HW11

(a) The probabilistic learning model used here we was $P(y_{jt} = 1) = \text{logit}(\beta_{0jt} + \beta_{1jt}X_{1jt} + \beta_{2jt}X_{2jt})$ where j indexes dogs (from 1 to 30), t indexes trials (from 1 to 25), X_{1jt} is the number of avoid trials so far and X_{2jt} is the number of shock trials so far. In addition, we fit another leaerning model ($P(y_{jt} = 1) = \text{logit}(\beta_{0jt} + \beta_{1jt}X_{1jt} + \beta_{2jt}X_{2jt})$) where X_{3jt} represents the number of trials since the previous shock trial.

(b) In order to better estimate the βs, we used a hierarchical model where $\beta \sim N(\alpha, \tau)$ and $\alpha \sim N(0, 100)$

(c) First, we got the data into a usable form:

```
library(tidyr)
library(rstan)
library(ggplot2)
setwd("~/Documents/BDA/Homework 11")
y1 <- read.table("dogs.txt", header = TRUE, skip = 1)
y <- ifelse (y1[,]=="S",1,0)
n_dogs <- nrow(y)
n_trials <- ncol(y)
```

Then we build the two stan models:

```
data {
  int<lower=0> n_dogs;
  int<lower=0> n_trials;
  int<lower=0,upper=1> y[n_dogs,n_trials];
}
parameters {
  vector[4] alpha;
  vector<lower=0>[4] tau;
  vector[4] beta[n_dogs];
}
transformed parameters {
  matrix[n_dogs,n_trials] n_avoid;
  matrix[n_dogs,n_trials] n_shock;
  matrix[n_dogs,n_trials] n_trials_since_shock;
  matrix[n_dogs,n_trials] p;

  for (j in 1:n_dogs) {
    n_avoid[j,1] <- 0;
    n_shock[j,1] <- 0;
    n_trials_since_shock[j,1] <- 0;
    for (t in 2:n_trials) {
      n_avoid[j,t] <- n_avoid[j,t-1] + 1 - y[j,t-1];
      n_shock[j,t] <- n_shock[j,t-1] + y[j,t-1];
      if (y[j,t-1] == 0) {
        n_trials_since_shock[j,t] <- n_trials_since_shock[j,t-1] + 1;
      }
      else {
        n_trials_since_shock[j,t] <- 0;
      }
    }
```

```
    for (t in 1:n_trials)
      p[j,t] <- beta[j, 1] + beta[j, 2] * n_avoid[j,t] + beta[j, 3] * n_shock[j,t] + beta[j, 4] * n_tria
  }
}
model {
  alpha ~ normal(0, 100);
  for (i in 1:n_dogs) {
    beta[i] ~ normal(alpha, tau);
    for (j in 1:n_trials)
      y[i,j] ~ bernoulli_logit(p[i,j]);
  }
}

data {
  int<lower=0> n_dogs;
  int<lower=0> n_trials;
  int<lower=0,upper=1> y[n_dogs,n_trials];
}
parameters {
  vector[3] alpha;
  vector<lower=0>[3] tau;
  vector[3] beta[n_dogs];
}
transformed parameters {
  matrix[n_dogs,n_trials] n_avoid;
  matrix[n_dogs,n_trials] n_shock;
  matrix[n_dogs,n_trials] p;

  for (j in 1:n_dogs) {
    n_avoid[j,1] <- 0;
    n_shock[j,1] <- 0;
    for (t in 2:n_trials) {
      n_avoid[j,t] <- n_avoid[j,t-1] + 1 - y[j,t-1];
      n_shock[j,t] <- n_shock[j,t-1] + y[j,t-1];
    }
    for (t in 1:n_trials)
      p[j,t] <- beta[j, 1] + beta[j, 2] * n_avoid[j,t] + beta[j, 3] * n_shock[j,t];
  }
}
model {
  alpha ~ normal(0, 100);
  for (i in 1:n_dogs) {
    beta[i] ~ normal(alpha, tau);
    for (j in 1:n_trials)
      y[i,j] ~ bernoulli_logit(p[i,j]);
  }
}
```

We now fit these models and get parameter estimates:

```
fit_X3 <- stan("dogs_X3.stan");
fit <- stan("dogs_noX3.stan");
post <- rstan::extract(fit)
```

```
beta <- colMeans(post$beta)
alpha <- colMeans(post$alpha)
tau <- colMeans(post$tau)
post <- rstan::extract(fit_X3)
beta_X3 <- colMeans(post$beta)
alpha_X3 <- colMeans(post$alpha)
tau_X3 <- colMeans(post$tau)
```

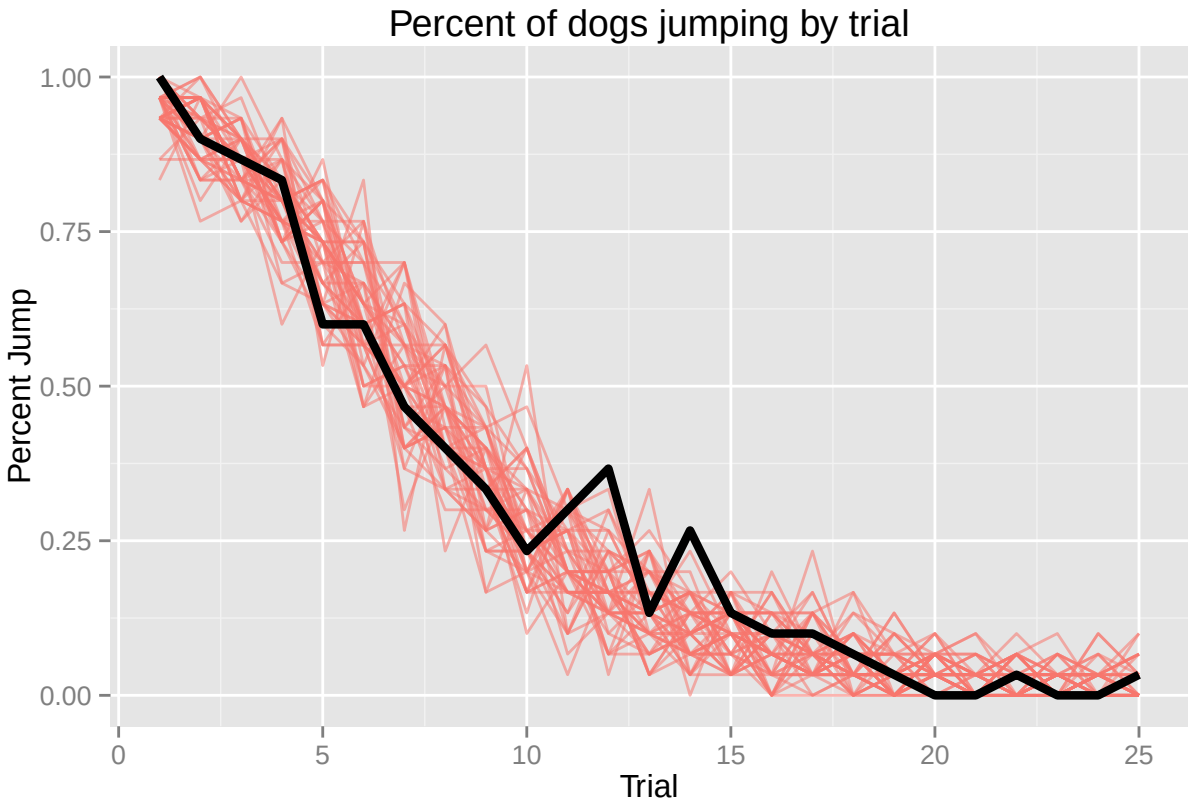
Now we do some posterior predictive checks to see if the two models fit the data:

```
invlogit <- function(x) {
  exp(x)/(1+exp(x))
}

n_sims <- 50
rep_data <- function(n_sims, n_dogs, n_trials, beta)
{
  y_rep <- array (NA, c(n_sims, n_dogs, n_trials))
  for (j in 1:n_dogs) {
    n_avoid_rep <- rep (0, n_sims)
    n_shock_rep <- rep (0, n_sims)
    n_trials_since_shock_rep <- rep (0, n_sims)
    for (t in 1:n_trials) {
      if(length(beta) == 4)
        p_rep <- invlogit (beta[j, 1] + beta[j, 2]*n_avoid_rep + beta[j, 3]*n_shock_rep + beta[j, 4]*n_
      else
        p_rep <- invlogit (beta[j, 1] + beta[j, 2]*n_avoid_rep + beta[j, 3]*n_shock_rep)
      y_rep[,j,t] <- rbinom (n_sims, 1, p_rep)
      n_avoid_rep <- n_avoid_rep + 1 - y_rep[,j,t]
      n_shock_rep <- n_shock_rep + y_rep[,j,t]
      for(s in 1:n_sims){
        if (y_rep[s,j,t] == 0) {
          n_trials_since_shock_rep[s] <- n_trials_since_shock_rep[s] + 1;
        }
        else {
          n_trials_since_shock_rep[s] <- 0;
        }
      }
    }
  }
  y_rep
}

rep_data_full <- function(n_sims, n_dogs, n_trials, tau, alpha)
{
  y_rep <- array (NA, c(n_sims, n_dogs, n_trials))
  for (j in 1:n_dogs){
    n_avoid_rep <- rep (0, n_sims)
    n_shock_rep <- rep (0, n_sims)
    n_trials_since_shock <- rep (0, n_sims)
    beta = rnorm(alpha, tau)
    for (t in 1:n_trials){
      if(length(beta) == 4)
```

```
        p_rep <- invlogit (beta[1] + beta[2]*n_avoid_rep + beta[3]*n_shock_rep + beta[4]*n_trials_since
      else
        p_rep <- invlogit (beta[1] + beta[2]*n_avoid_rep + beta[3]*n_shock_rep)
      y_rep[,j,t] <- rbinom (n_sims, 1, p_rep)
      n_avoid_rep <- n_avoid_rep + 1 - y_rep[,j,t]
      n_shock_rep <- n_shock_rep + y_rep[,j,t]
      for(s in 1:n_sims){
        if (y_rep[s,j,t] == 0) {
          n_trials_since_shock[s] <- n_trials_since_shock[s] + 1;
        }
        else {
          n_trials_since_shock[s] <- 0;
        }
      }
    }
  }
  y_rep
}

y_rep_X3 <- rep_data(n_sims, n_dogs, n_trials, beta_X3)
y_rep_full_X3 <- rep_data_full(n_sims, n_dogs, n_trials, alpha_X3, tau_X3)
y_rep <- rep_data(n_sims, n_dogs, n_trials, beta)
y_rep_full <- rep_data_full(n_sims, n_dogs, n_trials, alpha, tau)

get_means_df <- function(y, y_rep) {
  means <- data.frame(colMeans(y), row.names = 1:25)
  colnames(means) <- "data"
  for (i in 1:50)
  {
    means[, toString(i)] <- colMeans(y_rep[i,,])
  }
  means <- gather(means, sim, percent)
  means$trial <- rep(1:25, 51)
  means
}

means <- get_means_df(y, y_rep)
means_full <- get_means_df(y, y_rep_full)
means_X3 <- get_means_df(y, y_rep_X3)
means_full_X3 <- get_means_df(y, y_rep_full_X3)
```

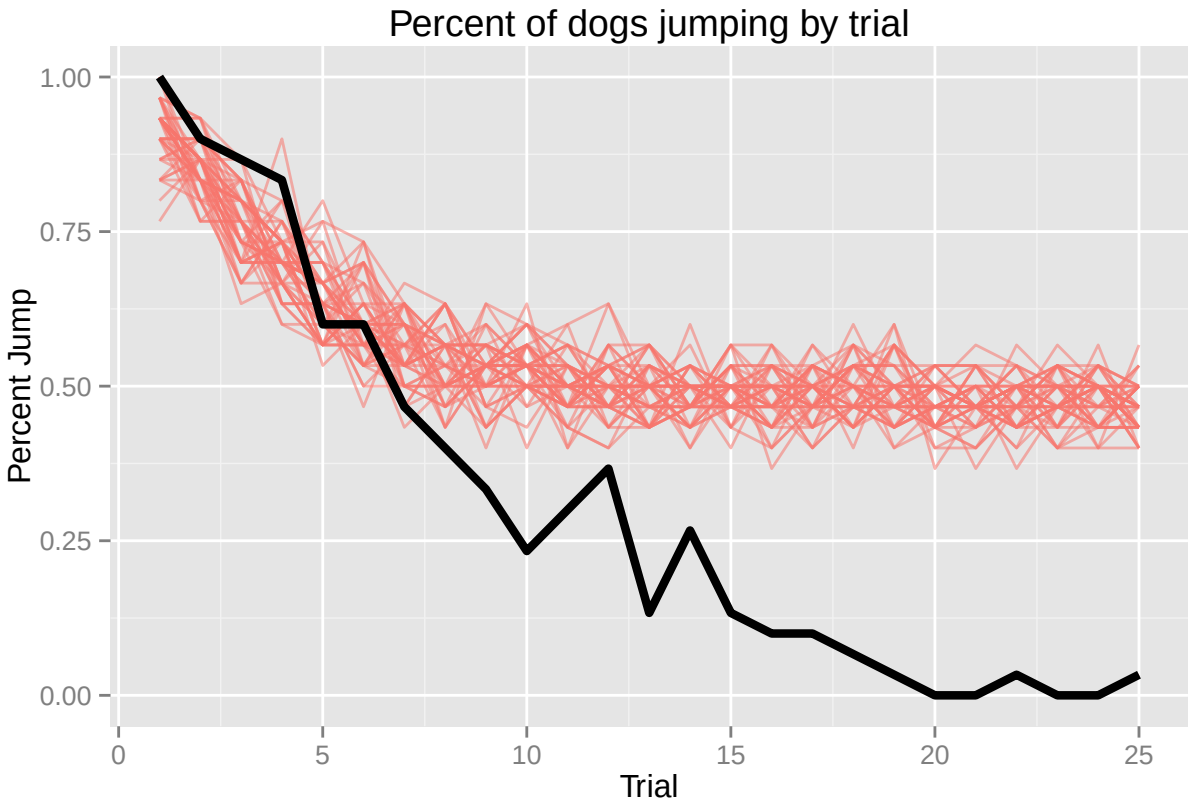
Now we plot the proportion of dogs jumping on a given trial in the real dataset and simulated datasets. Here is a plot of the data (in black) and 50 simulations from the model without X3 (in red) using the estimated betas:

```
m_plot <- ggplot(means, aes(x = trial, y = percent, group=sim, color = "red", alpha = .1)) + geom_line(
  xlab('Trial') + ylab('Percent Jump') +
  labs(title = "Percent of dogs jumping by trial") +
  geom_line(data=subset(means, sim == "data"), alpha = 1, color = "black", size=1.5)
m_plot + guides(colour = FALSE) + guides(alpha = FALSE)
```



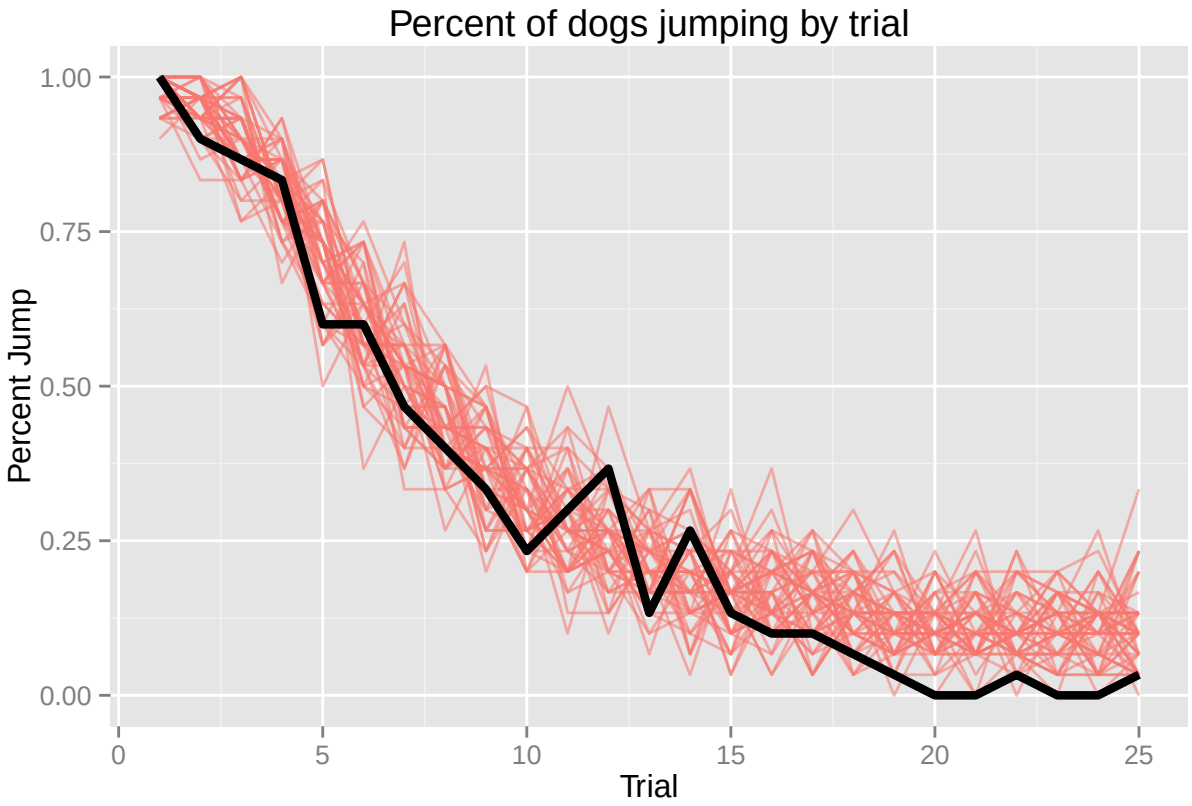
This is the same plot but with the betas drawn from the prior:

```
mf_plot <- ggplot(means_full, aes(x = trial, y = percent, group=sim, color = "red", alpha = .1)) + geom_
  xlab('Trial') + ylab('Percent Jump') +
  labs(title = "Percent of dogs jumping by trial") +
  geom_line(data=subset(means_full, sim == "data"), alpha = 1, color = "black", size=1.5)
mf_plot + guides(colour = FALSE) + guides(alpha = FALSE)
```



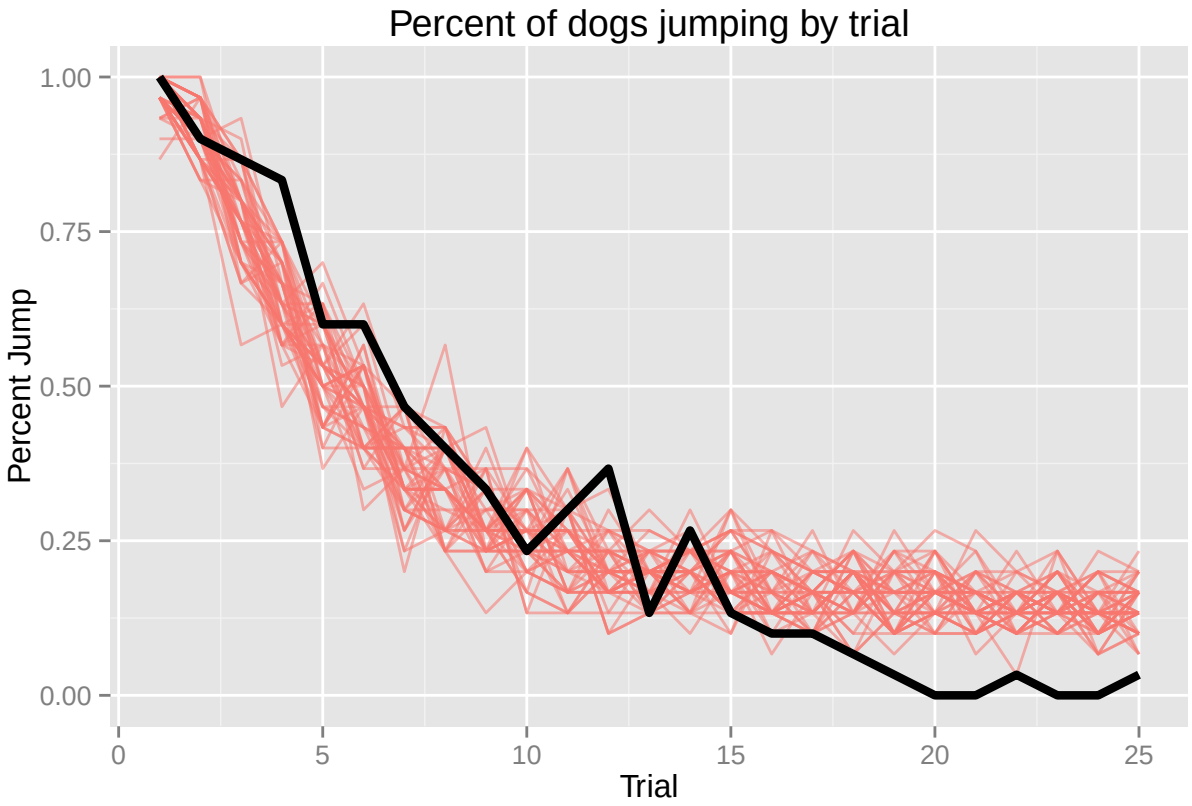
And the same plot using the model with X3 using estimated betas:

```
mX3_plot <- ggplot(means_X3, aes(x = trial, y = percent, group=sim, color = "red", alpha = .1)) + geom_l
  xlab('Trial') + ylab('Percent Jump') +
  labs(title = "Percent of dogs jumping by trial") +
  geom_line(data=subset(means_X3, sim == "data"), alpha = 1, color = "black", size=1.5)
mX3_plot + guides(colour = FALSE) + guides(alpha = FALSE)
```



And the same plot using betas drawn from the prior:

```
mX3f_plot <- ggplot(means_full_X3, aes(x = trial, y = percent, group=sim, color = "red", alpha = .1)) +
  xlab('Trial') + ylab('Percent Jump') +
  labs(title = "Percent of dogs jumping by trial") +
  geom_line(data=subset(means_full_X3, sim == "data"), alpha = 1, color = "black", size=1.5)
mX3f_plot + guides(colour = FALSE) + guides(alpha = FALSE)
```



When using the betas estimated from the data, the model without X3 predicts the end behavior well but predicts that the dogs learn too quickly while the model with X3 predicts learning well but fails to capture the dogs' almost perfect learning at the end of the experiment. However, when we draw the betas from the prior, the X3 model predicts the data slightly better (although still not very well.)

(d) Here is the true data set:

```
library(knitr)
kable(y)
```

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X
13	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	
17	1	1	1	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	0	
18	1	0	0	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	
21	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
27	1	1	1	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	
29	1	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	
30	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	
32	1	1	1	1	1	0	1	0	1	0	0	1	0	1	1	1	0	0	0	
33	1	1	1	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	
34	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	
36	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	
37	1	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	
41	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	
42	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
43	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X1
45	1	0	1	0	1	1	1	0	1	0	0	0	0	1	0	0	0	0	0	
47	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
48	1	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	
46	1	1	1	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	
49	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
50	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	
52	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
54	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	0	0	1	
57	1	1	1	1	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0	
59	1	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	
67	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
66	1	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	
69	1	1	1	1	0	0	1	1	0	0	0	1	0	1	0	1	0	1	0	
71	1	1	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	

Here are datasets using the estimated betas for the model without X3:

```
kable(y_rep[1,,])
```

1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1
1	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	1	1	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
1	1	0	1	1	0	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	1	1
1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

And with X3:

```
kable(y_rep_X3[1,,])
```

1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	0	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0
1	1	1	1	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	0	0	0
1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0
1	1	1	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
1	1	1	1	1	1	0	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
1	1	1	0	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0
1	1	1	1	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1	1
1	1	1	1	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
1	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0

And here are simulated datasets using betas drawn from the prior for the model without X3:

```
kable(y_rep_full[1,,])
```

1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	1	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	0	0	1
1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1
1	1	1	1	1	0	1	0	1	0	1	1	1	0	1	1	1	0	1	0	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	1
1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
1	1	0	1	0	1	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

And with X3:

```
kable(y_rep_full_X3[1,,])
```

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	0	1	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1
1	1	1	1	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	1	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0
1	1	1	0	0	1	0	1	1	0	0	0	0	0	0	1	1	0	1	1	0	0	1	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	1	0	1	1	0	0	1	0	0	1	0	1	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0	1	1	0

(Sorry I couldn't figure out how to get them all on one page!) You can see even in the single datasets that the learning curves are way too shallow, especially when the betas are drawn from the prior