BARS Package Test Output

Developed by Andy Shen

Introduction and Installation

Hello! Thank you for taking the time to test the `bars` package in R. This package lays out the foundation for Bayesian Adaptive Regression Splines, a common tool used to fit nonlinear data from a Bayesian perspective. This algorithm utilizes various MCMC techniques, such as Reversible-Jump algorithms and Gibbs sampling to adaptively predict the optimal number of knots and their locations without overfitting the data. I am still writing the Vignette and will upload it to the repository when it is completed.

Please run the code below and ensure all outputs appear exactly as the file `TEST_FILE.pdf` which can be found on the GitHub repository for this project: https://github.com/aashen12/BARS. In its entirety, the file should take around 1 minute to run completely, depending on your processor speed.

If you encounter any errors that you cannot fix after multiple attempts, please send me a screenshot of the full error message as well as the entire pdf/Rmd file with the error. You can email me (Andy) at aashen@ucla.edu. Please do not hesitate to email me if you have any questions as well.

Thanks again, and enjoy!

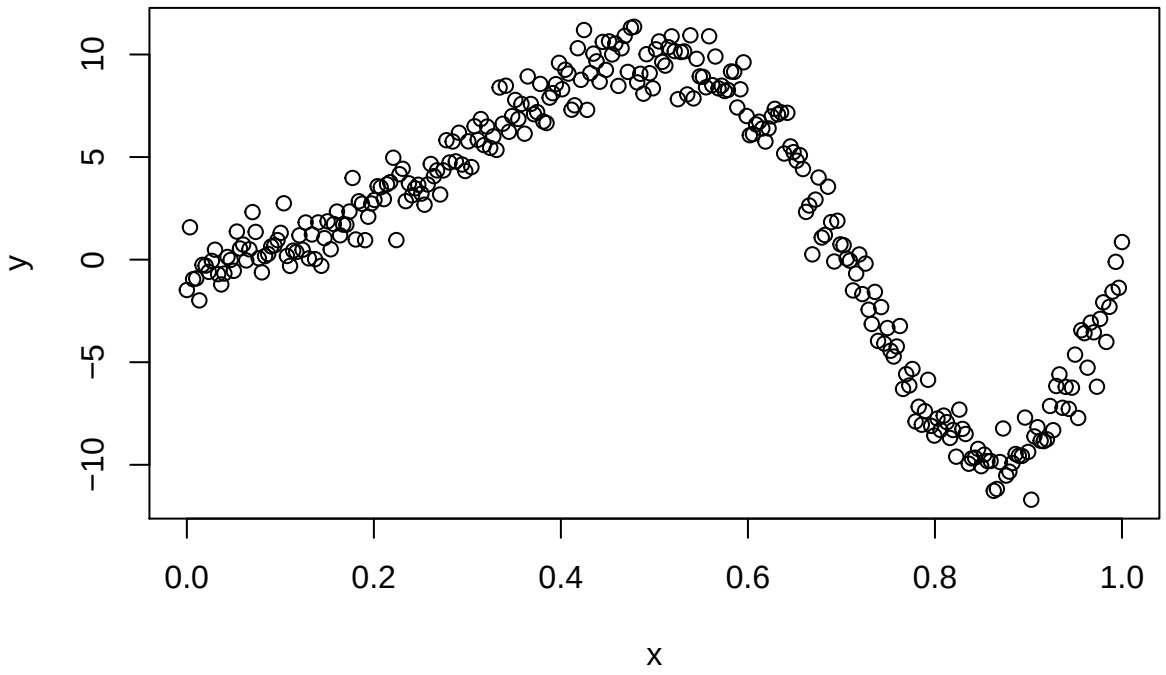
To install the `bars` package, please run the following command in your RStudio console:

```
devtools::install_github("aashen12/BARS") # only run this command once
```

Your name here

Data Generation

```
rm(list = ls())
set.seed(12) # feel free to modify once you get the code working
n <- 300
x <- seq(0,1,length.out=n)
y <- sin(2*pi*x^2)*10+rnorm(n)
plot(x, y)
```



Feel free to play around with the data and modify however you like once you get this code working.

Bayesian Spline with Random Knots

```
library(bars) #crucial step in calling the package
library(mvtnorm) #this line is necessary...for now
nmcmc <- 5000
res <- bars(its = nmcmc, verbose = TRUE)
```

```
## Iteration number 1000    sigma^2 = 0.8929665
## Iteration number 2000    sigma^2 = 0.8660429
## Iteration number 3000    sigma^2 = 0.9946919
## Iteration number 4000    sigma^2 = 1.086504
## Iteration number 5000    sigma^2 = 1.050121
```

Results

Beta

```
beta <- res$beta
beta <- beta[,colSums(is.na(beta)) != nrow(beta)]
round(beta[nrow(beta),], 2)
```

```
##  [1]  11.86  61.17 -48.31  51.35  64.75 -23.57 -34.99     NA     NA     NA
## [11]     NA     NA
```

Sigma

```
sig <- res$sig
tail(round(sig, 2))
```

```
## [1] 0.99 0.96 1.07 1.05 1.10 1.05
```

Knots

```
knots <- res$knots
knots <- knots[,colSums(is.na(knots)) != nrow(knots)]
round(knots[nrow(knots),], 2)
```

```
##  [1] 0.82 0.63 0.92 0.87 0.57 0.47   NA   NA   NA   NA   NA
```

```
res$knots_total
```

```
## [1] 6
```

Signs

```
signs <- res$signs
signs <- signs[,colSums(is.na(signs)) != nrow(signs)]
round(signs[nrow(signs),], 2)
```

```
##  [1]  1  1  1  1 -1  1 NA NA NA NA NA
```

Basis Functions

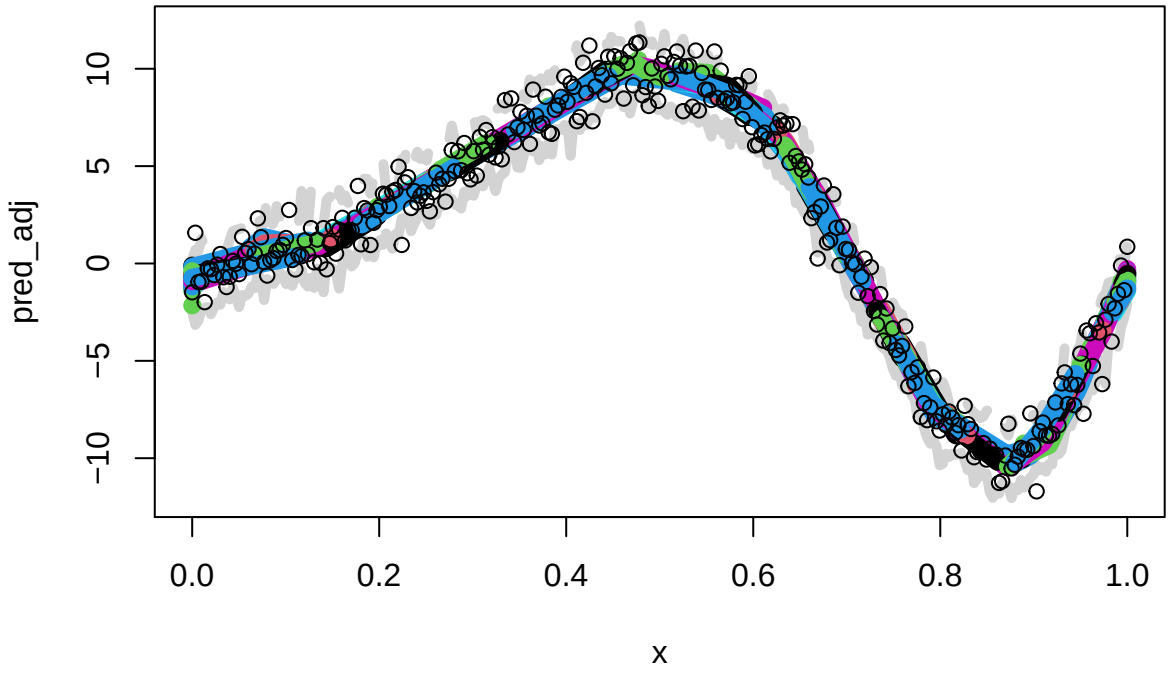
```
X <- res$X
tail(round(X, 2))
```

```
##          basis_vec basis_vec basis_vec basis_vec basis_vec basis_vec
## [295,] 1      0.17      0.35      0.06      0.11         0      0.51
## [296,] 1      0.17      0.36      0.06      0.12         0      0.51
## [297,] 1      0.17      0.36      0.07      0.12         0      0.52
## [298,] 1      0.18      0.36      0.07      0.12         0      0.52
## [299,] 1      0.18      0.37      0.07      0.13         0      0.52
## [300,] 1      0.18      0.37      0.08      0.13         0      0.53
```

Plotting Predicted Values

```
knotnum <- ncol(knots)
mean.pred <- matrix(NA, nrow = nmcmc, ncol = length(x))
pred <- mean.pred
for(p in 1:nmcmc) {
  splb <- spline.basis(nknot = knotnum, knots = knots[p,], signs = signs[p,])
  mean.pred[p,] <- splb %*% beta[p,]
  pred[p,] <- mean.pred[p,] + rnorm(length(x), sd = sqrt(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 of Averages

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