2.4 Exercise: posterior

Benjamin Rosenbaum

November 3, 2021

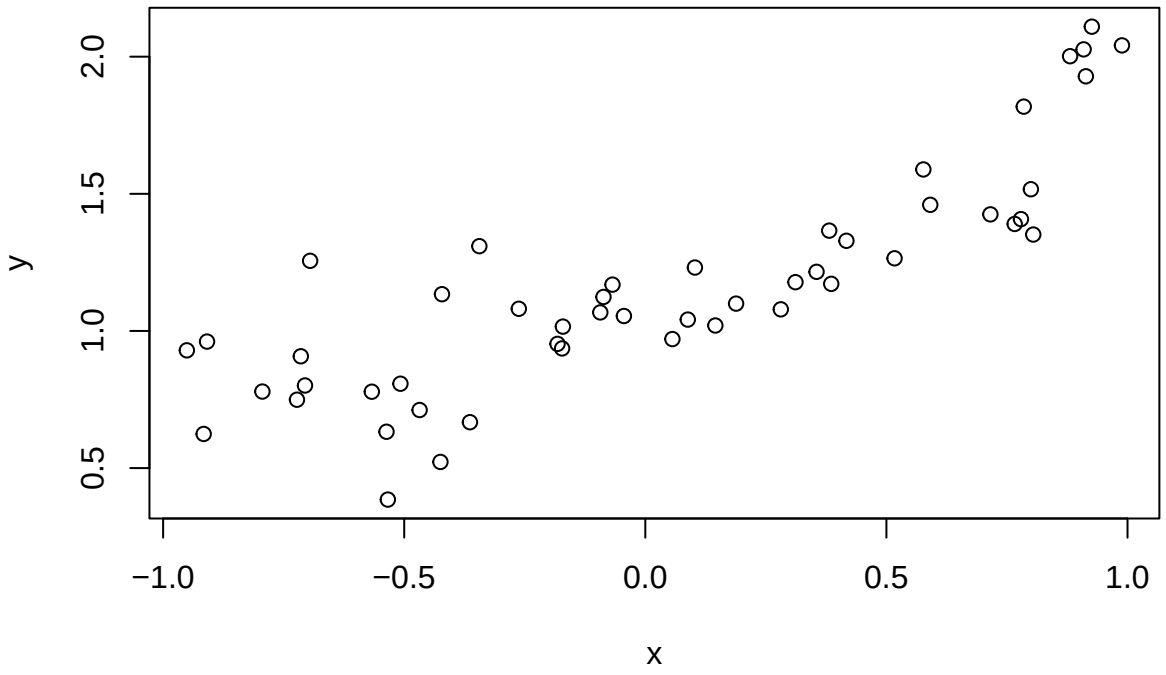
Repeat the steps of the last exercise 2.3 and fit a quadratic regression (U-shaped or hump-shaped response).

Setup

```
rm(list=ls())
library(rstan)
library(coda)
library(BayesianTools)

rstan_options(auto_write = TRUE)
options(mc.cores = 4) # number of CPU cores
```

Generate data

```
set.seed(123) # initiate random number generator for reproducibility

n=50

a=1.0
b=0.5
c=0.4
sigma=0.2

x = runif(n=n, min=-1, max=1)
y = rnorm(n=n, mean=a+b*x+c*x^2, sd=sigma)

df = data.frame(x=x,
                y=y)

plot(df)
```



Statistical model

$$y_i \sim \text{normal}(\mu_i, \sigma)$$
$$\mu_i = a + b \cdot x_i + c \cdot x_i^2$$

```
stan_code_quad = '
data {
  int n;
  vector[n] x;
  vector[n] y;
}
parameters {
  real a;
  real b;
  real c;
  real<lower=0> sigma;  // standard deviation
}
model {
  // priors
  a ~ normal(0, 10);
  b ~ normal(0, 10);
  c ~ normal(0, 10);
  sigma ~ normal(0, 10);
  // likelihood
  y ~ normal(a + b*x + c * x .* x , sigma);
}
'
```

Data and sampler preparation, MCMC sampling

```
data = list(n=n,
            x=df$x,
            y=df$y)


stan_model_quad = stan_model(model_code=stan_code_quad)

fit.2  = sampling(stan_model_quad,
                  data=data)
```

Explore the posterior distribution

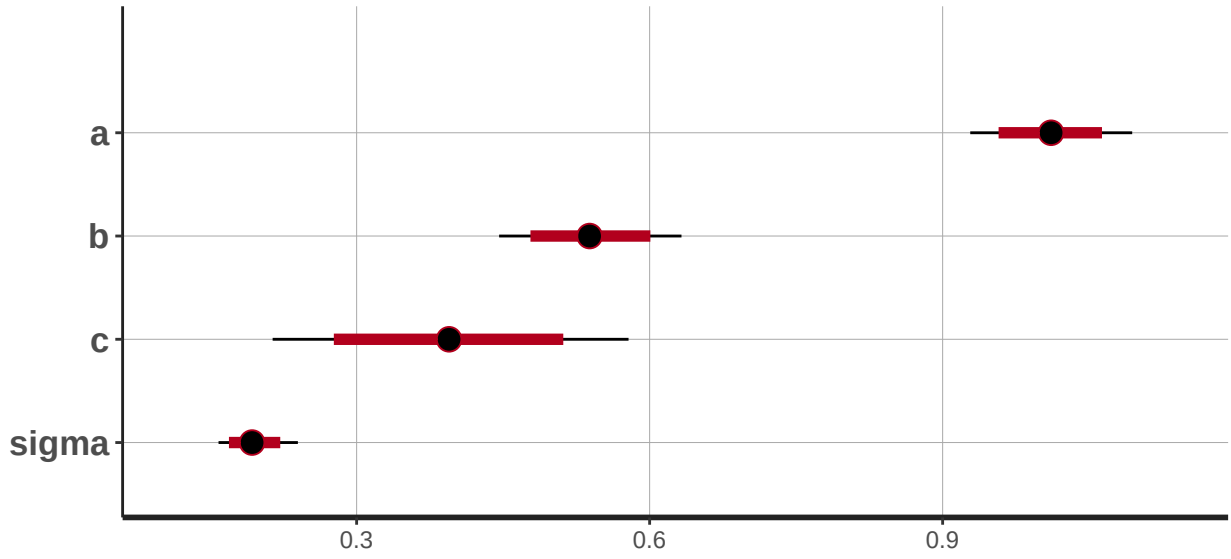
```
print(fit.2, digits=3, probs=c(0.025, 0.975))
```

```
## Inference for Stan model: 2d5f3e7154334f52f36cd1e67784a774.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##         mean se_mean    sd   2.5%  97.5% n_eff  Rhat
## a      1.011   0.001 0.042  0.928  1.094  2118 1.000
## b      0.539   0.001 0.048  0.446  0.633  3173 0.999
## c      0.394   0.002 0.093  0.214  0.578  2171 1.000
## sigma  0.195   0.000 0.021  0.159  0.240  2821 1.003
## lp__  55.791   0.035 1.470 52.011 57.604  1764 1.004
##
## Samples were drawn using NUTS(diag_e) at Tue Oct 19 14:59:27 2021.
## 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).
```

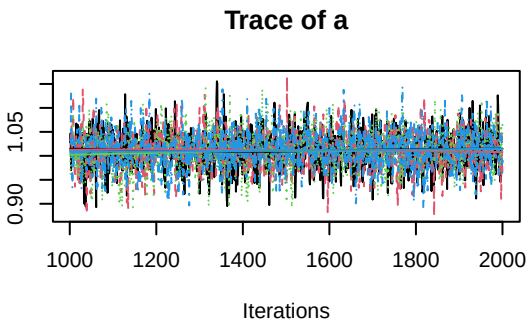
```
plot(fit.2)
```

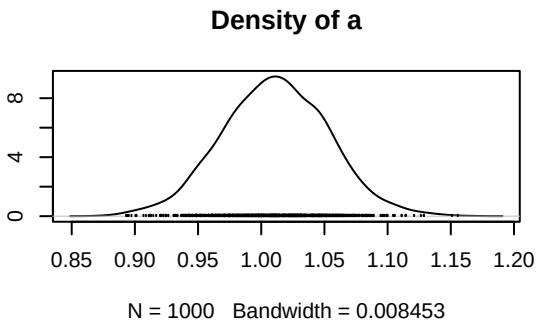
```
## ci_level: 0.8 (80% intervals)
```

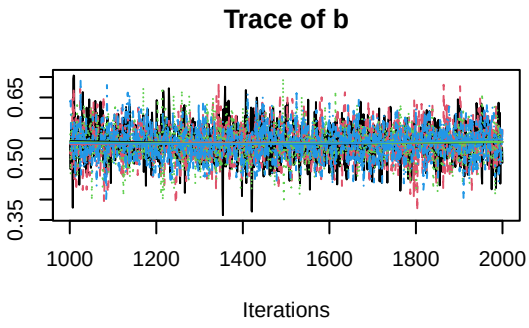
```
## outer_level: 0.95 (95% intervals)
```

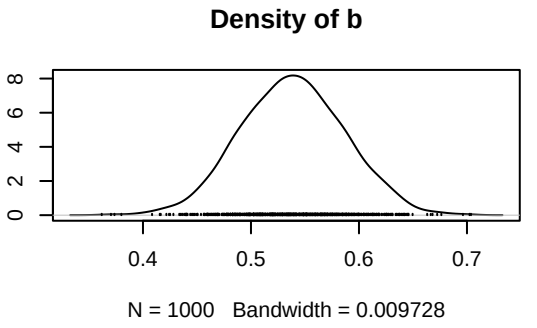


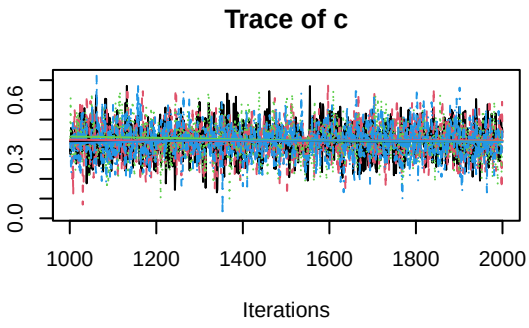

```
plot(As.mcmc.list(fit.2)[, 1:4]) # from coda package
```

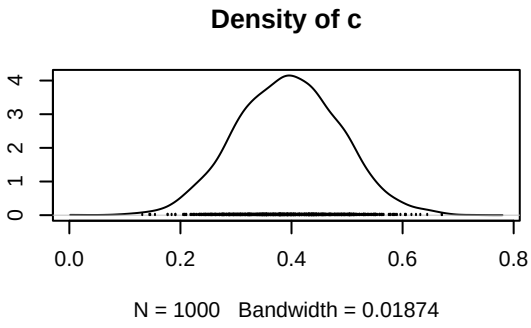


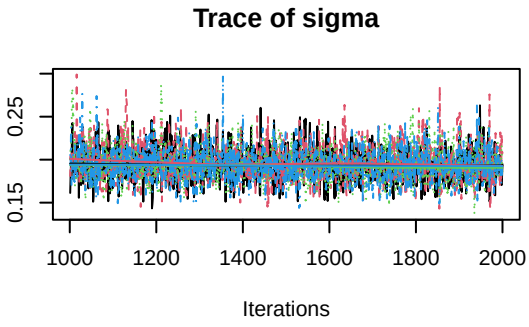


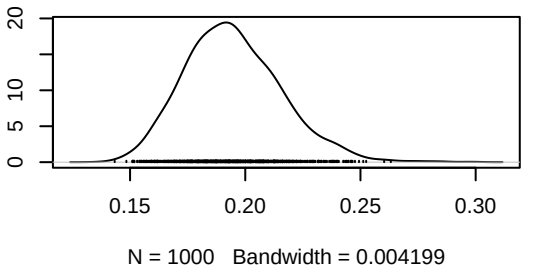




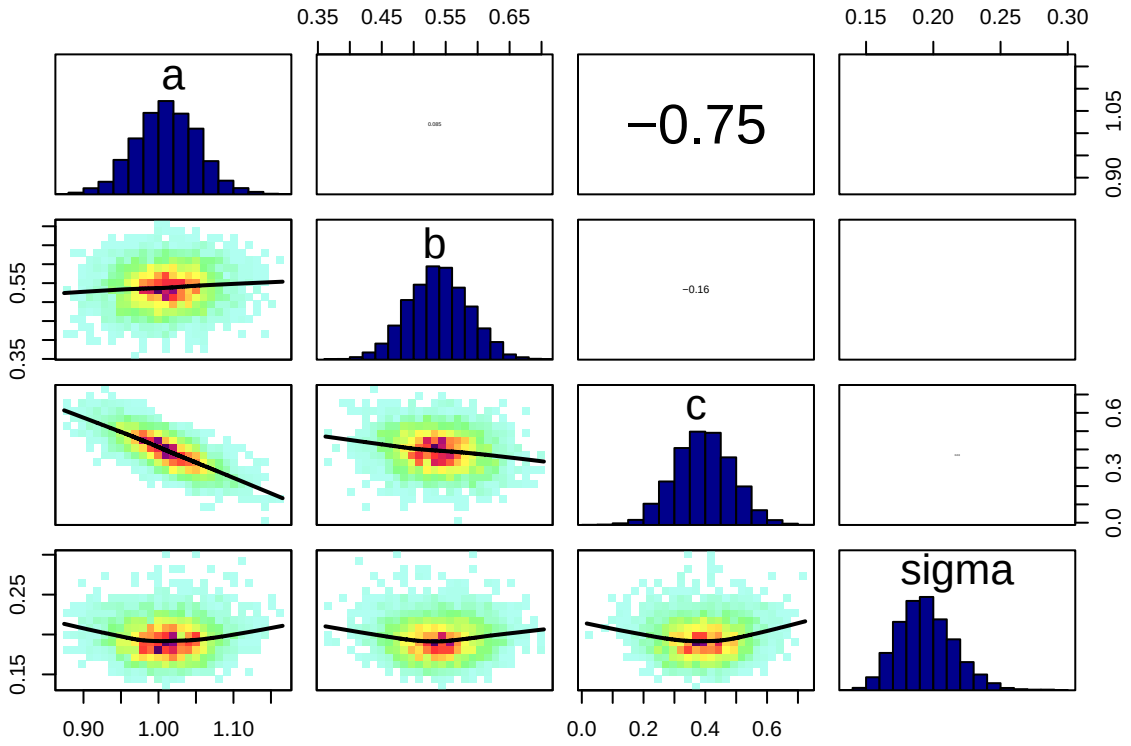




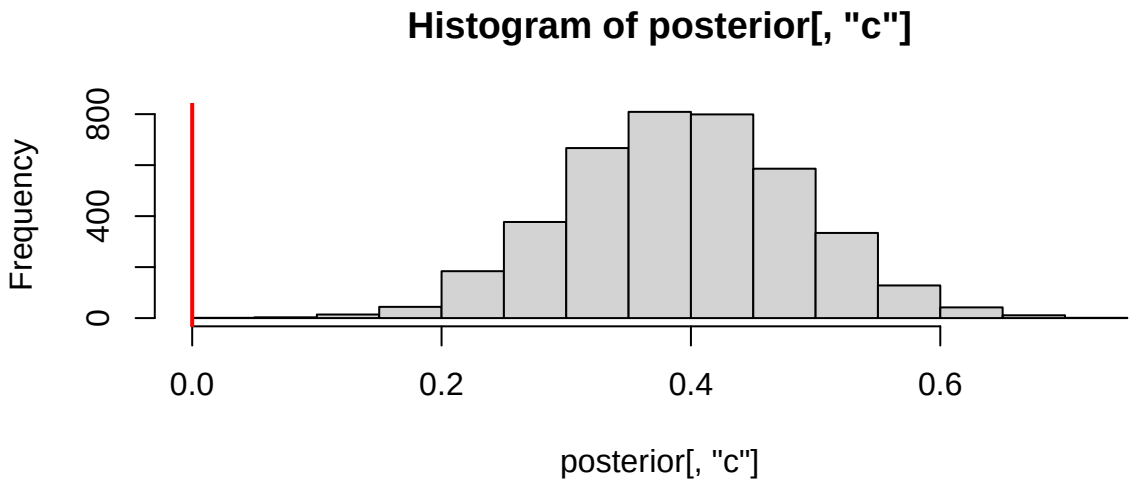





```
posterior=as.matrix(fit.2)

correlationPlot(posterior[, 1:4], thin=1) # from BayesianTools package
```




```
hist(posterior[, "c"])
abline(v=0, col="red", lwd=2)
```



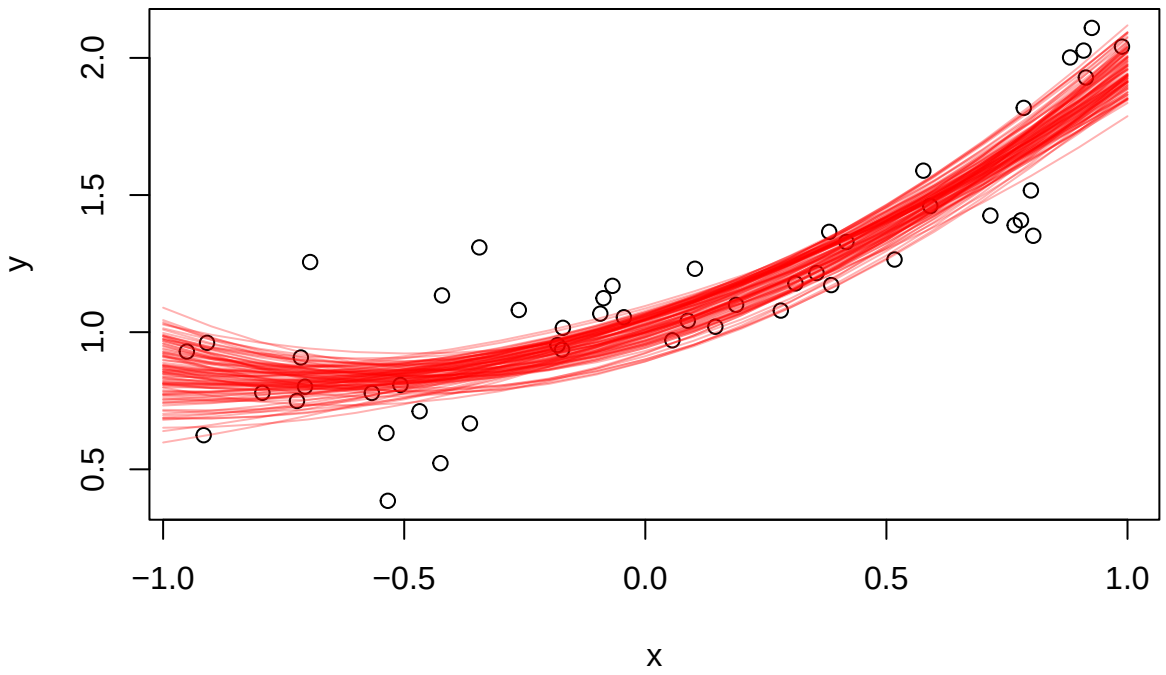

```
sum(posterior[, "c"]>0)/nrow(posterior)
```

```
## [1] 1
```

Posterior predictions

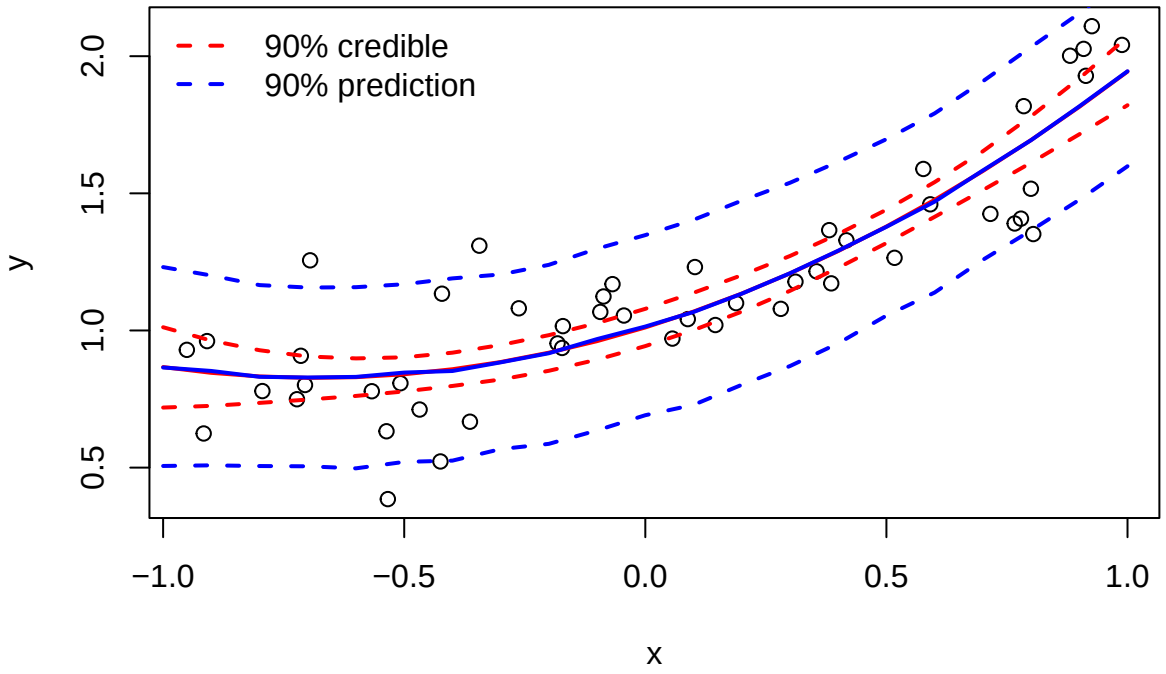
```
x.pred = seq(from=-1, to=1, by=0.1)
y.cred = matrix(0, nrow=nrow(posterior), ncol=length(x.pred))

for(i in 1:nrow(posterior)){
  y.cred[i, ] = posterior[i,"a"] + posterior[i,"b"]*x.pred + posterior[i,"c"]*x.pred^2
```

```
}

plot(df)
for(i in 1:100){
  lines(x.pred, y.cred[i, ], col=adjustcolor("red", alpha.f=0.3))
}
```



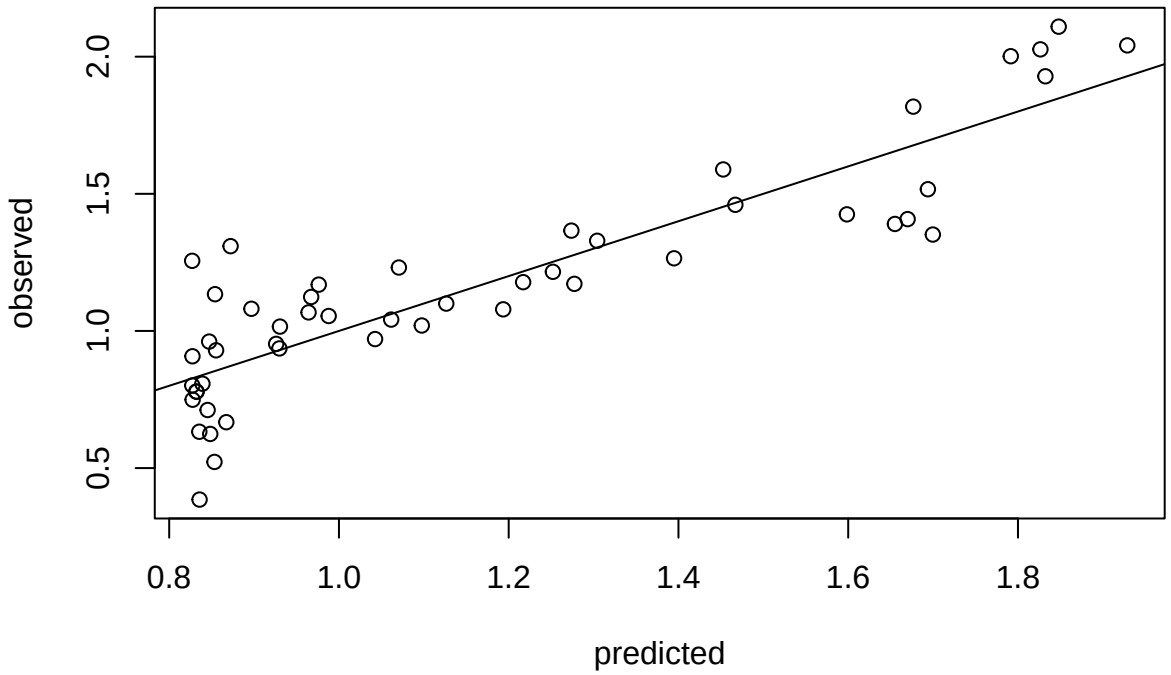

```
plot(df)

y.cred.mean = apply(y.cred, 2, function(x) mean(x))
lines(x.pred, y.cred.mean, col="red", lwd=2)

y.cred.q05 = apply(y.cred, 2, function(x) quantile(x, probs=0.05))
lines(x.pred, y.cred.q05, col="red", lwd=2, lty=2)

y.cred.q95 = apply(y.cred, 2, function(x) quantile(x, probs=0.95))
lines(x.pred, y.cred.q95, col="red", lwd=2, lty=2)

y.pred = matrix(0, nrow=nrow(posterior), ncol=length(x.pred))
for(i in 1:nrow(posterior)){
  y.pred[i, ] = rnorm(n=length(x.pred), mean=y.cred[i, ], sd=rep(posterior[i, "sigma"],length(x.pred)) )
}

y.pred.mean = apply(y.pred, 2, function(x) mean(x))
lines(x.pred, y.pred.mean, col="blue", lwd=2)

y.pred.q05 = apply(y.pred, 2, function(x) quantile(x, probs=0.05))
lines(x.pred, y.pred.q05, col="blue", lwd=2, lty=2)

y.pred.q95 = apply(y.pred, 2, function(x) quantile(x, probs=0.95))
lines(x.pred, y.pred.q95, col="blue", lwd=2, lty=2)
```

```
legend("topleft", legend=c("90% credible","90% prediction"), lwd=c(2,2), col=c("red","blue"), bty="n", l
```



Observed vs. predicted

```
x.pred = df$x
y.cred = matrix(0, nrow=nrow(posterior), ncol=length(x.pred))

for(i in 1:nrow(posterior)){
  y.cred[i, ] = posterior[i,"a"] + posterior[i,"b"]*x.pred + posterior[i,"c"]*x.pred^2
}

y.cred.mean = apply(y.cred, 2, function(x) mean(x))

plot(y.cred.mean, df$y, ylab="observed", xlab="predicted")
abline(0,1)
```




```
plot(y.cred.mean, df$y-y.cred.mean, ylab="residuals", xlab="predicted")
abline(0,0)
```