Week 6 - Data Science II

Phileas Dazeley-Gaist

01/02/2022

Today: Cross validation and resampling methods.

Note: The standard training data to testing data ratio is 70% to 30%. Note: The central limit theorem states that the average of averages in a data set tends to be normally distributed.

```
# Applied Data Science II - Week 6
# # ---------------------------------------------------------------

# Today we are going to talk about SPLINES and GAMS!
#
#
# # ---------------------------------------------------------------
#
# Load your libraries!
#
# # ---------------------------------------------------------------

library(ISLR2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

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

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(splines)
library(npreg)
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.
```

```
library(PerformanceAnalytics)
```

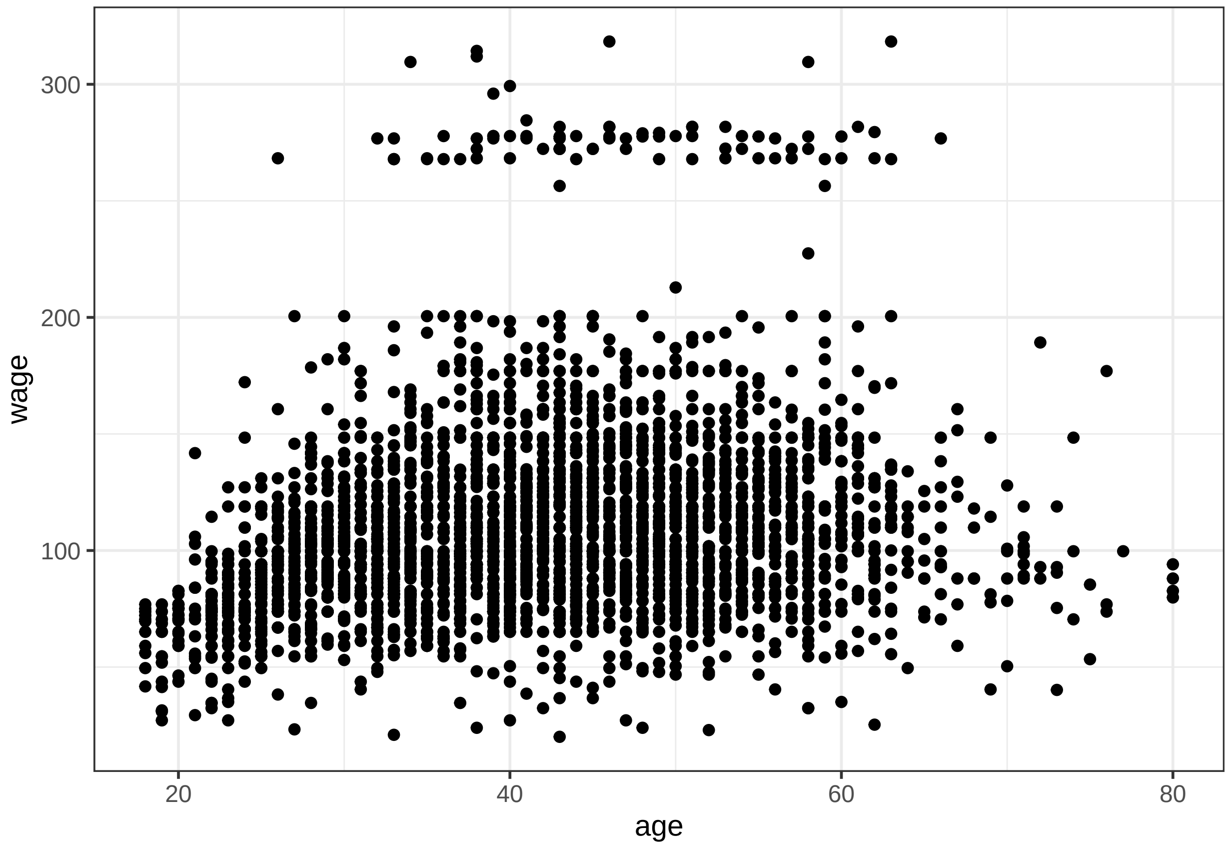
```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
# # ---------------------------------------------------------------
#
# Splines!
#
# # ---------------------------------------------------------------

# Let's start by checking out the Wage dataset. It's not very fancy, but it's useful for teach
# This dataset is just wages and other data for a group of 3000 male workers in the Mid-Atlant

attach(Wage)
head(Wage)
```

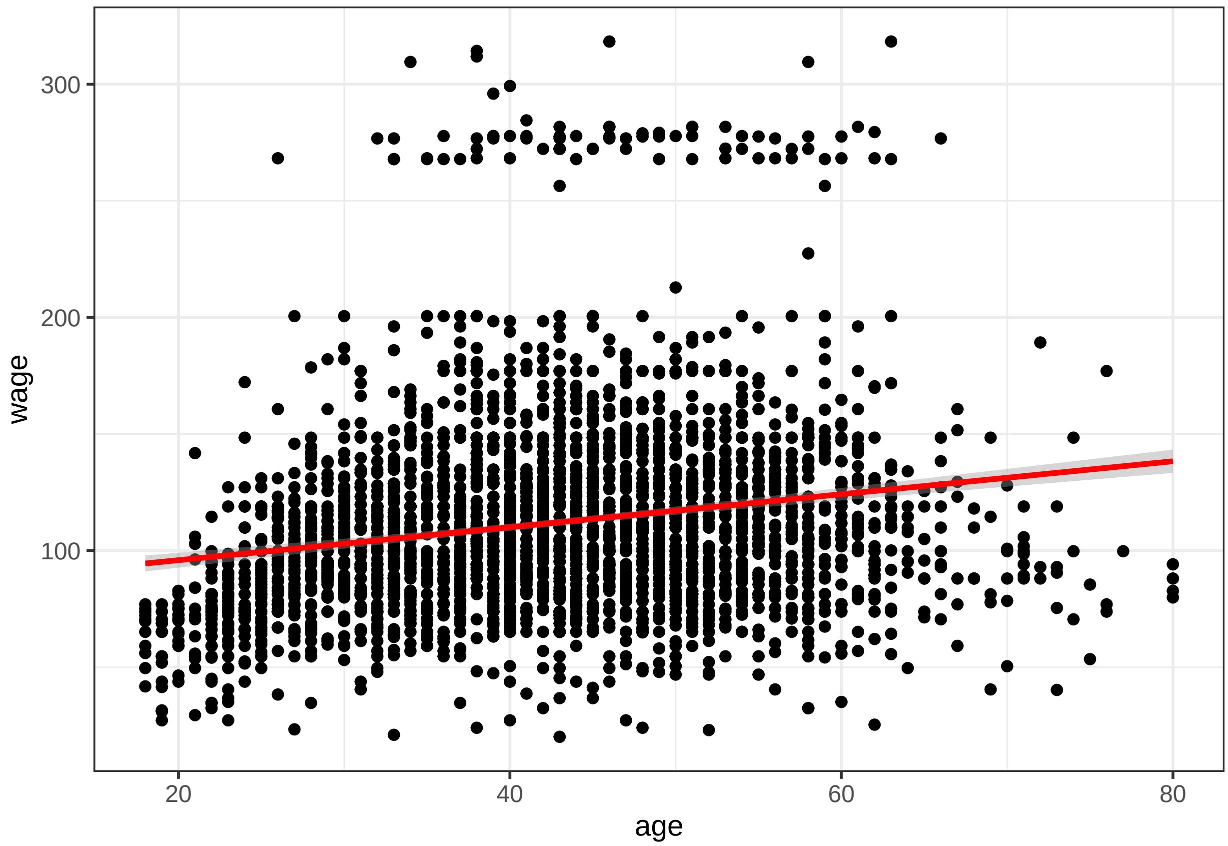
```
##        year age            maritl     race       education             region
## 231655 2006  18 1. Never Married 1. White    1. < HS Grad 2. Middle Atlantic
## 86582  2004  24 1. Never Married 1. White 4. College Grad 2. Middle Atlantic
## 161300 2003  45       2. Married 1. White 3. Some College 2. Middle Atlantic
## 155159 2003  43       2. Married 3. Asian 4. College Grad 2. Middle Atlantic
## 11443  2005  50      4. Divorced 1. White      2. HS Grad 2. Middle Atlantic
## 376662 2008  54       2. Married 1. White 4. College Grad 2. Middle Atlantic
##              jobclass         health health_ins  logwage      wage
## 231655  1. Industrial      1. <=Good      2. No 4.318063  75.04315
## 86582  2. Information 2. >=Very Good      2. No 4.255273  70.47602
## 161300  1. Industrial      1. <=Good     1. Yes 4.875061 130.98218
## 155159 2. Information 2. >=Very Good     1. Yes 5.041393 154.68529
## 11443  2. Information      1. <=Good     1. Yes 4.318063  75.04315
## 376662 2. Information 2. >=Very Good     1. Yes 4.845098 127.11574
```

```
# Let's take a look at a plot between wages and age...

Wage %>%
        ggplot(aes(x = age, y = wage)) +
        geom_point() +
        theme_bw()
```




```
# It's got some pretty funky behavior, right? If we add a linear model to it, we can see it's 
# fit really well ...

Wage %>%
        ggplot(aes(x = age, y = wage)) +
        geom_point() +
        theme_bw() +
        geom_smooth(method = "lm", color = "red")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```r
# We can also just run the math and see that the fit probably won't be great:

summary(lm(wage ~ age, data = Wage))
```

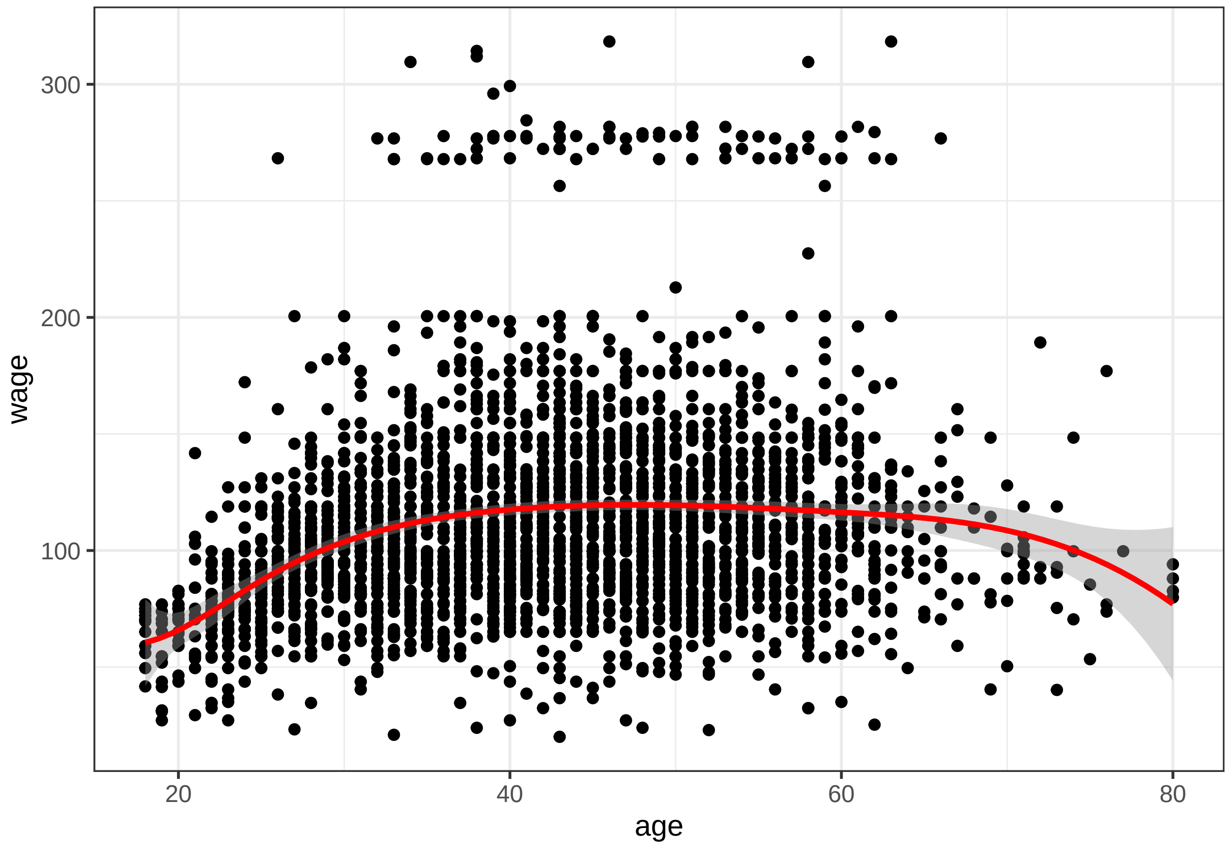
```
##
## Call:
## lm(formula = wage ~ age, data = Wage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -100.265  -25.115   -6.063   16.601  205.748
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 81.70474    2.84624   28.71   <2e-16 ***
## age          0.70728    0.06475   10.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.93 on 2998 degrees of freedom
## Multiple R-squared:  0.03827,    Adjusted R-squared:  0.03795
## F-statistic: 119.3 on 1 and 2998 DF,  p-value: < 2.2e-16
```

```r
# Yep, not super great.
```

```r
# So since we suspect some nonlinearity here, we can use a spline!
# To use a spline is straightforward - you can call the bs() function (from the spline library
# and this lets you manually add in "knots" to your spline regression!

spline_regression <- lm(wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)
# the bs() function changes everything, it stands for B-spline analysis
# the knots parameter is the position of the knots we would like to see

# for information on the relationship between the number of knots and the degrees of freedom,

# and, as we can see - the fit is better!
summary(spline_regression)
```

```
##
## Call:
## lm(formula = wage ~ bs(age, knots = c(25, 40, 60)), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.832 -24.537  -5.049  15.209 203.207
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       60.494      9.460   6.394 1.86e-10 ***
## bs(age, knots = c(25, 40, 60))1    3.980     12.538   0.317 0.750899
## bs(age, knots = c(25, 40, 60))2   44.631      9.626   4.636 3.70e-06 ***
## bs(age, knots = c(25, 40, 60))3   62.839     10.755   5.843 5.69e-09 ***
## bs(age, knots = c(25, 40, 60))4   55.991     10.706   5.230 1.81e-07 ***
## bs(age, knots = c(25, 40, 60))5   50.688     14.402   3.520 0.000439 ***
## bs(age, knots = c(25, 40, 60))6   16.606     19.126   0.868 0.385338
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.92 on 2993 degrees of freedom
## Multiple R-squared:  0.08642,    Adjusted R-squared:  0.08459
## F-statistic: 47.19 on 6 and 2993 DF,  p-value: < 2.2e-16
```

```r
# we can also visualize it!

Wage %>%
        ggplot(aes(x = age, y = wage)) +
        geom_point() +
        theme_bw() +
        geom_smooth(method = "lm", color = "red",
                    formula = y ~ bs(x, knots = c(25, 40, 60)))
```




```
# note the somewhat funky way you have to pass these arguments into ggplot2!
```

```
# now, in the above, we SPECIFIED where to put the knots. We can achieve a similar result
# by letting R choose for us!

# Let's show that these produce a vector with the same dimensions...

dim(bs(age, knots = c(25, 40, 60)))
```

```
## [1] 3000    6
```

```
dim(bs(age, df = 6))
```

```
## [1] 3000    6
```

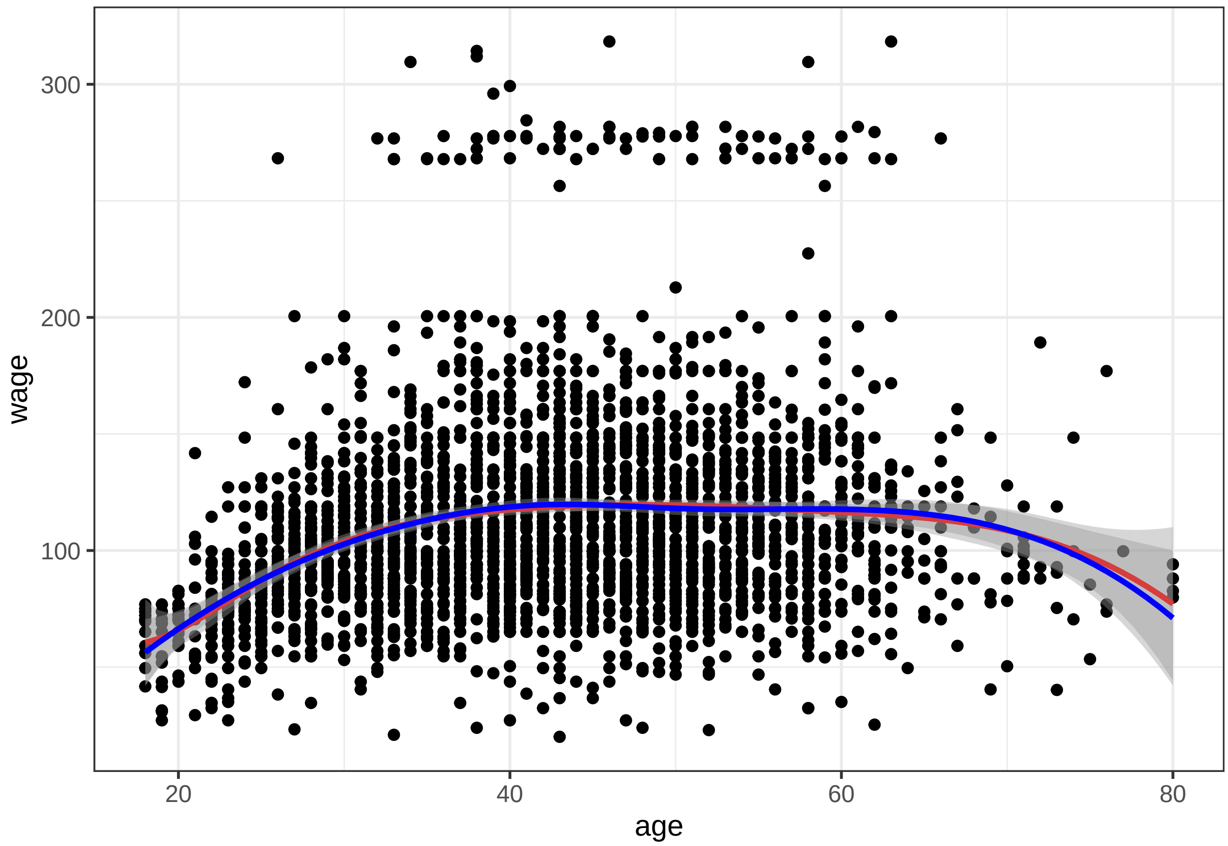
```
# why 6?
# This df command produces a spline with six basis functions.
# This is because the bs() function naturally produces a  a cubic spline which, when it has
# three knots, has seven degrees of freedom; six basis functions + one intercept.

# HOWEVER MANY KNOTS YOU WOULD LIKE IS: 2*knots = number of degrees of freedom.

attr(bs(age, df = 6), "knots")
```

```
##   25%   50%   75%
## 33.75 42.00 51.00
```

```
# If we just let R choose...

summary(lm(wage ~ bs(age, df = 6), data = Wage))
```

```
##
## Call:
## lm(formula = wage ~ bs(age, df = 6), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.681 -24.403  -5.202  15.441 201.413
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          56.314      7.258   7.759 1.17e-14 ***
## bs(age, df = 6)1     27.824     12.435   2.238   0.0253 *
## bs(age, df = 6)2     54.063      7.127   7.585 4.41e-14 ***
## bs(age, df = 6)3     65.828      8.323   7.909 3.62e-15 ***
## bs(age, df = 6)4     55.813      8.724   6.398 1.83e-10 ***
## bs(age, df = 6)5     72.131     13.745   5.248 1.65e-07 ***
## bs(age, df = 6)6     14.751     16.209   0.910   0.3629
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.91 on 2993 degrees of freedom
## Multiple R-squared:  0.08729,    Adjusted R-squared:  0.08546
## F-statistic: 47.71 on 6 and 2993 DF,  p-value: < 2.2e-16
```

```
# slightly better! does it look all that different?

Wage %>%
        ggplot(aes(x = age, y = wage)) +
        geom_point() +
        theme_bw() +
        geom_smooth(method = "lm", color = "red",
                    formula = y ~ bs(x, knots = c(25, 40, 60))) +
        geom_smooth(method = lm, color = "blue",
                    formula = y ~ bs(x, df = 6))
```

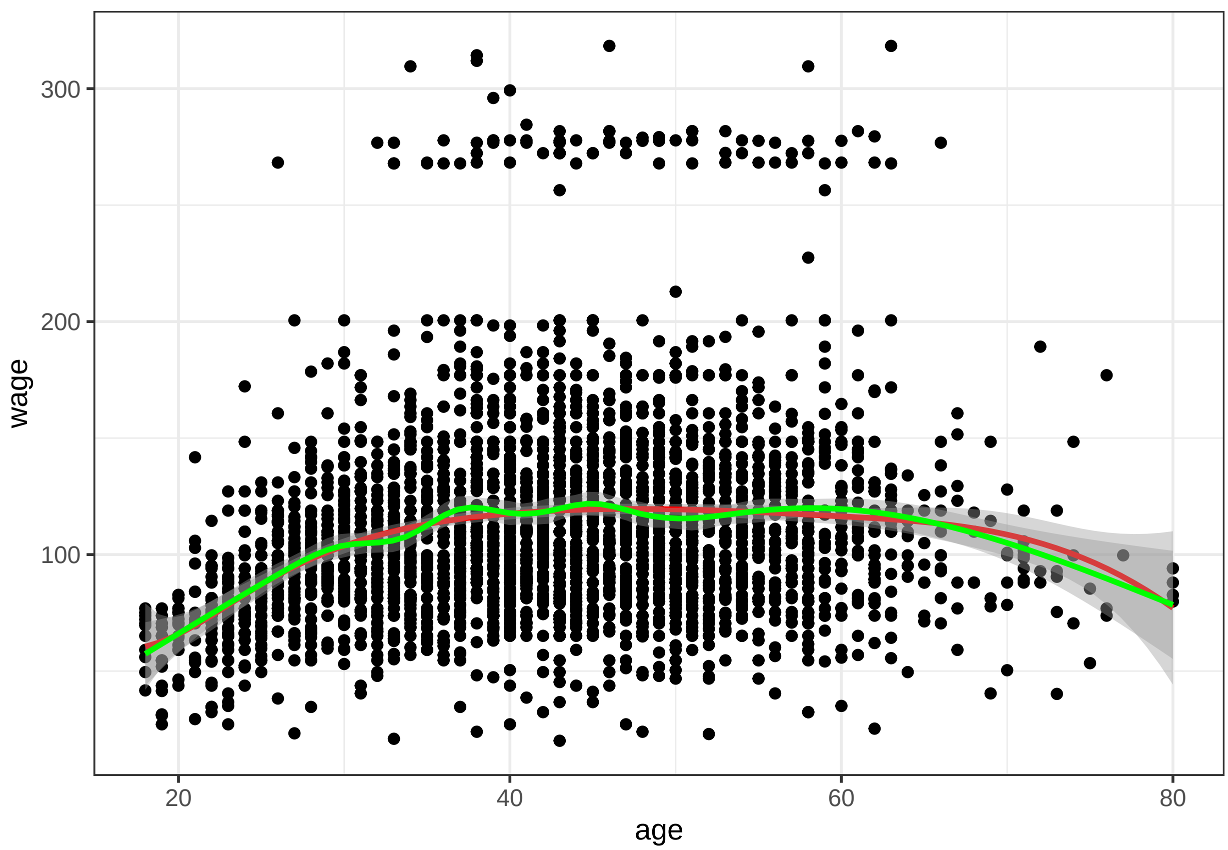



```r
# nah.

# Want to fit a spline of ANY degree (and not just a cubic one?) Use the NS function! This uses
# "natural" splines which are even more flexible:

summary(lm(wage ~ ns(age, df = 12), data = Wage))
```

```
##
## Call:
## lm(formula = wage ~ ns(age, df = 12), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.668 -24.334  -5.014  15.246 201.186
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           57.384      6.921   8.292  < 2e-16 ***
## ns(age, df = 12)1     49.813      7.188   6.930 5.12e-12 ***
## ns(age, df = 12)2     45.386      9.318   4.871 1.17e-06 ***
## ns(age, df = 12)3     66.918      8.595   7.785 9.52e-15 ***
## ns(age, df = 12)4     59.476      8.747   6.799 1.26e-11 ***
## ns(age, df = 12)5     60.372      8.646   6.983 3.55e-12 ***
```

```
## ns(age, df = 12)6    67.180      9.009   7.457 1.15e-13 ***
## ns(age, df = 12)7    58.753      8.696   6.756 1.69e-11 ***
## ns(age, df = 12)8    57.421      8.488   6.765 1.60e-11 ***
## ns(age, df = 12)9    61.185      7.942   7.704 1.78e-14 ***
## ns(age, df = 12)10   55.098      7.848   7.020 2.73e-12 ***
## ns(age, df = 12)11   67.378     17.242   3.908 9.52e-05 ***
## ns(age, df = 12)12    7.282     12.226   0.596    0.551
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.89 on 2987 degrees of freedom
## Multiple R-squared:  0.08973,    Adjusted R-squared:  0.08607
## F-statistic: 24.54 on 12 and 2987 DF,  p-value: < 2.2e-16
```

```r
Wage %>%
        ggplot(aes(x = age, y = wage)) +
        geom_point() +
        theme_bw() +
        geom_smooth(method = "lm", color = "red",
                    formula = y ~ bs(x, knots = c(25, 40, 60))) +
        geom_smooth(method = lm, color = "green",
                    formula = y ~ ns(x, df = 12))
```



```r
# But why have any silly constraints at all! Let's just use a SMOOTHING spline.
# Note: your textbook uses the smooth.spline function. We're going to use a different one
# from the npreg library as it gives you way more flexibility.

smoooooooooth <- npreg::ss(age, wage, nknots = 16)
smoooooooooth
```

```
##
## Call:
## npreg::ss(x = age, y = wage, nknots = 16)
##
## Smoothing Parameter  spar = 0.3135218   lambda = 1.097456e-05
## Equivalent Degrees of Freedom (Df) 6.441072
## Penalized Criterion (RSS) 4762520
## Generalized Cross-Validation (GCV) 1594.345
```

```r
summary(smoooooooooth)
```

```
##
## Call:
## npreg::ss(x = age, y = wage, nknots = 16)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.540 -24.432  -5.069  15.219 202.353
##
## Approx. Signif. of Parametric Effects:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    106.21      1.446  73.424  0.00000 ***
## x               27.01     12.026   2.246  0.02478   *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approx. Signif. of Nonparametric Effects:
##                  Df  Sum Sq Mean Sq F value Pr(>F)
## s(x)          4.441  250650   56439   35.48      0 ***
## Residuals 2993.559 4762520    1591
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.89 on 2994 degrees of freedom
## Multiple R-squared:  0.08804,    Adjusted R-squared:  0.08635
## F-statistic: 52.05 on 5.441 and 2994 DF,  p-value: <2e-16
```

```r
smoooooooooth$fit$knot
```

```
##  [1] 18 22 26 30 34 38 42 46 50 54 58 62 66 70 74 80
```

```r
# and, lastly, let's let the ss function actually choose the number of knots for us
# through cross-validation!

final_smooth <- npreg::ss(age, wage)
final_smooth
```

```
##
## Call:
## npreg::ss(x = age, y = wage)
##
## Smoothing Parameter  spar = 0.3169286   lambda = 1.161449e-05
## Equivalent Degrees of Freedom (Df) 6.468966
## Penalized Criterion (RSS) 4762594
## Generalized Cross-Validation (GCV) 1594.4
```

```r
summary(final_smooth)
```

```
##
## Call:
## npreg::ss(x = age, y = wage)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -99.57 -24.43  -5.07  15.22 202.42
##
## Approx. Signif. of Parametric Effects:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   106.23      1.444  73.555   0.0000 ***
## x              27.09     11.920   2.273   0.0231   *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approx. Signif. of Nonparametric Effects:
##                 Df  Sum Sq Mean Sq F value Pr(>F)
## s(x)         4.469  250220   55991   35.19      0 ***
## Residuals 2993.531 4762594    1591
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.89 on 2994 degrees of freedom
## Multiple R-squared:  0.08803,    Adjusted R-squared:  0.08632
## F-statistic: 51.73 on 5.469 and 2994 DF,  p-value: <2e-16
```

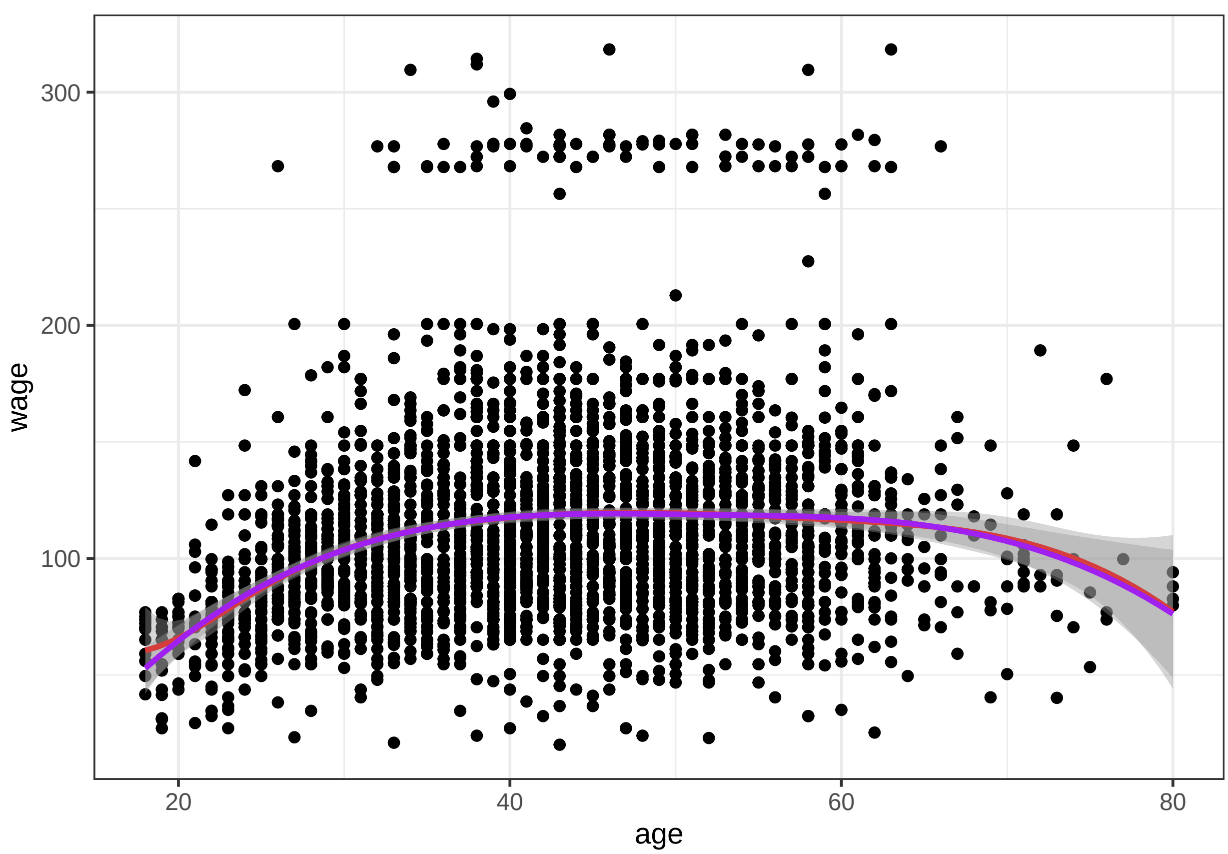
```r
final_smooth$fit$knot
```

```
##  [1] 18 19 20 21 22 23 25 26 27 28 29 30 32 33 34 35 36 38 39 40 41 42 43 45 46
## [26] 47 48 49 50 52 53 54 55 56 58 59 60 61 62 63 65 66 67 68 69 70 72 73 74 75
## [51] 76 80
```

```r
length(final_smooth$fi$knot)
```

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

```r
# and let's see what this looks like...

pred <- predict(final_smooth, Wage$wage)

Wage %>%
        ggplot(aes(x = age, y = wage)) +
        geom_point() +
        theme_bw() +
        geom_smooth(aes(x=age, y=wage),method = "lm", color = "red",
                    formula = y ~ bs(x, knots = c(25, 40, 60))) +
        stat_smooth(method = "gam", formula = y ~ bs(x, k = 52), color = "purple")
```



```r
# # ----------------------------------------------------------------
#
# Stop! Back to the lecture!
#
# # ----------------------------------------------------------------
```

```r
# # ----------------------------------------------------------------
#
# Let's TALK ABOUT THEM GAMS
#
# # ----------------------------------------------------------------

# Go ahead and download the dataset called "pisa_data.csv" from the Google Drive.

pisa_data <- read_csv("w6 data//pisa_data.csv")
```

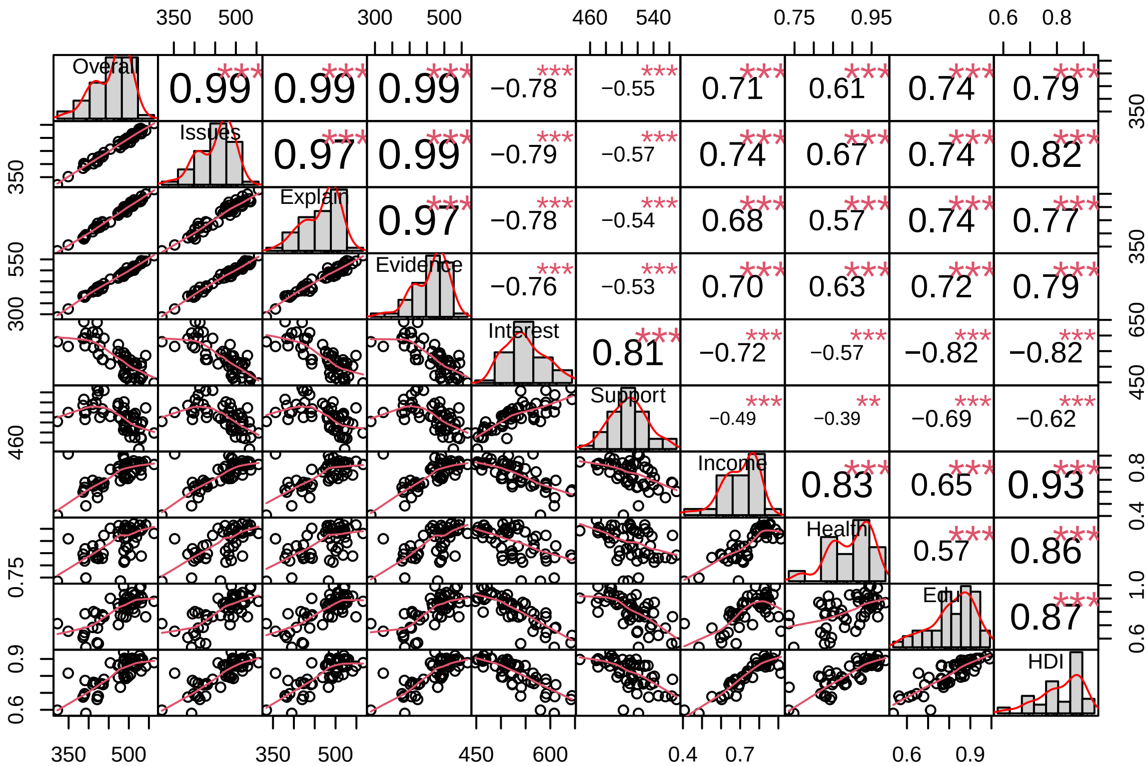
```
## Rows: 65 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): Country
## dbl (10): Overall, Issues, Explain, Evidence, Interest, Support, Income, Hea...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

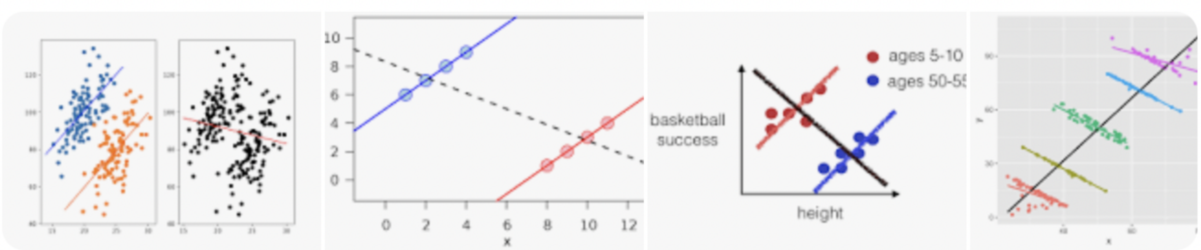
```r
head(pisa_data)
```

```
## # A tibble: 6 x 11
##   Country  Overall Issues Explain Evidence Interest Support Income Health    Edu
##   <chr>      <dbl>  <dbl>   <dbl>    <dbl>    <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 Albania       NA     NA      NA       NA       NA      NA  0.599  0.886  0.716
## 2 Argenti~     391    395     386      385      567     506  0.678  0.868  0.786
## 3 Austral~     527    535     520      531      465     487  0.826  0.965  0.978
## 4 Austria      511    505     516      505      507     515  0.835  0.944  0.824
## 5 Azerbai~     382    353     412      344      612     542  0.566  0.78  NA
## 6 Belgium      510    515     503      516      503     492  0.831  0.935  0.868
## # ... with 1 more variable: HDI <dbl>
```

```r
# This is a nifty little dataset we've put together for you based on education.
# The data set has been constructed using average Science scores by country from
# the Programme for International Student Assessment (PISA) 2006, along with GNI per capita,
# Educational Index, Health Index, and Human Development Index from UN data.

# Drop nulls!

pisa_data <- pisa_data %>%
```

```r
        na.omit()

# Let's peek at the data with the chart.Correlation function

chart.Correlation(pisa_data[,2:11], histogram = TRUE, method = "pearson")
```





Simpson's paradox, also called Yule-Simpson effect, in statistics, an effect that **occurs when the marginal association between two categorical variables is qualitatively different from the partial association between the same two variables** after controlling for one or more other variables.

```r
# okay, let's start simple and fit a linear model *first*. we're going to use the
# mgcv library to use the gam() function and not pass anything fancy in. Note that there are
# LOTS of libraries that use GAMs, so it's probably good to specifcy WHICH library you want to
# with ::
# Let's try predicting the overal score based solely on income.
```

```r
pisa_lm_simple <- mgcv::gam(Overall ~ Income, data = pisa_data)
summary(pisa_lm_simple)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ Income
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   206.36      37.89   5.447 1.56e-06 ***
## Income        354.18      50.17   7.060 4.84e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.489   Deviance explained = 49.9%
## GCV = 1548.4  Scale est. = 1488.9    n = 52
```

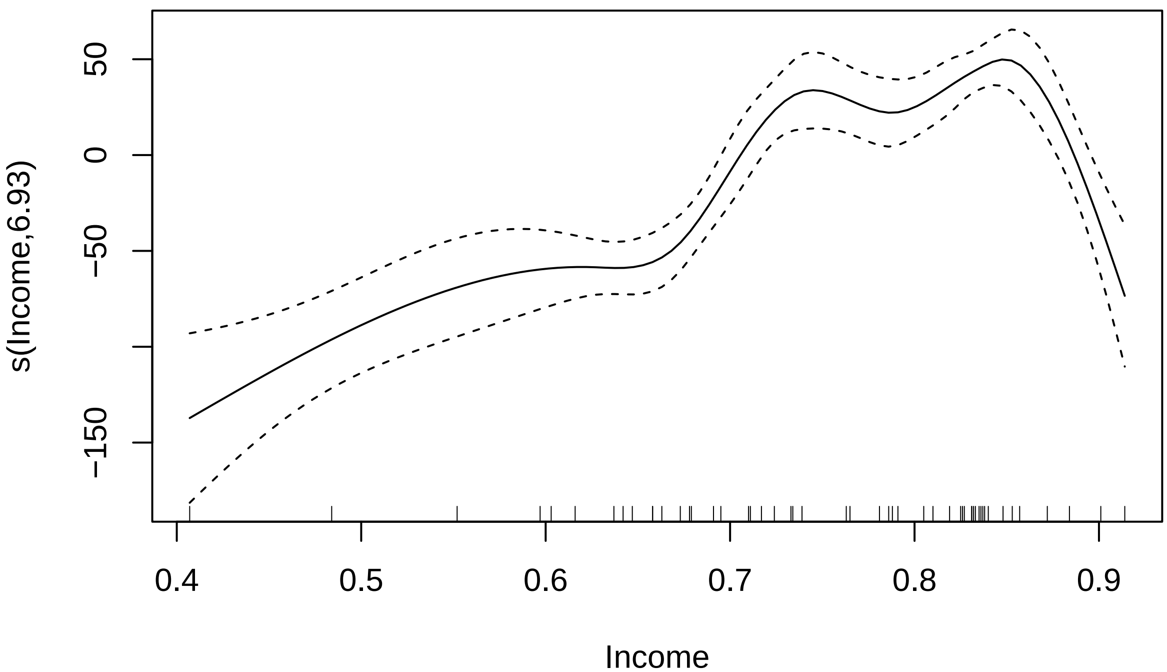
```r
# okay, not terribly bad. The overal adjusted R2 isn't too shabby. Now let's try fitting
# a very straightforward GAM that takes advantage of splines.
```

```r
pisa_gam_simple <- gam(Overall ~ s(Income, bs="cr"), data = pisa_data)
# Note: We again use the gam function as before for basic model fitting, but now we are using
# the s function within the formula to denote the smoothing spline terms. Within that function
# also specify the type of smooth, though a default is available. I chose bs = cr, denoting cu
# regression splines (how we started above!)

summary(pisa_gam_simple)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  471.154      3.386   139.1   <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) 6.935  7.787 25.77  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.795   Deviance explained = 82.3%
## GCV = 703.74  Scale est. = 596.36    n = 52
```

```
plot(pisa_gam_simple)
```




```
# looks better! Now let's try multiple predictors - and this time, let's use a test and train
# approach to get a better idea of overall model performance.

# start linear...

set.seed(12345)
index <- createDataPartition(pisa_data$Overall, p = .8, list=FALSE)
training_data <- pisa_data[ index,]
test_data  <- pisa_data[-index,]

pisa_lm_multivariate <-  mgcv::gam(Overall ~ Income + Edu + Health, data = training_data)
summary(pisa_lm_multivariate)
```

```
##
## Family: gaussian
## Link function: identity
```

```
##
## Formula:
## Overall ~ Income + Edu + Health
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    48.97     100.46   0.488  0.62856
## Income        194.66     108.03   1.802  0.07909 .
## Edu           248.07      61.49   4.034  0.00024 ***
## Health         83.30     174.83   0.476  0.63632
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.614   Deviance explained = 64.1%
## GCV = 1326.3  Scale est. = 1205.8    n = 44
```

```
# ...get rmse!

predictions_lm <- predict(pisa_lm_multivariate, test_data)
RMSE(predictions_lm, test_data$Overall)
```

```
## [1] 30.24088
```

```
# now let's go full GAMMMMMSSSSS
pisa_gam_multivariate <- gam(Overall ~ s(Income) + s(Edu) + s(Health), data = training_data)
summary(pisa_gam_multivariate)
```

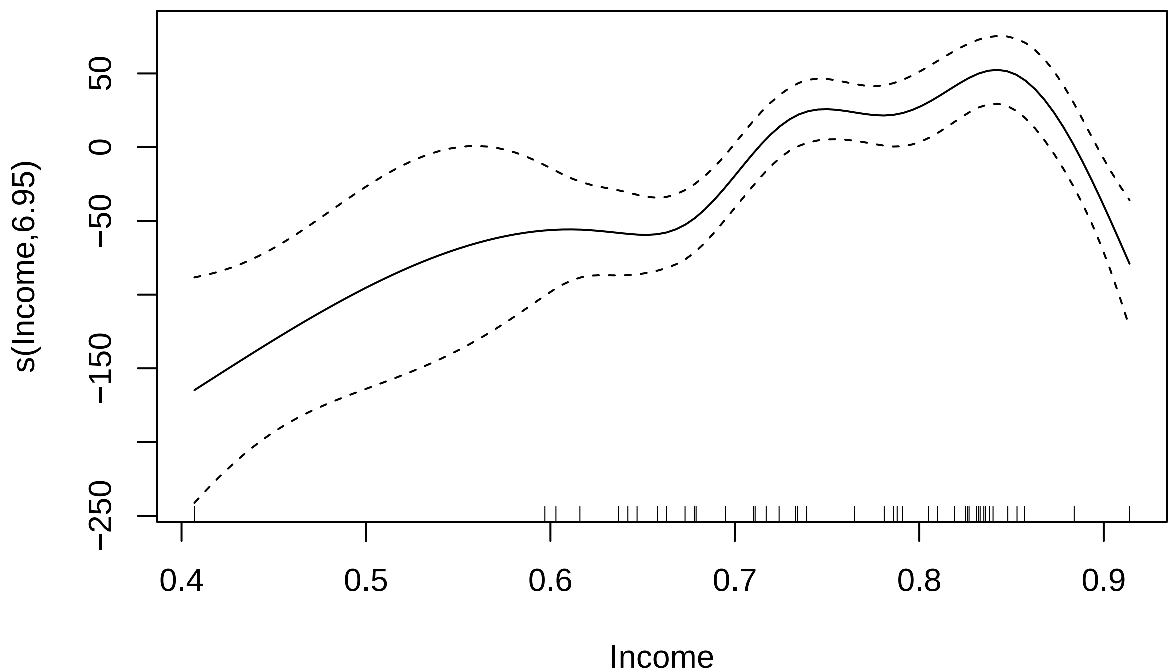
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Edu) + s(Health)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  470.318      3.048   154.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df     F  p-value
## s(Income) 6.948  7.904 8.025 1.03e-05 ***
## s(Edu)    5.334  6.305 2.219   0.0623 .
## s(Health) 1.000  1.000 0.953   0.3368
```

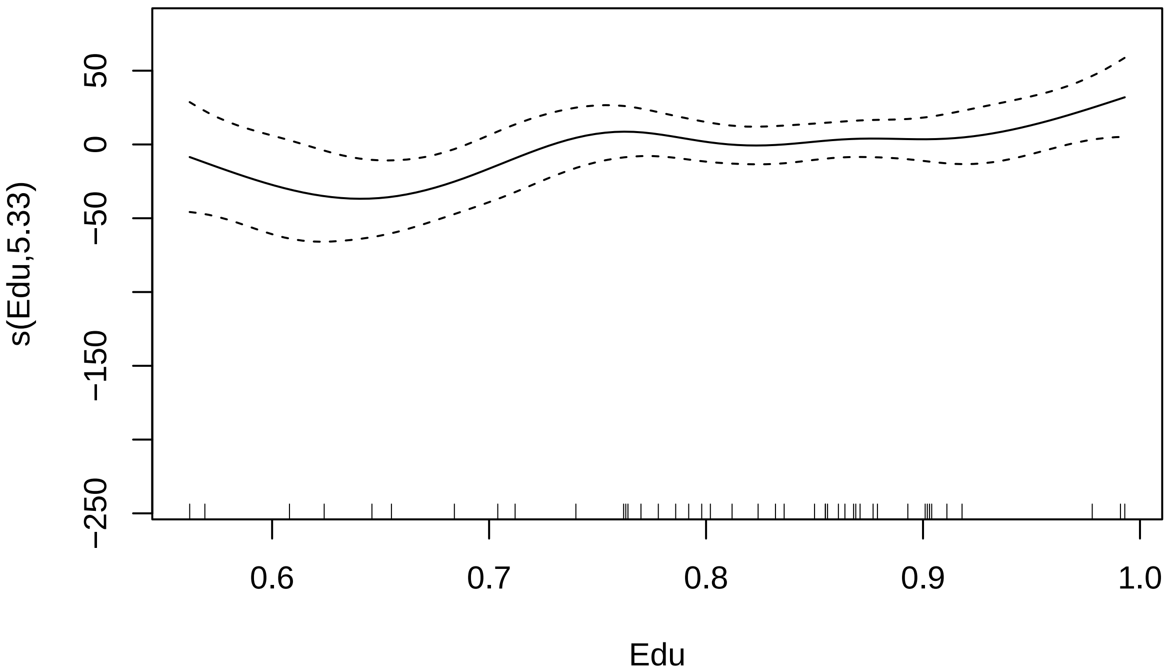
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.869   Deviance explained =   91%
## GCV = 605.14  Scale est. = 408.72    n = 44
```

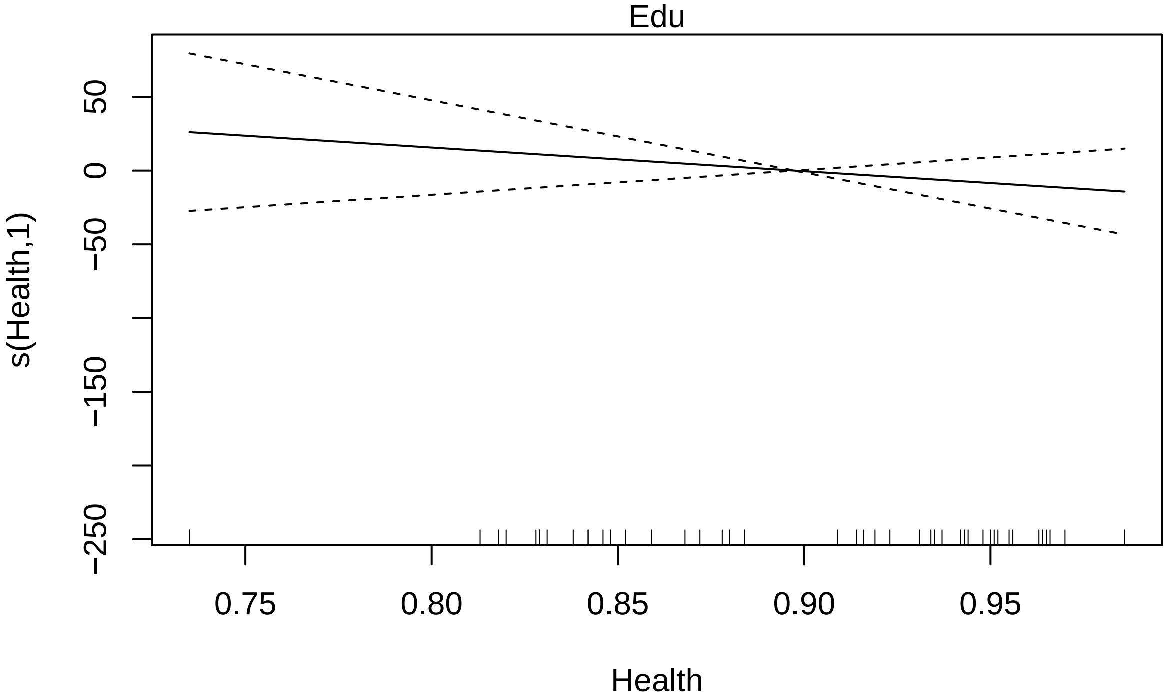
```r
predictions_gam <- predict(pisa_gam_multivariate, test_data)
RMSE(predictions_gam, test_data$Overall)
```

```
## [1] 25.23021
```

```r
# want to see how each of these effects is being modeled? use plot!
plot(pisa_gam_multivariate)
```








```
# if you want it to be prettier, check out the visreg package!
```

```
# # --------------------------------------------------------------
#
# Data Project Time!
#
# # --------------------------------------------------------------

# Go into the Drive and open the file "walmart.csv".
# this is some data related to weekly store sales at a bunch of different walmart stores
```

```
# around the country.Your job is to use all the tools at your disposal (linear models,
# lasso models, ridge models, or GAMS) to try and have the BEST fitting model possible as base
# minimizing RMSE.
# You are trying to predict the weekly_sales based on the data available to you.
# good luck!

walmart <- read_csv("w6 data/walmart.csv")
```

```
## Rows: 6435 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): date
## dbl (7): store, weekly_sales, holiday_flag, temperature, fuel_price, cpi, un...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(walmart)
```

```
## # A tibble: 6 x 8
##   store date       weekly_sales holiday_flag temperature fuel_price   cpi
##   <dbl> <chr>             <dbl>        <dbl>       <dbl>      <dbl> <dbl>
## 1     1 05-02-2010     1643691.            0        42.3       2.57  211.
## 2     1 12-02-2010     1641957.            1        38.5       2.55  211.
## 3     1 19-02-2010     1611968.            0        39.9       2.51  211.
## 4     1 26-02-2010     1409728.            0        46.6       2.56  211.
## 5     1 05-03-2010     1554807.            0        46.5       2.62  211.
## 6     1 12-03-2010     1439542.            0        57.8       2.67  211.
## # ... with 1 more variable: unemployment <dbl>
```

```
walmart_cleaned <- walmart %>%
        mutate(store = as.factor(store),
               holiday_flag = as.factor(holiday_flag),
               year = as.factor(lubridate::year(lubridate::dmy(date))),
               month = as.factor(lubridate::month(date))) %>%
        select(-c(date))


set.seed(12345)
index <- createDataPartition(walmart_cleaned$weekly_sales, p = .8, list=FALSE)
training_data <- walmart_cleaned[ index,]
test_data  <- walmart_cleaned[-index,]

walmart_model <- gam(weekly_sales ~ store + holiday_flag +
                                     s(temperature) +
                                     s(fuel_price) + s(cpi) + s(unemployment) +
```

```
                                     year + month, data = training_data)
summary(walmart_model)
```

```
## 
## Family: gaussian 
## Link function: identity 
## 
## Formula:
## weekly_sales ~ store + holiday_flag + s(temperature) + s(fuel_price) + 
##     s(cpi) + s(unemployment) + year + month
## 
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)     1454624     257049   5.659 1.61e-08 ***
## store2           376933      18781  20.070  < 2e-16 ***
## store3         -1193481      21059 -56.674  < 2e-16 ***
## store4           668728     548014   1.220 0.222418    
## store5         -1302932      21643 -60.200  < 2e-16 ***
## store6           -55595      20884  -2.662 0.007791 ** 
## store7          -723136      80315  -9.004  < 2e-16 ***
## store8          -733813      23966 -30.619  < 2e-16 ***
## store9         -1101456      24143 -45.622  < 2e-16 ***
## store10          518738     547790   0.947 0.343702    
## store11         -236011      21201 -11.132  < 2e-16 ***
## store12         -293097     552588  -0.530 0.595852    
## store13          614313     547736   1.122 0.262107    
## store14          765706     103027   7.432 1.25e-13 ***
## store15         -847598     526213  -1.611 0.107296    
## store16         -828160      80278 -10.316  < 2e-16 ***
## store17         -507798     547380  -0.928 0.353613    
## store18         -374261     526884  -0.710 0.477533    
## store19          -42418     526098  -0.081 0.935741    
## store20          578201      27630  20.926  < 2e-16 ***
## store21         -804433      18745 -42.914  < 2e-16 ***
## store22         -470571     513460  -0.916 0.359463    
## store23         -218273     525356  -0.415 0.677811    
## store24         -118722     526448  -0.226 0.821587    
## store25         -822434      27997 -29.376  < 2e-16 ***
## store26         -465109     526290  -0.884 0.376873    
## store27          274482     513129   0.535 0.592730    
## store28           16625     552728   0.030 0.976006    
## store29         -924656     527766  -1.752 0.079831 .  
## store30        -1116328      18912 -59.028  < 2e-16 ***
## store31         -174666      18942  -9.221  < 2e-16 ***
## store32         -129996      80188  -1.621 0.105048    
## store33        -1105113     547800  -2.017 0.043710 *  
## store34         -370518     549730  -0.674 0.500342    
```

```
## store35        -580067     513736  -1.129 0.258903
## store36       -1181026      19108 -61.810  < 2e-16 ***
## store37       -1034095      19068 -54.230  < 2e-16 ***
## store38        -924065     552602  -1.672 0.094545 .
## store39         -94166      19261  -4.889 1.05e-06 ***
## store40        -649033     525420  -1.235 0.216789
## store41         -82595      79726  -1.036 0.300255
## store42        -809127     547749  -1.477 0.139688
## store43        -859916      35975 -23.903  < 2e-16 ***
## store44       -1105192     547493  -2.019 0.043577 *
## store45        -476212     103073  -4.620 3.93e-06 ***
## holiday_flag1    26246       8519   3.081 0.002075 **
## year2011        -94787      22025  -4.304 1.71e-05 ***
## year2012       -134851      35978  -3.748 0.000180 ***
## month2          118182      11837   9.984  < 2e-16 ***
## month3           66011      13993   4.717 2.45e-06 ***
## month4           67517      16244   4.156 3.29e-05 ***
## month5           66405      18460   3.597 0.000325 ***
## month6          106135      19786   5.364 8.50e-08 ***
## month7           76267      21366   3.570 0.000361 ***
## month8           87531      21520   4.067 4.83e-05 ***
## month9           19603      20214   0.970 0.332215
## month10          20247      18508   1.094 0.274034
## month11         156146      18928   8.249  < 2e-16 ***
## month12         307268      18794  16.349  < 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(temperature)  2.838  3.614 5.863 0.000261 ***
## s(fuel_price)   4.486  5.631 5.376 3.34e-05 ***
## s(cpi)          7.496  8.250 4.836 2.91e-06 ***
## s(unemployment) 4.726  5.921 7.004 8.81e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.939   Deviance explained =   94%
## GCV = 1.998e+10  Scale est. = 1.9676e+10  n = 5151
```

```
predictions_gam <- predict(walmart_model, test_data)
RMSE(predictions_gam, test_data$weekly_sales)
```

```
## [1] 138804.5
```