Informe Pisa

Sergio Casares

8/11/2020

1.Introducción al trabajo

El trabajo consiste en la realización de modelos predictivos del dataset utilizado, donde se tratará de predecir la nota media de cada país en función del conjunto de variables que forman el modelo, para ello se utilizarán modelos splines y GAM, además de realizar la debida Validación Cruzada.

El conjunto de datos se ha construido utilizando la puntuación media en Ciencias por país del Programa para la Evaluación Internacional de Estudiantes (PISA) 2006, junto con el GNI per cápita (paridad del poder adquisitivo, dólares de 2005), el índice educativo, el índice de salud y el índice de desarrollo humano de la ONU (HDI).

Las variables clave son las siguientes:

Overall Science Score (average score for 15 year olds) Interest in science Support for scientific inquiry Income Index Health Index Education Index Human Development Index (composed of the Income index, Health Index, and Education Index)

2.Importación de librerías y Dataset

```
library(readr)
library(glmnet)
library(tidyverse)
library(fBasics)
library(car)
library(dplyr)
library(ggplot2)
library(knitr)
library(MASS)
library(corrplot)
library(PerformanceAnalytics)
library(gvlma)
library(tinytex)
library(devtools)
library(rsample)
library(tidyr)
library(broom)
library(flextable)
library(mgcv)
library(reshape2)
library(patchwork)
```

```r
library(corrplot)
library(visreg) #VISUALIZACION DE MODELO GAM
library(splines) #SPLINES
library(Metrics) #SPLIN DE REGRESION
library(MLmetrics) #SPLIN DE REGRESION
library(graphics)
library(tidyr)

pisa <- read_csv("pisasci2006.csv",
                                 col_types = cols(Overall = col_number(),
                                                  Issues = col_number(), Explain = col_number(),
                                                  Evidence = col_number(), Interest = col_number(),
                                                  Support = col_number(), Income = col_number(),
                                                  Health = col_number(), Edu = col_number(),
                                                  HDI = col_number()))

pisa <- drop_na(pisa)
dim(pisa)
```

```
## [1] 52 11
```

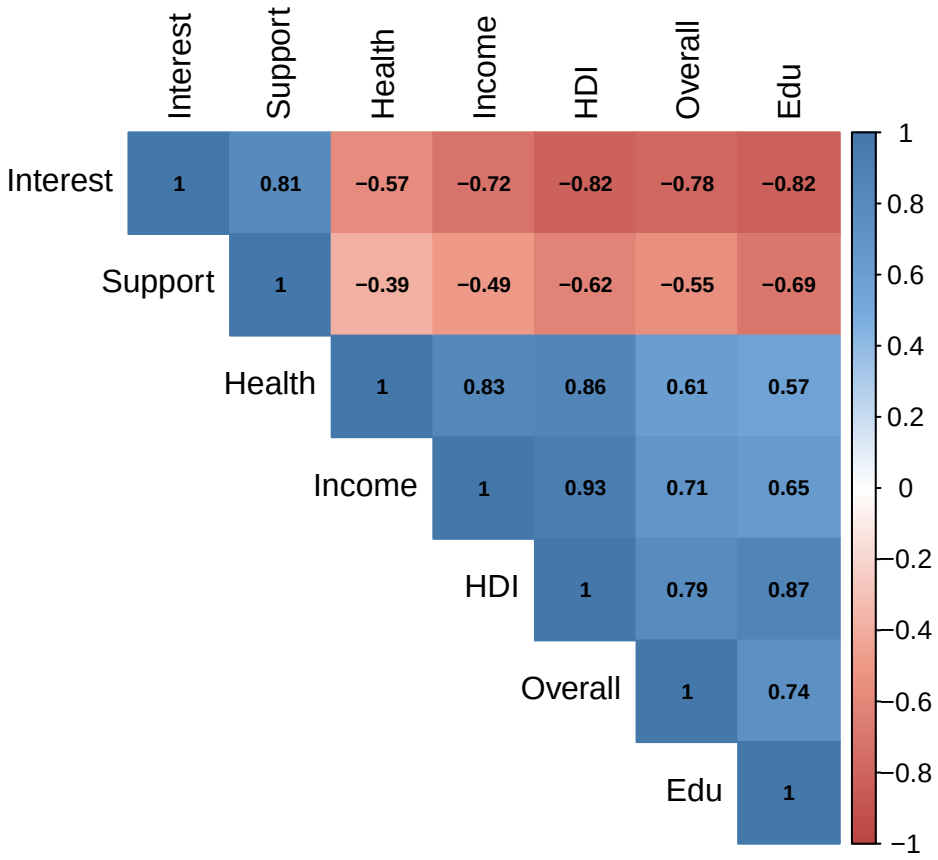
```r
View(pisa)
```

2.1 Análisis Exploratorio de Datos (EDA)

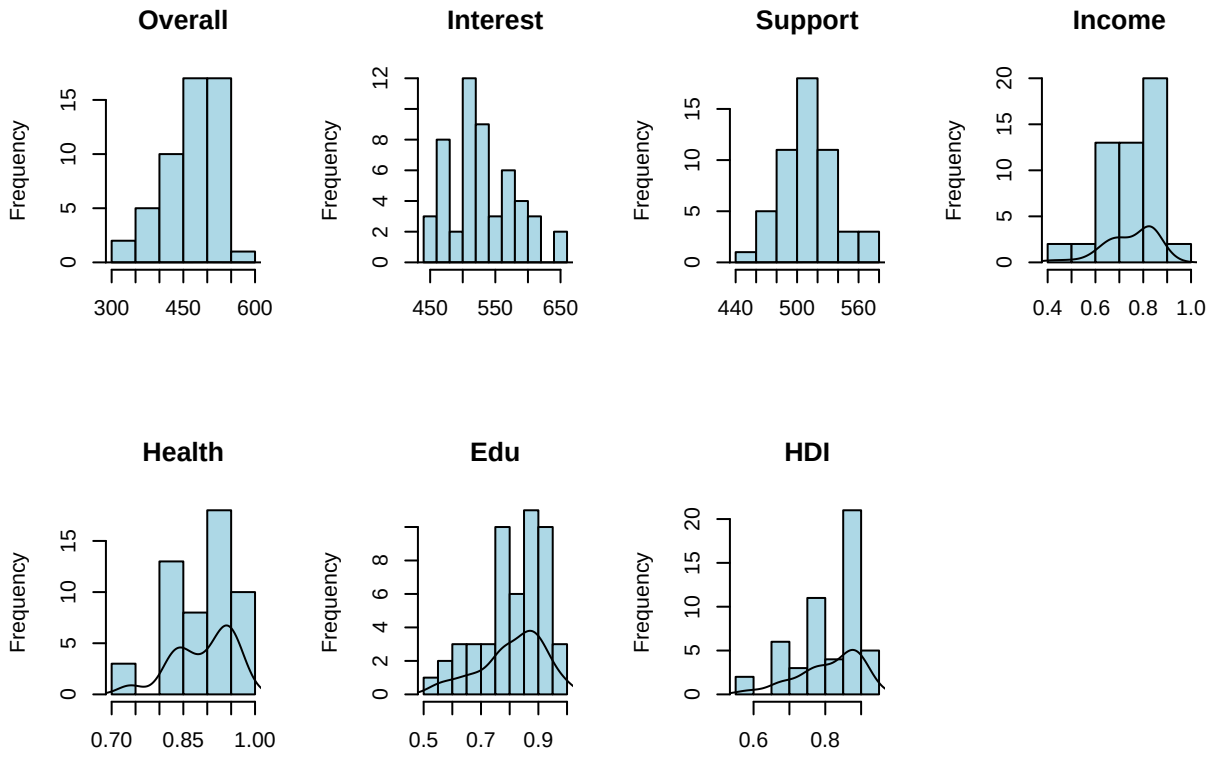
En el mapa de correlaciones se puede observar una alta correlación entre la variable dependiente (Overall) con las variables de HDI, Income, Edu e Interest (de manera negativa), lo que puede indicar que sean las variables que más afecten a Overall, aunque entre ellas presentan también alta correlación , lo que podría indicar cierta multicolinealidad. Por el contrario, las variables de Support y Health presentan una correlación más baja con la variable dependiente.

En el gráfico de correlaciones se puede confirmar lo que se ha explicado en el párrafo anterior, la distibución de las observaciones presentan cierta oblicuidad (skewness) en la variable Overall y en aquellas otras que tienen una alta correlación (caso interesanto de la variable Interest, que su distribución presenta una skewness simétrica a la de la variable Overall). Por el otro lado, Support y Health presntan variables más normalizadas, alejadas de la distribución que presenta Overall.

```r
pisa <- dplyr::select(pisa, -Country, -Issues, -Explain, -Evidence)
par(mfrow=c(1,1))

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(cor(pisa), method = "color", col = col(200),
         type = "upper", order = "hclust", number.cex = .7,
         addCoef.col = "black", # Add coefficient of correlation
         tl.col = "black", tl.srt = 90, # Text label color and rotation
         # Combine with significance
         sig.level = 0.01, insig = "blank",
         # hide correlation coefficient on the principal diagonal
         diag = TRUE)
```




```
par(mfrow=c(2,4))

pisa <- as.data.frame(pisa)

for (p in 1:7) {
  hist(pisa[,p],main=colnames(pisa)[p],xlab="",col="lightblue")
  lines(density(pisa[,p]))
}
```



3.División dataset Training vs Testing

Se divide la muestra en dos parte, un 70€ para entrenar el modelo y el 30% restante para testearlo.

```
set.seed(1998)

pisa_split <- initial_split(pisa, prop = .7)

dim(pisa)
```

```
## [1] 52  7
```

```
#Practica (70%)
pisa_training_data <- training(pisa_split)
dim(pisa_training_data)
```

```
## [1] 37  7
```

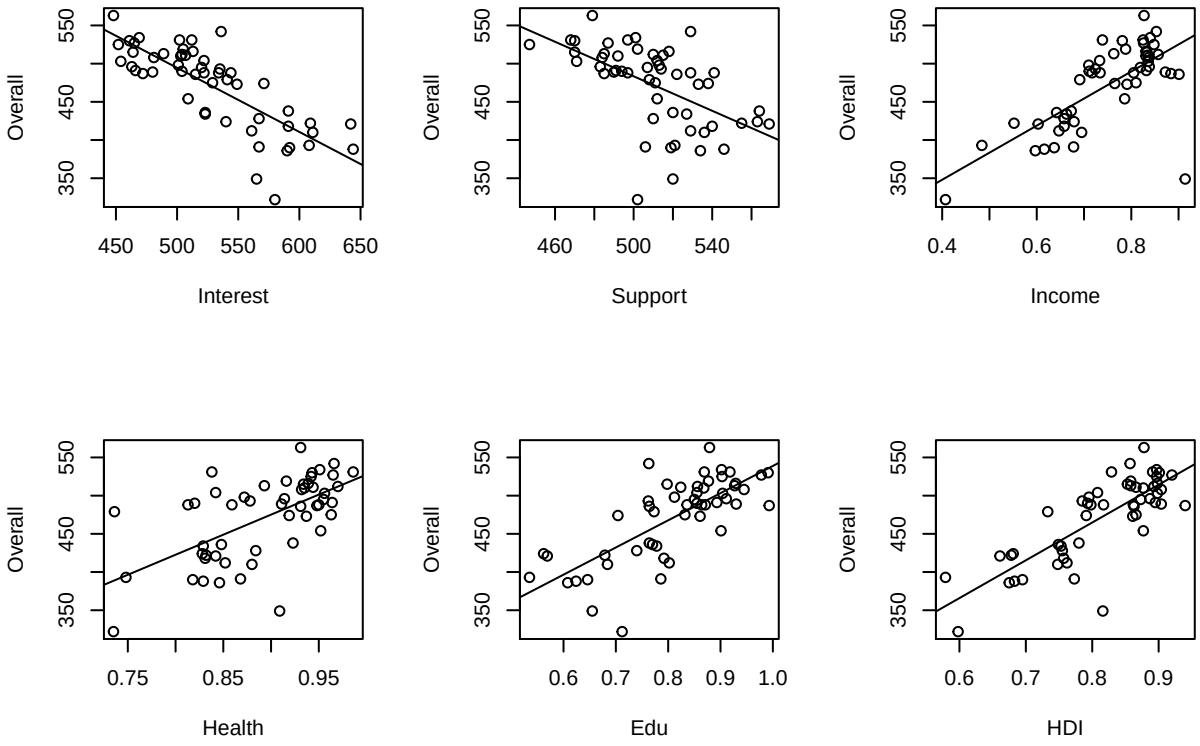
```
#Test (30%)
pisa_testing_data <- testing(pisa_split)
dim(pisa_testing_data)
```

```
## [1] 15  7
```

4.Regresión Lineal

Antes de comenzar con la modelización del dataset, creo conveniente la visualización de los datos de la variable dependiente (Overall) en relación con las demás variables independientes de manera individual por medio de la función plot junto una linea que representa la regresión lineal de esas dos variables.

```
par(mfrow=c(2,3))

g_1 <- plot(Overall ~ Interest, data=pisa) +
  abline(lm(Overall ~ Interest, data = pisa))
g_2 <- plot(Overall ~ Support, data=pisa) +
  abline(lm(Overall ~ Support, data = pisa))
g_3 <- plot(Overall ~ Income, data=pisa) +
  abline(lm(Overall ~ Income, data = pisa))
g_4 <- plot(Overall ~ Health, data=pisa) +
  abline(lm(Overall ~ Health, data = pisa))
g_5 <- plot(Overall ~ Edu, data=pisa) +
  abline(lm(Overall ~ Edu, data = pisa))
g_6 <- plot(Overall ~ HDI, data=pisa) +
  abline(lm(Overall ~ HDI, data = pisa))
```



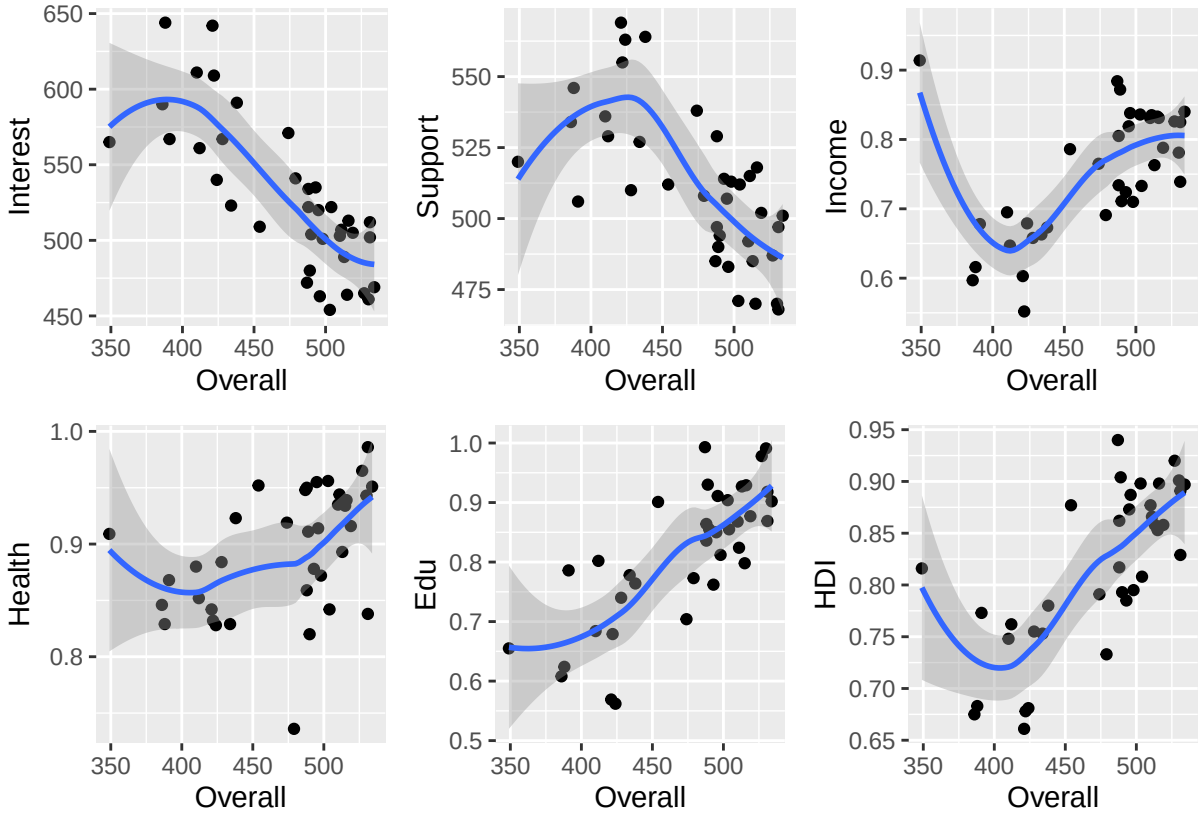
4.Modelos Splines

4.1 Splines Suavizados

Por medio de la función ggplot y stat_smooth(). La función stat_smooth() tiene como objetivO crear una función g(x) que se ajuste bien a las observaciones, es decir, que minimice la suma de los residuos al cuadrado.

Como se puede observar, el modelo de los Splines Suavizados ajusta el modelo mucho mejor que la regresión lineal.

```
g1 <- ggplot(pisa_training_data, aes(Overall, Interest) ) +
  geom_point() +
  stat_smooth()
g2 <- ggplot(pisa_training_data, aes(Overall, Support) ) +
  geom_point() +
  stat_smooth()
g3 <- ggplot(pisa_training_data, aes(Overall, Income) ) +
  geom_point() +
  stat_smooth()
g4 <- ggplot(pisa_training_data, aes(Overall, Health) ) +
  geom_point() +
  stat_smooth()
g5 <- ggplot(pisa_training_data, aes(Overall, Edu) ) +
  geom_point() +
  stat_smooth()
g6 <- ggplot(pisa_training_data, aes(Overall, HDI) ) +
  geom_point() +
  stat_smooth()

(g1 + g2 + g3)/
  (g4 + g5 + g6)
```



4.2 Splines de Regresión

Los Regression splines tratan de una extensión de la regresión polinómica y de las step functions que consigue una mayor flexibilidad. Consiste en dividir el rango del predictor X en K subintervalos. Para cada una de las nuevas regiones se ajusta una función polinómica, introduciendo una serie de restricciones que hacen que los extremos de cada función se aproximen a los de las funciones de las regiones colindantes.

Las fases por las que incurriremos en este apartado son:

- Determinación de los knots (o partes en las que se divide el rango de X)
- Creación de los modelos de la variable dependiente con respecto de cada variable independiente
- Visualización de los modelos
- Calculo de las predicciones (Creación del objeto de prueba y calculo de RMSE Y R2)

```
knots <- quantile(pisa_training_data$Overall, p = c(0.25, 0.5, 0.75))

#MODELIZO y genero el B-Spline para un Spline polinomial
model_Interest <- lm (Overall ~ bs(Interest, knots = knots), data = pisa_training_data)
model_Support <- lm (Overall ~ bs(Support, knots = knots), data = pisa_training_data)
model_Income <- lm (Overall ~ bs(Income, knots = knots), data = pisa_training_data)
model_Health <- lm (Overall ~ bs(Health, knots = knots), data = pisa_training_data)
model_Edu <- lm (Overall ~ bs(Edu, knots = knots), data = pisa_training_data)
model_HDI <- lm (Overall ~ bs(HDI, knots = knots), data = pisa_training_data)
```

Visualizamos de manera independiente cada variable independiente con la variable Overall

```
gg_1 <- ggplot(pisa_training_data, aes(Overall, Interest) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 4))
gg_2 <- ggplot(pisa_training_data, aes(Overall, Support) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 4))
gg_3 <- ggplot(pisa_training_data, aes(Overall, Income) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 4))
gg_4 <- ggplot(pisa_training_data, aes(Overall, Health) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 4))
gg_5 <- ggplot(pisa_training_data, aes(Overall, Edu) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 4))
gg_6 <- ggplot(pisa_training_data, aes(Overall, HDI) ) +
  geom_point() +
  stat_smooth(method = lm, formula = y ~ splines::bs(x, df = 4))
```

Creamos los objetos cuya finalidad es la de predecir los valores del conjunto de Testing

```
#realizo las predicciones
options(warn=-1)

predictions_Interest <- model_Interest %>% predict(pisa_testing_data)
predictions_Support <- model_Support %>% predict(pisa_testing_data)
predictions_Income <- model_Income %>% predict(pisa_testing_data)
predictions_Health <- model_Health %>% predict(pisa_testing_data)
predictions_Edu <- model_Edu %>% predict(pisa_testing_data)
predictions_HDI <- model_HDI %>% predict(pisa_testing_data)

options(warn=1)
```

Para cada variable, calculamos el RMSE y R2

```
#Calculo la rmse y el r2

data.frame(
  RMSE = rmse(predictions_Interest, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_Interest, pisa_testing_data$Overall)
)
```

```
##       RMSE        R2
## 1 37.63669 0.6408523
```

```
data.frame(
  RMSE = rmse(predictions_Support, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_Support, pisa_testing_data$Overall)
)
```

```
##       RMSE        R2
## 1 159.6737 -5.464232
```

```r
data.frame(
  RMSE = rmse(predictions_Income, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_Income, pisa_testing_data$Overall)
)
```

```
##       RMSE        R2
## 1 164.5365 -5.863959
```

```r
data.frame(
  RMSE = rmse(predictions_Health, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_Health, pisa_testing_data$Overall)
)
```

```
##       RMSE        R2
## 1 56.29787 0.1964103
```

```r
data.frame(
  RMSE = rmse(predictions_Edu, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_Edu, pisa_testing_data$Overall)
)
```

```
##      RMSE        R2
## 1 44.1255 0.5063381
```

```r
data.frame(
  RMSE = rmse(predictions_HDI, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_HDI, pisa_testing_data$Overall)
)
```

```
##       RMSE        R2
## 1 54.82426 0.2379281
```

5.Modelo GAM

5.1. Creación y Summary de los modelos

```r
linear_model <- lm(Overall ~ Interest + Support + Income + Health + Edu + HDI, data=pisa_training_data)
summary(linear_model)
```

```
##
## Call:
## lm(formula = Overall ~ Interest + Support + Income + Health +
##     Edu + HDI, data = pisa_training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.867 -21.304   4.154  16.552  48.993
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   556.6702   228.9101   2.432   0.0212 *
## Interest       -0.5233     0.2188  -2.391   0.0233 *
## Support         0.2278     0.3379   0.674   0.5053
## Income      -2744.4269  1045.7926  -2.624   0.0135 *
## Health      -2405.2367   984.1650  -2.444   0.0206 *
## Edu         -2536.6275  1063.6038  -2.385   0.0236 *
## HDI          7783.7850  3048.6419   2.553   0.0160 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.14 on 30 degrees of freedom
## Multiple R-squared:  0.734,  Adjusted R-squared:  0.6808
## F-statistic:  13.8 on 6 and 30 DF,  p-value: 1.812e-07
```

```
gam_model_1 <- gam(Overall ~ s(Income) + s(Health)  + s(Edu) + s(HDI), data=pisa_training_data)
summary(gam_model_1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Health) + s(Edu) + s(HDI)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  472.378      3.091   152.8   <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(Income) 8.010  8.668 7.608 9.76e-06 ***
## s(Health) 2.098  2.644 1.774   0.1961
## s(Edu)    1.000  1.000 3.421   0.0766 .
## s(HDI)    1.000  1.000 3.098   0.0910 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.858   Deviance explained = 90.5%
## GCV = 547.29  Scale est. = 353.4     n = 37
```

```
gam_model_2 <- gam(Overall ~ s(Income)  + s(Edu) + s(HDI), data=pisa_training_data)
summary(gam_model_2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Edu) + s(HDI)
```

```
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  472.378      3.166   149.2   <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(Income) 6.049  7.140 6.561 0.000103 ***
## s(Edu)    4.179  5.059 2.131 0.093370 .
## s(HDI)    1.000  1.000 2.020 0.167564
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.85   Deviance explained = 89.7%
## GCV = 553.98  Scale est. = 370.89    n = 37
```

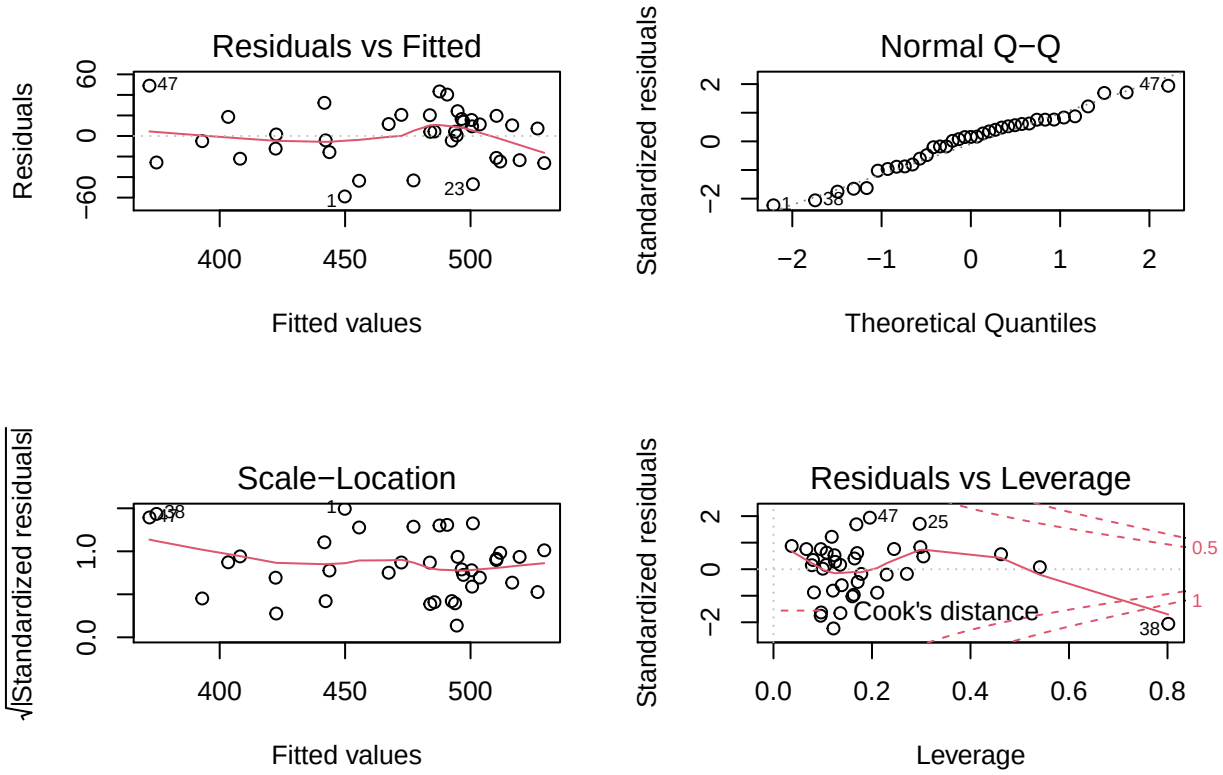
```
gam_model_3 <- gam(Overall ~ s(Income) + s(Edu) , data=pisa_training_data)
summary(gam_model_3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Edu)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  472.378      3.391   139.3   <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(Income) 6.182  7.295 6.796 3.76e-05 ***
## s(Edu)    1.000  1.000 9.691  0.00407 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.828   Deviance explained = 86.3%
## GCV =  546.4  Scale est. = 425.57    n = 37
```

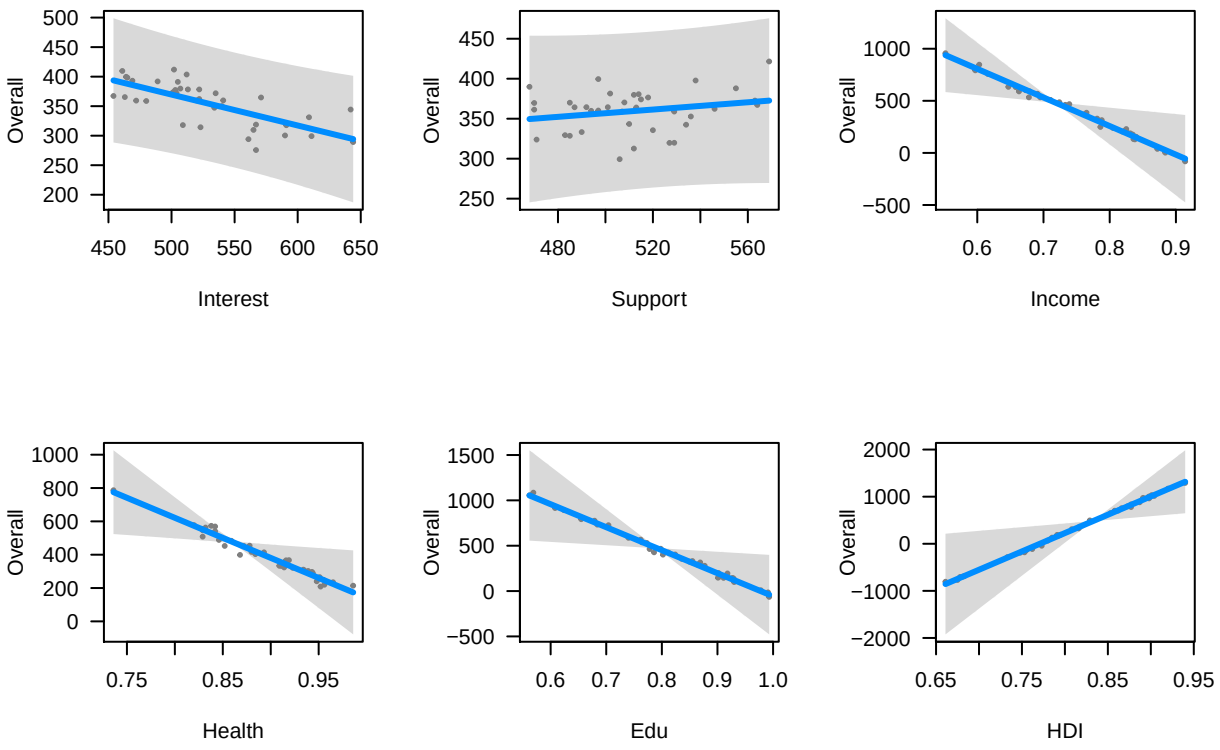
5.2. Visualización de los modelos

Model 1: Overall ~ Interest + Support + Income + Health + Edu + HDI

```
par(mfrow=c(2,2))
plot(linear_model)
```

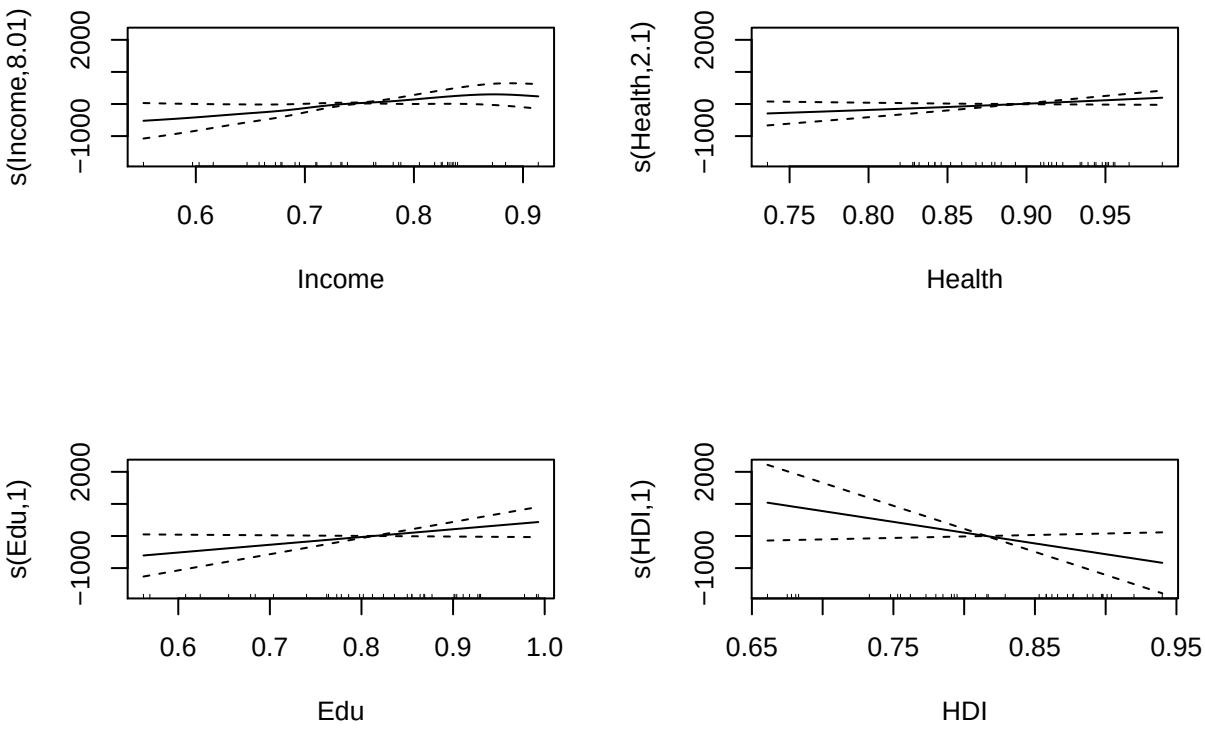



```
par(mfrow=c(2,3))
visreg(linear_model)
```



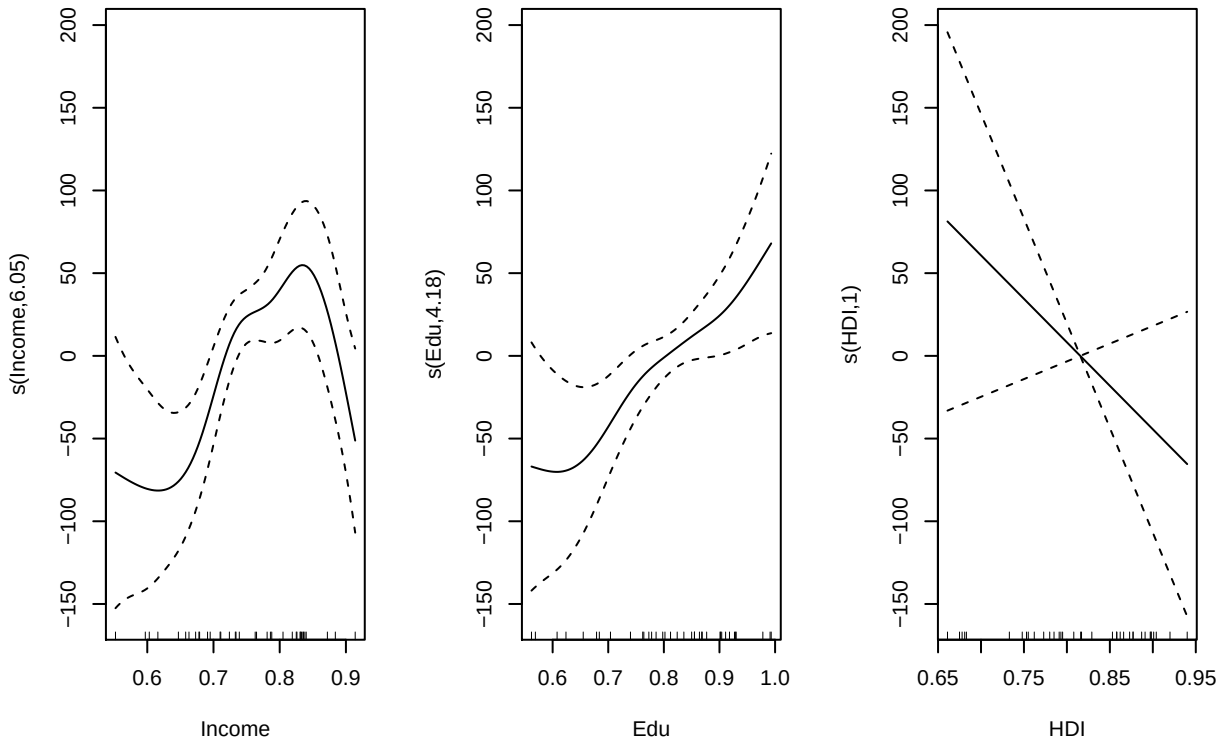
Model 2: Overall ~ s(Income) + s(Health) + s(Edu) + s(HDI)

```
par(mfrow = c(2,2))
plot(gam_model_1)
```



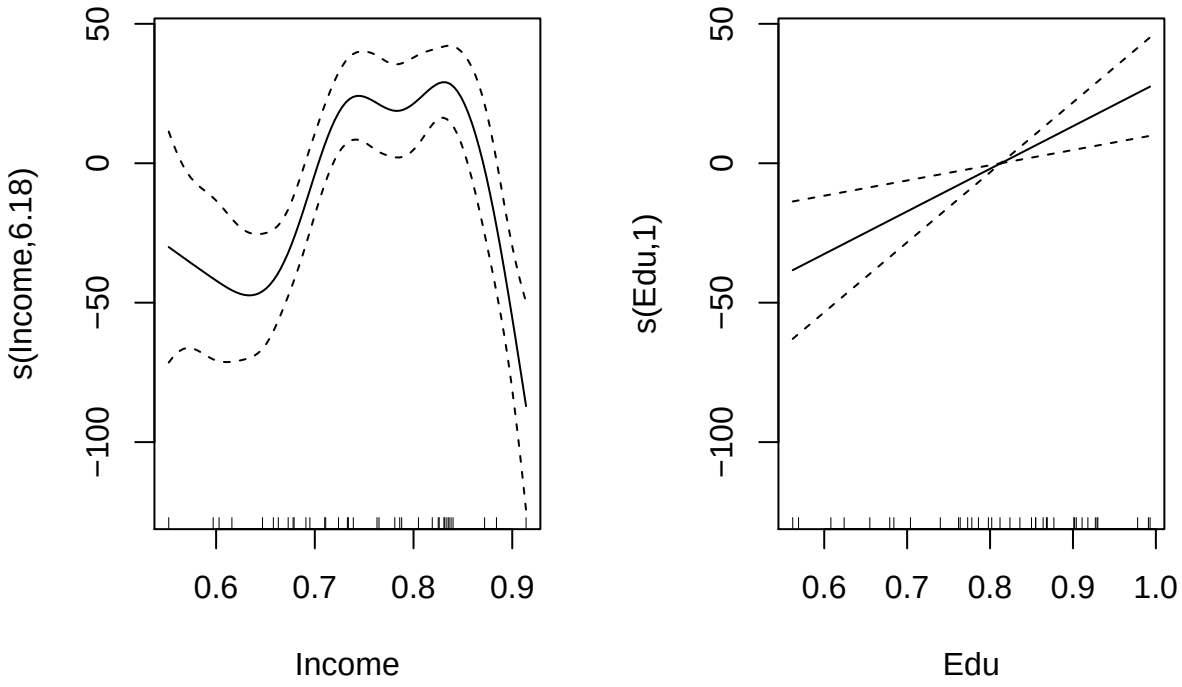
Model 3: Overall ~ s(Income) + s(Edu) + s(HDI)

```
par(mfrow=c(1,3))
plot(gam_model_2)
```

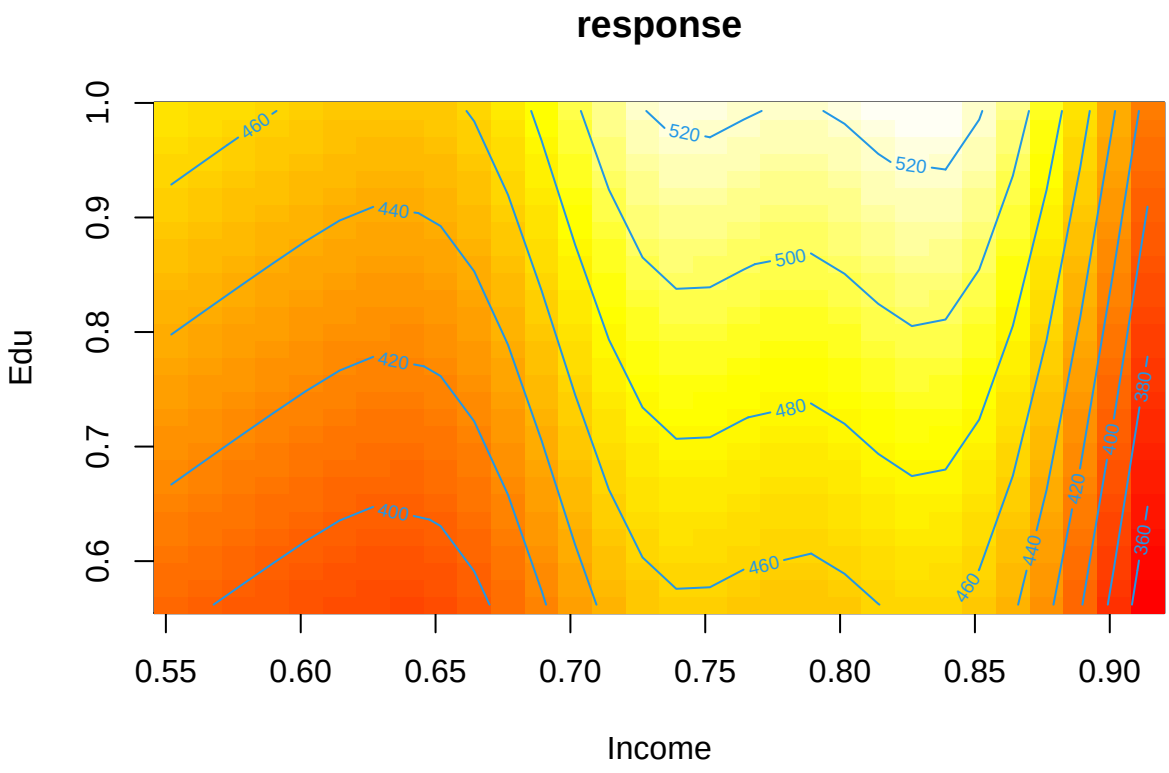


Model 4: Overall ~ s(Income) + s(Edu)

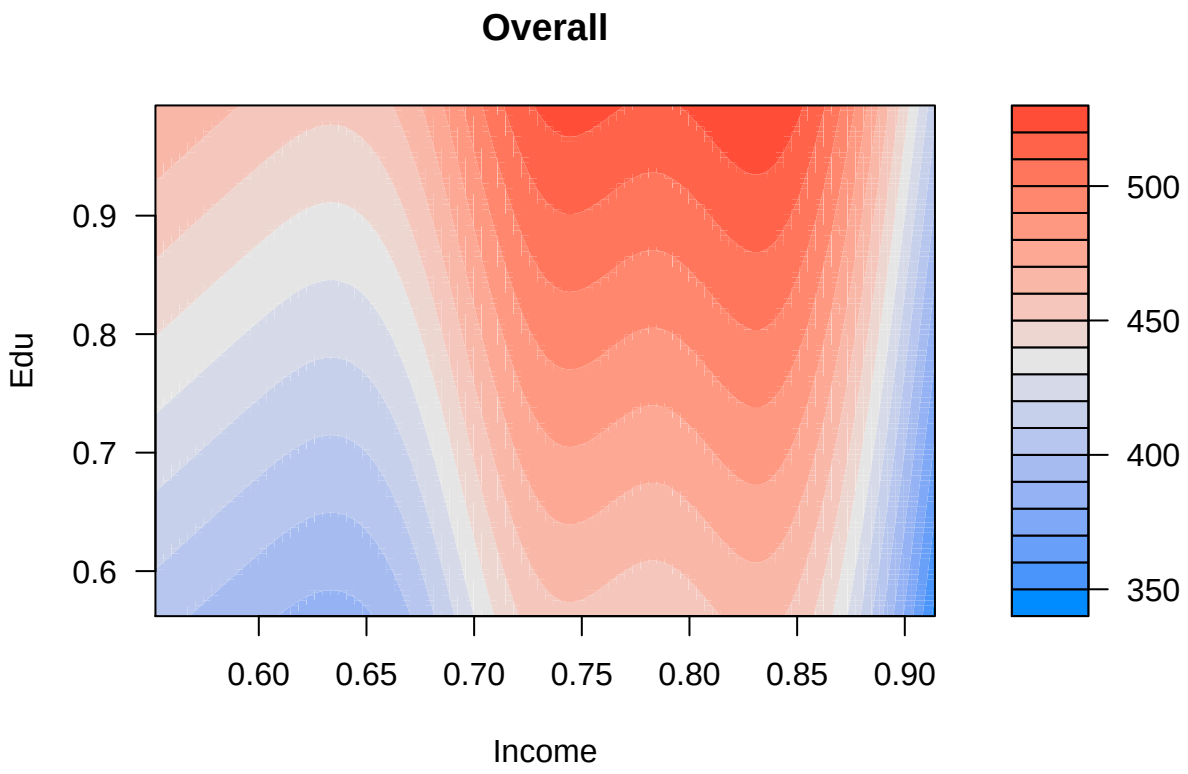
```
par(mfrow=c(1,2))
plot(gam_model_3)
```




```
par(mfrow=c(1,1))
vis.gam(gam_model_3, type='response', plot.type='contour')
```

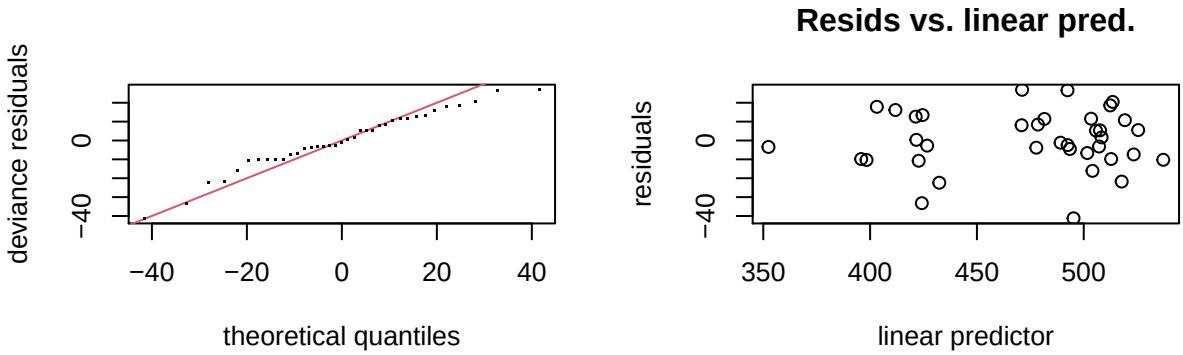


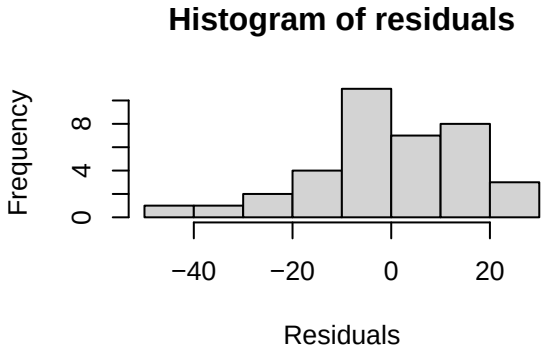

```
visreg2d(gam_model_3, xvar='Income', yvar='Edu', scale='response')
```

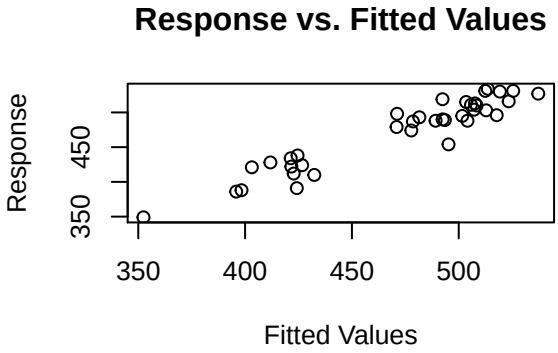


5.3. Análisis de Resultados

```
par(mfrow=c(2,2))
gam.check(gam_model_1)
```

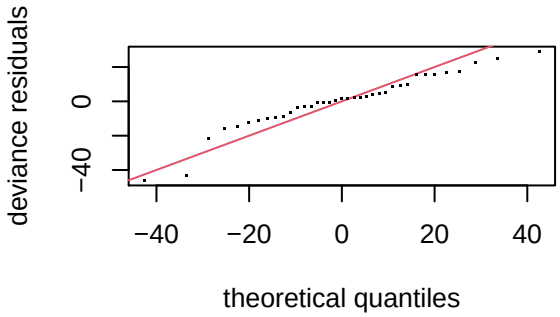


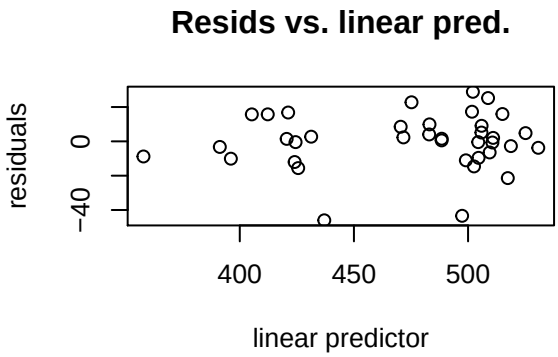


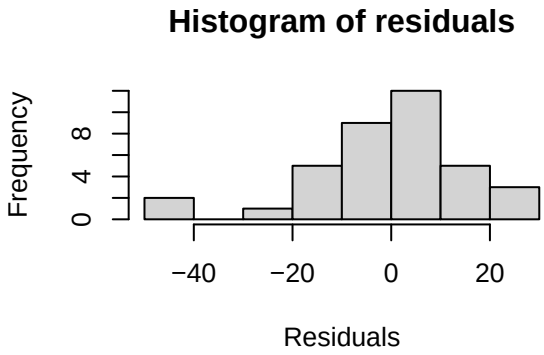


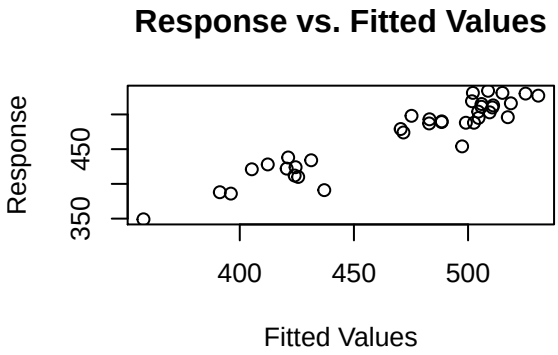

```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 19 iterations.
## The RMS GCV score gradient at convergence was 2.194351e-05 .
## The Hessian was positive definite.
## Model rank =  37 / 37
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##             k'  edf k-index p-value
## s(Income) 9.00 8.01    1.49    1.00
## s(Health) 9.00 2.10    1.22    0.90
## s(Edu)    9.00 1.00    1.27    0.92
## s(HDI)    9.00 1.00    1.02    0.52
```

```
gam.check(gam_model_2)
```



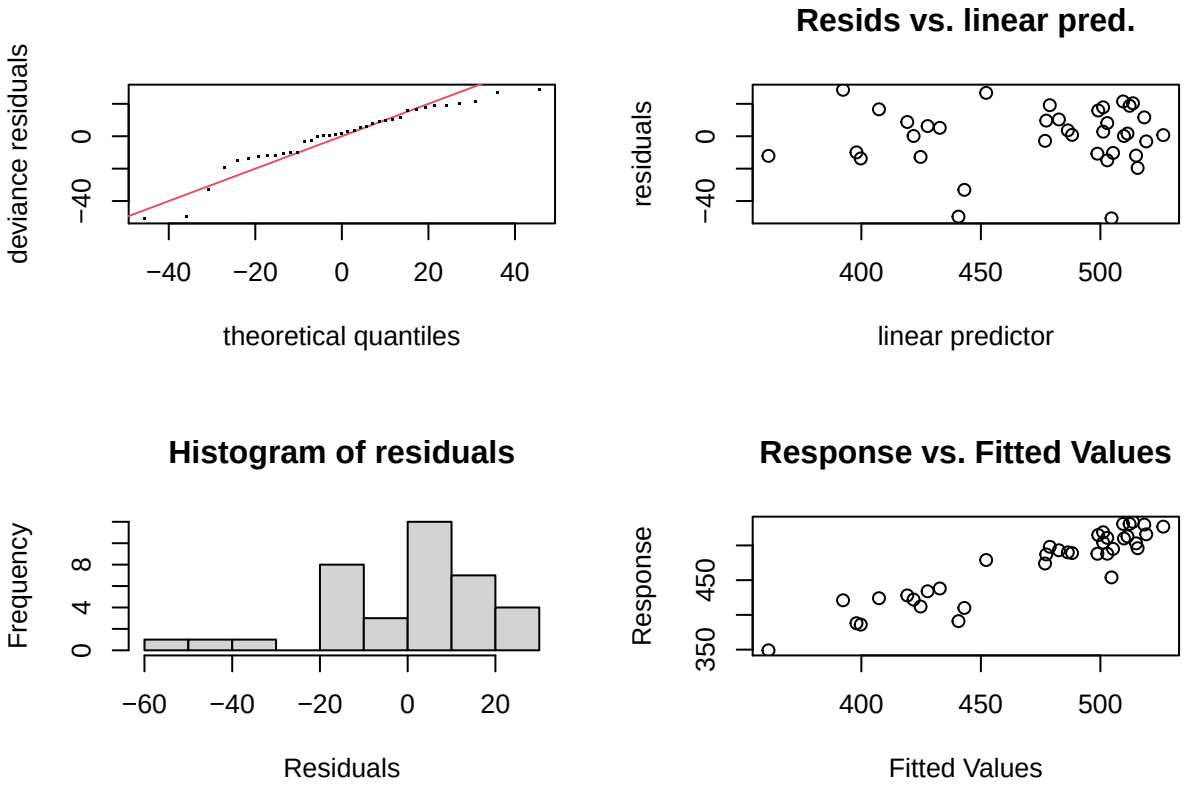







```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 21 iterations.
## The RMS GCV score gradient at convergence was 1.944642e-05 .
## The Hessian was positive definite.
## Model rank =  28 / 28
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##             k'  edf k-index p-value
## s(Income) 9.00 6.05    1.42    0.99
## s(Edu)    9.00 4.18    1.26    0.93
## s(HDI)    9.00 1.00    0.98    0.52
```

```
gam.check(gam_model_3)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 18 iterations.
## The RMS GCV score gradient at convergence was 4.67622e-05 .
## The Hessian was positive definite.
## Model rank =  19 / 19
##
## 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(Income) 9.00 6.18    1.44    1.00
## s(Edu)    9.00 1.00    1.21    0.88
```

5.4. Selección de Modelos

5.4.1 ANOVA

Identificar el modelo polinómico más simple que permite explicar la relación entre variables equivale a identificar el grado de polinomio a partir del cual ya no hay una mejora significativa del ajuste. Cuando se comparan dos modelos anidados (el modelo de menor tamaño está formado por un subset de predictores del modelo mayor), se puede saber si el modelo mayor aporta una mejora sustancial estudiando si los coeficientes de regresión de los predictores adicionales son distintos a cero. El test estadístico empleado para hacerlo es el ANOVA.

De los resultados obtenidos a continuación podemos observar que el P-Value de la comparación entre el modelo lineal y el modelo de GAM 1 es prácticamente 0 lo que nos indica que el modelo lineal no es suficiente. Pero según vamos eliminando variables al modelo, obtenemos que el p-valor aumenta, lo que nos indica que son modelos inferiores y que, por eso, mediante la comparación de modelos ANOVA, debemos quedarnos con el Modelo 2.

Model 2: Overall ~ s(Income) + s(Health) + s(Edu) + s(HDI).

```
#ANOVA
anova(linear_model,gam_model_1,gam_model_2,gam_model_3)
```

```
## Analysis of Variance Table
##
## Model 1: Overall ~ Interest + Support + Income + Health + Edu + HDI
## Model 2: Overall ~ s(Income) + s(Health) + s(Edu) + s(HDI)
## Model 3: Overall ~ s(Income) + s(Edu) + s(HDI)
## Model 4: Overall ~ s(Income) + s(Edu)
##   Res.Df     RSS       Df Sum of Sq      F    Pr(>F)
## 1 30.000 23753.0
## 2 23.892  8443.5  6.10805   15309.5 7.0924 0.0001889 ***
## 3 24.771  9187.4 -0.87938    -743.9 2.3938 0.1368801
## 4 28.818 12264.1 -4.04661   -3076.7 2.1514 0.1049413
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5.4.2. Cross Validation

Otra forma de identificar con que polinomio se consigue el mejor modelo es mediante cross-validation. El proceso consiste en ajustar un modelo para cada grado de polinomio y estimar su test error (Mean Square Error). El mejor modelo es aquel a partir del cual ya no hay una reducción sustancial del test error.

Según los resultados obtenidos, podemos observar que el modelo (obviando el lineal) que presenta un menor Mean Squared Error es el modelo 4, con un 48.87.

Model 4: Overall ~ s(Income) + s(Edu)

```
#CV
set.seed(1998)

#MODELOS
linear_model <- lm(Overall ~ Interest + Support + Income + Health + Edu + HDI, data=pisa_training_data)
predictions_linear_model <- linear_model %>% predict(pisa_testing_data)
data.frame(
  RMSE = rmse(predictions_linear_model, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_linear_model, pisa_testing_data$Overall)
)
```

```
##       RMSE       R2
## 1 28.91924 0.787957
```

```
gam_model_1 <- gam(Overall ~ s(Income) + s(Health)  + s(Edu) + s(HDI), data=pisa_training_data)
predictions_gam_model_1 <- gam_model_1 %>% predict(pisa_testing_data)
data.frame(
  RMSE = rmse(predictions_gam_model_1, pisa_testing_data$Overall),
```

```
  R2 = R2_Score(predictions_gam_model_1, pisa_testing_data$Overall)
)
```

```
##       RMSE         R2
## 1 74.79354 -0.4183341
```

```
gam_model_2 <- gam(Overall ~ s(Income)  + s(Edu) + s(HDI), data=pisa_training_data)
predictions_gam_model_2 <- gam_model_2 %>% predict(pisa_testing_data)
data.frame(
  RMSE = rmse(predictions_gam_model_2, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_gam_model_2, pisa_testing_data$Overall)
)
```

```
##       RMSE           R2
## 1 62.99677 -0.006205989
```

```
gam_model_3 <- gam(Overall ~ s(Income) + s(Edu) , data=pisa_training_data)
predictions_gam_model_3 <- gam_model_3 %>% predict(pisa_testing_data)
data.frame(
  RMSE = rmse(predictions_gam_model_3, pisa_testing_data$Overall),
  R2 = R2_Score(predictions_gam_model_3, pisa_testing_data$Overall)
)
```

```
##       RMSE        R2
## 1 48.87032 0.3944629
```

6.Bibliografía Utilizada:

Spline Regression: http://www.sthda.com/english/articles/40-regression-analysis/162-nonlinear-regression-essentials-in-r-polynomial-and-spline-regression-models/

Spline Regression: https://medium.com/analytics-vidhya/spline-regression-in-r-960ca82aa62c

Métodos de regresión no lineal: https://rpubs.com/Joaquin_AR/250069

Modelo GAM: https://m-clark.github.io/generalized-additive-models/preface.html

Modelo GAM: http://environmentalcomputing.net/intro-to-gams/