# Posterior predictive check: Mauna Loa CO2 continued

Week6-ex2, solution

R-template `ex_linearRegression_postcheck.R`.

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 *data set* and do the following:

1. Plot histograms of the replicate data sets. Compare the histograms of replicate data sets to the histogram of the real data. Discuss whether the replicate histograms look similar to the real data histogram – remember to justify your discussion.

To sample a replicate data set you must sample $\tilde{y} = [\tilde{y}_1, \ldots, \tilde{y}_n]$ values from $\tilde{y}_i \sim N(\mu_i, \sigma^2)$, where $\mu_i = a + bx_i$ and $a, b, \sigma^2$ are drawn from the posterior. For example, pick 20 random triplets of $a, b, \sigma^2$ 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:

2. Plot the posterior mean and central 95% credible interval of $\mu$ and $\tilde{y}$ as a function of $x$ with the new model. Overlay this plot with the data. Is there visual improvement in the fit between the 95% credible intervals and observations? If yes, how?
3. Do the same full data posterior predictive check as with the original model by visualizing the histograms of the full true data and 20 full replicate data sets. Discuss whether the replicate histograms look similar to the real data histogram – remember to justify your discussion. Did the model refinement improve models behaviour in this respect?

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,

4. compare these two alternative models (M1: $\mu_i = a + bx_i$, M2: $\mu_i = a + bx_i + cx_i^2$) with posterior predictive comparison by dividing the data into two parts and taking every other observation into training data and every other observation into test data.

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

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

and the root mean squared error

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

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,

$$p(\tilde{y}_i|\tilde{x}_i, y_{\text{training}}, x_{\text{training}}) = \int p(\tilde{y}_i|\tilde{x}_i, \theta)p(\theta|y_{\text{training}}, x_{\text{training}})d\theta$$
$$\approx \sum_{s=1}^{S} p(\tilde{y}_i|\tilde{x}_i, \theta^{(s)})$$

where $\theta^{(s)}$ is a sample from the posterior distribution of the parameters ($\theta = \{a, b, \sigma^2\}$ in the original model), that is $\theta^{(s)} \sim p(\theta|y_{\text{training}}, x_{\text{training}})$.

# Solution

Load the needed libraries.

```
library(ggplot2)
```

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
library(StanHeaders)
library(rstan)
```

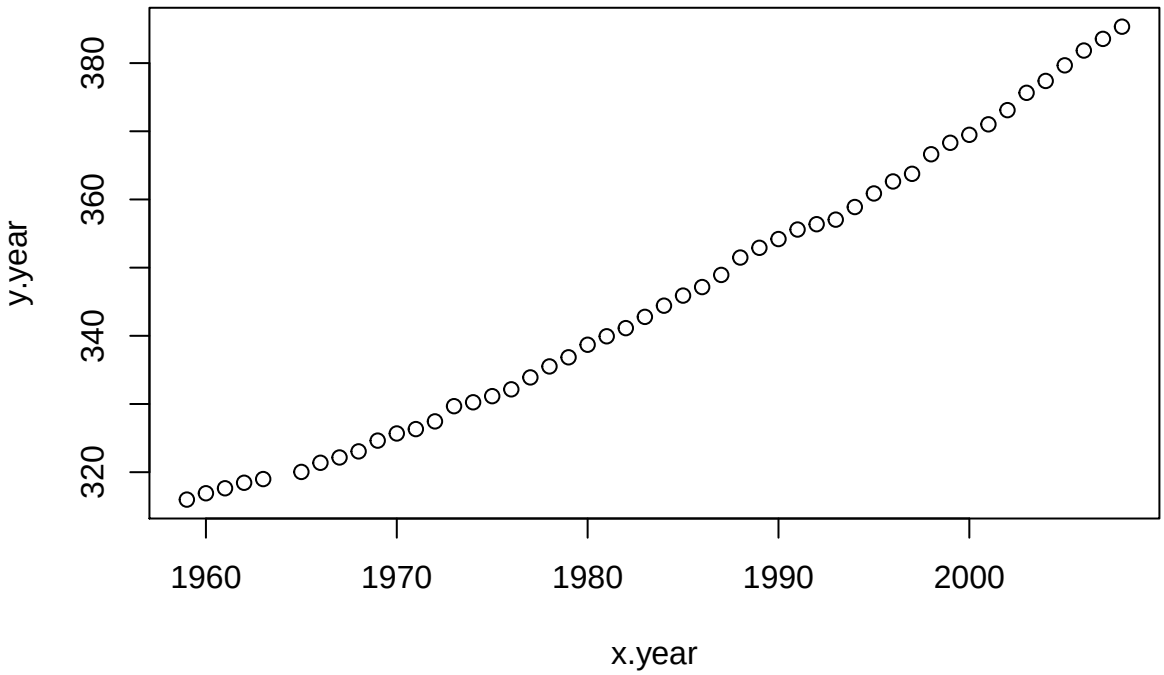
```
## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)
```

```
## 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)
```

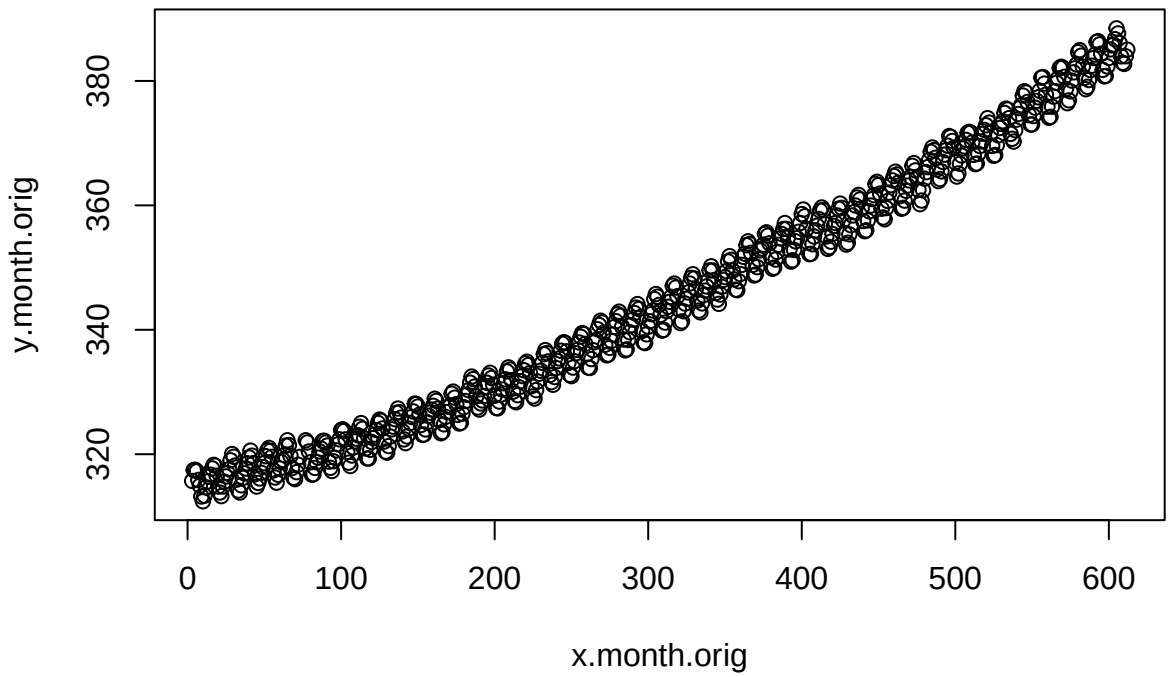
```
set.seed(123)

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)
```

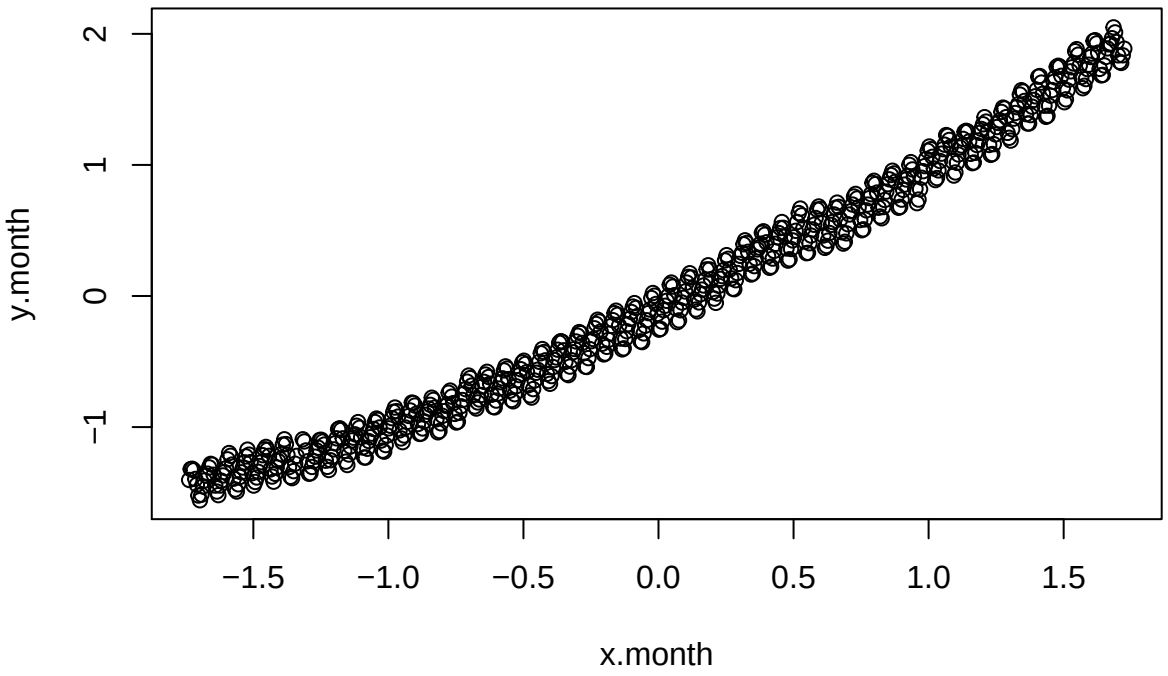



```
# Let's take the monthy values and construct a "running month" vector
y.month.orig = as.vector(t(maunaloa.dat[,2:13]))
x.month.orig = as.vector(seq(1,length(y.month.orig),1))

# 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)
```

```r
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)
```




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

## Posterior predictive check

Analysis with the original model

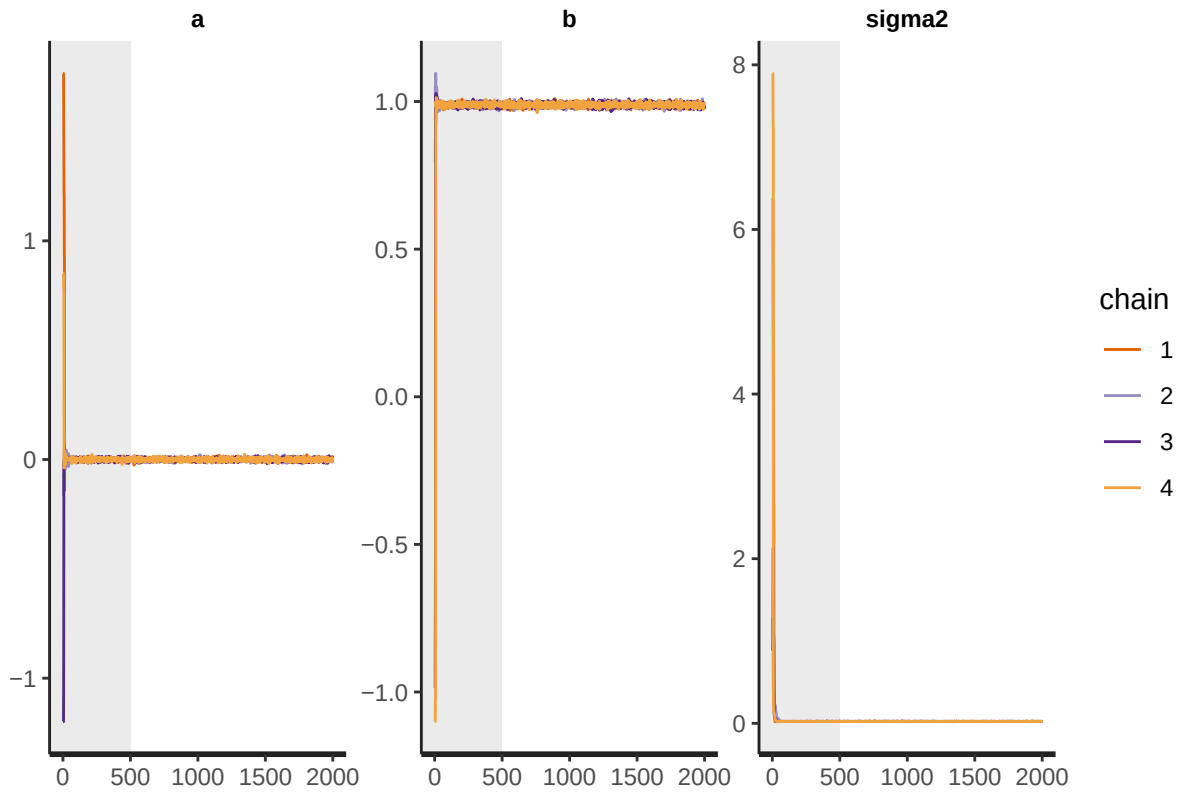
$y_i = a + bx_i + \epsilon_i$

```
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];
```

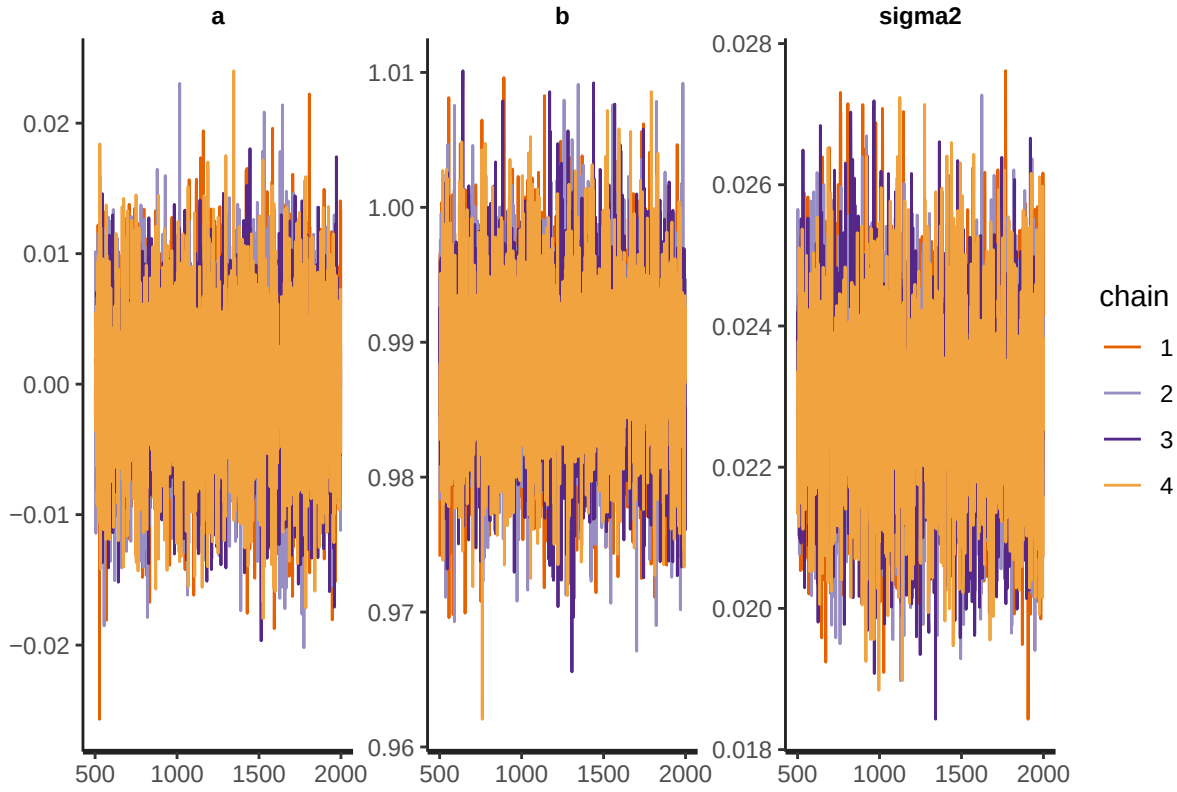
```
  sigma=sqrt(sigma2);

  for( i in 1 : N ) {
    mu[i] = a + b * x[i];
  }
}
model{
  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(a

# Check for convergence, see PSRF (Rhat in Stan)
print(post,pars=c("a","b","sigma2"))
```

```
## Inference for Stan model: 237139d61e51ca610fd34064e89729dd.
## 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  7841    1
## b      0.99       0 0.01  0.98 0.98 0.99 0.99  1.00  7538    1
## sigma2 0.02       0 0.00  0.02 0.02 0.02 0.02  0.03  3389    1
##
## Samples were drawn using NUTS(diag_e) at Thu Nov 26 12:41:30 2020.
## 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)
```

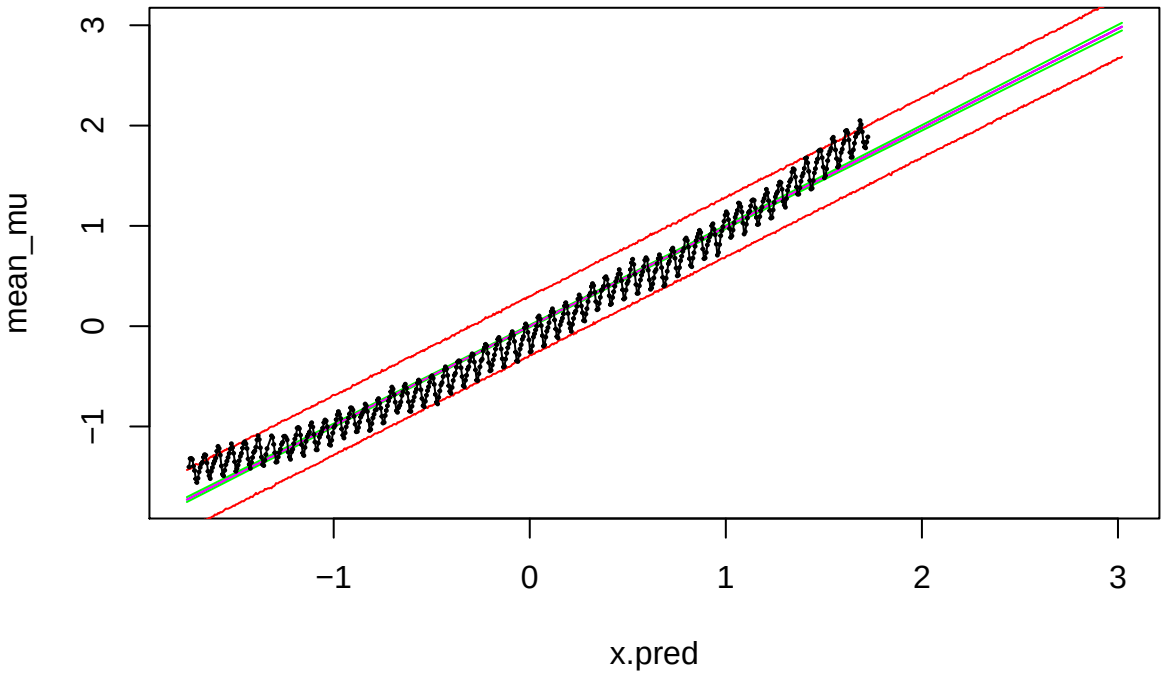



```
plot(post, pars=c("a","b","sigma2"), plotfun= "trace", inc_warmup = FALSE)
```




```
set.seed(123)
# extract the samples into matrix
```

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

# write.table(post_sample, file="param1.txt", row.names=FALSE, col.names=TRUE)

# Let's first check that we can reproduce the earlier result
# Note! we did not backtransform the parameters so we need to work with the transformed data as well.
# 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)
```



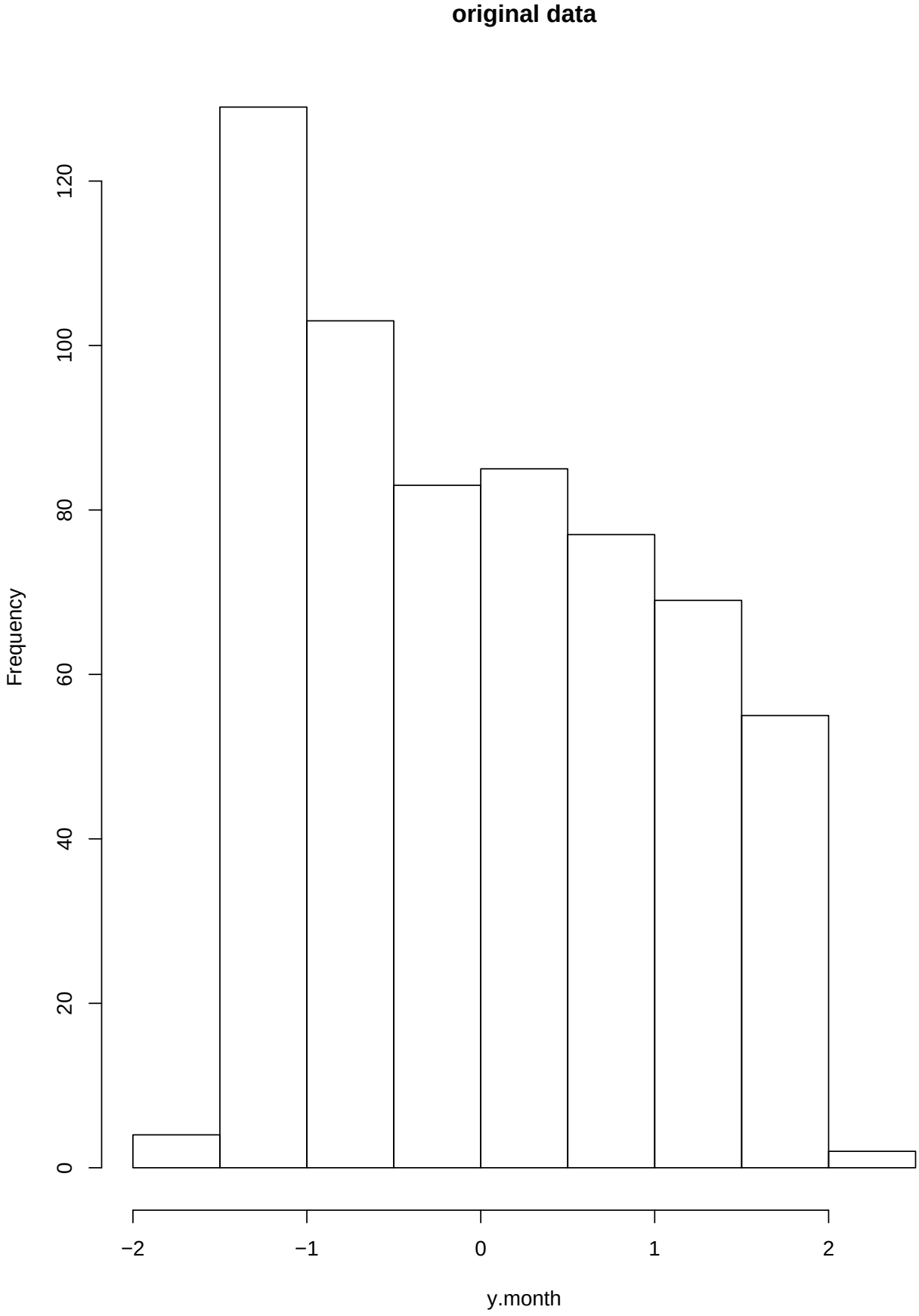

Now we are ready to conduct the posterior predictive check

First, sample 20 random draws from the Markov chain

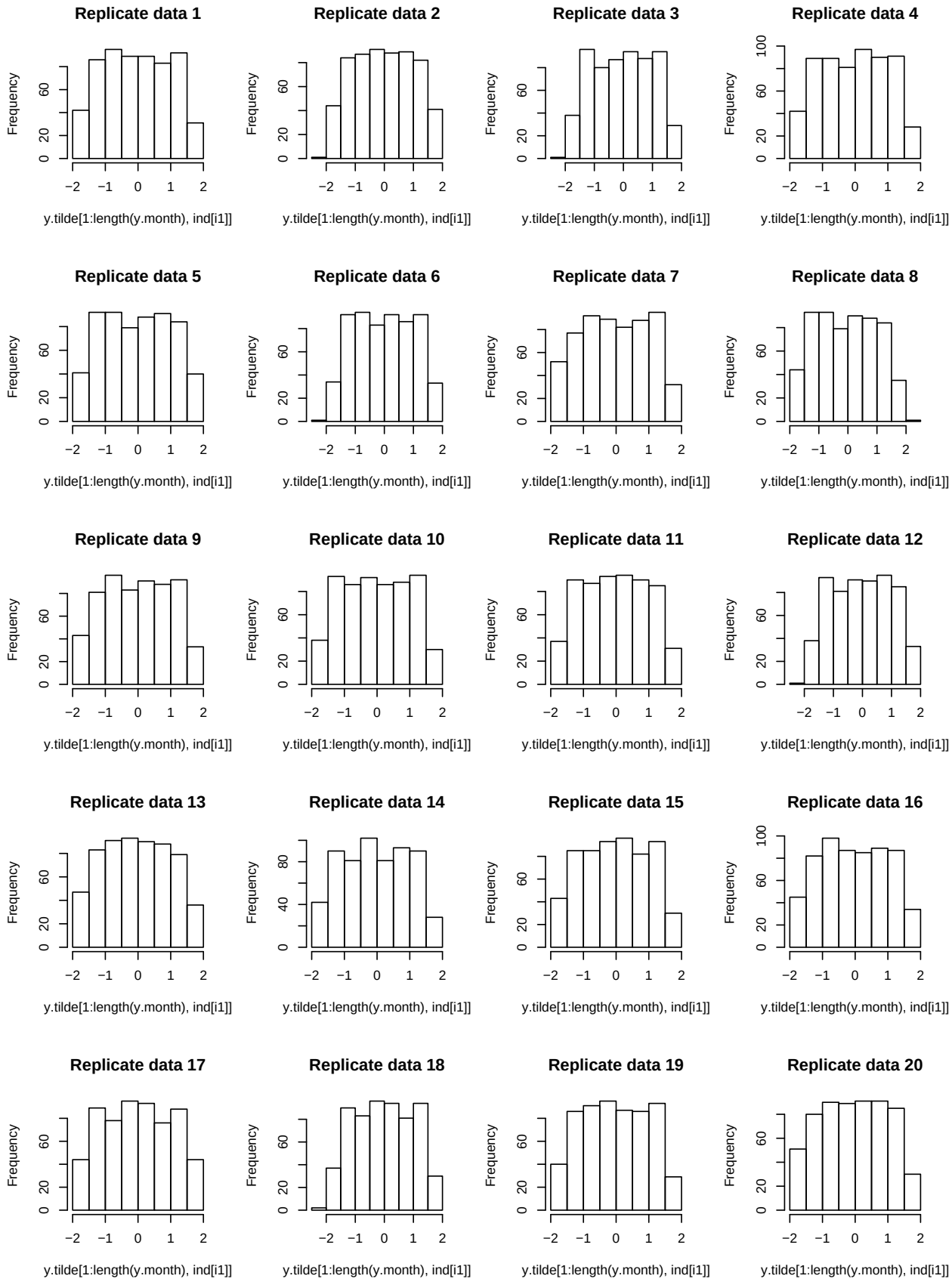
```
set.seed(123)
ind = sample(seq(1,length(a_dot),length=length(a_dot)),20,replace=FALSE)
```

The vector y.rep contains replicate y values for each observation time. Hence, we can plot the histogram of the replicate data by taking the ind columns on y.rep until the last observation time

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


original data
Frequency
0
20
40
60
80
100
120
−2
−1
0
1
2
y.month

```r
par(mfrow=c(5,4))
for (i1 in 1:20){
  hist(y.tilde[1:length(y.month),ind[i1]],main=paste("Replicate data", i1))
}
```


Replicate data 1
Replicate data 2
Replicate data 3
Replicate data 4
Replicate data 5
Replicate data 6
Replicate data 7
Replicate data 8
Replicate data 9
Replicate data 10
Replicate data 11
Replicate data 12
Replicate data 13
Replicate data 14
Replicate data 15
Replicate data 16
Replicate data 17
Replicate data 18
Replicate data 19
Replicate data 20
Frequency
y.tilde[1:length(y.month), ind[i1]]

The histogram of the original data is clearly skewed to the right whereas the histograms of replicate data are more uniform. Moreover, the replicate data histograms span smaller values than the real data histogram. Hence, the model does not explain the data perfectly. The explanation for why not can be seen already from the earlier predictive plot. The real data does not have pure linear trend.

Let's refine the model so that

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

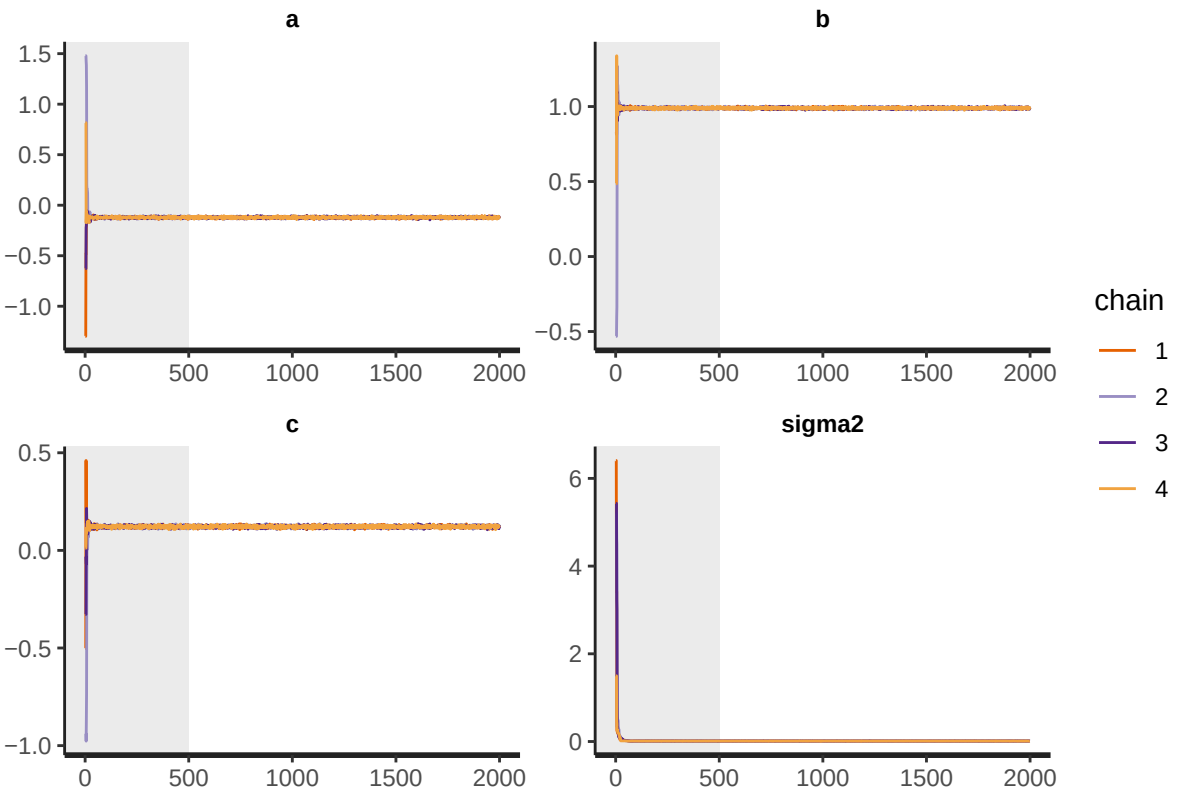
and rerun the analysis

```
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*x[i]*x[i];
  }
}
model{
  sigma2 ~ inv_gamma(0.001,0.001);

  for( i in 1 : N ) {
    y[i] ~ normal(mu[i],sigma);
  }
}"
set.seed(123)
post2=stan(model_code=mauna_loa_c02_model2,data=data,warmup=500,iter=2000,chains=4,thin=1,control = lis

# Check for convergence, see PSRF (Rhat in Stan)
print(post2,pars=c("a","b","c","sigma2"))
```

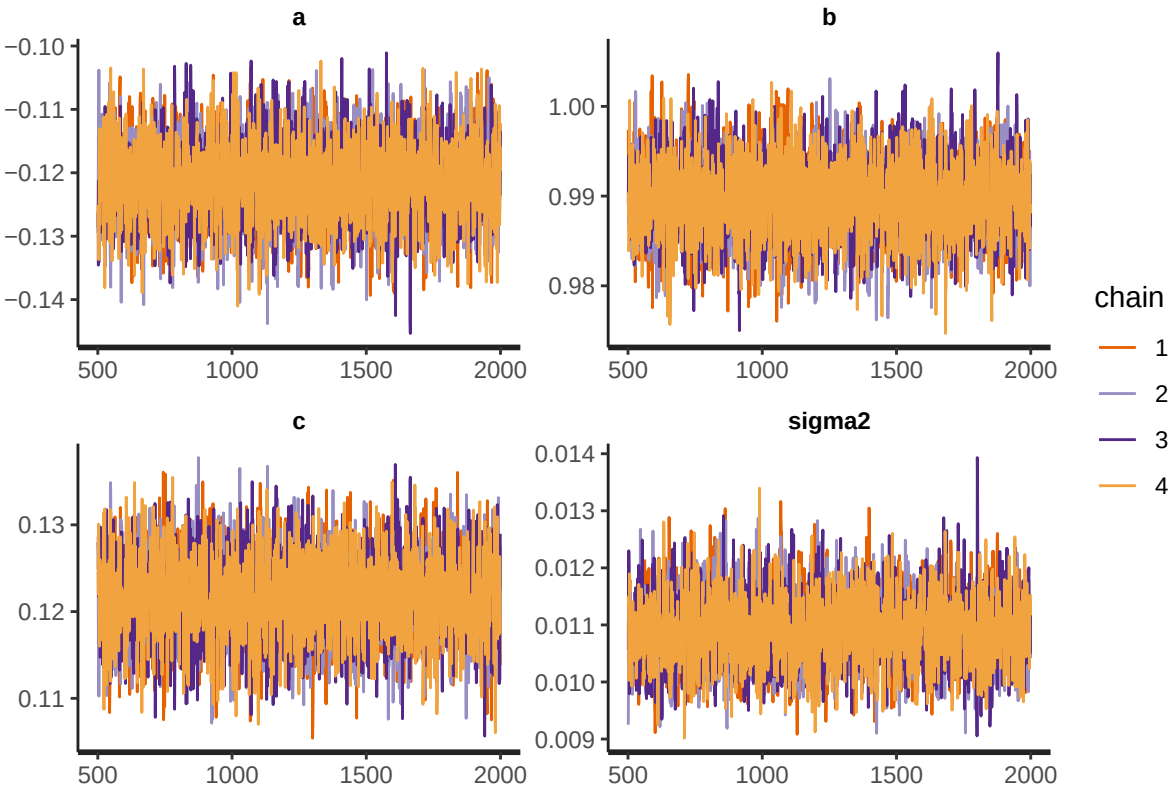
```
## Inference for Stan model: 36686673f5e4a95dc4b3a172e66bd43d.
## 4 chains, each with iter=2000; warmup=500; thin=1;
## post-warmup draws per chain=1500, total post-warmup draws=6000.
##
##         mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## a      -0.12       0 0.01 -0.13 -0.13 -0.12 -0.12 -0.11  3854    1
## b       0.99       0 0.00  0.98  0.99  0.99  0.99  1.00  6552    1
## c       0.12       0 0.00  0.11  0.12  0.12  0.12  0.13  4032    1
## sigma2  0.01       0 0.00  0.01  0.01  0.01  0.01  0.01  3685    1
##
```

```
## Samples were drawn using NUTS(diag_e) at Thu Nov 26 12:43:03 2020.
## 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)
```



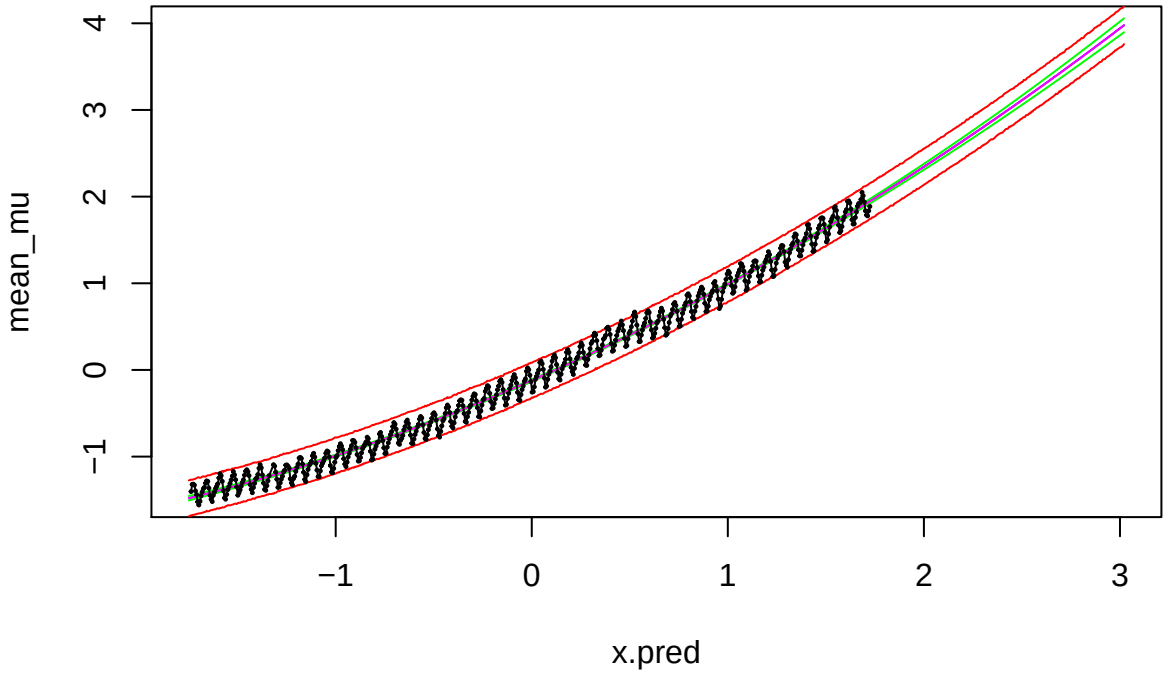

```
plot(post2, pars=c("a","b","c","sigma2"), plotfun= "trace", inc_warmup = FALSE)
```




Let's then examine the prediction along months

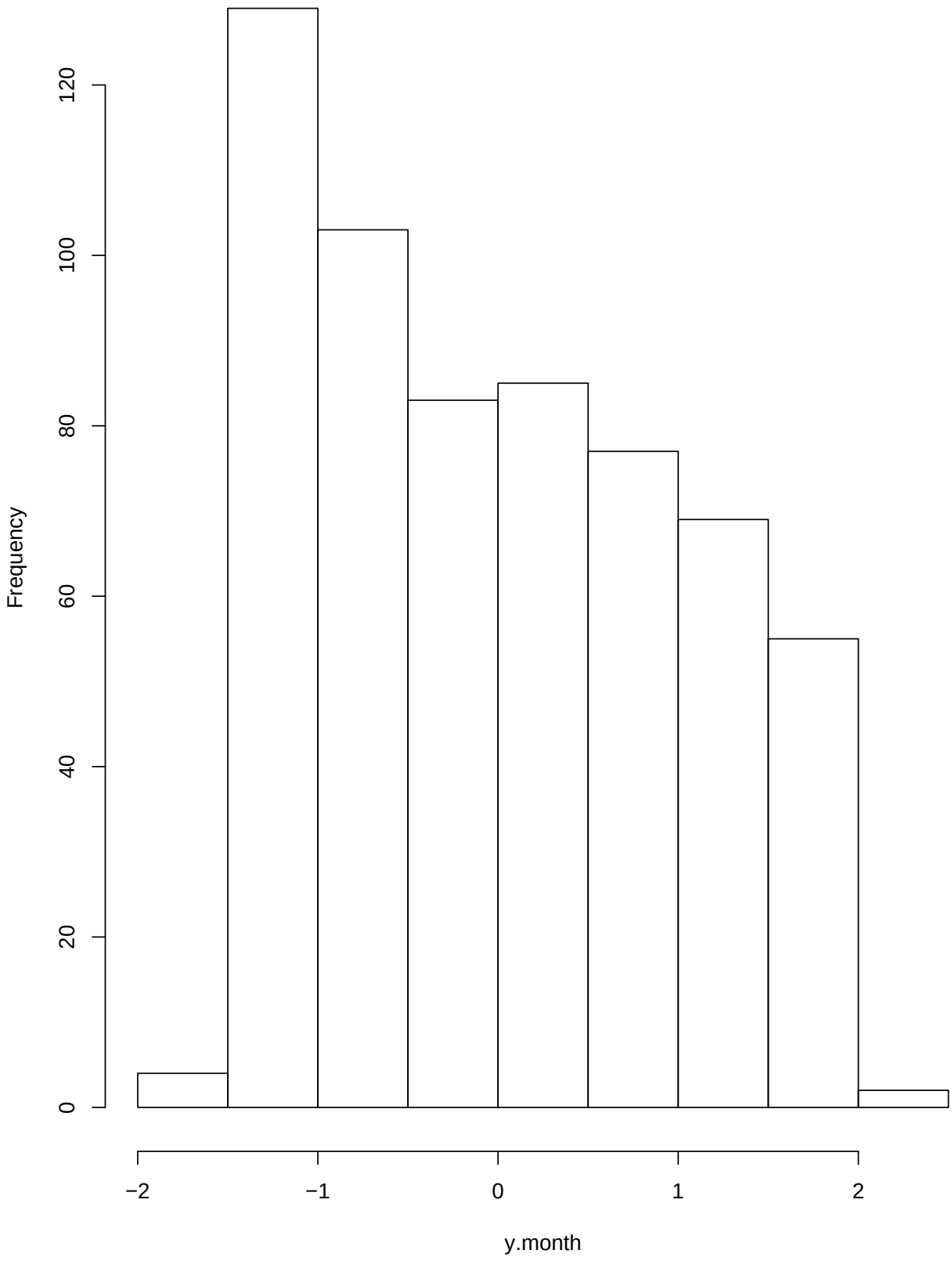
```
set.seed(123)
# extract the samples into matrix
post_sample <- as.matrix(post2, pars =c("a","b", "c","sigma2"))    # combine all chains into one matrix
a_dot=post_sample[,1]
b_dot=post_sample[,2]
c_dot=post_sample[,3]
sigma2_dot=post_sample[,4]

# write.table(post_sample, file="param2.txt", row.names=FALSE, col.names=TRUE)

# Let's first check that we can reproduce the earlier result
# Note! we did not backtransform the parameters so we need to work with the transformed data as well.
# 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] + c_dot*x.pred[i]*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)
```

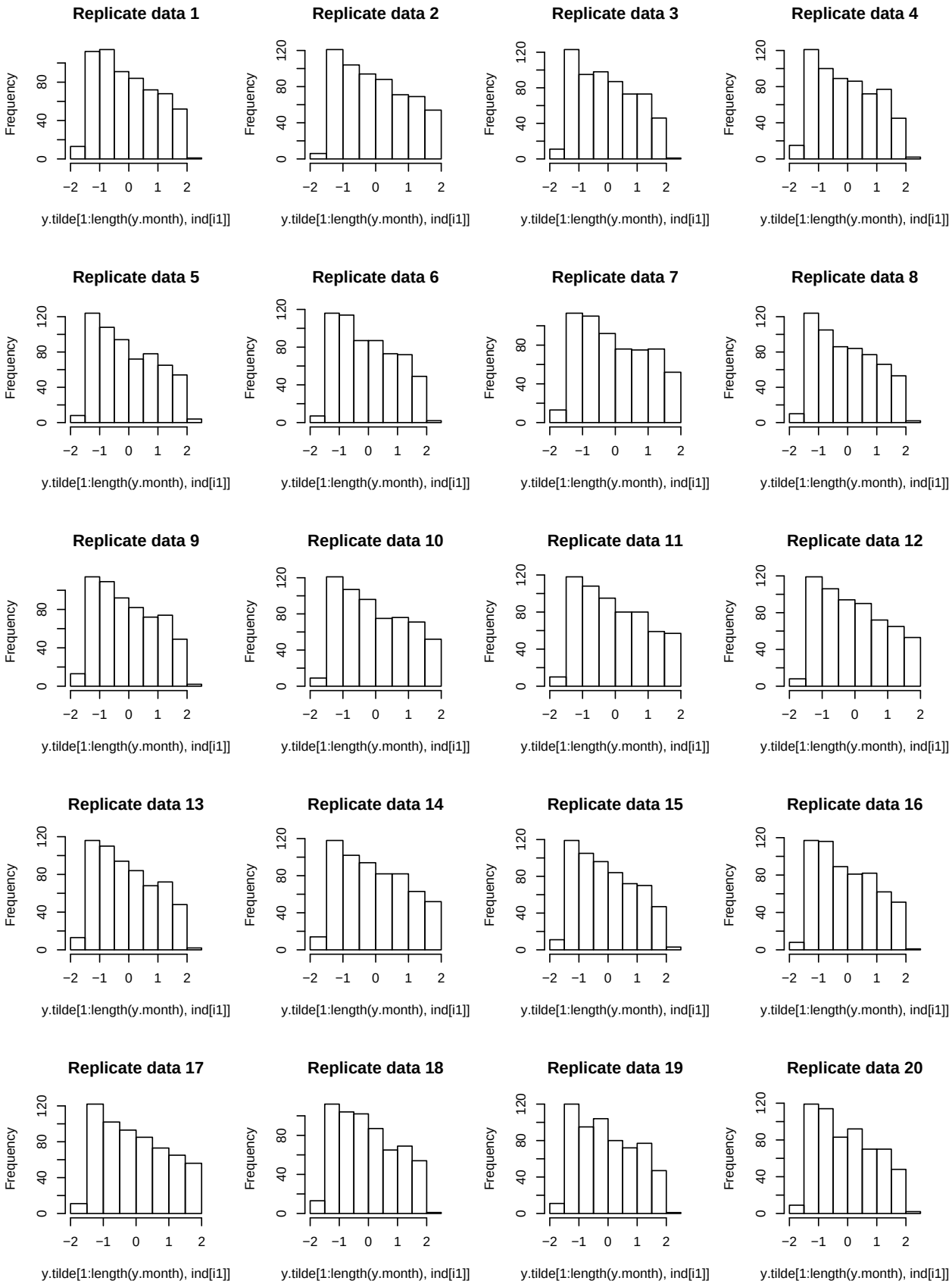



And then we can do the asked posterior predictive check

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


original data
Frequency
0
20
40
60
80
100
120
−2
−1
0
1
2
y.month

```r
par(mfrow=c(5,4))
for (i1 in 1:20){
  hist(y.tilde[1:length(y.month),ind[i1]],main=paste("Replicate data", i1))
}
```




Now the histograms seem to match well since both real data histogram and the replicate data histogram are

skewed to the right. Hence, we could say that the revised model is a better fit to the data than the original one.

## 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).

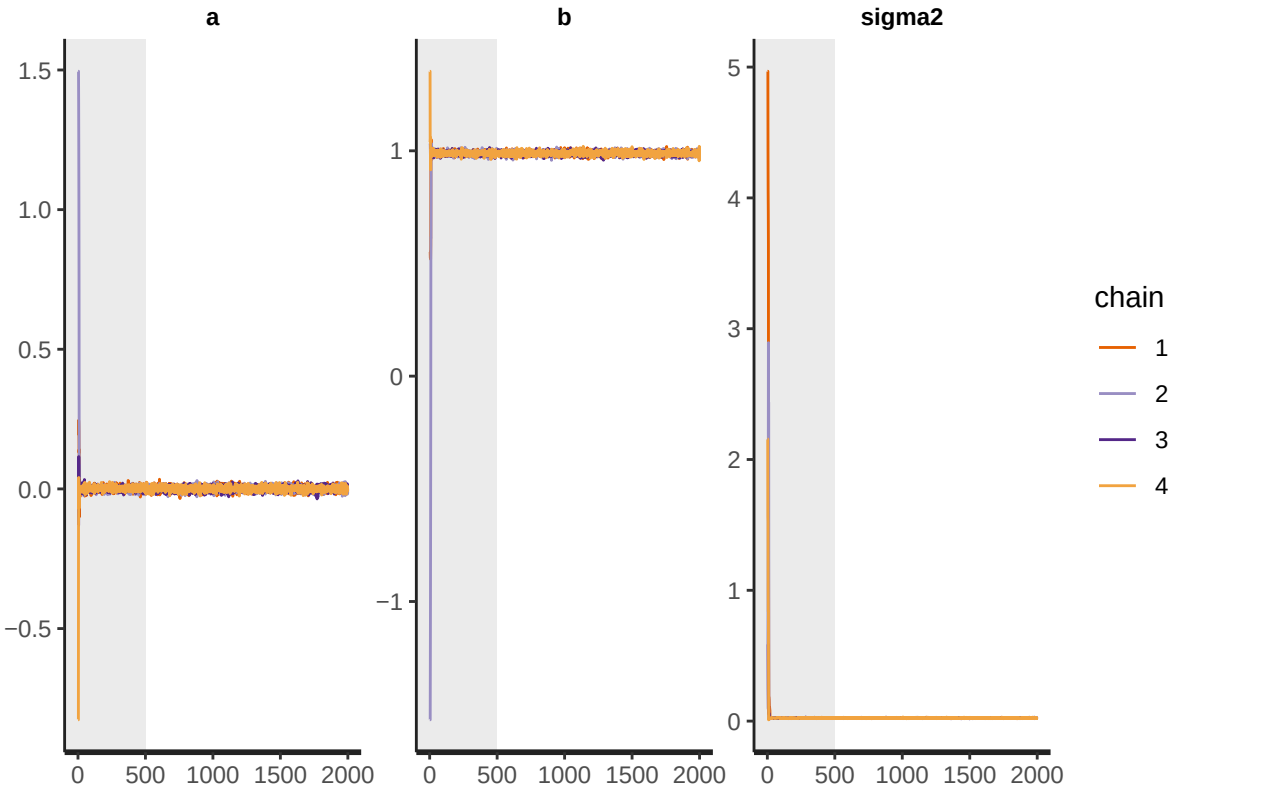
```
y.training = y.month[seq(from=1,to=length(y.month),by=2)]
x.training = x.month[seq(from=1,to=length(y.month),by=2)]
y.test = y.month[seq(from=2,to=length(y.month),by=2)]
x.test = x.month[seq(from=2,to=length(y.month),by=2)]

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

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

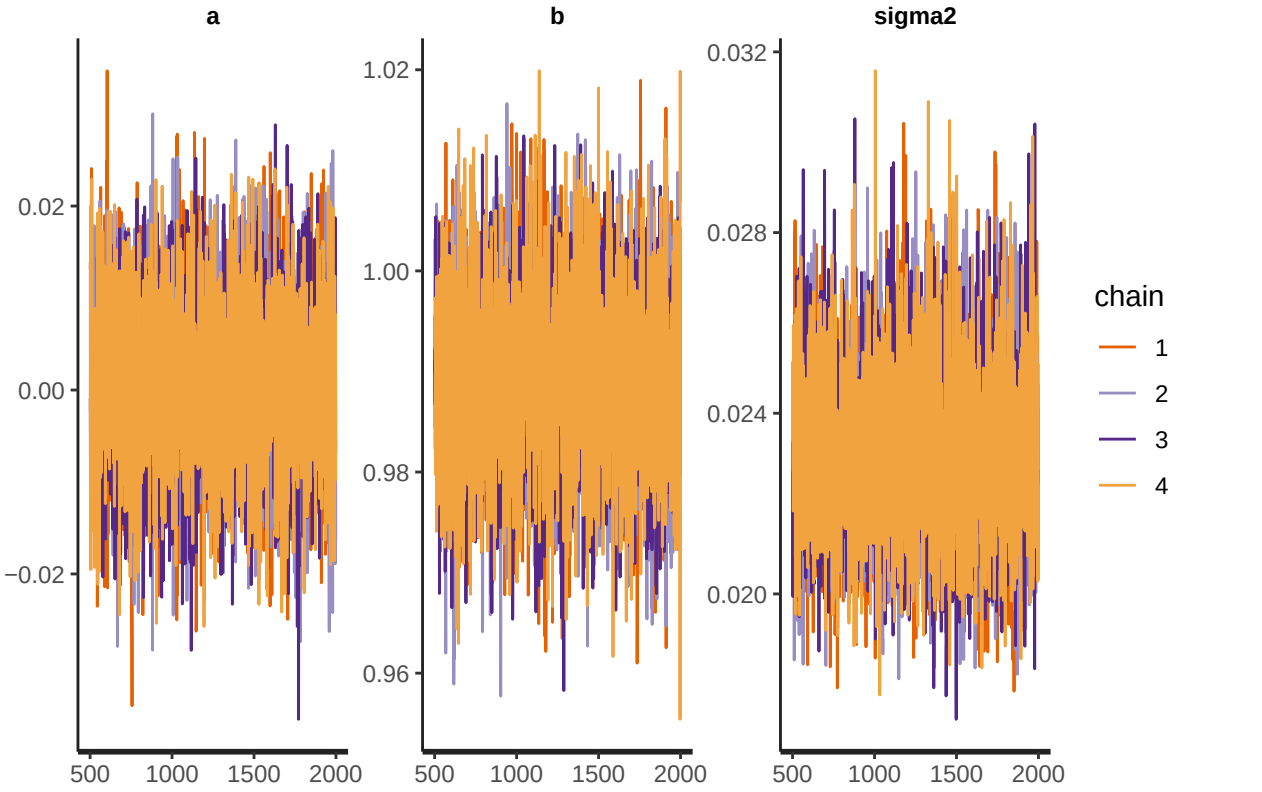
```
set.seed(123)
post=stan(model_code=mauna_loa_c02_model,data=data,warmup=500,iter=2000,chains=4,thin=1,control = list(a

print(post,pars=c("a","b","sigma2"))
```

```
## Inference for Stan model: 237139d61e51ca610fd34064e89729dd.
## 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  6805    1
## b      0.99       0 0.01  0.97  0.98 0.99 0.99  1.01  5970    1
## sigma2 0.02       0 0.00  0.02  0.02 0.02 0.02  0.03  5124    1
##
## Samples were drawn using NUTS(diag_e) at Thu Nov 26 12:43:16 2020.
## 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)
```

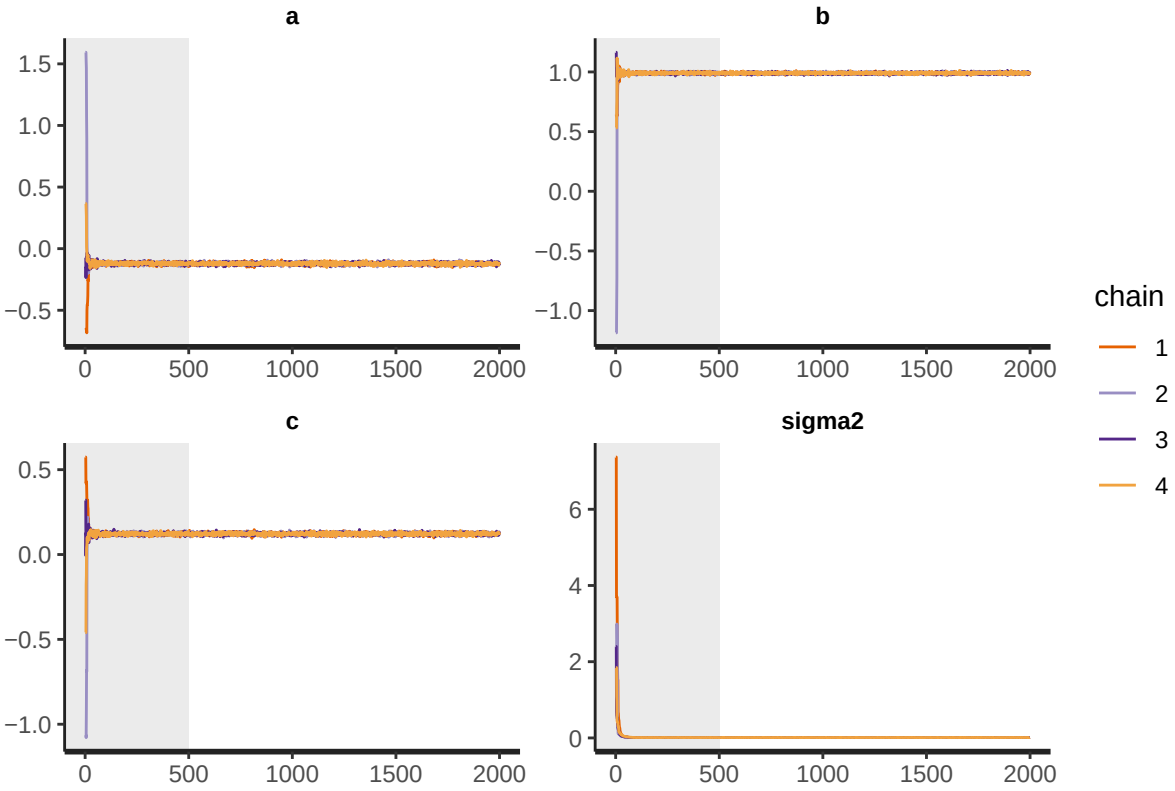



Sample from the posterior distribution of the second model ($\mu = a + bx + cx^2$) conditional on the training data and check for convergence

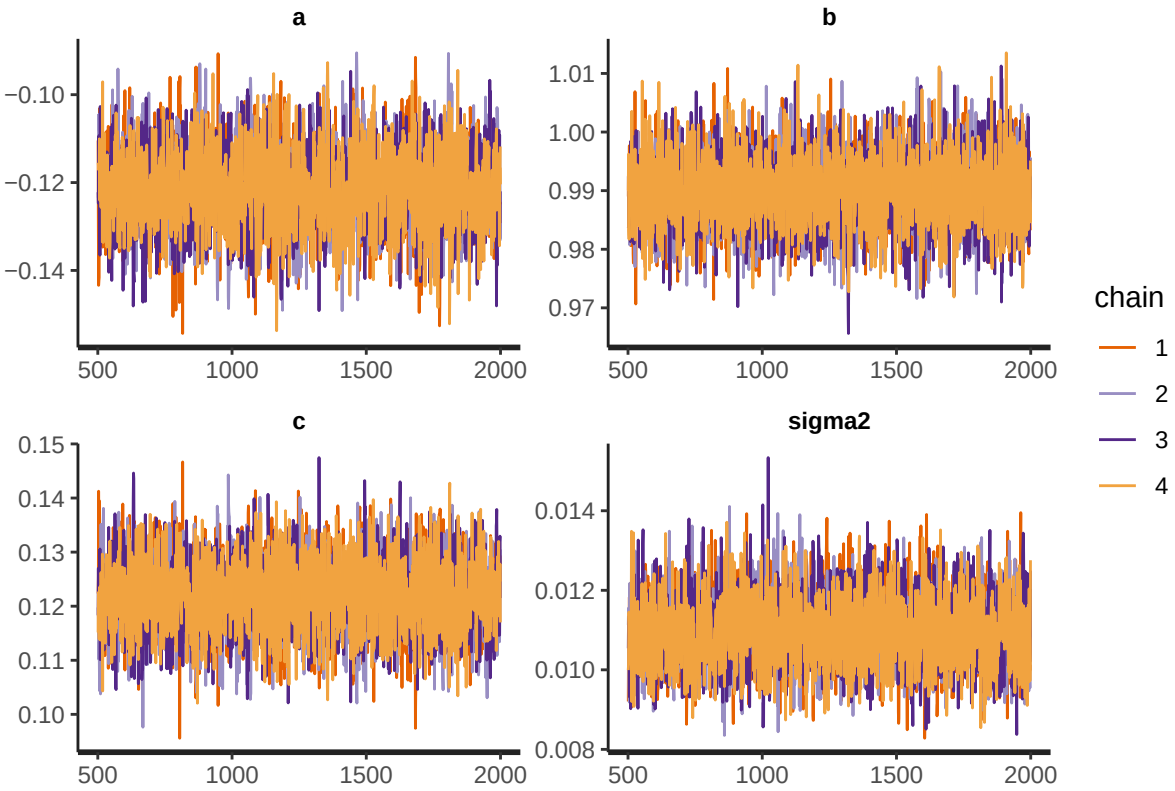
```
set.seed(123)
post2=stan(model_code=mauna_loa_c02_model2,data=data,warmup=500,iter=2000,chains=4,thin=1,control = lis

# Check for convergence, see PSRF (Rhat in Stan)
print(post2,pars=c("a","b","c","sigma2"))
```

```
## Inference for Stan model: 36686673f5e4a95dc4b3a172e66bd43d.
## 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  3319    1
## b       0.99       0 0.01  0.98  0.99  0.99  0.99  1.00  5738    1
## c       0.12       0 0.01  0.11  0.12  0.12  0.13  0.13  3554    1
## sigma2  0.01       0 0.00  0.01  0.01  0.01  0.01  0.01  4101    1
##
## Samples were drawn using NUTS(diag_e) at Thu Nov 26 12:43:23 2020.
## 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)
```




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

```
# extract the samples into matrix
post_sample <- as.matrix(post, pars =c("a","b","sigma2"))    # combine all chains into one matrix in R
a_dot1=post_sample[,1]
b_dot1=post_sample[,2]
sigma2_dot1=post_sample[,3]

# extract the samples into matrix
post_sample <- as.matrix(post2, pars =c("a","b", "c","sigma2"))    # combine all chains into one matrix
a_dot2=post_sample[,1]
b_dot2=post_sample[,2]
c_dot2=post_sample[,3]
sigma2_dot2=post_sample[,4]


lpd = matrix(NA,length(x.test),1)
rmse = matrix(NA,length(x.test),1)
lpd2 = matrix(NA,length(x.test),1)
rmse2 = matrix(NA,length(x.test),1)
for (i in 1:length(x.test)) {
  mu = (a_dot1 + b_dot1*x.test[i])
  rmse[i] = (mean(mu) - y.test[i])^2
  lpd[i] =  mean(dnorm(y.test[i],mean=mu, sqrt(sigma2_dot1)))
  mu2 = (a_dot2 + b_dot2*x.test[i] + c_dot2*x.test[i]^2)
  rmse2[i] = (mean(mu2) - y.test[i])^2
  lpd2[i] =  mean(dnorm(y.test[i],mean=mu2, sqrt(sigma2_dot2)))
}
```

```r
lpd = sum(log(lpd))
rmse = sqrt(mean(rmse))
lpd2 = sum(log(lpd2))
rmse2 = sqrt(mean(rmse2))

comparison = matrix(c(lpd,lpd2,rmse,rmse2),2,2)
colnames(comparison) <- c("LPD", "RMSE")
rownames(comparison) <- c("Model 1:", "Model 2:")
print(comparison)
```

```
##                LPD      RMSE
## Model 1: 144.2735 0.1503248
## Model 2: 255.9285 0.1038786
```

The model with the second order term on x (Model 2) works better with respect to both RMSE and LPD.

# Grading

**Total points 20:** 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.