Final Exam

There are five questions (50 total points). Please type in your answers directly in the R Markdown file. After completion, **successfully** knitr it as an html file. Submit **both** the html file and the R Markdown file via Canvas. Please name the R Markdown file in the following format: LastName_FirstName_Final.Rmd, e.g. Zhao_Zifeng_Final.Rmd.

NYZ Times Dataset (50 points)

NYZ Times is a newspaper publishing company. The company maintains a dataset on their last marketing campaigns for promoting online subscription. For each individual customer, the dataset contains the campaign result (Subscriber = Yes/No) and the corresponding customer characteristics. There are around 5,000 individual records in the dataset. It contains 7 variables `Age`, `Experience`, `Income`, `Family`, `AvgAmt`, `Education` and `Subscriber`. The data description is as follows.

- `Age`: Age of the customer (in years)
- `Experience`: Working experience of the customer (in years)
- `Income`: Annual income of the customer ($000)
- `Family`: Family size of the customer
- `AvgAmt`: Average amount of online spending per month by the customer ($00)
- `Education`: Education level of the customer (Bachelor / Master / PhD)
- `Subscriber`: Did the marketing campaign convert the customer into a subscriber? (Yes / No)

To promote online subscription more efficiently and effectively, NYZ Times decides to build several statistical models that can predict the conversion probability of each customer, and thus selectively send advertisement to potential customers that have higher conversion probability.

```
rm(list=ls())
```

Q1 [4 points] Read in Data and Data Partition

Q1(a) [1 point] Let's correctly read in the csv data and name it as `total_data`.

```
#reading in the total data as a csv,
total_data <- read.csv('/Users/TomTheIntern/Desktop/Mendoza/Mod 2/Advanced Stats/Final Exam/NYZTimes.csv
                       header=T, #with headers as true
                       stringsAsFactors=T) #and setting strings to factor
```

Q1(b) [1 point] What is the conversion rate in the current dataset stored in `total_data`?

```
#getting the number of yes counts from the total data
count_yes <- sum(total_data$Subscriber == 'Yes')
#showing the number of yes counts
count_yes
```

```
## [1] 467
```

```
#getting the number of no counts from the total date
count_no <- sum(total_data$Subscriber == 'No')
#showing the number of no counts
count_no
```

```
## [1] 4333
```

```
#dividing the number of yes counts by the sum of yes and no counts
prop <- count_yes / (count_yes + count_no)

prop
```

```
## [1] 0.09729167
```

Answer: The conversion rate is 0.0973 or roughly 9.73%. There are 4,333 counts of no and 467 counts of yes.

Q1(c) [2 points] Let's partition the data in `total_data` into training **(60%)** and test data **(40%)** and store them as `R` objects `train_data` and `test_data` respectively. Use random seed **`set.seed(7)`**!

```
#setting seed so we can repeat the code set
set.seed(7)
#storing the number of total observations into total obs
total_obs <- nrow(total_data)

#getting a random set of indicies
train_index <- sample(1:total_obs, 0.6*total_obs)

#setting up the train data
train_data <- total_data[train_index,]

#setting up the test data
test_data <- total_data[-train_index,]
```

Q2 [10 points] Logistic Regression

Q2(a) [4 points] Fit a logistic regression model of the dependent variable `Subscriber` w.r.t. all 6 predictors `Age`, `Experience`, `Income`, `Family`, `AvgAmt`, `Education` using the training data, name it `lm_full`.

```
#making a log model predicting subscribers
lm_full <- glm(Subscriber ~ Age + Experience + Income + Family +
                 AvgAmt + Education,
               data = train_data, #using the train data
               family = 'binomial') #and setting the family to binomial
```

Q2(b) [2 points] Examine the estimation result of `lm_full`. Are customers with higher Income more likely to be Subscribers?

```
#setting a summary of lm_full
summary(lm_full)
```

```
##
## Call:
## glm(formula = Subscriber ~ Age + Experience + Income + Family +
##     AvgAmt + Education, family = "binomial", data = train_data)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -14.061172   2.086783  -6.738 1.60e-11 ***
## Age               0.041553   0.076849   0.541 0.588707
## Experience       -0.033969   0.076793  -0.442 0.658241
## Income            0.058243   0.003387  17.197  < 2e-16 ***
## Family            0.593458   0.092490   6.416 1.39e-10 ***
## AvgAmt            0.172582   0.051465   3.353 0.000798 ***
## EducationMaster   3.747930   0.315896  11.864  < 2e-16 ***
## EducationPhD      3.774926   0.311490  12.119  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1924.66  on 2879  degrees of freedom
## Residual deviance:  804.96  on 2872  degrees of freedom
## AIC: 820.96
##
## Number of Fisher Scoring iterations: 7
```

Answer: The Income predictor has a coefficient of 0.058243 for the lm_full model, which suggests that customers with higher Income are more likely to be subscribers.

Q2(c) [2 points] Let's further conduct a backward selection for `lm_full` via BIC using the `step()` function and store the final selected model as `lm_bwd`. Make sure you use the **correct** number for the argument `k` in the `step()` function.

```
#using backward selection with a k as the log of the number of trows of train data
lm_bwd <- step(lm_full, direction = 'backward', k = log(nrow(train_data)))
```

```
## Start:  AIC=868.68
## Subscriber ~ Age + Experience + Income + Family + AvgAmt + Education
##
##              Df Deviance     AIC
## - Experience  1   805.15  860.91
## - Age         1   805.24  861.00
## <none>            804.96  868.68
## - AvgAmt      1   816.44  872.20
## - Family      1   850.09  905.85
## - Education   2  1069.16 1116.96
## - Income      1  1356.61 1412.37
##
## Step:  AIC=860.91
## Subscriber ~ Age + Income + Family + AvgAmt + Education
##
##             Df Deviance     AIC
## - Age        1   806.05  853.84
## <none>           805.15  860.91
```

```
## - AvgAmt     1   816.57  864.37
## - Family     1   850.20  897.99
## - Education  2  1078.39 1118.22
## - Income     1  1358.30 1406.09
##
## Step:  AIC=853.84
## Subscriber ~ Income + Family + AvgAmt + Education
##
##             Df Deviance     AIC
## <none>           806.05  853.84
## - AvgAmt     1   817.05  856.88
## - Family     1   850.39  890.22
## - Education  2  1080.75 1112.61
## - Income     1  1358.30 1398.13
```

Q2(d) [2 points] Which variables are removed during the backward selection?

Answer: The resulting model retained the AvgAmt, Family, Education and Income variables, meaning it removed Age and Experience.

Q3 [8 points] Generalized Additive Model

Q3(a) [4 points] To capture potential nonlinear relationship, we fit a GAM model of the dependent variable `Subscriber` w.r.t. all 6 predictors `Age`, `Experience`, `Income`, `Family`, `AvgAmt`, `Education` using the training data, name it `gam1`. Let's use splines with **df=4** for all **5** numerical predictors, which include `Age`, `Experience`, `Income`, `Family` and `AvgAmt`.

```
#adding in the gam library
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.4.1
```

```
## Loading required package: splines
```

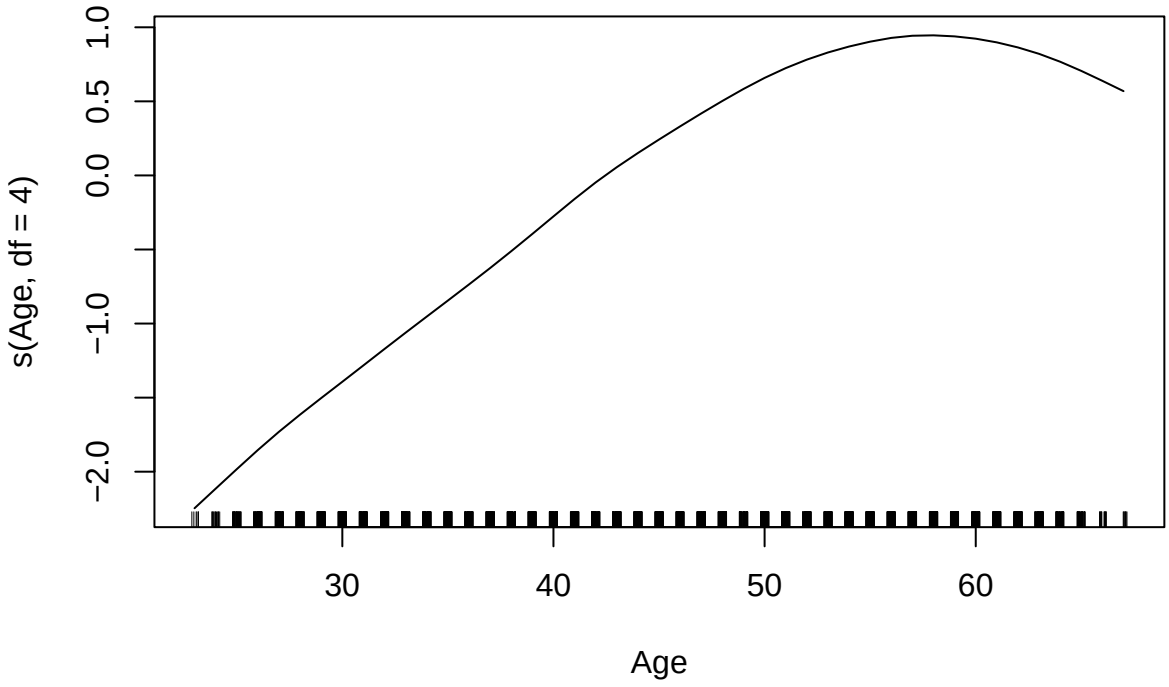
```
## Loading required package: foreach
```

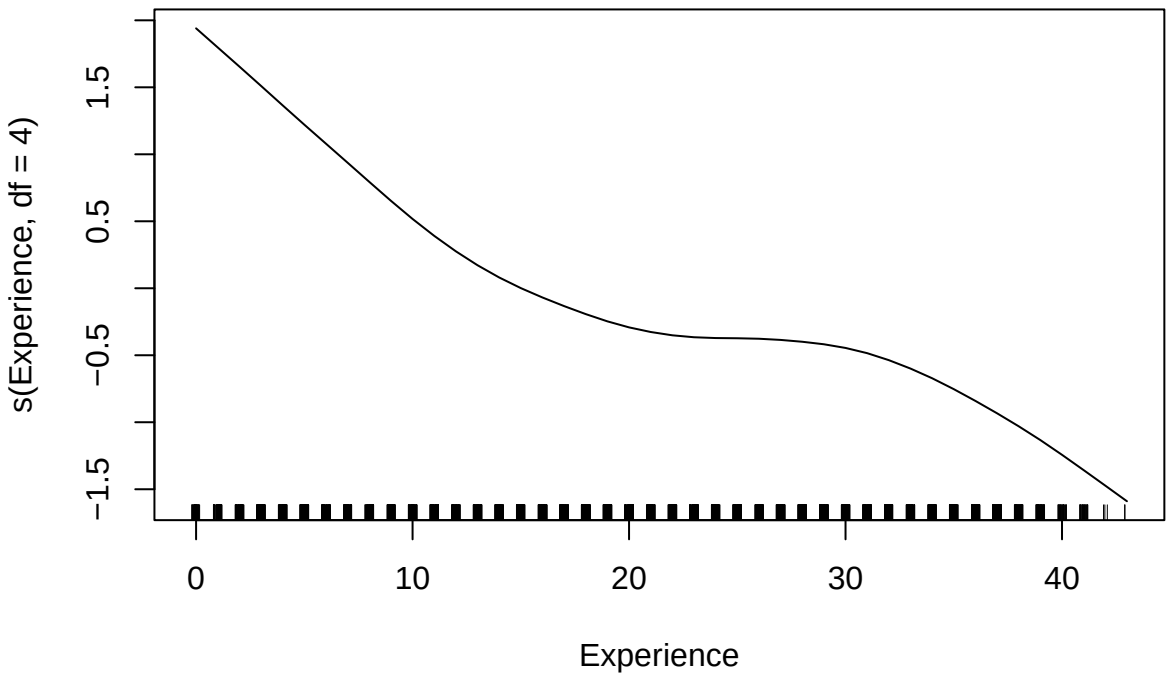
```
## Loaded gam 1.22-5
```

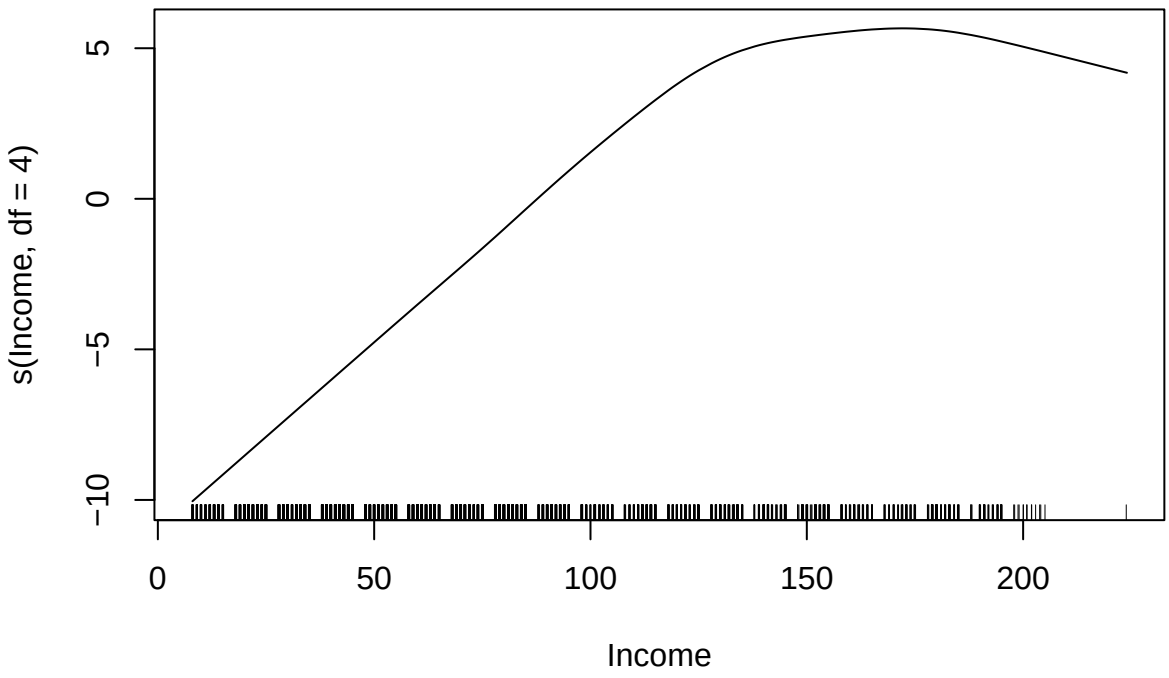
```
#making a gam model with all splines set to the default of df 4 for all numerical predictors
gam1 <- gam(Subscriber ~ s(Age, df=4) + s(Experience, df=4) + s(Income, df=4) +
              s(Family, df=4) + s(AvgAmt, df=4) + Education,
            data = train_data, #setting the train data
            family = 'binomial') #and making the family binomial
```

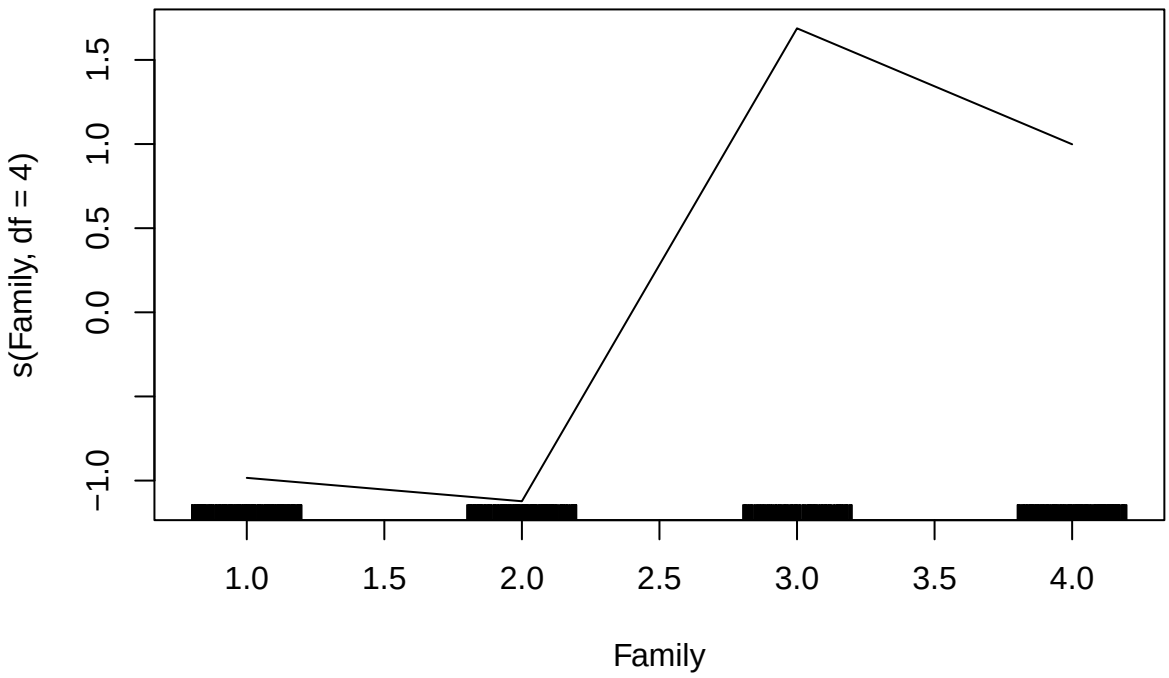
Q3(b) [2 points] Plot the estimated splines by `gam1`. For the predictor `Income`, is the relationship monotonic?

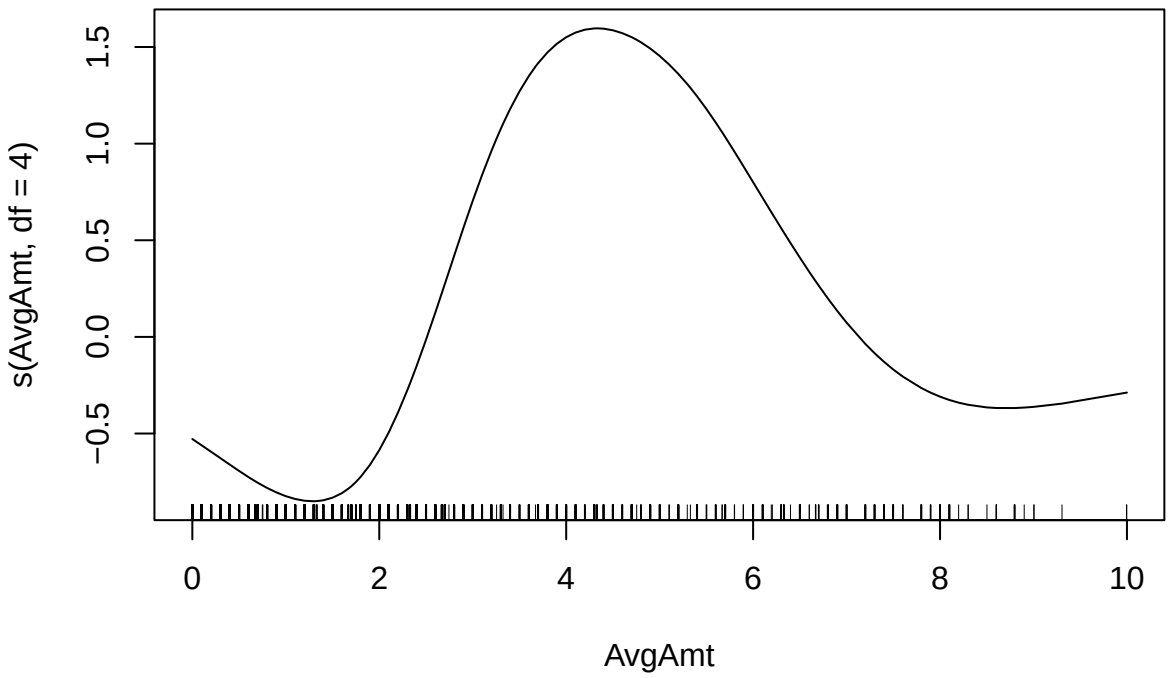
```
#plotting the charts
plot(gam1)
```

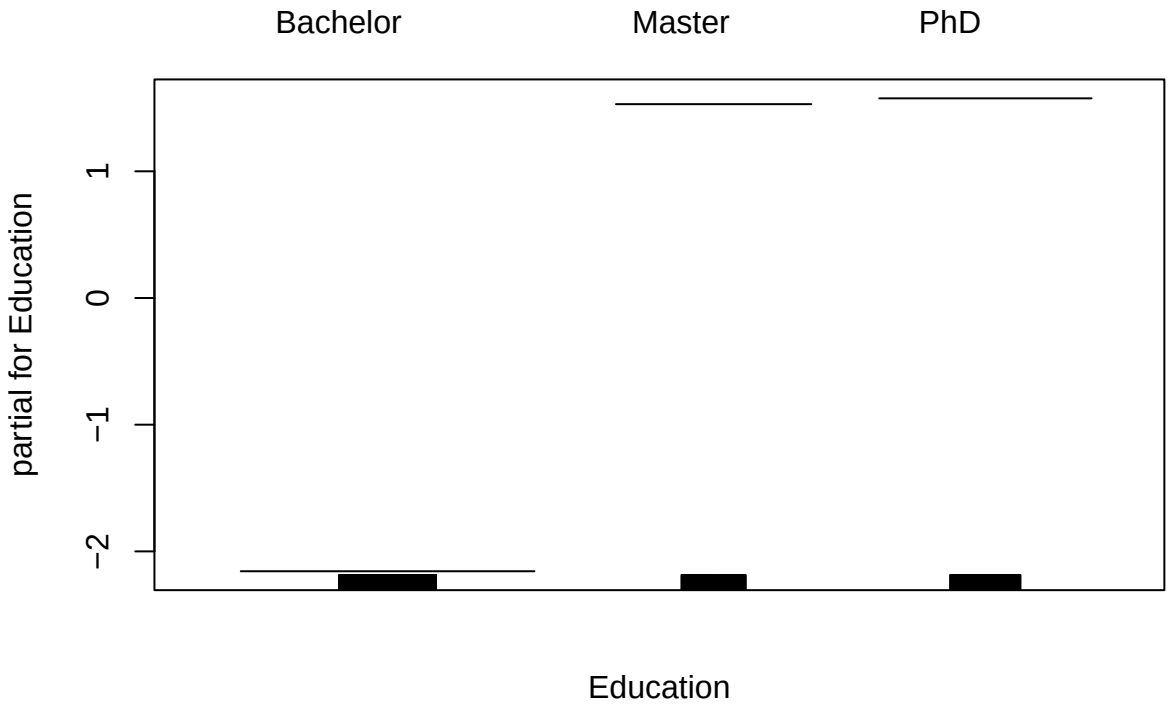

s(Age, df = 4)
−2.0
−1.0
0.0
0.5
1.0
30
40
50
60
Age


s(Experience, df = 4)
−1.5
−0.5
0.5
1.5
0
10
20
30
40
Experience


s(Income, df = 4)
5
0
−5
−10
0
50
100
150
200
Income


s(Family, df = 4)
−1.0
0.0
0.5
1.0
1.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Family

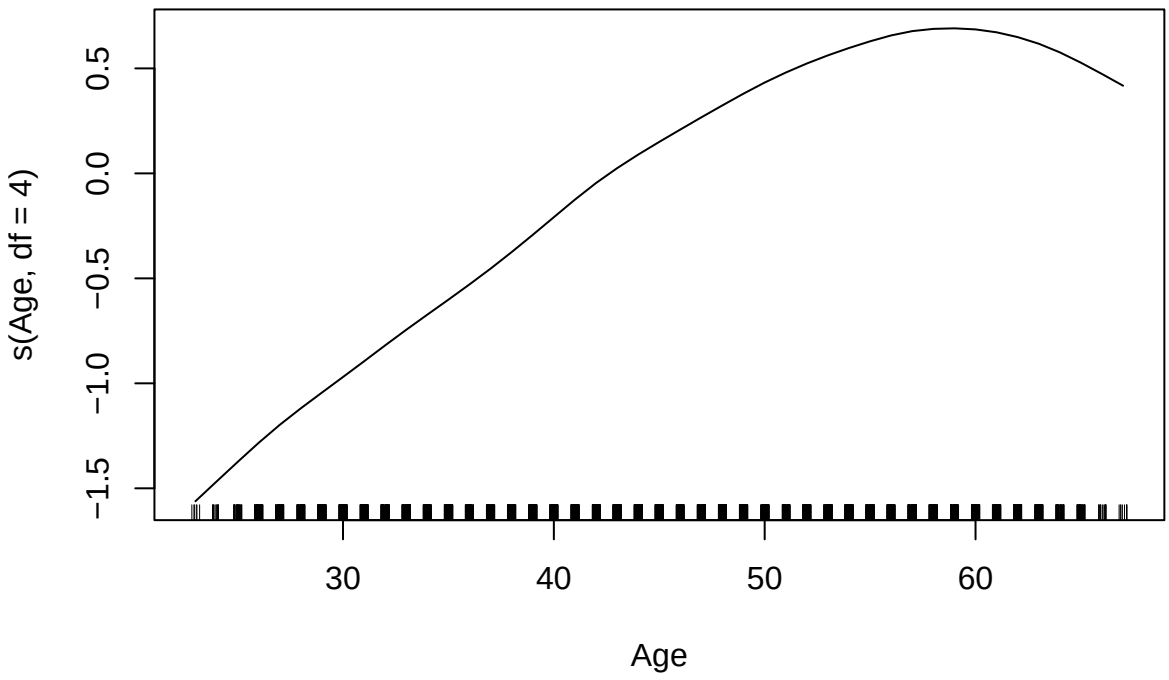

s(AvgAmt, df = 4)
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10
AvgAmt

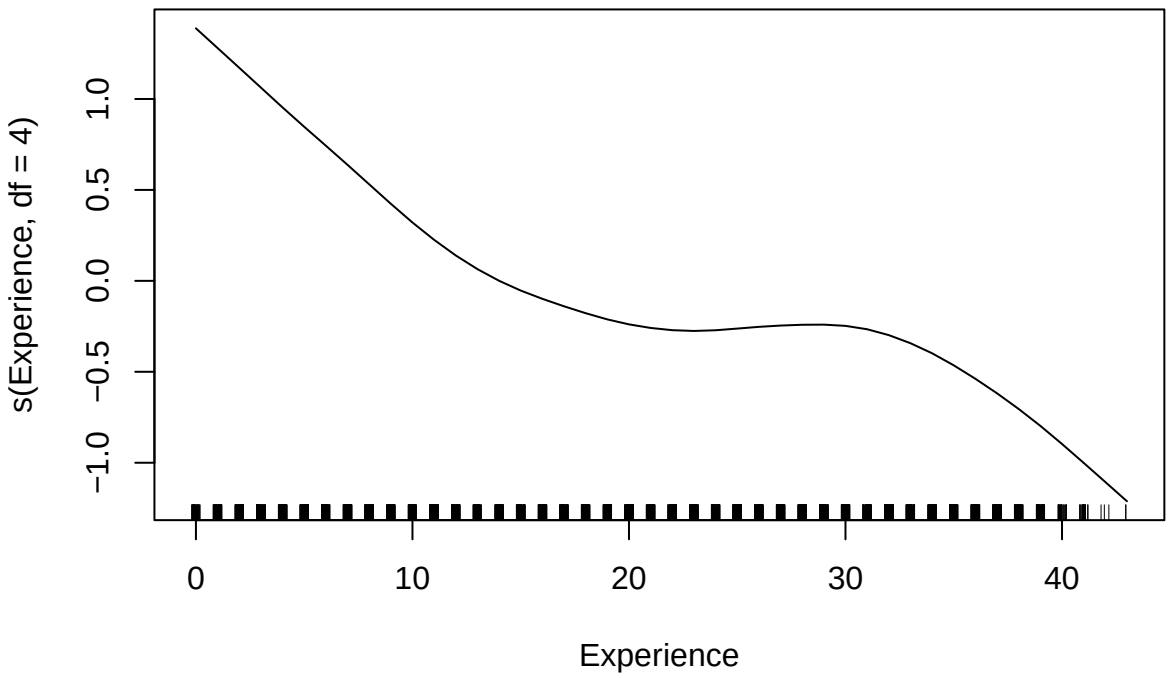


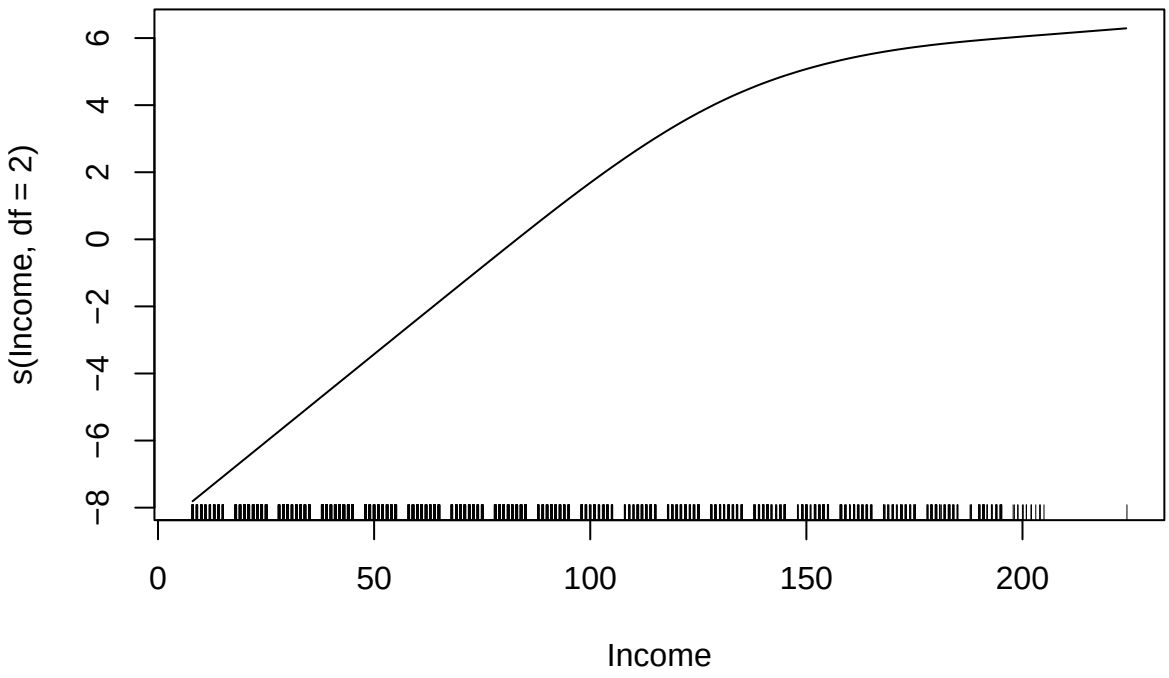
Answer: No, the relationship for Income is not monotonic because the Income predictor begins by increasing, then remains flat briefly, and then decreases. For a variable to be monotonic, it would have to either never increase or never decrease.

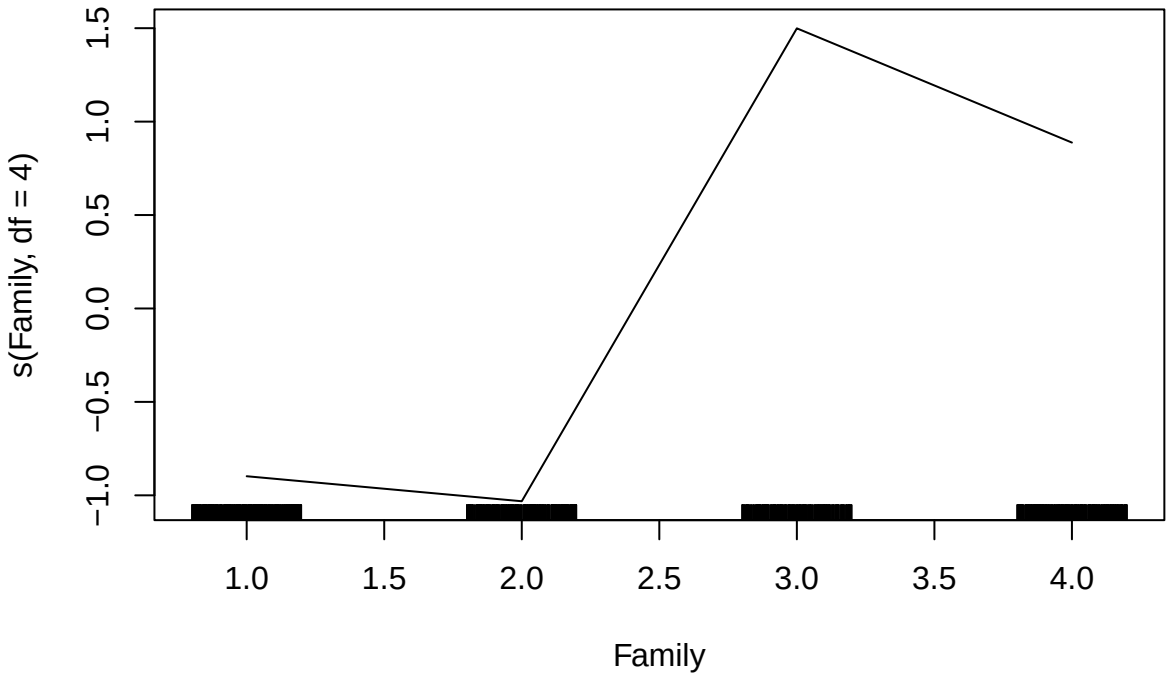
Q3(c) [2 points] Keeping everything else the same, further fit a GAM model but using splines with **df=2** for `Income`, name it `gam2`. Do we have a monotonic relationship for predictor `Income` now?

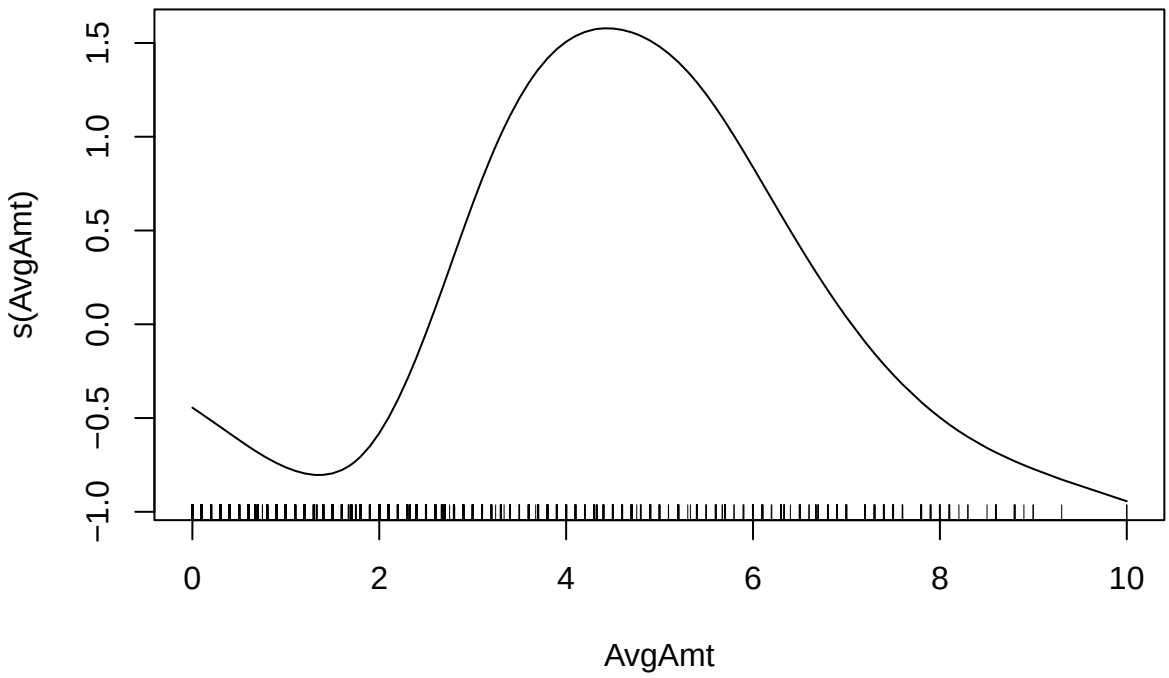
```
#using the same gam model as before but altering the df to 2 for income to see how it impacts the incom
gam2 <- gam(Subscriber ~ s(Age, df=4) + s(Experience, df=4) + s(Income, df=2) +
              s(Family, df=4) + s(AvgAmt) + Education,
            data = train_data,
            family = 'binomial')

plot(gam2)
```

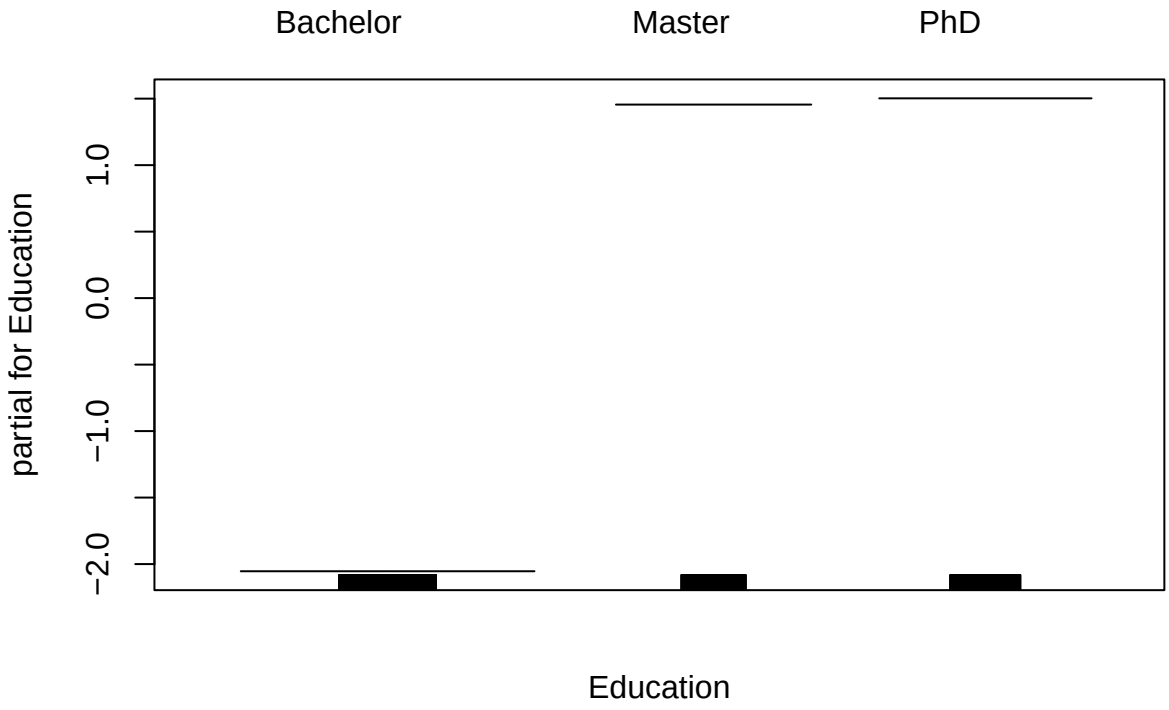

s(Age, df = 4)
−1.5
−1.0
−0.5
0.0
0.5
30
40
50
60
Age


s(Experience, df = 4)
−1.0
−0.5
0.0
0.5
1.0
0
10
20
30
40
Experience


s(Income, df = 2)
−8
−6
−4
−2
0
2
4
6
0
50
100
150
200
Income


s(Family, df = 4)
−1.0
−0.5
0.0
0.5
1.0
1.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Family


s(AvgAmt)
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10
AvgAmt



Answer: Yes, with a spline set to df = 2 for income and income alone, income is now monotonic. It begins with a steady increase from 0 to 150 and then slows, but continues to increase for theh duration of the plot.

Q4 [12 points] Neural Networks

Fit an NN model of the dependent variable `Subscriber` w.r.t. all 6 predictors `Age`, `Experience`, `Income`, `Family`, `AvgAmt`, `Education` using the training data, name it `nn1`. For the architecture of NN, let's use one hidden layer with **6** hidden units.

Q4(a) [2 points] Generate the training dataset that are needed for the estimation of NN using the function `model.matrix()` and store it in `x_train_nn`. In addition, use the `scale()` function to standardize the predictors by centering with mean and scaling with sd.

```
#passing data as a model matrix while dropping the response column
x_train_nn <- model.matrix( ~ ., data = train_data, na.rm = TRUE)[ , -9]

#getting the mean for each column
x_mean <- apply(x_train_nn, MARGIN = 2, FUN = mean)
#getting the sd for each column
x_sd <- apply(x_train_nn, MARGIN = 2, FUN = sd)
#scaling the train data
x_train_nn <- scale(x_train_nn, center = x_mean, scale = x_sd)
```

Q4(b) [2 points] Further combine the dependent variable `Subscriber` with the standardized predictors `x_train_nn` generated in Q4(a). Don't forget to rename the first column of the data frame as `Subscriber`.

```r
#dropping the intercept column
x_train_nn <- x_train_nn[ , -1]

#adding in the dependent
x_train_nn <- cbind.data.frame(train_data$Subscriber, x_train_nn)

#renaming the dependent
colnames(x_train_nn)[1] <- 'Subscriber'
```

Q4(c) [2 points] Generate the test dataset that are needed for the out-of-sample prediction evaluation of NN using the function `model.matrix` and store it in `x_test_nn`. Use the `scale()` function to standardize the predictors by centering with mean and scaling with sd as in Q4(a).

```r
#passing the test data to a matrix
x_test_nn <- model.matrix( ~ ., data = test_data, na.rm = TRUE)[ , -9]

#scaling the test data using the train mean and sd
x_test_nn <- scale(x_test_nn, center = x_mean, scale = x_sd)

#dropping the intercept
x_test_nn <- x_test_nn[ , -1]

#adding the dependent
x_test_nn <- cbind.data.frame(test_data$Subscriber, x_test_nn)

#renaming the dependent
colnames(x_test_nn)[1] <- 'Subscriber'
```

Q4(d) [6 points] Fit an NN that has one hidden layers with **6** hidden units and name it `nn1`. Make sure to use random seed **`set.seed(7)`**!

```r
#importing the neural net library
library('neuralnet')
#setting the random seed
set.seed(7)

#making the neural net model
nn1 <- neuralnet(Subscriber == 'Yes' ~ ., #all variables to predict yes
                 hidden = 6, #6 hidden units in 1 layer
                  data = x_train_nn,#adding on the train data
                   linear.output = FALSE) #not linear!


plot(nn1, type = "best") #plotting the nn
```

Q5 [16 points] Model Evaluation (Prediction)

Q5(a) [2 points] Use `lm_full`, `gam1` and `nn1` to generate probability predictions on the test data and store the prediction in `lm_full_pred`, `gam1_pred` and `nn1_pred` respectively.

```
#making predictions
lm_full_pred <- predict(lm_full, newdata = test_data, type = 'response')
gam1_pred <- predict(gam1, newdata = test_data, type = 'response')
nn1_pred <- predict(nn1, newdata = x_test_nn, type = 'response')
```

Q5(b) [**2 points**] What are the predicted conversion probability of the **fifth** customer in the test data by `lm_full`, `gam1` and `nn1`, respectively?

```
#pulling the 5th value (5th customer) from each of the prediction sets
lm_full_pred[5]
```

```
##         7
## 0.0326466
```

```
gam1_pred[5]
```

```
##           7
## 0.001575904
```

```
nn1_pred[5]
```

```
## [1] 1.354885e-61
```

Answer:

lm_full: 0.0326 gam1: 0.00157 nn1: 1.354885e-61

Q5(c) [**2 points**] Use the `R` package `caret` to evaluate the prediction performance of `lm_full`, `gam1` and `nn1` on the test data. What are the sensitivity of `lm_full`, `gam1` and `nn1`?

```
#importing caret to make confusion matrices
library('caret')
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
#making confusion matrices for each model, but doing so by using the predictions scaled to yes and no v
lm1_acc <- confusionMatrix(factor(ifelse(lm_full_pred > 0.5, 'Yes', 'No')), test_data$Subscriber, positi
gam1_acc <- confusionMatrix(factor(ifelse(gam1_pred > 0.5, 'Yes', 'No')), test_data$Subscriber, positive
nn1_acc <- confusionMatrix(factor(ifelse(nn1_pred > 0.5, 'Yes', 'No')), x_test_nn$Subscriber, positive =
```

Answer:

lm_full Accuracy: 95.62% gam1 Accuracy: 97.45% nn1 Accuracy: 97.76%

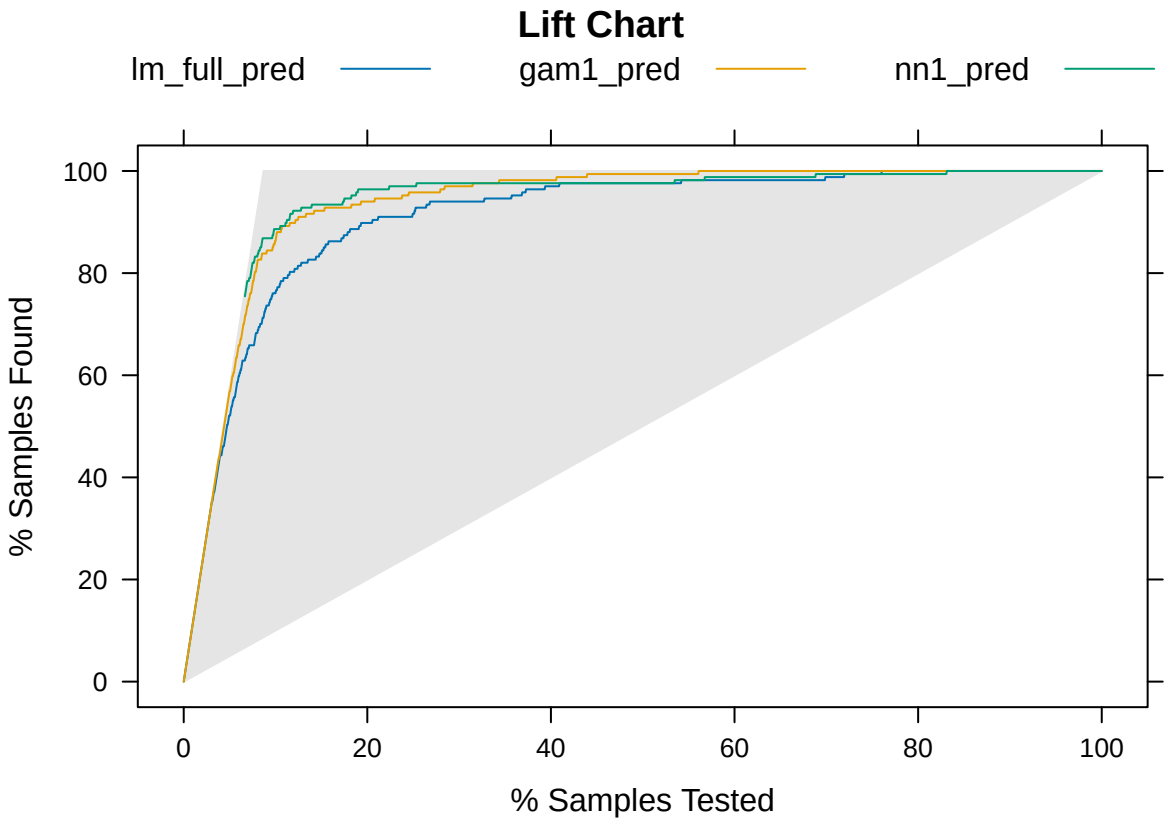
lm_full Sensitivity: 0.63473 gam1 Sensitivity: 0.79641 nn1 Sensitivity: 0.84431

Q5(d) [**4 points**] Based on the result in Q5(c), fill out the confusion matrix for `gam` and `nn1` on the written exam.

See written exam

Q5(e) [**2 points**] Further generate a lift chart to compare the prediction performance of `lm_full`, `gam1` and `nn1`. To make the lift chart look nice, do **NOT** use the `cuts` argument.

```r
#making a lift chart of all the models
lift_chart <- lift(test_data$Subscriber ~ lm_full_pred + gam1_pred + nn1_pred,
                   class = 'Yes')

#graphing the lift chart
xyplot(lift_chart, auto.key=list(columns=3), main='Lift Chart')
```



Q5(f) [2 points] Which model should we prefer if we only has the marketing budget to reach out to 20% of the customers?

Answer: Based on the lift chart we should target the top 20% of the customers using the neural network model.

Q5(g) [2 points] Take `lm_full` for example. If we want to capture more than 80% of all potential customers (i.e. customers who will become subscriber), what should be our minimum marketing budget? In other words, what is the percentage of customers we should reach out to? (The answer does not need to be exact, it only needs to be approximately correct.)

Answer: Based on an examination of the lift chart, it appears that we will need to reach out to 10-15% of the total customer base to capture 80% of all potential customers. A rough guess would be 13.5%