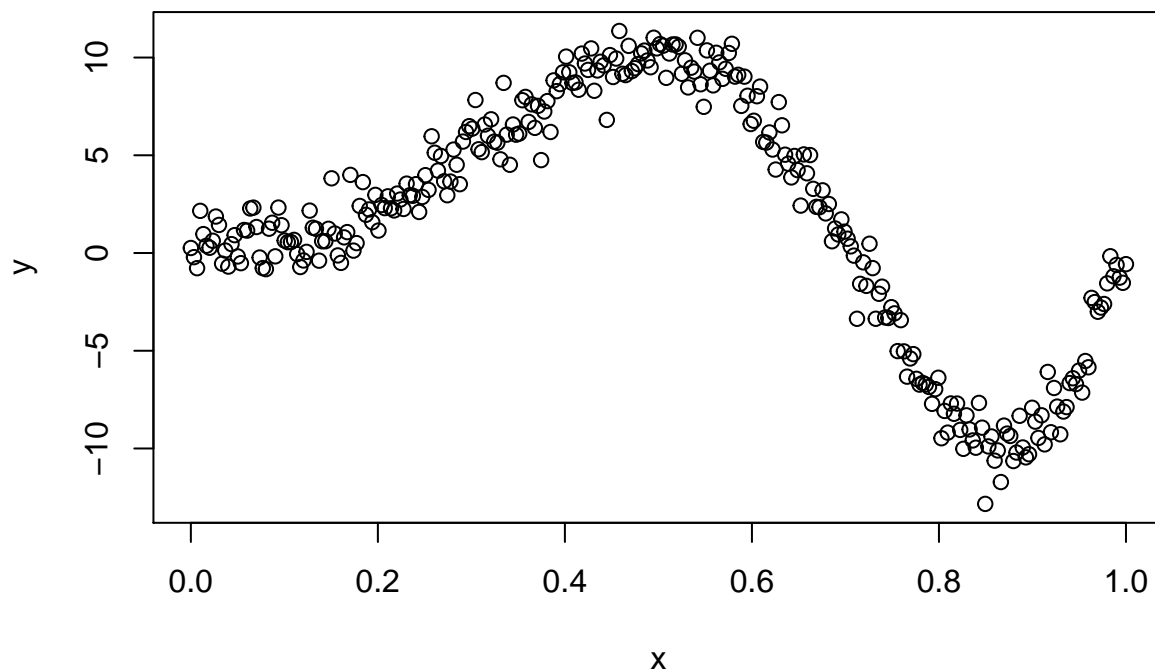


Bayesian Regression Splines with Reversible Jump MCMC

Andy Shen, Devin Francom

Let's say you want to fit a model using some wiggly data. Maybe

```
n<-300
x<-seq(0,1,length.out=n)
y<-sin(2*pi*x^2)*10+rnorm(n)
plot(x,y)
```



One way to fit a model to data like this is to come up with a linear basis and fit a linear model using the basis as the X matrix (which we will call B). People often use splines as a basis. The simplest set of spline basis functions would be to make the i th basis function (i.e., the i th column of B) look like

$$B_{ij} = [s_i(x_j - t_i)]_+$$

where $s \in \{-1, 1\}$, which we'll call the sign, and t is a value in the domain of x , which we will call a knot. Also, $[a]_+ = \max(0, a)$.

Try some combinations of s and t to see what your basis functions look like, and what the corresponding linear model fit looks like (using the `lm` function or your Bayesian linear model code). Try with different numbers of basis functions, also.

Bayesian Spline with Random Knots

```
set.seed(12)
source("mcmc_spline_rj.R")

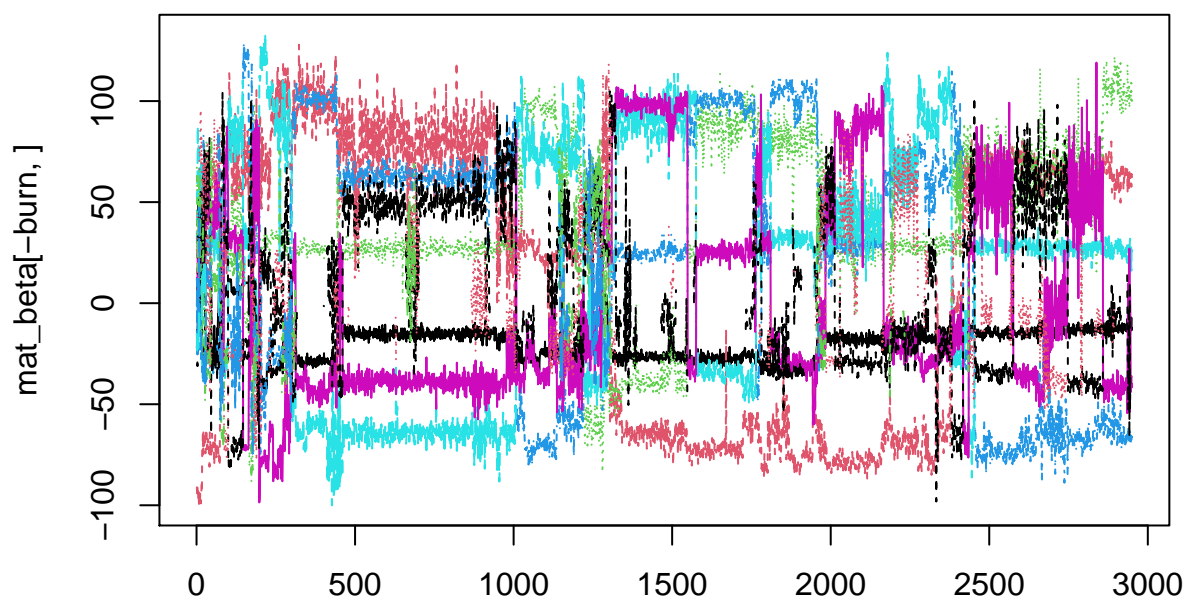
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##   filter, lag
## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union

iterations <- 3000
res <- mcmc_spline(its = iterations)

mat_beta <- res[[1]]
mat_beta <- mat_beta[,colSums(is.na(mat_beta)) != nrow(mat_beta)] %>% as.matrix()
mat_sig <- res[[2]]
mat_t <- res[[3]]
mat_t <- mat_t[,colSums(is.na(mat_t)) != nrow(mat_t)] %>% as.matrix()
mat_s <- res[[4]]
mat_s <- mat_s[,colSums(is.na(mat_s)) != nrow(mat_s)] %>% as.matrix()
X_curr <- res[[5]]

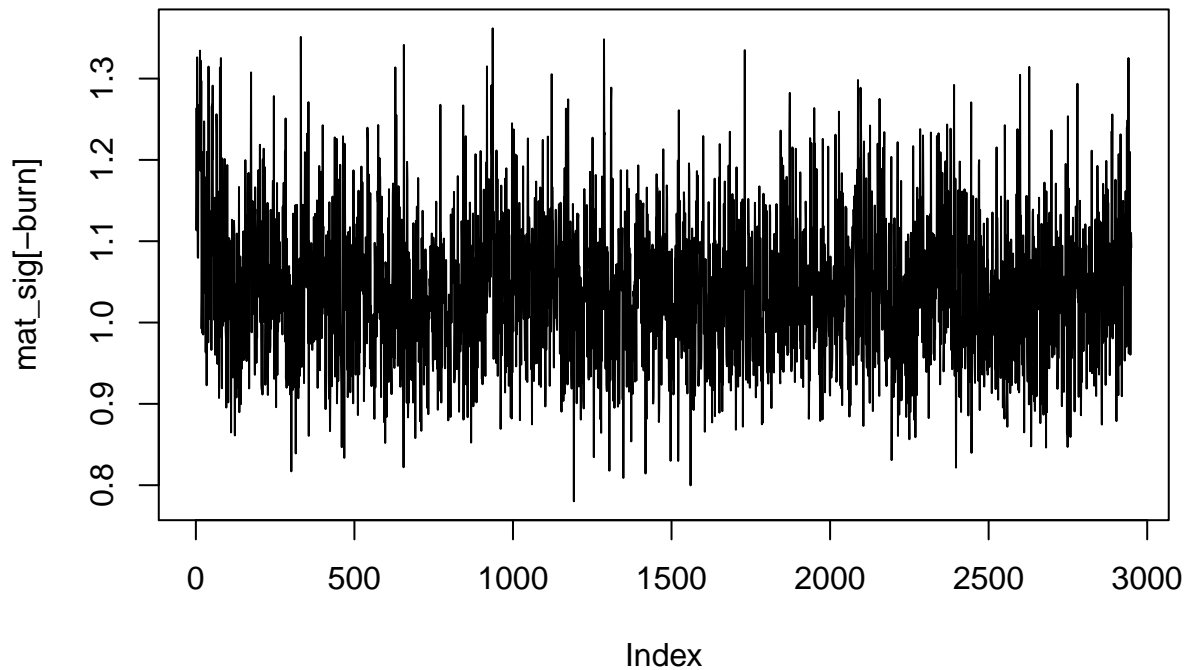
burn <- 1:50
matplot(mat_beta[-burn,], type = "l", main = "Plot of Regression Coefficients")
```

Plot of Regression Coefficients



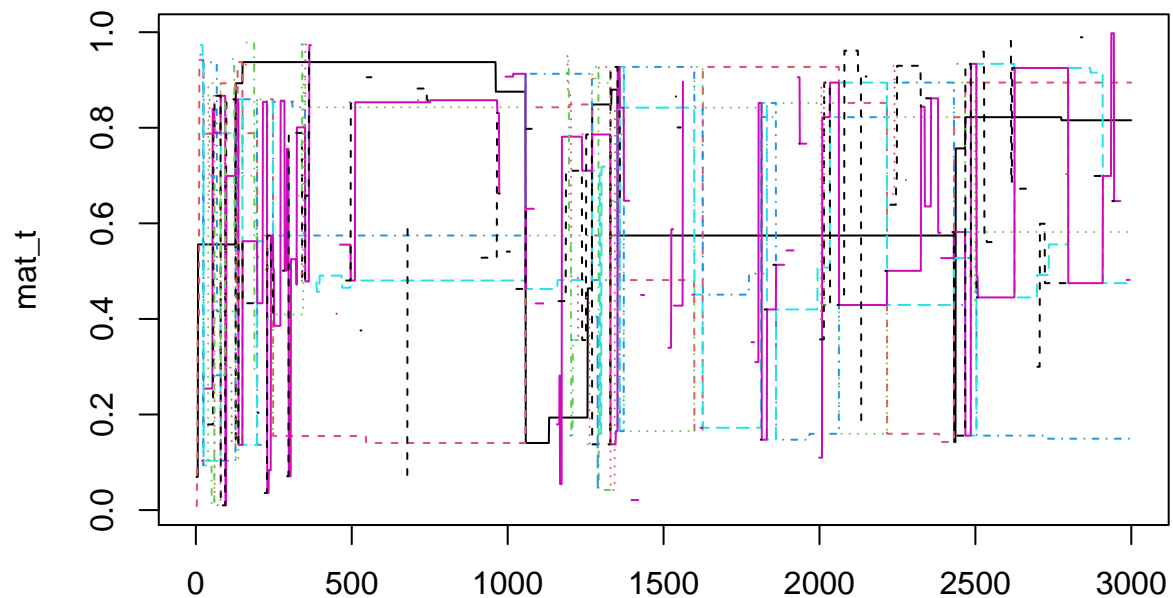
```
plot(mat_sig[-burn], type = "l", main = "Plot of Sigma^2")
```

Plot of Sigma^2



```
matplot(mat_t, type = "l", main = "Plot of Knot Locations")
```

Plot of Knot Locations



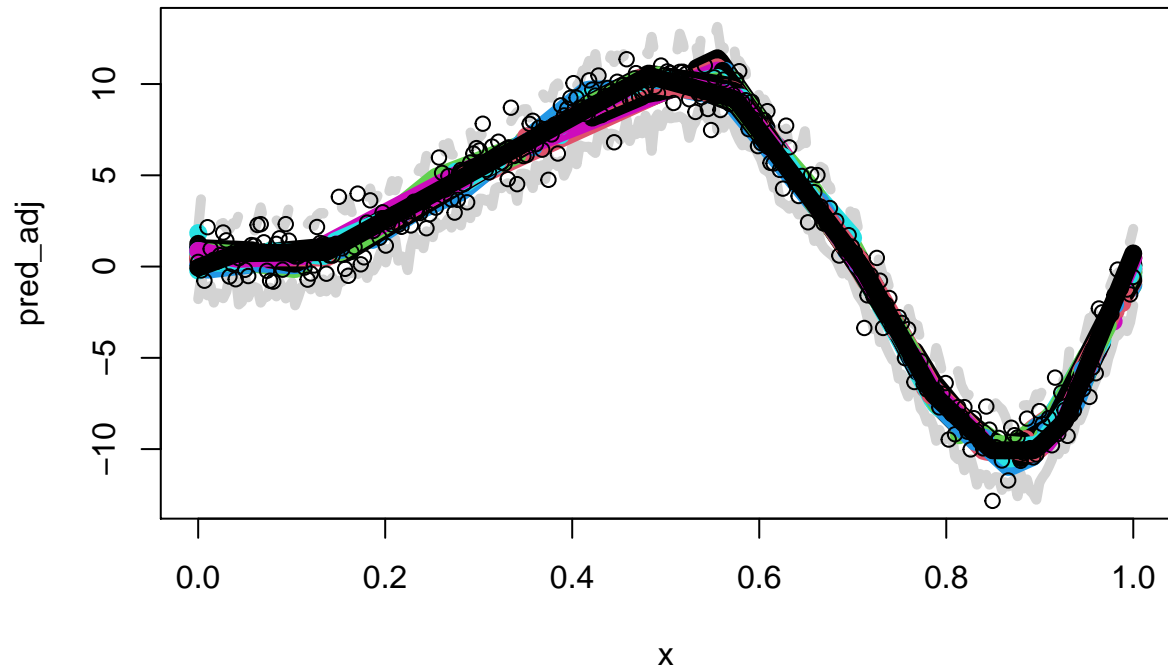
```
knotnum <- ncol(mat_t)
mean.pred <- matrix(NA, nrow = iterations, ncol = length(x))
pred <- mean.pred
for(p in 1:iterations) {
  splb <- spline.basis(nknot = knotnum, knots = mat_t[p,], signs = mat_s[p,])
  mean.pred[p,] <- splb %*% mat_beta[p,]
```

```

    pred[p,] <- mean.pred[p,] + rnorm(length(x), sd = sqrt(mat_sig[p]))
  }
  mean.pred <- t(mean.pred)

  pred_adj <- t(apply(pred, 2, quantile, probs = c(0.025, 0.975), na.rm = TRUE))
  matplot(x, pred_adj, col = "lightgrey", lwd = 5, type = "l")
  matplot(x, mean.pred, type = "l", lwd = 9, add = TRUE)
  points(x, y)

```



```

plot(x, rowMeans(mean.pred, na.rm = TRUE), lwd = 8, type = "l", col = "royalblue")
points(x, y)

```

