Econ 573 Project Mateo

Juan M. Alvarez

```
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.4.3
```

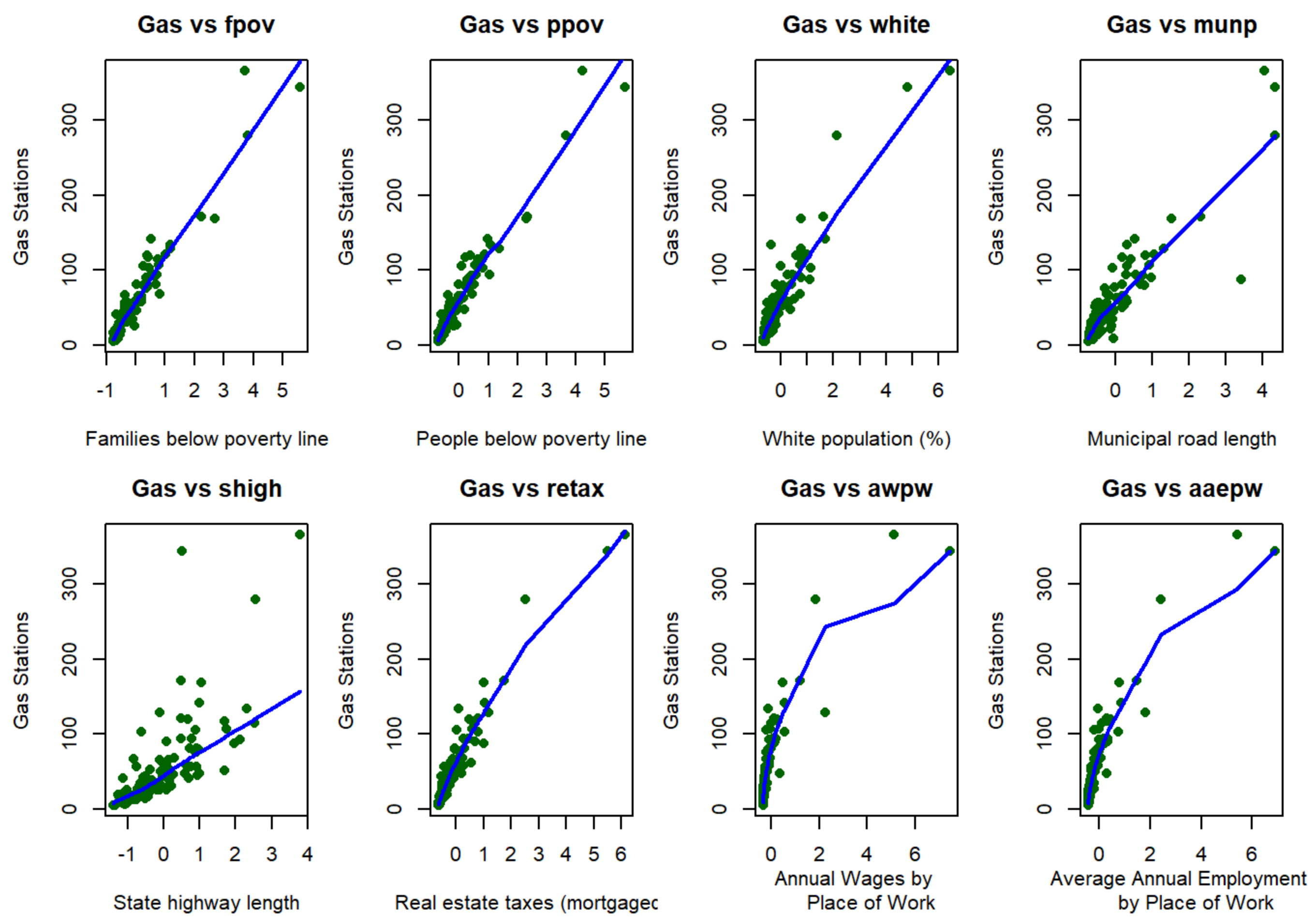
```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.4.3
```

```
# Loading our regularized data & raw data
Gas0 <- read.csv("C:/Users/mateo/nc_gas_scaled.csv", na.strings = "?", stringsAsFactors = T)
Gas_raw <- read.csv("C:/Users/mateo/RawData.csv", na.strings = "?", stringsAsFactors = T)
View(Gas_raw)
```

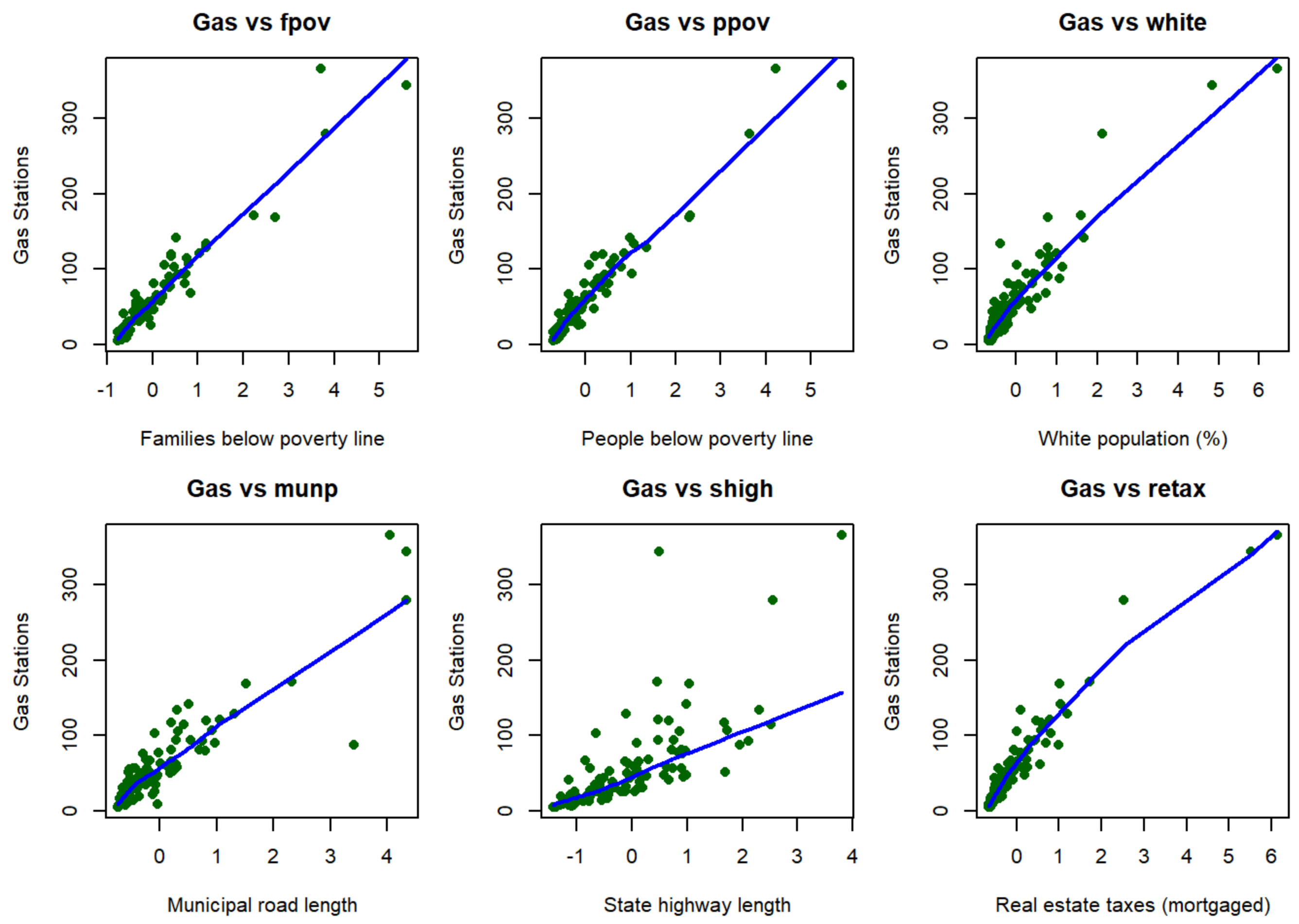
```r
# Set up a 3x3 plotting grid
par(mfrow = c(2, 4), mar = c(4, 4, 3, 1))  # 2 rows, 3 columns, adjusted margins

# Scatterplot: gas vs fpov
plot(Gas0$fpov, Gas0$gas,
     main = "Gas vs fpov",
     xlab = "Families below poverty line",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$fpov, Gas0$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs ppov
plot(Gas0$ppov, Gas0$gas,
     main = "Gas vs ppov",
     xlab = "People below poverty line",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$ppov, Gas0$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs white
plot(Gas0$white, Gas0$gas,
     main = "Gas vs white",
     xlab = "White population (%)",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$white, Gas0$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs munp
plot(Gas0$munp, Gas0$gas,
     main = "Gas vs munp",
     xlab = "Municipal road length",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$munp, Gas0$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs shigh
plot(Gas0$shigh, Gas0$gas,
     main = "Gas vs shigh",
     xlab = "State highway length",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$shigh, Gas0$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs retax
plot(Gas0$retax, Gas0$gas,
     main = "Gas vs retax",
     xlab = "Real estate taxes (mortgaged)",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$retax, Gas0$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs awpw
```

```
plot(Gas0$awpw, Gas0$gas,
     main = "Gas vs awpw",
     xlab = "Annual Wages by
     Place of Work",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$awpw, Gas0$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs aaepw
plot(Gas0$aaepw, Gas0$gas,
     main = "Gas vs aaepw",
     xlab = "Average Annual Employment
     by Place of Work",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas0$aaepw, Gas0$gas), col = "blue", lwd = 2)
```



```
# Choosing the model we get from lasso.
Gas_lasso <- subset(Gas0, select = c("fpov", "ppov", "white", "munp", "shigh", "retax", "gas"))
vars <- c("fpov", "ppov", "white", "munp", "shigh", "retax")
```

```
# Set up a 2x3 plotting grid
par(mfrow = c(2, 3), mar = c(4, 4, 3, 1))  # 2 rows, 3 columns, adjusted margins

# Scatterplot: gas vs fpov
plot(Gas_lasso$fpov, Gas_lasso$gas,
     main = "Gas vs fpov",
     xlab = "Families below poverty line",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_lasso$fpov, Gas_lasso$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs ppov
plot(Gas_lasso$ppov, Gas_lasso$gas,
     main = "Gas vs ppov",
     xlab = "People below poverty line",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_lasso$ppov, Gas_lasso$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs white
plot(Gas_lasso$white, Gas_lasso$gas,
     main = "Gas vs white",
     xlab = "White population (%)",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_lasso$white, Gas_lasso$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs munp
plot(Gas_lasso$munp, Gas_lasso$gas,
     main = "Gas vs munp",
     xlab = "Municipal road length",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_lasso$munp, Gas_lasso$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs shigh
plot(Gas_lasso$shigh, Gas_lasso$gas,
     main = "Gas vs shigh",
     xlab = "State highway length",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_lasso$shigh, Gas_lasso$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs retax
plot(Gas_lasso$retax, Gas_lasso$gas,
     main = "Gas vs retax",
     xlab = "Real estate taxes (mortgaged)",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_lasso$retax, Gas_lasso$gas), col = "blue", lwd = 2)
```



We observe several high leverage points in our data, corresponding to urban counties with exceptionally high gas station counts. While these reflect true observations, their influence on model fitting was assessed using leverage diagnostics. We retain these points but supplemented linear models with flexible, robust methods such as GAMs and smoothing splines to mitigate distortion and better capture nonlinear patterns in the remaining data.

Linear

```
lm.fit <- lm(gas ~ ., data = Gas_lasso)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = gas ~ ., data = Gas_lasso)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.359  -5.703  -1.632   4.298  33.271
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   58.040      1.092  53.126  < 2e-16 ***
## fpov           8.279      9.150   0.905  0.36787
## ppov          27.406     10.339   2.651  0.00944 **
## white         13.805      8.216   1.680  0.09628 .
## munp           3.203      2.777   1.154  0.25165
## shigh          9.121      1.592   5.729 1.23e-07 ***
## retax          2.707     10.458   0.259  0.79635
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.92 on 93 degrees of freedom
## Multiple R-squared:   0.97,  Adjusted R-squared:  0.9681
## F-statistic: 501.6 on 6 and 93 DF,  p-value: < 2.2e-16
```

To explore potential nonlinear effects in the predictors selected by Lasso, we fit a natural spline regression using all six variables with four degrees of freedom each. This allowed for flexible, piecewise-polynomial modeling while maintaining interpretability. Model comparisons indicated improvements in fit over a standard linear model, suggesting that key predictors such as state highway length and poverty levels may exert nonlinear effects on gas station density.

```
set.seed(6)
glm.fit <- glm(gas ~ ., data = Gas_lasso)
cv.error <- cv.glm(Gas_lasso, glm.fit, K = 10)
cv.error$delta
```

```
## [1] 197.493 191.702
```

Natural Splines

```
library(splines)
```

```
fit_ns <- lm(gas ~ ns(fpov, df = 4) + ns(ppov, df = 4) + ns(white, df = 4) + ns(munp, df = 4) +
ns(shigh, df = 4) + ns(retax, df = 4), data = Gas_lasso)

summary(fit_ns)
```

```
##
## Call:
## lm(formula = gas ~ ns(fpov, df = 4) + ns(ppov, df = 4) + ns(white,
##     df = 4) + ns(munp, df = 4) + ns(shigh, df = 4) + ns(retax,
##     df = 4), data = Gas_lasso)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.8129  -5.2015  -0.3934   4.3565  30.9747
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.521      5.224   0.674  0.50244
## ns(fpov, df = 4)1      9.899     29.452   0.336  0.73771
## ns(fpov, df = 4)2     60.472     43.820   1.380  0.17169
## ns(fpov, df = 4)3    -45.189     94.057  -0.480  0.63231
## ns(fpov, df = 4)4   -151.535    170.783  -0.887  0.37776
## ns(ppov, df = 4)1     10.859     31.841   0.341  0.73404
## ns(ppov, df = 4)2    -27.615     53.129  -0.520  0.60475
## ns(ppov, df = 4)3    313.555    120.487   2.602  0.01115 *
## ns(ppov, df = 4)4    611.429    234.485   2.608  0.01100 *
## ns(white, df = 4)1    -7.332     16.687  -0.439  0.66163
## ns(white, df = 4)2   -56.575     41.451  -1.365  0.17638
## ns(white, df = 4)3   244.049     82.613   2.954  0.00419 **
## ns(white, df = 4)4   525.973    160.635   3.274  0.00160 **
## ns(munp, df = 4)1      5.835      6.472   0.902  0.37014
## ns(munp, df = 4)2     33.207     13.126   2.530  0.01351 *
## ns(munp, df = 4)3     20.344     17.421   1.168  0.24657
## ns(munp, df = 4)4    -10.151     17.910  -0.567  0.57256
## ns(shigh, df = 4)1    -6.363      9.186  -0.693  0.49066
## ns(shigh, df = 4)2     6.453      9.884   0.653  0.51584
## ns(shigh, df = 4)3    25.577     19.594   1.305  0.19575
## ns(shigh, df = 4)4    41.356     19.436   2.128  0.03664 *
## ns(retax, df = 4)1    27.618     24.460   1.129  0.26245
## ns(retax, df = 4)2   128.562     57.535   2.234  0.02843 *
## ns(retax, df = 4)3  -207.197    110.680  -1.872  0.06510 .
## ns(retax, df = 4)4  -531.994    202.990  -2.621  0.01062 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.33 on 75 degrees of freedom
## Multiple R-squared:  0.9784, Adjusted R-squared:  0.9714
## F-statistic: 141.3 on 24 and 75 DF,  p-value: < 2.2e-16
```

The natural spline regression model achieved a remarkably high adjusted R^2 of 97.1%, indicating that allowing for non-linear relationships improved model performance relative to linear regression. Notably, the number of people below the poverty line, the percentage of white residents, and real estate taxes exhibited strong non-linear effects on gas station density. These variables showed significant higher-order spline terms, suggesting that their influence on gas station counts changes at extreme values. In contrast, other predictors such as families below the poverty line displayed primarily linear relationships.

Linear regression already provided a strong baseline model, with state highway length and the number of people below the poverty line emerging as key linear predictors of gas station counts. However, the natural spline model offered superior performance across all evaluation metrics. Importantly, splines uncovered non-linear effects in variables like white population percentage and real estate taxes that were not statistically significant in the linear model, but became highly significant when allowing for flexible curvature.

```
set.seed(6)

folds <- sample(rep(1:10, length.out = nrow(Gas_lasso)))

cv_error_ns <- rep(NA, 10)

for (k in 1:10) {

  train_data <- Gas_lasso[folds != k, ]
  test_data  <- Gas_lasso[folds == k, ]

  fit_ns_k <- lm(gas ~ ns(fpov, df = 4) + ns(ppov, df = 4) + ns(white, df = 4) + ns(munp, df =
4) + ns(shigh, df = 4) + ns(retax, df = 4), data = train_data)

  pred <- predict(fit_ns_k, newdata = test_data)

  cv_error_ns[k] <- mean((test_data$gas - pred)^2)
}

cv_error_ns
```

```
##  [1]  77.94814 410.34202 238.01134 315.27815  79.08716 427.78950  75.42968
##  [8] 180.05011 273.83119  80.33050
```

```
mean(cv_error_ns)
```

```
## [1] 215.8098
```

```
sd(cv_error_ns)
```

```
## [1] 138.8726
```

```
# Comparing AIC's
AIC(lm.fit, fit_ns)
```

	df <dbl>	AIC <dbl>
lm.fit	8	770.7408
fit_ns	26	774.1224

2 rows

Although the natural splines model provided better in-sample fit (higher R², lower residual error), its AIC was slightly higher than the linear model, indicating that the performance gains may not fully justify the increase in model complexity. This suggests that the linear model remains highly competitive, though non-linear modeling revealed important curvature in certain predictors.

```
anova(lm.fit, fit_ns)
```

	Res.Df <dbl>	RSS <dbl>	Df <dbl>	Sum of Sq <dbl>	F <dbl>	Pr(>F) <dbl>
1	93	11100.003	NA	NA	NA	NA
2	75	8010.568	18	3089.435	1.606958	0.07994046

2 rows

An ANOVA test comparing the linear and natural spline models indicated that while the spline model reduced residual variance, this improvement was only weakly significant (p = 0.0799). This suggests the presence of some non-linear relationships in the data, but also reinforces the strength of the linear model's predictive ability.

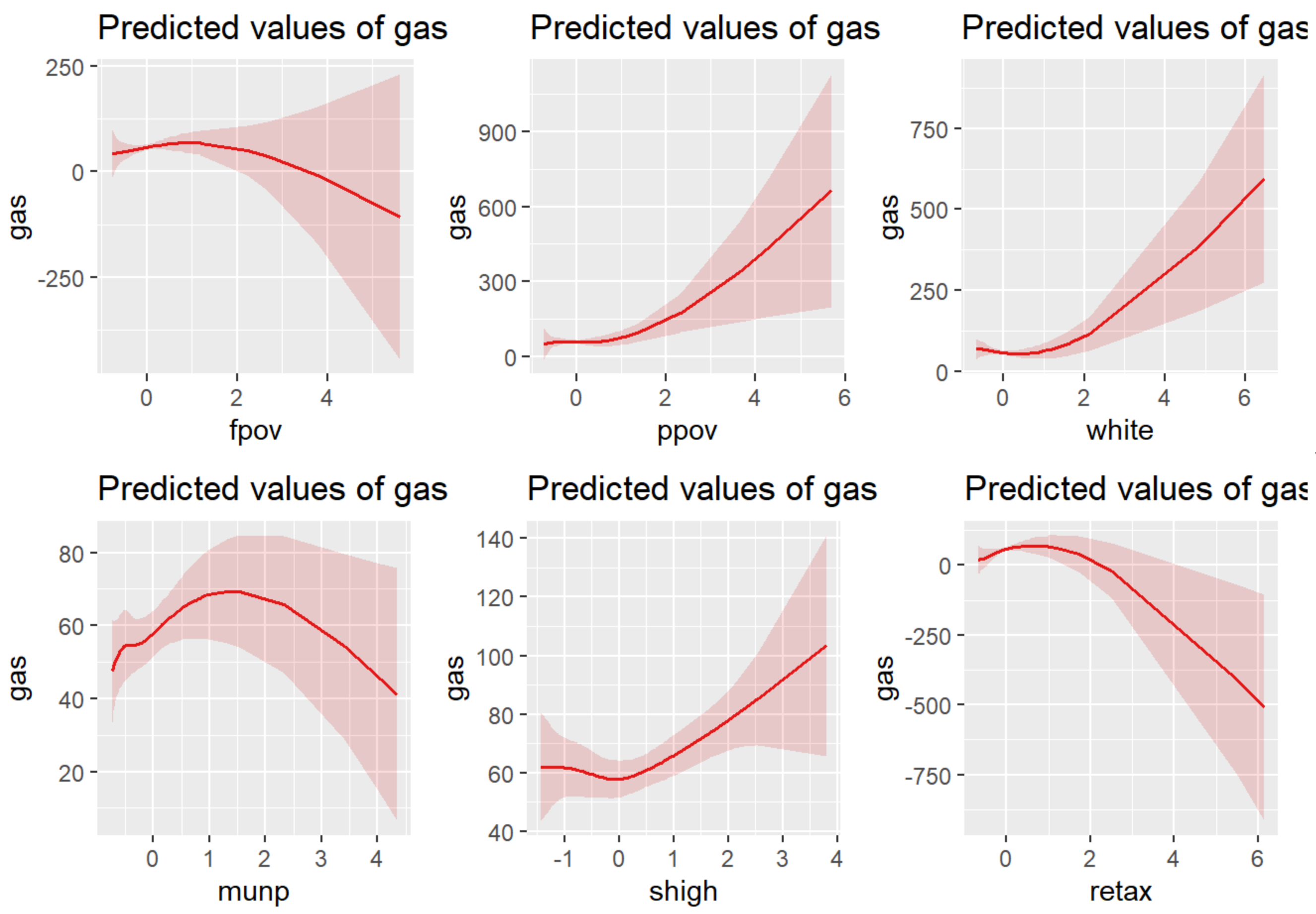
```
# Plotting the spline fits for each variable
library(sjPlot)
```

```
## Warning: package 'sjPlot' was built under R version 4.4.3
```

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.4.3
```

```
# Plot the spline effects
p1 <- plot_model(fit_ns, type = "pred", terms = c("fpov"))
p2 <- plot_model(fit_ns, type = "pred", terms = c("ppov"))
p3 <- plot_model(fit_ns, type = "pred", terms = c("white"))
p4 <- plot_model(fit_ns, type = "pred", terms = c("munp"))
p5 <- plot_model(fit_ns, type = "pred", terms = c("shigh"))
p6 <- plot_model(fit_ns, type = "pred", terms = c("retax"))
(p1 | p2 | p3) / (p4 | p5 | p6)
```



we can see how the high leverage points distort the relationship.

For multivariable modeling, we proceed with generalized additive models (GAMs), which provide a more flexible and interpretable framework for capturing non-linearity across multiple predictors.

GAM’s

The data will decide how non-linear each variable is without having to manually picking degrees of freedom.

```
library(mgcv)
```

```
## Warning: package 'mgcv' was built under R version 4.4.3
```

```
## Cargando paquete requerido: nlme
```

```
## Warning: package 'nlme' was built under R version 4.4.3
```

```
## This is mgcv 1.9-3. For overview type 'help("mgcv-package")'.
```

```
library(boot)
```

```
gam_fit <- gam(gas ~ s(fpov) + s(ppov) + s(white) + s(munp) + s(shigh) + s(retax), data = Gas_la
sso)
summary(gam_fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## gas ~ s(fpov) + s(ppov) + s(white) + s(munp) + s(shigh) + s(retax)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  58.0400     0.9047   64.16   <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(fpov)  1.000  1.000 5.002 0.028031 *
## s(ppov)  1.000  1.000 1.093 0.298825
## s(white) 6.638  7.370 2.872 0.009931 **
## s(munp)  5.012  5.866 4.416 0.000792 ***
## s(shigh) 1.828  2.280 3.991 0.023166 *
## s(retax) 1.344  1.529 2.132 0.070757 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.978   Deviance explained = 98.2%
## GCV = 99.594  Scale est. = 81.845    n = 100
```

The GAM model provided an excellent fit (adj. $R^2 = 0.978$), explaining 98.2% of the variance in gas station counts. Results indicated that families below the poverty line (fpov) had a significant linear effect, while people below the poverty line (ppov) did not offer additional predictive value once fpov was included. The percentage of white residents and municipal road length exhibited strong non-linear relationships, suggesting that their effects on gas station density vary across different values. State highway length demonstrated a modest non-linear effect, while real estate taxes showed a weak, borderline-significant non-linear pattern. Overall, the GAM approach effectively captured both linear and non-linear effects, supporting its use over fully linear models or manually specified splines.

```
set.seed(6)

# Create 10 folds
folds <- sample(rep(1:10, length.out = nrow(Gas_lasso)))

cv_error <- rep(NA, 10)

for (k in 1:10) {

  # Split data
  train_data <- Gas_lasso[folds != k, ]
  test_data  <- Gas_lasso[folds == k, ]

  # Fit GAM on training data
  gam_fit_k <- gam(gas ~ s(fpov) + s(ppov) + s(white) + s(munp) + s(shigh) + s(retax),
                   data = train_data)

  # Predict on test data
  pred <- predict(gam_fit_k, newdata = test_data)

  # Compute Test Error (MSE)
  cv_error[k] <- mean((test_data$gas - pred)^2)
}

# View all CV errors
cv_error
```

```
##  [1]   51.11033  559.25253  246.90551 1847.07801   56.93096  556.52052
##  [7] 1942.15429  822.98652  207.08799   55.66343
```

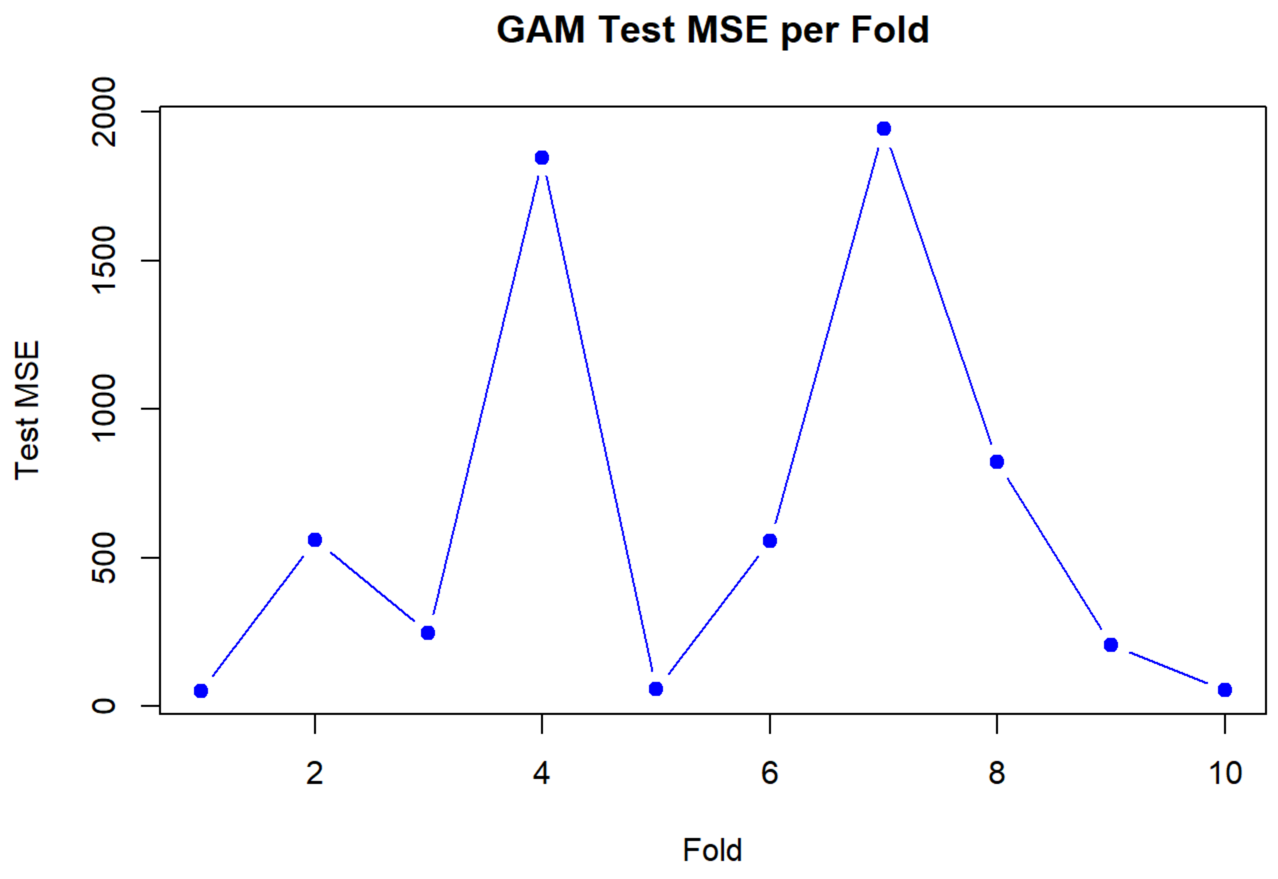
```
# Average Test Error across folds
mean(cv_error)
```

```
## [1] 634.569
```

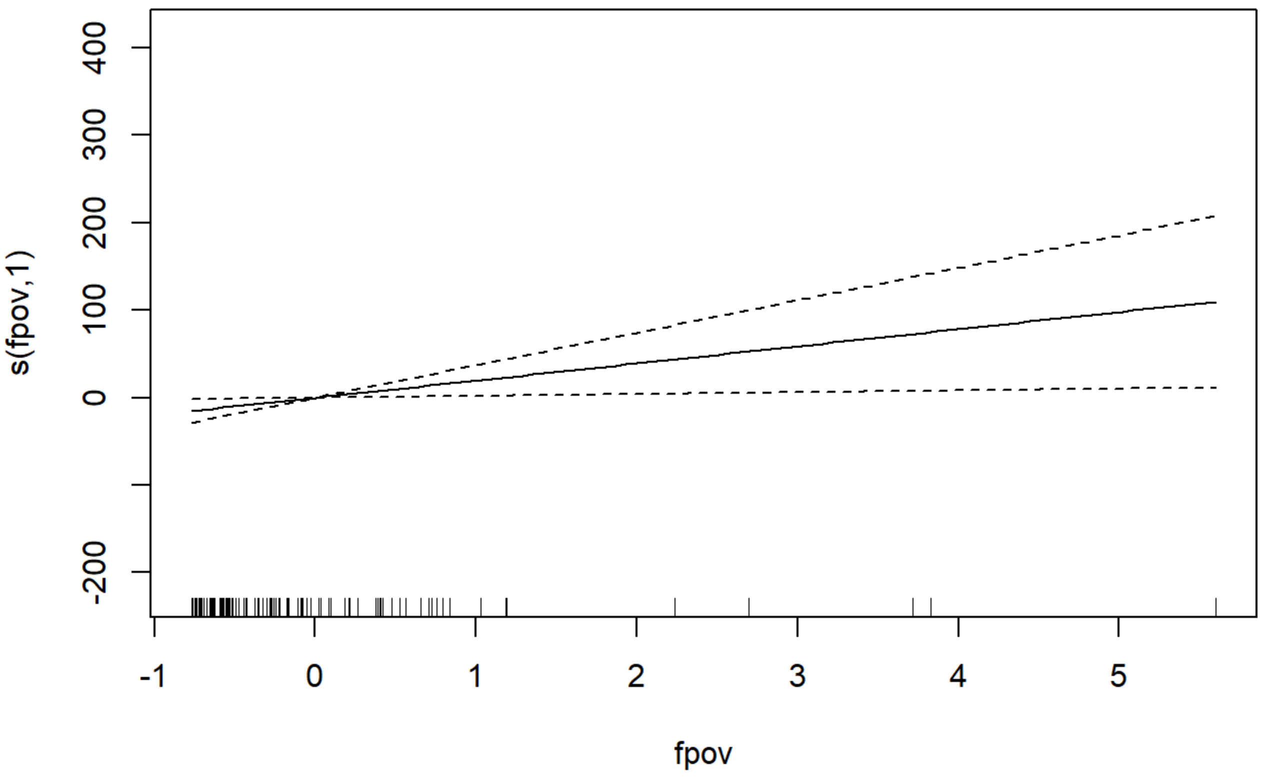
Given the limitations of applying cv.glm() to GAMs, we manually performed 10-fold cross-validation. The average test MSE across the 10 folds was approximately [insert mean(cv_error)], providing an unbiased estimate of the model's out-of-sample predictive accuracy.

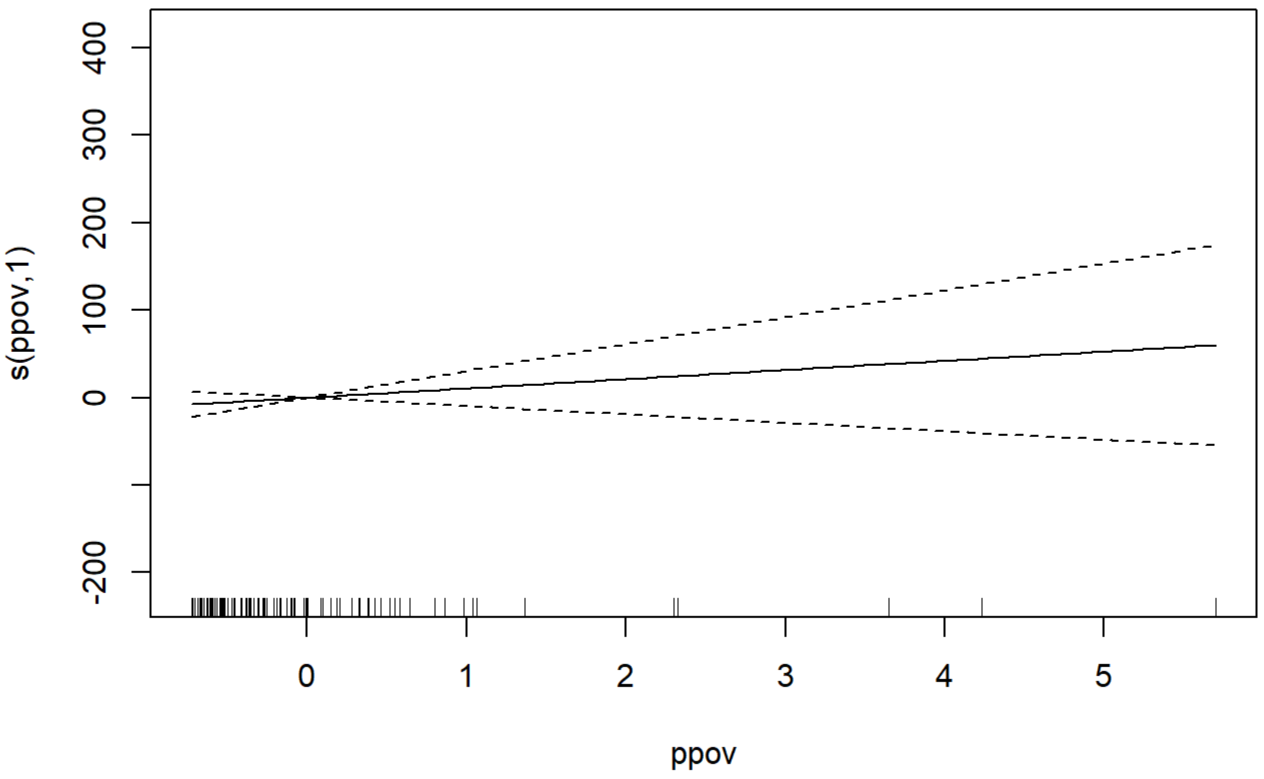
While the average test MSE of the GAM model was lower than the linear model, we observed substantial variation in performance across folds. In particular, some folds yielded very high prediction errors, suggesting that the GAM may be sensitive to specific data splits or overfitting certain regions of the data. This variability contrasts with the more stable performance observed in the LASSO model."

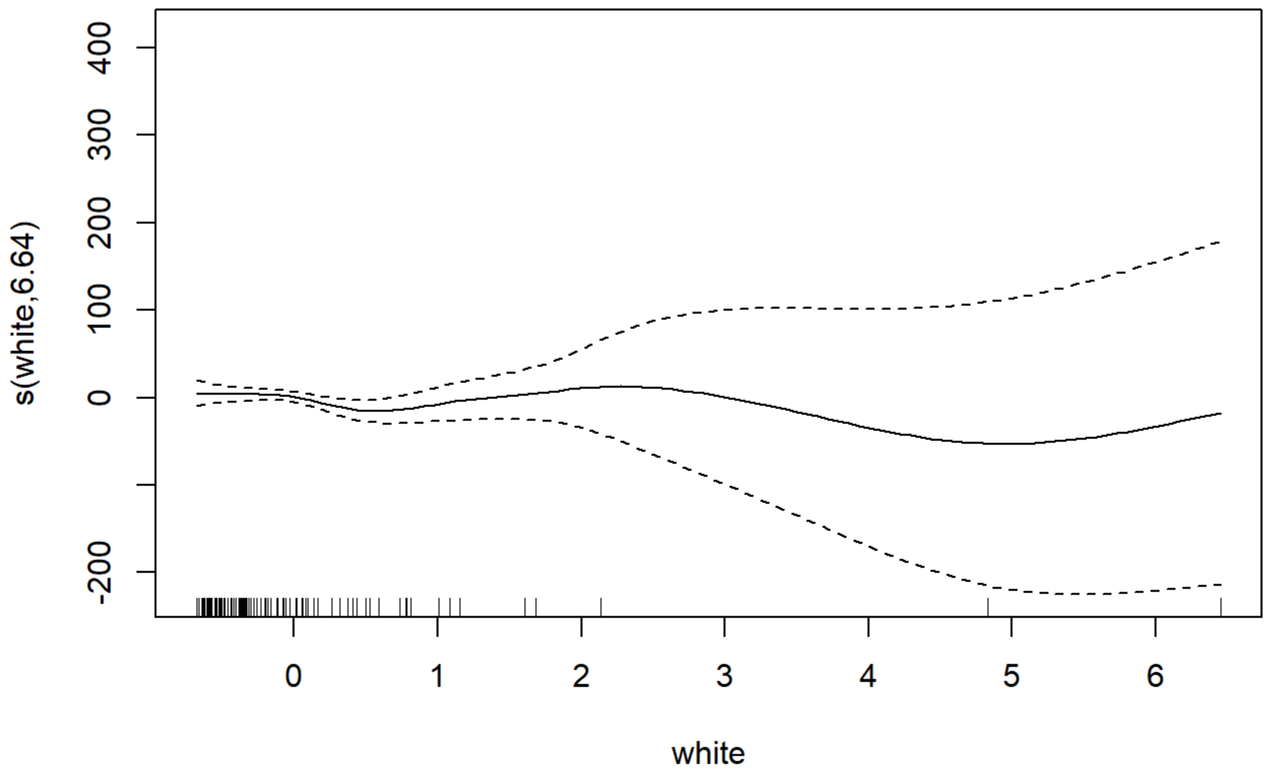
```
plot(1:10, cv_error, type = "b", pch = 19, col = "blue",
     main = "GAM Test MSE per Fold", xlab = "Fold", ylab = "Test MSE")
```

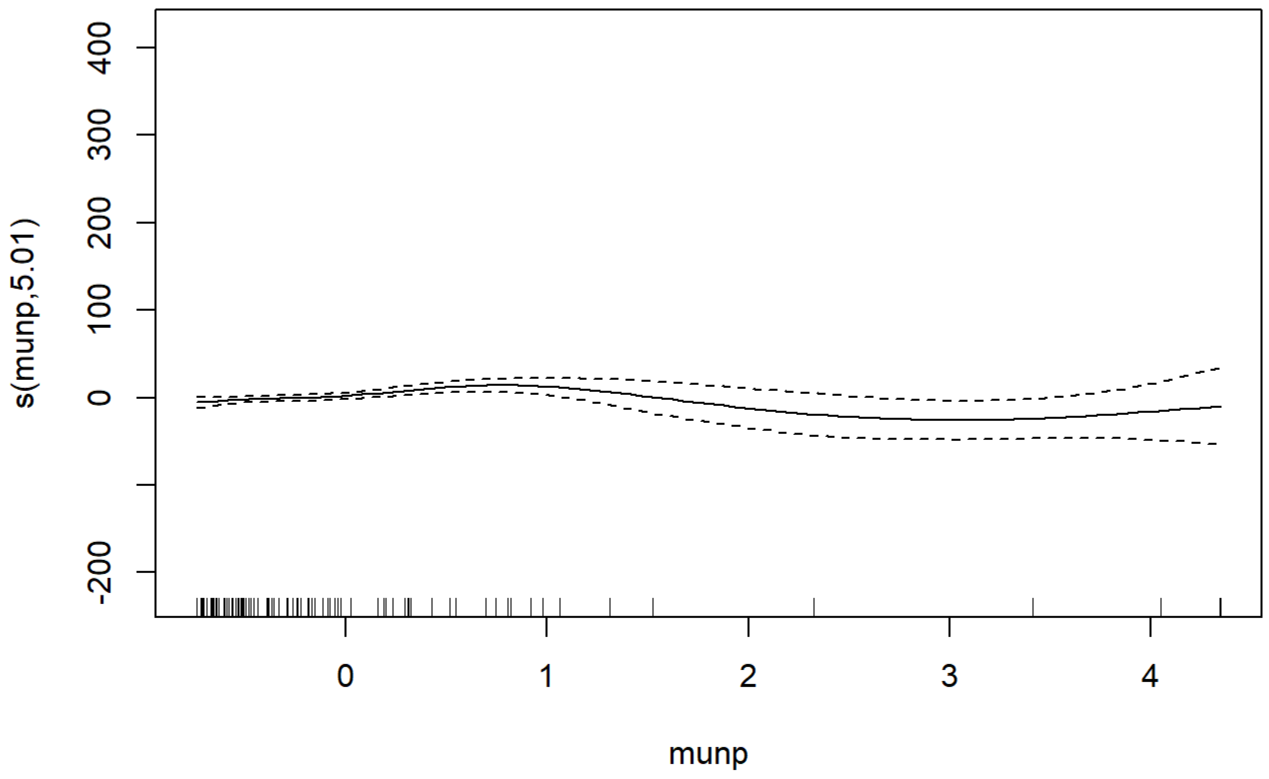


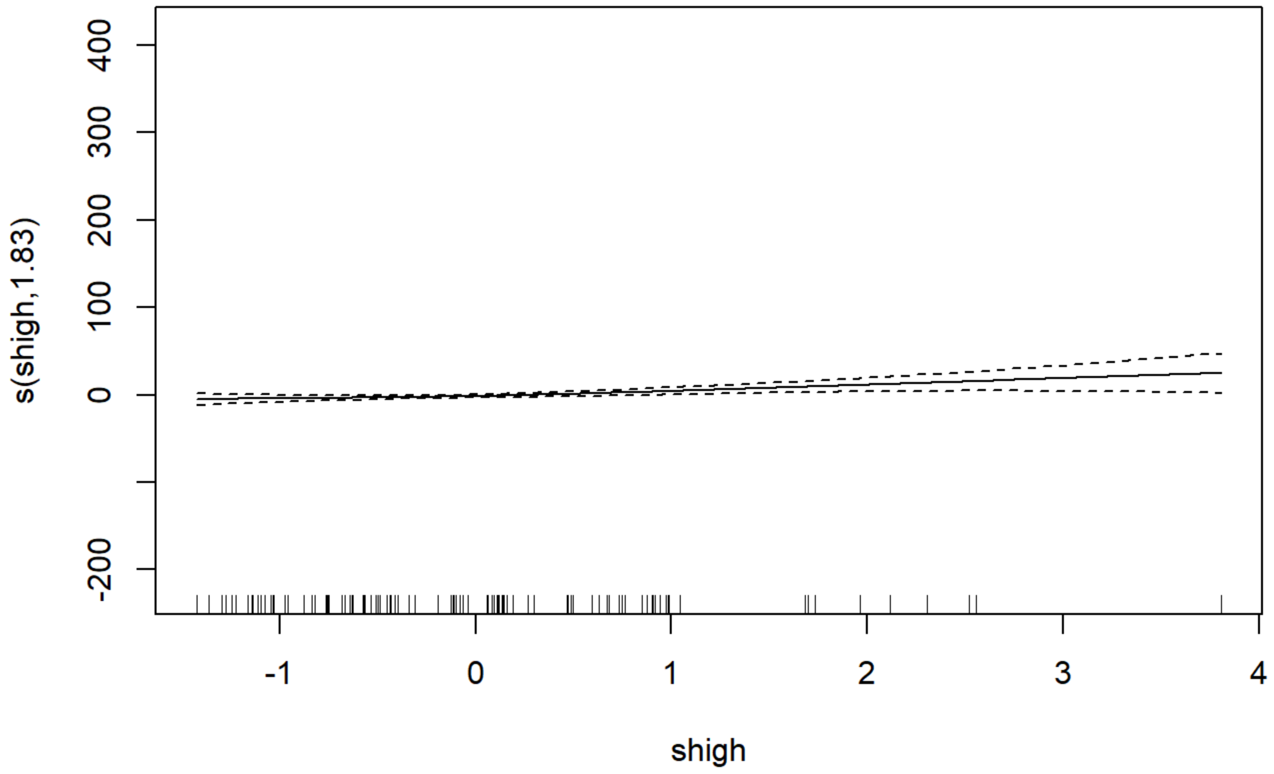
```
# Automatic plots for each variable
plot(gam_fit, se = TRUE, shade = TRUE)
```

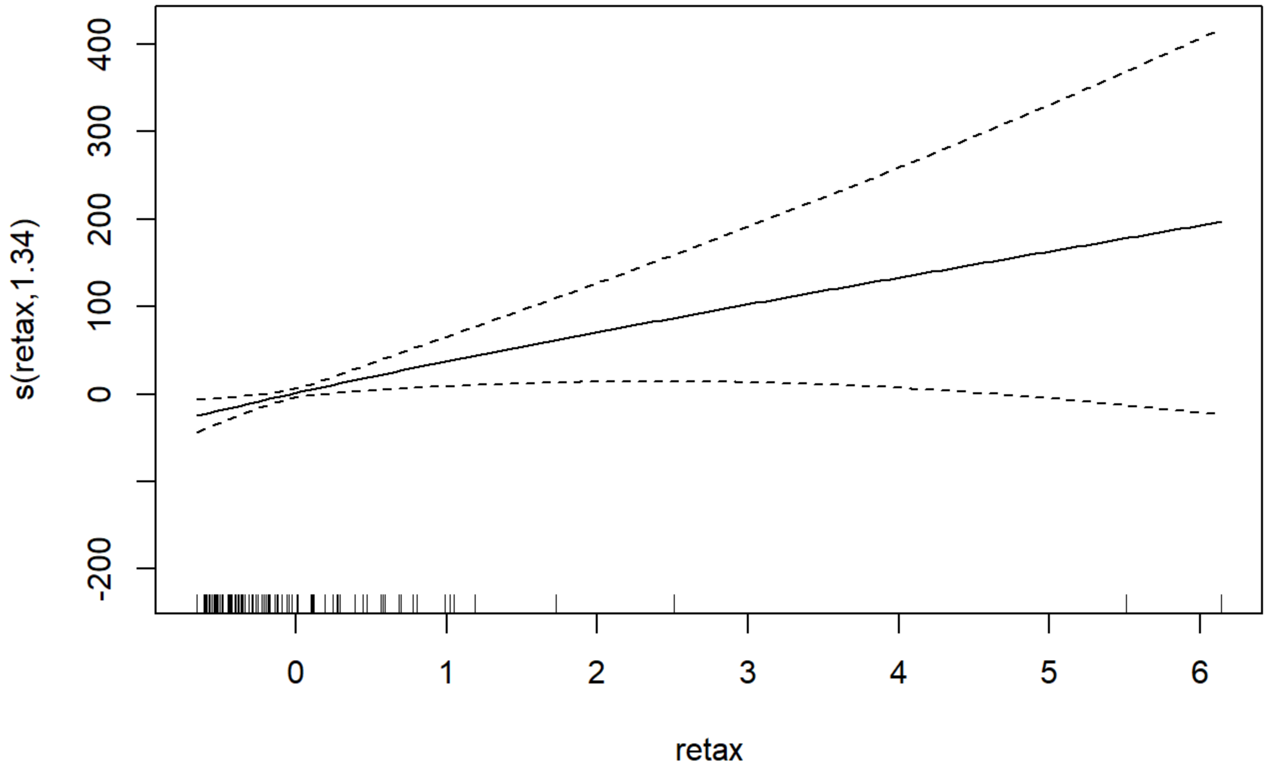

s(fpov,1)
400
300
200
100
0
-200
-1
0
1
2
3
4
5
fpov


s(ppov,1)
400
300
200
100
0
-200
0
1
2
3
4
5
ppov


s(white,6.64)
400
300
200
100
0
-200
0
1
2
3
4
5
6
white


s(munp,5.01)
400
300
200
100
0
-200
0
1
2
3
4
munp


s(shigh,1.83)
400
300
200
100
0
-200
-1
0
1
2
3
4
shigh


s(retax,1.34)
400
300
200
100
0
-200
0
1
2
3
4
5
6
retax

```
# Check AIC
AIC(lm.fit, fit_ns, gam_fit)
```

	df <dbl>	AIC <dbl>
lm.fit	8.00000	770.7408
fit_ns	26.00000	774.1224
gam_fit	18.82213	742.2857

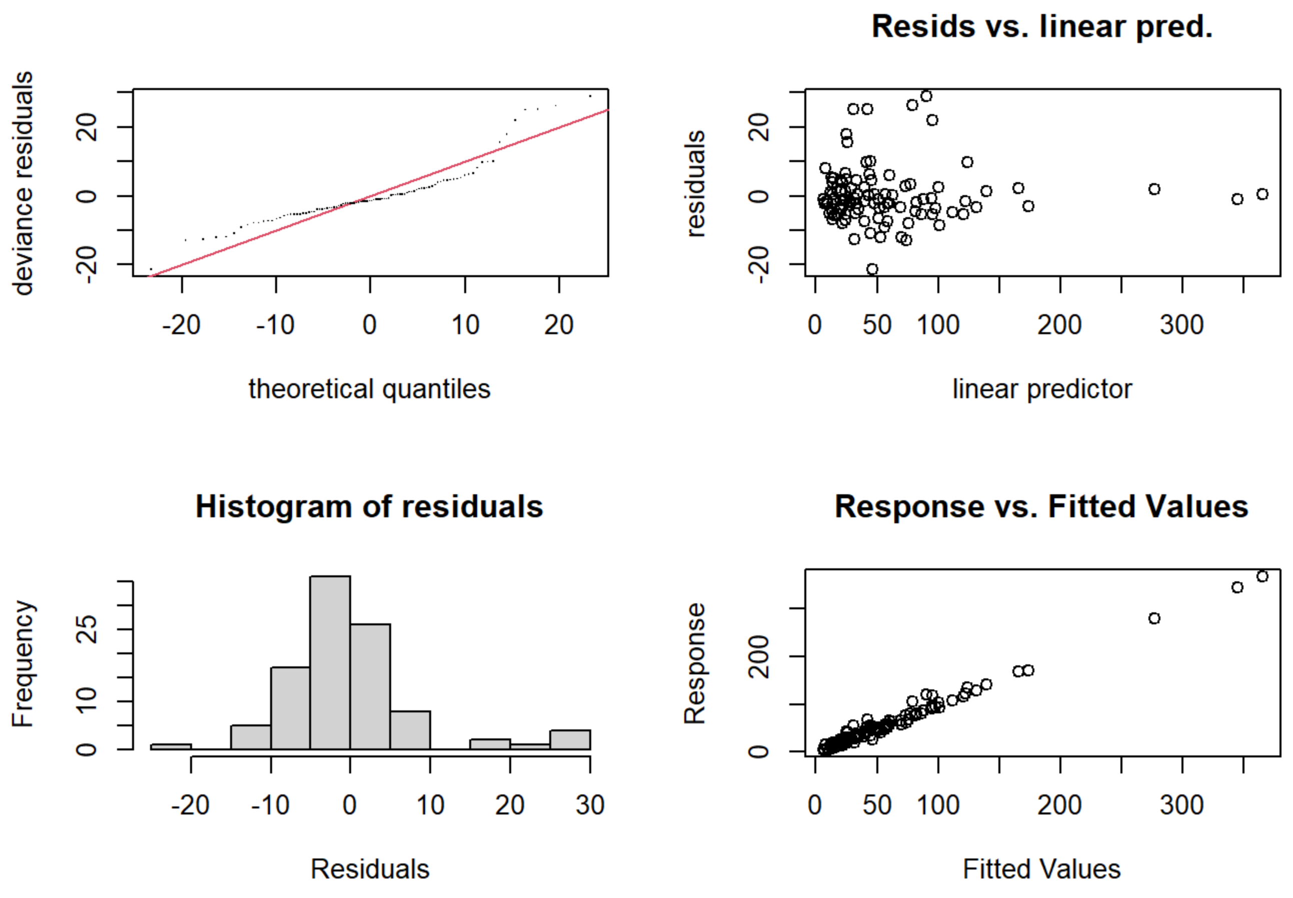
3 rows

```
# Check adjusted R-squared
summary(gam_fit)$r.sq
```

```
## [1] 0.9781187
```

Among all models considered, the generalized additive model (GAM) achieved the best overall performance with an AIC of 742.29, substantially lower than both the linear model (AIC = 770.74) and the natural spline model (AIC = 774.12). The GAM effectively captured both linear and non-linear relationships across predictors while maintaining a reasonable level of model complexity (effective degrees of freedom = 18.8). In particular, the variables representing families below the poverty line (fpov) and state highway length (shigh) displayed primarily linear effects, while white population percentage (white) and municipal road length (munp) exhibited substantial non-linear patterns.

```
gam.check(gam_fit)
```



```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 20 iterations.
## The RMS GCV score gradient at convergence was 2.710732e-06 .
## The Hessian was positive definite.
## Model rank =  55 / 55
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##             k'  edf k-index p-value
## s(fpov)  9.00 1.00    0.93    0.22
## s(ppov)  9.00 1.00    1.10    0.80
## s(white) 9.00 6.64    1.01    0.49
## s(munp)  9.00 5.01    1.05    0.66
## s(shigh) 9.00 1.83    1.20    0.97
## s(retax) 9.00 1.34    1.06    0.69
```

Diagnostic plots for the final GAM model indicated no major violations of model assumptions. The residuals appeared randomly distributed with stable variance, and the histogram of residuals suggested approximate normality. The response versus fitted values plot showed a strong linear pattern, with only mild deviation at the highest fitted values, consistent with a few urban counties having extreme gas station counts. The basis dimension checks confirmed that the chosen level of smoothness was appropriate, with no evidence of underfitting.

Trying GAM without ppov

Variables like ppov were not statistically significant within the GAM framework, likely due to their strong correlation with fpov. To assess model parsimony, we compared a reduced GAM excluding ppov, finding that its removal did not substantially degrade model performance. Thus, the final model retained variables that were either significant, theoretically important, or contributed to predictive accuracy.

```
gam_fit2 <- gam(gas ~ s(fpov) + s(white) + s(munp) + s(shigh) + s(retax),
                data = Gas_lasso)

summary(gam_fit2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## gas ~ s(fpov) + s(white) + s(munp) + s(shigh) + s(retax)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  58.0400     0.9035   64.24   <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(fpov)  1.000  1.000 45.620  < 2e-16 ***
## s(white) 6.627  7.355  3.770 0.001001 **
## s(munp)  5.147  6.028  4.963 0.000221 ***
## s(shigh) 1.939  2.426  3.595 0.030989 *
## s(retax) 1.212  1.331  4.006 0.019661 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.978   Deviance explained = 98.2%
## GCV = 98.262  Scale est. = 81.632    n = 100
```

The final GAM model excluding ppov achieved an adjusted R^2 of 0.978 and explained 98.2% of the deviance in gas station counts, outperforming all prior models in both fit and simplicity. The analysis revealed that the percentage of families below the poverty line (fpov) had a strong linear effect on gas station density. In contrast, variables such as white population percentage (white) and municipal road length (munp) exhibited significant non-linear relationships, indicating that their effects varied across different ranges of their values. Both state highway length (shigh) and real estate taxes (retax) showed mild non-linear trends, suggesting diminishing or accelerating effects at higher levels. The removal of ppov improved model parsimony without sacrificing predictive performance, further supporting the robustness of the selected predictors.

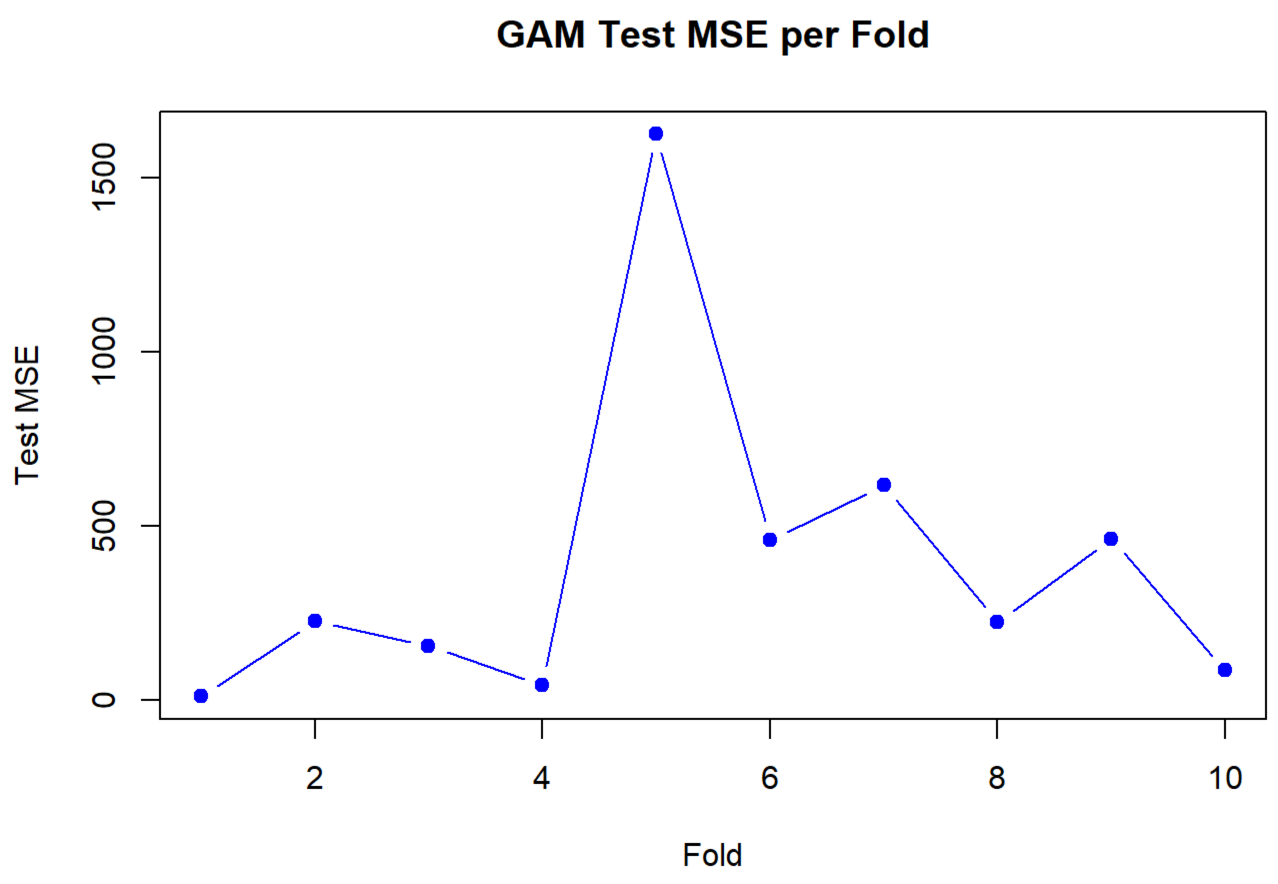
```
set.seed(6)

# Create 10 folds
folds <- sample(rep(1:10, length.out = nrow(Gas_lasso)-1))

cv_error2 <- rep(NA, 10)

for (k in 1:10) {

  # Split data
  train_data <- Gas_lasso[folds != k, ]
  test_data  <- Gas_lasso[folds == k, ]

  # Fit GAM on training data
  gam_fit_k <- gam(gas ~ s(fpov) + s(white) + s(munp) + s(shigh) + s(retax),
                   data = train_data)

  # Predict on test data
  pred <- predict(gam_fit_k, newdata = test_data)

  # Compute Test Error (MSE)
  cv_error2[k] <- mean((test_data$gas - pred)^2)
}

# View all CV errors
cv_error2
```

```
##  [1]    9.779729  224.718157  153.235071   42.059557 1624.208346  459.127544
##  [7]  615.926710  223.094333  462.390038   84.400761
```

```
# Average Test Error across folds
mean(cv_error2)
```

```
## [1] 389.894
```

```
plot(1:10, cv_error2, type = "b", pch = 19, col = "blue",
     main = "GAM Test MSE per Fold", xlab = "Fold", ylab = "Test MSE")
```



```
AIC(gam_fit, gam_fit2)
```

	df <dbl>	AIC <dbl>
gam_fit	18.82213	742.2857
gam_fit2	17.92414	741.3169

2 rows

Given the non-significance of ppov in the initial GAM and its likely redundancy with fpov, we refit the model excluding ppov. This reduced model achieved a slightly lower AIC (741.32 vs 742.29), indicating improved parsimony without sacrificing model fit. As such, the final model retained only variables that were either statistically significant or contributed meaningfully to predictive performance.

Regression Trees

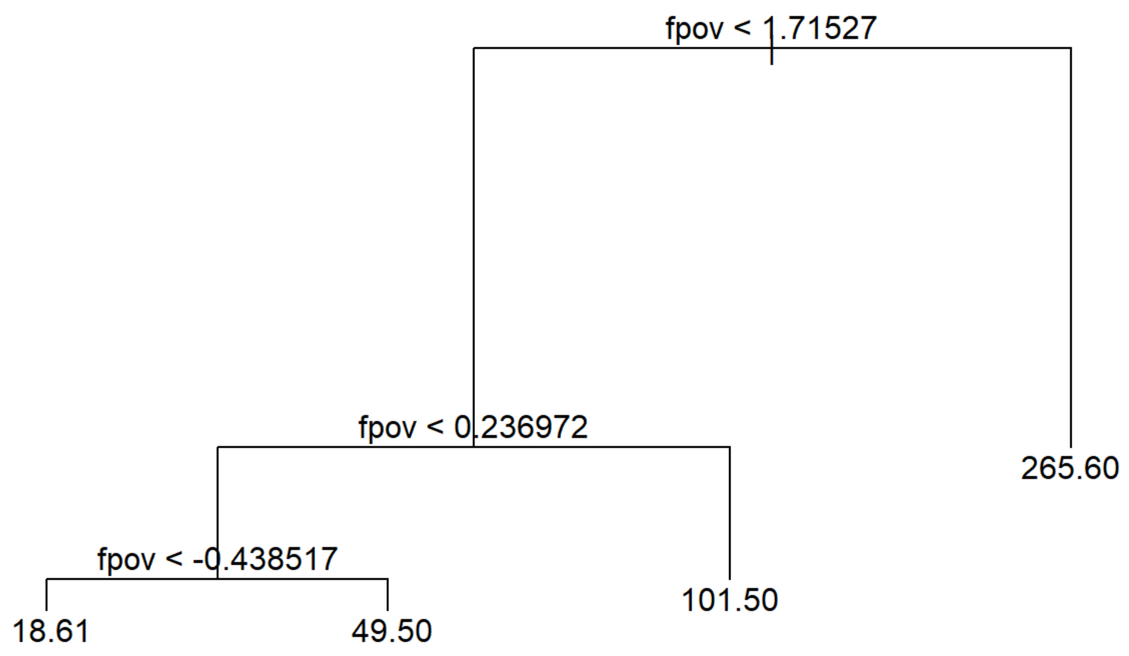
```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.4.3
```

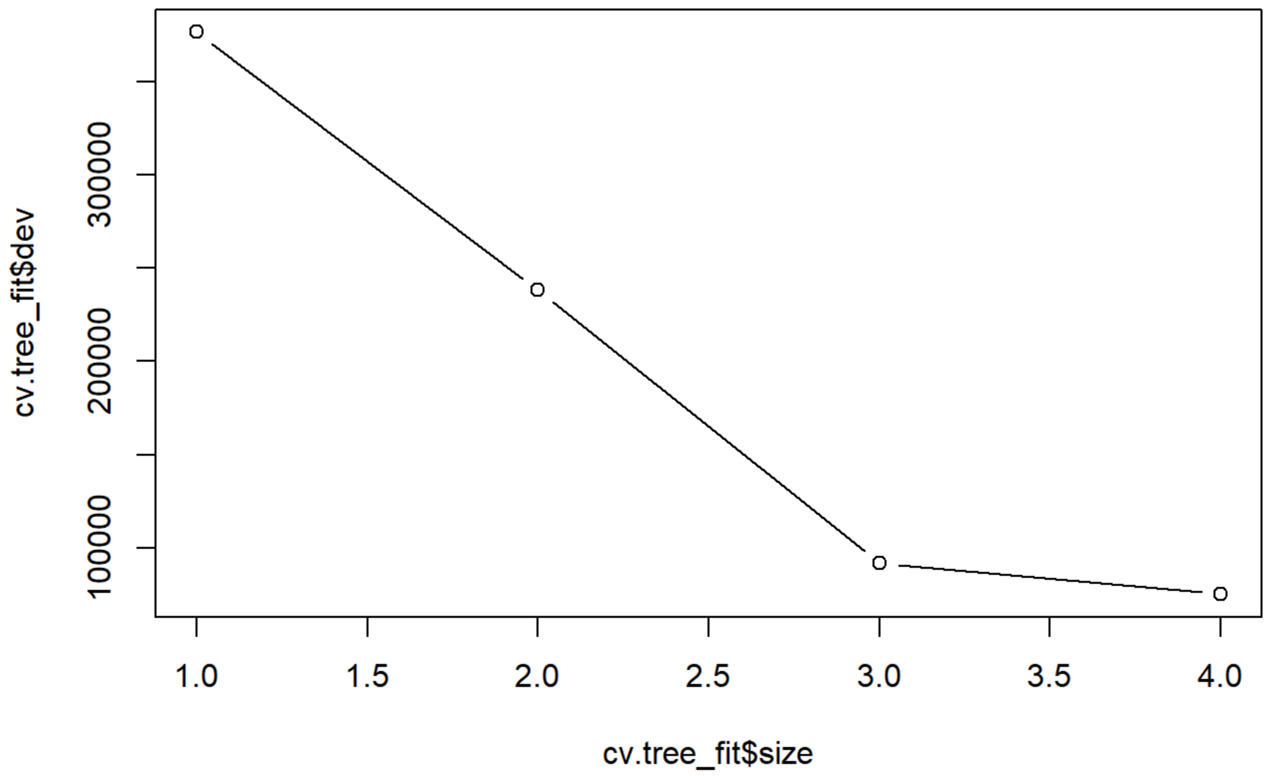
```r
# Fit tree
tree_fit <- tree(gas ~ fpov + white + munp + shigh + retax, data = Gas_lasso)
# Summary
summary(tree_fit)
```

```
##
## Regression tree:
## tree(formula = gas ~ fpov + white + munp + shigh + retax, data = Gas_lasso)
## Variables actually used in tree construction:
## [1] "fpov"
## Number of terminal nodes:  4
## Residual mean deviance:  530.1 = 50890 / 96
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -97.6000  -8.7770   0.3902   0.0000   7.7230 100.4000
```

```r
plot(tree_fit)
text(tree_fit, pretty=0)
```



```r
#cross-validate to check best size
cv.tree_fit <- cv.tree(tree_fit)

plot(cv.tree_fit$size, cv.tree_fit$dev, type = "b")
```



The regression tree selected only fpov as the primary splitting variable, highlighting its dominant role in predicting gas station counts. The model identified four distinct poverty thresholds, with higher poverty levels consistently associated with greater numbers of gas stations. While this tree provides a highly interpretable segmentation of the data, its simplicity came at the cost of reduced predictive accuracy, as other variables such as white population percentage, munp, shigh, and retax were not utilized in the tree splits.

Random Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.3
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

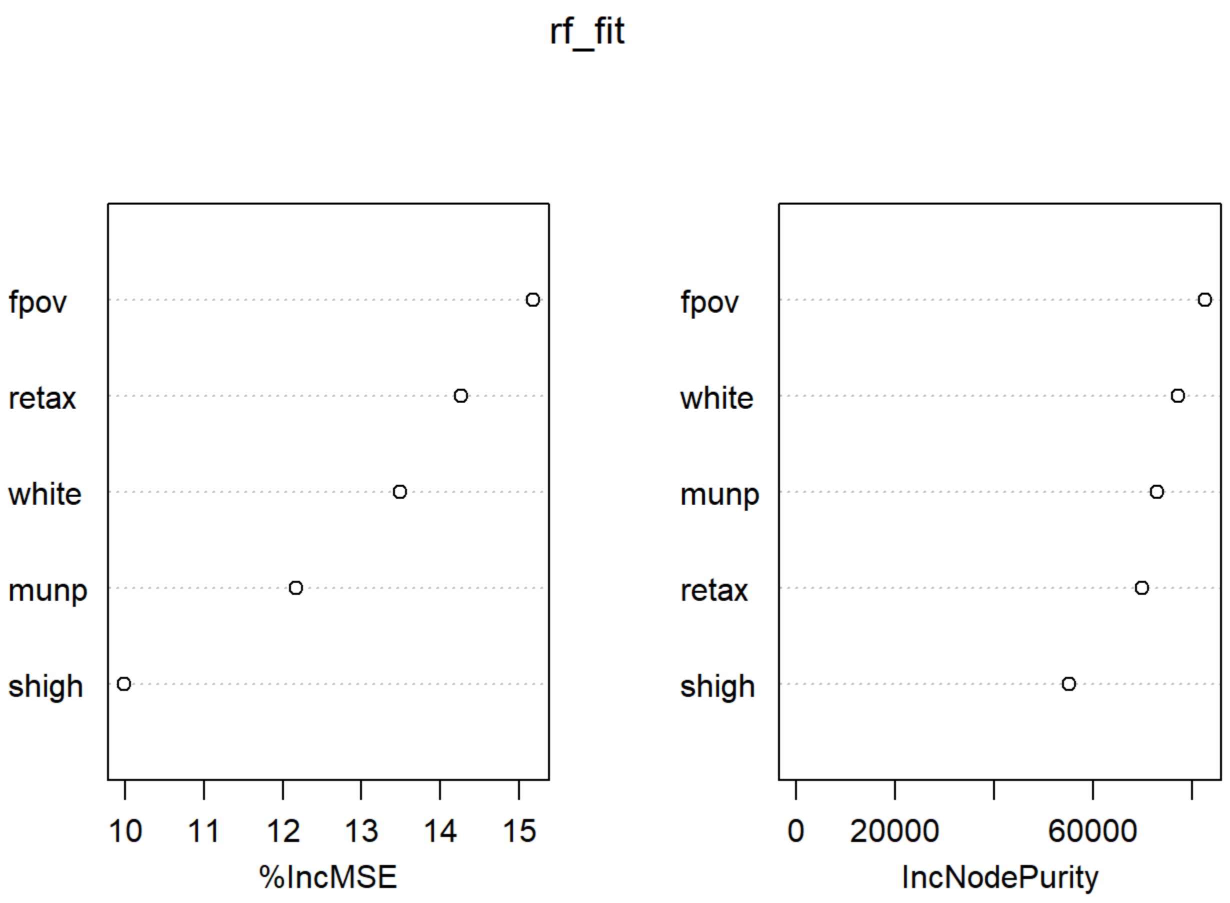
```
set.seed(123)

# Fit forest
rf_fit <- randomForest(gas ~ fpov + white + munp + shigh + retax, data = Gas_lasso, importance =
TRUE)

# View model
print(rf_fit)
```

```
##
## Call:
##  randomForest(formula = gas ~ fpov + white + munp + shigh + retax,      data = Gas_lasso, imp
ortance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 373.5165
##                     % Var explained: 89.91
```

```
# Variable Importance
importance(rf_fit)
```

```
##        %IncMSE IncNodePurity
## fpov  15.17458      82589.67
## white 13.49389      77139.90
## munp  12.16609      72976.53
## shigh  9.98303      55174.81
## retax 14.26182      69988.73
```

```
varImpPlot(rf_fit)
```



The Random Forest model provided strong predictive performance, explaining 89.9% of the variance in gas station counts. The variable importance measures indicated that fpov remained the dominant predictor, consistent with previous models. However, the forest also revealed the importance of variables such as real estate taxes (retax), white population percentage (white), and municipal road length (munp), which were not utilized in the single regression tree but contributed meaningfully across the ensemble of trees. This highlights the ability of Random Forests to capture complex interactions and non-linear effects that simpler models might overlook.

```
# Regression Tree RMSE
pred_tree <- predict(tree_fit, Gas_lasso)
sqrt(mean((Gas_lasso$gas - pred_tree)^2))
```

```
## [1] 22.55936
```

```
# Random Forest RMSE
pred_rf <- predict(rf_fit, Gas_lasso)
sqrt(mean((Gas_lasso$gas - pred_rf)^2))
```

```
## [1] 8.851447
```

```
# GAM RMSE
pred_gam <- predict(gam_fit2, Gas_lasso)
sqrt(mean((Gas_lasso$gas - pred_gam)^2))
```

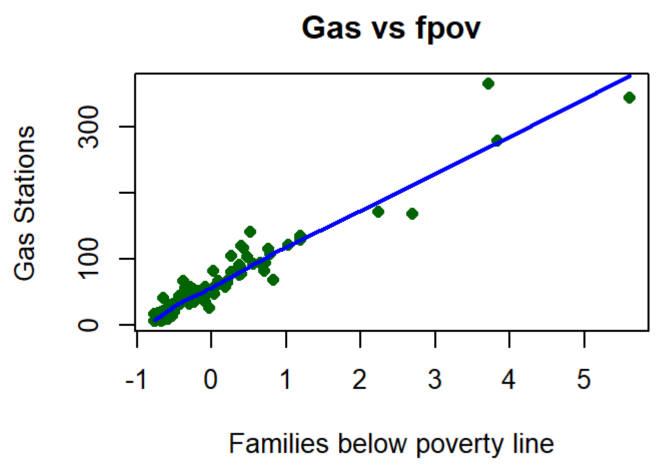
```
## [1] 8.235094
```

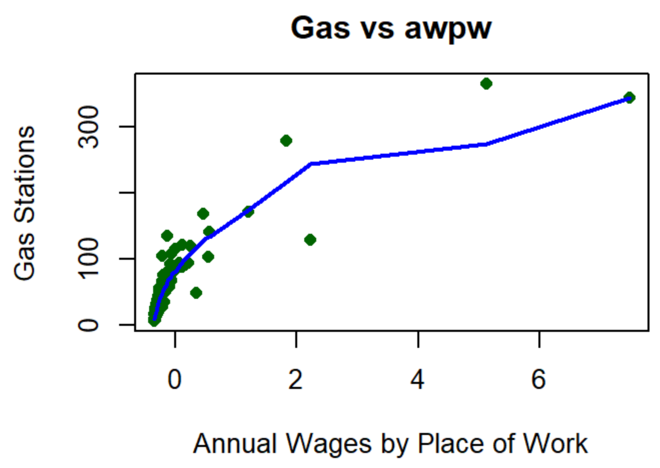
Among all models, the Generalized Additive Model (GAM) achieved the lowest Root Mean Squared Error (RMSE) of 8.24, slightly outperforming the Random Forest (RMSE = 8.85) and substantially outperforming the regression tree (RMSE = 22.56). While the Random Forest captured complex patterns through its ensemble structure, the GAM provided nearly equivalent predictive accuracy with the added benefit of interpretability and explicit non-linear term estimation.

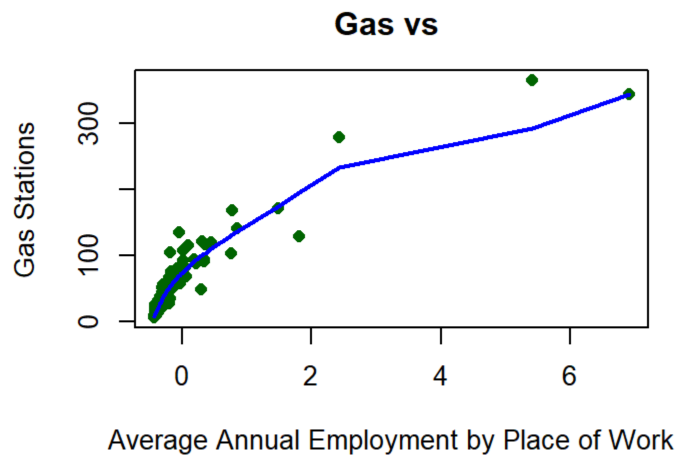
Using best susbet Selection Model

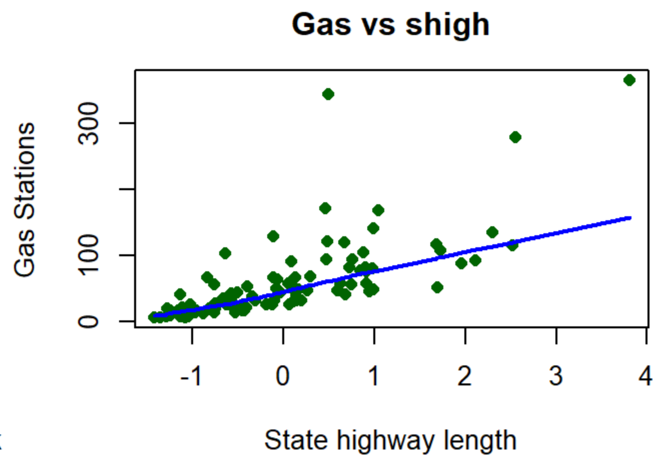
```
Gas_bs <- subset(Gas0, select = c("fpov", "awpw", "aaepw", "shigh", "gas"))
View(Gas_bs)
```

```
# Set up a 2x3 plotting grid
par(mfrow = c(2, 2), mar = c(4, 4, 3, 1))  # 2 rows, 3 columns, adjusted margins

# Scatterplot: gas vs fpov
plot(Gas_bs$fpov, Gas_bs$gas,
     main = "Gas vs fpov",
     xlab = "Families below poverty line",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_bs$fpov, Gas_bs$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs awpw
plot(Gas_bs$awpw, Gas_bs$gas,
     main = "Gas vs awpw",
     xlab = "Annual Wages by Place of Work",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_bs$awpw, Gas_bs$gas), col = "blue", lwd = 2)

# Scatterplot: gas vs aaepw
plot(Gas_bs$aaepw, Gas_bs$gas,
     main = "Gas vs ",
     xlab = "Average Annual Employment by Place of Work",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_bs$aaepw, Gas_bs$gas), col = "blue", lwd = 2)


# Scatterplot: gas vs shigh
plot(Gas_bs$shigh, Gas_bs$gas,
     main = "Gas vs shigh",
     xlab = "State highway length",
     ylab = "Gas Stations",
     pch = 16, col = "darkgreen")
lines(lowess(Gas_bs$shigh, Gas_bs$gas), col = "blue", lwd = 2)
```









Linear

```
lm.fit <- lm(gas ~ ., data = Gas_bs)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = gas ~ ., data = Gas_bs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.221  -6.065  -1.716   4.006  36.772
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  58.0400     0.9798  59.235  < 2e-16 ***
## fpov         20.3245     3.0361   6.694 1.51e-09 ***
## awpw        -78.0665    10.1594  -7.684 1.39e-11 ***
## aaepw       109.4808    11.6521   9.396 3.23e-15 ***
## shigh        10.7400     1.4639   7.336 7.34e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.798 on 95 degrees of freedom
## Multiple R-squared:  0.9754, Adjusted R-squared:  0.9743
## F-statistic: 940.5 on 4 and 95 DF,  p-value: < 2.2e-16
```

Cross-Validating Linear

```
library(boot)
library(leaps)
```

```
set.seed(6)
glm.fit <- glm(gas ~ ., data = Gas_bs)
cv.error <- cv.glm(Gas_bs, glm.fit, K = 10)
cv.error$delta
```

```
## [1] 111.3873 110.0477
```

Using 10-fold cross-validation on our linear model with all predictors, we estimate the test error to be approximately 110.05. This suggests that, on average, the model's predictions deviate from the true number of gas stations by around 10.5 stations (given that MSE units are squared). The similarity between the raw and bias-corrected error estimates indicates that the model is relatively stable and not heavily overfitting the training data.

Natural Splines for Best Subset Model

```
fit_ns <- lm(gas ~ ns(fpov, df = 4) + ns(awpw, df = 4) + ns(aaepw, df = 4) + ns(shigh, df = 4) ,
data = Gas_bs)
summary(fit_ns)
```

```
##
## Call:
## lm(formula = gas ~ ns(fpov, df = 4) + ns(awpw, df = 4) + ns(aaepw,
##     df = 4) + ns(shigh, df = 4), data = Gas_bs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.663  -5.230  -0.466   4.057  34.967
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5.107      4.187   1.220 0.226030
## ns(fpov, df = 4)1     8.549     11.304   0.756 0.451620
## ns(fpov, df = 4)2    60.224     16.102   3.740 0.000337 ***
## ns(fpov, df = 4)3    89.067     25.648   3.473 0.000821 ***
## ns(fpov, df = 4)4   100.207     27.668   3.622 0.000503 ***
## ns(awpw, df = 4)1    21.289     62.798   0.339 0.735463
## ns(awpw, df = 4)2  -152.419     88.535  -1.722 0.088875 .
## ns(awpw, df = 4)3  -251.870    126.847  -1.986 0.050377 .
## ns(awpw, df = 4)4  -460.671    126.889  -3.631 0.000488 ***
## ns(aaepw, df = 4)1    2.564     62.901   0.041 0.967579
## ns(aaepw, df = 4)2  247.741     88.168   2.810 0.006179 **
## ns(aaepw, df = 4)3  422.232    128.330   3.290 0.001471 **
## ns(aaepw, df = 4)4  681.334    133.661   5.097 2.13e-06 ***
## ns(shigh, df = 4)1    5.305      8.233   0.644 0.521170
## ns(shigh, df = 4)2   21.455      9.265   2.316 0.023040 *
## ns(shigh, df = 4)3   44.451     17.157   2.591 0.011309 *
## ns(shigh, df = 4)4   48.764     17.900   2.724 0.007856 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.605 on 83 degrees of freedom
## Multiple R-squared:  0.9793, Adjusted R-squared:  0.9753
## F-statistic: 245.7 on 16 and 83 DF,  p-value: < 2.2e-16
```

```
# Comparing AIC's
AIC(lm.fit, fit_ns)
```

	df <dbl>	**AIC** <dbl>
lm.fit	6	747.0992
fit_ns	18	753.6153

2 rows

Cross-Validated for Natural Splines

```r
set.seed(6)

folds <- sample(rep(1:10, length.out = nrow(Gas_bs)))

cv_error_ns <- rep(NA, 10)

for (k in 1:10) {

  train_data <- Gas_bs[folds != k, ]
  test_data  <- Gas_bs[folds == k, ]

  fit_ns_k <- lm(gas ~ ns(fpov, df=4) + ns(awpw, df=4) + ns(aaepw, df=4) + ns(shigh, df=4),
               data = train_data)

  pred <- predict(fit_ns_k, newdata = test_data)

  cv_error_ns[k] <- mean((test_data$gas - pred)^2)
}

cv_error_ns
```

```
##  [1]  69.81513 186.34043 247.34180  87.05284 290.03825 190.70114  62.79190
##  [8] 616.93958  75.65756  55.34125
```

```r
mean(cv_error_ns)
```

```
## [1] 188.202
```

```r
sd(cv_error_ns)
```

```
## [1] 172.558
```

GAM

```r
gam.fit <- gam(gas ~ s(fpov) + s(awpw) + s(aaepw) + s(shigh), data = Gas_bs)
summary(gam.fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## gas ~ s(fpov) + s(awpw) + s(aaepw) + s(shigh)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  58.0400     0.9482   61.21   <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(fpov)  2.240  2.818 17.78  < 2e-16 ***
## s(awpw)  1.000  1.000 44.98  < 2e-16 ***
## s(aaepw) 1.000  1.000 76.45  < 2e-16 ***
## s(shigh) 1.389  1.666 24.88 6.52e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.976   Deviance explained = 97.7%
## GCV = 96.297  Scale est. = 89.914    n = 100
```

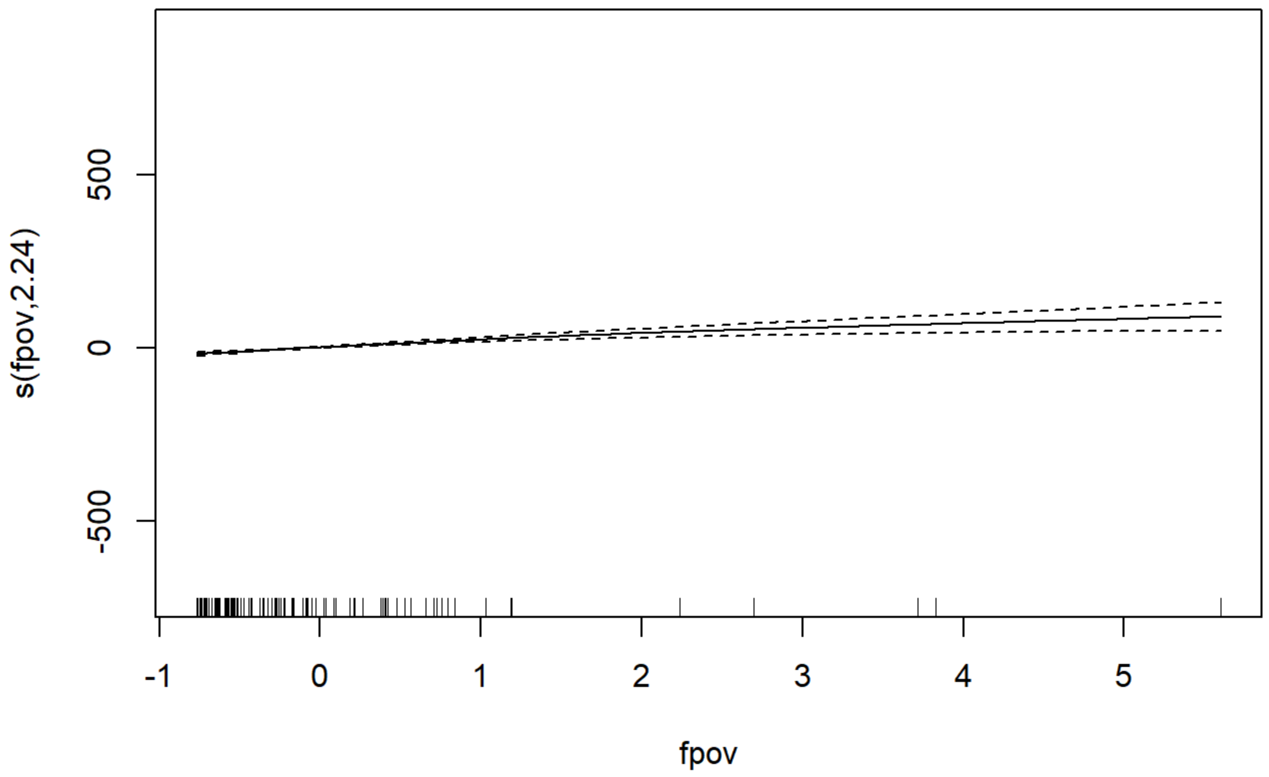
```
set.seed(6)

# Create 10 folds
folds <- sample(rep(1:10, length.out = nrow(Gas_bs)-1))

cv_error <- rep(NA, 10)

for (k in 1:10) {

  # Split data
  train_data <- Gas_bs[folds != k, ]
  test_data  <- Gas_bs[folds == k, ]

  # Fit GAM on training data
  gam_fit_k <- gam(gas ~ s(fpov) + s(awpw) + s(aaepw) + s(shigh),
                   data = train_data)

  # Predict on test data
  pred <- predict(gam_fit_k, newdata = test_data)

  # Compute Test Error (MSE)
  cv_error[k] <- mean((test_data$gas - pred)^2)
}

# View all CV errors
cv_error
```

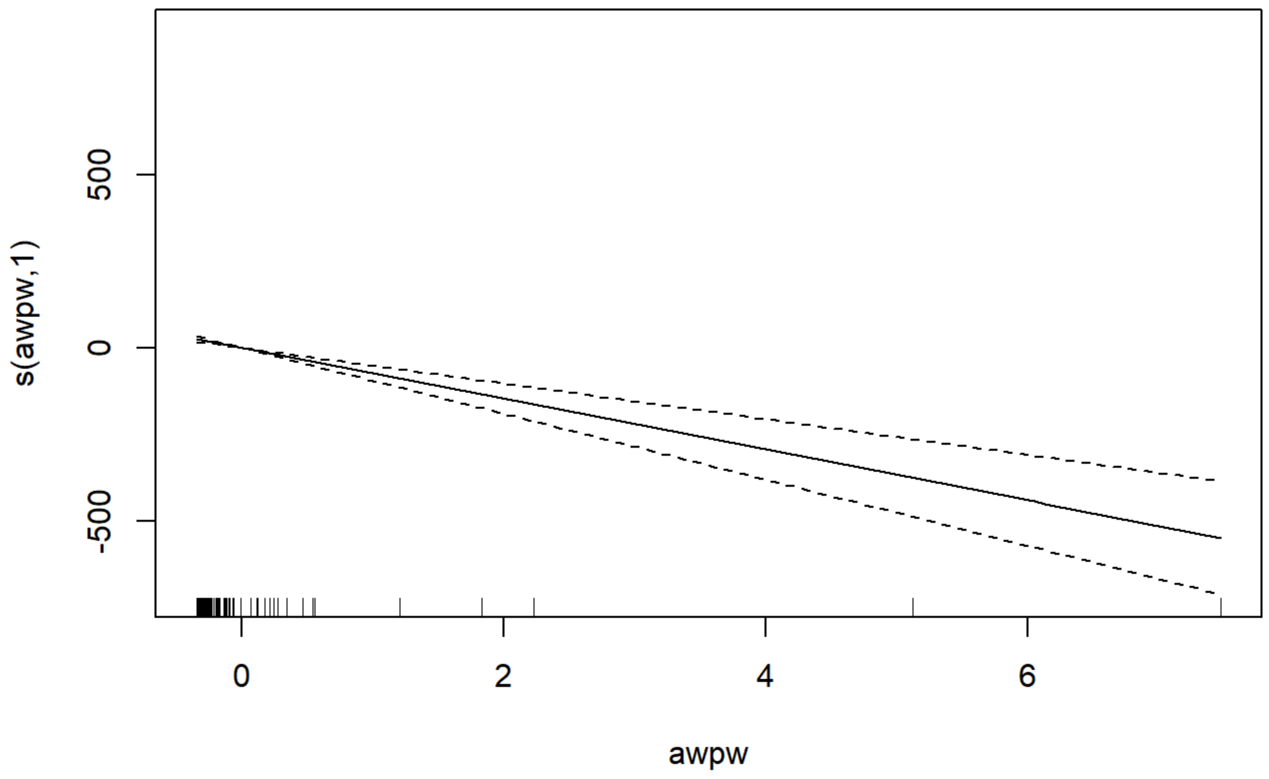
```
##  [1]   6.965432 250.998277 140.604411  42.934892 305.941641  36.281838
##  [7] 128.678395 130.494640  84.661758  29.956528
```

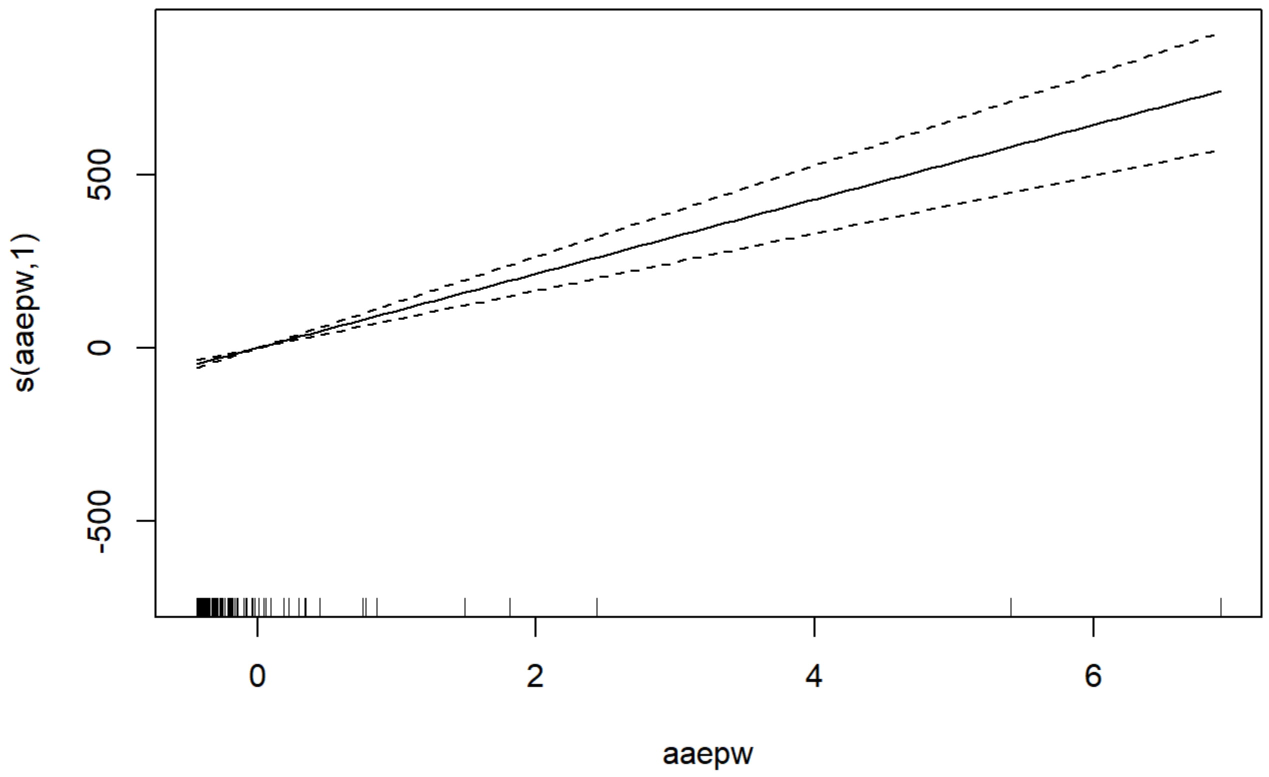
```
# Average Test Error across folds
mean(cv_error)
```

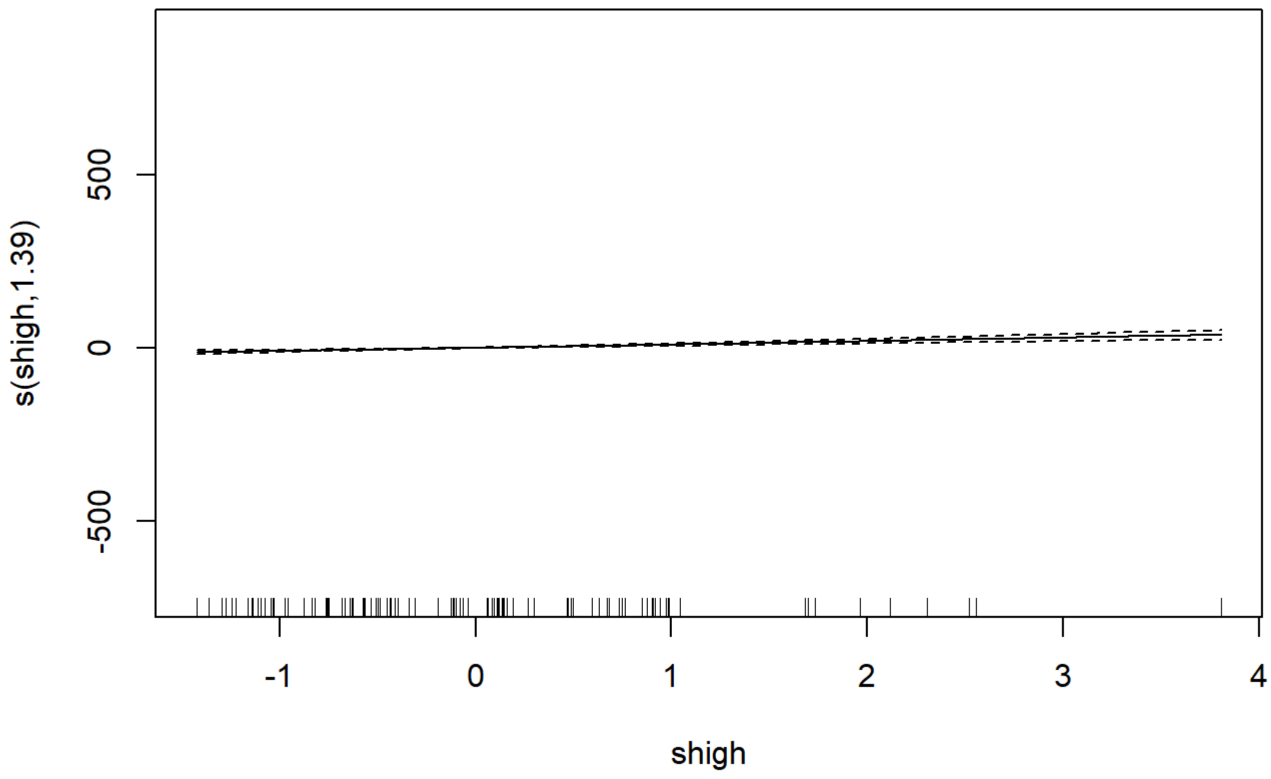
```
## [1] 115.7518
```

```
# Automatic plots for each variable
plot(gam.fit, se = TRUE, shade = TRUE)
```


s(fpov,2.24)
500
0
-500
-1
0
1
2
3
4
5
fpov


s(awpw,1)
500
0
-500
0
2
4
6
awpw


s(aaepw,1)
500
0
-500
0
2
4
6
aaepw


s(shigh,1.39)
500
0
-500
-1
0
1
2
3
4
shigh

```
# Check AIC
AIC(lm.fit, fit_ns, gam.fit)
```

	df <dbl>	AIC <dbl>
lm.fit	6.00000	747.0992
fit_ns	18.00000	753.6153
gam.fit	7.62836	742.0719

3 rows

```
# Check adjusted R-squared
summary(gam.fit)$r.sq
```

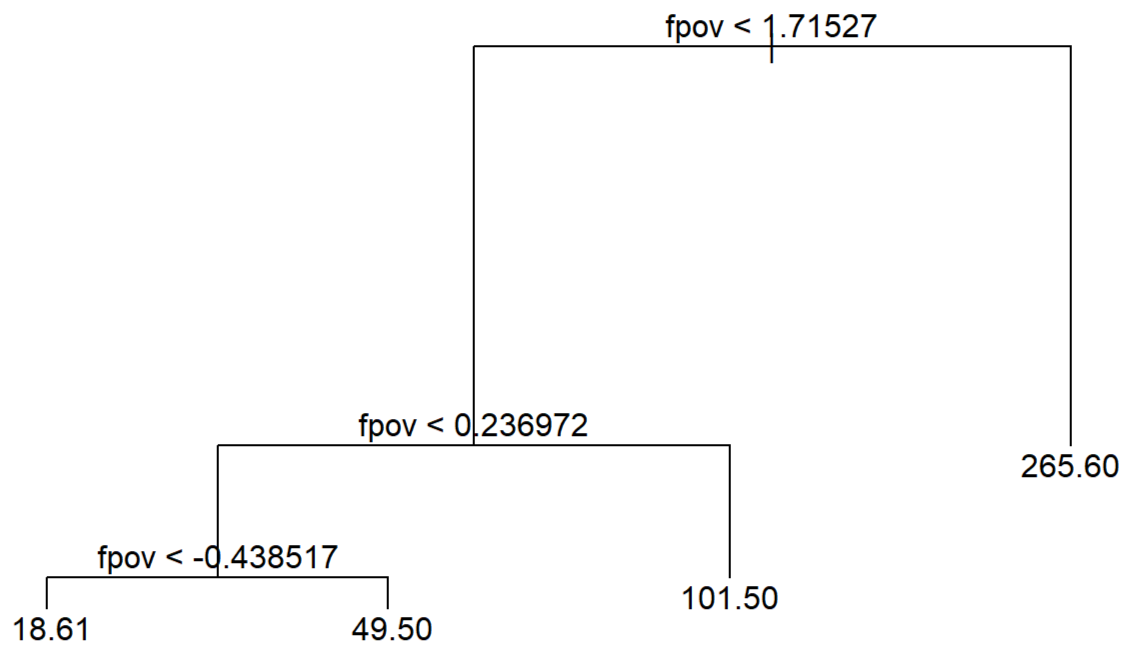
```
## [1] 0.9759612
```

Regression Trees

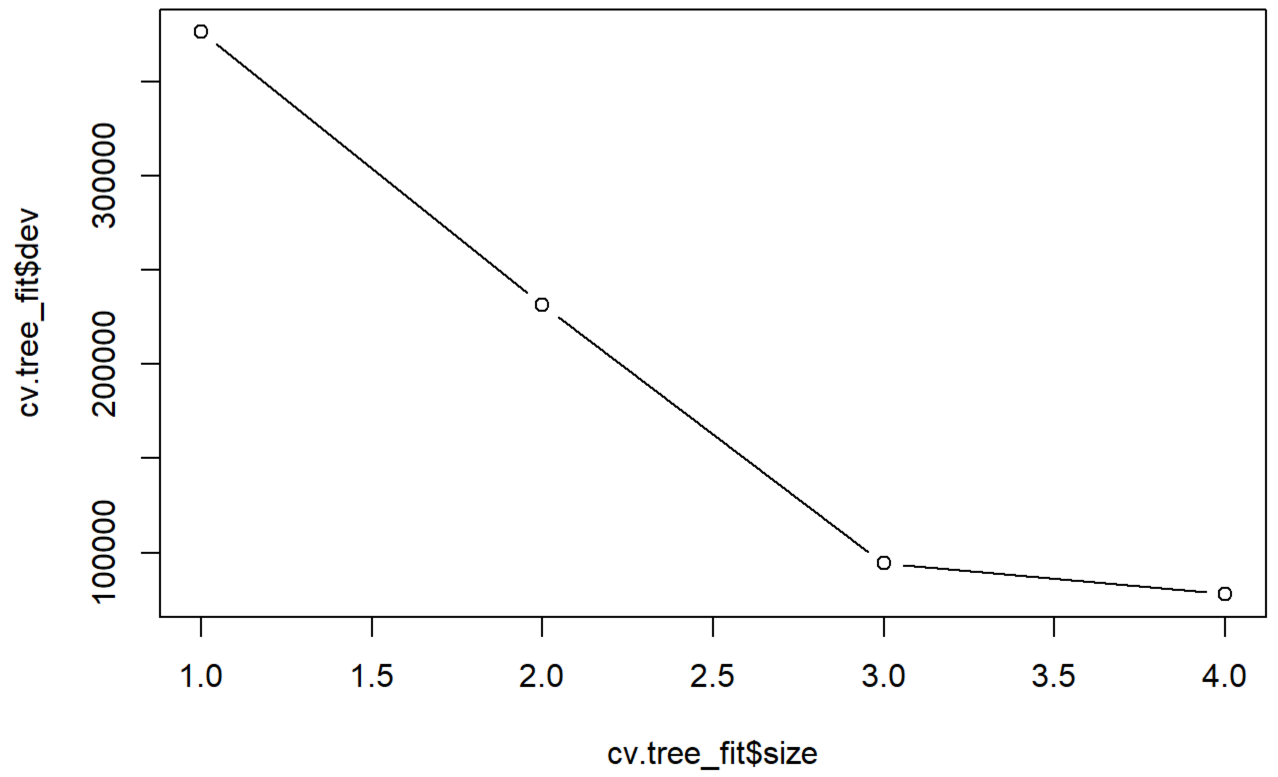
```
# Fit tree
tree_fit <- tree(gas ~ fpov + awpw + aaepw + shigh, data = Gas_bs)
# Summary
summary(tree_fit)
```

```
##
## Regression tree:
## tree(formula = gas ~ fpov + awpw + aaepw + shigh, data = Gas_bs)
## Variables actually used in tree construction:
## [1] "fpov"
## Number of terminal nodes:  4
## Residual mean deviance:  530.1 = 50890 / 96
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -97.6000  -8.7770   0.3902   0.0000   7.7230 100.4000
```

```
plot(tree_fit)
text(tree_fit, pretty=0)
```



```
#cross-validate to check best size
cv.tree_fit <- cv.tree(tree_fit)

plot(cv.tree_fit$size, cv.tree_fit$dev, type = "b")
```



#

Random Forest

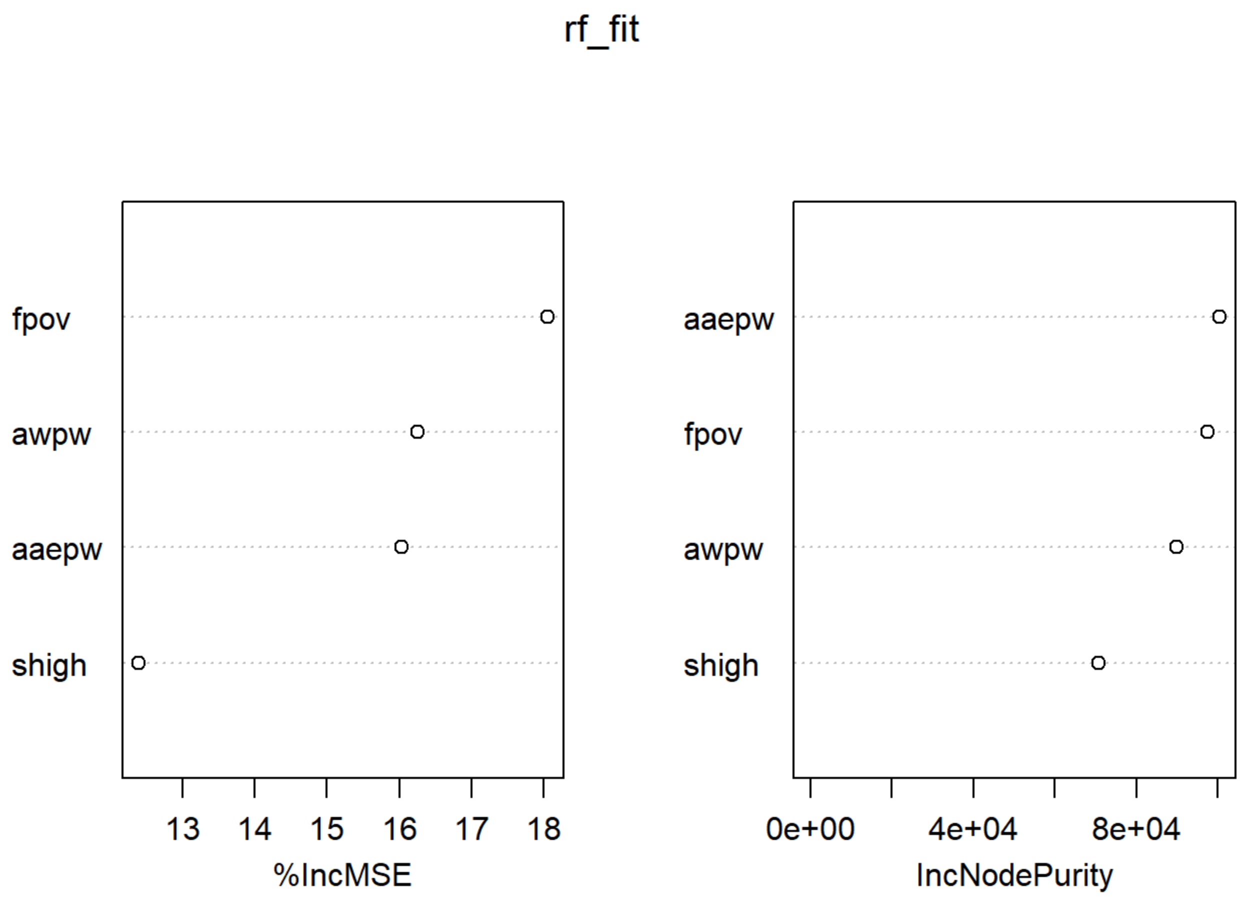
```
set.seed(123)

# Fit forest
rf_fit <- randomForest(gas ~ fpov + awpw + aaepw + shigh, data = Gas_bs, importance = TRUE)

# View model
print(rf_fit)
```

```
##
## Call:
##  randomForest(formula = gas ~ fpov + awpw + aaepw + shigh, data = Gas_bs,      importance = T
RUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 402.93
##                     % Var explained: 89.12
```

```
importance(rf_fit)
```

```
##        %IncMSE IncNodePurity
## fpov  18.04976      97383.15
## awpw  16.25120      89883.47
## aaepw 16.03863     100346.15
## shigh 12.39299      70662.07
```

```
varImpPlot(rf_fit)
```



```
# Regression Tree RMSE
pred_tree <- predict(tree_fit, Gas_bs)
sqrt(mean((Gas_bs$gas - pred_tree)^2))
```

```
## [1] 22.55936
```

```
# Random Forest RMSE
pred_rf <- predict(rf_fit, Gas_bs)
sqrt(mean((Gas_bs$gas - pred_rf)^2))
```

```
## [1] 9.046172
```

```
# GAM RMSE
pred_gam <- predict(gam.fit, Gas_bs)
sqrt(mean((Gas_bs$gas - pred_gam)^2))
```

```
## [1] 9.162667
```