gamselBayes: Bayesian generalized additive model selection including a fast variational option

Virginia X. He and Matt P. Wand

University of Technology Sydney

1st September, 2023

1 Introduction

The R package gamselBayes implements algorithms for Bayesian generalized additive model selection developed by and described in He & Wand (2022). The underlying model is similar to, and based on, that developed by Chouldechova & Hastie (2015), and implemented in the R package gamsel (Chouldechova & Hastie, 2018).

The default fitting and inference algorithm in gamselBayes is a Markov chain Monte Carlo scheme. For problems of moderate size, the run time with this default approach is likely to be acceptable. However, it is well-known that Markov chain Monte Carlo approaches do not scale well to very large problems. With such circumstances in mind, the gamselBayes package also provides the option of using a faster mean field variational Bayes algorithm for generalized additive model selection.

In Sections 2–7 of this vignette we work through some illustrative examples involving simulated and actual data. Section 8 describes limitations of the gamselBayes and contains some trouble-shooting advice.

2 Ilustrations with Simulated Data

We start with two data sets simulated from Gaussian response additive models. Such examples have the advantage of explaining the use of gamselBayes in simple terms, and also allowing comparison with true functions from which the data are simulated.

2.1 All Candidate Predictors Continuous

The following code generates data corresponding to the additive model

$$y_i \overset{\text{ind.}}{\sim} N\Big(\sum_{j=1}^{6} f_j(x_{ji}), \sigma^2\Big), \quad 1 \le i \le n, \tag{1}$$

with $n = 500$ and $\sigma = 0.15$:

```
> sigmaTrue <- 0.15
> f1True <- function(x) return(0.96*x)
> f2True <- function(x) return(0.5*(pnorm(6*x - 3) + 1))
> f3True <- function(x) return(0.5*(sin(3*pi*x) + 1))
> f4True <- function(x) return(0.5*(0.04*cosh(x^3 - 9*x^2 + 4) + 1))
> set.seed(1) ; n <- 500
> x1 <- runif(n) ; x2 <- runif(n) ; x3 <- runif(n)
> x4 <- runif(n) ; x5 <- runif(n) ; x6 <- runif(n)
> y <- rnorm(n,f1True(x1) + f2True(x2) + f3True(x3) + f4True(x4),sigmaTrue)
> Xgeneral <- data.frame(x1,x2,x3,x4,x5,x6)
```

The data for all 6 candidate predictors $x_1, \ldots, x_6$ are stored in the data frame Xgeneral. The response data are stored in the vector y.

These data are simulated from f_j functions such that:

$$f_1 \text{ is linear;} \quad f_2,\ f_3,\ \text{and } f_4 \text{ are non-linear;} \quad f_5 \text{ and } f_6 \text{ are zero.}$$

2.1.1 Analysis Using the Default Method: Markov Chain Monte Carlo

The main function in the gamselBayes package is gamselBayes(). The default gamselBayes() fit object, labeled fit1, is obtained via:

```
> library(gamselBayes) ; fit1 <- gamselBayes(y = y,Xgeneral = Xgeneral)
```

A quick look at the estimated effect types is achieved via:

```
> effectTypesVector(fit1)
```

```
        x1          x2          x3          x4          x5          x6
  "linear" "nonlinear" "nonlinear" "nonlinear"      "zero"      "zero"
```

To obtain a tabulated version of the same information, issue:

```
> effectTypes(fit1)
```

```
   ---------------------------------------------
   Predictor selected as having a linear effect:
   ---------------------------------------------

   x1

   -------------------------------------------------
   Predictors selected as having non-linear effects:
   -------------------------------------------------

   x2 x3 x4
```

This output shows that, for this example, gamselBayes() correctly estimates the effect of x_1 to be linear, the effects of x_2, x_3 and x_4 to be non-linear and the effects of x_5 and x_6 to be zero.

A Bayesian inferential summary of the linear effect coefficient is achieved using:

```
> summary(fit1)
```

```
      posterior mean   95% credible interval
   x1        0.97612        0.93155    1.026
```

and is in keeping with the true value of the coefficient, 0.96, that generated the data.

Visualisation of the non-linear estimated effects is obtained using:

```
> plot(fit1)
```

and is shown in Figure 1. Issuing code such as the following can be used to compare the estimates shown in Figure 1 with the true functions:

```
> par(mfrow=c(2,2)) ; xg <- seq(0,1,length = 1001)
> f2g <- f2True(xg) ; f3g <- f3True(xg) ; f4g <- f4True(xg)
> plot(xg,f2g,type = "l") ; plot(xg,f3g,type = "l") ; plot(xg,f4g,type = "l")
```

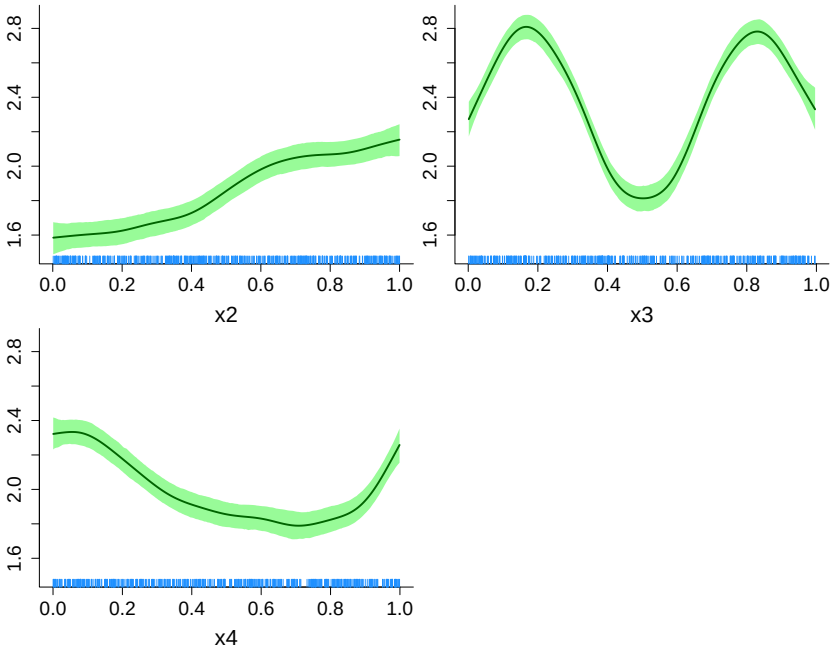

Figure 1: *The plots produced by the command* plot(fit1) *for the first* gamselBayes() *fit, stored as* fit1*, to the data simulated according to (1). The curves are estimates of f_2, f_3 and f_4. The convention of* plot() *for* gamselBayes() *is to display relevant slices of the additive model fit, for which each of the other functions are evaluated at the median value of their predictor. The shaded regions correspond to pointwise 95% credible intervals. The rugs at the base of each plot show values of each predictor.*

Whenever the default Bayesian inference method is used then it is important to keep in mind that the results depend on behaviour of the Markov chain Monte Carlo samples, also known as *chains*. The function checkChains() facilitates a cursory graphical check of the particular chains and the command:

```
> checkChains(fit1)
```

leads to the graphic shown in Figure 2. This graphic shows that key chains for each predictor are well-behaved and that the resultant Bayesian inference is quite trustworthy.

Details on the graphics produced by the checkChains() function are provided by the command:

```
> help(checkChains)
```

2.1.2 Analysis Using a Faster Alternative Method: Mean Field Variational Bayes

The default method for Bayesian inference is Markov chain Monte Carlo. For large sample sizes and dimensions, this approach can be slow to compute. To help mitigate this problem gamselBayes() offers an alternative faster approach based on mean field variational Bayesian approximate inference. The relevant argument specification is method = "MFVB" and is illustrated for this section's example via:

```
> fit2 <- gamselBayes(y = y,Xgeneral = Xgeneral,method = "MFVB")
```

The estimated effect types are obtained using:

```
> effectTypes(fit2)
```

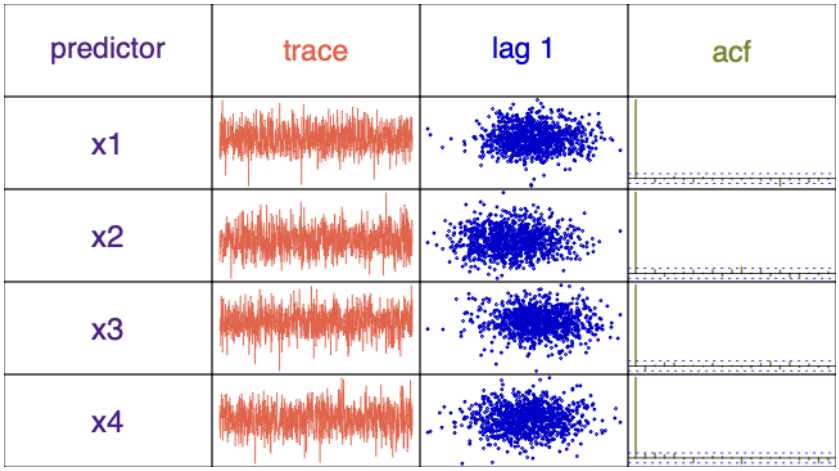

Figure 2: *The graphic produced by the command* checkChains(fit1) *for the* gamselBayes() *fit object* fit1. *For the predictor, x_1, selected as having a linear effect the chain is the coefficient of this predictor. For the predictors, x_2, x_3, x_4, selected as having non-linear effects the chains corresponds to vertical slices of the non-linear fits at the median of each predictor. The second column is a time series (trace) plot of the chain. The third column is a scatterplot of the chain values against their previous (lag 1) values. The fourth column is the sample autocorrelation function (acf), based on the* R *function* acf().

```
-------------------------------------------
Predictor selected as having a linear effect:
-------------------------------------------

x1

-------------------------------------------------
Predictors selected as having non-linear effects:
-------------------------------------------------

x2 x3 x4 x6
```

This output shows that, for this example, gamselBayes() with method = "MFVB" correctly estimates the effect of x_1 to be linear, the effects of x_2, x_3 and x_4 to be non-linear and the effect of x_5 to be zero. The effect of x_6 is incorrectly estimated to be non-linear.

A Bayesian inferential summary of the linear effect coefficient is achieved using:

```
> summary(fit2)

   posterior mean   95% credible interval
x1        0.98076        0.93034   1.0314
```

Visualisation of the non-linear estimated effects is obtained using:

```
> plot(fit2)
```

and is shown in Figure 3.

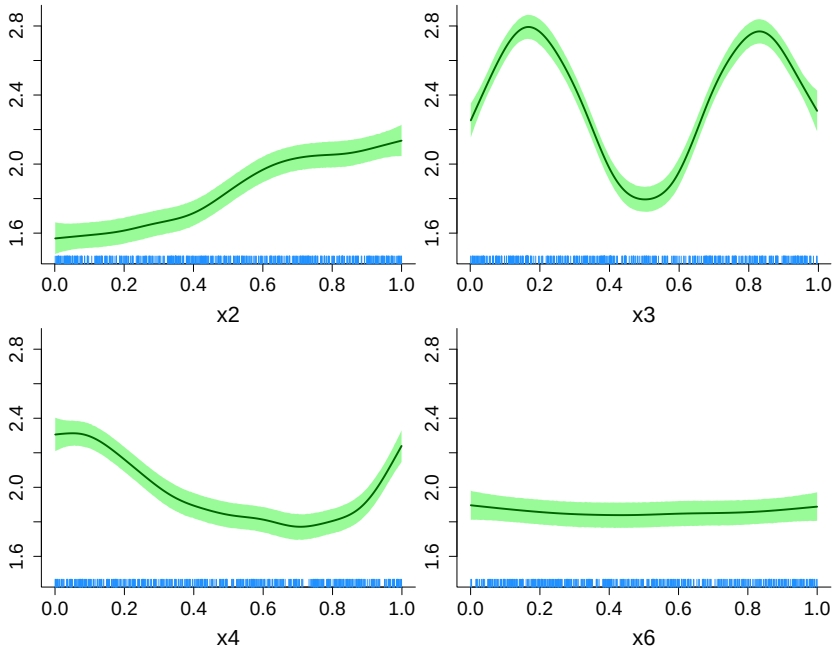

Figure 3: *The plots produced from the command* plot(fit2) *for the* gamselBayes() *fit with* method = "MFVB", *stored as* fit2, *to the data simulated according to (1). The curves are vertically centred estimates of* f_2, f_3, f_4 *and* f_6. *The shaded regions correspond to pointwise 95% credible intervals. The rugs at the base of each plot show values of each predictor.*

2.2 Candidate Predictors Continuous Both Continuous and Binary

We now consider a simulated data example where not all of the predictors are continuous. Three binary predictors, x_7, x_8 and x_9, are added so that the model becomes:

$$y_i \overset{\text{ind.}}{\sim} N\Big(\sum_{j=1}^{9} f_j(x_{ji}), \sigma^2\Big), \quad 1 \le i \le n, \tag{2}$$

Data from model (2) are generated according to:

```
> f7True <- function(x) return(-0.78*x)
> f8True <- function(x) return(0.53*x)
> set.seed(1) ; n <- 500
> x7 <- rbinom(n,1,0.5) ; x8 <- rbinom(n,1,0.5) ; x9 <- rbinom(n,1,0.5)
> y <- rnorm(n,f1True(x1) + f2True(x2) + f3True(x3) + f4True(x4)
+            + f7True(x7) + f8True(x8),sigmaTrue)
> Xgeneral <- data.frame(x1,x2,x3,x4,x5,x6)
> Xlinear <- data.frame(x7,x8,x9)
```

where the data frame Xlinear corresponds to a design matrix that contains predictors that can only have a zero or linear effect.

The functions $f_1, \ldots, f_6$ are the same as in the previous example and the code for

f1True(),...,f4True()

still applies. The functions f_7 and f_8 are linear and non-zero. The function f_9 is zero.

The call to gamselBayes() should now use the Xlinear argument as follows:

```
> fit3 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral)
```

The estimated effect types are tabulated using:

```
> effectTypes(fit3)
```

```
-------------------------------------------
Predictors selected as having linear effects:
-------------------------------------------

x1 x7 x8

-----------------------------------------------
Predictors selected as having non-linear effects:
-----------------------------------------------

x2 x3 x4
```

A Bayesian inferential summary of the linear effect coefficients is achieved using:

```
> summary(fit3)

   posterior mean   95% credible interval
x1        0.97223         0.92644  1.01820
x7       -0.76559        -0.79171 -0.73932
x8        0.54234         0.51673  0.57052
```

These are in keeping with the true value of the coefficients, 0.96, −0.78, 0.53 that generated the data.

The command:

```
> checkChains(fit3)
```

leads to the graphic shown in Figure 4. The Figure 4 graphic shows good behaviour of the

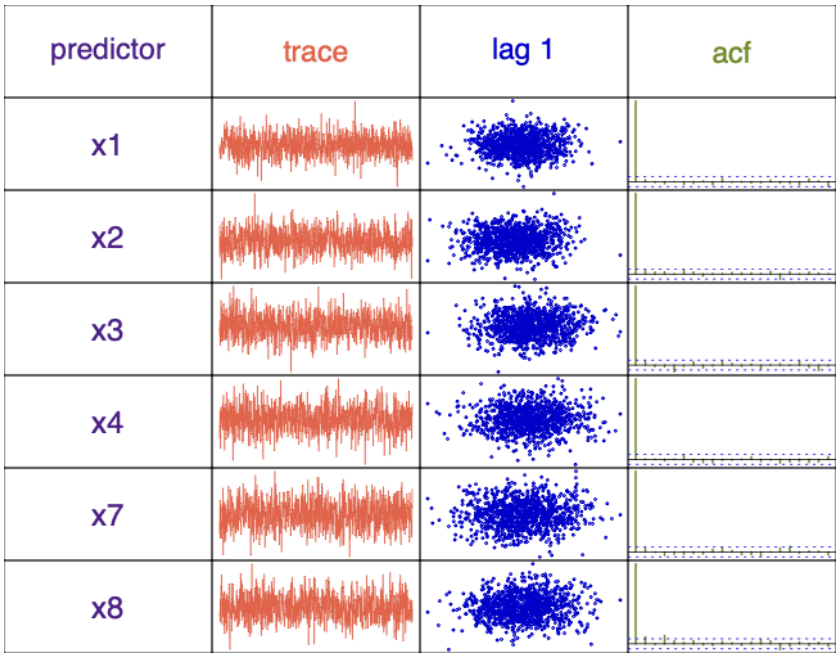

Figure 4: *The graphic produced by the command* checkChains(fit3) *for the* gamselBayes() *fit object* fit3. *The Figure 2 caption provides full details on* checkChain() *graphics.*

chains.

3 Analysis of California Schools Data

The R package Ecdat (Crossaint, 2020) contains a data frame named Caschool. The data frame contains data on 14 variables for 420 school districts in the state of California, U.S.A. The following code loads in the data and produces descriptions of each of the variables:

```
> library(Ecdat) ; data(Caschool) ; help(Caschool)
```

Two outcome variables are

mathscr average mathematics score for the district,

readscr average reading score for the district.

We will use the first of these as a response variable. Ten of the remaining 11 variables in Caschool are continuous. These are:

enrltot total enrolment,

teachers number of teachers,

calwpct percentage of students qualifying for the CalWORKS welfare programme,

mealpct percentage of students qualifying for reduced-price lunches,

computer number of computers,

compstu number of computers per student,

expnstu expenditure per student,

str student teacher ratio,

avginc average income of the district,

elpct percentage of English learners.

The other variable, Caschool[,"grspan"] is categorical and codes whether the school has grades spanning from kindergarten to year 6 or from kindergarten to year 8. The following binary variable is obtained for use in gamselBayes():

```
> upToYear8 <- as.numeric(Caschool[,"grspan"]=="KK-08")
```

3.1 Analysis for Raw Data

We are ready to set up the response vector y and the Xlinear and Xgeneral predictor variable data frames, with the following commands:

```
> y <- Caschool[,c("mathscr")]
> Xlinear <- data.frame(upToYear8)
> Xgeneral <- Caschool[,c("enrltot","calwpct","mealpct","compstu",
+                         "expnstu","str","elpct","avginc")]
```

Note that Xgeneral contains each of the available continuous predictors except for Caschool[,"teachers"] and Caschool[,"computer"], since they are simple functions of other predictors. The following default call to gamselBayes() fit command:

```
> fit4 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral)
```

leads to the fit object being stored in fit4. Then:

```
> effectTypes(fit4)
```

leads to

```
-------------------------------------------
Predictors selected as having linear effects:
-------------------------------------------

avginc elpct mealpct upToYear8
```

We see that, out of the 9 candidate predictors, 4 are chosen and each of them are found to have linear effects. The selected model is a linear combination of mealpct, elpct, avginc and upToYear8. The command

```
> summary(fit4)
```

leads to the following inferential summary of the coefficients:

```
           posterior mean   95% credible interval
avginc            0.71579         0.52503  0.89461
elpct            -0.13910        -0.22407  0.00000
mealpct          -0.35041        -0.43218 -0.25332
upToYear8        -3.08800        -6.57070  0.00000
```

It shows, for example, a significant elevation in mean mathematics scores for districts with higher average incomes.

Since these results are dependent on Markov chain Monte Carlo sampling, it is prudent to perform a check of the chains using:

```
> checkChains(fit4)
```

This leads to the chain visual summary shown in Figure 5. The chains are reasonably well-

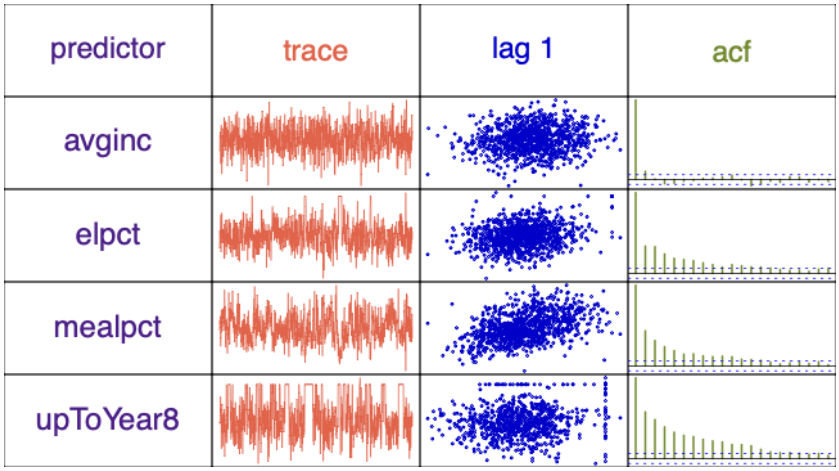

Figure 5: *The graphic produced by the command* checkChains(fit4) *for the* gamselBayes() *fit object* fit4. *The Figure 2 caption provides full details on* checkChain() *graphics.*

behaved, but it may be worth considering larger warm-up and kept chain sizes. The following command leads to a new fit with a two-fold increase in the chain sizes compared to their default values:

```
> fit5 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral,
+                control = gamselBayes.control(nWarm = 2000,nKept = 2000))
```

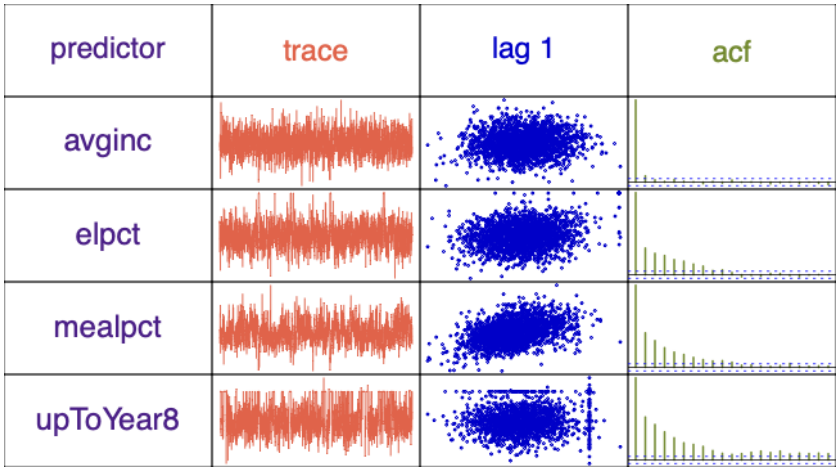

Figure 6: *The graphic produced by the command* checkChains(fit5) *for the* gamselBayes() *fit object* fit5. *The Figure 2 caption provides full details on* checkChain() *graphics.*

The longer chains are visualised using:

```
> checkChains(fit5)
```

and the result is shown in Figure 6. The coefficient summary for this longer chain fit is obtained using:

```
> summary(fit5)
```

```
            posterior mean   95% credible  interval
avginc             0.71893         0.52842  0.912240
elpct             -0.13870        -0.21967 -0.031572
mealpct           -0.35160        -0.43830 -0.251910
upToYear8         -3.02580        -6.55260  0.000000
```

The coefficient summary shows that the variables *percentage of students qualifying for reduced-price lunches*, *percentage of English learners* and *school going up to Year 8* have statistically significant negative effect on mean mathematics score. The *average income of the district* has a significant positive effect.

3.2 Analysis with Some Logarithmically Transformed Predictors

Three of the continuous predictors used in the previous analysis are quite skewed. It may be worthwhile to see what happens if skewness-reducing logarithmic transformations are used. The following code achieves this:

```
> Caschool$log.enrltot <- log(Caschool$enrltot)
> Caschool$log.avginc <- log(Caschool$avginc)
> Caschool$log.elpct <- log(Caschool$elpct + 1)
> Xgeneral <- Caschool[,c("log.enrltot","calwpct","mealpct","compstu",
+                         "expnstu","str","log.elpct","log.avginc")]
> fit6 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral)

> effectTypes(fit6)

   ---------------------------------------------
   Predictors selected as having linear effects:
   ---------------------------------------------
```

```
calwpct log.elpct mealpct upToYear8

-------------------------------------------------
Predictor selected as having a non-linear effect:
-------------------------------------------------

log.avginc
```

The main change is the presence of a non-linear effect for the logarithm of the district average income. This effect can be viewed using:

```
> plot(fit6,xlab = "log(average income of school district)")
```

and is shown in Figure 7. The linear effects summary is:

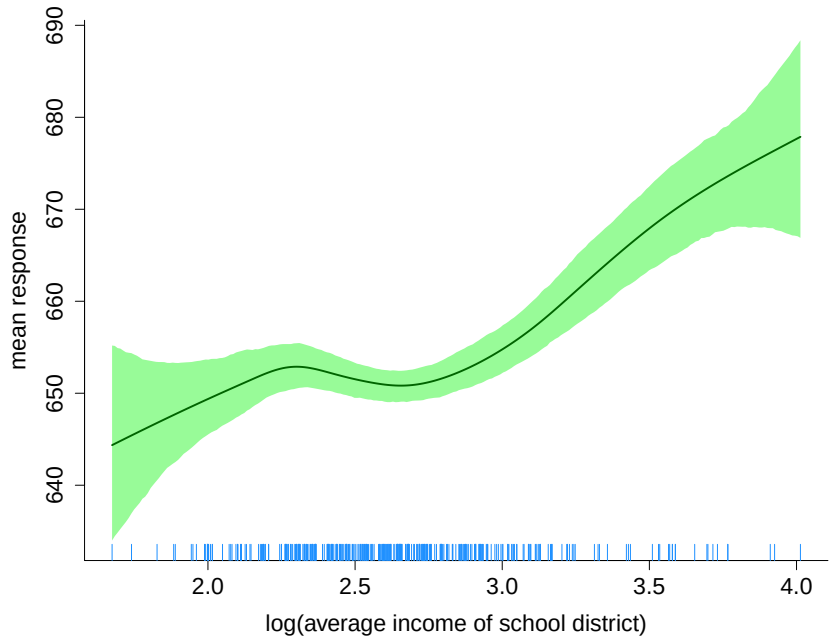

Figure 7: *The plots produced from the command* plot(fit6,xlab = "log(average income of school district)") *for the first* gamselBayes() *fit, stored as* fit6. *The shaded regions correspond to pointwise 95% credible intervals. The rugs at the base of each plot show values of each predictor.*

```
> summary(fit6)

           posterior mean   95% credible interval
calwpct         -0.087453       -0.25997  0.00000
log.elpct       -1.848300       -2.92950 -0.71768
mealpct         -0.388840       -0.48439 -0.29326
upToYear8       -4.584100       -7.58210  0.00000
```

The gamselBayes chain check for fit6 involves:

```
> checkChains(fit6)
```

and leads to the chain visual summary shown in Figure 8.

This final gamselBayes() fit to the California schools data has the following aspects:

- 5 of the 9 candidate predictors are selected,

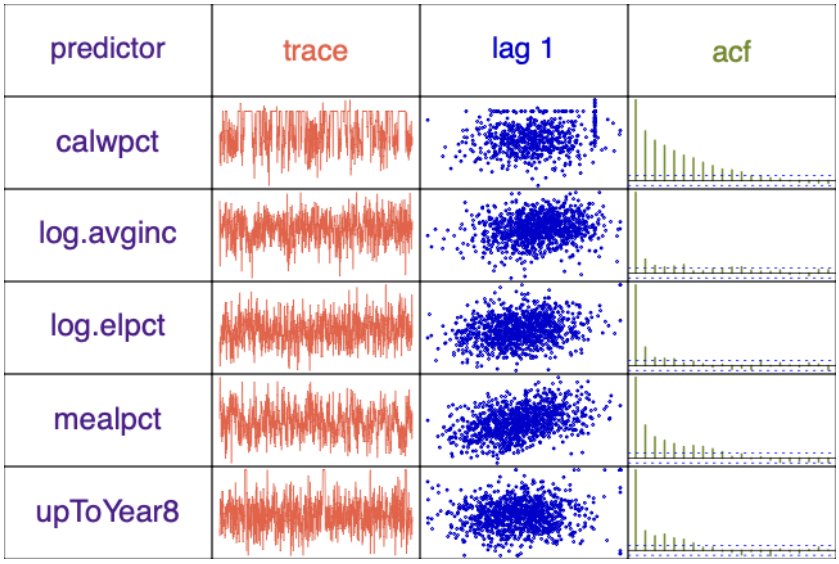

Figure 8: *The graphic produced by the command* checkChains(fit6) *for the* gamselBayes() *fit object* fit6. *The Figure 2 caption provides full details on* checkChain() *graphics.*

- 4 of them, calwpct, log(elpct + 1) , mealpct and upToYear8 have linear effects on the mean response,
- 1 of them, log(avginc) impacts the response in a non-linear fashion, with two ramps and a plateau, as shown in Figure 7.

4 Analysis of Boston Mortgage Applications Data

The data frame BostonMortgages within the HRW package (Harezlak, Ruppert & Wand, 2021) contains data on 13 variables for 2,380 mortgage applications in Boston, U.S.A., during the years 1998–1999. The data are loaded using:

```
> library(HRW) ; data(BostonMortgages)
```

Descriptions of the variables are provided by:

```
> help(BostonMortgages)
```

A key variable is BostonMortgages[,"deny"], and codes whether or not the mortgage application was denied. A numerical version of this variable, for which 1 codes the application being denied and 0 codes the application being approved, is used as a response variable in this illustration. The code for setting up the response data vector is:

```
> y <- as.numeric(BostonMortgages[,"deny"] == "yes")
```

Next we set up the data frame of vectors which enter the model linearly. The first phase involves conversion of five factor variables with yes/no coding to binary forms:

```
> Xlinear <- BostonMortgages[,c("pbcr","dmi","self","single","black")]
> for (j in 1:ncol(Xlinear))
+    Xlinear[,j] <- as.numeric(Xlinear[,j] == "yes")
```

The indicator of whether or not the mortgage application pertains to a condominium is added on using:

```
> Xlinear$condominium <- BostonMortgages[,"condominium"]
```

The BostonMortgages data frame includes a 6-level ordinal variable that represents credit risk. The following code creates and adds to Xlinear five indicator variables for each of the credit scores equalling 1, 2, 3, 4 and 5:

```
> Xlinear$CCSeq1 <- as.numeric(BostonMortgages$ccs==1)
> Xlinear$CCSeq2 <- as.numeric(BostonMortgages$ccs==2)
> Xlinear$CCSeq3 <- as.numeric(BostonMortgages$ccs==3)
> Xlinear$CCSeq4 <- as.numeric(BostonMortgages$ccs==4)
> Xlinear$CCSeq5 <- as.numeric(BostonMortgages$ccs==5)
```

The next set of commands are similar to the credit score processing, but are for a 4-level ordinal mortgage credit score variable:

```
> Xlinear$MCSeq1 <- as.numeric(BostonMortgages$mcs==1)
> Xlinear$MCSeq2 <- as.numeric(BostonMortgages$mcs==2)
> Xlinear$MCSeq3 <- as.numeric(BostonMortgages$mcs==3)
```

The variable BostonMortgages[,"uria"] is quantitative and represents the unemployment rate in the applicant's industry. However, it only has 10 unique values and is not conducive to spline-based estimation of non-linear effects. Therefore, it is included in the linear effects-only data frame using:

```
> Xlinear$uria <- BostonMortgages[,"uria"]
```

The three remaining predictors are quantitive ratio variables with hundreds of unique values. These are definitely worth considering for possible non-linear effects and make up the Xgeneral data frame:

```
> Xgeneral <- BostonMortgages[,c("dir","hir","lvr")]
```

In the call to gamselBayes() it is important to specify family = "binomial" due to the binary nature of y for this example:

```
> fit7 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral,
+                     family = "binomial")
```

The last command leads to the fit object being stored in fit7. Then:

```
> effectTypes(fit7)
```

leads to

```
-------------------------------------------
Predictors selected as having linear effects:
-------------------------------------------

CCSeq1 CCSeq2 black dmi pbcr self single

-------------------------------------------------
Predictors selected as having non-linear effects:
-------------------------------------------------

dir lvr
```

This output shows that

- 7 of the candidate predictors are selected as having linear effects,

- 2 of the candidate predictors are selected as having non-linear effects.

To assess the effects of the variables chosen as having linear effects, we issue:

```
> summary(fit7)
```

which leads to the following inferential summary of the coefficients:

```
        posterior mean   95% credible interval
CCSeq1       -0.69062      -0.898040 -0.45129
CCSeq2       -0.32384      -0.586870  0.00000
black         0.34609       0.084246  0.54041
dmi           2.76200       2.142600  3.51720
pbcr          0.73498       0.492610  0.98477
self          0.17026       0.000000  0.43627
single        0.13679       0.000000  0.34169
```

It shows, for example, that being self-employed, single or African-American each have positive impacts on the probability of mortgage denial.

The two predictors having non-linear effects in fit7 are *ratio of the debt payments to the total income* and *ratio of the loan size to the assessed value of the property*. Figure 9 shows these effects.

```
> plot(fit7,xlim = rbind(c(0,1),c(0,1)),ylim = rbind(c(0,0.8),c(0,0.8)),
+      xlab = c("debt payment to income ratio",
+               "loan size to property value ratio"))
```

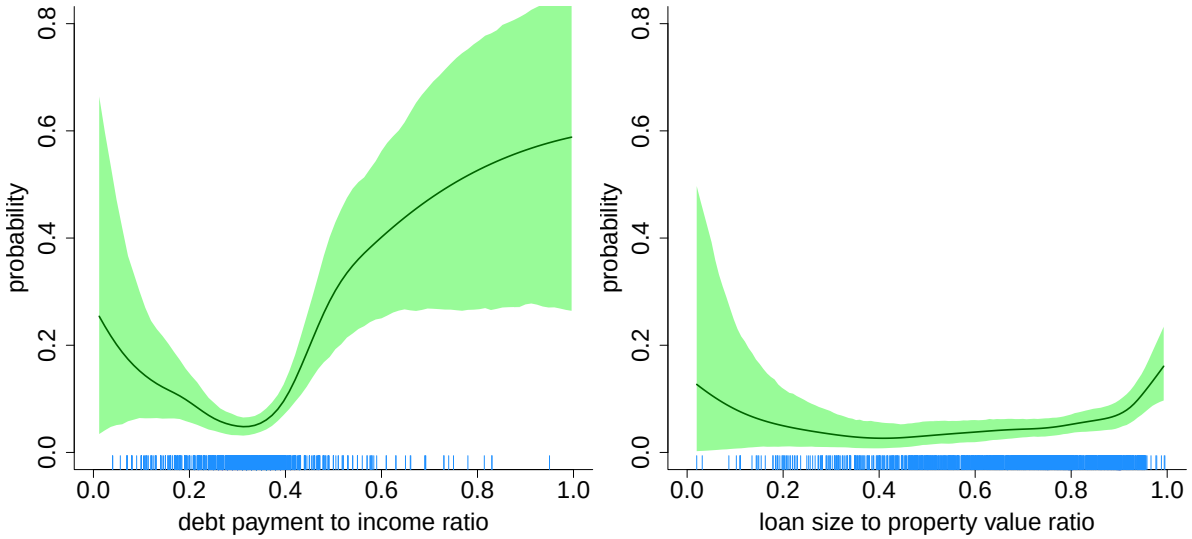

Figure 9: *The plots produced from the command* plot(fit7,xlim = rbind(c(0,1),c(0,1)),ylim = rbind(c(0,0.8),c(0,0.8)), xlab = c("debt payment to income ratio","loan size to property value ratio")) *for the first* gamselBayes() *fit, stored as* fit7. *The curves are vertically centred estimates of the effects of ratio of the debt payments to the total income (left panel) and ratio of the loan size to the assessed value of the property (right panel) on the probability of mortgage application denial. The shaded regions correspond to pointwise 95% credible intervals. The rugs at the base of each plot show values of each predictor.*

Lastly, we carry out a check of the chains on which the fitting and inference of fit7 is based. Since there are 9 selected predictors in fit7, the checkChains() graphics are divided into two graphics with a maximum of 6 predictors per graphic.

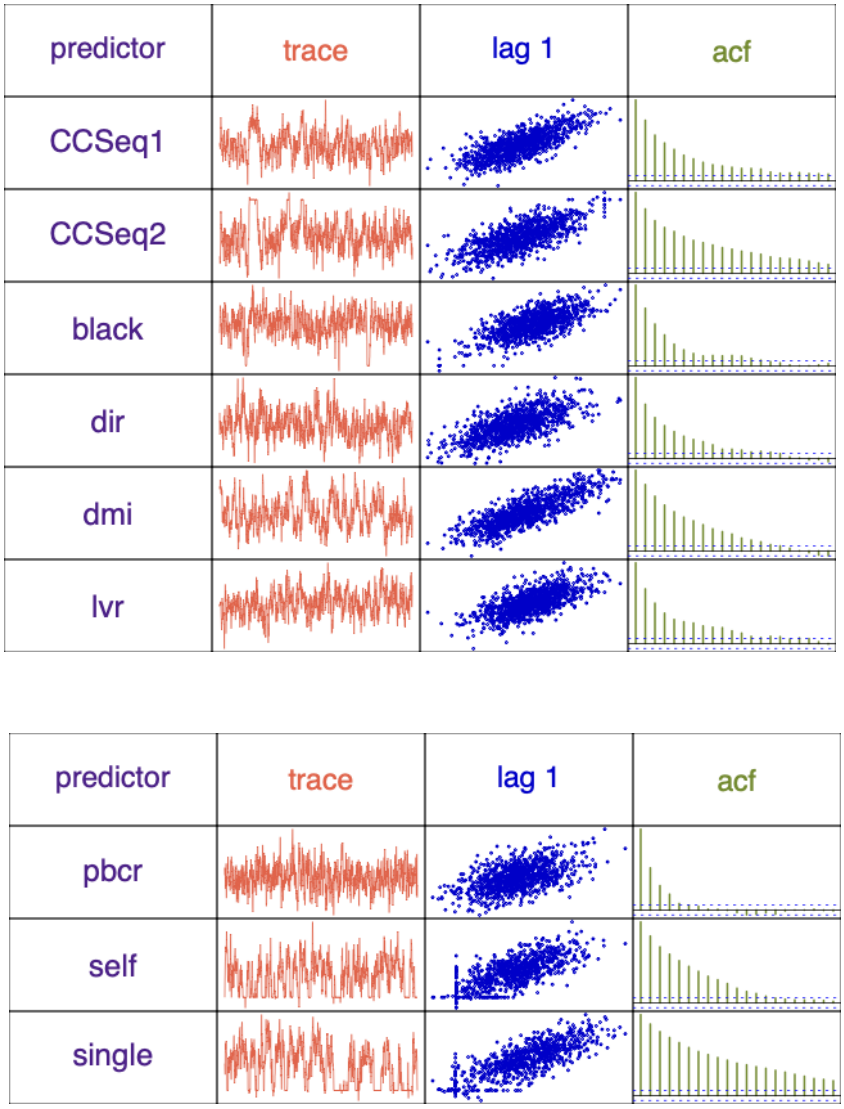

Figure 10: *The graphic produced by the command* checkChains(fit7) *for the* gamselBayes() *fit object* fit7. *The Figure 2 caption provides full details on* checkChain() *graphics.*

```
> checkChains(fit7)
```

The result is shown in Figure 10. All chains are seen to be reasonably well-behaved and the approximate Bayesian inference is sound.

5 Analysis of Data on House Sales in Sydney, Australia

This example depends upon the R package HRW, which accompanies the Harezlak, Ruppert & Wand (2018) book. After making sure that HRW is installed, load the data using:

```
> library(HRW) ; data(SydneyRealEstate)
```

The command:

```
> help(SydneyRealEstate)
```

leads to a detailed description of the SydneyRealEstate data frame. It has 39 variables corresponding to 37,676 house sales in Sydney, Australia, during the year 2001.

The following code determines the indices of important parts of the SydneyRealEstate data frame:

```
> indResponse <- 1 ; indsContinEarly <- c(2:4,7)
> indSaleDate <- 5 ; indSaleQtr <- 6
> indPostCode <- 8 ; indCrimeRate <- 10
> indAirNoise <- 24 ; indsContLater <- c(11:23,25:26,33:39)
```

Next, we obtain indicator variables for the sale quarter variable and presence of aircraft noise:

```
> saleQtrEq2 <- as.numeric(SydneyRealEstate[,indSaleQtr] == 2)
> saleQtrEq3 <- as.numeric(SydneyRealEstate[,indSaleQtr] == 3)
> saleQtrEq4 <- as.numeric(SydneyRealEstate[,indSaleQtr] == 4)
> aircraftNoise <- as.numeric(SydneyRealEstate[,indAirNoise] > 0)
```

Most of the other potential predictor variables are well-behaved. However, running the commands:

```
> par(mfrow = c(1,1))
> hist(SydneyRealEstate[,"crimeRate"],breaks = 100,col = "plum")
```

shows that SydneyRealEstate[,"crimeRate"], which is a measure of the crime rate of each house's suburb, is highly skewed and with a small fraction of the houses having a large outlying value (the histogram is not shown here). Application of the the $\log(x+1)$ transformation makes the data more amenable to the `gamsel()` methodology. From now on we work with this transformation of SydneyRealEstate[,"crimeRate"].

Three other potential predictors, SydneyRealEstate[,c("neph","PM10","SO2")], are quantitative but have low numbers of unique values. To avoid problems with spline fitting, these predictors are considered as having only a linear effect, and not permitted to have possible non-linear effects.

We are now ready to set up the the response vector and predictor data frames:

```
> y <- SydneyRealEstate$logSalePrice
> Xlinear <- data.frame(saleQtrEq2,saleQtrEq3,saleQtrEq4,aircraftNoise,
+                       SydneyRealEstate[,c("NO","NO2","ozone","neph",
+                                           "PM10","SO2")])
> Xgeneral <- data.frame(SydneyRealEstate[,indsContinEarly],
+                        log(SydneyRealEstate[,indCrimeRate] + 1),
+                        SydneyRealEstate[,indsContLater])
> names(Xgeneral)[5] <- "log(crimeRate + 1)"
> print(dim(Xlinear))

[1] 37676    10

> print(dim(Xgeneral))

[1] 37676    27
```

Given the high sample size of $37,676$ and the fact that the numbers of potential predictors in Xlinear and Xgeneral are, respectively, 10 and 27 we use the faster mean field variational Bayes approach.

```
> fit8 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral,
+                     method = "MFVB")
```

The effect types of the selected predictors are tabulated using:

```
> effectTypes(fit8)
```

are as follows:

```
-----------------------------------------------
Predictors selected as having linear effects:
-----------------------------------------------

NO PM10 SO2 distToHighway distToMedical
distToTunnel longitude neph ozone saleQtrEq2
saleQtrEq3 saleQtrEq4

-------------------------------------------------
Predictors selected as having non-linear effects:
-------------------------------------------------

distToAmbulance distToBusStop distToFactory
distToHospital distToMainRoad distToPark
distToSchool distToSealedRoad distToUnsealedRoad
foreignerRatio income infRate latitude
log(crimeRate + 1) lotSize
```

For the 12 selected linear effects the command:

```
> summary(fit8)
```

leads to

```
                posterior mean   95% credible   interval
NO                  0.0121340     0.01075700  0.0135130
PM10                0.0138840     0.01109700  0.0166670
SO2                -1.5262000    -1.71840000 -1.3287000
distToHighway       0.0017178     0.00057697  0.0028439
distToMedical      -0.0036716    -0.00629090 -0.0010810
distToTunnel        0.0089135     0.00775600  0.0100650
longitude           2.5793000     2.49920000  2.6576000
neph               -0.4298800    -0.51935000 -0.3396100
ozone               0.3396200     0.31465000  0.3658200
saleQtrEq2          0.0190840     0.00893360  0.0291840
saleQtrEq3          0.0641250     0.05470700  0.0733930
saleQtrEq4          0.1311400     0.12168000  0.1402800
```

This output shows, for example, that houses sold later in the calendar year, such as the 4th quarter period of October–December, tend to have higher sales prices.

The commands:

```
> xlabDetailed <- c("lot size","degrees latitude","inflation rate",
+                   "log(crime rate +1)","average income of suburb",
+                   "distance to bus stop","distance to park",
+                   "distance to main road","distance to sealed road",
+                   "distance to unsealed road","foreigner ratio in suburb",
+                   "distance to ambulance","distance to factories",
+                   "distance to hospital","distance to school")
> plot(fit8,xlab = xlabDetailed,rugSampSize = 1000)
```

lead to the plot of the estimated non-linear effects, shown in Figure 11. In all, there are 15 estimated non-linear effects shown in Figure 11. The first panel shows that the effect of lot size is mainly monotonically increasing, but with some ramps and a plateau.

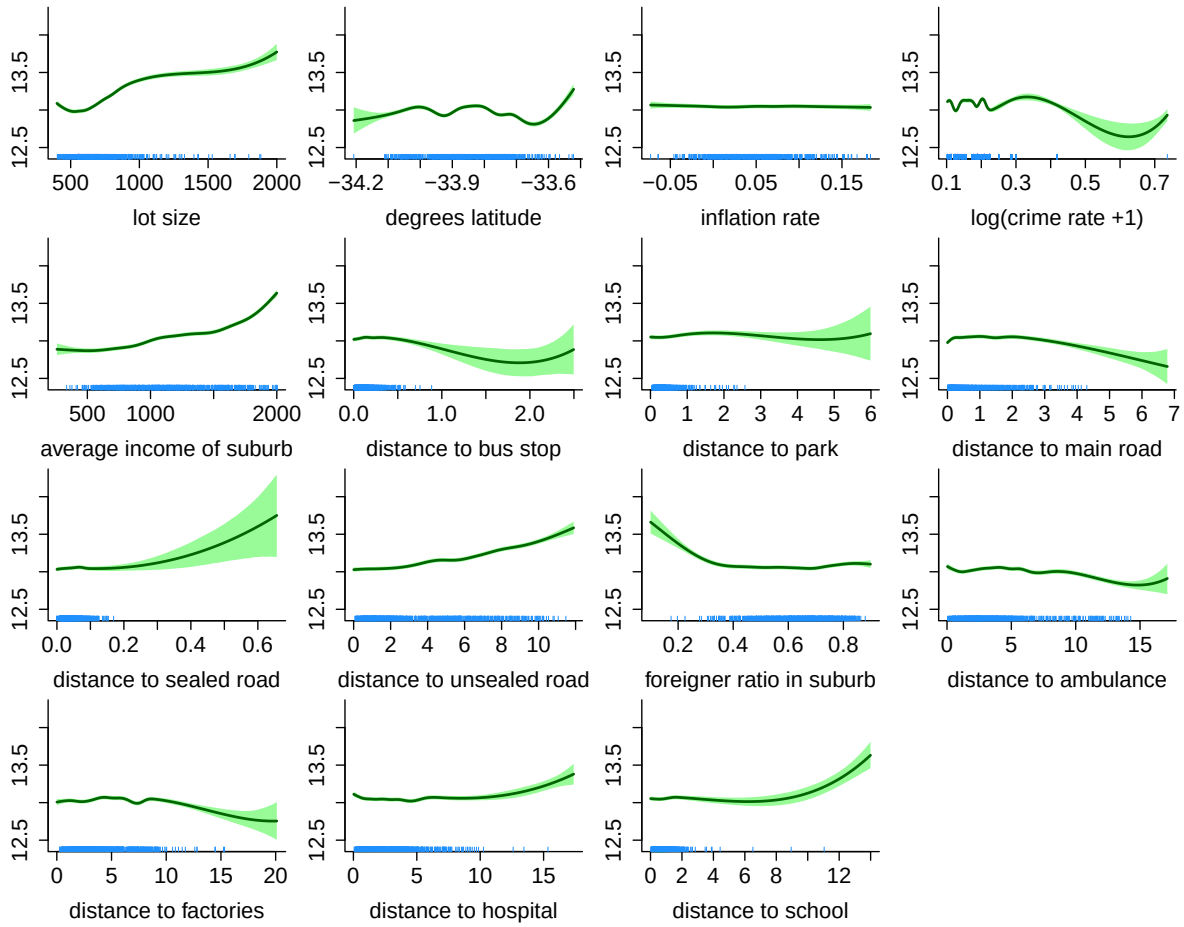

Figure 11: *The graphic produced by the command* plot(fit8,xlab = xlabDetailed) *for the* gamselBayes() *fit object* fit8. *The curves are vertically centred estimates of the effects of each predictor selected as having a non-linear effect. The shaded regions correspond to pointwise 95% credible intervals. The rugs at the base of each plot show values of each predictor. Due to the very large sample size, random subsets of size* 1,000 *are used in the rugs.*

6 Analysis of Car Auction Data

This example also depends upon the R package HRW. Assuming that HRW is installed, load the data via:

```
> library(HRW) ; data(carAuction)
```

Then issue:

```
> help(carAuction)
```

to obtain a detailed description of the carAuction data frame. This data frame has 51 variables corresponding to 72,983 cars purchased at auctions by automobile dealerships in the United States of America and originates from the competition titled "Don't Get Kicked!" that ran on the kaggle platform (www.kaggle.com) during 2011-2012.

In this illustration we hold back data on 2,983 of the cars for testing. The remaining 70,000 will be used as a training set for fitting a probit additive model via gamselBayes():

```
> namesNonPred <- c("RefId","IsBadBuy")
> namesGeneral <- c("odomRead","price","costAtPurch","warrantyCost")
> namesLinear <- setdiff(names(carAuction),union(namesNonPred,namesGeneral))
```

```
> nTotal <- nrow(carAuction)
> nTest <- 2983 ; nTrain <- nTotal - nTest
> testInds <- sample(1:nTotal,nTest,replace = FALSE)
> trainInds <- setdiff(1:nTotal,testInds)
```

Now set up the response data vector and predictor data frames:

```
> yTrain <- carAuction$IsBadBuy[trainInds]
> XlinearTrain <- carAuction[trainInds,namesLinear]
> XgeneralTrain <- carAuction[trainInds,namesGeneral]
```

The number of candidate predictors in XlinearTrain is 45, whilst the number of candidate predictors in XgeneralTrain is 4. In the call to gamselBayes() it is important to specify family = "binomial" due to yTrain being binary:

```
> fit9 <- gamselBayes(y = yTrain,Xlinear = XlinearTrain,
+                     Xgeneral = XgeneralTrain,family = "binomial")
```

The last command leads to the fit object being stored in fit9. Then:

```
> effectTypes(fit9)
```

leads to the following estimated effects types:

```
   ---------------------------------------------
   Predictors selected as having linear effects:
   ---------------------------------------------

   AmericanMade ageAtSale aucEqAdesa
   aucEqManheim colourEqRed makeEqChevrolet
   makeEqChrysler odomRead purchIn2010
   purchInFlorida purchInNorthCarolina
   purchInTexas sizeEqMedium sizeEqSUV
   transEqManual trimEqBas wheelEqAlloy
   wheelEqCovers

   -------------------------------------------------
   Predictor selected as having a non-linear effect:
   -------------------------------------------------

   costAtPurch
```

We see from this output that, out of the 49 candidate parameters, 18 are selected as having a linear effect and 1 is selected as having a non-linear effect.

A Bayesian inferential summary of the linear effect coefficients is achieved using:

```
> summary(fit9)
```

and leads to

```
                     posterior mean   95% credible    interval
   AmericanMade          -4.8501e-02    -1.2929e-01  0.0000e+00
   ageAtSale              9.5052e-02     8.3936e-02  1.0454e-01
   aucEqAdesa             5.1162e-02     0.0000e+00  9.8698e-02
   aucEqManheim           5.2225e-02     0.0000e+00  9.3681e-02
   colourEqRed            2.8276e-02     0.0000e+00  9.1637e-02
   makeEqChevrolet       -1.1959e-01    -1.7515e-01 -6.0241e-02
```

```
makeEqChrysler           5.7094e-02     0.0000e+00  1.2664e-01
odomRead                 3.4378e-06     2.4554e-06  4.4517e-06
purchIn2010              1.0056e-01     7.1407e-02  1.2898e-01
purchInFlorida          -1.2609e-01    -1.7060e-01 -8.3081e-02
purchInNorthCarolina    -1.1550e-01    -1.6167e-01 -6.7014e-02
purchInTexas             8.4710e-02     4.6216e-02  1.1783e-01
sizeEqMedium            -5.7743e-02    -9.5144e-02  0.0000e+00
sizeEqSUV                1.7890e-01     1.3341e-01  2.2379e-01
transEqManual           -1.5971e-01    -2.3401e-01 -8.9339e-02
trimEqBas                6.0669e-02     0.0000e+00  1.0840e-01
wheelEqAlloy            -1.5245e+00    -1.5703e+00 -1.4796e+00
wheelEqCovers           -1.5935e+00    -1.6354e+00 -1.5480e+00
```

Some of the effects match intuition, such as older cars having a higher probability of being a bad buy at auction. Less intuitive is the fact that the same is true for cars that were purchased in Texas.

The command:

```
> plot(fit9,xlim = rbind(c(0,20000)),xlab = "cost at purchase (U.S. dollars)",
+      rugSampSize = 1000)
```

leads to the figure shown in Figure 12, which shows the effect of

the acquisition cost paid for the car at the time of purchase in U.S. dollars

which has an interesting U-shaped effect.

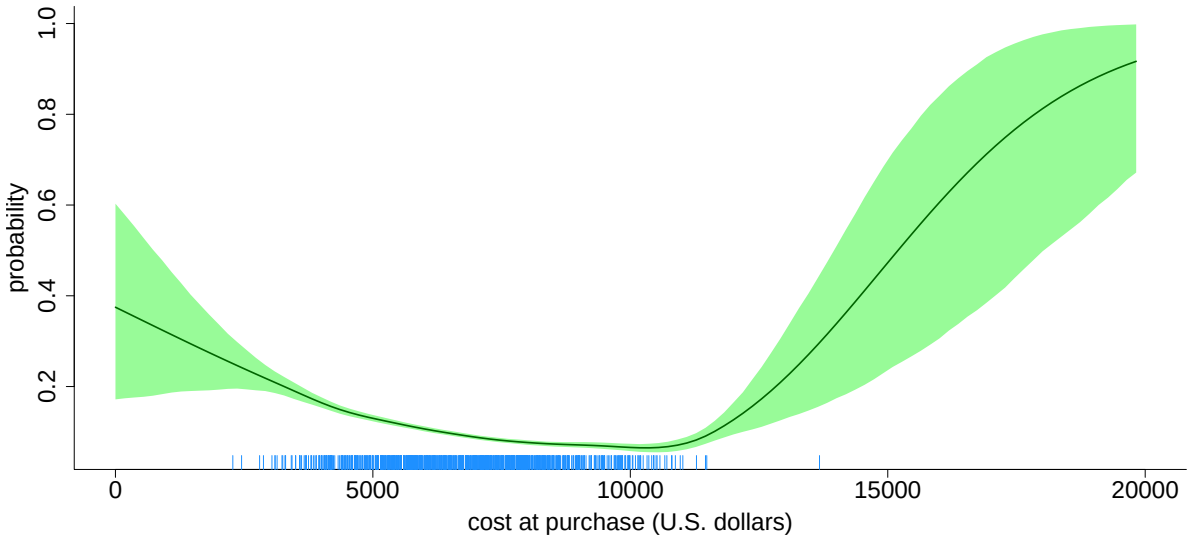

Figure 12: *The plots produced from the command* plot(fit9,xlim = rbind(c(0,20000)),xlab ="cost at purchase (U.S. dollars)") *for the* gamselBayes() *fit, stored as* fit9. *The shaded region correspond to pointwise 95% credible intervals. The rug at the base of the plot shows values of the predictor. Due to the very large sample size, a random subset of size* 1,000 *are used in the rug.*

Next we assess how well cars in the test data set are classified according to the fit in fit9, starting with setting up test versions of the y, Xlinear and Xgeneral inputs:

```
> yTest <- carAuction$IsBadBuy[testInds]
> XlinearTest <-  carAuction[testInds,namesLinear]
> XgeneralTest <- carAuction[testInds,namesGeneral]
```

The vector of predictions on the probit (Standard Normal quantile function) scale is obtained using:

```
> probVecTest <- predict(fit9,newdata = list(XlinearTest,XgeneralTest))
```

The following code then obtains and prints the confusion matrix:

```
> yTestHat <- as.numeric(probVecTest>0.5)
> confusionMatrix <- table(yTestHat,yTest)
> dimnames(confusionMatrix)[[1]] <- c("classified good buy",
+                                     "classified bad buy")
> dimnames(confusionMatrix)[[2]] <- c("actually good buy",
+                                     "actually bad buy")
> cat("The confusion matrix is:\n")

The confusion matrix is:

> print(confusionMatrix)

                    yTest
yTestHat             actually good buy actually bad buy
  classified good buy              2570              284
  classified bad buy                 57               72
```

Lastly, we obtain an estimate of the misclassification rate via:

```
> estMisClassRate <- 100*sum(yTestHat != yTest)/nTest
> cat("The estimated misclassification rate is ",
+     signif(estMisClassRate,4),"%.\n",sep = "")

The estimated misclassification rate is 11.43%.
```

The classifier based on fit9 classifies about 88.57% of cars correctly in terms of their bad buy versus good buy status.

7 Fitting and Plotting Options

Whenever `gamselBayes()` is applied to a particular data set there are numerous options concerning, for example, the size of the spline basis and hyperparameter choices. In this section we illustrate the tweaking of such values. Similar comments apply to plotting a `gamselBayes()` object.

Throughout this section, we work with the California schools data set that was described and analysed in Section 3. The following commands from that section load the required packages and set up the input data objects:

```
> library(gamselBayes) ; library(Ecdat) ; data(Caschool)
> y <- Caschool$mathscr
> upToYear8 <- as.numeric(Caschool[,"grspan"] == "KK-08")
> Xlinear <- data.frame(upToYear8)
> Caschool$log.enrltot <- log(Caschool$enrltot)
> Caschool$log.avginc <-  log(Caschool$avginc)
> Caschool$log.elpct <-   log(Caschool$elpct + 1)
> Xgeneral <- Caschool[,c("log.enrltot","calwpct","mealpct","compstu",
+                         "expnstu","str","log.elpct","log.avginc")]
```

7.1 Sparsity Imposition Options

Let β be a generic regression coefficient attached to one of the linear predictors. In models used by `gamselBayes()`, $\beta = \gamma_\beta \widetilde{\beta}$ where γ_β is binary and $\widetilde{\beta}$ is continuous. Therefore

$$P(\beta = 0|\boldsymbol{y}) = P(\gamma_\beta = 0|\boldsymbol{y}) = 1 - E(\gamma_\beta|\boldsymbol{y}),$$

and the posterior distribution of γ_β can be used to decide between hypotheses $H_0 : \beta = 0$ and $H_1 : \beta \neq 0$. A natural rule is to accept H_0 if and only if $E(\gamma_\beta|\boldsymbol{y}) \leq \frac{1}{2}$. However, with parsimony in mind, `gamselBayes()` also supports less stringent rules. Rather than thresholding $E(\gamma_\beta \boldsymbol{y})$ at $\frac{1}{2}$, consider a family of rules indexed by a threshold parameter $\tau \in (0, 1)$. After fixing τ, our strategy for deciding between an effect being zero or linear is

$$\text{the effect is zero if } E(\gamma_\beta|\boldsymbol{y}) \leq 1 - \tau, \text{ otherwise the effect is linear.}$$

Analogous rules are used for deciding between an effect being linear or non-linear.

Lower values of τ lead to sparser fits, so in the `gamselBayes()` function we use the identifier lowerMakesSparser for the τ parameter. The default value of lowerMakesSparser is 0.5 when method="MCMC" and 0.1 when method="MFVB". The following command stipulates that lowerMakesSparser be set to the lower value of 0.3:

```
> fit10 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral,
+                      lowerMakesSparser = 0.3)
```

For this choice of thresholding, the command:

```
> effectTypes(fit10)
```

shows that the estimated effect types become:

```
   ---------------------------------------------
   Predictors selected as having linear effects:
   ---------------------------------------------

   log.elpct mealpct upToYear8

   -------------------------------------------------
   Predictor selected as having a non-linear effect:
   -------------------------------------------------

   log.avginc
```

which is sparser than the default fit. The command:

```
> summary(fit10)
```

then leads to the following inferential summary for the linear coefficients:

```
           posterior mean   95% credible interval
log.elpct       -1.80050       -3.02940  0.00000
mealpct         -0.39045       -0.49982 -0.28695
upToYear8       -4.36920       -7.42400  0.00000
```

The function `gamselBayesUpdate()` allows one to change the value of lowerMakesSparser without having to re-do the fitting phase. If, for example, a change of from τ from 0.3 to 0.1 is of interest then the command:

```
> fit11 <- gamselBayesUpdate(fit10,lowerMakesSparser = 0.1)
```

quickly provides the new fit object with the newer thresholding.

7.2 Control Options

The control argument of the gamselBayes() function can be used to specify various aspects of the Bayesian model and the strategy for achieving approximate inference. Suppose, for example, that the following departures from the default settings are desired:

- the number of interior knots in the spline basis functions is 15,
- the prior Normal distribution standard deviation of the linear coefficients parameters is $\sigma_\beta = 150$,
- the prior Normal distribution standard deviation of the spline basis coefficients parameters is $\sigma_u = 2,000$,
- the number of warm-up Markov chain Monte Carlo samples is $3,000$,
- the number of kept Markov chain Monte Carlo samples is $10,000$,
- the thinning factor of the kept Markov chain Monte Carlo samples is 2.

Then the following call to gamselBayes() achieves this:

```
> fit12 <- gamselBayes(y = y,Xlinear = Xlinear,Xgeneral = Xgeneral,
+                      control = gamselBayes.control(numIntKnots = 15,
+                                sbeta = 150,su = 2000,nWarm = 3000,
+                                nKept = 10000,nThin = 2))
```

For fit12, the estimated effect types are found by issuing

```
> effectTypes(fit12)
```

are:

```
   ---------------------------------------------
   Predictors selected as having linear effects:
   ---------------------------------------------

   calwpct log.elpct mealpct upToYear8

   -------------------------------------------------
   Predictor selected as having a non-linear effect:
   -------------------------------------------------

   log.avginc
```

and the linear coefficients summary from

```
> summary(fit12)
```

is

```
             posterior mean   95% credible interval
   calwpct        -0.085515       -0.26558  0.00000
   log.elpct      -1.819100       -2.95960 -0.51544
   mealpct        -0.389900       -0.48417 -0.28941
   upToYear8      -4.412100       -7.48330  0.00000
```

Access to the full set of control options is provided by:

```
> help(gamselBayes.control)
```

7.3 Plotting Options

Lastly, we demonstrate some of the options available for display of non-linear effects via the `plot()` function for `gamselBayes()` fit objects. These illustrations are for the fit object fit12 from the previous subsection.

The command:

```
> plot(fit12,estCol = "darkmagenta",varBandCol = "gold",rugCol = "seagreen",
+      xlab = "log(average income of school district)",cex.lab = 1.75)
```

modifies the colours of the function estimates, variability bands and predictor data rug-plot displays and leads to the result shown in Figure 13. Our plotting options are use of dashed

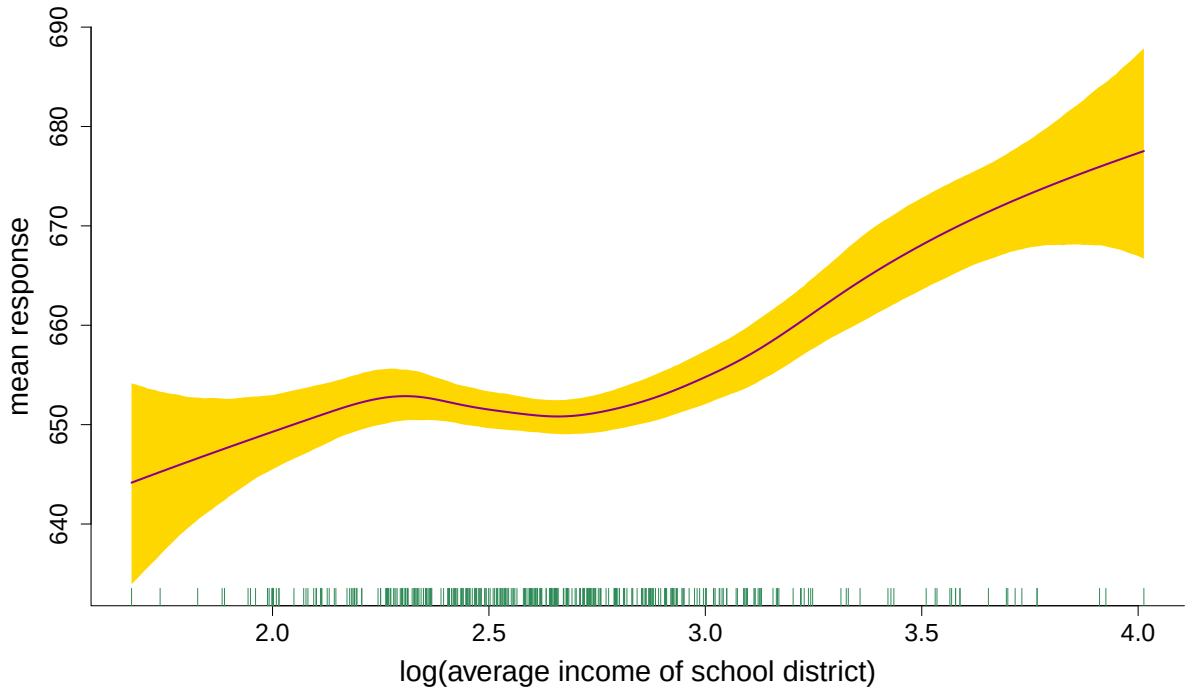

Figure 13: *The plots produced from the command* plot(fit12,estCol = "darkmagenta",varBandCol = "gold",rugCol = "seagreen", xlab = "log(average income of school district)",cex.lab = 1.75) *for the* `gamselBayes()` *fit, stored as* fit12. *The shaded regions correspond to pointwise 95% credible intervals. The rug at the base of the plot show values of each predictor.*

curves, rather than shaded polygons, to display variability bands and specification of the vertical frame limits using the ylim argument. These are illustrated by the command:

```
> plot(fit12,shade = FALSE,estCol = "steelblue",varBandCol = "darkred",
+      rugCol = "limegreen",xlab = "log(average income of school district)",
+      cex.lab = 1.75)
```

and leads to Figure 14. Further details on plotting options can be obtained by issuing:

```
> help(plot.gamselBayes)
```

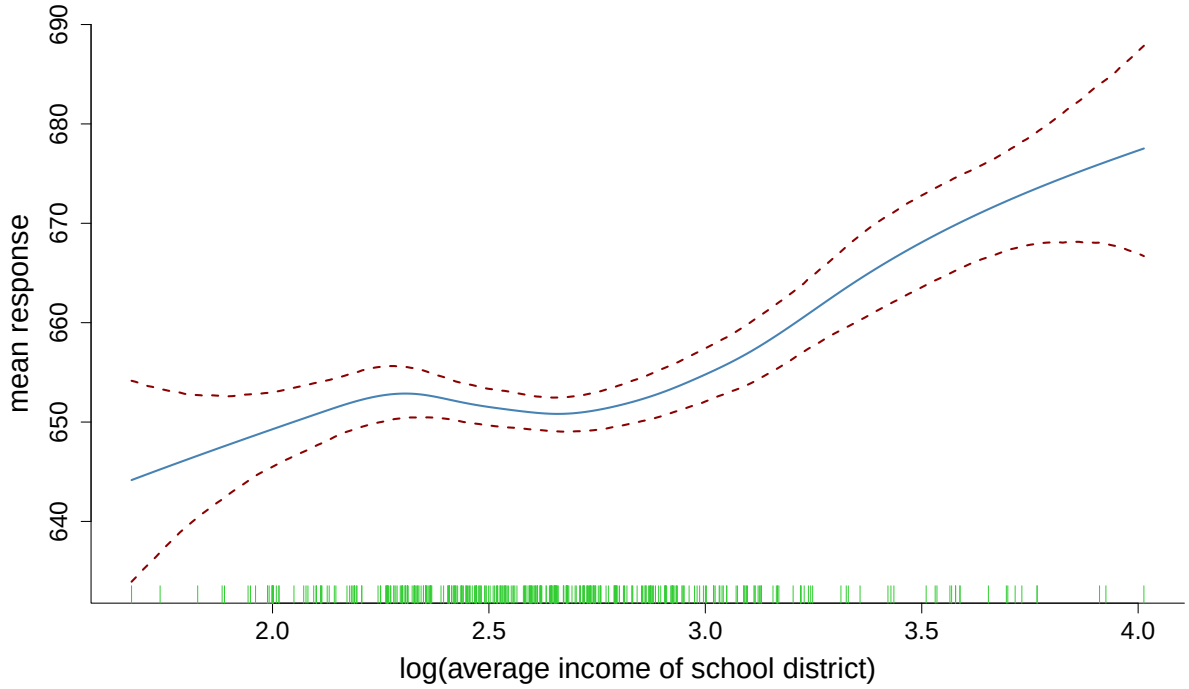

Figure 14: *The plots produced from the command* plot(fit12,shade = FALSE,estCol = "steelblue",varBandCol = "darkred", rugCol = "limegreen",xlab = "log(average income of school district)", cex.lab = 1.75) *for the* gamselBayes() *fit, stored as* fit12. *The shaded regions correspond to pointwise 95% credible intervals. The rug at the base of the plot show values of each predictor.*

8 Limitations and Trouble-Shooting

The gamselBayes package was written and is maintained, solely, by the authors of this vignette. A reasonable amount of effort has been made to safeguard against breakdown for arbitrary inputs. However, since gamselBayes is a two-person and non-industrial software package it has limitations in this regard. In this final section we list some trouble-shooting tips that may aid successful use of gamselBayes:

1. Ensure that the input data are free of missing values and non-numerical objects. In other words, the input data in y, Xlinear and Xgeneral should contain only numbers.

2. Predictors containing gross outliers can lead to problems with penalized spline fitting, so checks and pre-processing to avoid such an issue may be worthwhile.

3. If a predictor is strongly skewed then problems with penalized spline fitting are also more likely. Hence, depending on the strength of the skewness, pre-transformation of such predictors may be beneficial.

4. Each of the predictors in the Xgeneral input are treated as continuous variables. In theory, such variables have a large number of unique values. However, if its measurements have been discretized to the point that the number of unique values is smaller than around $15 - 25$ then this can lead to problems with penalized spline fitting. One simple remedy is to move the predictor to the Xlinear input and live with its effect being restricted to zero or linear, but not non-linear.

5. In a similar vein to the last point, if the sample size is smaller than around $15 - 25$ then all candidate predictors have the limitation described there and it may not be feasible to include them in the Xgeneral input.

Feedback concerning your experiences with gamselBayes may be sent to the package maintainer at the e-mail address provided by the command help(package = "gamselBayes").

References

Chouldechova, A., and Hastie, T. (2015). Generalized additive model selection. Unpublished manuscript. https://arxiv.org/pdf/1506.03850

Chouldechova, A., and Hastie, T. (2022). gamsel 1.8: Fit regularisation path for generalized additive models. R package. http://cran.r-project.org.

Croissant, Y. (2022). Ecdat 0.4: Data sets for econometrics. R package. http://cran.r-project.org.

Harezlak, J., Ruppert, D. and Wand, M.P. (2018). *Semiparametric Regression with R*. New York: Springer.

Harezlak, J., Ruppert, D. and Wand, M.P. (2021). HRW 1.0: Datasets, functions and scripts for semiparametric regression supporting Harezlak, Ruppert & Wand (2018). R package. http://cran.r-project.org.

He, V.X. and Wand, M.P. (2023). Bayesian generalized additive model selection including a fast variational option. http://arxiv.org/abs/2201.00412.