In [ ]:

In [ ]:

In [ ]:

In [1]:
```
library(ggplot2)
library(mgcv)
```

```
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Loading required package: nlme
This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.
```

C3M4 Peer Reviewed Assignment

Outline:

The objectives for this assignment:

1. Observe the difference between GAMs and other regression models on simulated data.
2. Review how to plot and interpret the coefficient linearity for GAM models.

General tips:

1. Read the questions carefully to understand what is being asked.
2. This work will be reviewed by another human, so make sure that you are clear and concise in what your explanations and answers.

Problem 1: GAMs with Simulated Data

In this example, we show how to check the validity of a generalized additive model (GAM) (using the `gam()` function) using simulated data. This allows us to try and understand the intricacies of `gam()` without having to worry about the context of the data.

1. (a) Simulate the Data

Let n = 200. First, construct three predictor variables. The goal here is to construct a GAM with different types of predictor terms (e.g., factors, continuous variables, some that will enter linearly/parametrically, some that enter transformed).

1. x1: A continuous predictor that, we will suppose has a nonlinear relationship with the response.
2. x2: A categorical variable with three levels: `s`, `m`, and `t`.
3. x3: A categorical variable with two levels: `TRUE` and `FALSE`.

Then, make the response some nonlinear/nonparametric function of x. For our case, use: $\log(\mu_i) = \beta_1 + \sin(0.5x_{2i,1}) - x_{i,2} + x_{i,3}$ This model is a Poisson GAM. In a realworld situation, we wouldn't know this functional relationship and would estimate it. Other terms are modeled parametrically. The response has normal noise.

Note that:

1. The construction of $\mu = (\mu_1, \ldots, \mu_n)^T$ has the linear predictor exponentiated, because of the nature of the link function.
2. We use \boldsymbol\mu to construct \mathbf{y} = (y_1,...,y_n)^T . The assumption for Poisson regression is that the random variable Y_i that generates y_i is Poisson with mean \mu_i.
3. `as.integer(as.factor(VARIABLE))` converts the labels of VARIABLE to 1, 2, 3,.. so that we can construct the relationship for these factors.

Plot the relationship of \mathbf{y} to each of the predictors. **Then, split the data into a training (`train_sim`) and test (`test_sim`) set.**

In [2]:
```
set.seed(0112358)
# n = number of data points
n = 200

# Create predictors based on specifications
d = data.frame(
   x1=rnorm(n, mean = 45, sd = 15),
   x2=as.factor(sample(c('s','m','t'),size=n,replace=TRUE)),
   x3=sample(c(F,T),size=n,replace=TRUE),
 stringsAsFactors=F)
# Construct Response from predictors
d$mu = with(d, exp(log(0.5*x1^2) - as.integer(as.factor(x2))
                   + as.integer(as.factor(x3))))

d$y = rpois(n, d$mu);

# Plot relationship of y to each of the predictors
ggplot(d, aes(x1,y)) +
    geom_point(alpha = 0.5) +
    theme_bw()

head(d) #simulated mean values
summary(d)

# Traing and test set
set.seed(1771) #set the random number generator seed.
n = floor(0.8 * nrow(d)) #find the number corresponding to 80% of the data
index = sample(seq_len(nrow(d)), size = n) #randomly sample indicies to be
 included in the training set

train_sim = d[index, ] #set the training set to be the randomly sampled ro
ws of the dataframe
```

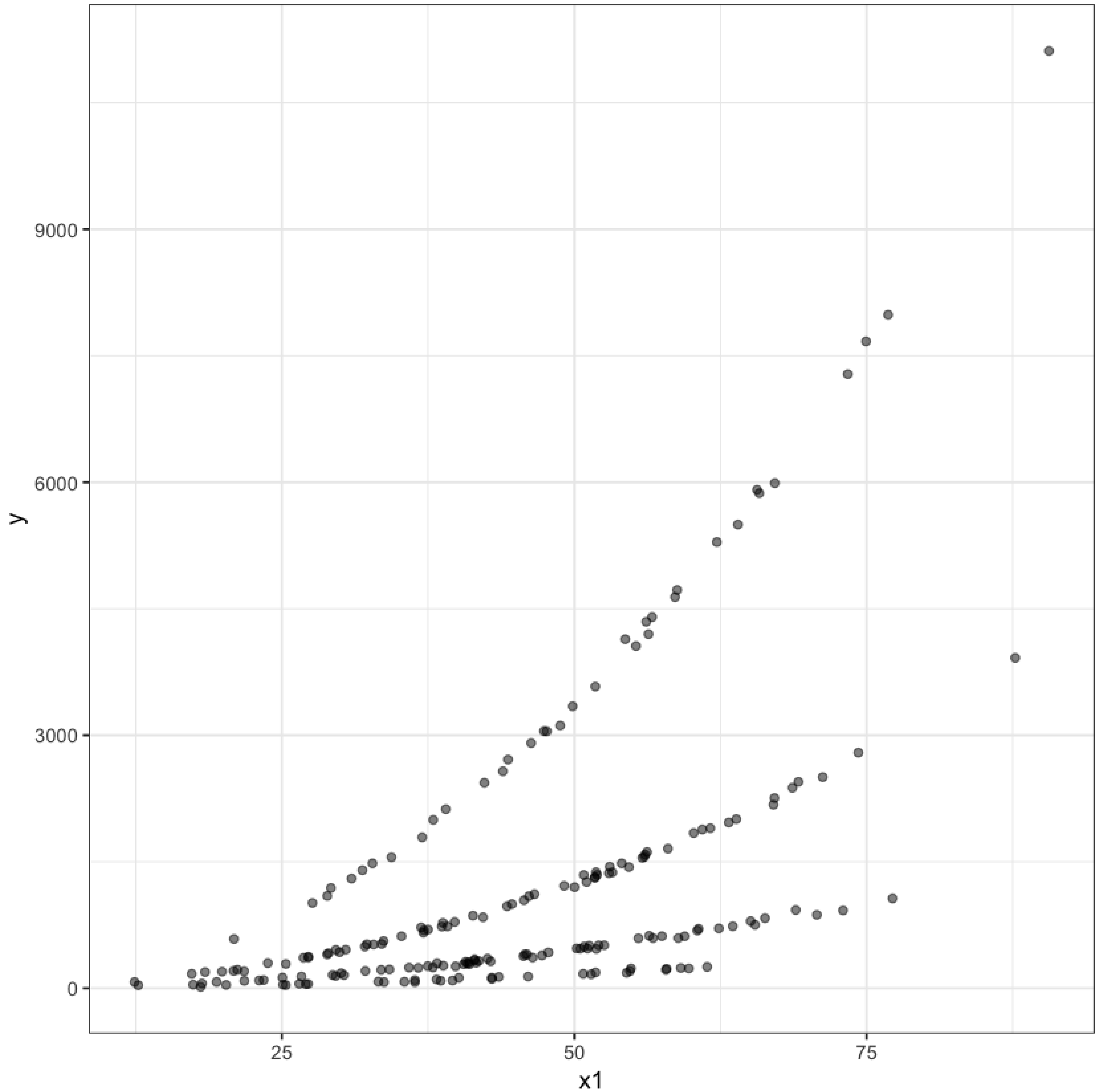
```r
test_sim = d[-index, ] #set the testing set to be the remaining rows
dim(test_sim) #check the dimensions
dim(train_sim) #check the dimensions
```

x1	x2	x3	mu	y
37.93226	t	TRUE	264.66281	248
62.19000	m	TRUE	5256.60758	5291
52.95074	s	TRUE	1401.89021	1365
39.17245	s	TRUE	767.24046	734
63.85257	m	FALSE	2038.57529	2006
17.41148	t	TRUE	55.76312	43

x1	x2	x3	mu	y
Min. :12.40	m:64	Mode :logical	Min. : 22.06	Min. : 17.0
1st Qu.:33.54	s:76	FALSE:95	1st Qu.: 233.99	1st Qu.: 234.0
Median :44.31	t:60	TRUE :105	Median : 564.47	Median : 542.5
Mean :44.98			Mean : 1205.04	Mean : 1200.8
3rd Qu.:56.01			3rd Qu.: 1408.78	3rd Qu.: 1409.2
Max. :90.61			Max. :11159.30	Max. :11114.0

40 5

160 5



1. (b) Other Regression Models

Before jumping straight into GAMs, let's test if other regression models work. What about a regular linear regression model with ordinary least squares, and a generalized linear model for Poisson regression?

First fit a linear regression model to your `train_sim` data. We know that all of the predictors were used to make the response, but are they all significant in the linear regression model? Explain why this may be.

Then fit a Generalize Linear Model (GLM) to the `train_sim` data. Plot three diagnostic plots for your GLM:

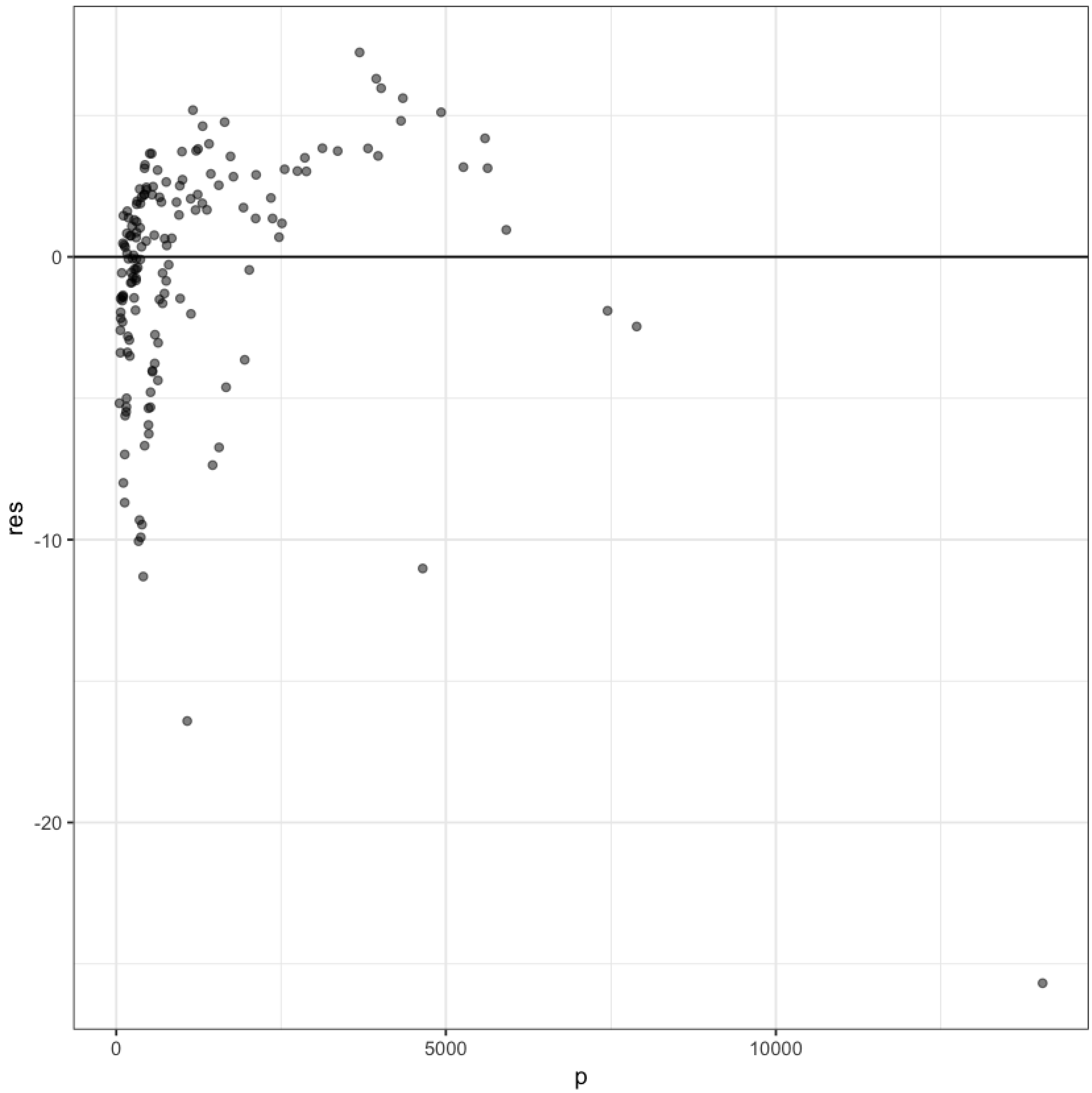
1. Residual vs. log(Fitted Values)
2. QQPlot of the Residuals
3. Actual Values vs. Fitted Values

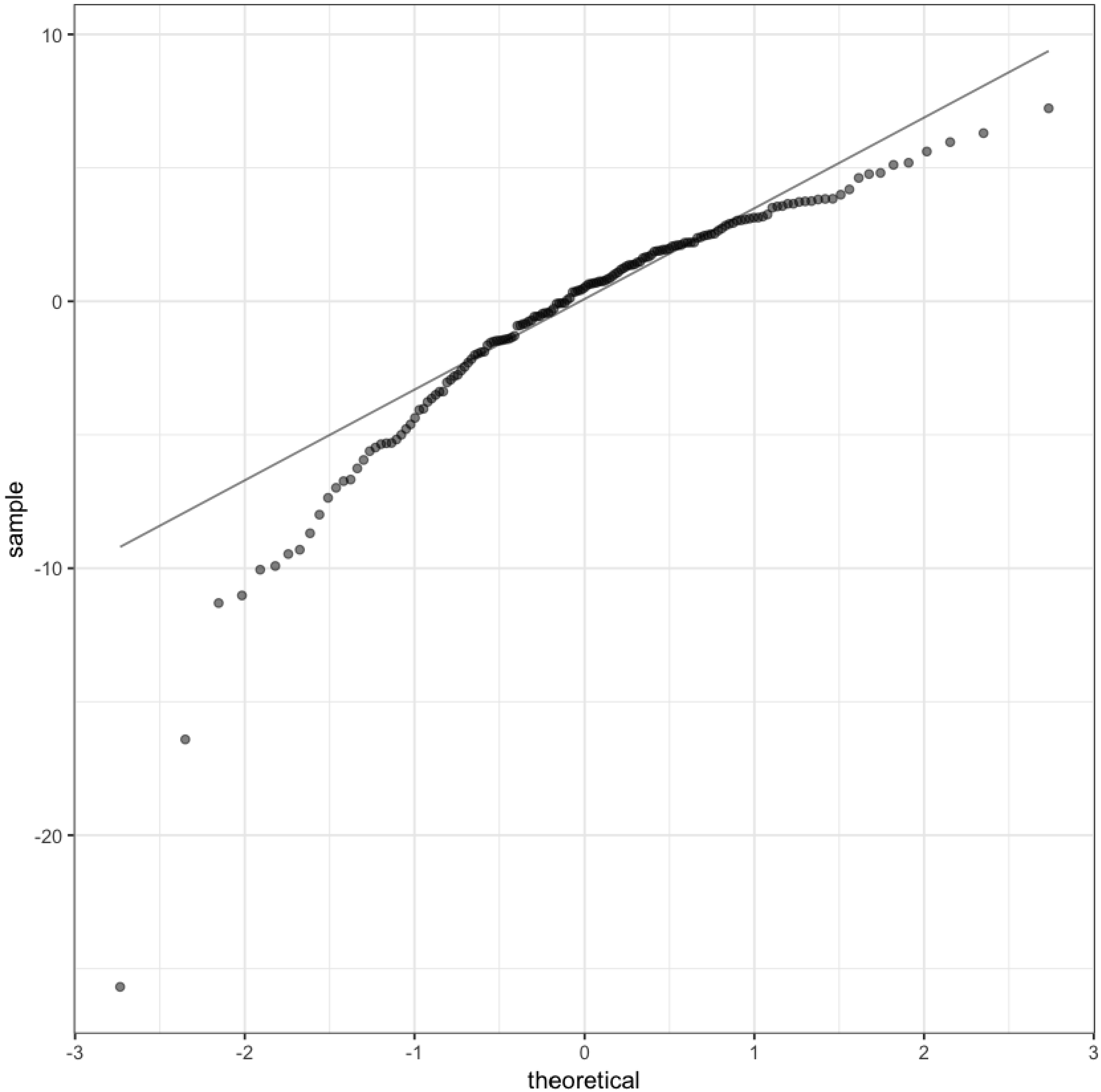
Using these plots, determine whether this model is a good fit for the data. Make sure to explain your conclusions and reasoning.

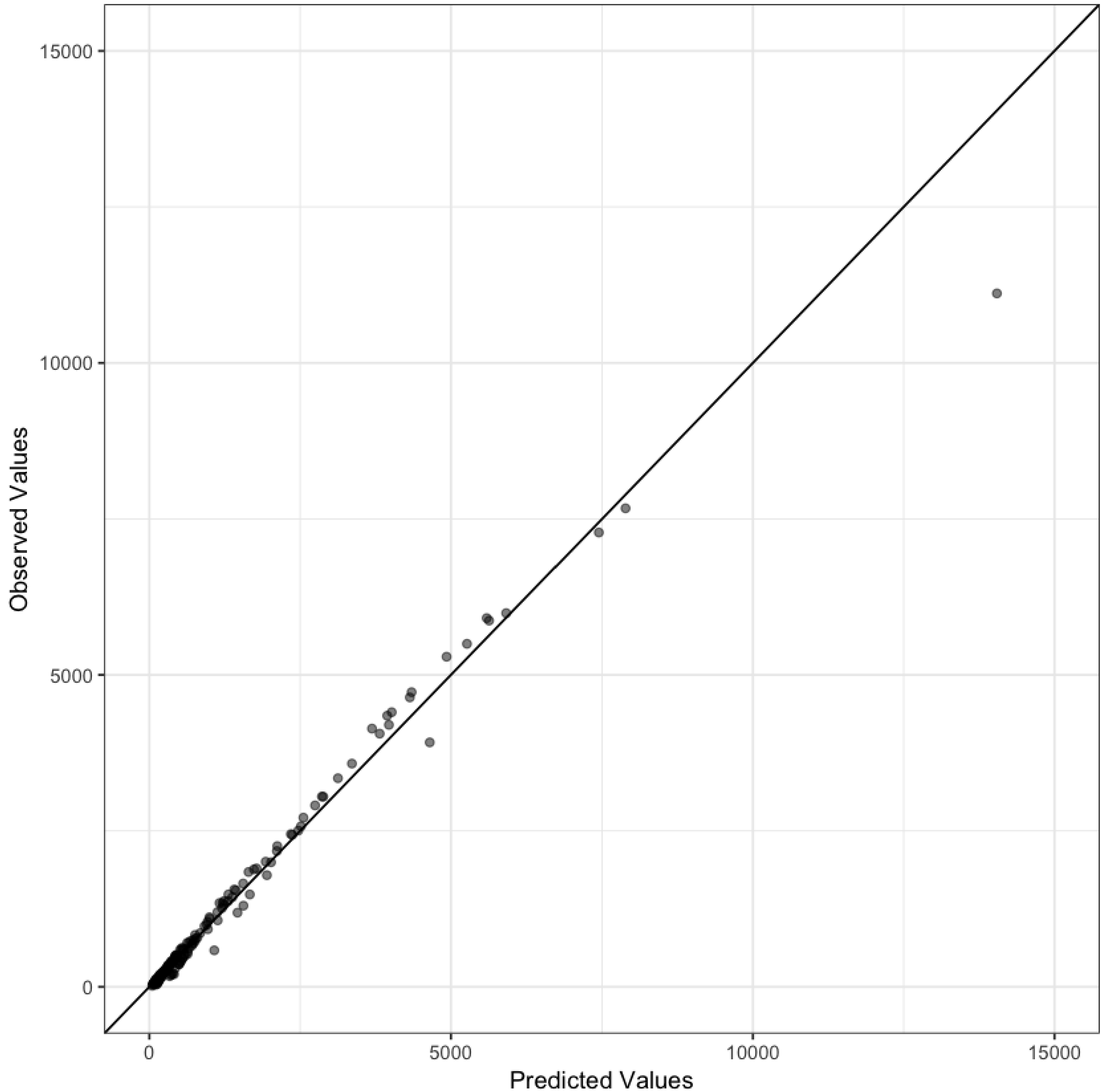
In [3]:
```
# Fit a LM model to the data
lm_sim = lm(y ~ x1 + x2 + x3, data = train_sim)
summary(lm_sim)

# Fit a GLM model to the data
glm_sim = glm(y ~ x1 + x2 + x3, train_sim, family = poisson)


#residual plot
res = residuals(glm_sim, type="deviance") #compute the deviance residuals
p = predict(glm_sim, type = "response")

# Create the three specified plots

d_glm = data.frame(p, res,y = train_sim$y)
#residual vs fitted plot
ggplot(d_glm,aes(p, res)) +
    geom_point(alpha = 0.5) +
    geom_hline(yintercept = 0) +
    theme_bw()

## qqplot
ggplot(d_glm,aes(sample = res)) +
    stat_qq( alpha = 0.5) + stat_qq_line( alpha = 0.5) +
    theme_bw()

#fitted vs actual
ggplot(d_glm,aes(p,y)) +
    geom_point(alpha = 0.5) +
    geom_abline(slope=1) +
    xlim(c(0,15000)) +
    ylim(c(0,15000)) +
    xlab("Predicted Values") +
    ylab("Observed Values") +
    theme_bw()
```

```
Call:
lm(formula = y ~ x1 + x2 + x3, data = train_sim)

Residuals:
    Min      1Q  Median      3Q     Max 
-1182.1  -540.9  -185.9   422.7  5582.8 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -621.355    290.317  -2.140   0.0339 *  
x1             54.909      4.898  11.211  < 2e-16 ***
x2s         -1649.806    175.579  -9.396  < 2e-16 ***
x2t         -1997.408    182.848 -10.924  < 2e-16 ***
x3TRUE       1177.096    141.558   8.315 4.37e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 892.5 on 155 degrees of freedom
Multiple R-squared:   0.74,     Adjusted R-squared:  0.7333
F-statistic: 110.3 on 4 and 155 DF,  p-value: < 2.2e-16
```




10
0
-10
-20
sample
-3
-2
-1
0
1
2
3
theoretical



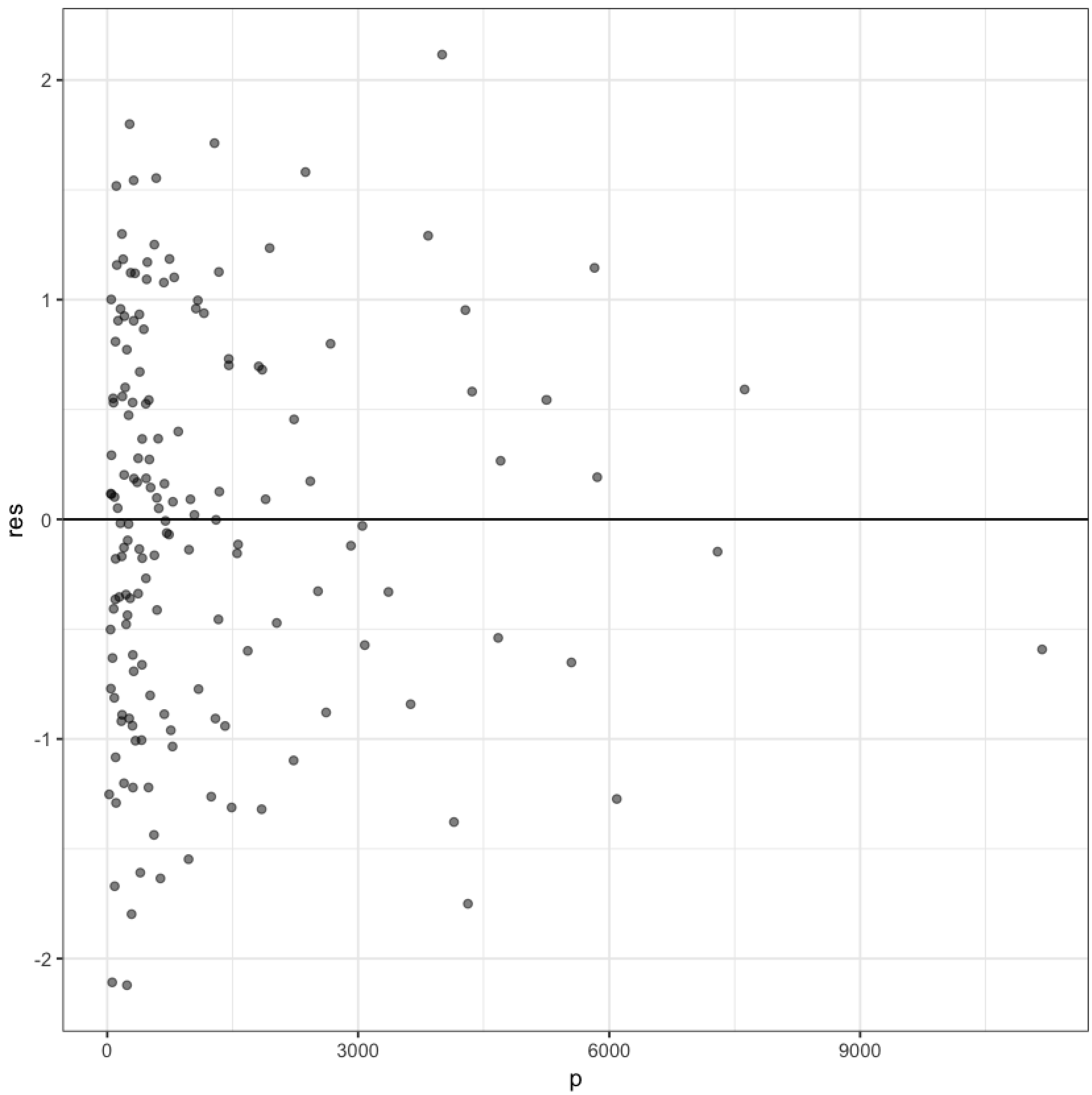
- Linear model: Yes, the predictors are statistically significant, but they enter the model in an incorrect way. This is one reason why hypothesis tests can be deceiving!
- GLM: Notice above that the residual plot isn't that bad, but there's some skew. The QQ-plot looks poor, especially in the tails of the distribution. The observed vs predicted values shows some curvature around the line $y = x$ suggesting that predictions are systematically off.

1. (c) Looking for those GAMs

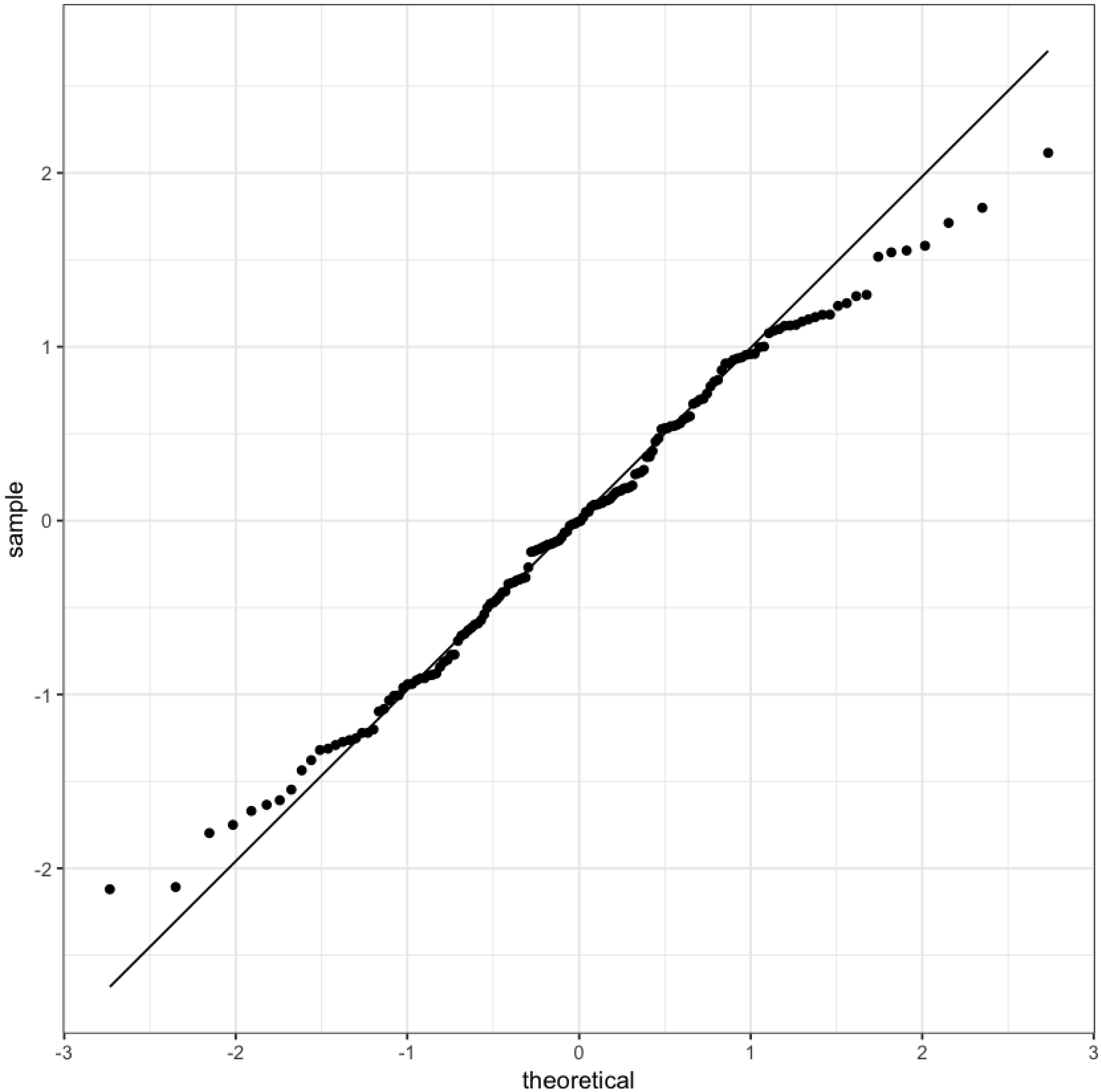
Now, it's time to see how a generalized additive model (GAM) performs! Fit a GAM to the data. Construct the same three plots for your GAM model. Do these plots look better than those of the GLM?

In [4]:
```
# Fit a GAM model to the data
gam_sim = gam(y ~ s(x1) + x2 + x3, data= train_sim, family = poisson)
```

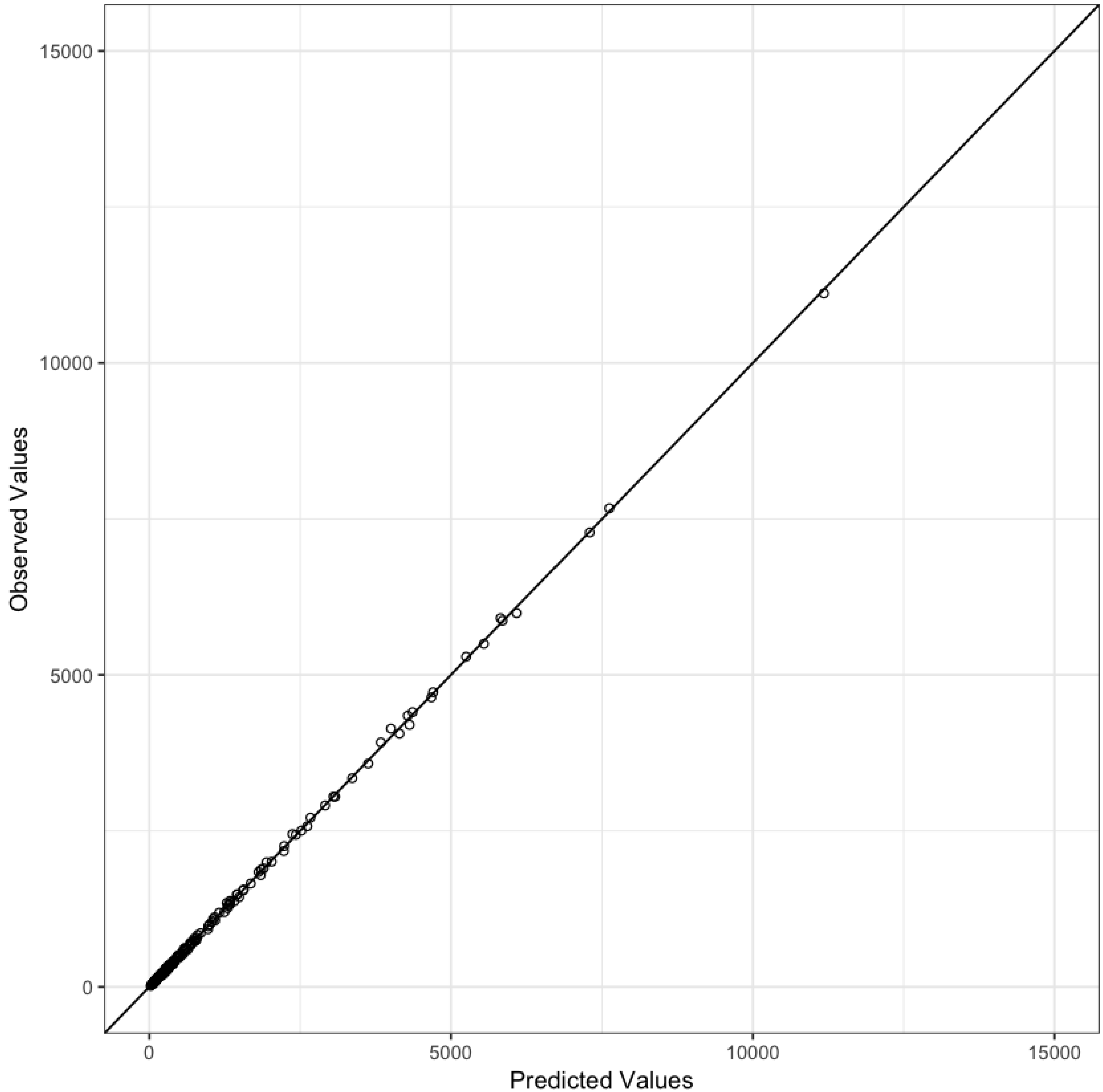
```r
# Construct the three specified plots
res = residuals(gam_sim, type="pearson") #compute the deviance residuals
p = predict(gam_sim, type = "response");
d_gam = data.frame(p, res, y = train_sim$y)


#residual vs fitted
ggplot(train_sim,aes(p, res)) +
    geom_point(alpha = 0.5) +
    geom_hline(yintercept = 0) +
    theme_bw()


## qqplot
ggplot(train_sim,aes(sample = res)) +
    stat_qq() + stat_qq_line() +
    theme_bw()


#fitted vs actual
ggplot(train_sim,aes(p,y)) +
    geom_point(shape=1) +
    geom_abline(slope=1) +
    xlim(c(0,15000)) +
    ylim(c(0,15000)) +
    xlab("Predicted Values") +
    ylab("Observed Values") +
    theme_bw()
```



2
1
0
-1
-2
res
0
3000
6000
9000
p



sample
2
1
0
-1
-2
-3
-2
-1
0
1
2
3
theoretical



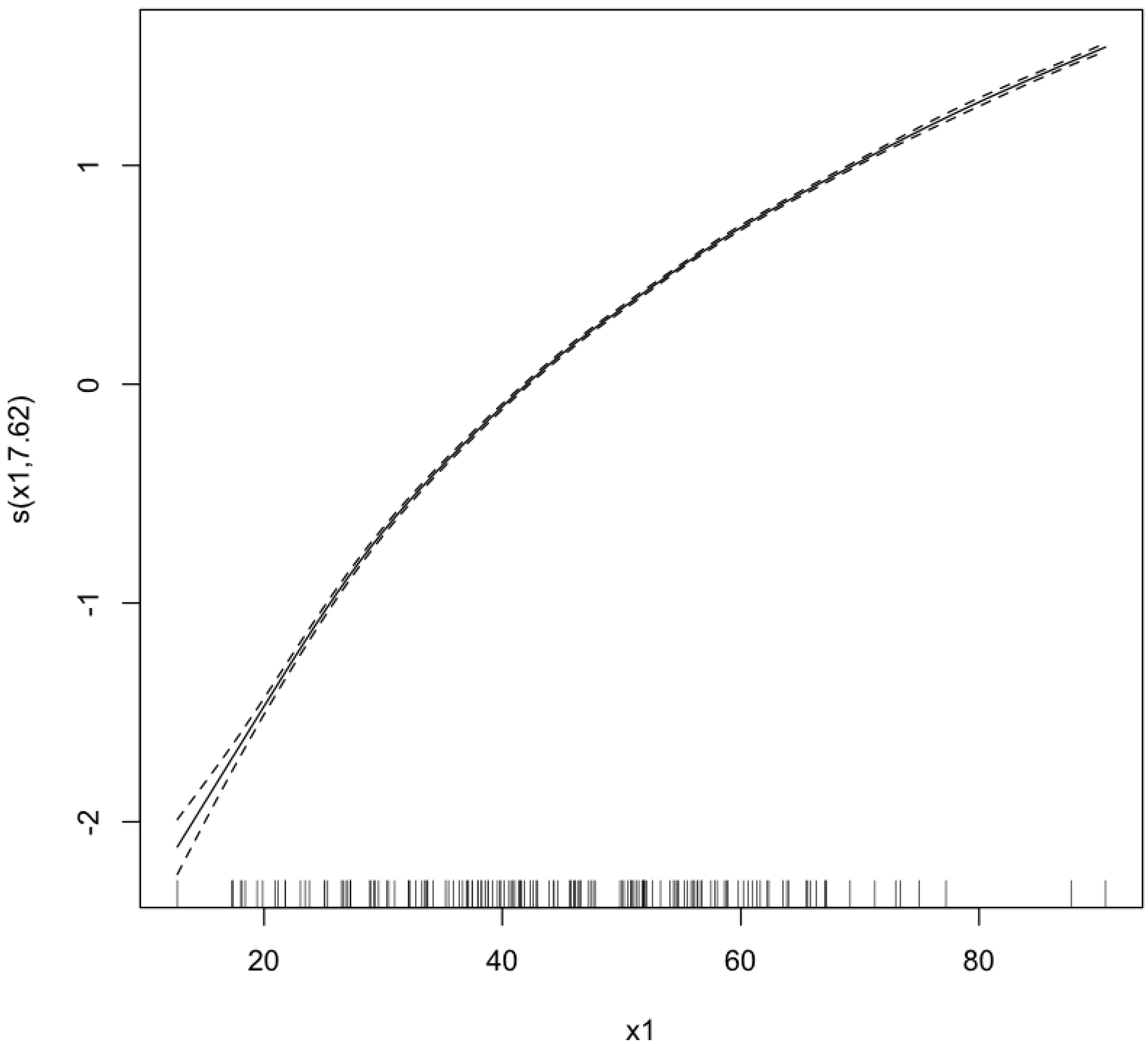
These plots look much better. The deviance residual vs fitted plot looks as we might expect: random scatter around zero (though there are more points at lower fitted values than larger ones). The observed vs predcited value plot looks great; values cluster very tightly on the line $y = x$. The QQ plot looks, still with some deviations from normality in the tails of the distribution. The fit appears to be much better than the above models.

1. (d) Interpreting GAMs

We made a GAM model! However GAMs are harder to interpret than regular linear regression models. How do we determine if a GAM model was necesary? Or, in other words, how do we determine if our predictors have a linear relationship with the response?

Use the `plot.gam()` function in the mgcv library to plot the relationship between `y` and `x1`. Recall that x_1 entered our model as $\sin(0.5x_{i,1}^2)$, and we plotted that relationship in **1.(a)**. Does your plot confirm this relationship?

In [5]:
```
plot.gam(gam_sim)
```



Yes, the plot confirms this relationship. We see that the relationship between the response and `x1` is stronger for lower values of `x1` and becomes weaker as `x1` gets larger.

1.(e) Model comparison

Compute the mean squared prediction error (MSPE) for each of the three models above (regression model, GLM, and GAM). State which model performs based according to this metric.

Remember, the MSPE is given by
\begin{align*} MSPE = \frac{1}{k}\sum^k_{i=1}\left(y^\star_i - \widehat{y}^\star_i \right)^2 \end{align*}

where y^\star_i are the observed response values in the test set and \widehat{y}^\star_i are the predicted values for the test set (using the model fit on the training set).

In [6]:
```
#mspe for lm
lm_predict = predict(lm_sim, test_sim)
mspe_lm = mean((test_sim$y - lm_predict)^2);
cat("The MSPE for the additive model from part (a) is", mspe_lm, ".")


#mspe for glm
glm_predict = predict(glm_sim, test_sim, type = "response")

mspe_glm = mean((test_sim$y - glm_predict)^2);
cat("The MSPE for the GLM from part (b) is", mspe_glm, ".")

# mspe for gam
gam_predict = predict(gam_sim, test_sim, type = "response")
mspe_gam = mean((test_sim$y - gam_predict)^2);
cat("The MSPE for the GAM is", mspe_gam, ".")
```

```
The MSPE for the additive model from part (a) is 611955.1 .The MSPE for th
e GLM from part (b) is 22313.79 .The MSPE for the GAM is 1186.63 .
```

The MSPE is best for the GAM!

Problem 2 Additive models with the advertising data

The following dataset containts measurements related to the impact of three advertising medias on sales of a product, P. The variables are:

- `youtube` : the advertising budget allocated to YouTube. Measured in thousands of dollars;
- `facebook` : the advertising budget allocated to Facebook. Measured in thousands of dollars; and
- `newspaper` : the advertising budget allocated to a local newspaper. Measured in thousands of dollars.
- `sales` : the value in the i^{th} row of the sales column is a measurement of the sales (in thousands of units) for product P for company i.

The advertising data treat "a company selling product P" as the statistical unit, and "all companies selling product P" as the population. We assume that the $n = 200$ companies in the dataset were chosen at random from the population (a strong assumption!).

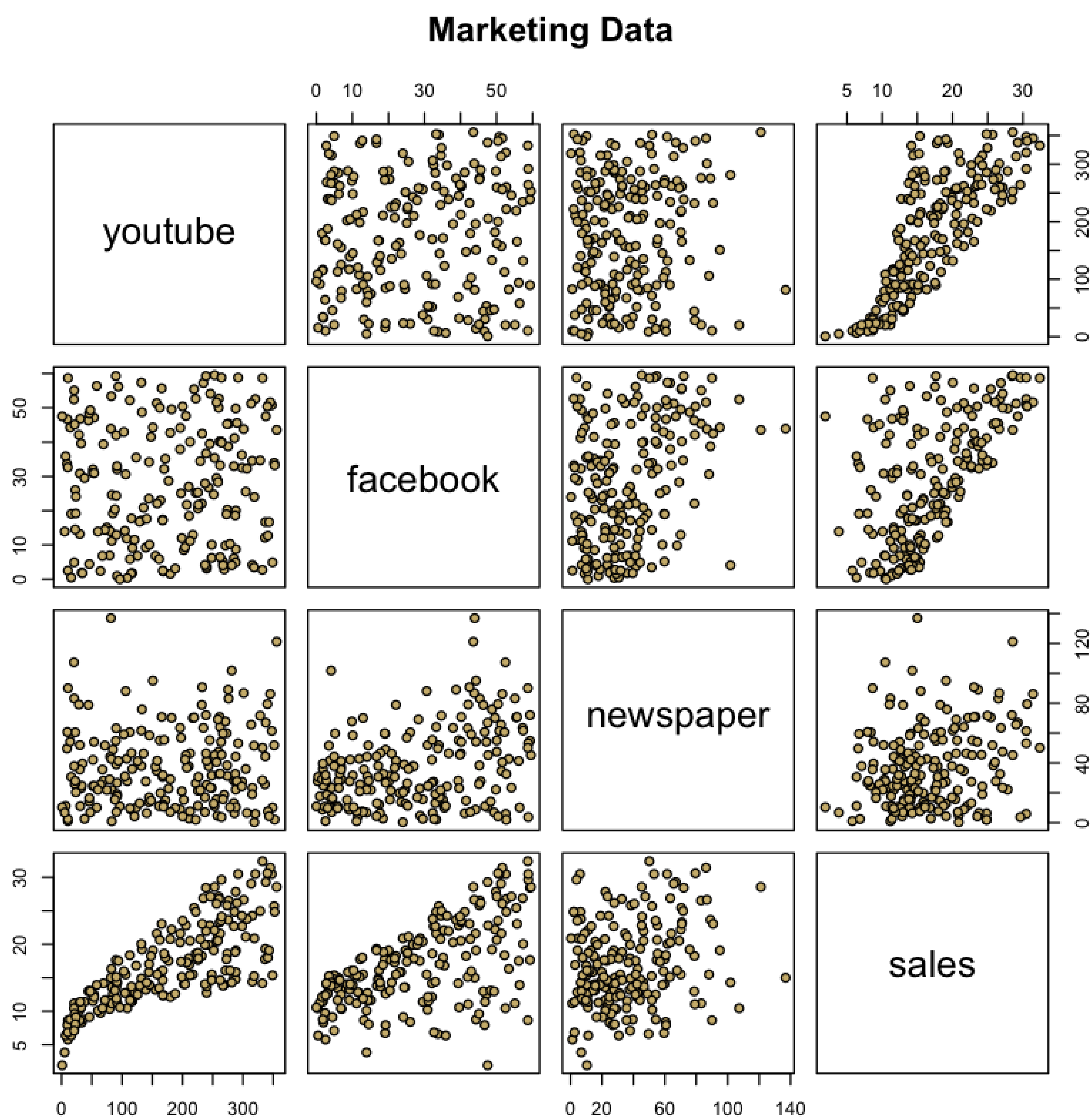
First, we load the data, plot it, and split it into a training set (`train_marketing`) and a test set (`test_marketing`).

In [7]:
```
library(RCurl) #a package that includes the function getURL(), which allow
s for reading data from github.
library(ggplot2)
library(mgcv)

url = getURL("https://raw.githubusercontent.com/bzaharatos/-Statistical-Mo
deling-for-Data-Science-Applications/master/Modern%20Regression%20Analysis
%20/Datasets/marketing.txt")
marketing = read.csv(text = url, sep = "")
```

```r
head(marketing)
pairs(marketing, main = "Marketing Data", pch = 21,
      bg = c("#CFB87C"))
```

```
Loading required package: bitops
```

youtube	facebook	newspaper	sales
276.12	45.36	83.04	26.52
53.40	47.16	54.12	12.48
20.64	55.08	83.16	11.16
181.80	49.56	70.20	22.20
216.96	12.96	70.08	15.48
10.44	58.68	90.00	8.64



In [8]:
```r
set.seed(177) #set the random number generator seed.
```

```
n = floor(0.8 * nrow(marketing)) #find the number corresponding to 80% of the data
index = sample(seq_len(nrow(marketing)), size = n) #randomly sample indicies to be included in the training set

train_marketing = marketing[index, ] #set the training set to be the randomly sampled rows of the dataframe
test_marketing = marketing[-index, ] #set the testing set to be the remaining rows
dim(test_marketing) #check the dimensions
dim(train_marketing) #check the dimensions
```

40 4

160 4

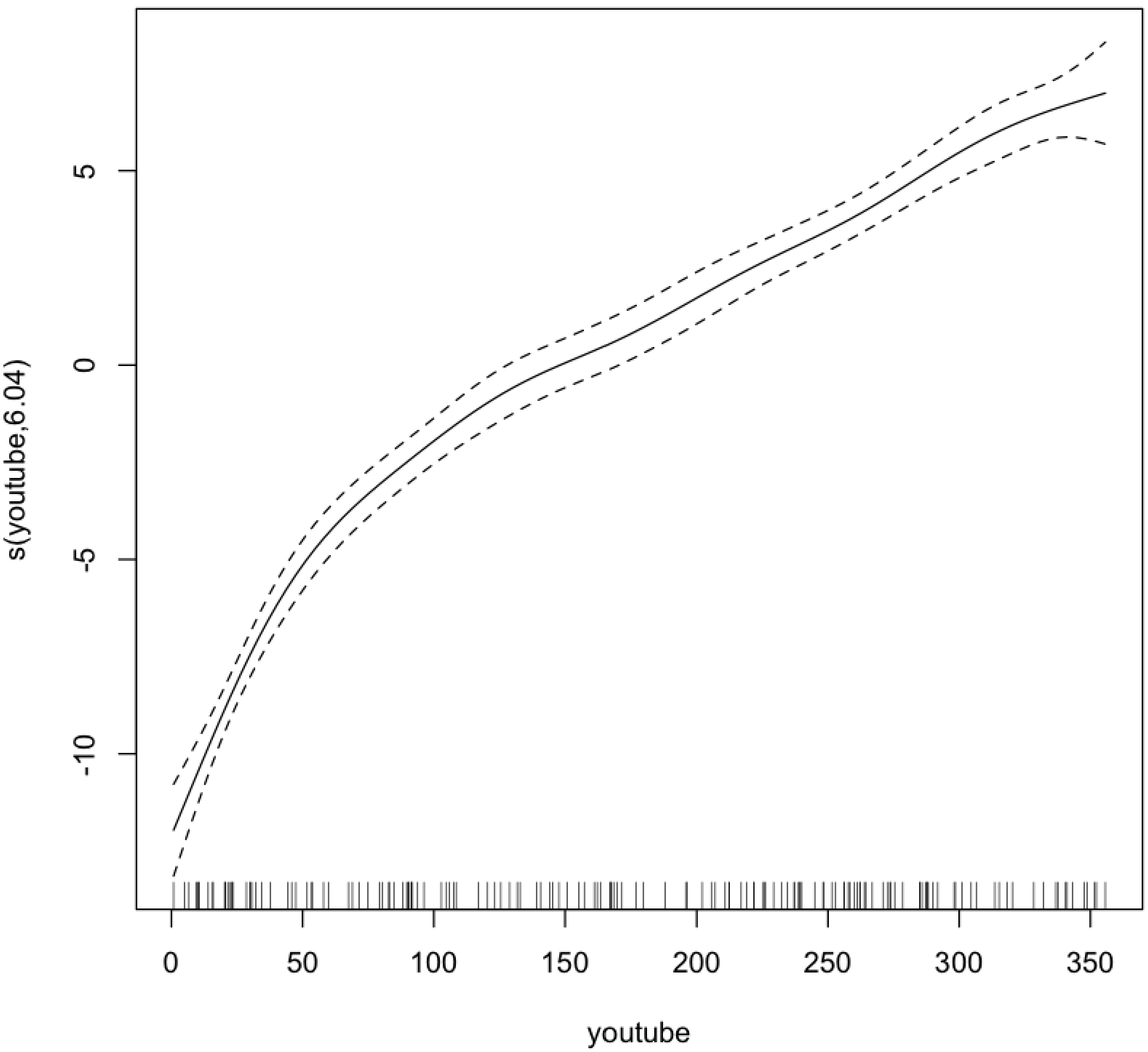
2.(a) Let's try a GAM on the marketing data!

Note that the relationship between `sales` and `youtube` is nonlinear. This was a problem for us back in the first course in this specialization, when we modeled the data as if it were linear. In the last module, we focused on modeling the relationship between `sales` and `youtube`, omitting the other variables. Now it's time to include the additional predictors.

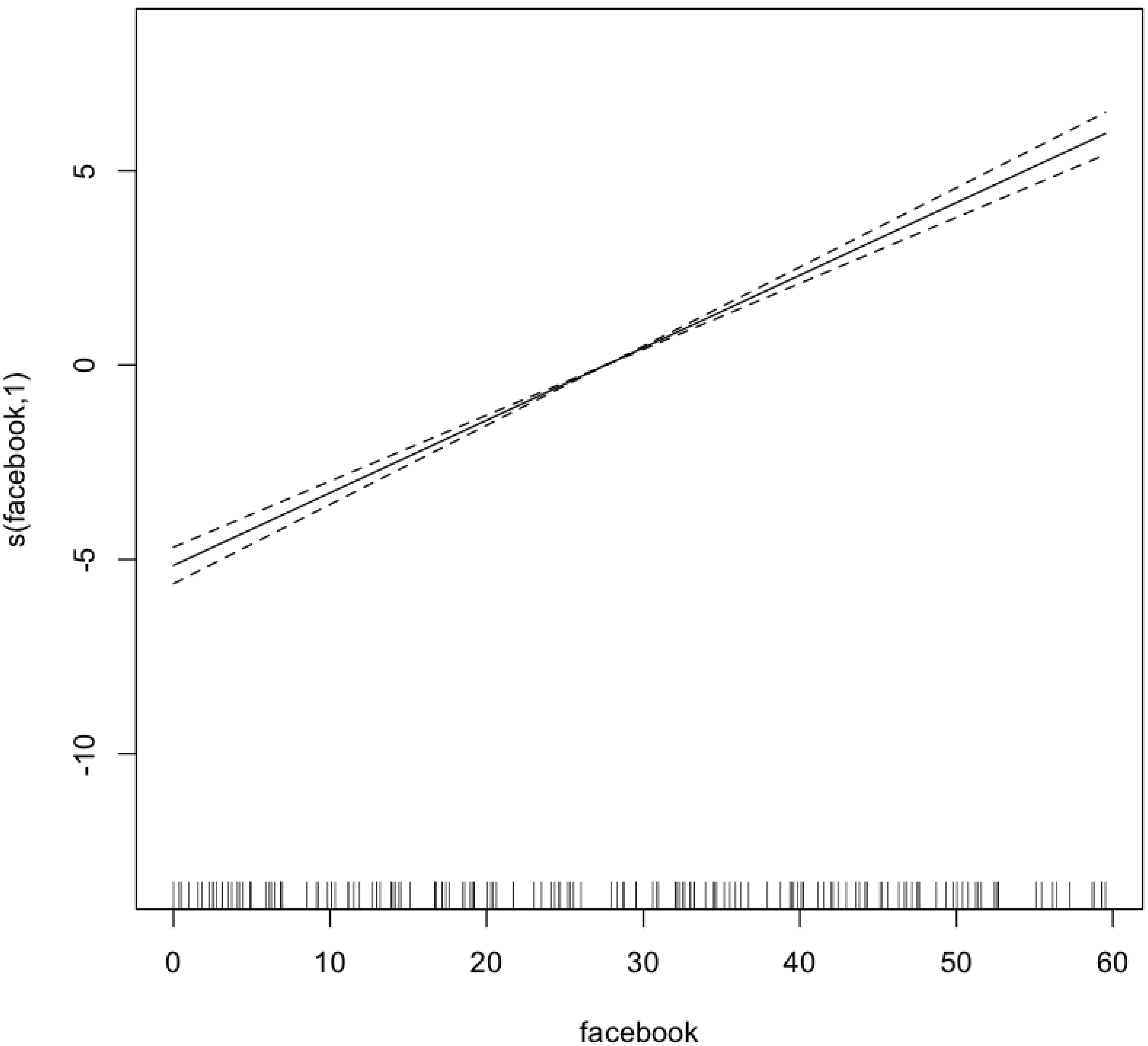
Using the `train_marketing` fit an additive model to the data and store it in `gam_marketing`. Produce the relevant added variable plots using `plot(gam_marketing)`. Comment on the fit of the model.

In [9]:
```
gam_marketing = gam(sales ~ s(youtube) + s(facebook) + s(newspaper), data = train_marketing)
summary(gam_marketing)
plot(gam_marketing)
```

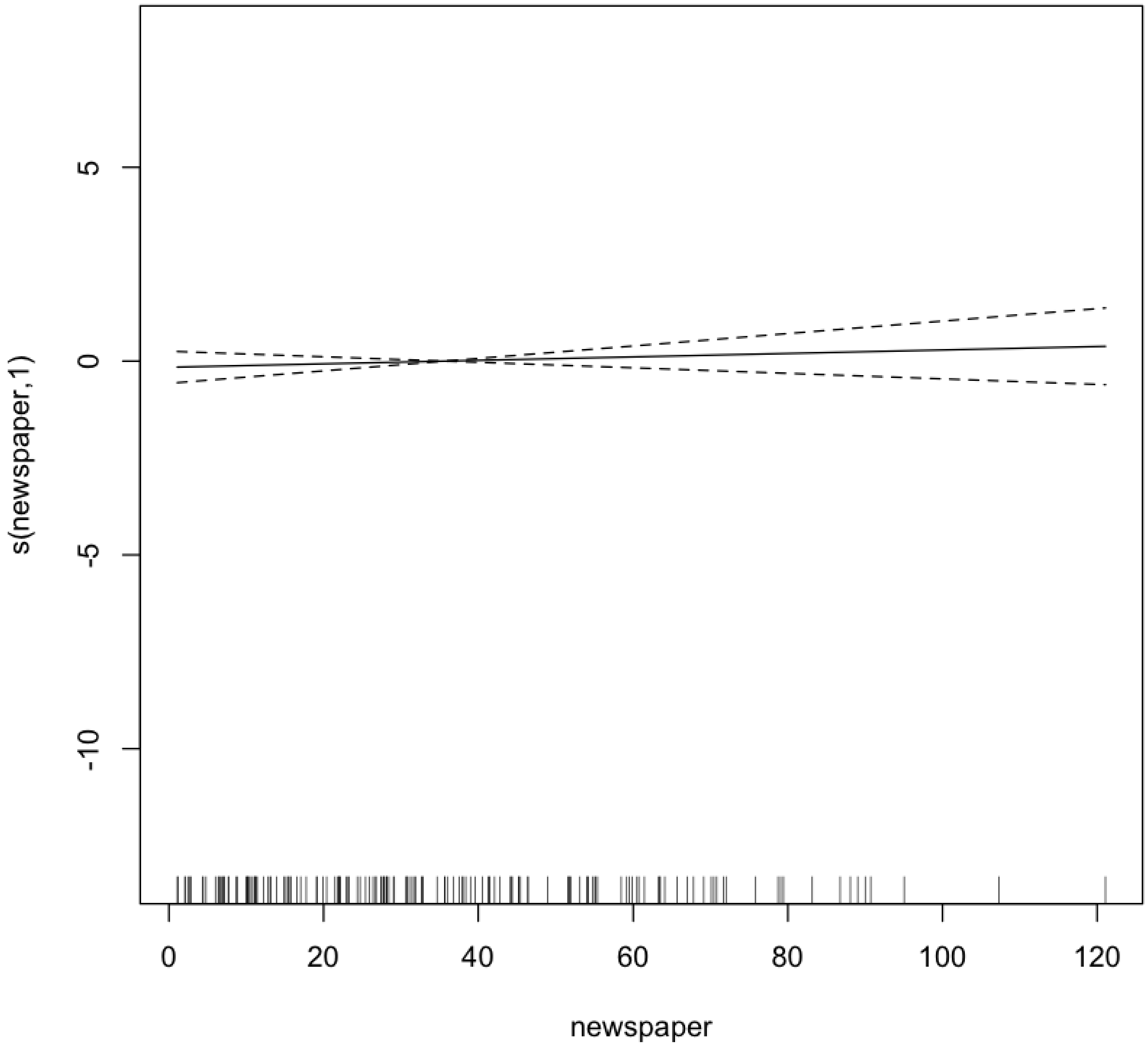
```
Family: gaussian
Link function: identity

Formula:
sales ~ s(youtube) + s(facebook) + s(newspaper)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  16.5742     0.1321   125.5   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
               edf Ref.df       F p-value
s(youtube)   6.037  7.127 201.356  <2e-16 ***
s(facebook)  1.000  1.000 481.319  <2e-16 ***
s(newspaper) 1.000  1.000   0.593   0.442
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.929   Deviance explained = 93.3%
GCV = 2.9572  Scale est. = 2.7902    n = 160
```



s(youtube,6.04)
5
0
-5
-10
0
50
100
150
200
250
300
350
youtube



s(facebook,1)
5
0
-5
-10
0
10
20
30
40
50
60
facebook



We note that the effective degrees of freedom for `newspaper` and `facebook` are close to one, suggesting that those predictors can enter the model linearly. Similarly, the added variable plots suggest that the relationships between `sales` and `newspaper` and `sales` and `facebook` are linear (that is, we can fit a straight line through the confidence bands).

(Note that in some training/test splits of the data, answers may differ! For example, sometimes, `facebook` might stay nonlinearly.)

2.(b) Semiparametric modeling of the marketing data

Refit the additive model based on your results from 2.(a). That is, if any predictors above should enter linearly, refit the model to reflect that. If any predictors are statistically insignificant, remove them from the model. Store your final model in `semiparametric_marketing`.

In [10]:
```
semiparametric_marketing = gam(sales ~ s(youtube) + facebook + newspaper,
data = marketing)
```

```r
summary(semiparametric_marketing)
```

```
Family: gaussian
Link function: identity

Formula:
sales ~ s(youtube) + facebook + newspaper

Parametric coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.3705531  0.2515706  45.198   <2e-16 ***
facebook     0.1958493  0.0073130  26.781   <2e-16 ***
newspaper   -0.0003011  0.0049641  -0.061    0.952
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
             edf Ref.df     F p-value
s(youtube) 6.347  7.499 214.2  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.926   Deviance explained = 92.9%
GCV = 3.0299  Scale est. = 2.8883    n = 200
```

`newspaper` is statistically insignificant and appears to be practically unimportant. So, we remove it.

In [11]:

```r
semiparametric_marketing = gam(sales ~ s(youtube) + facebook, data = marke
ting)
summary(semiparametric_marketing)
```

```
Family: gaussian
Link function: identity

Formula:
sales ~ s(youtube) + facebook

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.363835   0.225319   50.43   <2e-16 ***
facebook     0.195695   0.006834   28.63   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
             edf Ref.df     F p-value
s(youtube) 6.365  7.516 215.1  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.927   Deviance explained = 92.9%
GCV = 2.9985  Scale est. = 2.8731    n = 200
```

2.(c) Model comparisons

Now, let's do some model comparisons on the test data. Compute the mean squared prediction error

(MSPE) on the `test_marketing` data for the following three models:

- `gam_marketing` from 2.(a)
- `semiparametric_marketing` from 2.(b)
- `lm_marketing`, a linear regression model with `sales` is the response and `youtube` and `facebook` are predictors (fit on the `train_marketing` data).

State which model performs based according to this metric.

In [12]:
```
gam_predict = predict(gam_marketing, test_marketing)
mspe_gam = mean((test_marketing$sales - gam_predict)^2);
cat("The MSPE for the additive model from part (a) is", mspe_gam, ".")

semiparametric_predict = predict(semiparametric_marketing, test_marketing)
mspe_semiparametric = mean((test_marketing$sales - semiparametric_predict)
^2);
cat("The MSPE for the semiparametric model from part (b) is", mspe_semipar
ametric, ".")

lm_marketing = lm(sales ~ youtube + facebook, data = train_marketing)
lm_predict = predict(lm_marketing, test_marketing)
mspe_lm = mean((test_marketing$sales - lm_predict)^2);
cat("The MSPE for the linear regression model is", mspe_lm, ".")
```

```
The MSPE for the additive model from part (a) is 3.438202 .The MSPE for th
e semiparametric model from part (b) is 3.054544 .The MSPE for the linear
regression model is 4.197701 .
```

We see that the semiparametric model from part (b) performs slightly better than the GAM model from part (a). Both perform better than the linear regression model.

In [ ]:
```
```