Predicting Total Wealth: A Predictive Analysis Using the 1991 SIPP Data

Xueshan (Kevin) Peng

Introduction

Loading and Inspecting the Data

Let's take a look at the first 6 rows of the data.

```r
data <- read.table('data_tr.txt', head = T)[,-1]
head(data)
```

```
##       tw  ira e401  nifa   inc hmort   hval hequity educ male twoearn nohs hs
## 1  53550    0    0   100 28146 60150  69000    8850   12    0       0    0  1
## 2 124635    0    0 61010 32634 20000  78000   58000   16    0       0    0  0
## 3 192949 1800    0  7549 52206 15900 200000  184100   11    1       1    1  0
## 4   -513    0    0  2487 45252     0      0       0   15    0       1    0  0
## 5 212087    0    0 10625 33126 90000 300000  210000   12    0       0    0  1
## 6  24400    0    0  9000 76860 99600 120000   20400   15    0       1    0  0
##   smcol col age fsize marr
## 1     0   0  31     5    1
## 2     0   1  52     5    0
## 3     0   0  50     3    1
## 4     1   0  28     4    1
## 5     0   0  42     3    0
## 6     1   0  49     6    1
```

The variables in this dataset is defined as follows:

- tw: Total wealth (in US $), which is defined as "net financial assets, including Individual Retirement Account (IRA) and 401(k) assets, plus housing equity plus the value of business, property, and motor vehicles."
- ira: individual retirement account (IRA) balance (in US $).
- e401: Binary variable, where 1 indicates eligibility for a 401(k)-retirement plan, and 0 indicates otherwise.
- nifa: Non-401k financial assets (in US $).
- inc: Income (in US $).
- hmort: Home mortgage (in US $).
- hval: Home value (in US $).
- hequity: Home value minus home mortgage.
- educ: Education (in years).
- male: Binary variable, where 1 indicates male and 0 indicates otherwise.
- twoearn: Binary variable, where 1 indicates two earners in the household, and 0 indicates otherwise.

- nohs, hs, smcol, col: Dummy variables for education levels - no high school, high school, some college, college.
- age: Age.
- fsize: Family size.
- marr: Binary variable, where 1 indicates married and 0 indicates otherwise.

```
colSums(is.na(data))
```

```
##      tw     ira    e401    nifa     inc   hmort    hval hequity    educ    male 
##       0       0       0       0       0       0       0       0       0       0 
## twoearn    nohs      hs   smcol     col     age   fsize    marr 
##       0       0       0       0       0       0       0       0
```

```
any(duplicated(data))
```

```
## [1] FALSE
```

We can see that the data is in good shape, where categorical variables are already transformed into dummy variables. We can also see that there exists multi-collinearity in education levels (**nohs**, **hs**, **smcol**, **col**) and home-ownership-related variables (**hmort**, **hval**, and **hequity**).

```
summary(data)
```

```
##        tw                ira               e401             nifa        
##  Min.   :-502302   Min.   :     0   Min.   :0.0000   Min.   :      0  
##  1st Qu.:   3246   1st Qu.:     0   1st Qu.:0.0000   1st Qu.:    200  
##  Median :  25225   Median :     0   Median :0.0000   Median :   1687  
##  Mean   :  63629   Mean   :  3471   Mean   :0.3714   Mean   :  13611  
##  3rd Qu.:  82173   3rd Qu.:     0   3rd Qu.:1.0000   3rd Qu.:   8875  
##  Max.   :1887115   Max.   :100000   Max.   :1.0000   Max.   :1425115  
##       inc             hmort             hval           hequity      
##  Min.   :    -9   Min.   :     0   Min.   :     0   Min.   :-40000  
##  1st Qu.: 19413   1st Qu.:     0   1st Qu.:     0   1st Qu.:     0  
##  Median : 31575   Median :  8000   Median : 50000   Median : 10000  
##  Mean   : 37177   Mean   : 30207   Mean   : 63965   Mean   : 33757  
##  3rd Qu.: 48615   3rd Qu.: 52000   3rd Qu.: 95000   3rd Qu.: 48000  
##  Max.   :242124   Max.   :150000   Max.   :300000   Max.   :300000  
##       educ           male           twoearn            nohs       
##  Min.   : 1.0   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:12.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :12.0   Median :0.0000   Median :0.0000   Median :0.0000  
##  Mean   :13.2   Mean   :0.2018   Mean   :0.3808   Mean   :0.1277  
##  3rd Qu.:15.0   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000  
##  Max.   :18.0   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##        hs             smcol             col              age       
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :25.00  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:32.00  
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :40.00  
##  Mean   :0.3819   Mean   :0.2422   Mean   :0.2482   Mean   :41.08  
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:48.00  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :64.00  
##      fsize            marr       
##  Min.   : 1.00   Min.   :0.0000
##  1st Qu.: 2.00   1st Qu.:0.0000
##  Median : 3.00   Median :1.0000
##  Mean   : 2.87   Mean   :0.6075
##  3rd Qu.: 4.00   3rd Qu.:1.0000
##  Max.   :13.00   Max.   :1.0000
```

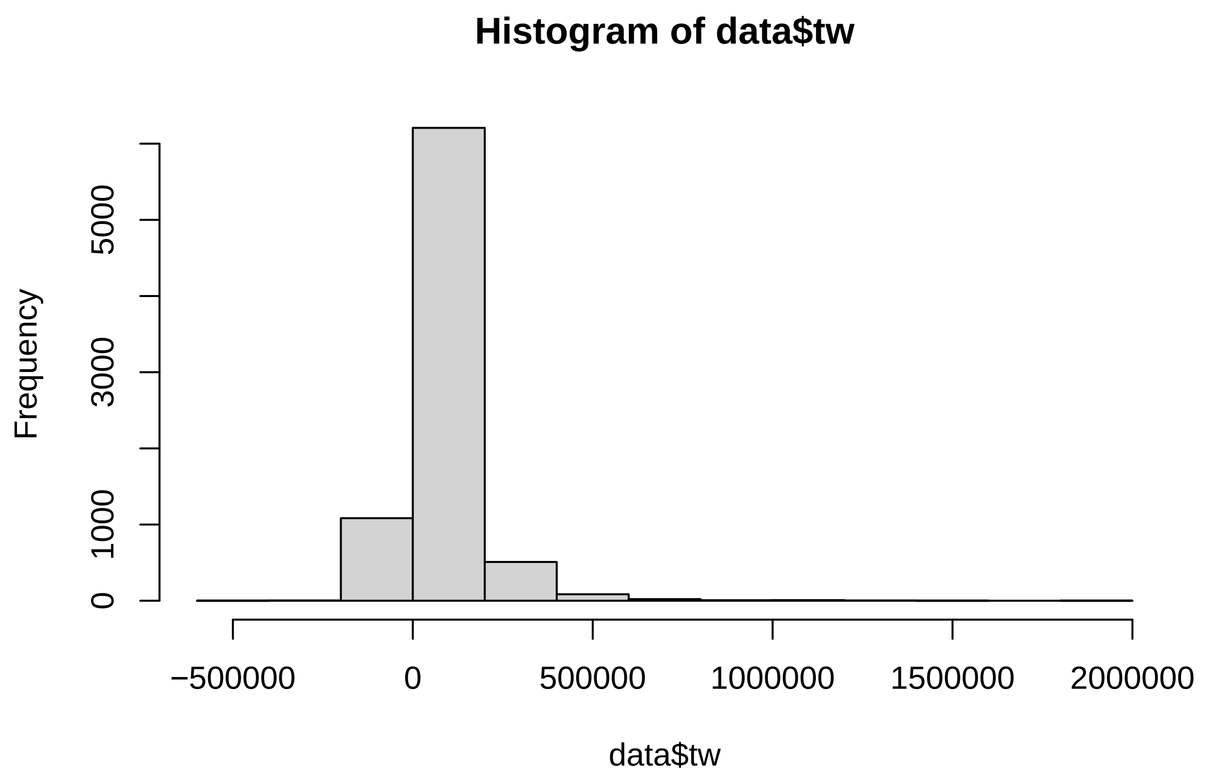
While there exist observations where total wealth is negative, it should be noted that the variable includes home equity, which can be negative, so it does not necessarily indicate that there are incorrect data entries.

The variables **ira**, **nohs**, **smcol**, **col**, and **male** exhibited a value of 0 at the 3rd quantile. They are probably a significant number of data points taking on the value of 0. Since **male** is on the list, it should also be noted that most observations are associated with female participants.

Also, the variable **tw**, **nifa**, **hmort**, and **hequity** have means that are much greater than medians, showing signs of large outliers.

In the histogram below, we can visualize the existence of outliers with enormous wealth.

```r
hist(data$tw)
```



Using the graph, we can determine that removing the outliers with **tw** above $1,000,000 would be appropriate.

```r
data = subset(data, data$tw<1000000)
```

Testing and Removing Multi-collinearity

Let's test whether removing different educational level predictors affect my model's performance, gauged by (MSPE). For simplicity sake, I did not use k-fold cross validation.

```
k <- 10
set.seed(123)
rand <- sample(nrow(data), floor(nrow(data)/k))
train <- setdiff(c(1:nrow(data)), rand)
y_rand <- data$tw[rand]

regnohs <- lm(tw ~ 1 + hs + smcol + col, data = data[train,])
reghs <- lm(tw ~ 1 + nohs + smcol + col, data = data[train,])
regsmcol <- lm(tw ~ 1 + nohs + hs + col, data = data[train,])
regcol <- lm(tw ~ 1 + nohs + hs + smcol, data = data[train,])

prnohs <- predict(regnohs, newdata = data[rand,])
prhs <- predict(reghs, newdata = data[rand,])
prsmcol <- predict(regsmcol, newdata = data[rand,])
prcol <- predict(regcol, newdata = data[rand,])

MSEnohs <- mean((y_rand-prnohs)^2)
MSEhs <- mean((y_rand-prhs)^2)
MSEsmcol <- mean((y_rand-prsmcol)^2)
MSEcol <- mean((y_rand-prcol)^2)

c(MSEnohs, MSEhs, MSEsmcol, MSEcol)
```

```
## [1] 9119936474 9119936474 9119936474 9119936474
```

No difference in performance is found between removing different terms for multi-collinearity. For interpretability, we choose to remove **hs** for education level.

More Feature Selections

Since **hequity** represents home value minus home mortgage, it is intuitively a better predictor of total wealth than **hval** or **hmort** itself. Hence, choosing **hequity** over **hval** and **hmort** is the more sensible choice.

Including years of education (**educ**) along with education levels is redundant. Considering that diplomas are usually much more important than years of education, prioritizing education level over years of education is appropriate.

```
data <- data[, !(names(data) %in% c("hs", "hval", "hmort", "educ"))]
```

Creating a Linear Baseline Model

Using Lasso and Forward/Backward Stepwise Selection

We will strive to create a linear baseline model. This will serve as a baseline to compare to when we later add nonlinear transformations and interaction terms.

For this approach, we are going to include all the features in the dataset. We will let the feature selection algorithms, Lasso and Stepwise Selection, to select the features for us.

For better accuracy, I employed 10-fold cross validation. Leave-one-out cross validation would yield a even more accurate result, but doing so on a dataset containing 7919 observations would take too much computational power.

```r
source("KfoldCVFunctions.R")

mean_mspe_lasso <- compute_lasso_mspe(data, response_var = "tw")
mean_mspe_forward <- compute_forward_stepwise_mspe(data, response_var = "tw")
mean_mspe_backward <- compute_backward_stepwise_mspe(data, response_var = "tw")

# Print results
print(c(Lasso = mean_mspe_lasso,
        Forward_Stepwise = mean_mspe_forward,
        Backward_Stepwise = mean_mspe_backward))
```

```
##             Lasso  Forward_Stepwise Backward_Stepwise
##        1427662119        1427849850        1427849850
```

As shown in the results above, Lasso yielded a lower MSPE than forward or backward stepwise selection.

Let's now inspect the coefficients that Lasso chose and their associated p-values.

Since Lasso performs both variable selection and shrinkage, leading to biased coefficient estimates, traditional significance tests for coefficients (like p-values) are not straightforwardly available. Therefore, we use the *hdi* (High Dimensional Inference) package to approximate the p-values.

```r
library(hdi)

response_var <- "tw"
y <- data[[response_var]]
X <- as.matrix(data[ , !(names(data) %in% response_var)])

lasso_cv <- cv.glmnet(X, y, alpha = 1)
best_lambda <- lasso_cv$lambda.min

lasso_model <- glmnet(X, y, lambda = best_lambda, alpha = 1)

lasso_inference <- hdi::lasso.proj(X, y)

print(coef(lasso_model))
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -1.576371e+04
## ira          1.606418e+00
## e401         7.935739e+03
## nifa         1.102051e+00
## inc          2.500733e-01
## hequity      1.081333e+00
## male         3.169643e+03
## twoearn     -5.386380e+03
## nohs          .
## smcol         1.006324e+03
## col           .
## age           2.470886e+02
## fsize         .
## marr          1.473061e+03
```

As shown above, Lasso has selected all of the features except nohs, col, and fsize. The coefficients are shown above.

```r
print(lasso_inference$pval)
```

```
##           ira          e401          nifa           inc       hequity
## 5.152649e-198  1.807363e-17  0.000000e+00  1.982564e-23  0.000000e+00
##          male       twoearn          nohs         smcol           col
##  1.683754e-03  1.802943e-07  8.852471e-01  2.187133e-01  8.723161e-01
##           age         fsize          marr
##  1.172631e-07  7.921987e-01  5.334928e-02
```

Above are the approximated p-values the coefficients. We can gauge how strong of a predictor each feature is. As expected, ira and e401 have a really small p-value as they are literally a part of **tw**.

- tw: Total wealth (in US $), which is defined as "net financial assets, including Individual Retirement Account (IRA) and 401(k) assets, plus housing equity plus the value of business, property, and motor vehicles."

Let's also compute the MSPE for a simple OLS regression model and for a ridge regression model with the selected features for comparison.

```r
data_subset <- data[, !(names(data) %in% c("nohs", "col", "fsize"))]

source("KfoldCVFunctions.R")
mean_mspe_ols <- compute_ols_mspe(data_subset, response_var = "tw")
mean_mspe_ridge <- compute_ridge_mspe(data_subset, response_var = "tw")

print(c(
  OLS = mean_mspe_ols,
  Ridge = mean_mspe_ridge,
  Lasso = mean_mspe_lasso,
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
##               OLS            Ridge            Lasso  Forward_Stepwise
##        1427051036       1449149272       1427662119        1427849850
## Backward_Stepwise
##        1427849850
```

Surprisingly, a simple OLS regression on selected features yielded better results than ridge regression, Lasso, forward/backward stepwise selection.

The mean MSPE of our OLS regression will than serve as our benchmark, which we will strive to improve upon when fitting nonlinear transformations.

```
benchmark = mean_mspe_ols
```

Finding Nonlinear Relationships and Applying Transformations

Inspecting the Relationships between tw and Features

After creating an appropriate linear model, the appropriate next step to improve predictive performance is through applying nonlinear transformations.

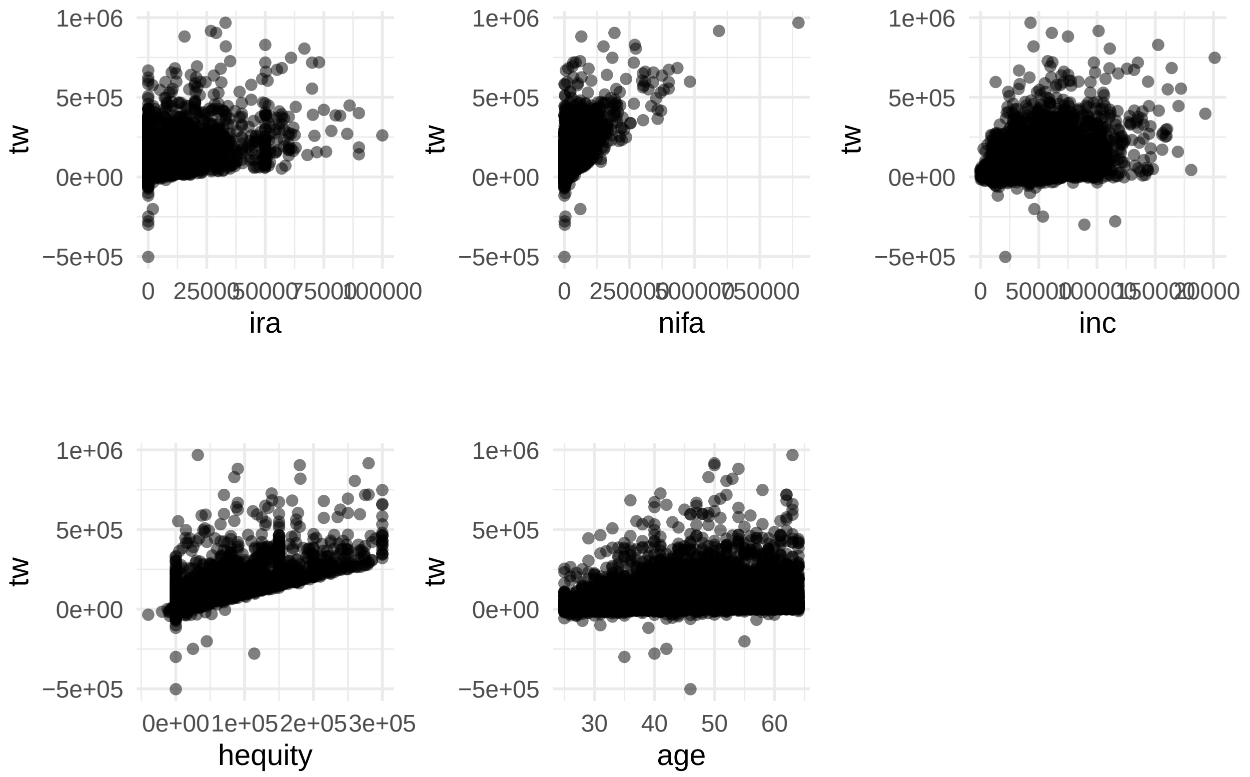
Let's first inspect the relationships between **tw** and all the quantitative (non-binary) features in the dataset. Nonlinear transformations are inappropriate for binary features because they only have two distinct values, making such transformations ineffective and potentially meaningless.

The below function compiles all the scatterplots into a list of *ggplot* objects.

```
library(tidyverse)
library(gridExtra)

plot_scatterplots <- function(data, response_var, feature_names) {

  plot_list <- list()

  for (feature in feature_names) {
    p <- ggplot(data = data) +
      geom_point(mapping = aes(x = !!sym(feature), y = !!sym(response_var)), alpha = 0.5) +
      labs(x = feature, y = response_var) +
      theme_minimal() +
      theme(aspect.ratio = 1)
    plot_list[[feature]] <- p
  }

  return(plot_list)
}
```

We can then call the function and use the *gridExtra* package to arrange them.

```
plots <- plot_scatterplots(data_subset, response_var = "tw", c("ira", "nifa", "inc", "hequity", "age"))

do.call(grid.arrange, c(plots, ncol = 3))
```



Since **tw** is defined as net financial assets including **ira**, it is expected that there is a linear relationship between **tw** and **ira**, and this is what we can see on the scatterplot, too. The same goes with **hequity**.

Similarly, **nifa** is defined as non-401k financial assets and has a seemingly linear relationship with **tw** despite it having a more data points on the lower end than the higher end of **nifa**.

One that stands out as being nonlinear is the relationship between **age** and **tw**. In the area where there are more data points, represented by a darker color, there seems to be a sharper increase in **tw** at a younger age.

Applying Nonlinear Transformations to age

For the following reasons, I will be using cubic splines as opposed to polynomials.

- Polynomials must use a high degree for flexible fits, but splines are able to do so with the degree fixed. This is likely to produce more stable estimates.
- Polynomials lack the ability to incorporate thresholds like splines, leading to undesirably global outcomes. In other words, observations within one range of the predictor strongly influence the model's behavior across different ranges.
- A polynomial fit is likely to produce undesirable results at the boundaries.

I have constructed a function to find the optimal number of knots and also where to place them. The function returns a set of knots, boundary knots (endpoints of the feature), as well as a plot showing the MSE vs. Number of Knots.

```r
source("select_optimal_knots.R")

result = select_optimal_knots(data, "tw", "age", 20)
print(result$knots_chosen)
```

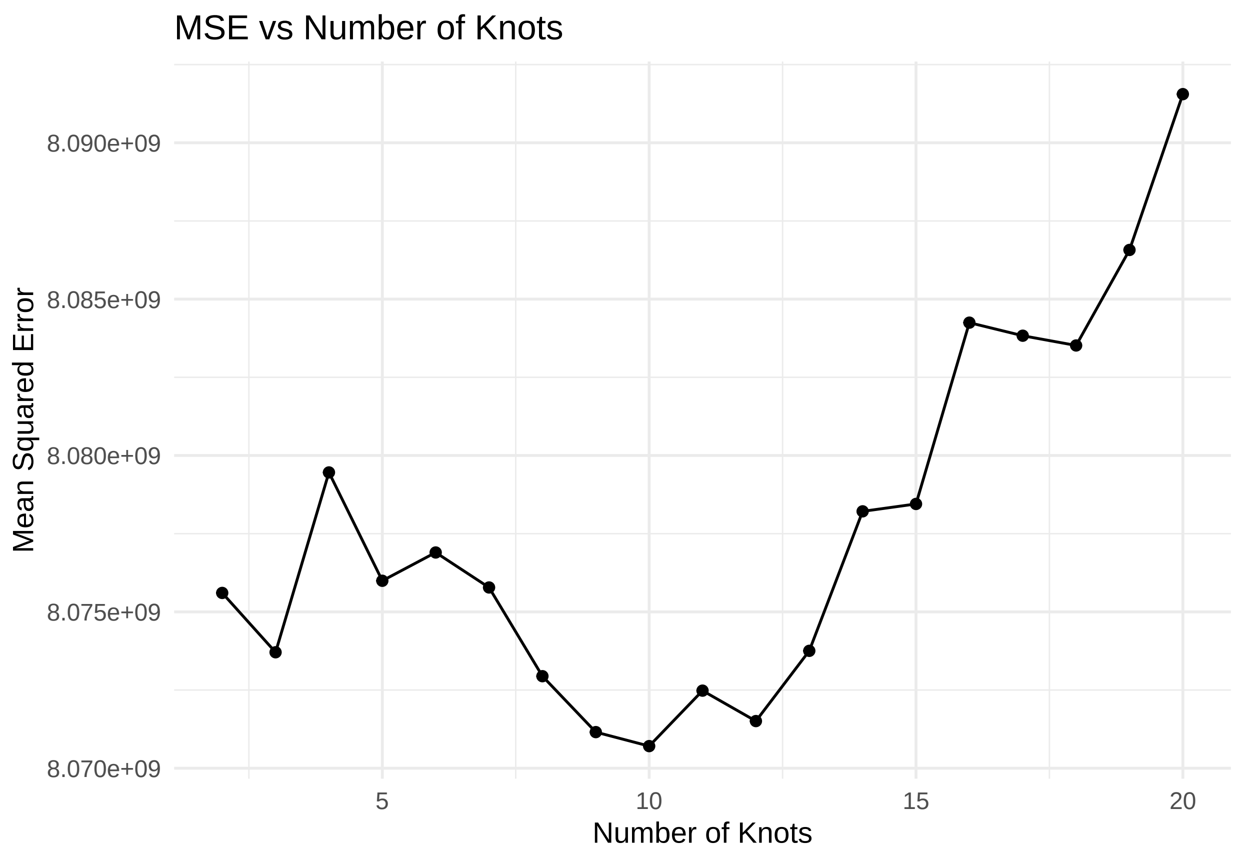
```
##  [1] 28.54545 32.09091 35.63636 39.18182 42.72727 46.27273 49.81818 53.36364
##  [9] 56.90909 60.45455
```

We can see that the algorithms returns the optimal set of knots, which, in this case, contains 10 items.

```r
print(result$boundary_knots)
```

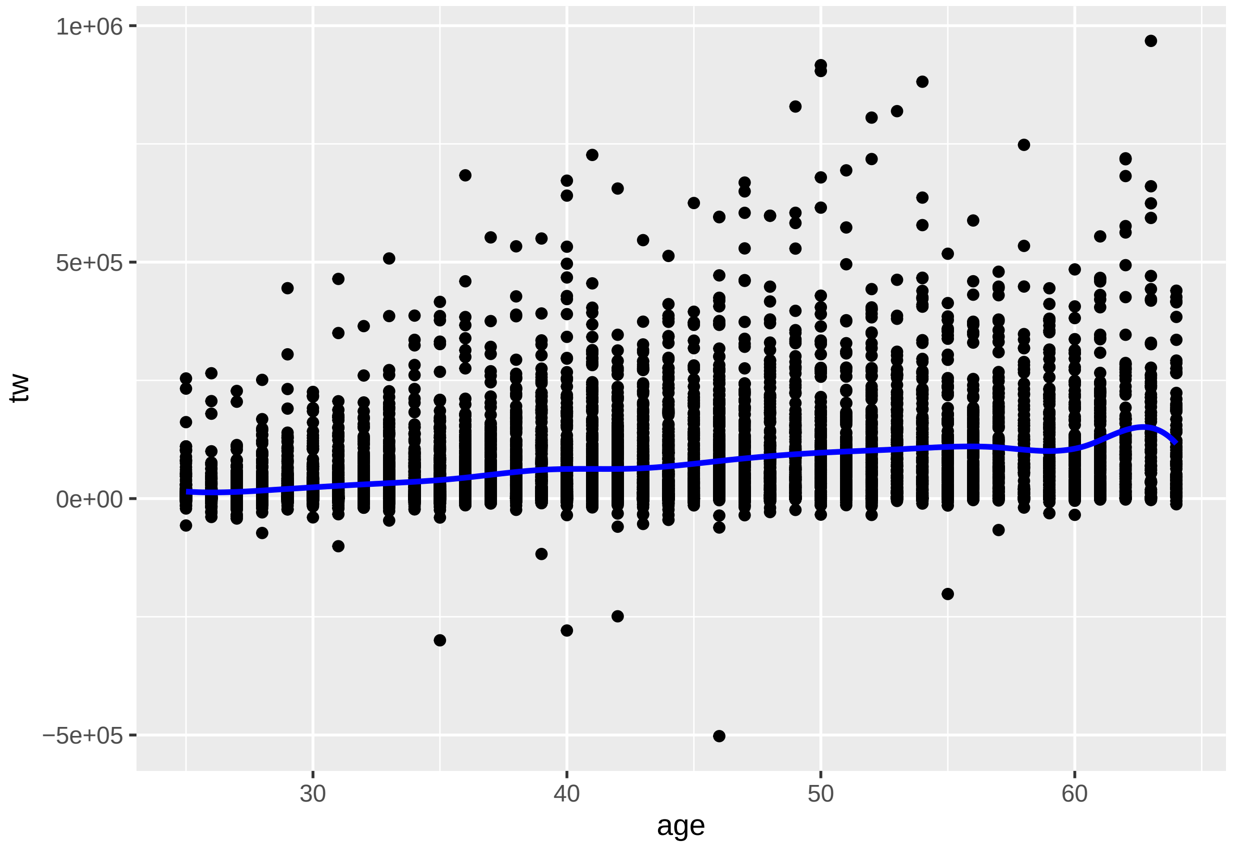
```
## [1] 25 64
```

```r
print(result$plot)
```



We can see through the visualization that the MSE is the lowest when the number of knots is 10.

```r
age_spline_model <- lm(tw ~ bs(age,
                                knots = result$knots_chosen,
                                Boundary.knots = result$boundary_knots),
                         data = data)

x.grid <- seq(from = result$boundary_knots[1],
               to = result$boundary_knots[2],
               length.out = 1000)
pr <- predict(age_spline_model, newdata = data.frame(age = x.grid))

# Create a new data frame for the predictions
predictions <- data.frame(age = x.grid, tw = pr)

# Plot the data and the fitted spline curve
ggplot(data = data) +
  geom_point(mapping = aes(x = age, y = tw)) +
  geom_line(data = predictions,
            mapping = aes(x = age, y = tw),
            lwd = 1,
            color = "blue")
```



We can then use the previous output to construct a cubic spline model, from which predictions can be obtained and visualized through the graph.

However, the visualization is suggesting that there might be some overfitting of the data, as there is no reason for wealth to have a surge of increase at an older age. We will validate this through the below analysis.

```r
source("spline_transform_dataset.R")

transformed_data <- spline_transform_dataset(data, "age", result$knots_chosen, result$boundary_knots)
transformed_data_sub <- spline_transform_dataset(data_subset, "age", result$knots_chosen, result$bounda
source("KfoldCVFunctions.R")

mean_mspe_lasso <- compute_lasso_mspe(transformed_data, response_var = "tw")
mean_mspe_forward <- compute_forward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_backward <- compute_backward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_ols <- compute_ols_mspe(transformed_data_sub, response_var = "tw")
mean_mspe_ridge <- compute_ridge_mspe(transformed_data_sub, response_var = "tw")

print(c(
  OLS_without_transformations = benchmark,
  OLS = mean_mspe_ols,
  Ridge = mean_mspe_ridge,
  Lasso = mean_mspe_lasso,
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
## OLS_without_transformations                         OLS
##                  1427051036                  1430609484
##                       Ridge                       Lasso
##                  1452436233                  1431052699
##            Forward_Stepwise           Backward_Stepwise
##                  1432084379                  1432221028
```

Applying the transformation to **age** and incorporating it into the dataset, we can now use General Additive Method (GAM) to compare the model's performance with the benchmark.

Unfortunately, the original linear model yielded a better performance, suggesting that the relationship between **tw** and **age** is approximately linear.

Applying Nonlinear Transformations to inc

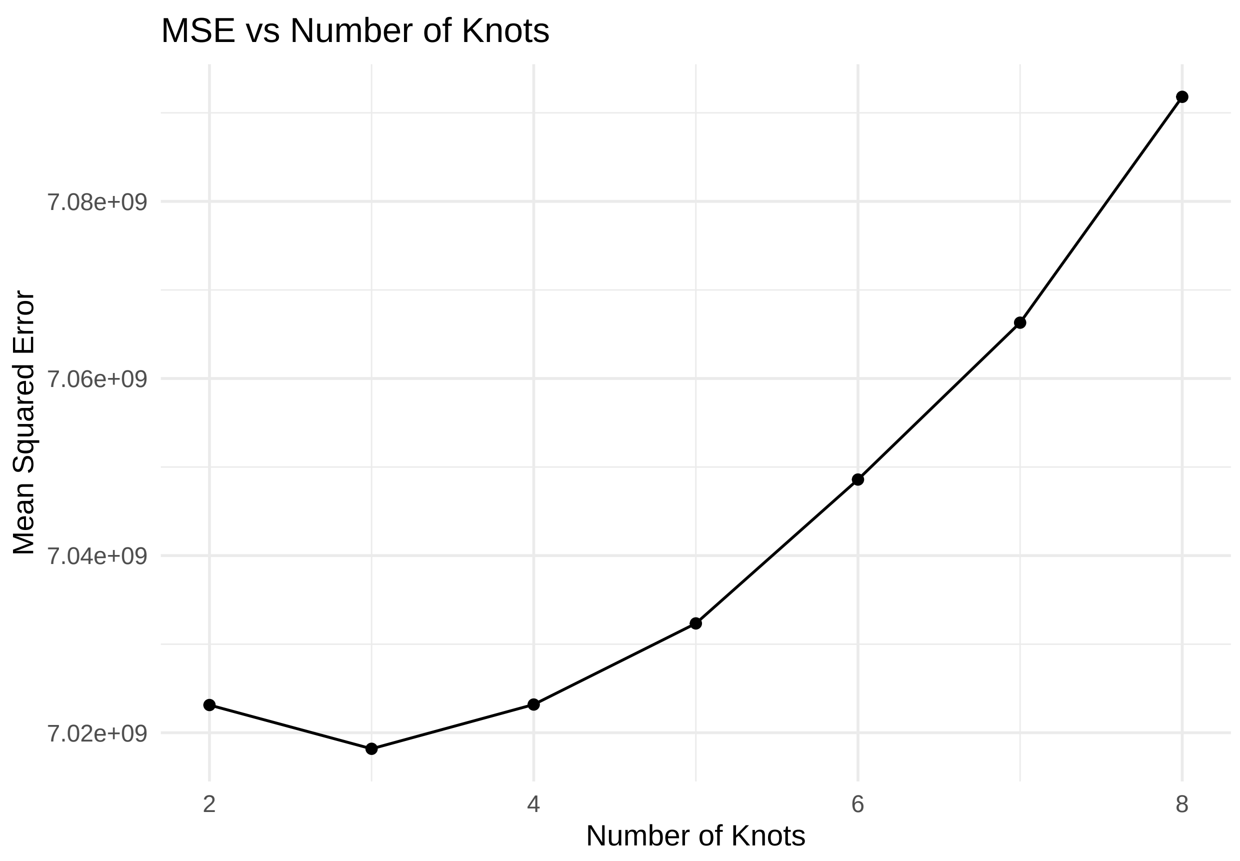
```r
source("select_optimal_knots_natural.R")

result = select_optimal_knots_natural(data, "tw", "inc", 8)
print(result$knots_chosen)
```

```
## [1]  50242.5 100494.0 150745.5
```

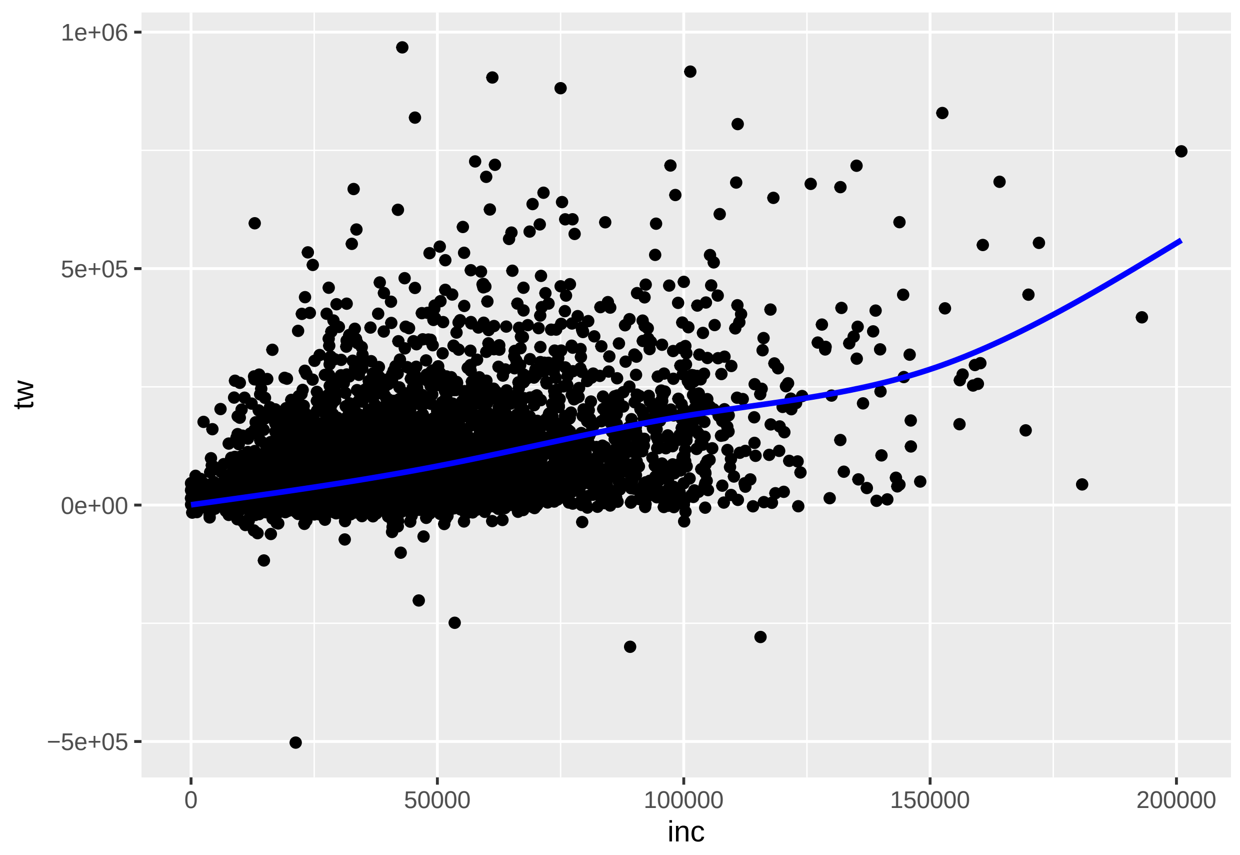
```r
print(result$boundary_knots)
```

```
## [1]     -9 200997
```

```r
print(result$plot)
```



```
inc_spline_model <- lm(tw ~ ns(inc,
                               knots = result$knots_chosen,
                               Boundary.knots = result$boundary_knots),
                       data = data)

x.grid <- seq(from = result$boundary_knots[1],
              to = result$boundary_knots[2],
              length.out = 1000)
pr <- predict(inc_spline_model, newdata = data.frame(inc = x.grid))

# Create a new data frame for the predictions
predictions <- data.frame(inc = x.grid, tw = pr)

# Plot the data and the fitted spline curve
ggplot(data = data) +
  geom_point(mapping = aes(x = inc, y = tw)) +
  geom_line(data = predictions,
            mapping = aes(x = inc, y = tw),
            lwd = 1,
            color = "blue")
```



```
source("natural_spline_transform_dataset.R")

transformed_data <- natural_spline_transform_dataset(data, "inc", result$knots_chosen, result$boundary_
transformed_data_sub <- natural_spline_transform_dataset(data_subset, "inc", result$knots_chosen, resul

source("KfoldCVFunctions.R")

mean_mspe_lasso <- compute_lasso_mspe(transformed_data, response_var = "tw")
mean_mspe_forward <- compute_forward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_backward <- compute_backward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_ols <- compute_ols_mspe(transformed_data_sub, response_var = "tw")
mean_mspe_ridge <- compute_ridge_mspe(transformed_data_sub, response_var = "tw")

print(c(
  OLS_without_transformations = benchmark,
  OLS = mean_mspe_ols,
  Ridge = mean_mspe_ridge,
  Lasso = mean_mspe_lasso,
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
## OLS_without_transformations                         OLS
##                  1427051036                  1430497167
##                       Ridge                       Lasso
##                  1449872475                  1429314763
##            Forward_Stepwise            Backward_Stepwise
##                  1429742512                   1429742512
```