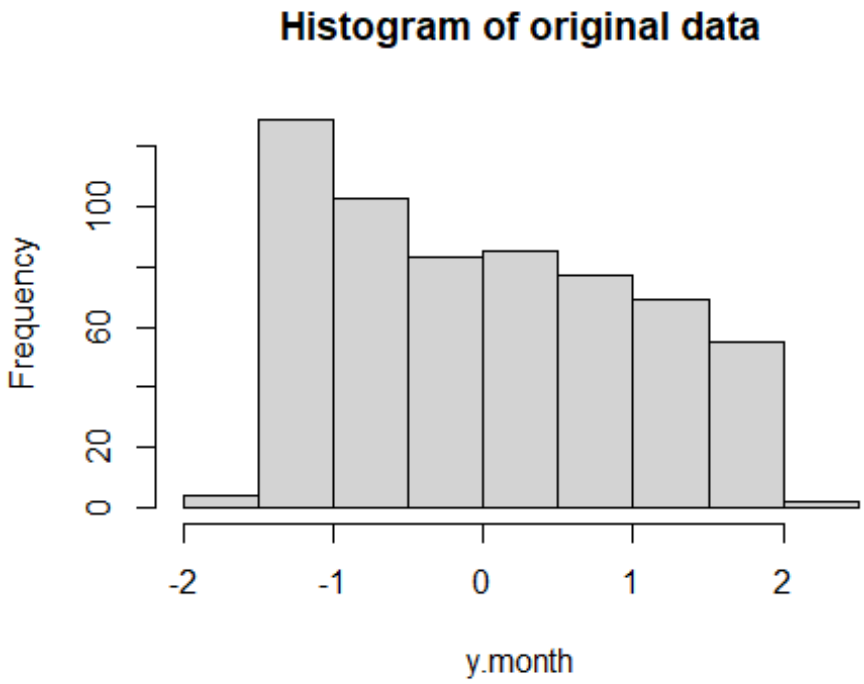
```r
set.seed(123)
# extract the samples into matrix
post_sample <- as.matrix(post, pars =c("a","b","sigma2"))    # combine all
chains into one matrix in R workspace
a_dot=post_sample[,1]
b_dot=post_sample[,2]
sigma2_dot=post_sample[,3]


# 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)
```



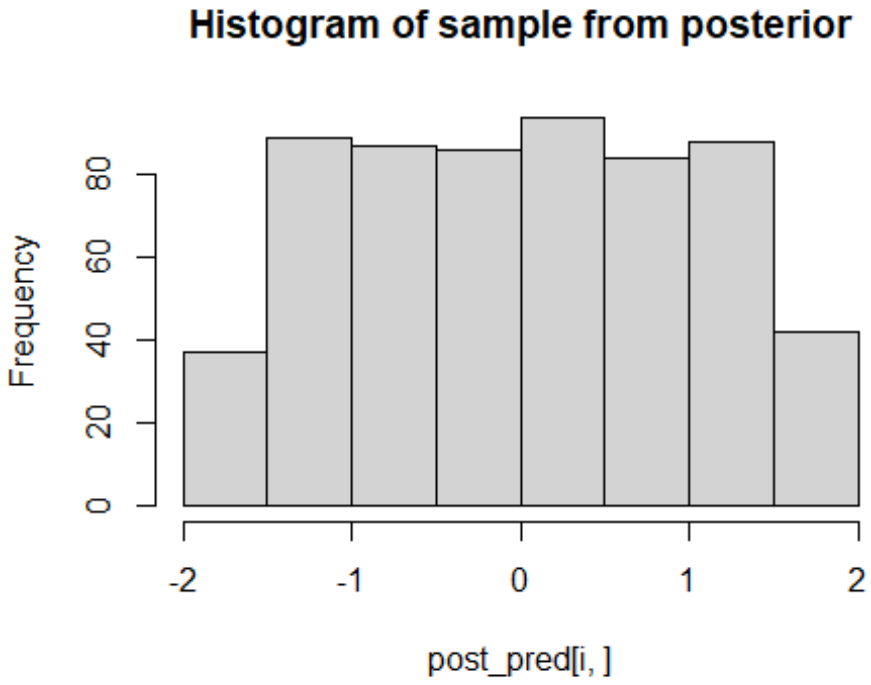
Now we are ready to conduct the posterior predictive check

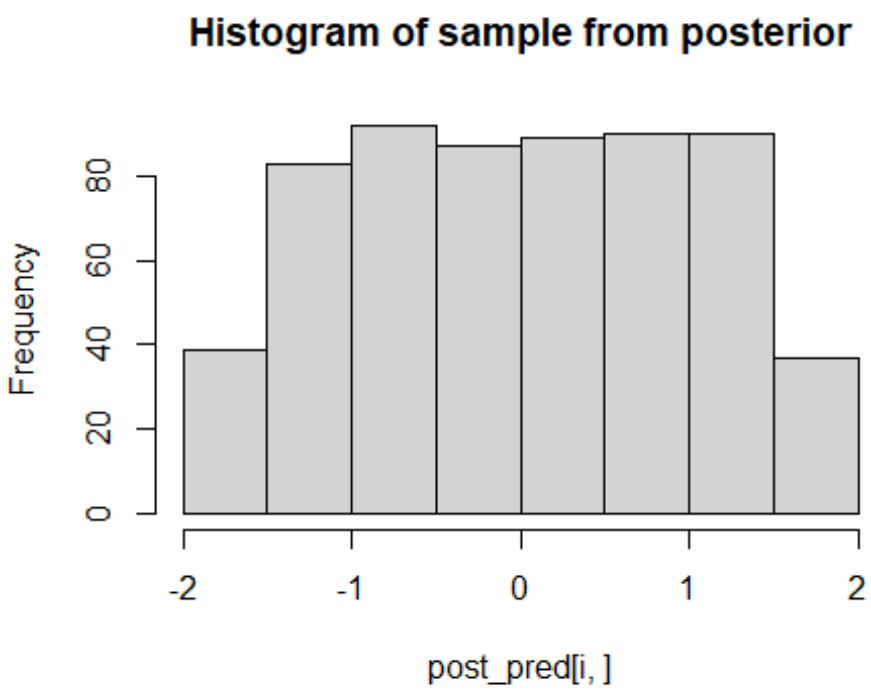
First, sample 20 random draws from the Markov chain

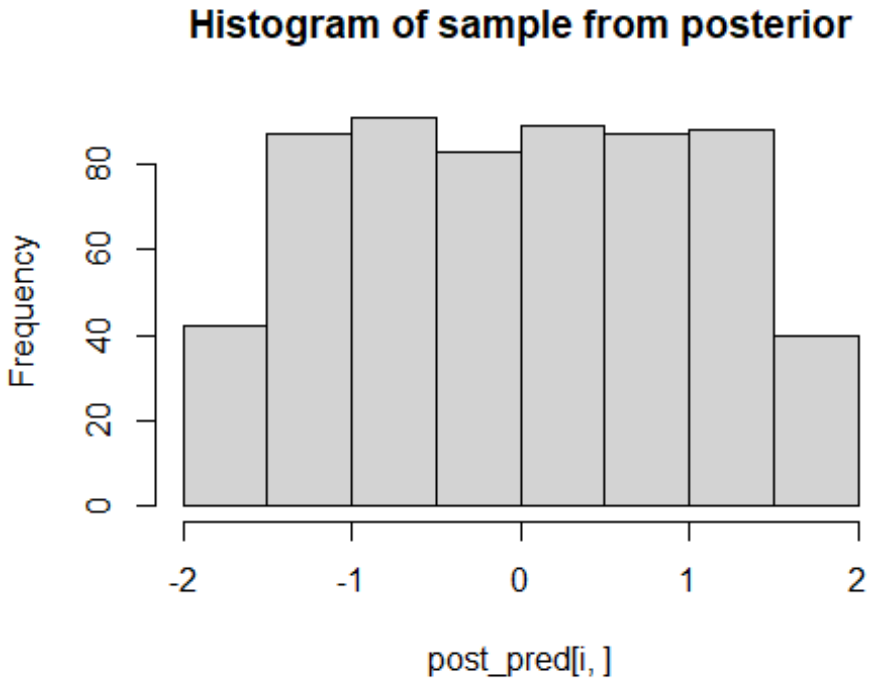
```
set.seed(123)

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")
```

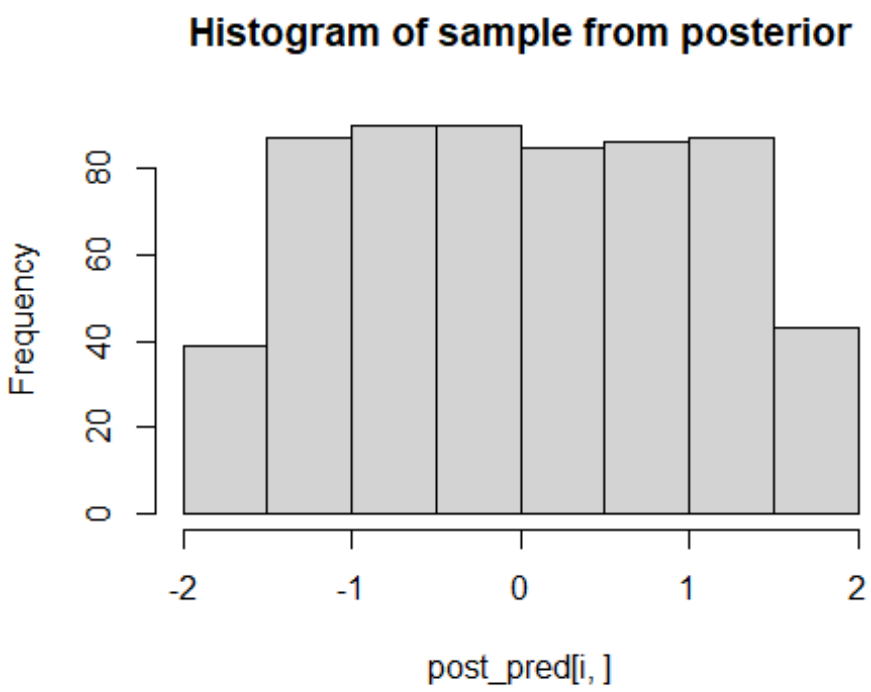


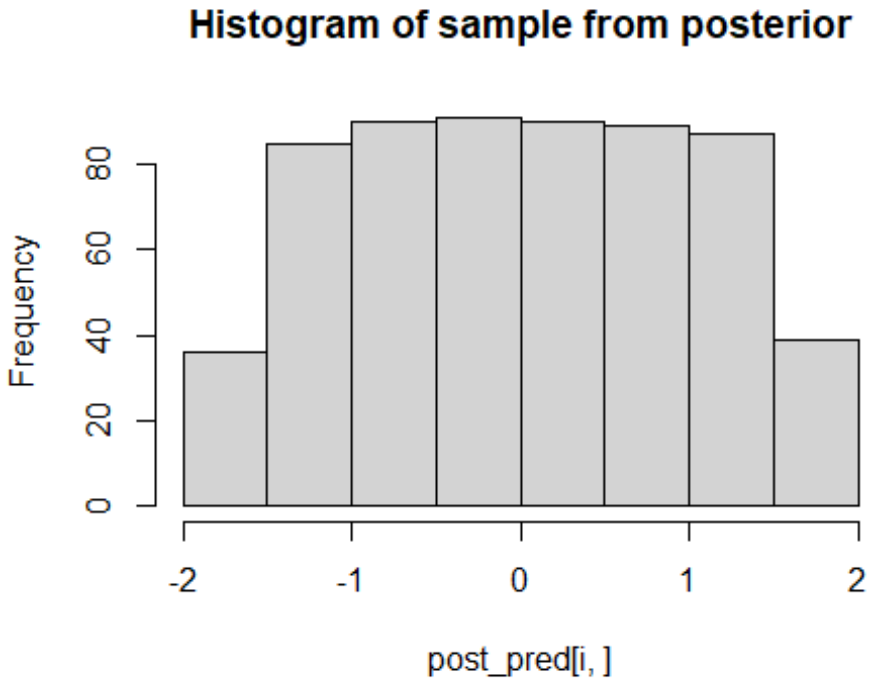

```
for (i in 1:20){hist(post_pred[i,], main="Histogram of sample from
posterior")}
```

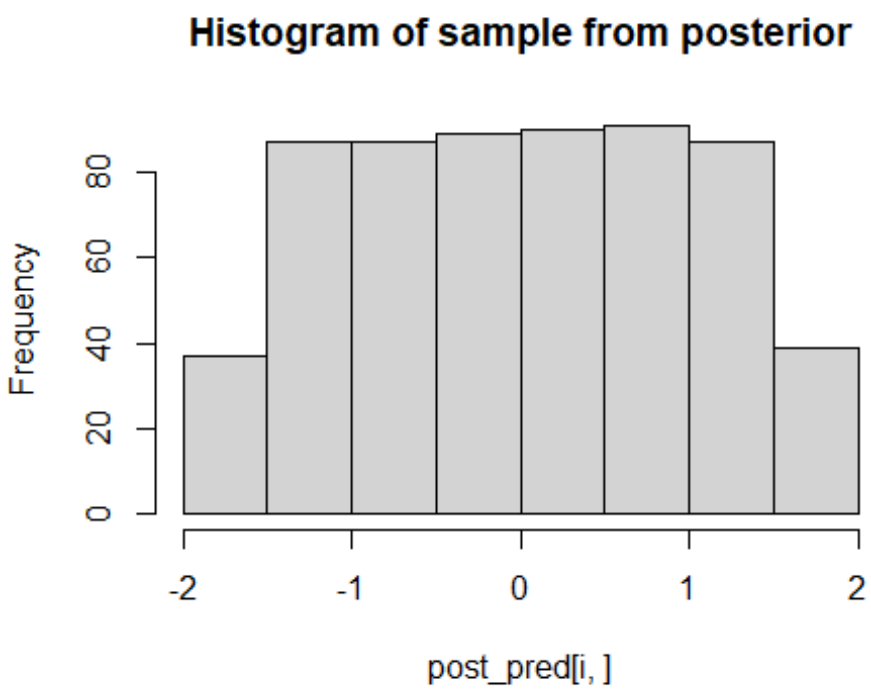

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

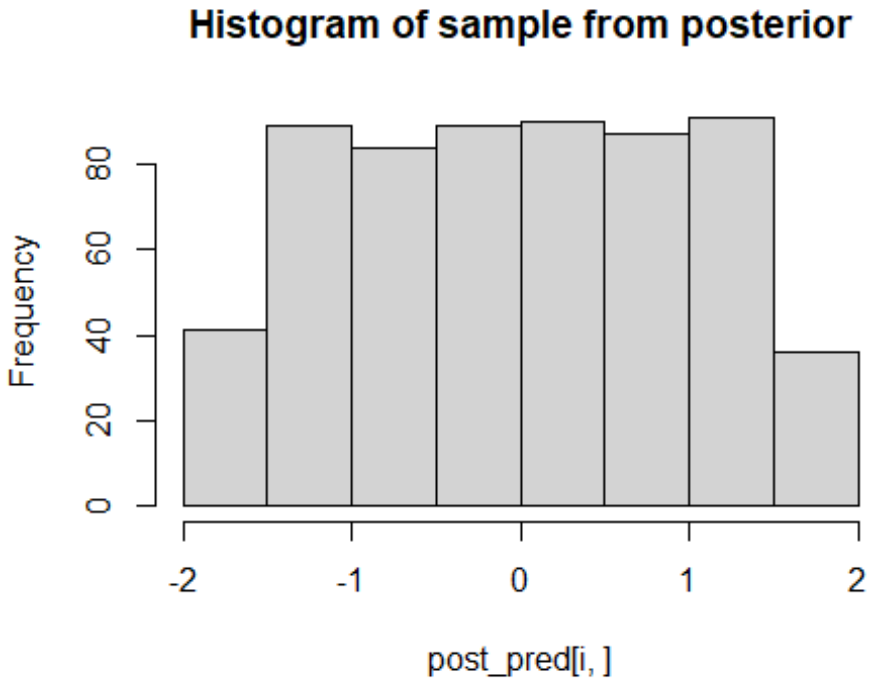

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

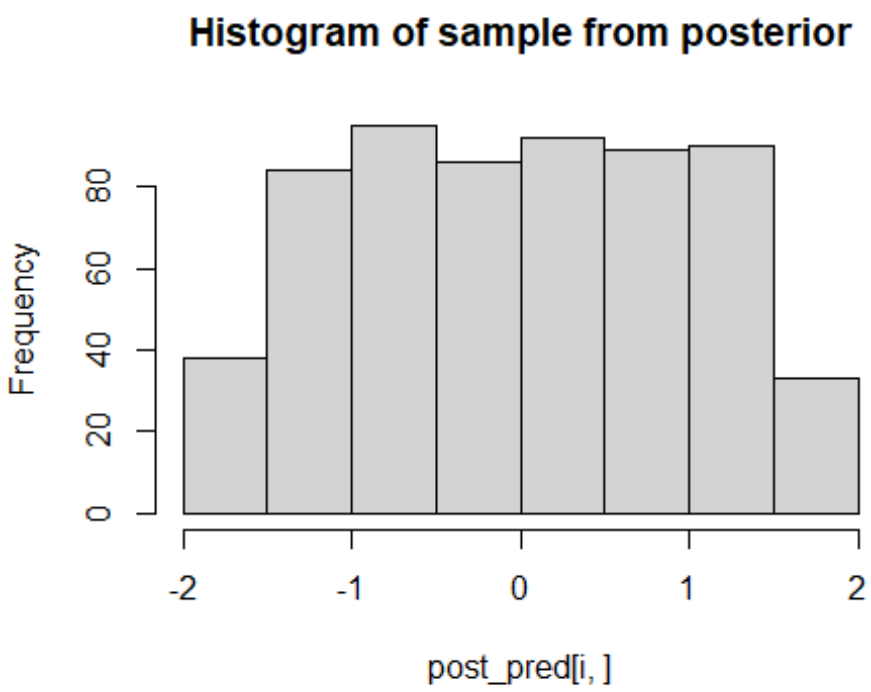

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

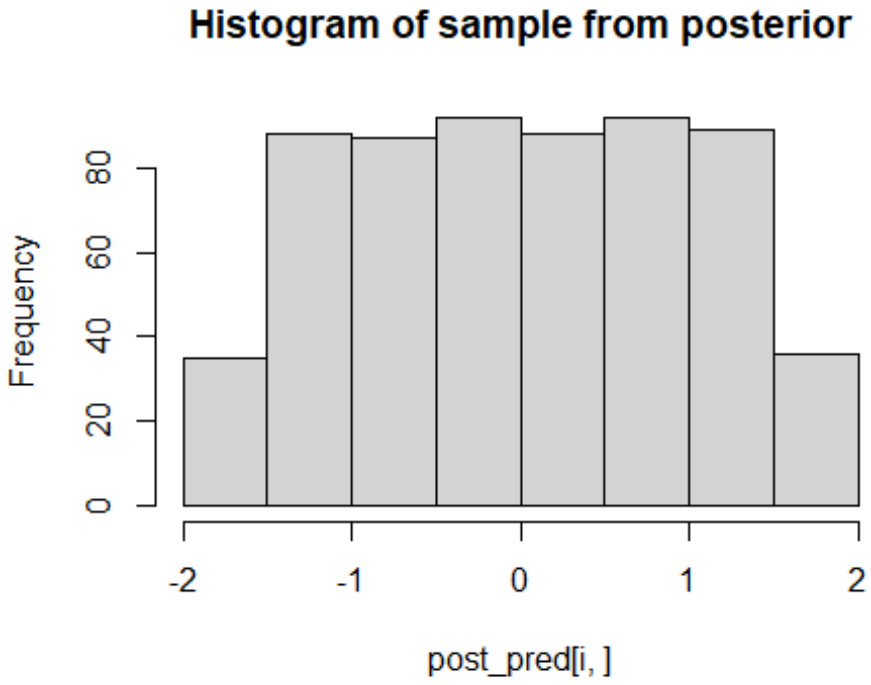

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

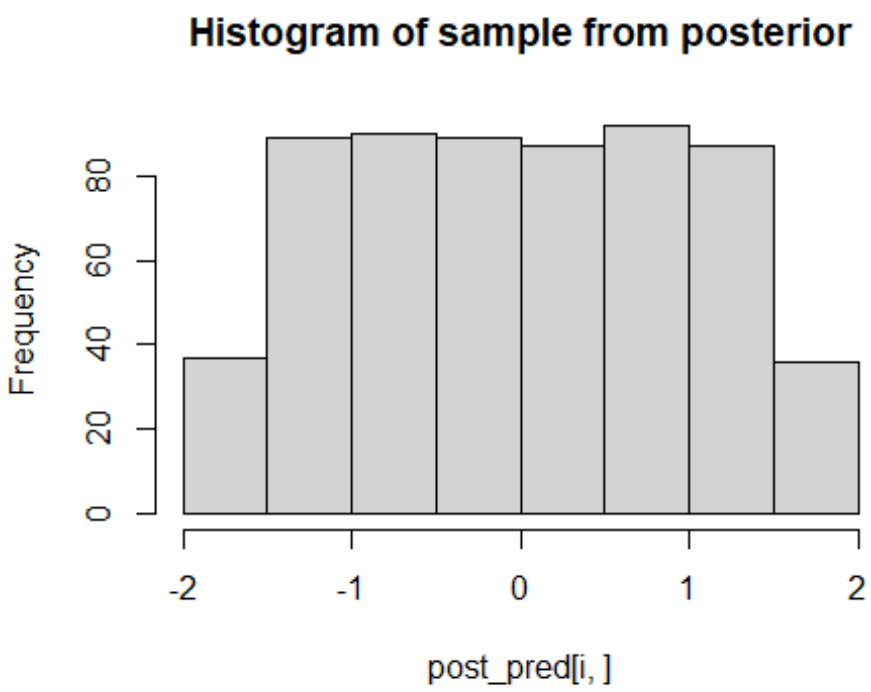

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

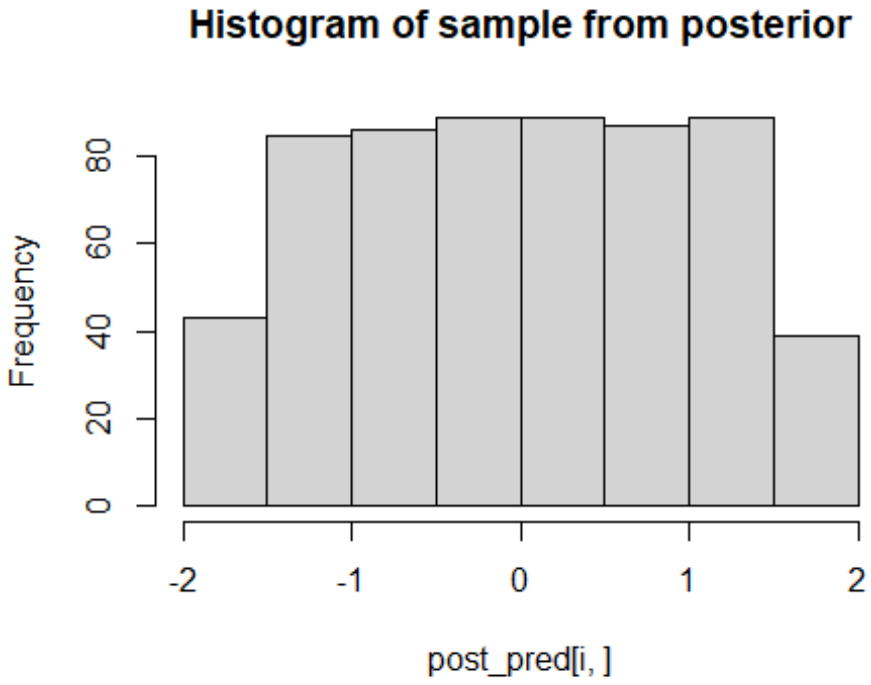

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

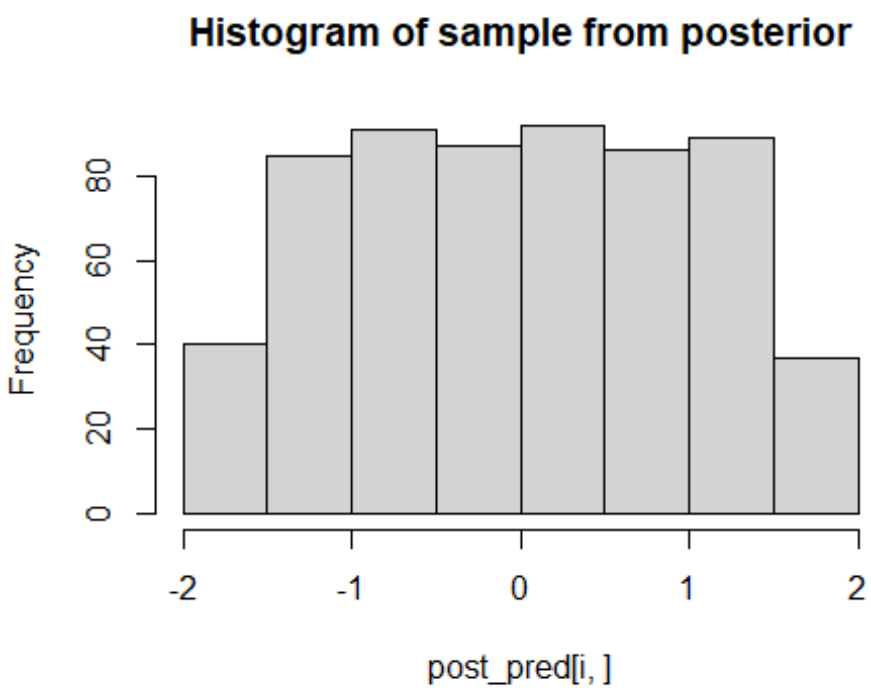

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

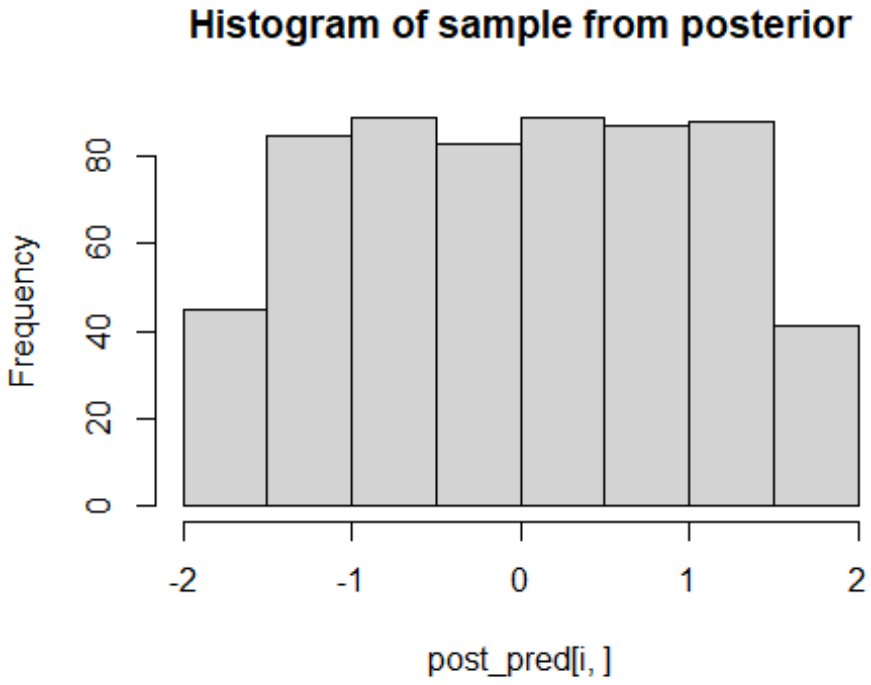

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

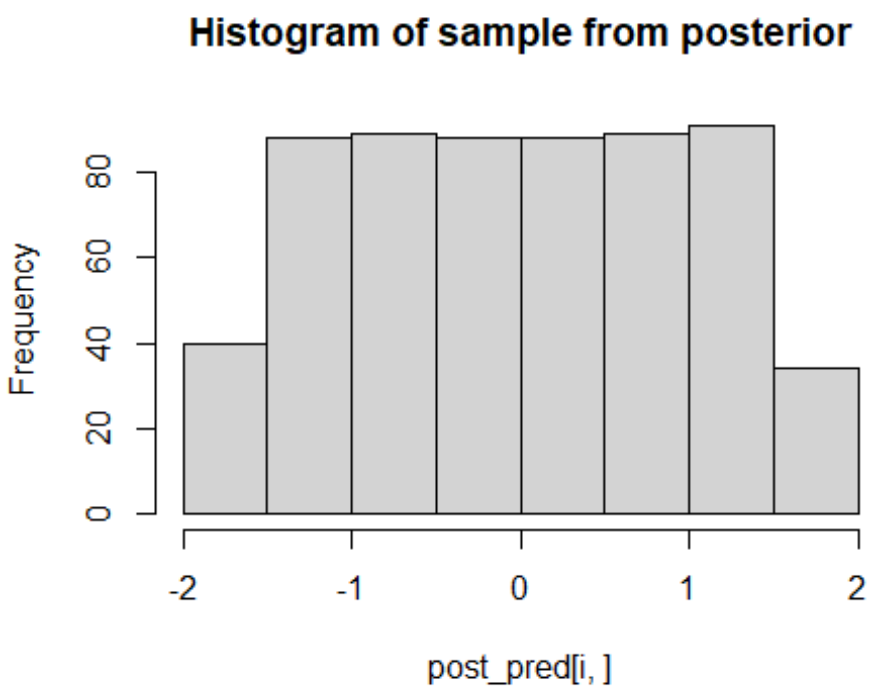

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

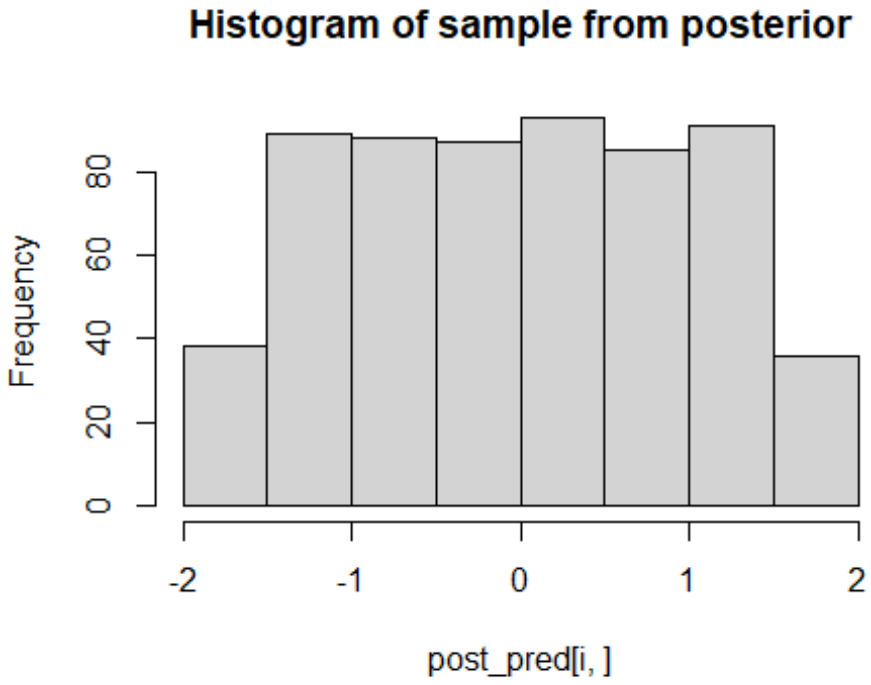

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

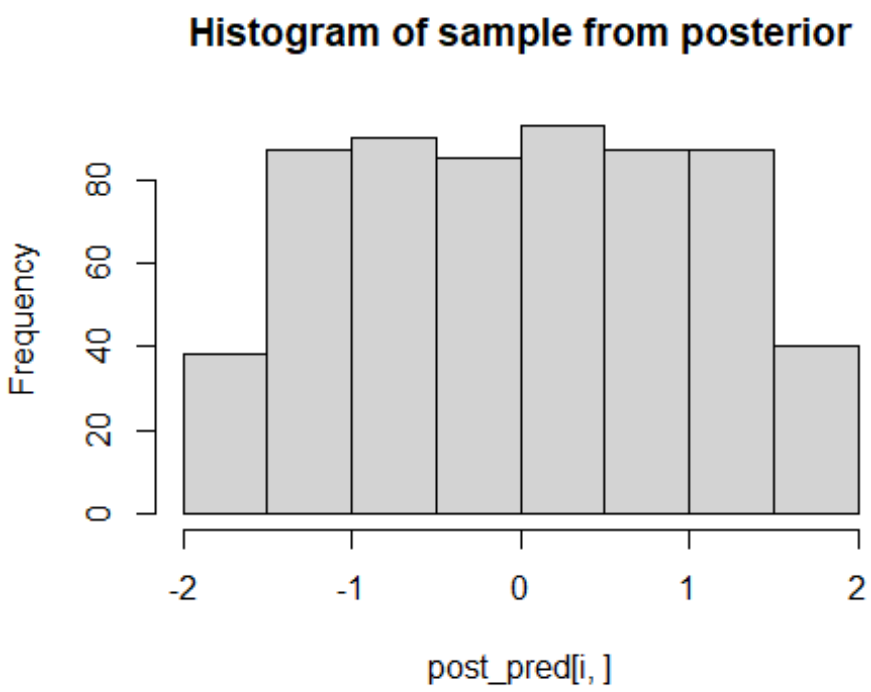

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

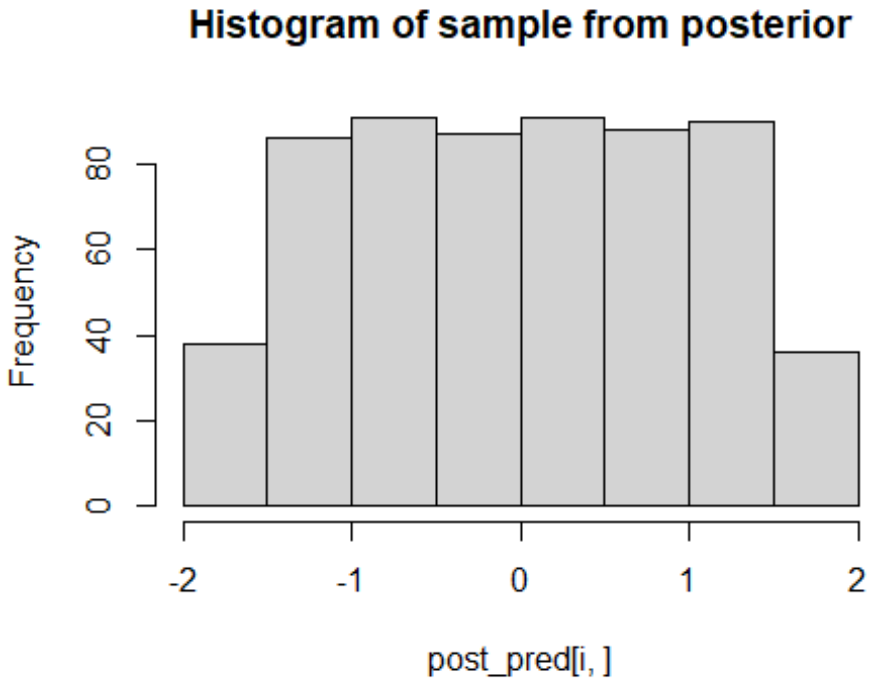

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

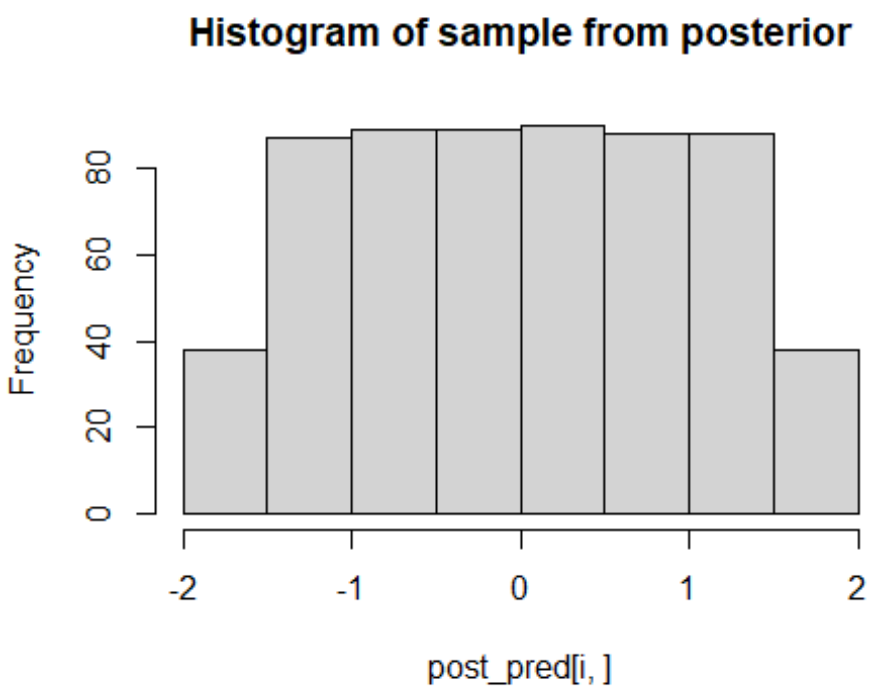

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

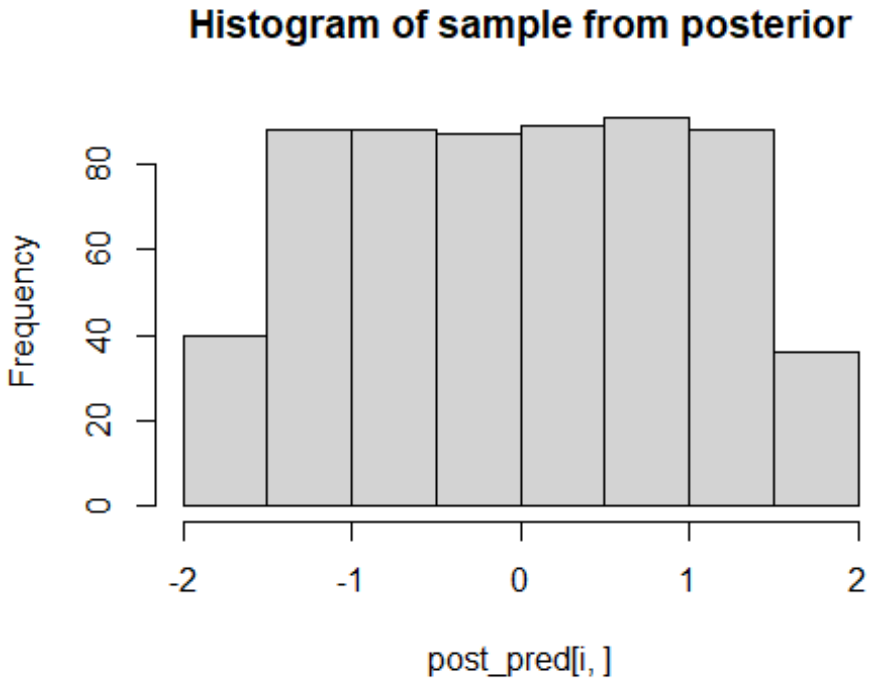

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]

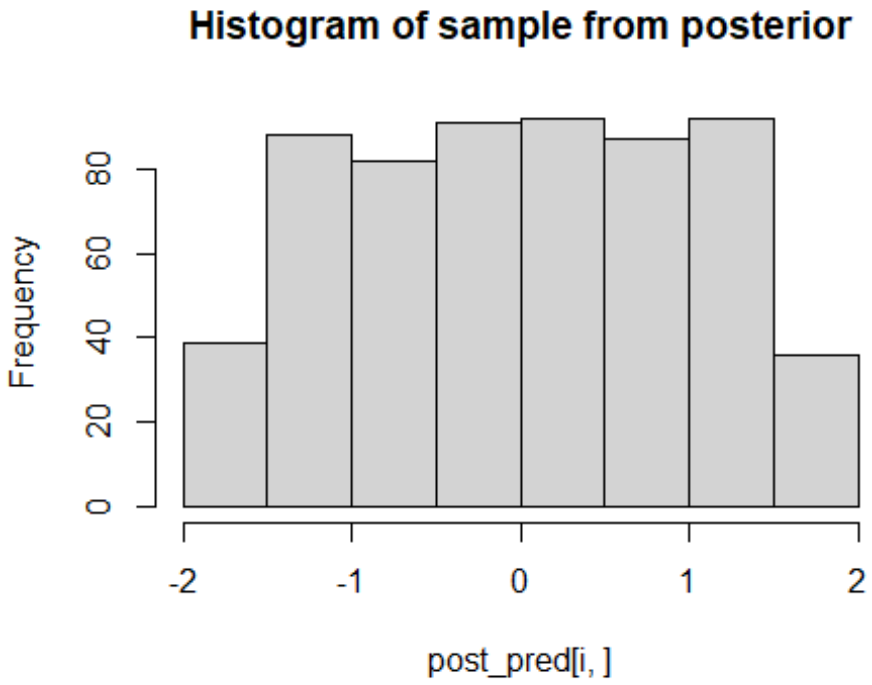

Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]


Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]


Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]


Histogram of sample from posterior
Frequency
0
20
40
60
80
-2
-1
0
1
2
post_pred[i,]





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

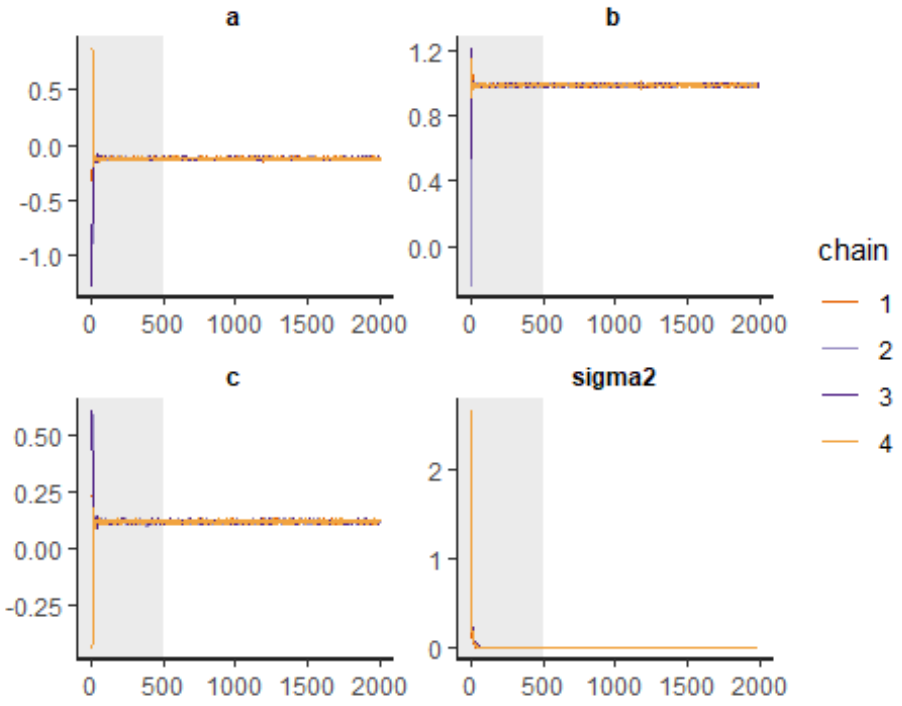
$$mu = a + b * x + c * x^2$$

and rerun the analysis

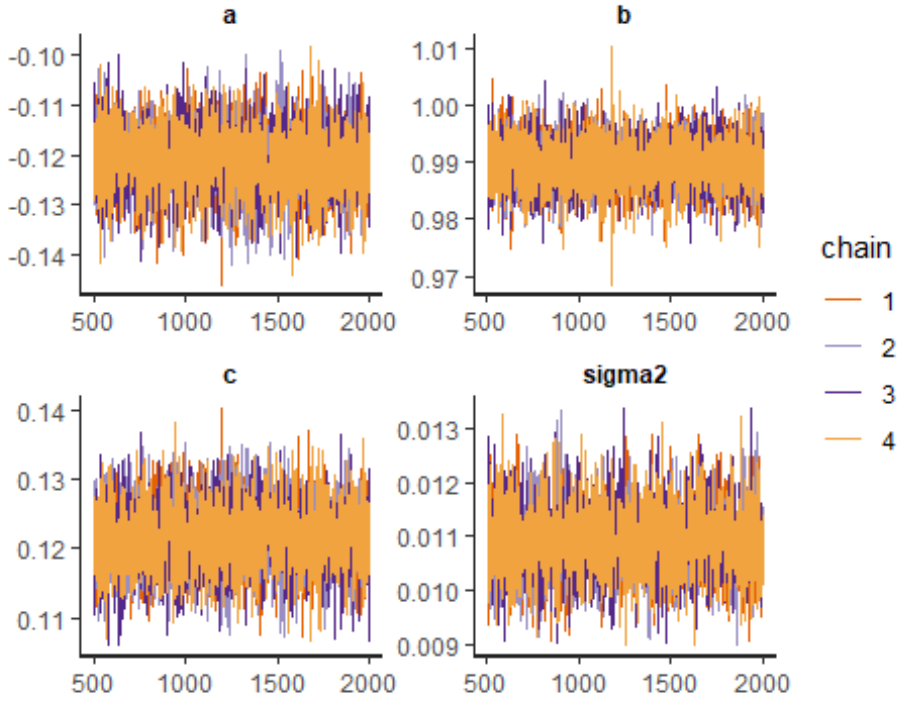
```
mauna_loa_c02_model2 = "
data{
  int<lower=0> N; // number of observations
  real y[N];     // observed CO2 values
  real x[N];    // observed times
}
parameters{
  real a;
  real b;
  real c;
  real<lower=0> sigma2;
}
transformed parameters{
  real<lower=0> sigma;
  real mu[N];

  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,cha
ins=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 18:22:28 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)
```

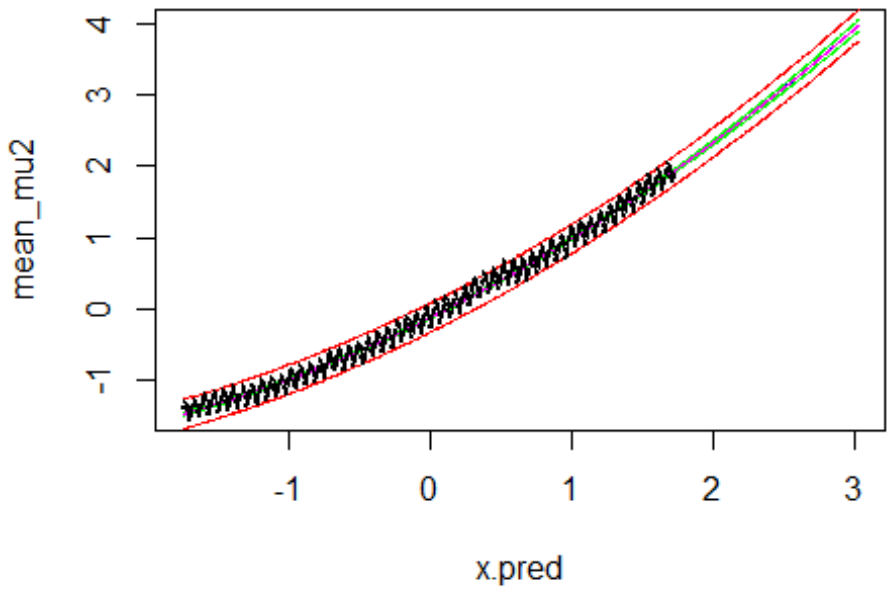


The Rhat-values of 1 and the convergence and autocorrelation seem good.

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)
```



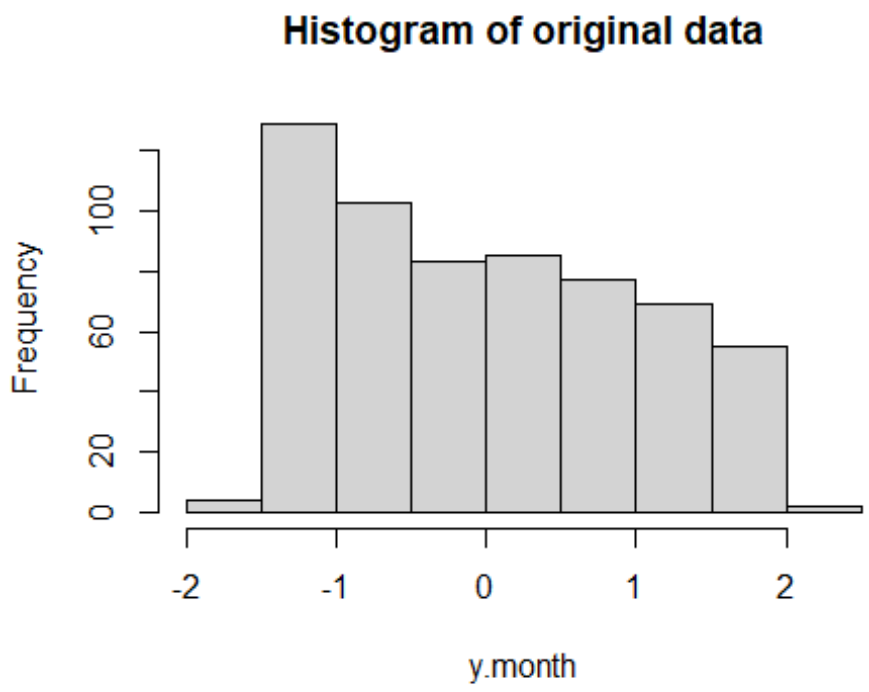
Analysis of linear graph

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.

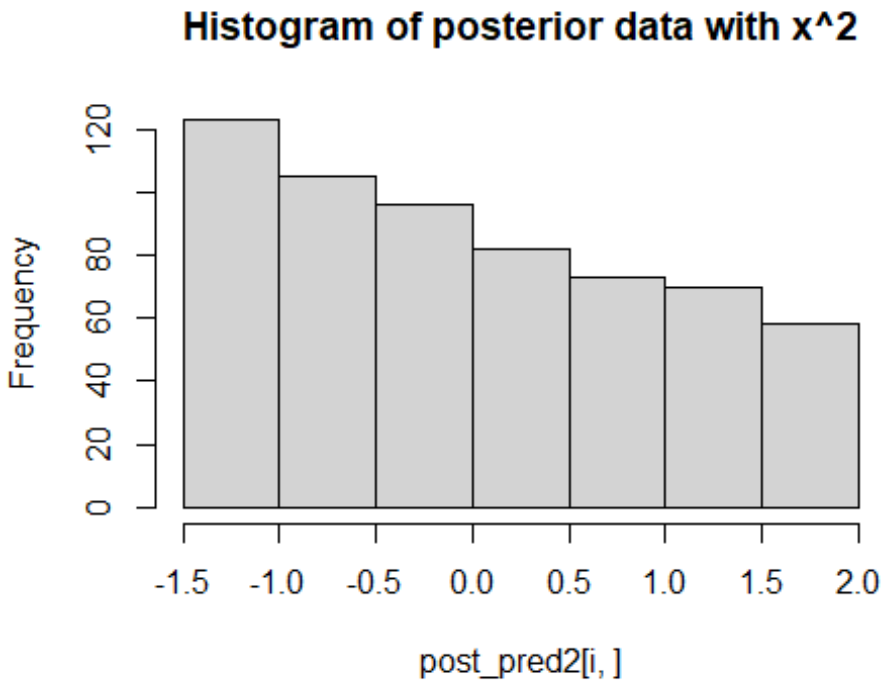
And then we can do the asked posterior predictive check

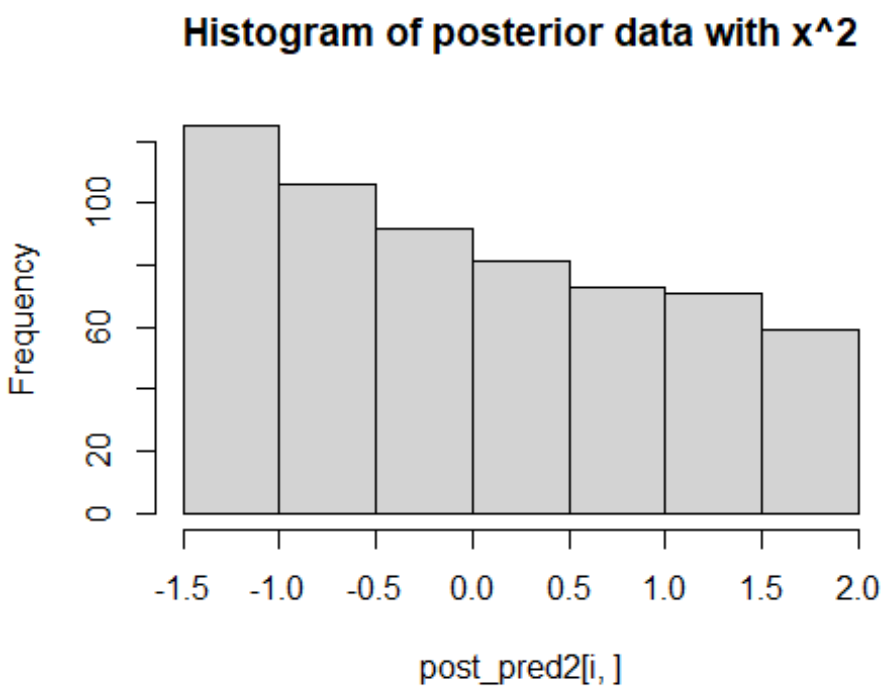
```
pred_sample2 <-
post_sample2[sample(nrow(post_sample2),size=20,replace=TRUE),]
```

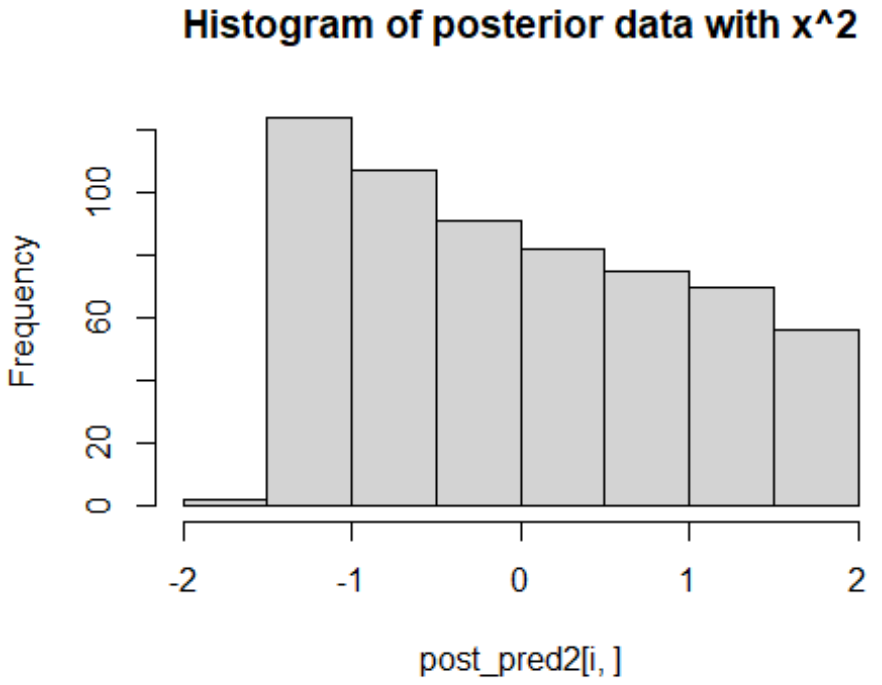
```
post_pred2 <- array(dim=c(20, length(x.month)))
for (i in 1:20){
  for (j in 1:length(x.month)){
    post_pred2[i,j] <- rnorm(1, (pred_sample2[i, 'a'] + pred_sample2[i, 'b']
* x.month[j] + pred_sample2[i, 'c'] * (x.month[j]^2)), pred_sample[i,
'sigma2'])
  }
}
#for (i in 1:20){plot(x.month, post_pred2[i,])}

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

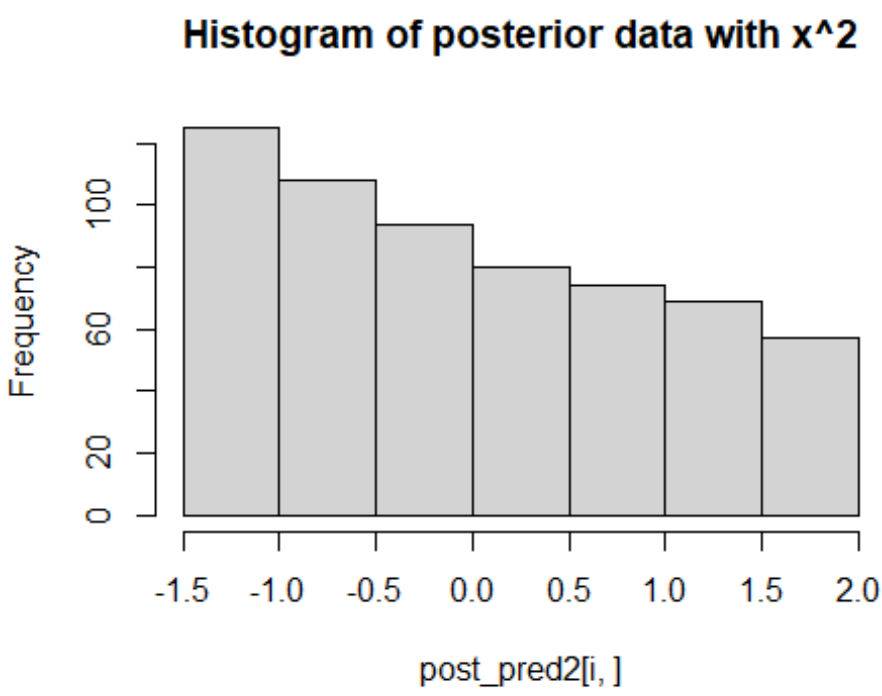


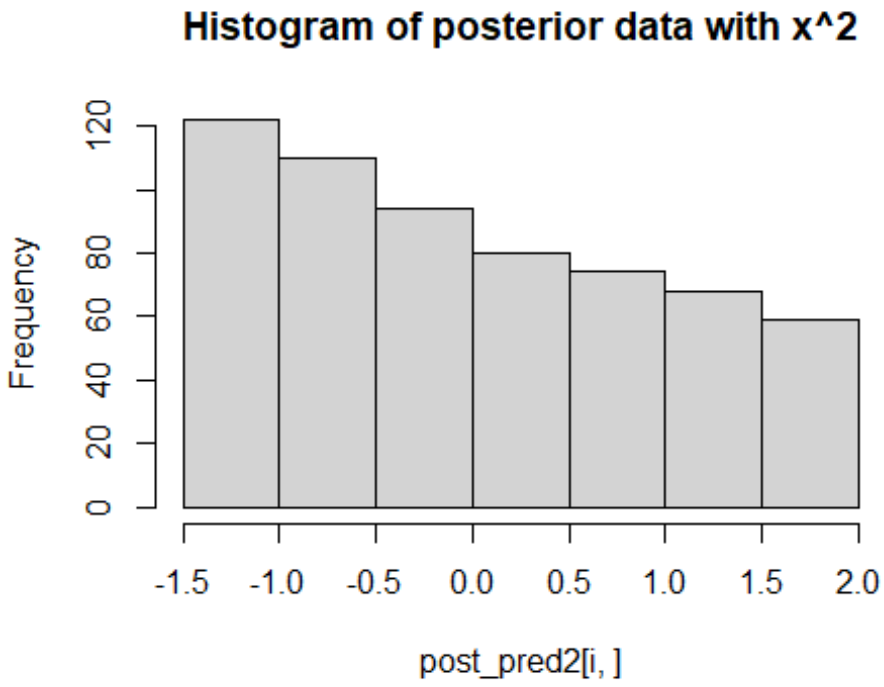

```
for (i in 1:20){hist(post_pred2[i,], main="Histogram of posterior data with
x^2")}
```

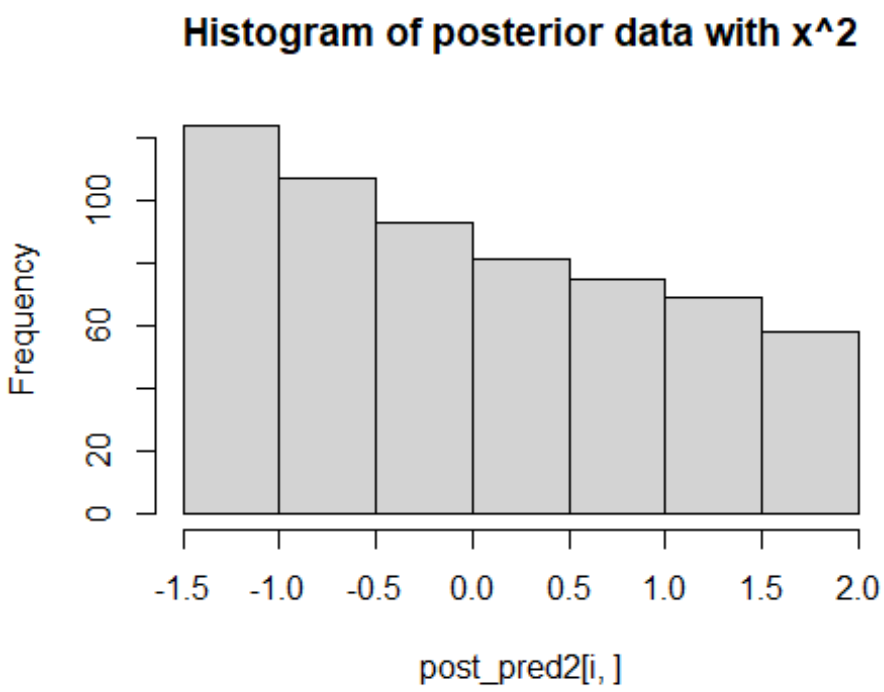

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]

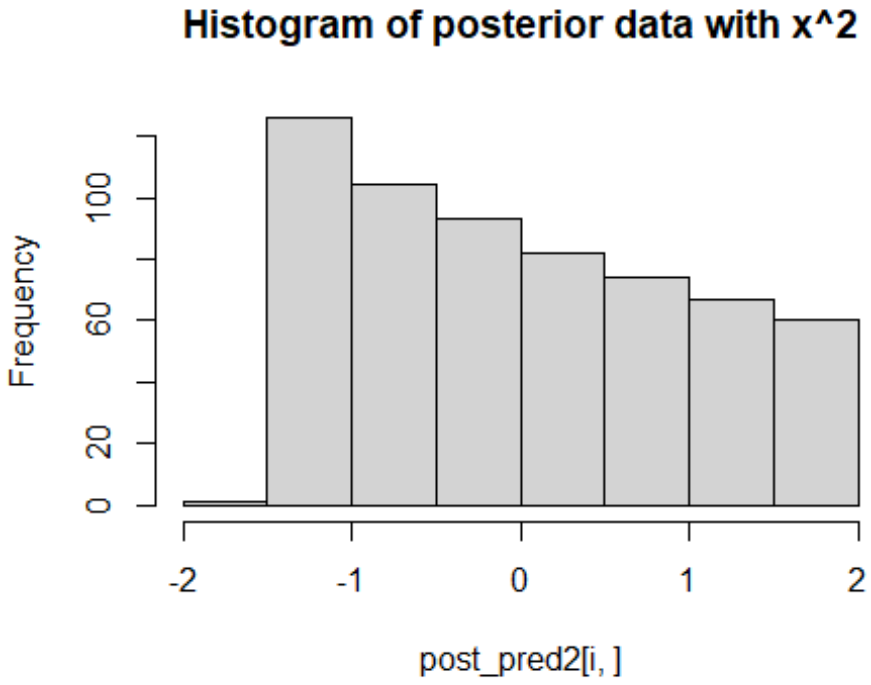

Histogram of posterior data with x^2
Frequency
0
20
60
100
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]

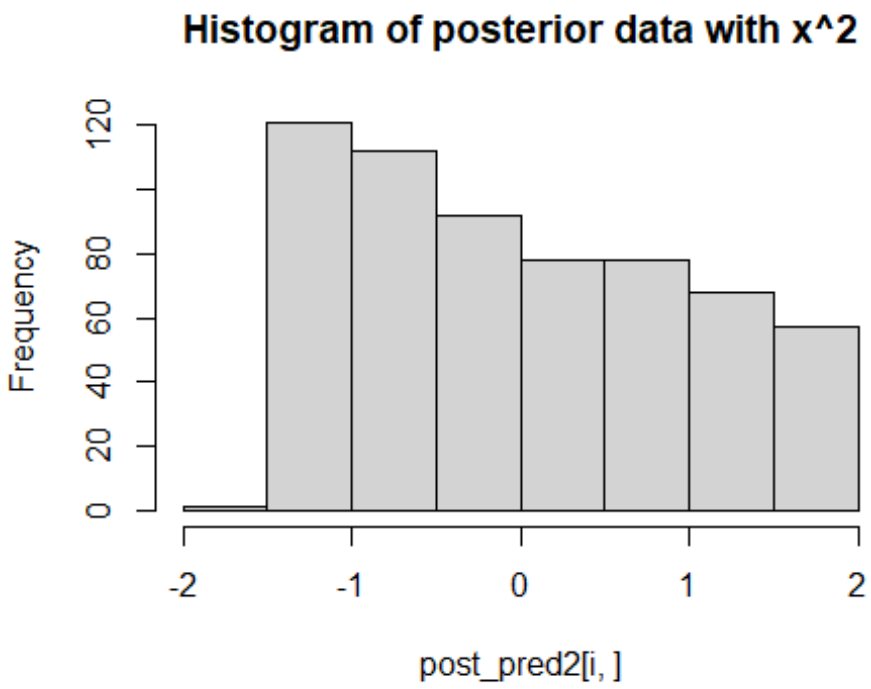

Histogram of posterior data with x^2
Frequency
0
20
60
100
-2
-1
0
1
2
post_pred2[i,]

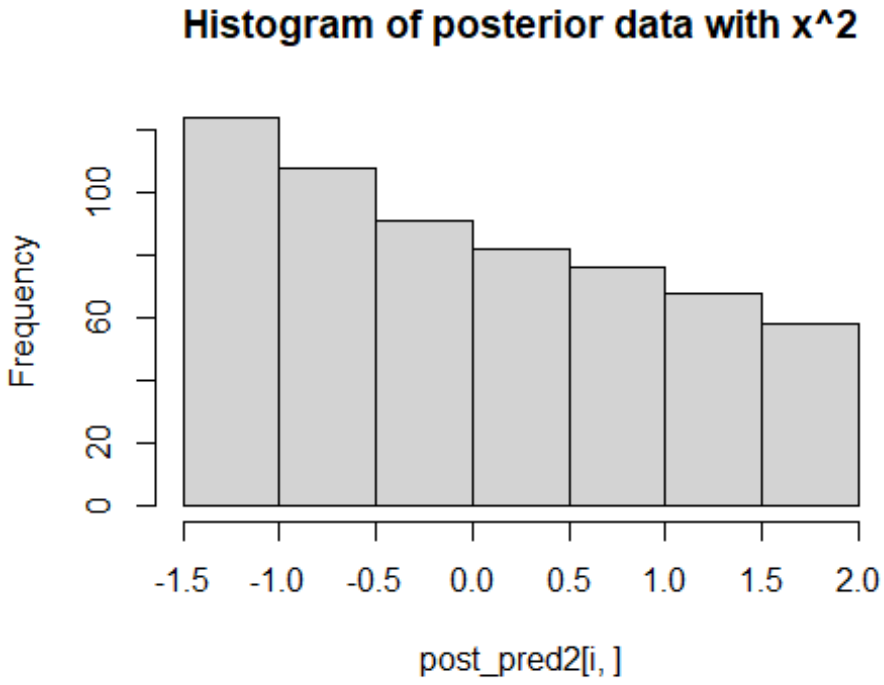

Histogram of posterior data with x^2
Frequency
0
20
60
100
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]

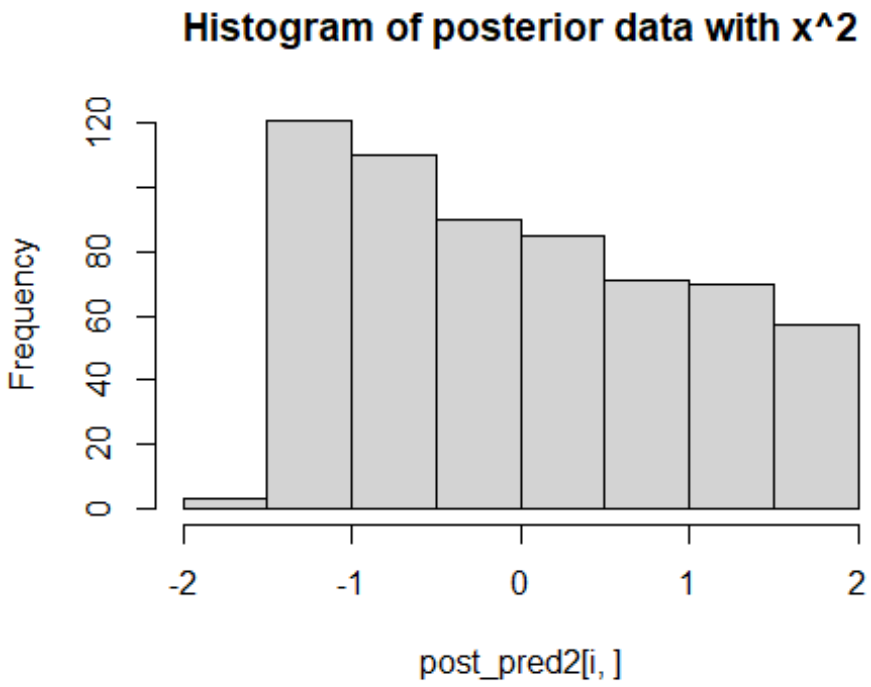

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]

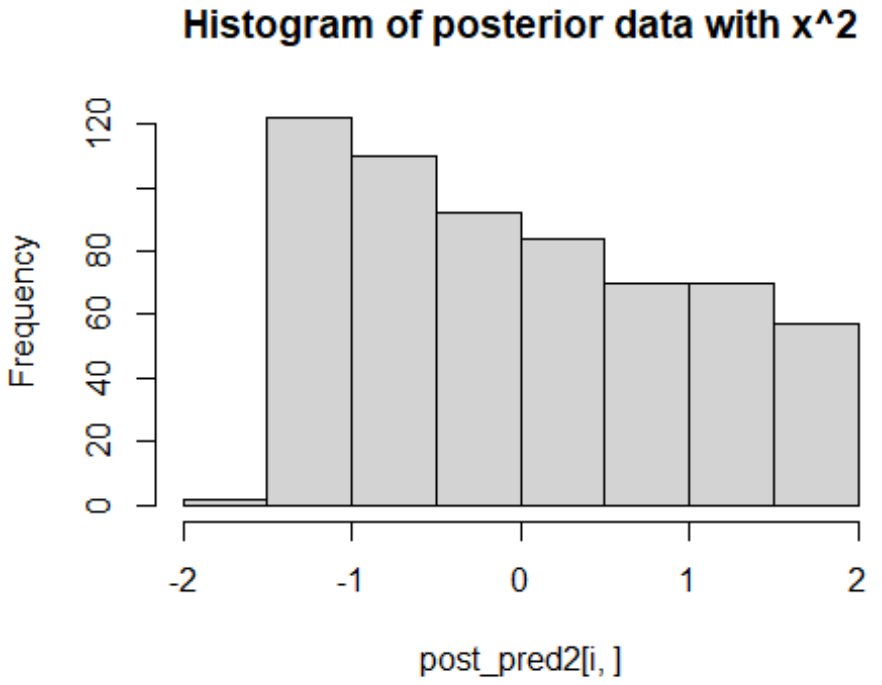

Histogram of posterior data with x^2
Frequency
0
20
60
100
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]

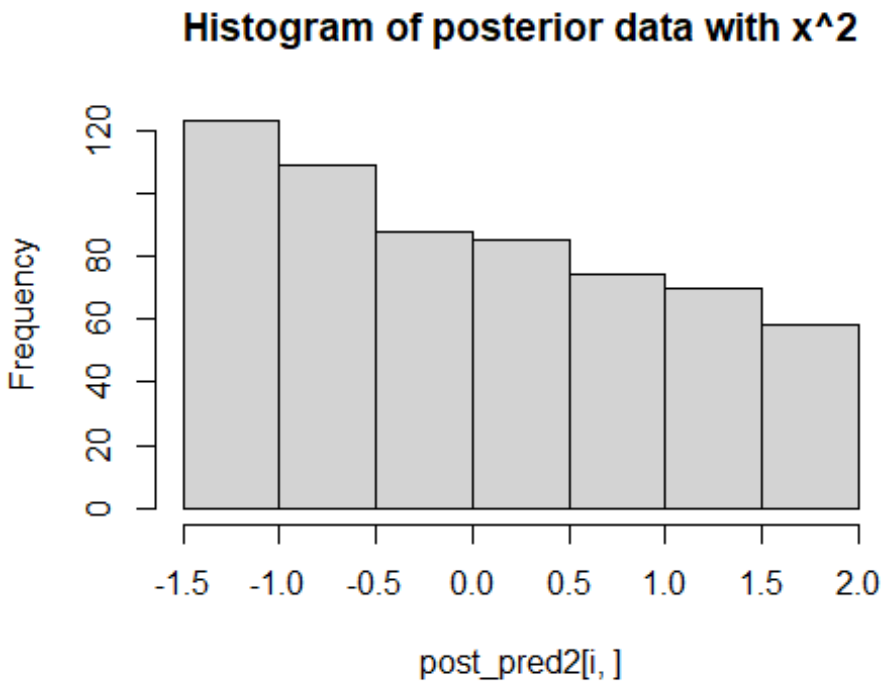

Histogram of posterior data with x^2
Frequency
0
20
60
100
-2
-1
0
1
2
post_pred2[i,]

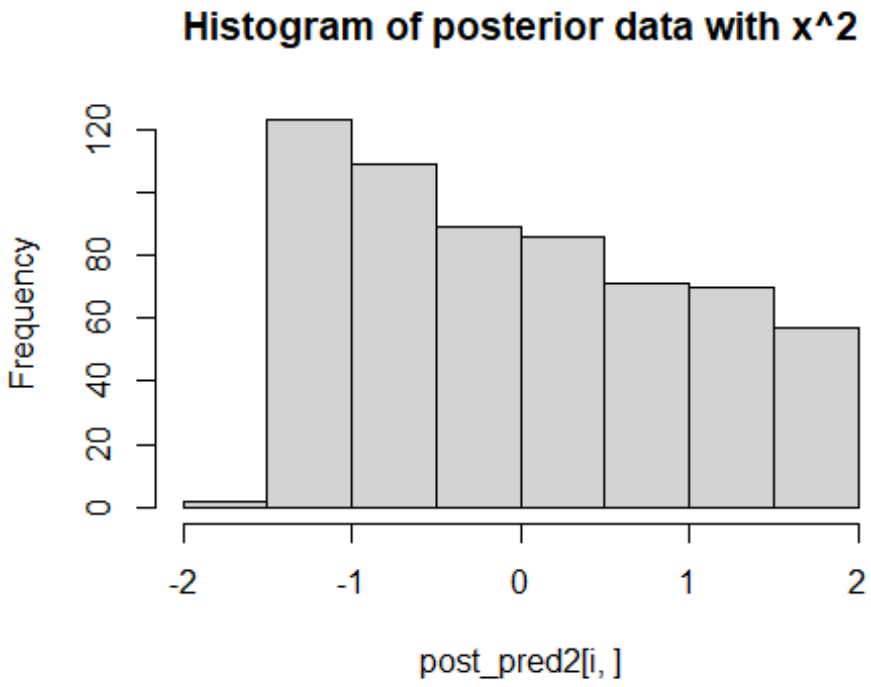

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-2
-1
0
1
2
post_pred2[i,]

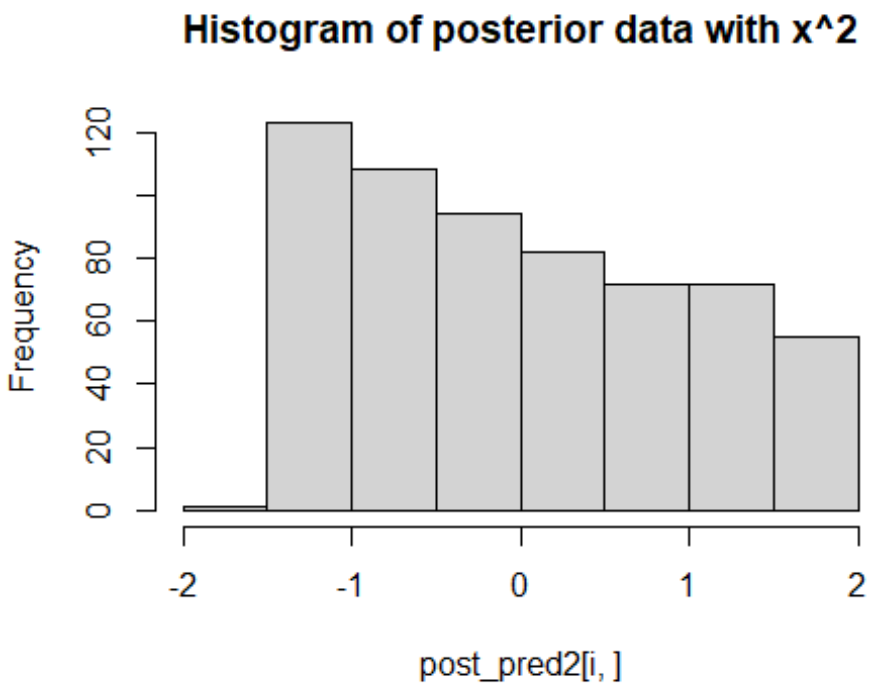

Histogram of posterior data with x^2
Frequency
0
20
60
100
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]

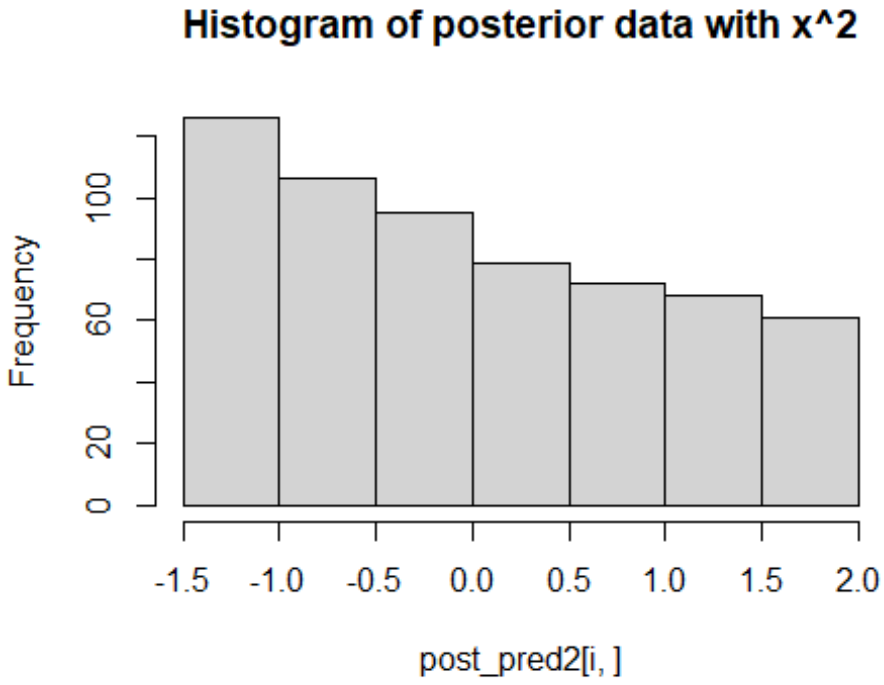

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-2
-1
0
1
2
post_pred2[i,]

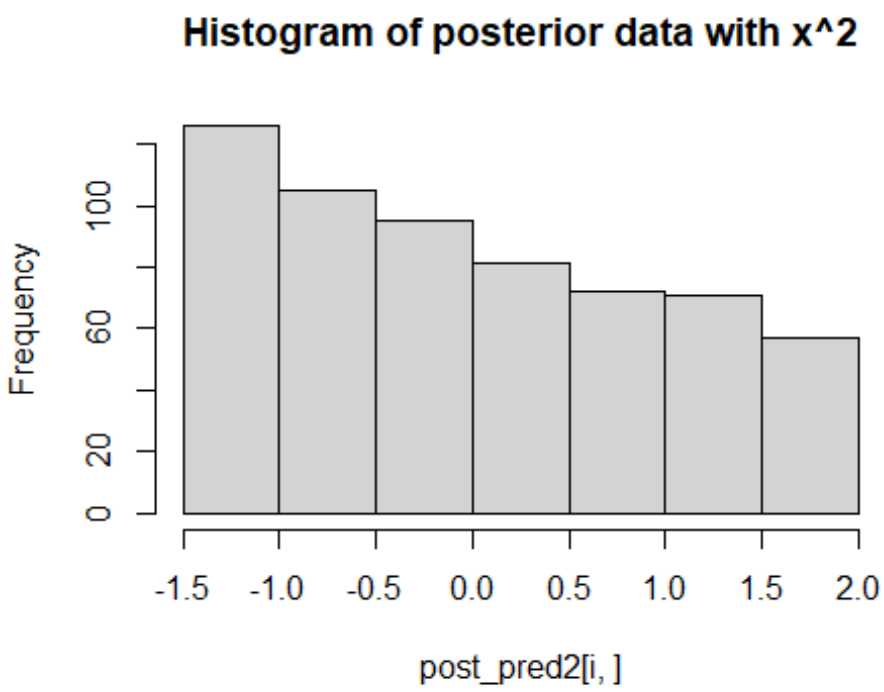

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-2
-1
0
1
2
post_pred2[i,]

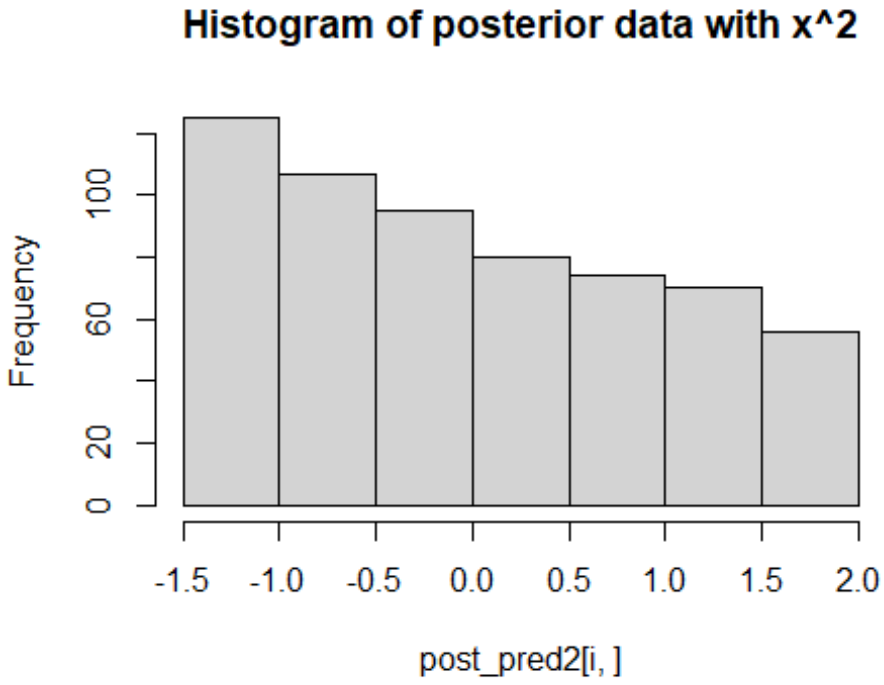

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]

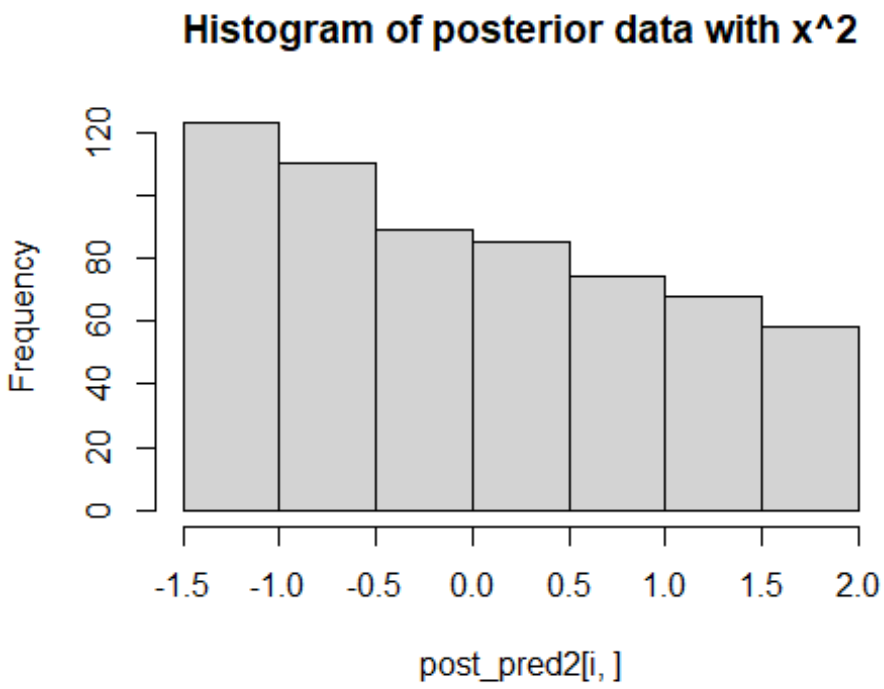

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-2
-1
0
1
2
post_pred2[i,]

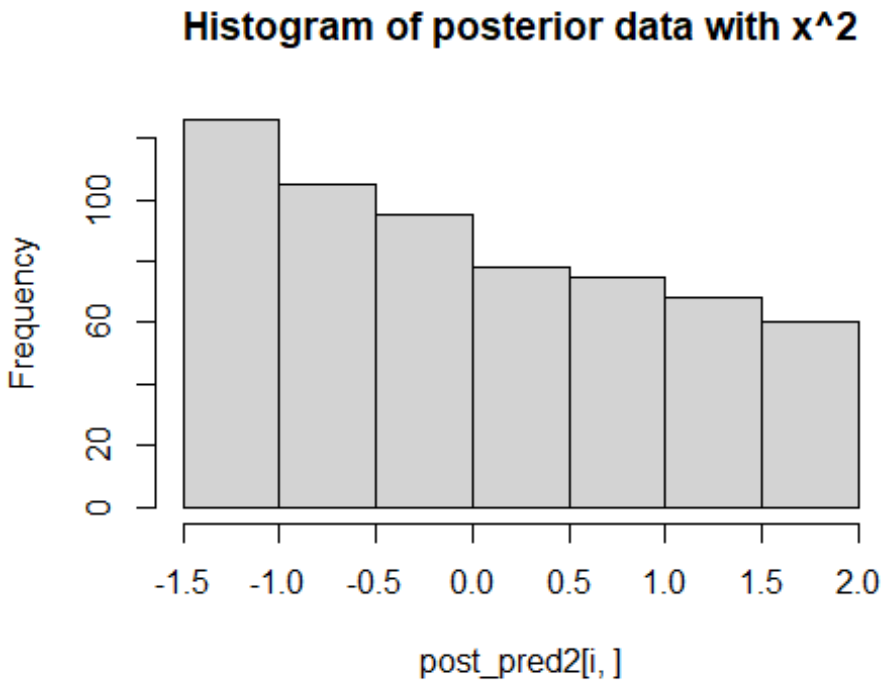

Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-2
-1
0
1
2
post_pred2[i,]

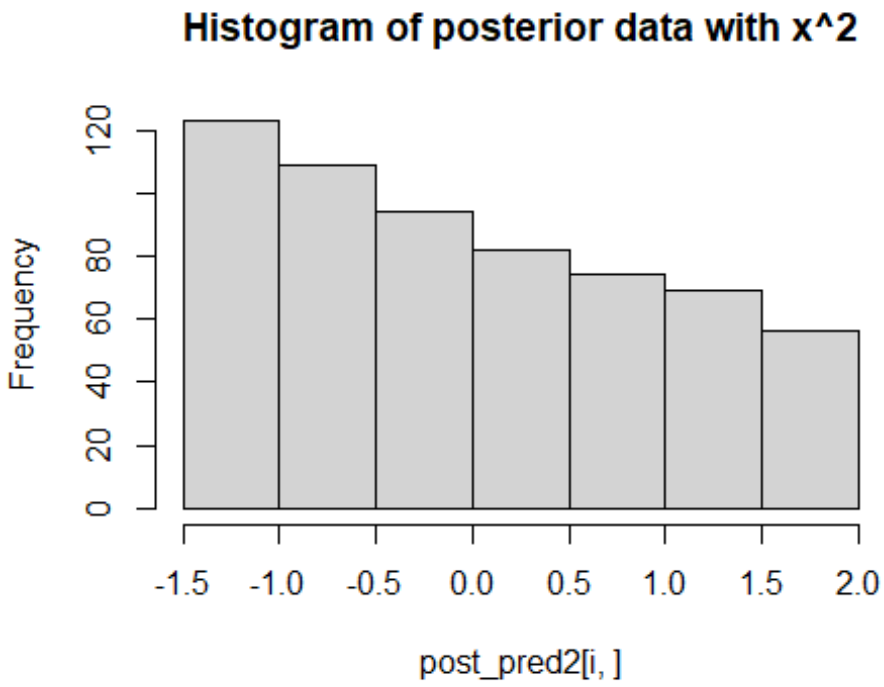

Histogram of posterior data with x^2
Frequency
0
20
60
100
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]


Histogram of posterior data with x^2
Frequency
0
20
60
100
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]


Histogram of posterior data with x^2
Frequency
0
20
60
100
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]


Histogram of posterior data with x^2
Frequency
0
20
40
60
80
120
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
post_pred2[i,]





Analysis of histograms

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).

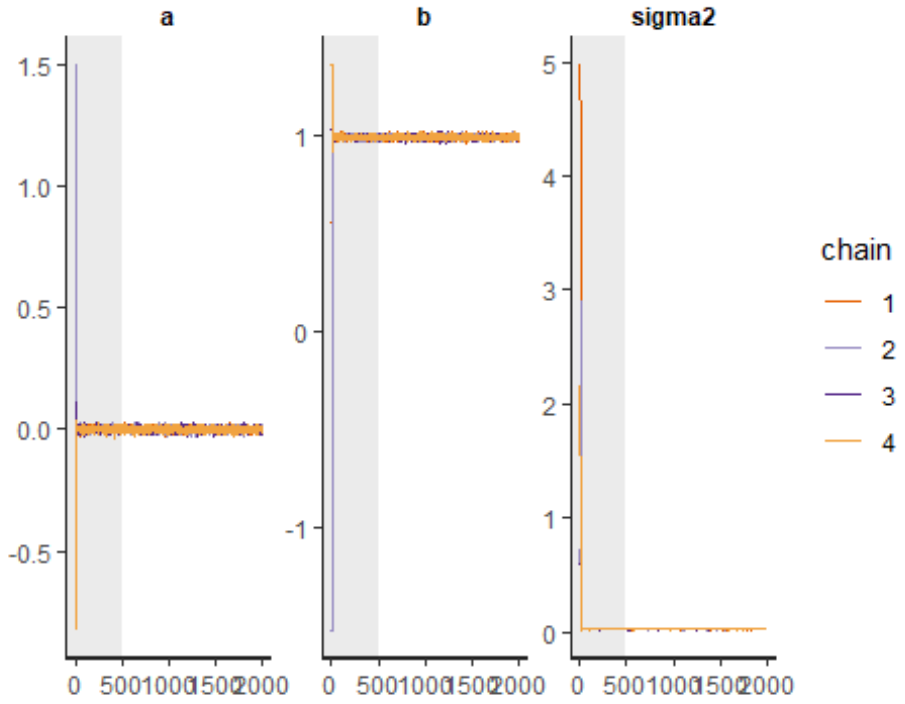
```
x.month.train <- x.month[seq(1, length(x.month), 2)]
x.month.test <- x.month[seq(2, length(x.month), 2)]
y.month.train <- y.month[seq(1, length(y.month), 2)]
y.month.test <- y.month[seq(2, length(y.month), 2)]
```

Sample from the posterior distribution of the first model ($\mu = a + bx$) conditional on the training data and check for convergence

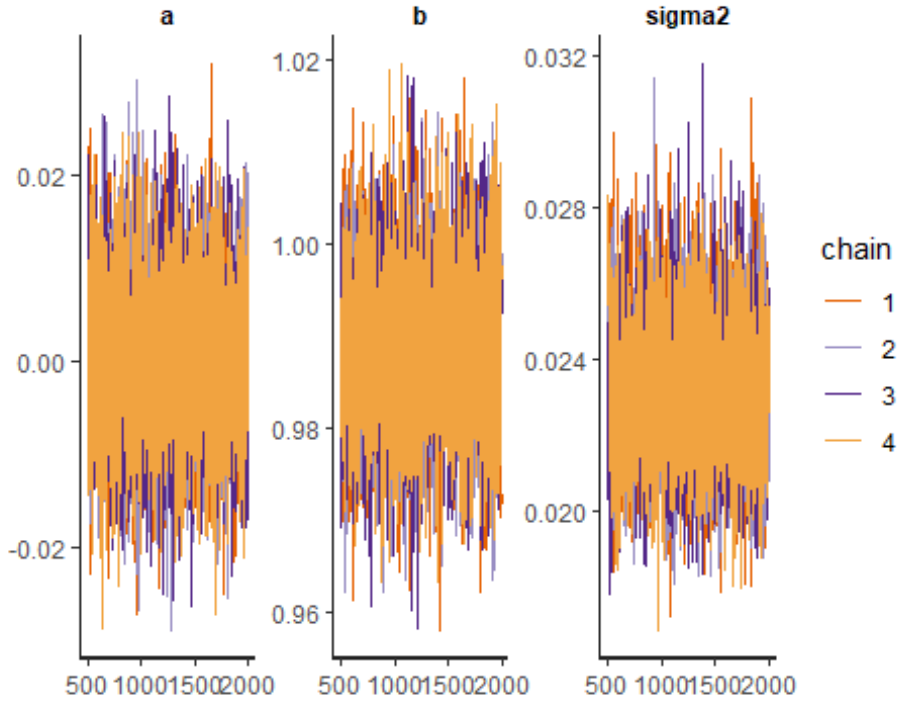
```
set.seed(123)
data <- list (N=length(x.month.train), y=y.month.train, x=x.month.train)
post3=stan(model_code=mauna_loa_c02_model,data=data,warmup=500,iter=2000,chai
ns=4,thin=1,control = list(adapt_delta = 0.8,max_treedepth = 10))

# 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 18:23:17 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(post3, pars=c("a","b","sigma2"),plotfun= "trace", inc_warmup = TRUE)
```



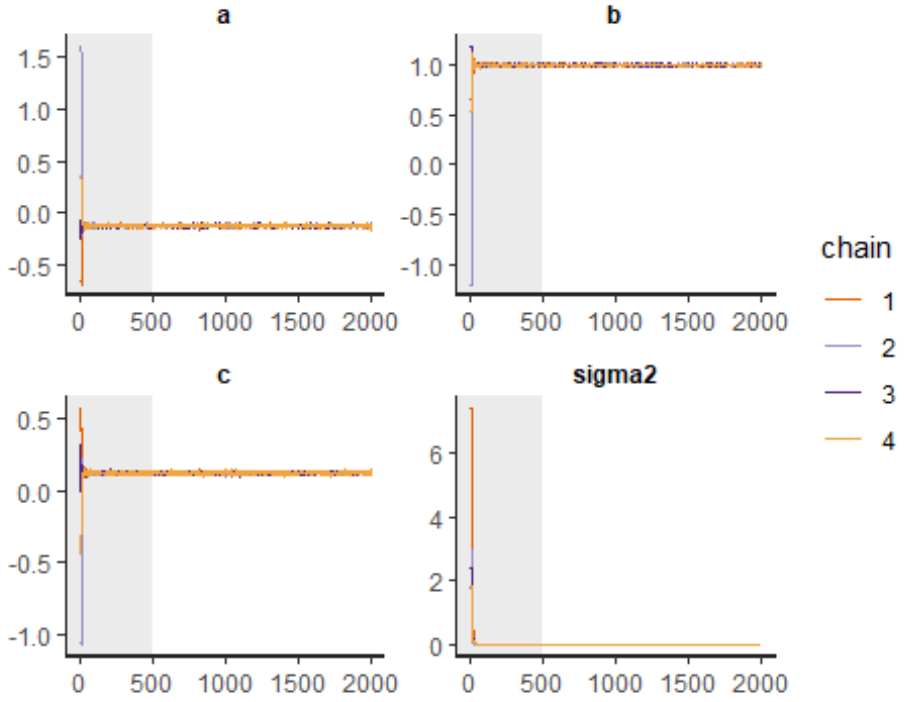

```
plot(post3, pars=c("a","b","sigma2"), plotfun= "trace", inc_warmup = FALSE)
```



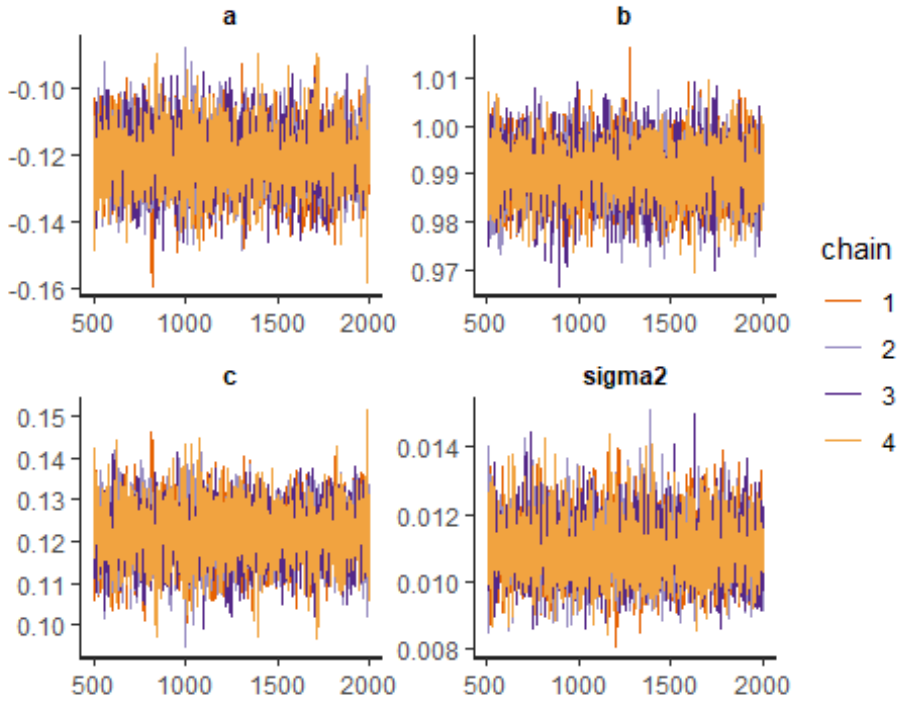
The Rhats are good at 1 and visually one can see that the chains have converged.

Sample from the posterior distribution of the second model ($\mu = a + bx + cx^2$) 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,cha
ins=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 18:23:52 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 ≈ 1.81 lower than the RMSE for the first linear model at ≈ 2.62, which indicates that the second model ($mu = a + b * x + c * x^2$) 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.