gamsel: Generalized Additive Model Selection

Trevor Hastie and Matt P. Wand

Stanford University and University of Technology Sydney

25th June, 2024

1 Introduction

The R package gamsel implements an algorithm for generalized additive model selection developed by and described in Chouldechova & Hastie (2018). The algorithm delivers a *path* of selected models that is parameterized by a positive scalar parameter, denoted by λ. Higher values of λ correspond to sparser models.

In this vignette we work through some illustrative examples involving simulated and actual data. We explain how to deal with issues arising in actual data such as candidate predictors that are categorical, heavily skewed or ostensibly are continuous but have a low number of unique observed values.

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 gamsel 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

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

with $n = 1000$ and $\sigma = 0.15$:

```
> sigmaTrue <- 0.15
> f1True <- function(v) return(0.5*(pnorm(6*v - 3) + 1))
> f2True <- function(v) return(0.5*(sin(3*pi*v) + 1))
> f3True <- function(v) return(0.5*(0.04*cosh(v^3 - 9*v^2 + 4) + 1))
> f4True <- function(v) return(0.96*v)
> set.seed(34) ; n <- 1000
> v1 <- runif(n) ; v2 <- runif(n) ; v3 <- runif(n)
> v4 <- runif(n) ; v5 <- runif(n) ; v6 <- runif(n)
> y <- rnorm(n,f1True(v1) + f2True(v2) + f3True(v3) + f4True(v4),sigmaTrue)
> X <- data.frame(v1,v2,v3,v4,v5,v6)
```

The data for all 6 candidate predictors $v_1, \ldots, v_6$ are stored in the data frame X. The response data are stored in the vector y.

These data are simulated from f_j functions such that:

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

The color-coded text here matches the color scheme used in gamsel fitted function graphics given later in this vignette. Non-linear fits are displayed in red and linear fits are colored green. The color blue is used to indicate that a predictor is not selected.

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

```
> library(gamsel) ; fit1 <- gamsel(X,y)
```

A visual summary of the fit is obtained using:

```
> par(mfrow = c(2,1)) ; summary(fit1)
```

and is shown in Figure 1. The horizontal axes correspond to the λ path which, in this case, runs from $\lambda = 23$ down to $\lambda = 0.23$. If, for a fixed λ, one draws vertical lines through the plots in Figure 1 then the selected components and the linear versus non-linear status for that λ are apparent. For example, if $\lambda = 11$ then the selected model has three linear components. If $\lambda = 2$ then the selected model has three linear and two non-linear components.

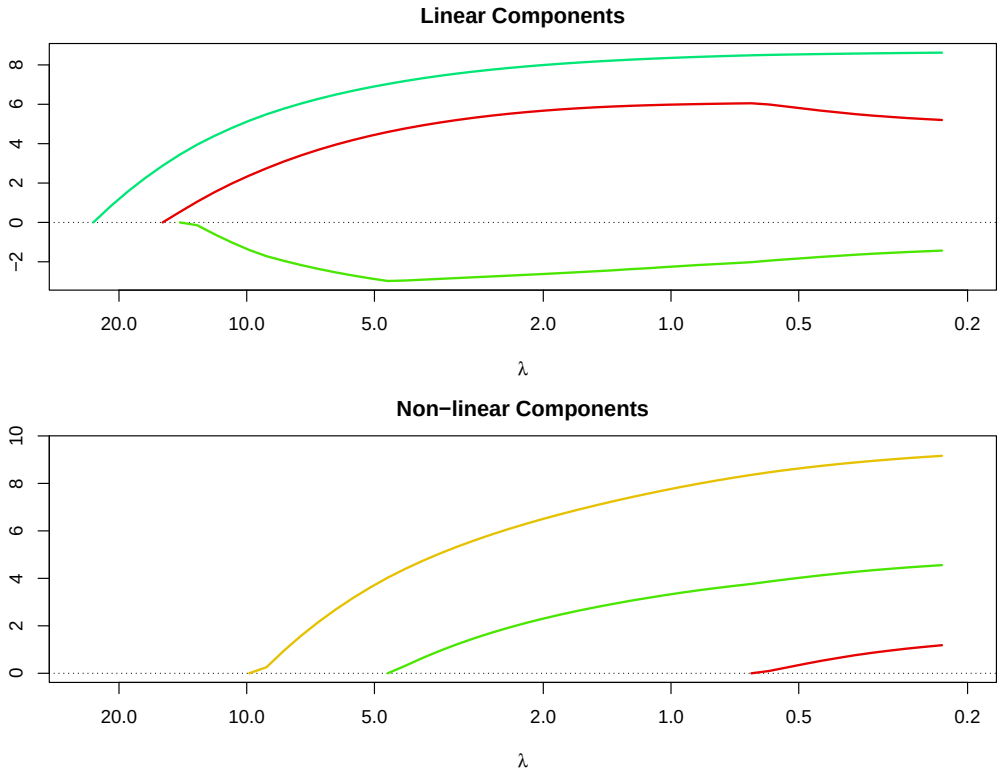

Figure 1: *The plots produced from the command* summary(fit1) *for the first* gamsel() *fit, stored as* fit1*, to the data simulated according to (1).*

2.1.1 Example Fits Along the λ Path

The fit object fit1 contains fits for a path of 50 values of λ. We now visualize three of these fits, corresponding to positions 10, 25 and 40. Note that the ordering of λ in fit1 is from high to low. The vector of the three corresponding λ values is obtained via:

```
> indexVec <-  c(10,25,40)
> lambdaVec <- signif(fit1$lambdas[indexVec],3)
```

The following code produces Figure 2, with $\lambda = 9.87$:

```
> ilambda <- 1 ; par(mfrow = c(2,3))
> for (iPred in 1:6)
+ {
+    mainString <- ""
+    if (iPred == 1) mainString <- bquote(paste(lambda," = ",.(lambdaVec[ilambda])))
```

```
+     plot(fit1,newx = X,index = indexVec[ilambda],which = iPred,
+                  bty = "l",ylim = c(-0.55,0.55),main = mainString)
+ }
```

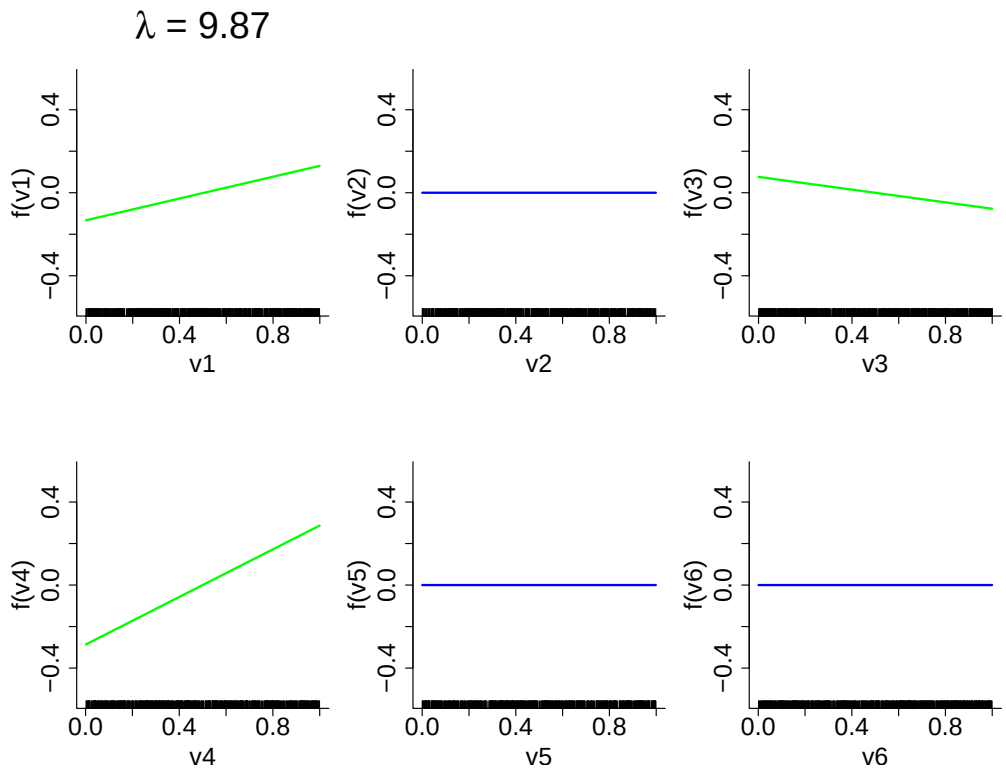

Figure 2: *The* `gamsel()` *estimates of* $f_1, \ldots, f_6$ *for the data simulated according to (1) when* $\lambda = 9.87$. *These graphics are obtained using the function* `plot()` *applied to the fit object* `fit1` *as detailed in the accompanying code. Linear fits are shown in green. The flat blue lines indicate zero fitted functions or, equivalently, non-selection of the corresponding predictors.*

For $\lambda = 9.87$ we see that the estimates of f_1, f_3 and f_4 are straight line functions and the estimates of f_2, f_5 and f_6 are zero functions. Hence, `gamsel()` selects only three predictors when $\lambda = 9.87$.

As λ decreases the selection of predictors becomes less stringent. Also, there is more freedom for non-linear effects. Setting:

```
> ilambda <- 2
```

and re-running the above plotting commands leads to Figure 3 with $\lambda = 2.41$. Notice that the estimates of f_2 and f_3 are both non-linear for this value of λ. Also, the solution is less sparse with four selected functions rather than three.

Lastly, we investigate the fit corresponding to the 40th value of (out of 50) along the λ path in `fit1` which, in this case, is $\lambda = 0.588$. Setting:

```
> ilambda <- 3
```

and then repeating the graphics code from previous two figures leads to Figure 4.

When $\lambda = 0.588$ the selected predictors are the same as those for $\lambda = 2.41$. However, f_1 is now estimated to be non-linear rather than linear. The non-linearities in f_2 and f_3 are more pronounced.

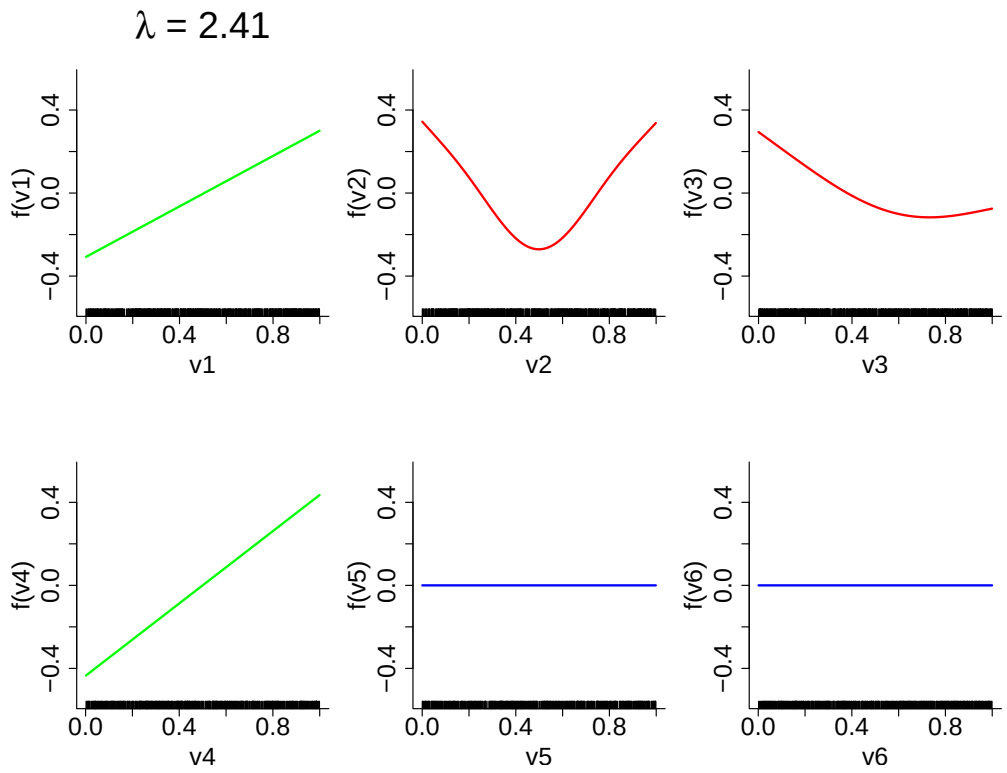

Figure 3: *The* `gamsel()` *estimates of* $f_1, \ldots, f_6$ *for the data simulated according to (1) when* $\lambda = 2.41$. *These graphics are obtained using the function* `plot()` *applied to the fit object* fit1 *as detailed in the accompanying code. Linear fits are shown in green. Non-linear fits are shown in red. The flat blue lines indicate zero fitted functions or, equivalently, non-selection of the corresponding predictors.*

2.1.2 Selection of λ Via Cross-Validation

The function `cv.gamsel()` facilitates the selection of a single value of λ using k-fold cross-validation. The underlying principle is that k-fold cross-validation estimates a measure of the quality of out-of-sample predictions of response values.

The following commands produce Figure 5, corresponding to a default call to `cv.gamsel()` for which $k = 10$. Since cross-validation involves random partitions of the data, the cross-validation curve depends on a random seed. Figure 5 is the realization that arises from set.seed(1).

```
> set.seed(1) ; fit1CVdefault <- cv.gamsel(X,y)
> par(mfrow = c(1,1)) ; plot(fit1CVdefault)
```

In Figure 5 we see a downward trend as λ becomes smaller. However, the minimum of the red dots is at the left boundary of the λ path. The default λ path is somewhat deficient and there is a case for moving the path farther to the left, corresponding to lower λ values. Making the path finer may also help identify a minimum. The next R code chunk achieves this. It also shows how to change the $k = 10$ default to a different value of k.

```
> myLambdaPath <- exp(seq(log(0.1),log(5),length = 100))
> set.seed(1)
> fit1CVmanual <- cv.gamsel(X,y,lambda = myLambdaPath,nfolds = 25)
> plot(fit1CVmanual)
```

In Figure 6 the cross-validation function values are jumpier for lower λ, but at least there is an interior local minimum. The vertical dotted line corresponding to the 89th λ path value

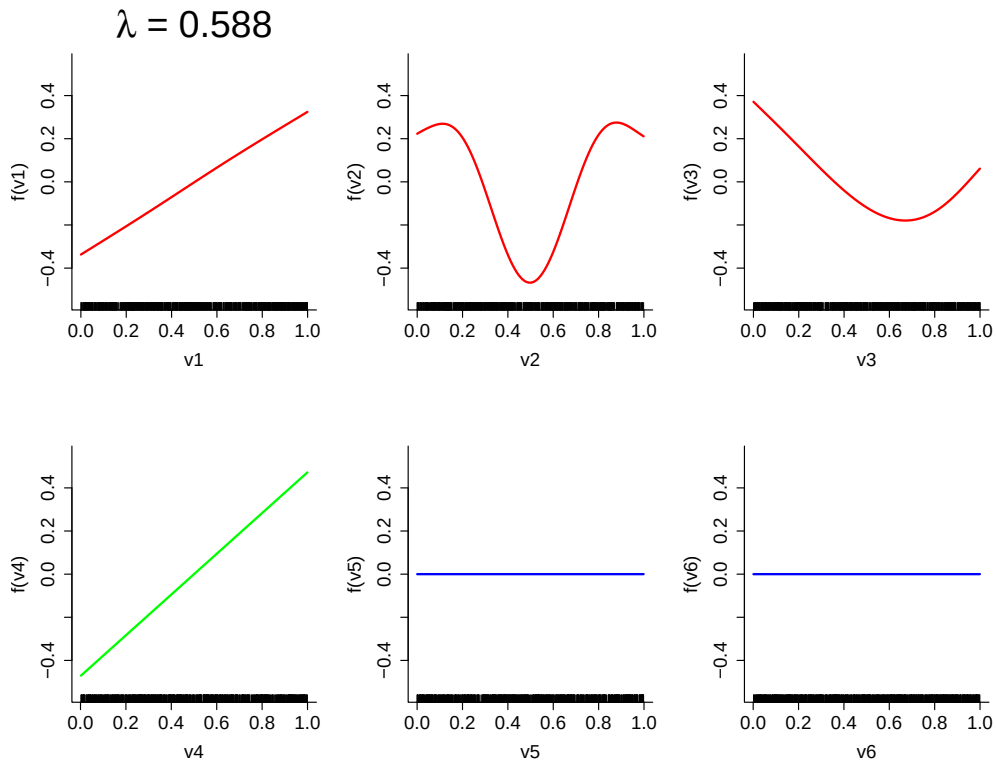

Figure 4: *The* `gamsel()` *estimates of* $f_1, \ldots, f_6$ *for the data simulated according to (1) when* $\lambda = 0.588$. *These graphics are obtained using the function* `plot()` *applied to the fit object* fit1 *as detailed in the accompanying code. Linear fits are shown in green. Non-linear fits are shown in red. The flat blue lines indicate zero fitted functions or, equivalently, non-selection of the corresponding predictors.*

indicates the largest λ such that the cross-validatory estimate of error is within 1 standard error of the minimum. This is a reasonable choice for the minimizing λ since it takes cross-validation variability into account.

The following code selects this value of λ using the $lambda.1se component of the `cv.gamsel()` fit object, $\lambda = 0.588$, and then plots the estimated functions. The result is shown in Figure 7.

```
> par(mfrow = c(2,3)) ; index1se <- fit1CVmanual$index.1se
> for (iPred in 1:6)
+ {
+    mainString <- ""
+    if (iPred == 1) mainString <- bquote(paste(lambda," = ",
+                    .(signif(fit1CVmanual$lambda[index1se],3))))
+    plot(fit1CVmanual$gamsel.fit,newx = X,index = index1se,which = iPred,
+         bty = "l",ylim = c(-0.55,0.55),main = mainString)
+ }
```

The minimum 25-fold cross-validation estimates of $f_1, \ldots, f_6$ shown in Figure 7 are close to the true functions from which the data were generated. In particular, the triaging into non-linear, linear and zero matches the truth. This will not always happen, which can be verified if this example is re-run with different seeds.

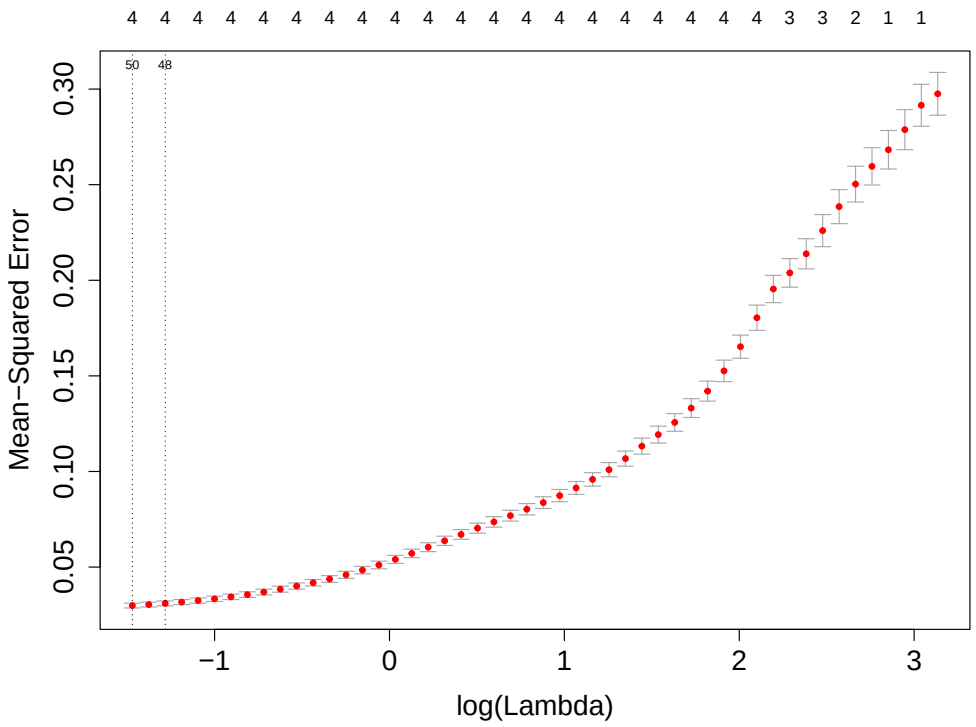

Figure 5: *The 10-fold cross-validation function produced by the default call to* `cv.gamsel()` *for the first simulated data example. This particular realization corresponds to* `set.seed(1)`. *The function is represented as red dots corresponding to an estimate of the mean-squared error and horizontal bars corresponding to plus and minus 1 standard error.*

2.2 Candidate Predictors Both Continuous and Binary

We now consider a simulated data example where not all of the predictors are continuous. Three binary predictors, v_7, v_8 and v_9, are added so that the model becomes:

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

Data from model (2) are generated according to:

```
> f7True <- function(v) return(-0.78*v)
> f8True <- function(v) return(0.53*v)
> set.seed(34) ; n <- 1000
> v7 <- rbinom(n,1,0.5) ; v8 <- rbinom(n,1,0.5) ; v9 <- rbinom(n,1,0.5)
> y <- rnorm(n,f1True(v1) + f2True(v2) + f3True(v3) + f4True(v4)
+            + f7True(v7) + f8True(v8),sigmaTrue)
> X <- data.frame(v1,v2,v3,v4,v5,v6,v7,v8,v9)
```

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.

Since the v_7, v_8 and v_9 data are binary, adjustments in the calls to `gamsel()` and `cv.gamsel()` are necessary. The call to `gamsel()` needs to be:

```
> fit2 <- gamsel(X,y,degrees = c(rep(10,6),rep(1,3)))
```

with the instructions to restrict spline basis functions for the binary variables to have degree equal to 1. The continuous predictors have a degree value of 10 — the default when the `degrees` argument is omitted.

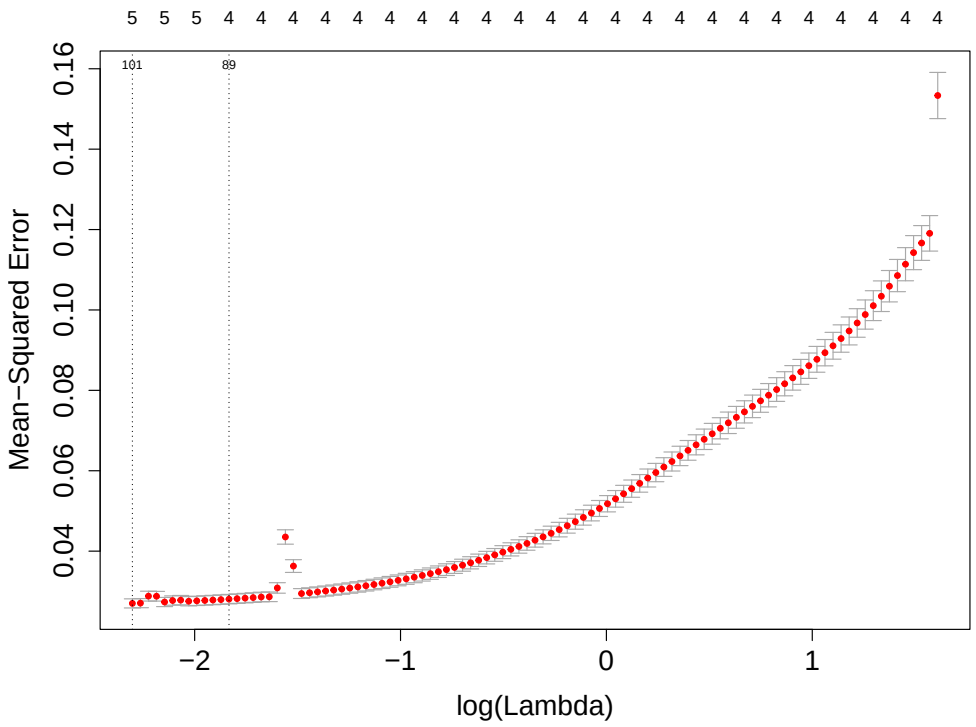

Figure 6: *This plot corresponds to the same data and seed as that shown in Figure 5. The only differences are that, instead of using* `cv.gamsel()`*'s default* λ *path, the path is chosen manually and the cross-validation is 25-fold rather than 10-fold.*

The next code R code chunk sets up a manual λ path and plots the 25-fold cross-validation function produced by `cv.gamsel()`. The result is shown in Figure 8.

```
> set.seed(1)
> fit2CVmanual <- cv.gamsel(X,y,degrees = c(rep(10,6),rep(1,3)),
+                           lambda = myLambdaPath,nfolds = 25)
> par(mfrow = c(1,1)) ; plot(fit2CVmanual)
```

We see from Figure 8 that the $lambda.1se component of the λ path provides a reasonable minimizer of the cross-validation curve. The next piece of code computes the resultant estimates of $f_1, \ldots, f_9$. The result is displayed in Figure 9.

```
> par(mfrow = c(3,3))
> index1se <- fit2CVmanual$index.1se
> for (iPred in 1:9)
+ {
+    mainString <- ""
+    if (iPred == 1)
+       mainString <- bquote(paste(lambda," = ",
+                     .(signif(fit2CVmanual$lambda[index1se],3))))
+    plot(fit2CVmanual$gamsel.fit,newx = X,index = index1se,which = iPred,
+         rugplot = TRUE,bty = "l",ylim = c(-0.55,0.55),main = mainString)
+ }
```

The selected model corresponding to Figure 9, based on a 25-fold cross-validatory choice of λ, does not quite match the true functions from which the data were simulated. Specifically, the true f_4 function is linear and both of the true f_5 and f_6 functions are zero. In this case,

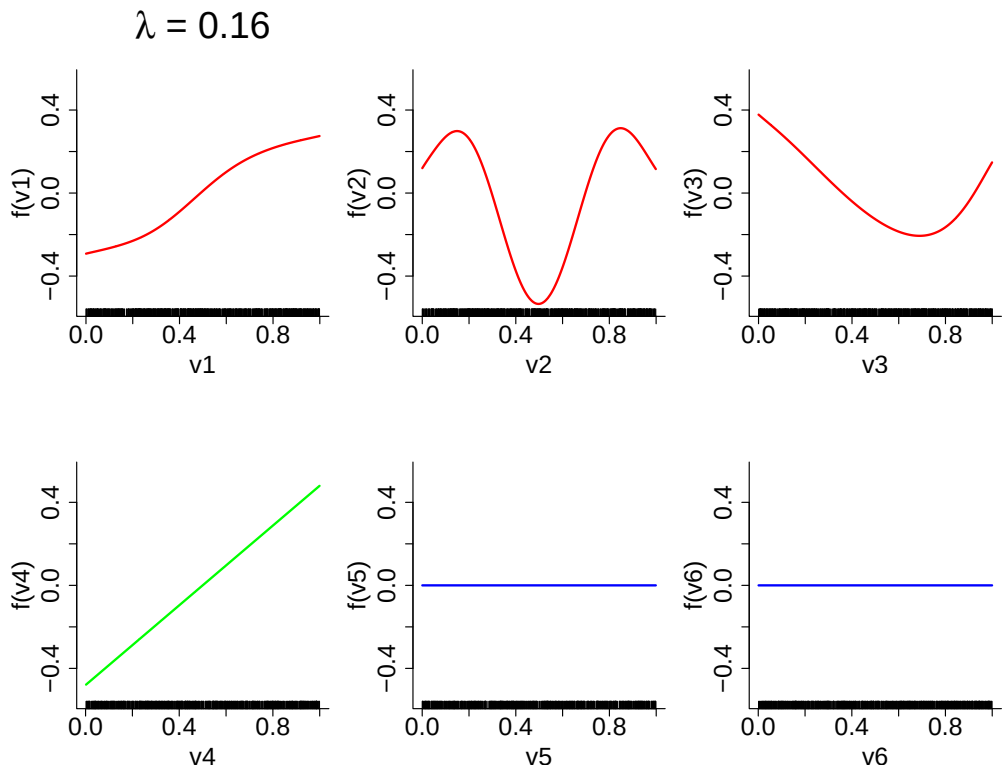

Figure 7: *The* `gamsel()` *estimates of* $f_1, \ldots, f_6$ *for the data simulated according to (1) when* $\lambda = 0.16$. *These graphics are obtained using the function* `plot()` *applied to the fit object* fit1CVmanual *as detailed in the accompanying code. Linear fits are shown in green. Non-linear fits are shown in red. The flat blue lines indicate zero fitted functions or, equivalently, non-selection of the corresponding predictors. The value of* λ *corresponds to the right vertical dotted line in Figure 6, corresponding to the* $lambda.1se *component of the* λ *path.*

`cv.gamsel()`, with λ chosen to be the $index.1se component of the λ path, estimates f_4, f_5 and f_6 to be non-linear functions. The other six functions have correct triaging with respect to the simulation truths.

3 Analysis of Data on House Sales in Sydney, Australia

This example depends upon the R package `HRW`, which accompanies Harezlak, Ruppert & Wand (2018). 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 ; indsContinLater <- c(11:23,25:39)
```

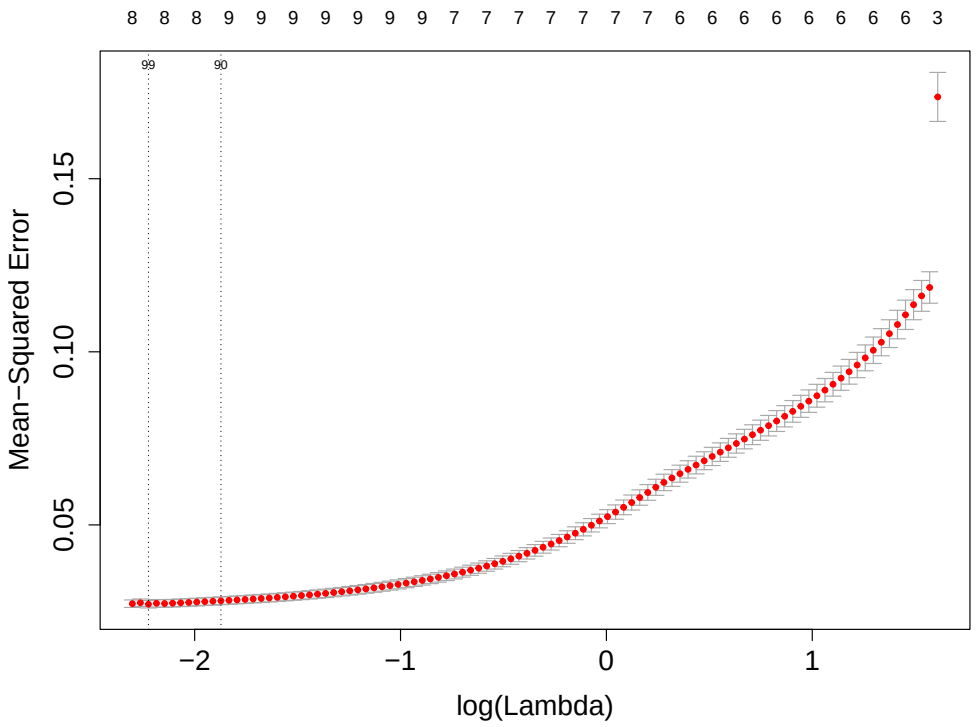

Figure 8: *The 25-fold cross-validation function produced by the call to* `cv.gamsel()` *for the second simulated data example and a manually chosen path of* λ *values.*

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 $\log(x+1)$ transform makes the data more amenable to the `gamsel()` methodology. From now on we work with this transformation of SydneyRealEstate$crimeRate.

The following code produces the X matrix, containing 37 candidate predictors and the y vector containing the response data of logarithmically transformed sale prices:

```
> X <- cbind(SydneyRealEstate[,indsContinEarly],saleQtrEq2,saleQtrEq3,
+            saleQtrEq4,aircraftNoise,log(SydneyRealEstate[,indCrimeRate] + 1),
+            SydneyRealEstate[,indsContinLater])
> names(X)[9] <- "log(crimeRate + 1)"
> y <- SydneyRealEstate[,indResponse]
```

Determine and print the number of unique values of each predictor:

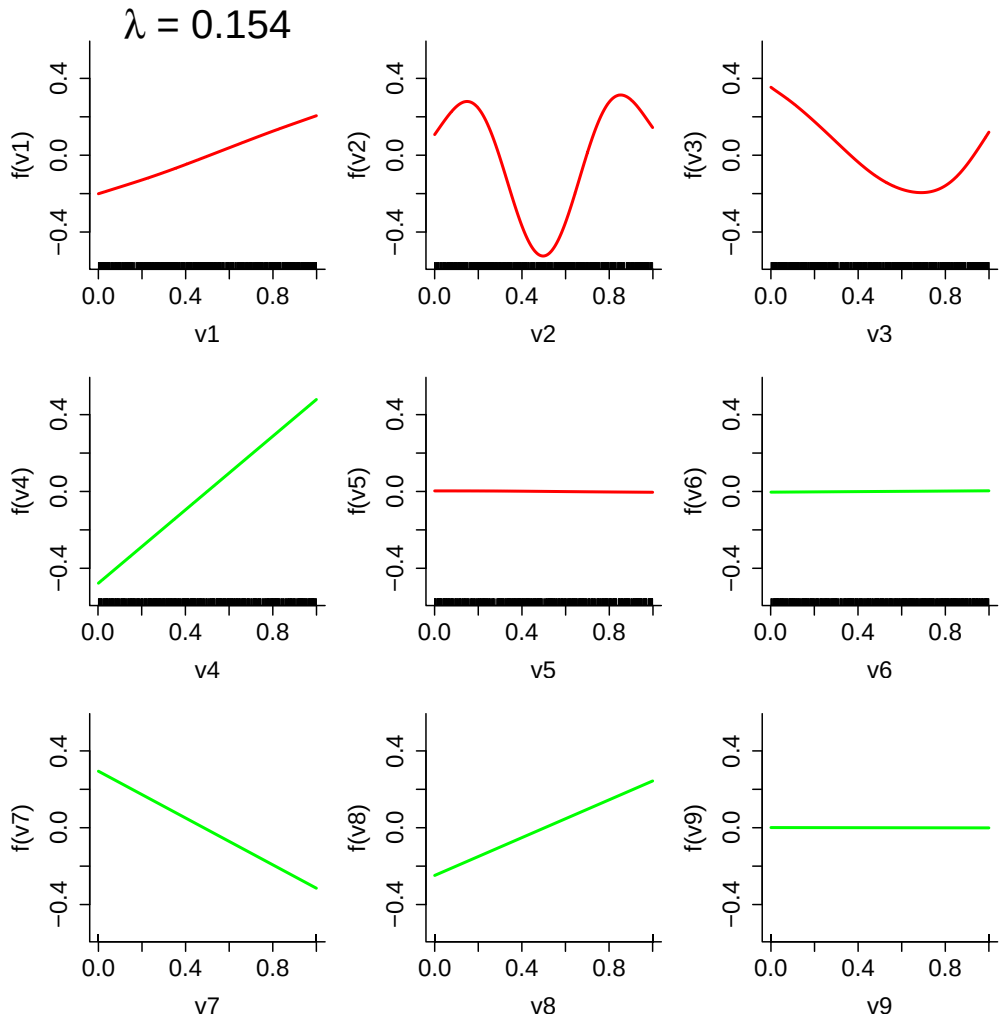

Figure 9: *The* `gamsel()` *estimates of* $f_1, \ldots, f_9$ *for the data simulated according to (2) when* $\lambda = 0.154$. *Linear fits are shown in green. Non-linear fits are shown in red. The value of* λ *corresponds to the right vertical dotted line in Figure 8, corresponding to the* $lambda.1se *component of the* λ *path.*

```
> numUniqVals <- function(x) return(length(unique(x)))
> numUniqInX <- as.numeric(apply(X,2,numUniqVals))
> names(numUniqInX) <- names(X)
> print(numUniqInX)
```

```
          lotSize         longitude          latitude           infRate         saleQtrEq
            18343             35896             35162               858
       saleQtrEq3        saleQtrEq4     aircraftNoise log(crimeRate + 1)            incom
                2                 2                 2                43              128
    distToBusStop   distToCoastline     distToNatPark       distToPark    distToRailLin
            37036             37040             37040             37040             3704
distToRailStation      distToHighway     distToFreeway      distToTunnel    distToMainRoa
            37040             37039             37040             37040             3704
 distToSealedRoad distToUnsealedRoad     foreignerRatio         distToGPO                 N
            36943             37040              4755             37040                 1
              NO2               ozone              neph              PM10                SO
               16                15                12                12
  distToAmbulance     distToFactory      distToFerry     distToHospital      distToMedica
```

```
            37040             37040             37040             37040             3704
      distToSchool   distToUniversity
            37040             37040
```

The last output reveals that, whilst most of the continuous predictors have thousands of unique values, a handful have a small number of unique values. There are also the four binary predictors that we created earlier. The next piece of code manually specifies the value of the degrees argument, corresponding to the maximum numbers of spline basis functions to use for each predictor:

```
> degreesVec <- rep(NA,ncol(X))
> degreesVec[numUniqInX == 2] <- 1 ; degreesVec[numUniqInX == 9] <- 5
> degreesVec[(numUniqInX >= 12) & (numUniqInX <= 16)] <- 8
> degreesVec[numUniqInX > 16] <- 10 ; print(degreesVec)

 [1] 10 10 10 10  1  1  1  1 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  8  8  8  8  8
[31] 10 10 10 10 10 10 10
```

Note that each binary predictor has a degree value of 1. Each continuous predictors with a high number of unique values has a degree value of 10, which is the maximum allowable value in gamsel(). The predictor SydneyRealEstate$SO2 has only 9 unique values and the degree value is set to the lower value of 5. For the predictors having between 12 and 16 unique values we use a maximum of 8 spline basis functions.

3.1 Example Fits Along the λ Path

With the spline basis degrees sorted out, we now obtain the gamsel() fit object for the Sydney real estate data:

```
> fitSydneyRealEstate <- gamsel(X,y,degrees = degreesVec)
```

A visual summary of the fit is provided by:

```
> par(mfrow = c(2,1)) ; summary(fitSydneyRealEstate)
```

and shown in Figure 10. For $\lambda = 50$ only 3 of the 37 predictors are selected, all having linear effects. At $\lambda = 5$ it is seen that several predictors are selected including 4 with non-linear effects.

We now look at some fits along the λ path, with index numbers 10, 25 and 40 of the λ path, starting with the highest λ:

```
> indexVec <-  c(10,25,40)
> lambdaVec <- signif(fitSydneyRealEstate$lambdas[indexVec],3)
```

Now run the plotting code:

```
> ilambda <- 1
> nonzeroPreds <- getActive(fitSydneyRealEstate,index = indexVec[ilambda])[[1]]
> lambdaString <- bquote(paste(lambda," = ",.(signif(lambdaVec[ilambda],3))))
> par(mfrow = c(2,2))
> plot(0,type = "n",xlim = c(0,1),ylim = c(0,1),xlab = "",
+      ylab = "",bty = "o",xaxt = "n",yaxt = "n")
> text(0.5,0.5,lambdaString,cex = 3)
> for (iPlot in 1:length(nonzeroPreds))
+ {
+    iPredCurr <- nonzeroPreds[iPlot]
```

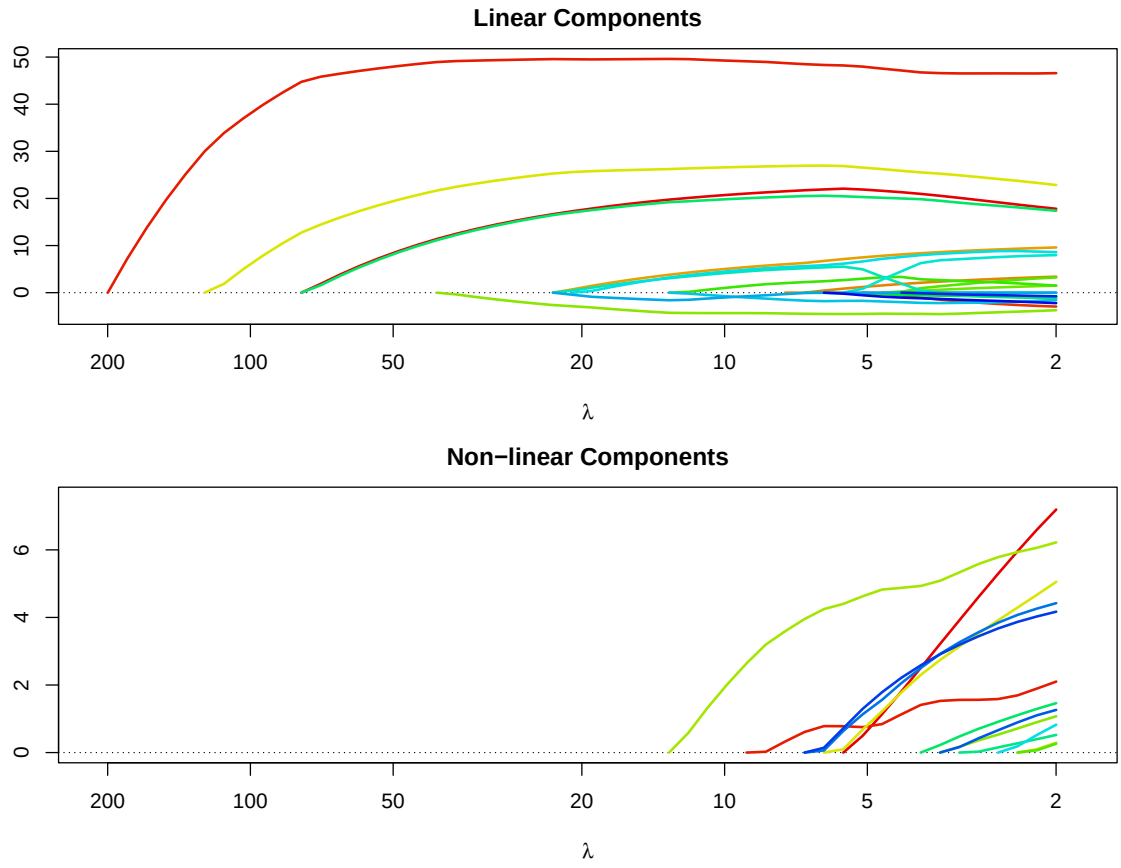

Figure 10: *The plots produced from the command* summary(fitSydneyRealEstate) *for the* gamsel() *fit, stored as* fitSydneyRealEstate, *to the Sydney real estate data.*

```
+    plot(fitSydneyRealEstate,newx = X,index = indexVec[ilambda],
+         which = iPredCurr,bty = "l",ylim = c(-0.2,0.2),
+         main = names(X)[iPredCurr])
+ }
```

The last code chunk leads to Figure 11 and corresponds to $\lambda = 85.8$. This time we only show the non-zero predictor effects, of which there are only 2: longitude and income. The first of these corresponds to the west-to-east location of the houses. The high slope for longitude is explained by proximity to the attractive Pacific Ocean coastline and beaches of eastern Sydney, and the prettier parts of Sydney Harbour.

If we set:

```
> ilambda <- 2 ; par(mfrow = c(3,3))
```

and re-run the above graphics code then we obtain Figure 12, which shows the fits at $\lambda = 20.9$ near the middle of the λ path. We see that 8 of predictors are selected at $\lambda = 20.9$, all having linear effects. From Figure 13 we see that gamsel() selects 17 predictors when $\lambda = 5.12$ and 6 of these have non-linear effects. The "hockey stick" for distToCoastline is due to many of Sydney's most affluent suburbs being 10-20 kilometres away from the coastline and near parts of Sydney Harbour and nature reserves.

For the last of the three λ values the commands:

```
> ilambda <- 3 ; par(mfrow = c(6,3))
```

followed by a re-running of the above fit plotting code lead to Figure 13.

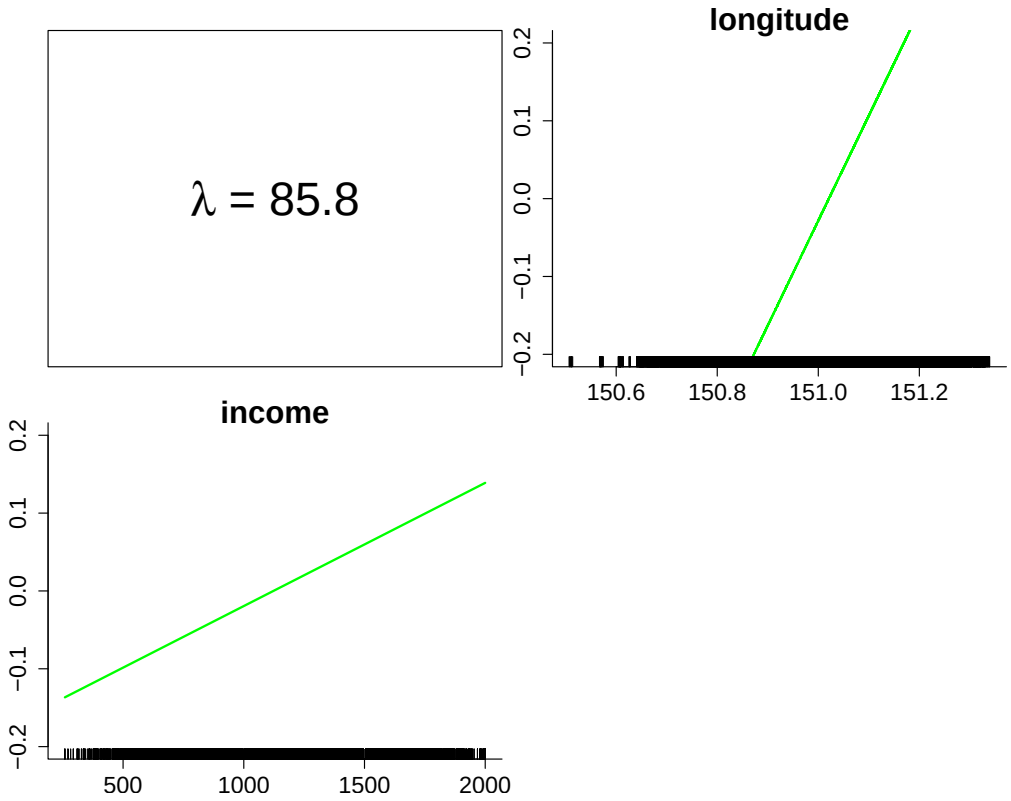

Figure 11: *The only 2 selected predictors for the* `gamsel()` *fit to the Sydney real estate data when* $\lambda = 85.8$. *Both effects are linear.*

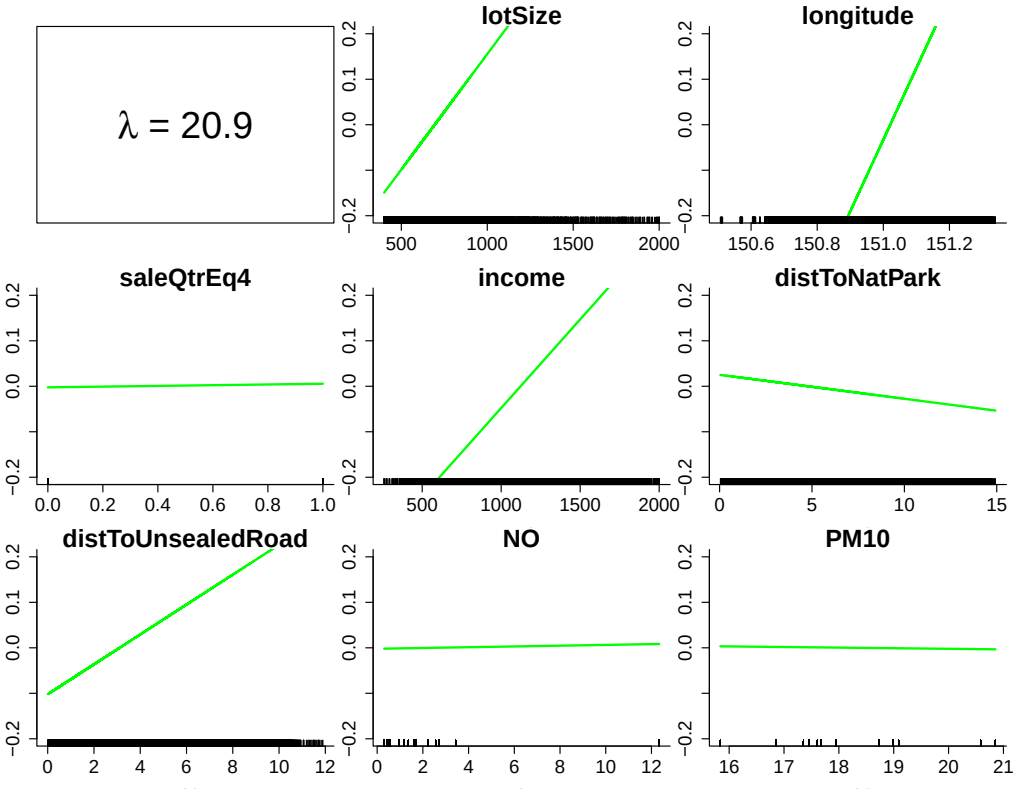

Figure 12: *The 8 selected predictors for the* `gamsel()` *fit to the Sydney real estate data when* $\lambda = 20.9$. *All effects are linear.*

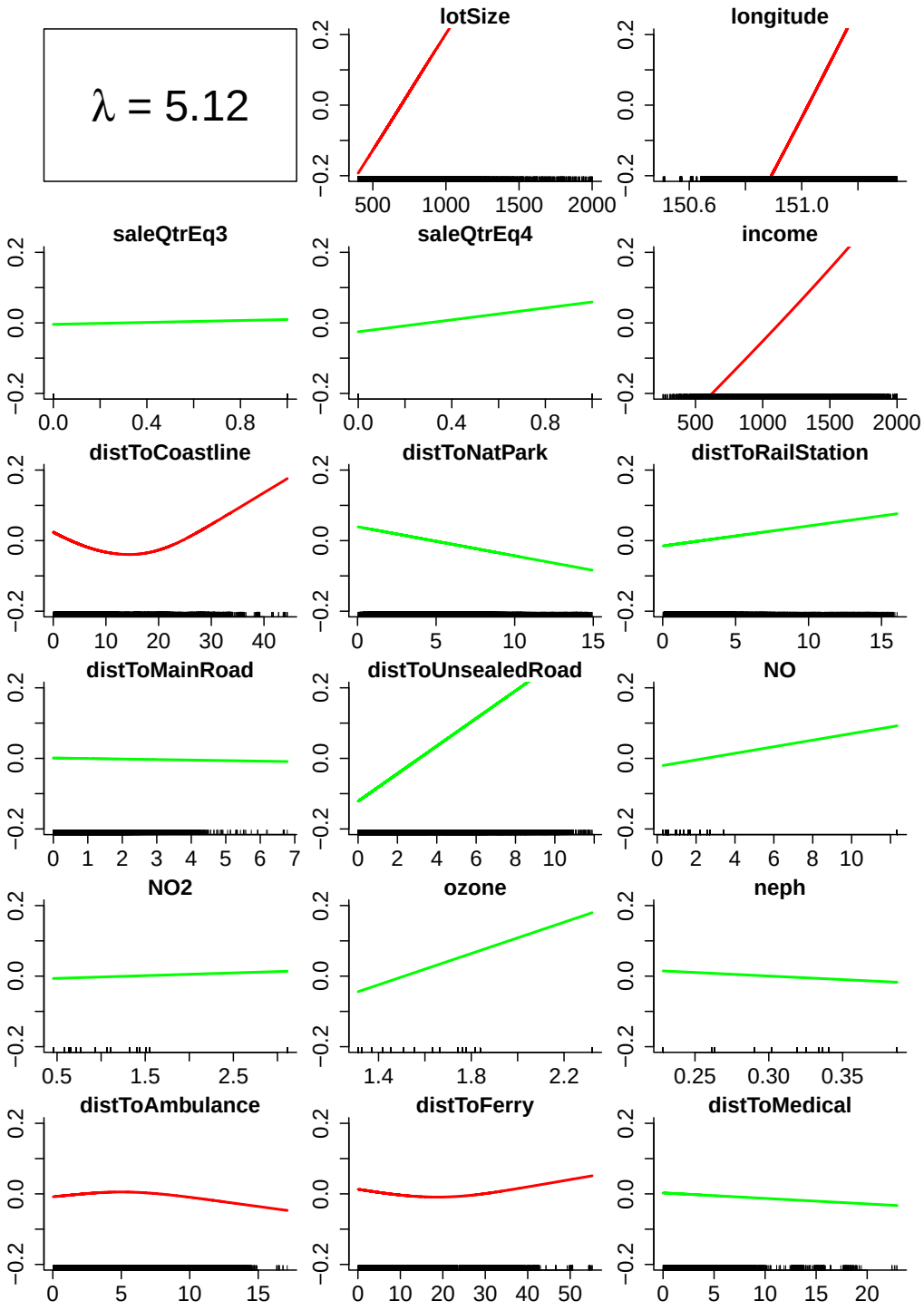

Figure 13: *The 17 selected predictors for the* `gamsel()` *fit to the Sydney real estate data when* $\lambda = 5.12$. *Six of effects are non-linear and 11 are linear.*

3.2 Selection of λ Via Cross-Validation

After some experimentation, the following λ path was found to be reasonable for the Sydney real estate data:

```
> myLambdaPath <- exp(seq(log(0.1),log(5),length = 50))
```

Now obtain and plot the `cv.gamsel()` object with set.seed(1) and lambda = myLambdaPath:

```
> set.seed(1)
```

```
> fitSydneyRealEstateCVmanual <- cv.gamsel(X,y,degrees = degreesVec,
+                                          lambda = myLambdaPath)

> par(mfrow=c(1,1)) ; plot(fitSydneyRealEstateCVmanual)

> par(mfrow=c(1,1))
> cex.labVal <- 1.5 ; cex.axisVal <- 1.5 ; cex.mainVal <- 1.5
> plot(fitSydneyRealEstateCVmanual,cex.lab = cex.labVal,
+      cex.axis = cex.axisVal,cex.main = cex.mainVal)
```

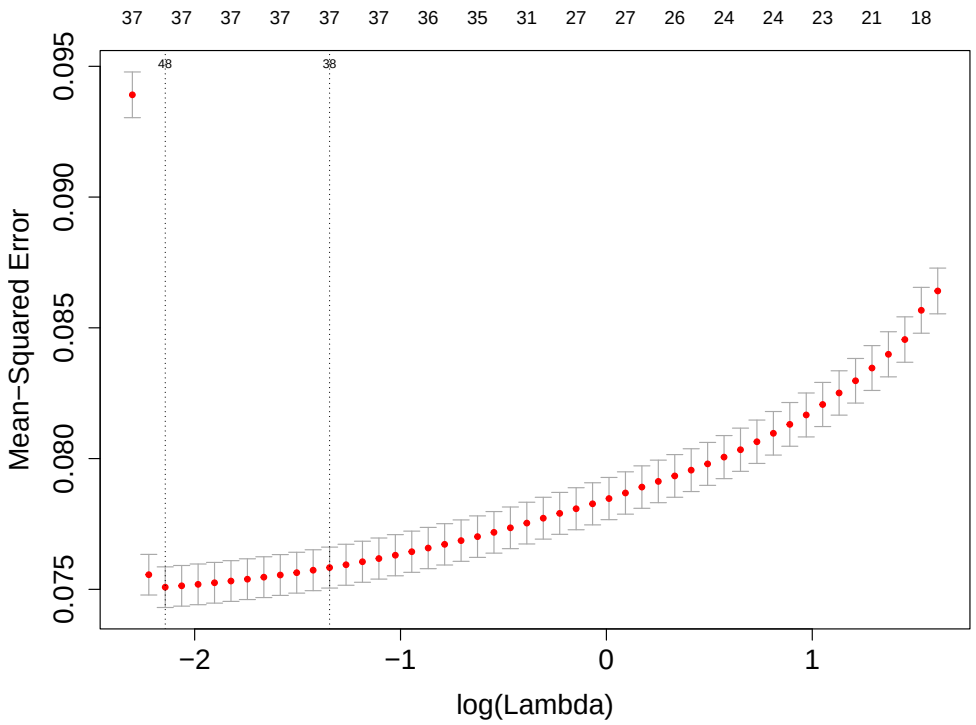

Figure 14: *The 10-fold cross-validation function produced by the call to* `cv.gamsel()` *for the Sydney real estate data example and a manually chosen path of* λ *values.*

Figure 14 shows the resulting 10-fold cross-validation function. We see that there is a global minimum at the 48th lowest value of λ. The 38th lowest value corresponds to the one standard error minimum and, as before, is the $lambbda.1se component of the `cv.gamsel()` fit object and equals $\lambda = 0.261$.

The estimated predicted effects corresponding to this 10-fold cross-validatory choice of λ is shown in Figure 15.

A first remark about Figure 15 is that *every* candidate predictor is in model. Also, all but one of the continuous predictors has a non-linear effect for this `gamsel()` fit. The strengths of the response/predictor relationships in these data combined with the high sample size partially explains this outcome. Depending on the goals of the generalized additive model analysis having such a high number of selected predictors may or may not be desirable. If a more parsimonious model is of interest then sparser fits along the λ path should be considered rather than k-fold cross-validation minimization.

Lastly, we note that the additivity assumption for the predictors longitude and latitude is quite tenuous. In Section 5.3.1 of Harezlak *et al.* (2018) the authors illustrate an extension of the generalized additive model which contains a *bivariate* function of longitude and latitude.

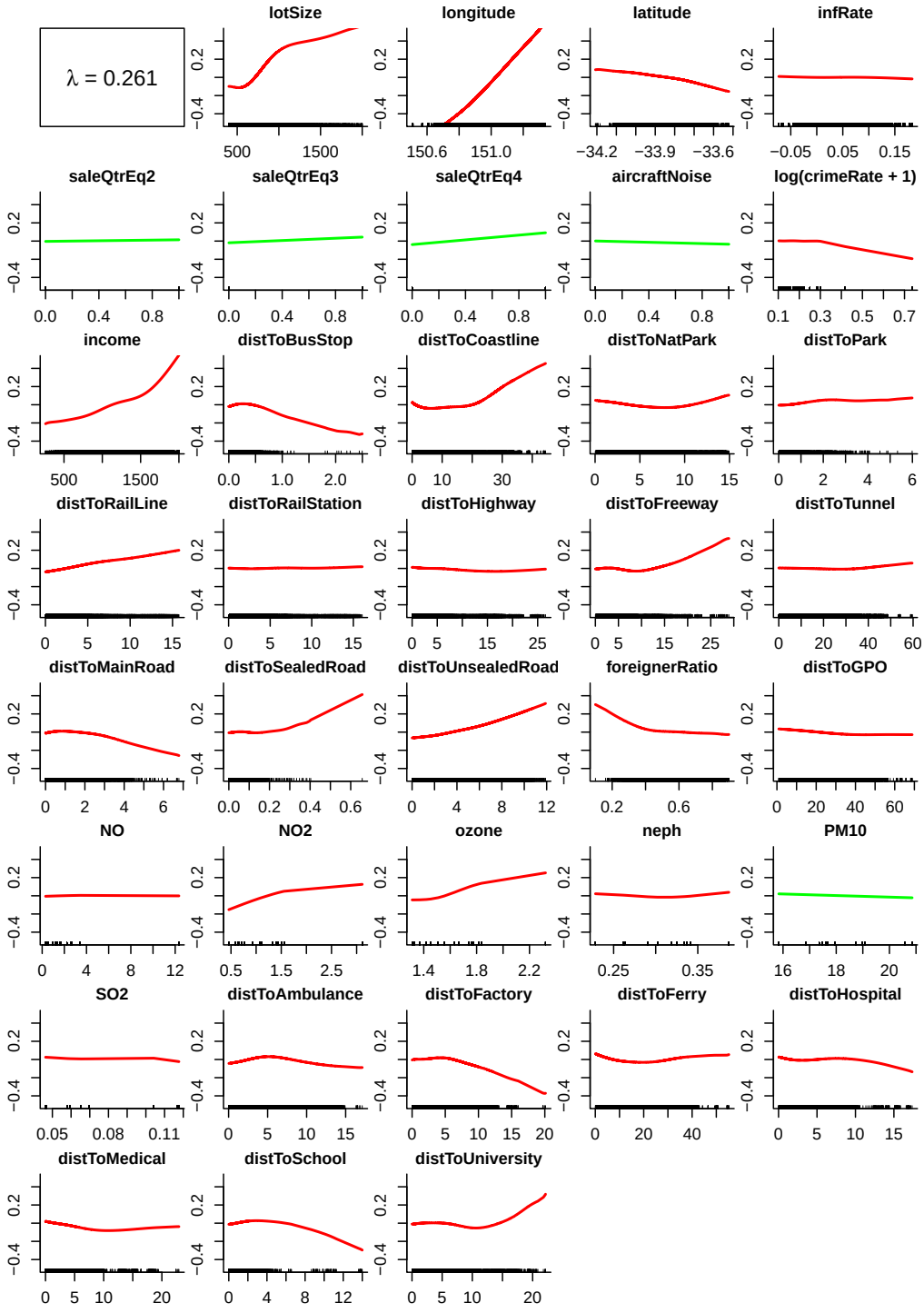

Figure 15: *The* `gamsel()` *estimated predictor effects for the Sydney real estate data when* $\lambda = 0.261$. *Linear fits are shown in green. Non-linear fits are shown in red. The value of* λ *corresponds to the right vertical dotted line in Figure 14, corresponding to the* `$lambda.1se` *component of the* `cv.gamsel()` *fit object.*

Analysis is achieved via the R package `mgcv` (Wood, 2019). The `gamsel` package only supports univariate functions, so bivariate effects cannot be incorporated in a natural way. However, *ad hoc* model selection involving residuals from of an estimated bivariate function of geographical position may be considered.

4 Analysis of Car Auction Data

The final 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.

Obtain the X matrix, containing 49 candidate predictors and the y vector containing indicators of whether or not a car was considered to be a bad purchase:

```
> X <- carAuction[,3:51] ; y <- carAuction[,2]
```

Determine and print the number of unique values of each predictor:

```
> numUniqVals <- function(x) return(length(unique(x)))
> numUniqInX <- as.numeric(apply(X,2,numUniqVals))
> names(numUniqInX) <- names(X) ; print(numUniqInX)
```

```
      purchIn2010         aucEqAdesa       aucEqManheim         vehYearEq03
                2                  2                  2                  2
      vehYearEq04         vehYearEq05         vehYearEq06         vehYearEq07
                2                  2                  2                  2
        ageAtSale      makeEqChevrolet          makeEqFord         makeEqDodge
               10                  2                  2                  2
   makeEqChrysler           trimEqBas           trimEqLS           trimEqSE
                2                  2                  2                  2
subModelEq4DSEDAN subModelEq4DSEDANLS subModelEq4DSEDANSE      colourEqSilver
                2                  2                  2                  2
    colourEqWhite        colourEqBlue        colourEqGrey       colourEqBlack
                2                  2                  2                  2
      colourEqRed        colourEqGold      colourEqOrange       transEqManual
                2                  2                  2                  2
     wheelEqAlloy       wheelEqCovers            odomRead        AmericanMade
                2                  2              39947                  2
   otherAsianMade         sizeEqTruck        sizeEqMedium           sizeEqSUV
                2                  2                  2                  2
    sizeEqCompact           sizeEqVan              price       purchInTexas
                2                  2              10343                  2
    purchInFlorida    purchInCalifornia purchInNorthCarolina      purchInArizona
                2                  2                  2                  2
 purchInColorado purchInSouthCarolina         costAtPurch          onlineSale
                2                  2               2072                  2
     warrantyCost
              281
```

It is apparent from the last command that most of the predictors are binary, corresponding to indicators of certain automobile features. For example, purchInTexas equals 1 for cars purchased in the state of Texas, and 0 otherwise.

Next we set the vector of degree values. The two continuous predictors having several hundred unique values are given degree values of 10. The binary predictors have 1 degree. The predictor ageAtSale has only 10 unique values, so we set its degree value to 5:

```
> degreesVec <- rep(10,ncol(X)) ; degreesVec[numUniqInX == 2] <- 1
> degreesVec[numUniqInX == 10] <- 5 ; print(degreesVec)

 [1]  1  1  1  1  1  1  1  1  5  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
[31] 10  1  1  1  1  1  1  1 10  1  1  1  1  1  1  1 10  1 10
```

4.1 Example Fits Along the λ Path

Now are ready to send the data to gamsel(). Since the response data are binary it is appropriate to set family = "binomial". The fit command is:

```
> fitCarAuction <- gamsel(X,y,degrees = degreesVec,family = "binomial")
```

As with previous examples, we start with a graphical summary of the gamsel() fit object across the λ path:

```
> par(mfrow = c(2,1)) ; summary(fitCarAuction)
```

The result is shown in Figure 16. It shows, for example, that the default gamsel() fits are sparse, relative to the 49 candidate predictors, across the λ path. Also, only one of the three continuous predictors has a non-linear effect for lower values of λ.

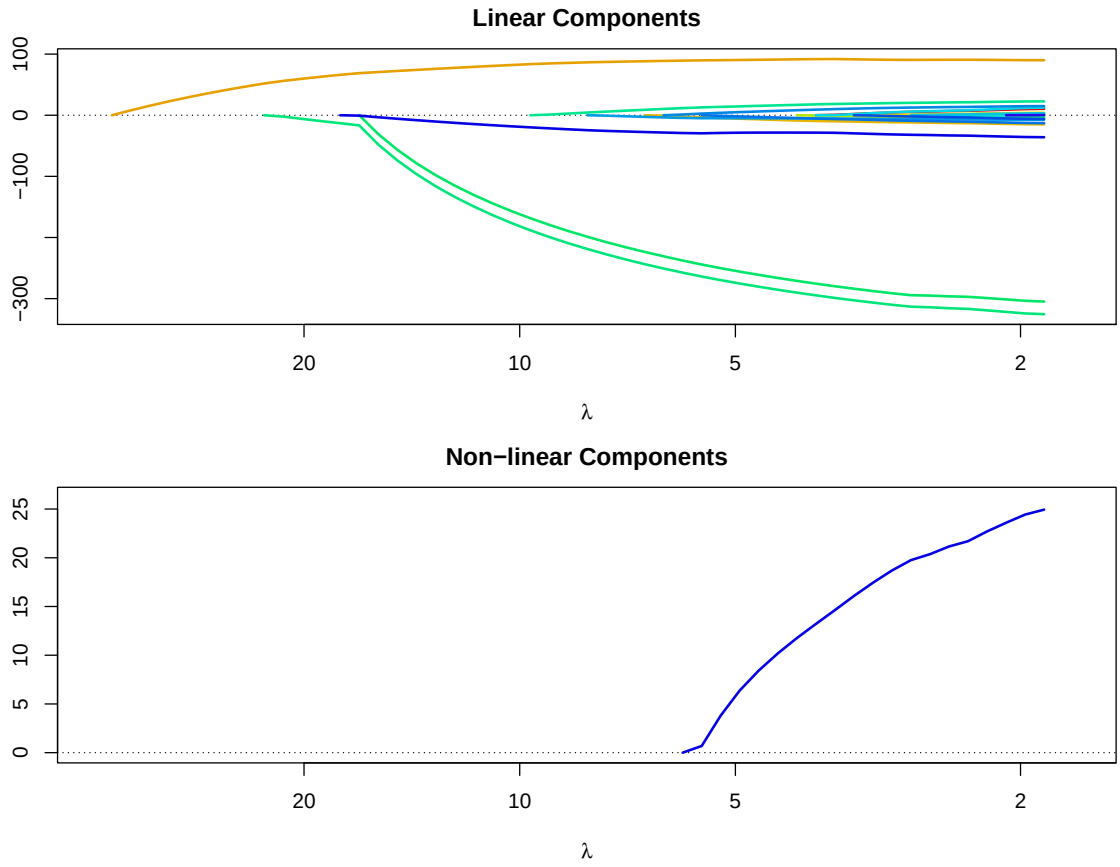

Figure 16: *The plots produced from the command* summary(fitCarAuction) *for the* gamsel() *fit, stored as* fitCarAuction, *to the car auction data.*

Next is to look at example fits along the λ path:

```
> indexVec <-  c(10,25,40)
> lambdaVec <- signif(fitCarAuction$lambdas[indexVec],3)
```

The code for showing the fits for the first example λ value is:

```
> ilambda <- 1
> nonzeroPreds <- getActive(fitCarAuction,index = indexVec[ilambda])[[1]]
> lambdaString <- bquote(paste(lambda," = ",.(signif(lambdaVec[ilambda],3))))
> par(mfrow = c(2,2))
> plot(0,type = "n",xlim = c(0,1),ylim = c(0,1),xlab = "",
+      ylab = "",bty = "o",xaxt = "n",yaxt = "n")
> text(0.5,0.5,lambdaString,cex = 2)
> for (iPlot in 1:length(nonzeroPreds))
+ {
+    iPredCurr <- nonzeroPreds[iPlot]
+    plot(fitCarAuction,newx = X,index = indexVec[ilambda],
+         which = iPredCurr,bty = "l",ylim = c(-2.5,2.5),
+         main = names(X)[iPredCurr])
+ }
```

The result for $\lambda = 21.4$, corresponding to the 10th highest value along the λ path, is shown in Figure 17. We see that only 2 of the 49 predictors are selected by `gamsel()` for this high

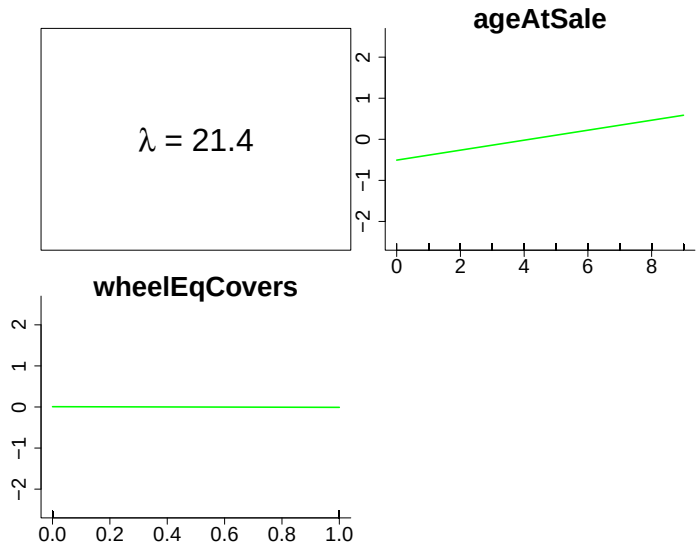

Figure 17: *The only 2 selected predictors for the* `gamsel()` *fit to the car auction data when* $\lambda = 21.4$. *Both effects are linear.*

choice of λ. In keeping with intuition, car age is seen to have a strong positive effect on the probability of a bad purchase.

If the same plotting code is re-run with:

```
> ilambda <- 2 ; par(mfrow = c(3,2))
```

then Figure **??** results. With $\lambda = 8.55$ we see from Figure **??** that 5 predictors are selected. Three of these predictors, ageAtSale, odomRead and costAtPurch, are continuous but have a linear effect at $\lambda = 8.55$.

Lastly, we set:

```
> ilambda <- 3 ; par(mfrow = c(5,3))
```

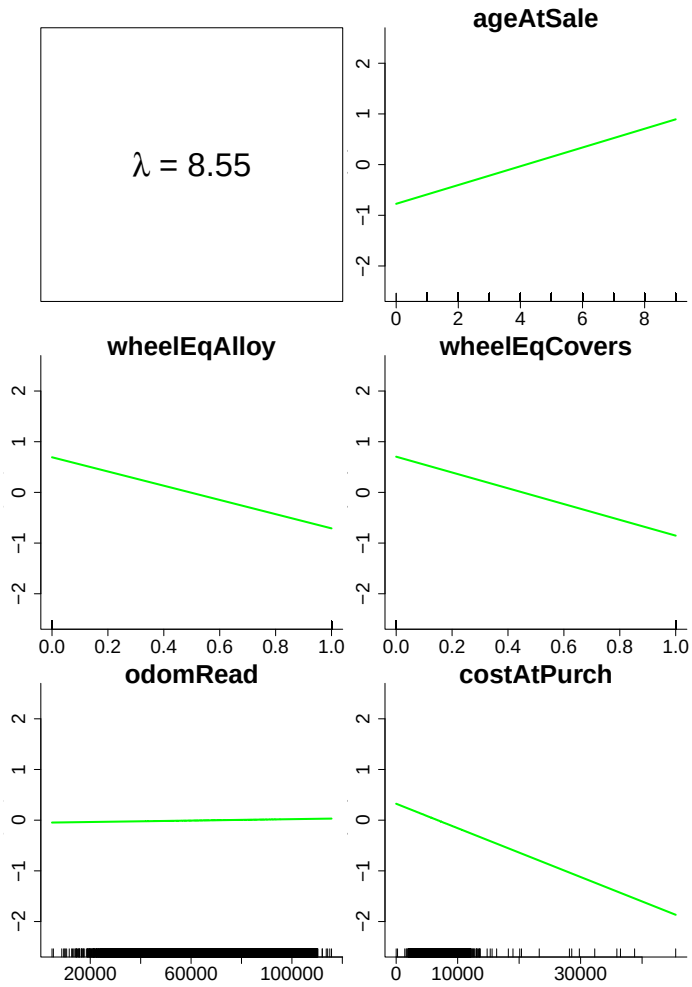

Figure 18: *The 5 selected predictors for the* `gamsel()` *fit to the car auction data when* $\lambda = 8.55$. *All effects are linear.*

and run the above plotting code one last time. The result is shown in Figure **??**.

In Figure **??**, with the low value of $\lambda = 3.42$, the greater freedom afforded to `gamsel()` leads to it choosing 12 predictors. This time costAtPurch has a non-linear "hockey stick" effect. It is apparent that cars with a cost at purchase around the median are less likely to be a bad buy whereas those with more extreme cost at purchase values (in either direction) are more likely to be a bad buy.

4.2 Selection of λ Via Cross-Validation

We use the same λ path as for the Sydney real estate data:

```
> myLambdaPath <- exp(seq(log(0.1),log(5),length = 50))
> set.seed(1)
> fitCarAuctionCVmanual <- cv.gamsel(X,y,degrees = degreesVec,
+                                    lambda = myLambdaPath,
+                                    family = "binomial")

> par(mfrow=c(1,1)) ; plot(fitCarAuctionCVmanual)
```

Figure **??** shows the 10-fold cross-validation function The $lambbda.1se component of the `cv.gamsel()` fit object is $\lambda = 1.10$. Figure **??** displays the predictor effects corresponding to this cross-validatory choice of λ.

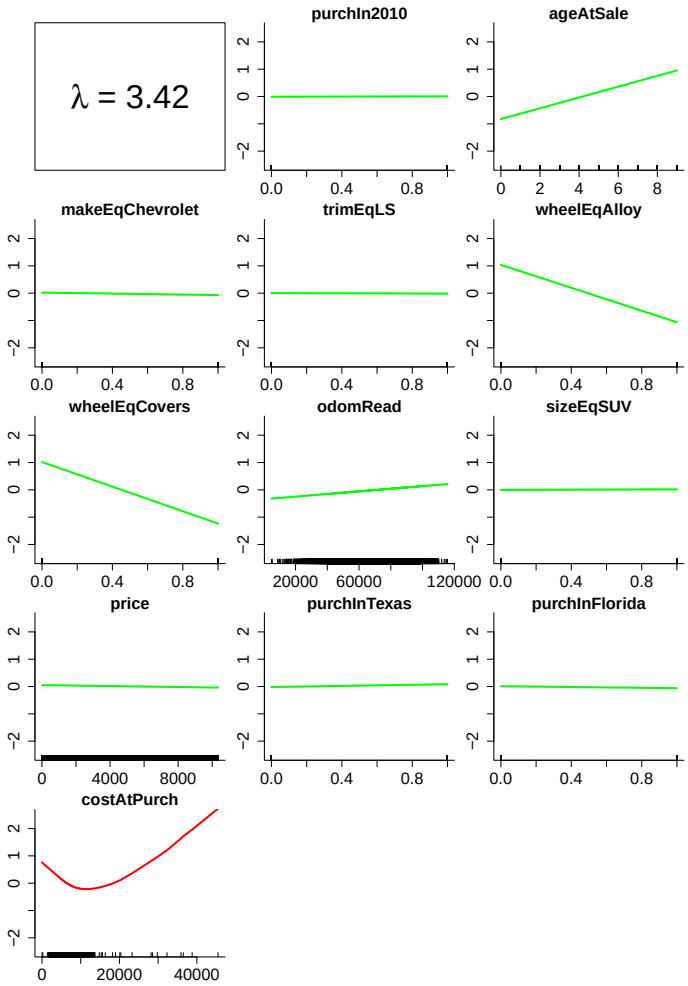

Figure 19: *The 12 selected effects for the* `gamsel()` *fit to the car auction data when* $\lambda = 3.42$. *Eleven of the effects are linear and 1 is non-linear.*

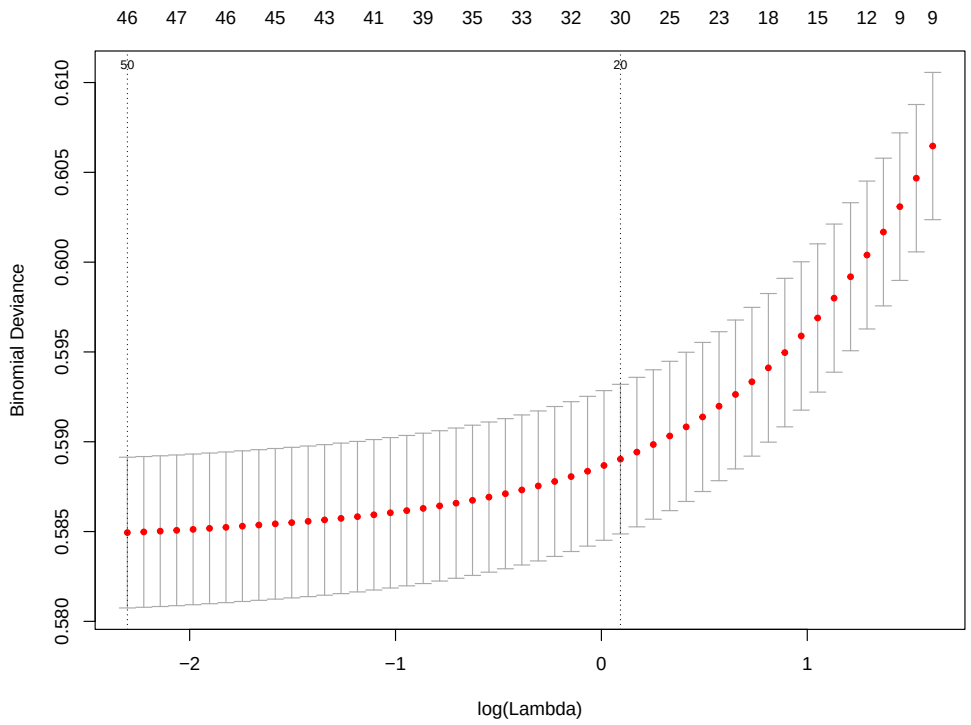

Figure 20: *The 10-fold cross-validation function produced by the call to* `cv.gamsel()` *for the car auction data example and a manually chosen path of* λ *values.*

This time 10-fold cross-validation selects only about 60% of the candidate predictors. Nineteen of the 49 candidate predictors are discarded by `cv.gamsel()`.

References

Chouldechova, A., and Hastie, T. (2018). Generalized additive model selection. Unpublished manuscript. https://arxiv.org/pdf/1506.03850

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

Wood, S.N. (2019). `mgcv`: Mixed GAM computation vehicle with GCV/AIC/REML smoothness estimation. R package version 1.8. http://cran.r-project.org.

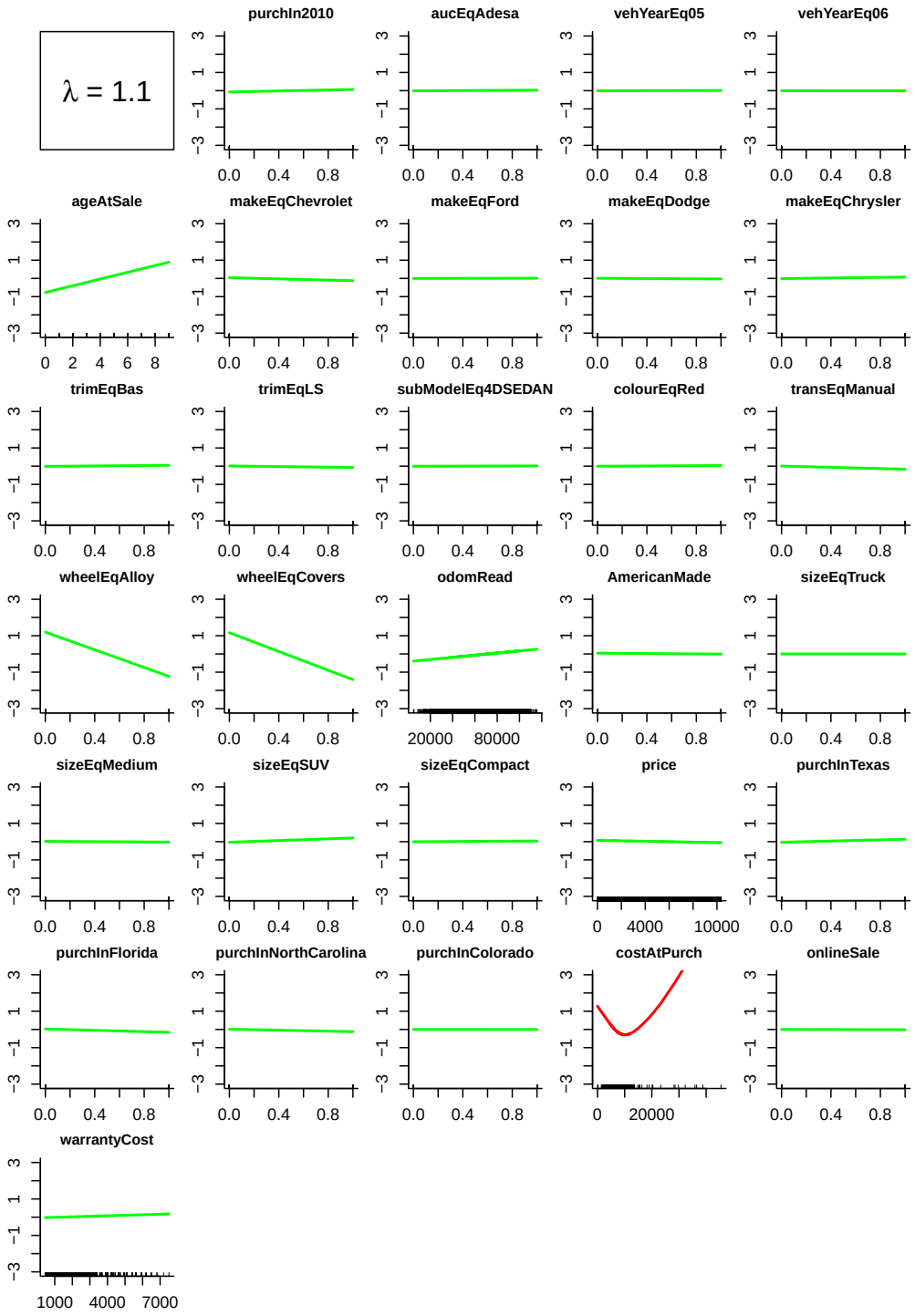

Figure 21: *The* `gamsel()` *estimated predictor effects for the car auction data when* $\lambda = 1.10$. *Linear fits are shown in green. The non-linear fits is shown in red. The value of* λ *corresponds to the right vertical dotted line in Figure 14, corresponding to the* `$lambda.1se` *component of the* `cv.gamsel()` *fit object.*