Homework 4: Moving Beyond Linearity

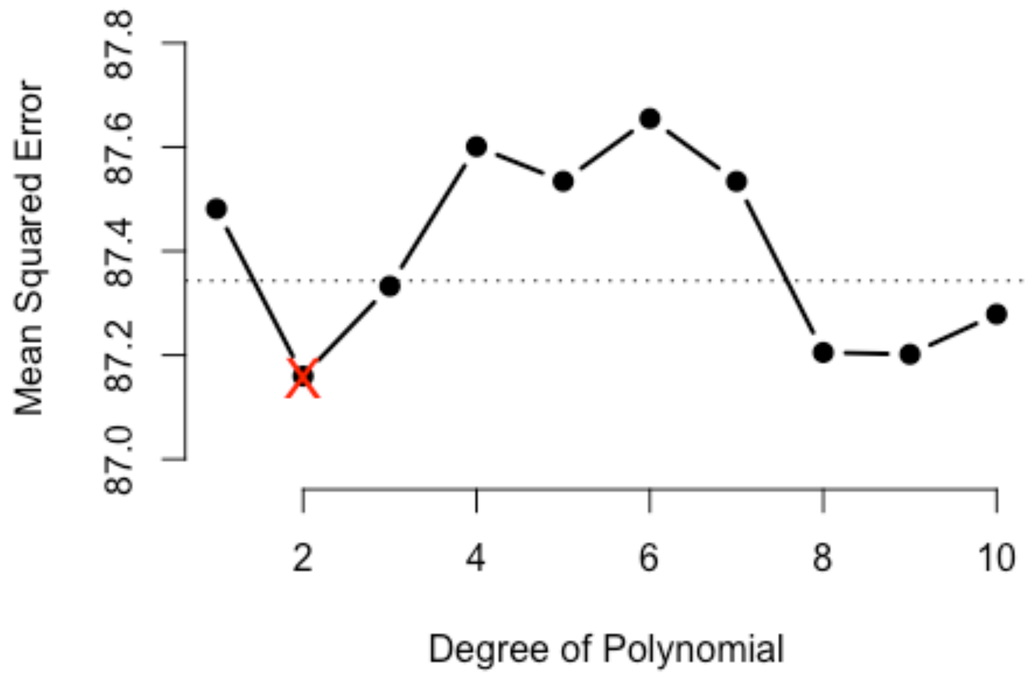
Luying Jiang

```
gss_train <- read.csv('~/Desktop/problem-set-4-master/data/gss_train.csv')
gss_test <- read.csv('~/Desktop/problem-set-4-master/data/gss_test.csv')
```

1. Perform polynomial regression to predict egalit_scale as a function of income06. Use and plot 10-fold cross-validation to select the optimal degree d for the polynomial based on the MSE. Plot the resulting polynomial fit to the data, and also graph the average marginal effect (AME) of income06 across its potential values. Be sure to provide substantive interpretation of the results.

```
library(ISLR)
library(boot)
library(caret)

library(tidyverse)

library(broom)
library(ggthemes)
library(knitr)
library(kableExtra)

##
## Attaching package: 'kableExtra'

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

set.seed(123)
cv.MSE <- NA
for (i in 1:10) {
  glm.fit <-  glm(egalit_scale ~ poly(income06, i), data = gss_train)
  cv.MSE[i] <-  cv.glm(gss_train, glm.fit, K = 10)$delta[1]
}
cv.MSE

##  [1] 87.48125 87.15967 87.33259 87.60095 87.53414 87.65479 87.53396 87.205
07
##  [9] 87.20160 87.27881

plot( x = 1:10, y = cv.MSE, xlab = "Degree of Polynomial", ylab = "Mean Squar
ed Error",
      type = "b", lwd = 2, pch= 19, bty = "n",
      ylim = c( min(cv.MSE) - sd(cv.MSE), max(cv.MSE) + sd(cv.MSE) ) )
```

```r
# horizontal line for 1se to less complexity
abline(h = min(cv.MSE) + sd(cv.MSE) , lty = "dotted")

# where is the minimum
points( x = which.min(cv.MSE), y = min(cv.MSE), col = "red", pch = "X", cex =
 1.5 )
```

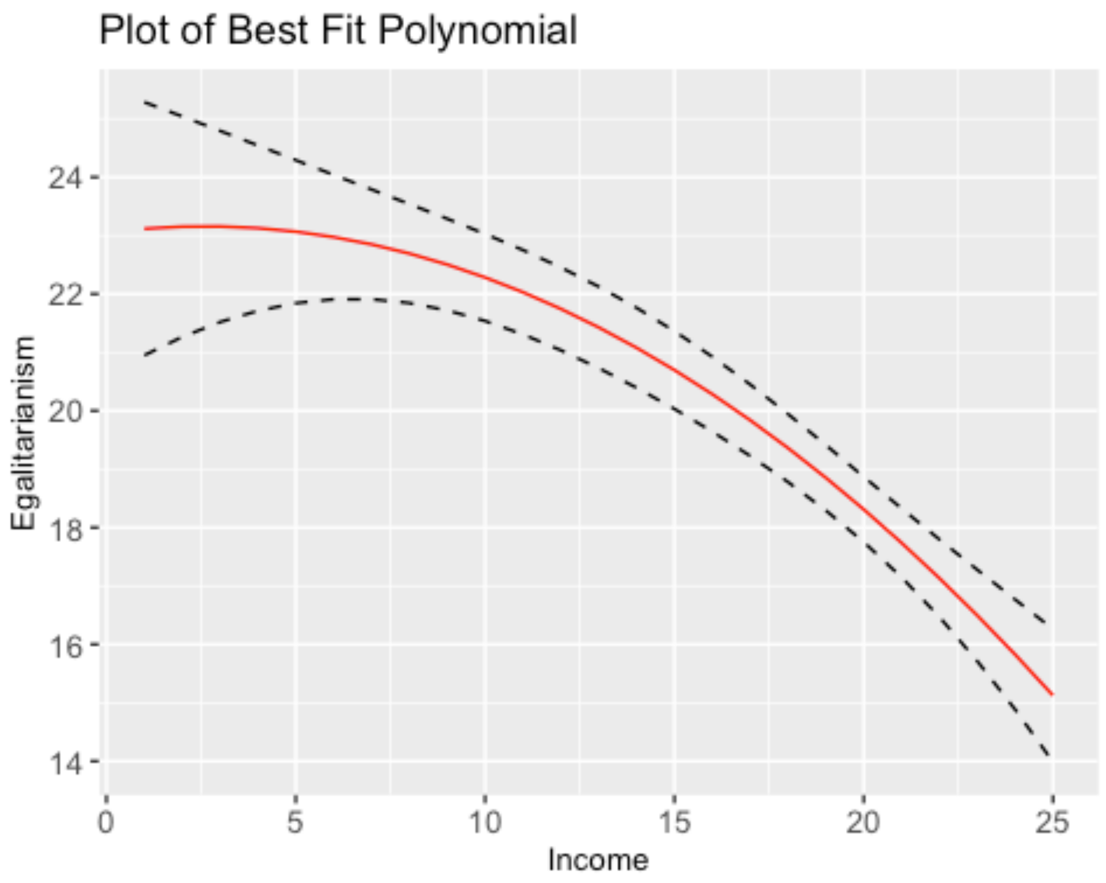


The optimal degree for the polynomial regression is 2.

```r
library(margins)
best_poly <- lm(egalit_scale ~ income06 + I(income06^2), data = gss_train)
margin <- margins(best_poly)
margin

## Average marginal effects

## lm(formula = egalit_scale ~ income06 + I(income06^2), data = gss_train)

##  income06
##   -0.4507

plot_best_poly <- cplot(best_poly, "income06",draw=FALSE)
```

```
##    xvals    yvals    upper    lower
## 1      1 23.11631 25.28371 20.94890
## 2      2 23.15180 25.04001 21.26359
## 3      3 23.15524 24.79232 21.51817
## 4      4 23.12665 24.54195 21.71134
## 5      5 23.06600 24.29036 21.84165
## 6      6 22.97331 24.03899 21.90764
## 7      7 22.84858 23.78870 21.90846
## 8      8 22.69180 23.53894 21.84467
## 9      9 22.50298 23.28678 21.71917
## 10    10 22.28211 23.02670 21.53752
## 11    11 22.02920 22.75132 21.30708
## 12    12 21.74424 22.45297 21.03552
## 13    13 21.42724 22.12502 20.72946
## 14    14 21.07819 21.76262 20.39376
## 15    15 20.69710 21.36290 20.03130
## 16    16 20.28396 20.92501 19.64291
## 17    17 19.83878 20.45044 19.22712
## 18    18 19.36155 19.94356 18.77955
## 19    19 18.85228 19.41247 18.29210
## 20    20 18.31096 18.86919 17.75274
```

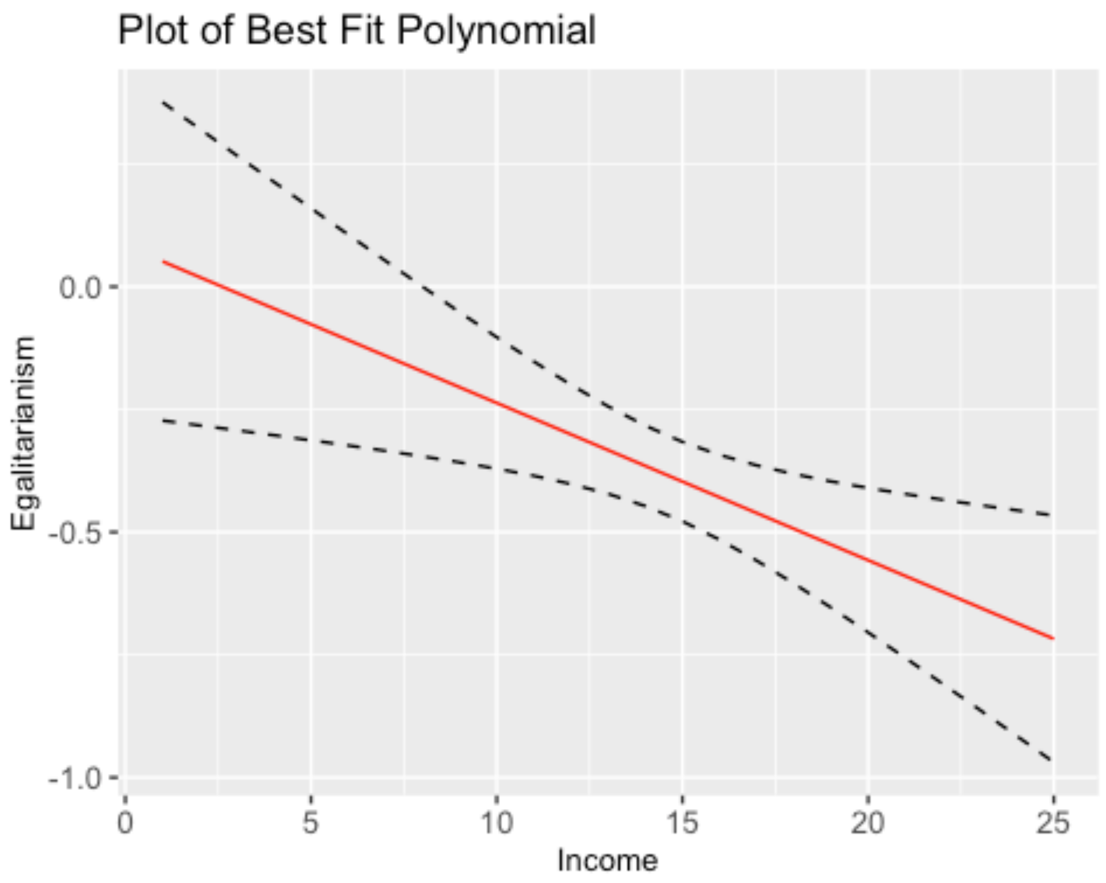
```
ggplot(plot_best_poly, aes(x = xvals)) +
  geom_line(aes(y = yvals), color = 'red') +
  geom_line(aes(y = upper), linetype = 2) +
  geom_line(aes(y = lower), linetype = 2) +
  labs(x = "Income", y = "Egalitarianism",
       title = "Plot of Best Fit Polynomial") +
  theme(axis.text = element_text(size = 10),
        axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10))
```



```
plot_best_poly <- cplot(best_poly, "income06", what ='effect',draw=FALSE)
plot_best_poly
```

```
##    xvals   yvals   upper   lower   factor
##   1.0000  0.0515  0.3759 -0.2729 income06
##   2.0000  0.0195  0.3216 -0.2827 income06
##   3.0000 -0.0126  0.2674 -0.2926 income06
##   4.0000 -0.0446  0.2134 -0.3027 income06
##   5.0000 -0.0767  0.1596 -0.3129 income06
##   6.0000 -0.1087  0.1061 -0.3235 income06
##   7.0000 -0.1408  0.0529 -0.3344 income06
##   8.0000 -0.1728  0.0002 -0.3458 income06
##   9.0000 -0.2048 -0.0519 -0.3578 income06
##  10.0000 -0.2369 -0.1029 -0.3708 income06
##  11.0000 -0.2689 -0.1526 -0.3853 income06
##  12.0000 -0.3010 -0.2000 -0.4020 income06
##  13.0000 -0.3330 -0.2440 -0.4221 income06
##  14.0000 -0.3651 -0.2831 -0.4470 income06
##  15.0000 -0.3971 -0.3162 -0.4781 income06
##  16.0000 -0.4292 -0.3428 -0.5155 income06
##  17.0000 -0.4612 -0.3642 -0.5583 income06
##  18.0000 -0.4932 -0.3817 -0.6048 income06
##  19.0000 -0.5253 -0.3967 -0.6538 income06
##   20.0000 -0.5573 -0.4101 -0.7045 income06
##   21.0000 -0.5894 -0.4224 -0.7563 income06
##   22.0000 -0.6214 -0.4340 -0.8089 income06
##   23.0000 -0.6535 -0.4450 -0.8619 income06
##   24.0000 -0.6855 -0.4557 -0.9154 income06
##   25.0000 -0.7176 -0.4660 -0.9691 income06
```

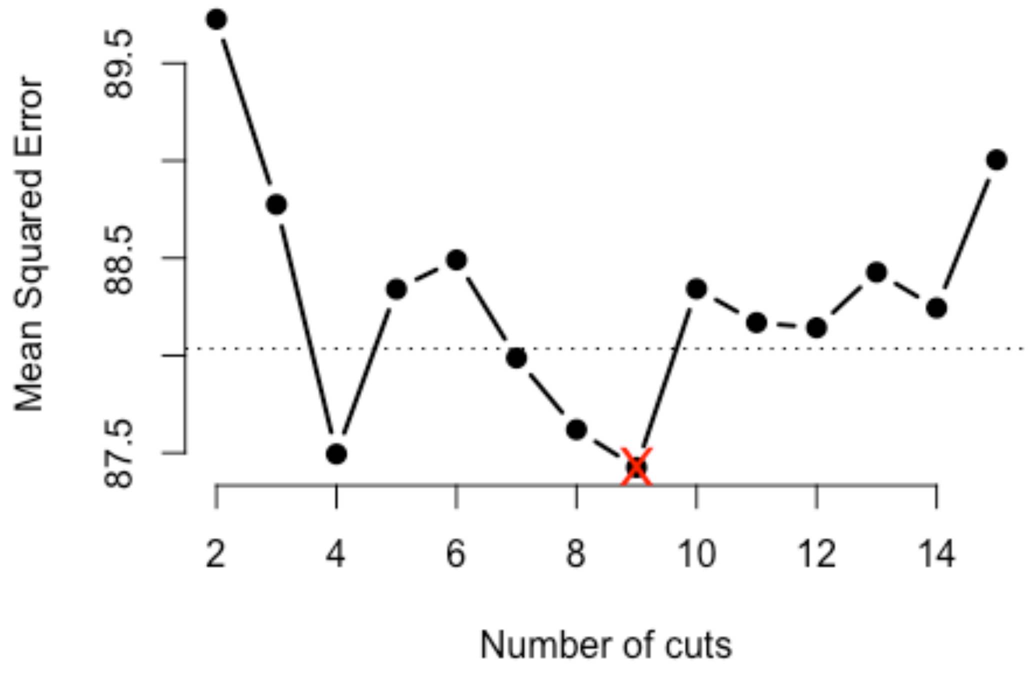
```
ggplot(plot_best_poly, aes(x = xvals)) +
  geom_line(aes(y = yvals), color = 'red') +
  geom_line(aes(y = upper), linetype = 2) +
  geom_line(aes(y = lower), linetype = 2) +
  labs(x = "Income", y = "Egalitarianism",
       title = "Plot of Best Fit Polynomial") +
  theme(axis.text = element_text(size = 10),
        axis.title.x = element_text(size = 10),
        axis.title.y = element_text(size = 10))
```



The marginal effect of income level on egalitarianism is largely negative over different income levels, which indicates that as individuals make more money, they are less likely to have egalitarian preferences. The AME for income06 is -0.4507319.

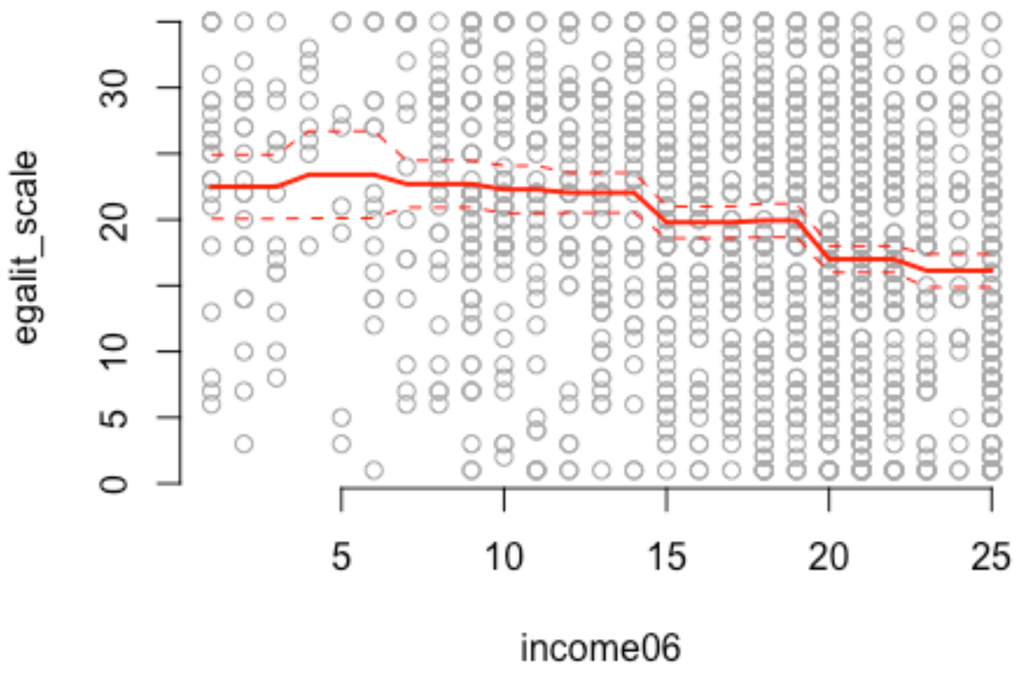
2. Fit a step function to predict egalit_scale as a function of income06, and perform 10-fold cross-validation to choose the optimal number of cuts. Plot the fit and interpret the results.

```r
cv.MSE <-  NA
# for each cut perform 10-fold cross-validation
for (i in 2:15) {
  gss_train$income06.cut <-  cut(gss_train$income06, i)
  gss_test$income06.cut <-  cut(gss_test$income06, i)
  slm.fit <-  glm(egalit_scale ~ income06.cut, data = gss_train)
  cv.MSE[i] <-  cv.glm(gss_train, slm.fit, K = 10)$delta[1]
}

# the first element of cv.MSE is NA because we started our loop at 2
plot(2:15, cv.MSE[-1], xlab = "Number of cuts", ylab = "Mean Squared Error",
     type = "b", pch = 19, lwd = 2, bty ="n")
abline(h = min(cv.MSE, na.rm = TRUE) + sd(cv.MSE, na.rm = TRUE) , lty = "dott
ed")

# highlight minimum
points( x = which.min(cv.MSE), y = min(cv.MSE, na.rm = TRUE), col = "red", pc
h = "X", cex = 1.5 )
```



From the above graph, the optimal number of steps is 9. (optimal number of bins = 9)

```r
lm.fit <- glm(egalit_scale ~ cut(income06, 9), data = gss_train)
incomelims <-range(gss_train$income06)
income.grid <- seq(from = incomelims[1], to = incomelims[2])
lm.pred <-predict(lm.fit, data.frame(income06 = income.grid), se = TRUE)
plot(egalit_scale ~ income06, data = gss_train, col = "darkgrey", bty = "n")
lines(income.grid, lm.pred$fit, col = "red", lwd = 2)
matlines(income.grid, cbind( lm.pred$fit + 2* lm.pred$se.fit,
                          lm.pred$fit - 2* lm.pred$se.fit),
         col = "red", lty ="dashed")
```



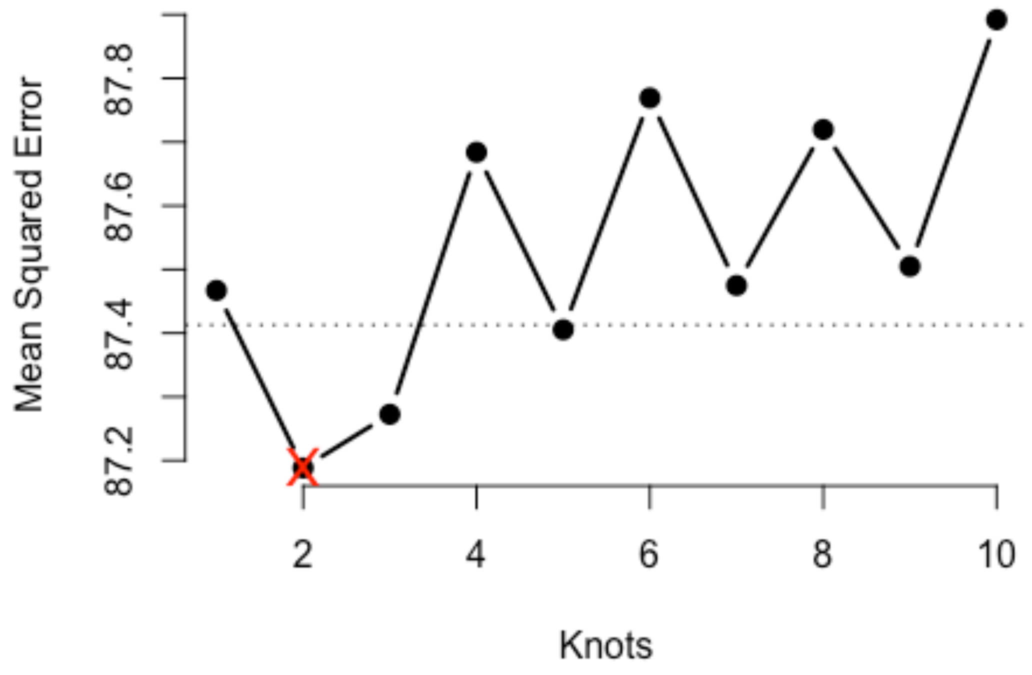
There are four bins of income06. As we can see above the predicted valuedecreases as income06 decreases.

3. Fit a natural regression spline to predict egalit_scale as a function of income06. Use 10-fold cross-validation to select the optimal number of degrees of freedom, and present the results of the optimal model.

```r
library(splines)
cv.MSE <-  NA
for (i in 1:10) {
  glm.fit <- glm(egalit_scale ~ ns(income06, df = i), data = gss_train)
  cv.MSE[i] <- cv.glm(gss_train, glm.fit, K = 10)$delta[1]
}
table <- cbind(1:10, cv.MSE) %>%
  as.data.frame() %>%
  arrange(cv.MSE)
table
```

```
##    V1   cv.MSE
## 1   2 87.18831
## 2   3 87.27250
## 3   5 87.40533
## 4   1 87.46702
## 5   7 87.47478
## 6   9 87.50475
## 7   4 87.68397
## 8   8 87.71962
## 9   6 87.76919
## 10 10 87.89205
```

```
plot(1:10, cv.MSE, xlab = "Knots", ylab = "Mean Squared Error",
     type = "b", pch = 19, lwd = 2, bty ="n")
abline(h = min(cv.MSE, na.rm = TRUE) + sd(cv.MSE, na.rm = TRUE) , lty = "dott
ed")

# highlight minimum
points( x = which.min(cv.MSE), y = min(cv.MSE, na.rm = TRUE), col = "red", pc
h = "X", cex = 1.5 )
```



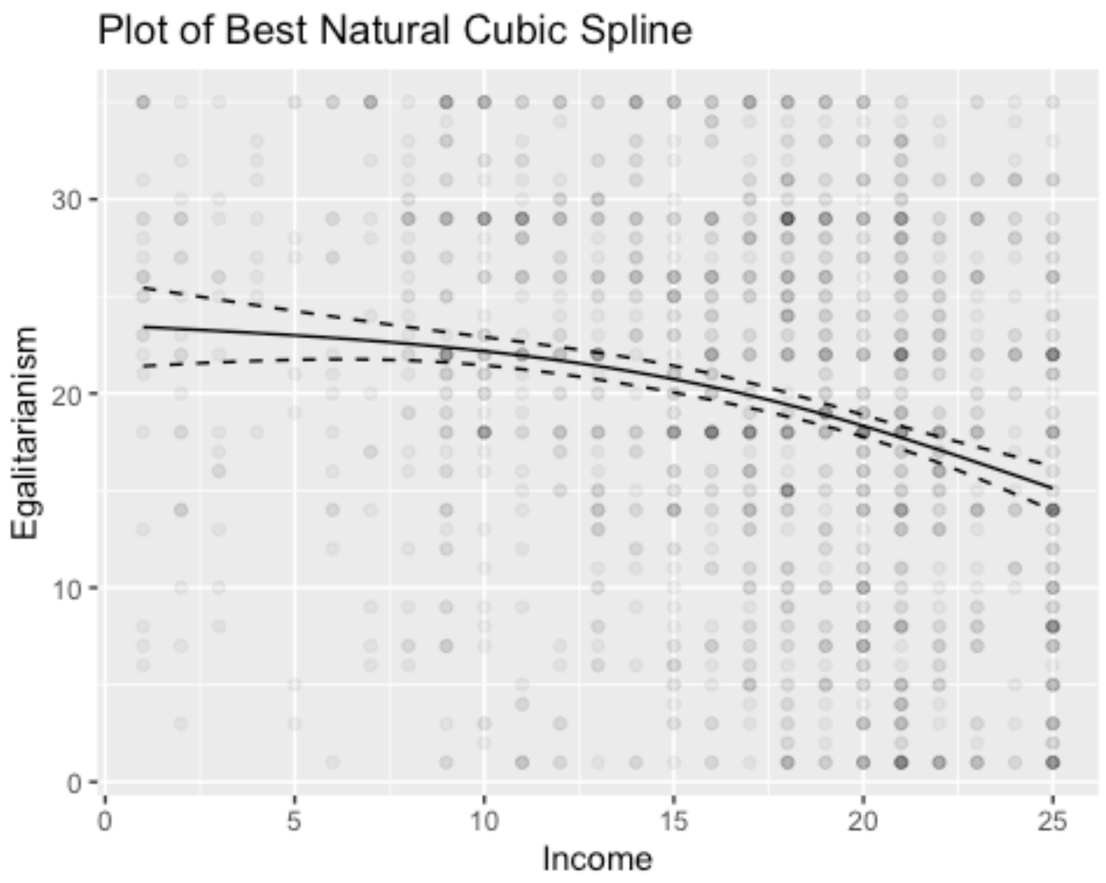
From above, the optimal number of knots is 2 knots.

```
best_natreg <- lm(egalit_scale ~ ns(income06, df = 2), data = gss_train)
plot_best_natreg <- cplot(best_natreg, "income06", draw=FALSE)
```

```
##    xvals    yvals    upper    lower
## 1      1 23.42271 25.43556 21.40987
## 2      2 23.32167 25.13244 21.51090
## 3      3 23.21783 24.83318 21.60248
## 4      4 23.10840 24.53808 21.67872
## 5      5 22.99058 24.24784 21.73332
## 6      6 22.86158 23.96357 21.75959
## 7      7 22.71860 23.68669 21.75050
## 8      8 22.55884 23.41854 21.69914
## 9      9 22.37951 23.15925 21.59977
## 10    10 22.17782 22.90625 21.44938
## 11    11 21.95096 22.65303 21.24888
## 12    12 21.69614 22.38954 21.00274
## 13    13 21.41057 22.10412 20.71702
## 14    14 21.09144 21.78564 20.39725
## 15    15 20.73597 21.42475 20.04720
## 16    16 20.34136 21.01450 19.66822
## 17    17 19.90481 20.55081 19.25881
## 18    18 19.42353 20.03365 18.81340
## 19    19 18.89631 19.47165 18.32097
## 20    20 18.32835 18.88975 17.76695
```

```
glm(egalit_scale ~ ns(income06, df = 2), data = gss_train) %>%
    cplot("income06", what = "prediction", n = 100, draw = FALSE) %>%
    ggplot(aes(x = xvals)) +
    geom_line(aes(y = yvals)) + # predicted value
    geom_line(aes(y = upper), linetype = 2) + # upper SE
    geom_line(aes(y = lower), linetype = 2) + # lower SE
    geom_point(data = gss_train, aes(income06, egalit_scale), alpha = 0.05) +
labs(x = "Income",
     y = "Egalitarianism",
title = "Plot of Best Natural Cubic Spline")
```

```
##        xvals    yvals    upper    lower
## 1  1.000000 23.42271 25.43556 21.40987
## 2  1.242424 23.39832 25.36173 21.43492
## 3  1.484848 23.37390 25.28812 21.45967
## 4  1.727273 23.34939 25.21474 21.48404
## 5  1.969697 23.32476 25.14157 21.50794
## 6  2.212121 23.29997 25.06864 21.53131
## 7  2.454545 23.27499 24.99593 21.55404
## 8  2.696970 23.24976 24.92345 21.57607
## 9  2.939394 23.22426 24.85121 21.59731
## 10 3.181818 23.19843 24.77920 21.61766
## 11 3.424242 23.17225 24.70745 21.63705
## 12 3.666667 23.14567 24.63595 21.65539
## 13 3.909091 23.11865 24.56472 21.67258
## 14 4.151515 23.09116 24.49377 21.68854
## 15 4.393939 23.06314 24.42311 21.70317
## 16 4.636364 23.03457 24.35276 21.71638
## 17 4.878788 23.00540 24.28273 21.72807
## 18 5.121212 22.97560 24.21304 21.73815
## 19 5.363636 22.94511 24.14371 21.74652
## 20 5.606061 22.91391 24.07475 21.75307
```



From the above plot, we can see that there is a negative relationship between income06 and egalit_scale. The higher an individual's income, the less egalitarian they are likely to be, which is similar to what we found out in the polynomial regression model.

4. Estimate the following models using all the available predictors (be sure to perform appropriate data pre-processing (e.g., feature standardization) and hyperparameter tuning (e.g. lambda for PCR/PLS, lambda and alpha for elastic net). Also use 10-fold cross-validation for each model to estimate the model's performance using MSE):

```
gss_train <- gss_train %>%
    mutate_if(is.integer, scale) %>%
    mutate_if(is.integer, c)
gss_test <- gss_test %>%
    mutate_if(is.integer, scale) %>%
    mutate_if(is.integer, c)
```

a. Linear regression

```
library(glmnet)
```

```r
train.control <- trainControl(method = "cv", number = 10)
lmodel <- train(egalit_scale ~., data = gss_train, method = "lm",
               trControl = train.control)
lmodel
```

```
## Linear Regression
##
## 1481 samples
##   45 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1334, 1334, 1333, 1334, 1332, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.8311914  0.3187364  0.6581222
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

MSE is 0.690879

b. Elastic net regression

```r
elr = train(
  egalit_scale ~ ., data = gss_train,
  method = "glmnet",
  trControl = train.control,
  tuneLength = 10
)
elr
```

```
## glmnet
##
## 1481 samples
##   45 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1332, 1332, 1332, 1334, 1334, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda        RMSE       Rsquared   MAE
##   0.1    0.0002201401  0.8365024  0.3140697  0.6604380
##   0.1    0.0005085521  0.8364392  0.3141377  0.6603988
##   0.1    0.0011748214  0.8353316  0.3153555  0.6596767
##   0.1    0.0027139898  0.8336774  0.3171557  0.6584219
##   0.1    0.0062696687  0.8316225  0.3193209  0.6570465
##   0.1    0.0144837482  0.8284939  0.3225750  0.6548755
##   0.1    0.0334593377  0.8225068  0.3295103  0.6508953
##   0.1    0.0772954117  0.8160702  0.3379655  0.6480541
##   0.1   0.1785624307  0.8111845  0.3492419  0.6488689
##   0.1   0.4125023847  0.8229573  0.3487924  0.6668048
##   0.2   0.0002201401  0.8370554  0.3134894  0.6607432
##   0.2   0.0005085521  0.8360511  0.3145854  0.6601440
##   0.2   0.0011748214  0.8344648  0.3163433  0.6590347
##   0.2   0.0027139898  0.8324672  0.3185328  0.6575864
##   0.2   0.0062696687  0.8298971  0.3212793  0.6557937
##   0.2   0.0144837482  0.8246404  0.3273564  0.6520976
##   0.2   0.0334593377  0.8178312  0.3358551  0.6482994
##   0.2   0.0772954117  0.8105222  0.3478099  0.6456388
##   0.2   0.1785624307  0.8139078  0.3514000  0.6551488
##   0.2   0.4125023847  0.8480193  0.3215197  0.6942420
##   0.3   0.0002201401  0.8368357  0.3137582  0.6605736
##   0.3   0.0005085521  0.8357000  0.3149880  0.6599231
##   0.3   0.0011748214  0.8337531  0.3171558  0.6584854
##   0.3   0.0027139898  0.8315345  0.3195866  0.6569496
##   0.3   0.0062696687  0.8280705  0.3234515  0.6543909
##   0.3   0.0144837482  0.8216980  0.3311345  0.6501077
##   0.3   0.0334593377  0.8147713  0.3403667  0.6469221
##   0.3   0.0772954117  0.8089562  0.3522843  0.6463119
##   0.3   0.1785624307  0.8223913  0.3432373  0.6658919
##   0.3   0.4125023847  0.8719852  0.2917205  0.7188842
##   0.4   0.0002201401  0.8366788  0.3139102  0.6605352
##   0.4   0.0005085521  0.8352242  0.3155264  0.6595792
##   0.4   0.0011748214  0.8331702  0.3178083  0.6580294
##   0.4   0.0027139898  0.8308010  0.3204064  0.6564107
##   0.4   0.0062696687  0.8262456  0.3256919  0.6530565
##   0.4   0.0144837482  0.8194862  0.3340117  0.6488347
##   0.4   0.0334593377  0.8119668  0.3449166  0.6453445
##   0.4   0.0772954117  0.8101266  0.3524262  0.6491239
##   0.4   0.1785624307  0.8348690  0.3264407  0.6796028
##   0.4   0.4125023847  0.8923065  0.2641063  0.7386402
##   0.5   0.0002201401  0.8364301  0.3141883  0.6603726
##   0.5   0.0005085521  0.8347722  0.3160408  0.6592685
##   0.5   0.0011748214  0.8325633  0.3185001  0.6576304
##   0.5   0.0027139898  0.8300065  0.3213157  0.6558275
##   0.5   0.0062696687  0.8247269  0.3275586  0.6520282
##   0.5   0.0144837482  0.8178056  0.3362407  0.6480486
##   0.5   0.0334593377  0.8098879  0.3486664  0.6444581
##   0.5   0.0772954117  0.8122676  0.3511004  0.6526047
##   0.5   0.1785624307  0.8471493  0.3093296  0.6928375
##   0.5   0.4125023847  0.9080094  0.2445916  0.7542887
##   0.6   0.0002201401  0.8362234  0.3144231  0.6602430
##   0.6   0.0005085521  0.8344032  0.3164504  0.6589953
##   0.6   0.0011748214  0.8320783  0.3190503  0.6573308
##   0.6   0.0027139898  0.8292202  0.3222342  0.6552451
##   0.6   0.0062696687  0.8233625  0.3292783  0.6511052
##   0.6   0.0144837482  0.8163779  0.3382229  0.6474394
##   0.6   0.0334593377  0.8088245  0.3509641  0.6444379
##   0.6   0.0772954117  0.8158192  0.3472845  0.6574573
##   0.6    0.1785624307  0.8573568  0.2954940  0.7037562
##   0.6    0.4125023847  0.9188464  0.2416608  0.7654113
##   0.7    0.0002201401  0.8360678  0.3145983  0.6601475
##   0.7    0.0005085521  0.8340422  0.3168661  0.6587134
##   0.7    0.0011748214  0.8317129  0.3194590  0.6570665
##   0.7    0.0027139898  0.8283890  0.3232303  0.6545898
##   0.7    0.0062696687  0.8221309  0.3308409  0.6502999
##   0.7    0.0144837482  0.8150606  0.3401370  0.6468107
##   0.7    0.0334593377  0.8084297  0.3522845  0.6450185
##   0.7    0.0772954117  0.8206431  0.3411196  0.6631886
##   0.7    0.1785624307  0.8663703  0.2833476  0.7127077
##   0.7    0.4125023847  0.9310998  0.2356313  0.7771055
##   0.8    0.0002201401  0.8358555  0.3148370  0.6599982
##   0.8    0.0005085521  0.8337246  0.3172219  0.6584639
##   0.8    0.0011748214  0.8313738  0.3198390  0.6568084
##   0.8    0.0027139898  0.8275628  0.3242337  0.6539421
##   0.8    0.0062696687  0.8210328  0.3322407  0.6495911
##   0.8    0.0144837482  0.8138662  0.3419453  0.6461870
##   0.8    0.0334593377  0.8086136  0.3526818  0.6460694
##   0.8    0.0772954117  0.8261585  0.3333851  0.6693788
##   0.8    0.1785624307  0.8751481  0.2707595  0.7210004
##   0.8    0.4125023847  0.9426506  0.2336733  0.7873713
##   0.9    0.0002201401  0.8356514  0.3150631  0.6598637
##   0.9    0.0005085521  0.8334734  0.3175032  0.6582701
##   0.9    0.0011748214  0.8310528  0.3201980  0.6565807
##   0.9    0.0027139898  0.8267688  0.3252018  0.6533739
##   0.9    0.0062696687  0.8200743  0.3334672  0.6490388
##   0.9    0.0144837482  0.8125701  0.3439879  0.6454075
##   0.9    0.0334593377  0.8091234  0.3525833  0.6472806
##   0.9    0.0772954117  0.8319568  0.3248590  0.6757371
##   0.9    0.1785624307  0.8837063  0.2578570  0.7292716
##   0.9    0.4125023847  0.9556264  0.2336733  0.7982316
##   1.0    0.0002201401  0.8353801  0.3153765  0.6596813
##   1.0    0.0005085521  0.8331579  0.3178558  0.6580239
##   1.0    0.0011748214  0.8307369  0.3205500  0.6563472
##   1.0    0.0027139898  0.8260325  0.3261027  0.6528689
##   1.0    0.0062696687  0.8192339  0.3345506  0.6486226
##   1.0    0.0144837482  0.8113546  0.3459931  0.6446587
##   1.0    0.0334593377  0.8099367  0.3520357  0.6485929
##   1.0    0.0772954117  0.8375177  0.3167066  0.6818205
##   1.0    0.1785624307  0.8912788  0.2465169  0.7369840
##   1.0    0.4125023847  0.9708450  0.2336733  0.8123445
## 
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.7 and lambda = 0.033459
34.
```

The MSE with alpha = 0.7 and lambda = 0.03345934 is 0.653559, which is smaller than Linear Regression.

c. Principal component regression

```
pcr <- train(egalit_scale ~ ., data = gss_train, method = "pcr",
         trControl = train.control,
         metric = "RMSE", tuneLength = 50)
pcr
```

```
## Principal Component Analysis
##
## 1481 samples
##   45 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1333, 1333, 1332, 1333, 1334, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared     MAE
##    1     0.9996836  0.007171004  0.8379895
##    2     0.9367006  0.125604412  0.7743511
##    3     0.9211007  0.154603560  0.7619968
##    4     0.9172501  0.161305442  0.7592386
##    5     0.9160026  0.163426974  0.7572842
##    6     0.9166104  0.162234650  0.7575246
##    7     0.9134960  0.167888965  0.7538114
##    8     0.9120177  0.170717602  0.7509096
##    9     0.9126518  0.169666651  0.7517092
##   10     0.9127463  0.169880588  0.7509546
##   11     0.8981347  0.197925175  0.7387062
##   12     0.8989773  0.196823761  0.7397834
##   13     0.8726978  0.238594136  0.7058676
##   14     0.8716767  0.240528587  0.7043353
##   15     0.8658886  0.251184870  0.6969301
##   16     0.8650553  0.252962019  0.6963689
##   17     0.8652499  0.252707915  0.6960416
##   18     0.8621964  0.257247505  0.6942521
##   19     0.8597529  0.261482161  0.6935778
##   20     0.8591647  0.262432010  0.6918476
##   21     0.8477705  0.281484341  0.6797883
##   22     0.8483754  0.280433506  0.6805776
##   23     0.8445453  0.286892667  0.6767306
##   24     0.8454402  0.285692409  0.6775026
##   25     0.8457528  0.285471845  0.6767809
##   26     0.8438522  0.289073244  0.6742334
##   27     0.8413847  0.293930418  0.6719234
##   28     0.8359354  0.302673455  0.6683623
##   29     0.8358956  0.303329804  0.6672151
##   30     0.8369368  0.301957629  0.6669607
##   31     0.8246537  0.321995194  0.6560556
##   32     0.8254945  0.320856880  0.6564826
##   33     0.8247774  0.322011026  0.6558780
##   34     0.8250309  0.321632024  0.6564038
##   35     0.8237395  0.323671980  0.6555391
##   36     0.8226679  0.325530413  0.6542066
##   37     0.8216073  0.327180083  0.6531404
##   38     0.8234569  0.324406718  0.6535599
##   39     0.8230071  0.325259578  0.6536168
##   40     0.8217102  0.327348454  0.6533278
##   41     0.8208202  0.328681904  0.6528992
##   42     0.8199218  0.330330478  0.6523966
##   43     0.8172985  0.334506987  0.6503641
##   44     0.8175953  0.334386990  0.6501202
##   45     0.8179146  0.334227812  0.6510198
##   46     0.8187544  0.332924164  0.6514097
##   47     0.8195342  0.331887785  0.6517854
##   48     0.8198212  0.331426902  0.6517961
##   49     0.8198976  0.331329456  0.6519498
##   50     0.8192549  0.332423125  0.6514422
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 43.
```

The best number of principal components is 43. The MSE is 0.66798 which is worse than Elastic Net but better than Linear Regression.

d. Partial least squares regression

```
pls = train(
  egalit_scale ~ ., data = gss_train,
  method = "pls",
  trControl = train.control,
  tuneLength = 50
)
pls
```

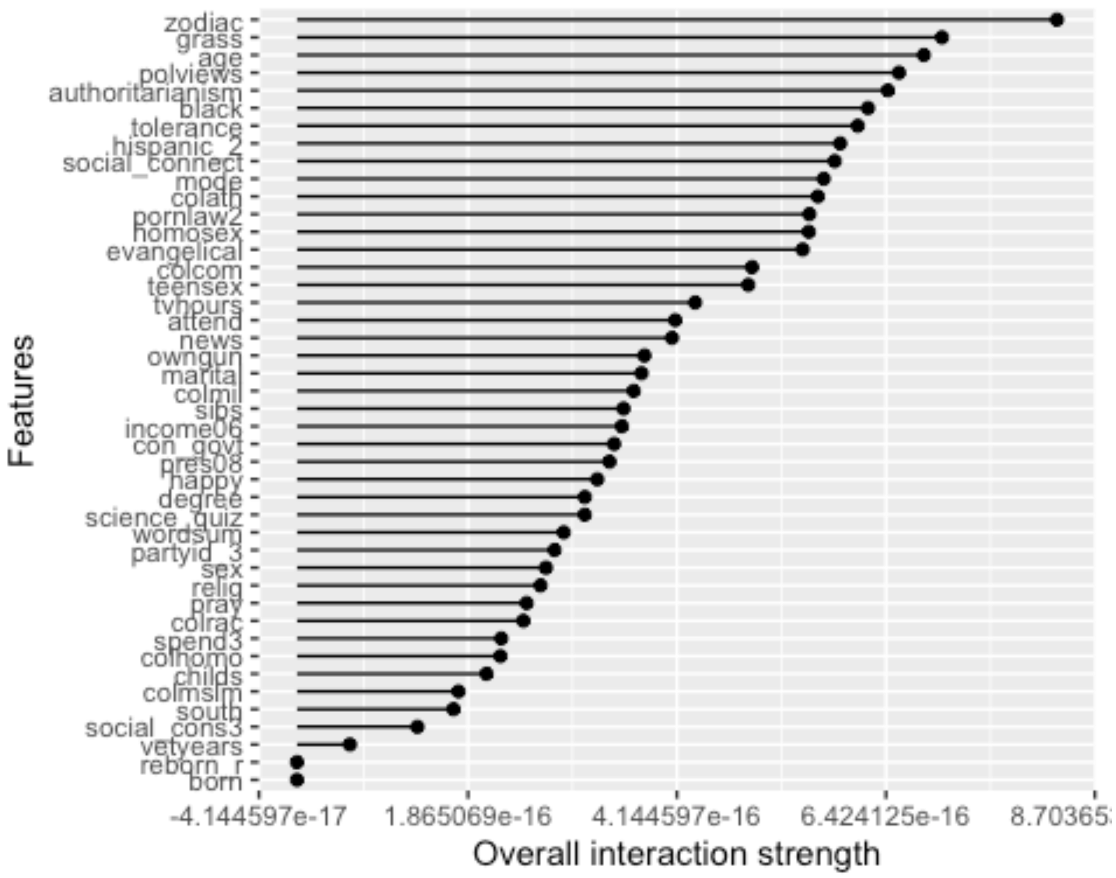
```
## Partial Least Squares
##
## 1481 samples
##   45 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1334, 1334, 1334, 1332, 1332, 1333, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE       Rsquared   MAE
##    1     0.8777560  0.2329988  0.7145092
##    2     0.8431473  0.2936222  0.6722379
##    3     0.8315987  0.3122365  0.6624590
##    4     0.8246272  0.3233038  0.6524952
##    5     0.8220281  0.3285686  0.6523111
##    6     0.8246923  0.3257007  0.6537868
##    7     0.8281283  0.3214520  0.6557273
##    8     0.8308319  0.3185297  0.6572824
##    9     0.8317398  0.3177886  0.6569875
##   10     0.8323539  0.3175554  0.6572395
##   11     0.8321146  0.3180799  0.6571540
##   12     0.8301190  0.3213693  0.6545401
##   13     0.8292263  0.3229184  0.6537179
##   14     0.8283409  0.3241549  0.6525641
##   15     0.8285332  0.3239940  0.6528852
##   16     0.8288595  0.3236448  0.6529116
##   17     0.8293823  0.3231619  0.6531783
##   18     0.8297630  0.3226391  0.6537405
##   19     0.8304124  0.3217059  0.6541253
##   20     0.8304573  0.3217161  0.6541223
##   21     0.8309551  0.3211280  0.6545872
##   22     0.8307994  0.3213349  0.6545620
##   23     0.8313209  0.3207078  0.6553449
##   24     0.8312885  0.3207917  0.6549490
##   25     0.8313487  0.3207471  0.6546072
##   26     0.8316463  0.3203758  0.6550663
##   27     0.8317374  0.3202606  0.6549979
##   28     0.8317954  0.3201621  0.6550590
##   29     0.8317867  0.3201634  0.6550650
##   30     0.8320297  0.3198300  0.6552192
##   31     0.8320539  0.3197894  0.6552900
##   32     0.8319783  0.3198792  0.6553162
##   33     0.8317907  0.3201312  0.6551271
##   34     0.8317480  0.3201731  0.6550332
##   35     0.8317695  0.3201573  0.6550276
##   36     0.8317809  0.3201406  0.6549899
##   37     0.8319677  0.3199133  0.6551349
##   38     0.8320562  0.3198192  0.6552136
##   39     0.8323638  0.3193987  0.6554692
##   40     0.8326703  0.3189718  0.6557165
##   41     0.8328532  0.3187369  0.6557931
##   42     0.8331943  0.3182776  0.6559403
##   43     0.8331117  0.3183879  0.6559335
##   44     0.8329476  0.3186932  0.6558402
##   45     0.8324048  0.3194773  0.6554411
##   46     0.8324381  0.3194680  0.6556794
##   47     0.8325683  0.3193471  0.6555748
##   48     0.8329309  0.3188085  0.6557993
##   49     0.8331389  0.3185127  0.6559339
##   50     0.8332875  0.3182699  0.6560132
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 5.
```

The best number of principal components is 5 which is much smaller than Principal Component Analysis. The MSE is 0.675730 which is larger than Principal Component Analysis but still better than Linear Regression.

Elastic Net is the best model for this dataset when comparing the MSE.

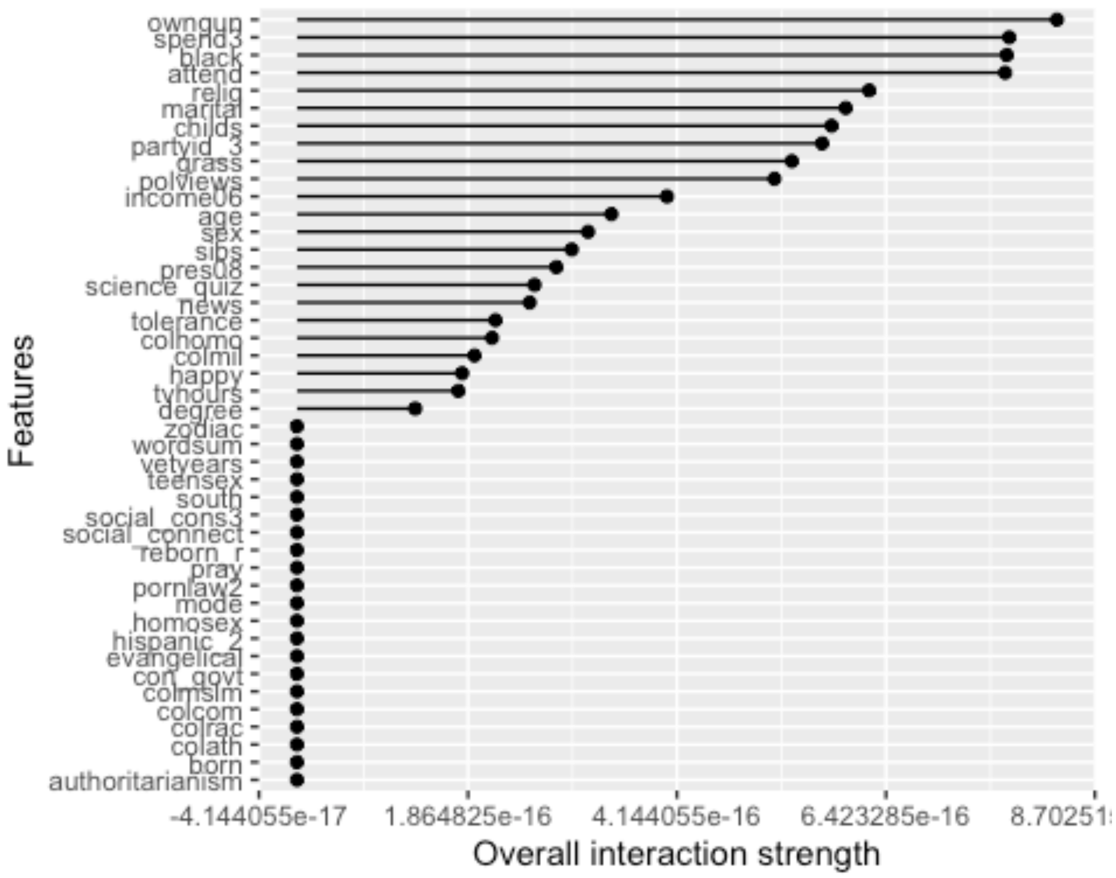
5. For each final tuned version of each model fit, evaluate feature importance by generating feature interaction plots. Upon visual presentation, be sure to discuss the substantive results for these models and in comparison to each other (e.g., talk about feature importance, conditional effects, how these are ranked differently across different models, etc.)

```
gss_train <- read.csv('~/Desktop/problem-set-4-master/data/gss_train.csv')
gss_test <- read.csv('~/Desktop/problem-set-4-master/data/gss_test.csv')


lmodel <- train(egalit_scale ~., data = gss_train, method = "lm",
               trControl = train.control)
elr = train(egalit_scale ~ ., data = gss_train,
  method = "glmnet",
  trControl = train.control,
  tuneLength = 10)
pcr <- train(egalit_scale ~ ., data = gss_train, method = "pcr",
         trControl = train.control,
         metric = "RMSE", tuneLength = 50)
pls = train(egalit_scale ~ ., data = gss_train,
  method = "pls",
  trControl = train.control,
  tuneLength = 50)

library(iml)
features <- gss_test %>%
  dplyr::select(- egalit_scale)
response <- as.integer(gss_test$egalit_scale)

pred_lm <- Predictor$new(
  model = lmodel,
  data = features,
  y = response
)
interact_lm <- Interaction$new(pred_lm)
plot(interact_lm)
```



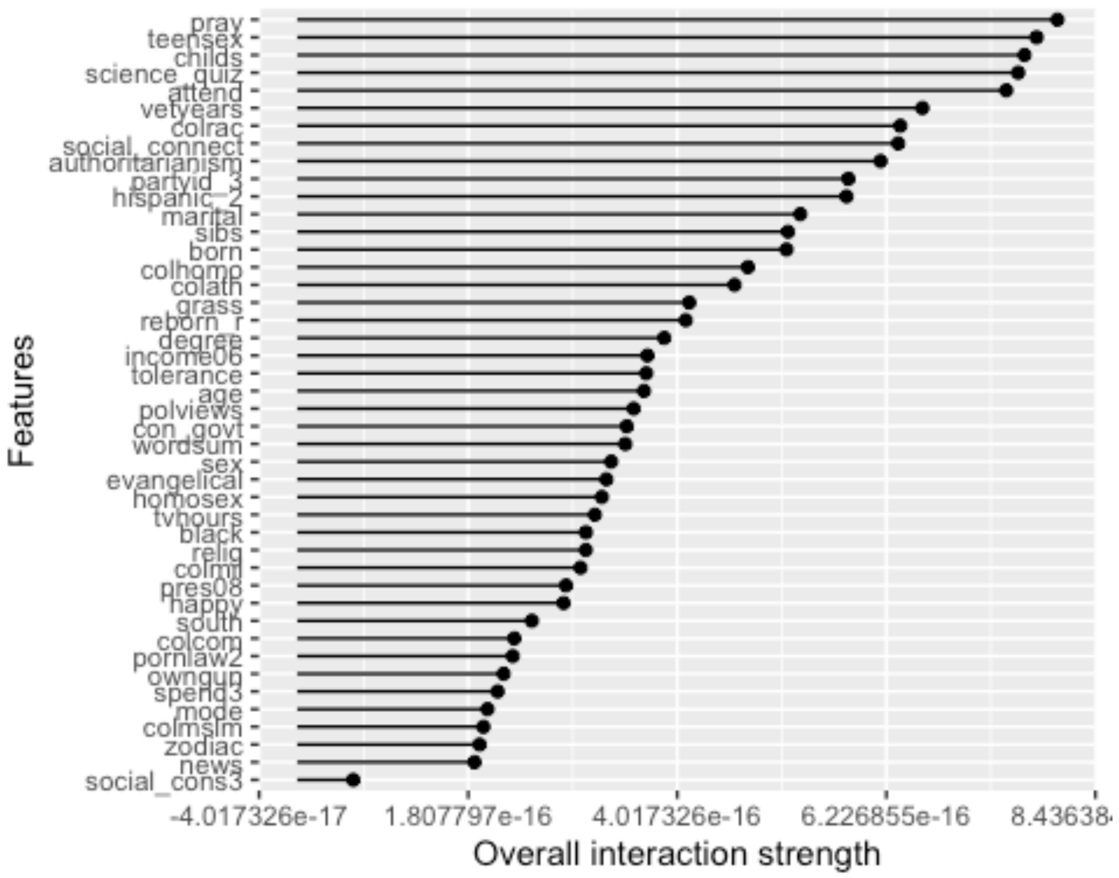
In this linear model, zodiac, grass, age are the three features with the highest interaction strength.

```r
pred_elr <- Predictor$new(
  model = elr,
  data = features,
  y = response
)
interact_elr <- Interaction$new(pred_elr)
plot(interact_elr)
```



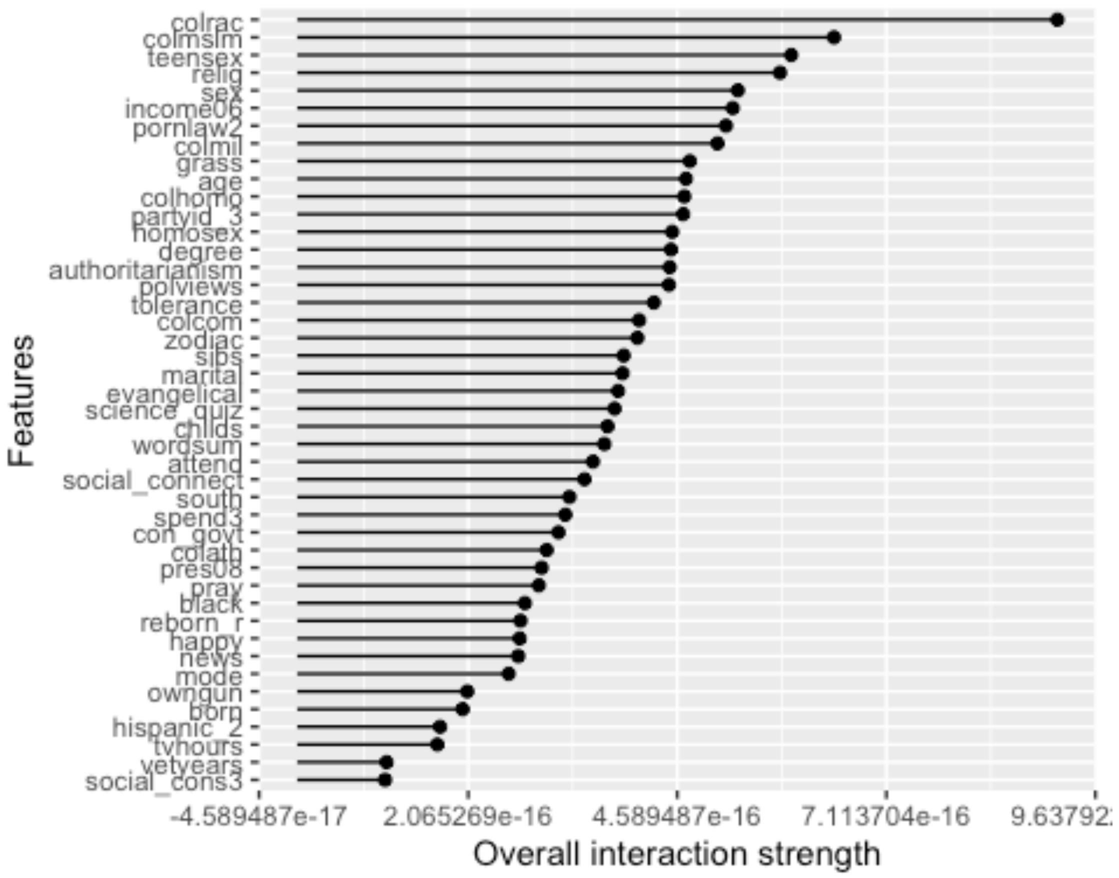
In this elastic net model, owngun, spend3, black are the three features with the highest interaction strength. In addition, attend is very close to black.

```r
pred_pcr <- Predictor$new(
  model = pcr,
  data = features,
  y = response
)
interact_pcr <- Interaction$new(pred_pcr)
plot(interact_pcr)
```



In this PCR model, pray, teensex, childs are the three features with the highest interaction strength.

```
pred_pls <- Predictor$new(
  model = pls,
  data = features,
  y = response
)
interact_pls <- Interaction$new(pred_pls)
plot(interact_pls)
```



In this PLS model, colrac, colmslm, teensex are the three features with the highest interaction strength. Relig is very close to teensex.

From these plots, we can see that all models have different rank of important features by interaction strength because they prioritize and penalize different things.