MACS30100 Homework 4

Takuyo Ozaki

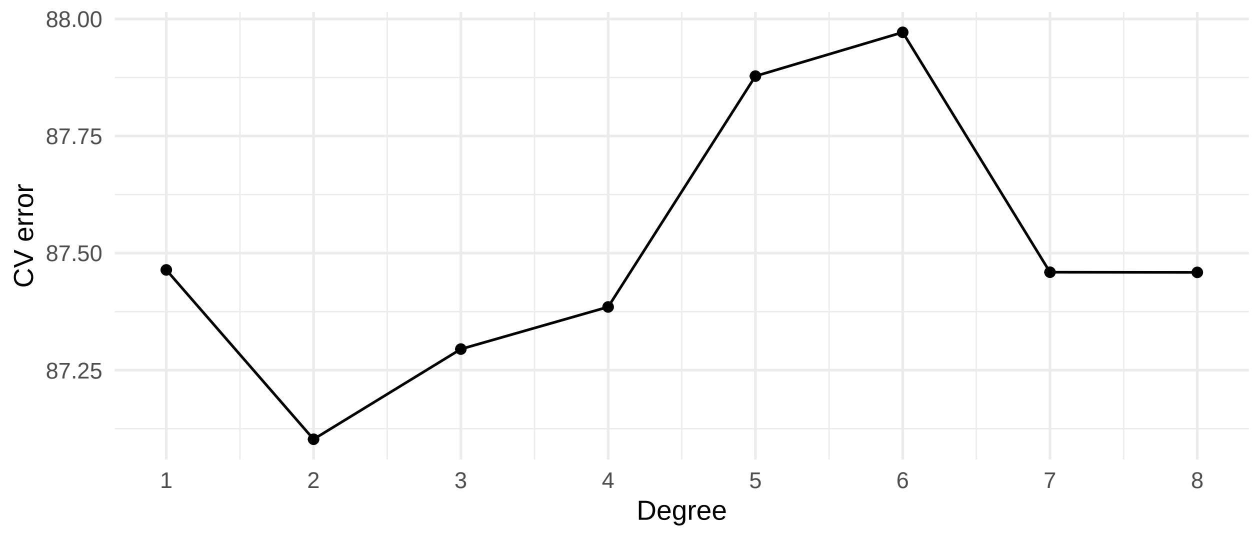
2/16/2020

```r
knitr::opts_chunk$set(message = FALSE, warning = FALSE)
knitr::opts_chunk$set(fig.width = 7, fig.height = 3, fig.align = 'center')

# load necessary package
library(tidyverse)
library(broom)
library(rsample)
library(tibble)
library(rcfss)
library(knitr)
library(boot)
library(splines)
library(magrittr)
library(glmnet)
library(margins)
library(plotly)
library(pls)
library(caret)
library(iml)
library(elasticnet)
library(gridExtra)

# Set the theme as minimal
theme_set(theme_minimal())

# Import the data
gss_test <- read_csv("data/gss_test.csv")
gss_train <- read_csv("data/gss_train.csv")
```

Egalitarianism and income

Question 1

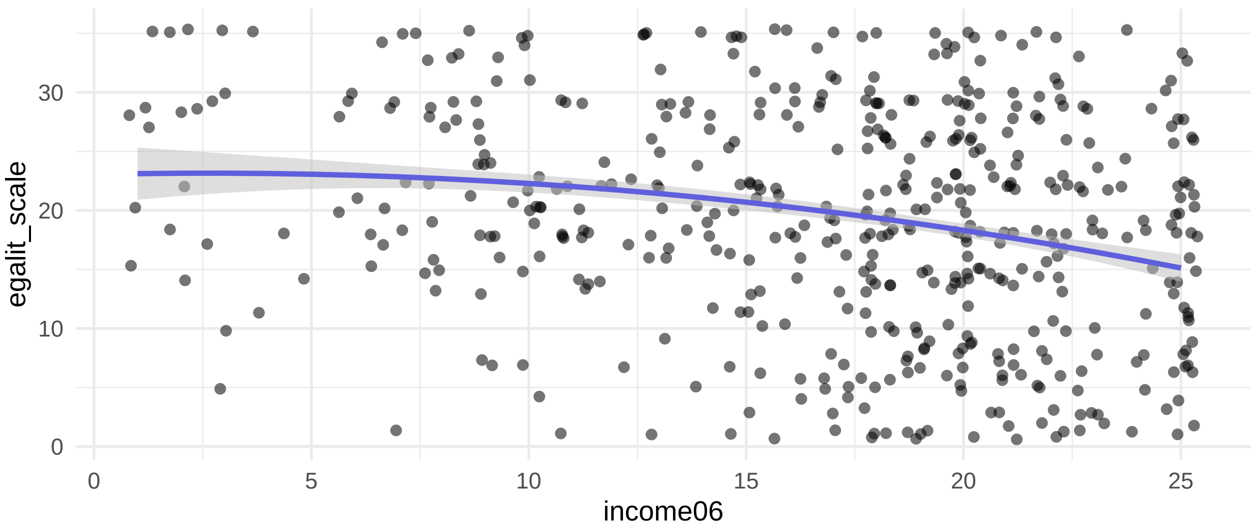
```r
set.seed(0) # Ensure the reproducibility

# Note: I use the cv.glm ("boot" package) to perform the 10-fold cross-validation
# which is different from the lab's way, but I found this is a simpler way (code)
# Fit the train data to polynomial functions and perform the CV by each degrees
degrees <- rep(NA, 8)
error_poly <- rep(NA, 8)
for (i in 1:8) {
  poly.fit <- glm(egalit_scale ~ poly(income06, i), data = gss_train)
  degrees[i] <- i
  error_poly[i] <- cv.glm(gss_train, poly.fit, K = 10)$delta[2]
  poly <- data.frame(degrees, error_poly)
}

# Plot the CV error by each degree
poly %>%
  ggplot(aes(x = degrees, y = error_poly)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = seq(0, 10, 1)) +
  labs(x = "Degree", y = "CV error")
```



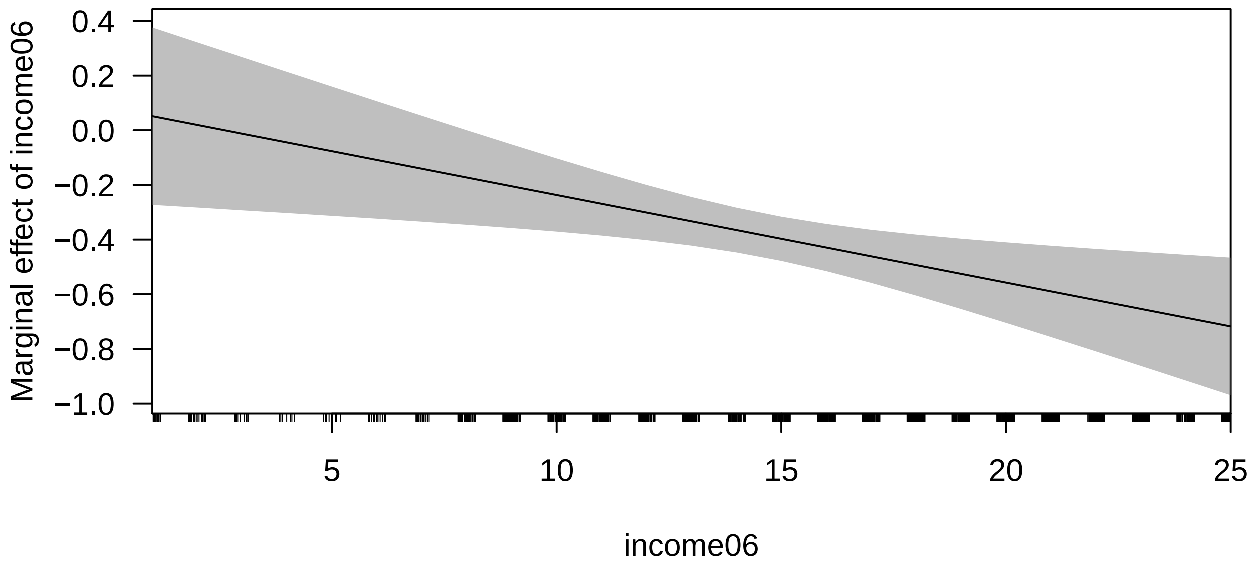
```r
# Show the degree when the CV error is minimum
poly %>%
  filter(error_poly == min(error_poly))
```

```
##   degrees error_poly
## 1       2    87.1025
```

```r
# Fit the test data and predict the egalit scale
poly_fit <- glm(egalit_scale ~ stats::poly(income06, 2), data = gss_train)
poly_pred <- augment(poly_fit, newdata = gss_test)

# Plot the income and egalit scale by jitter and prediction line by ggplot
poly_pred %>%
  ggplot(aes(x = income06)) +
  geom_jitter(aes(y = egalit_scale, alpha = 0.3)) +
  geom_line(aes(y = .fitted), data = poly_pred, color = "blue", size = 1) +
  geom_ribbon(aes(ymin = .fitted - 2*.se.fit, ymax = .fitted + 2*.se.fit),
              fill = "gray", alpha = 0.5) +
  theme(legend.position = 'none')
```



```
# Plot the  marginal effect of income by cplot
cplot(poly_fit, "income06", what = "effect")
```



```
# Calculate the average marginal effect
summary(margins(poly_fit))
```

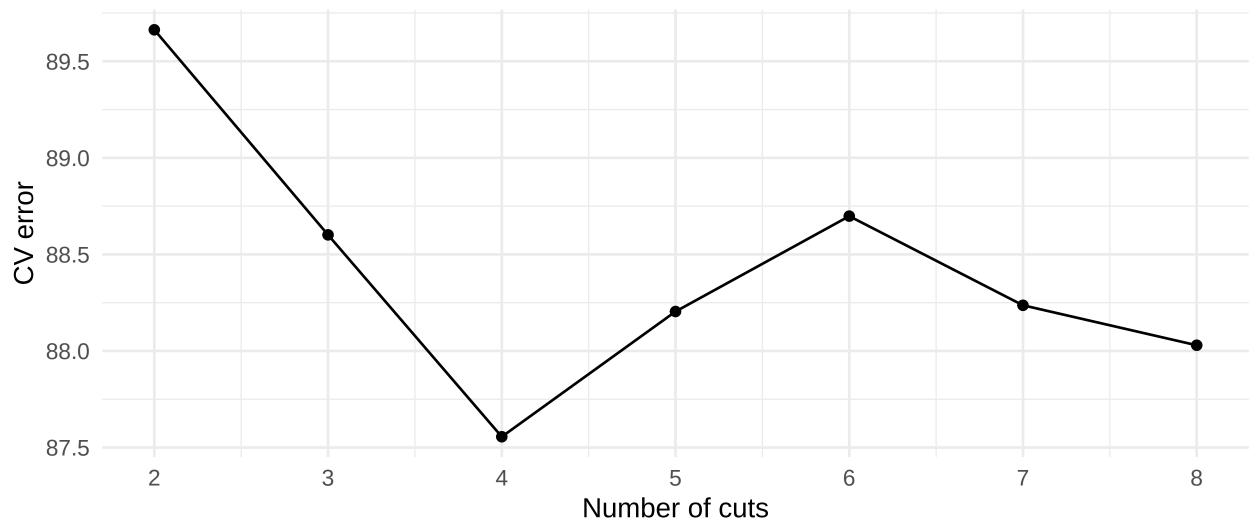
```
##     factor     AME     SE       z      p   lower   upper
##   income06 -0.4507 0.0475 -9.4933 0.0000 -0.5438 -0.3577
```

- To predict egalit_scale as a function of income06, the quadratic function is the best fit among the polynomial regression.

- The plot of the polynomial fit to the data shows the downward sloping. Thus, we can interpret that people with more income can be predicted to have lower feelings towards egalitarianism.
- The graph of the marginal effect of income also shows downward sloping. On average, the marginal effect is -0.4507 (SE: 0.0475). Thus, we can interpret that the change of the feeling towards egalitarianism could decrease when the income increases, but on average, we can predict one unit increase in income06 could decrease the feeling towards egalitarianism by 0.4507.

Question 2

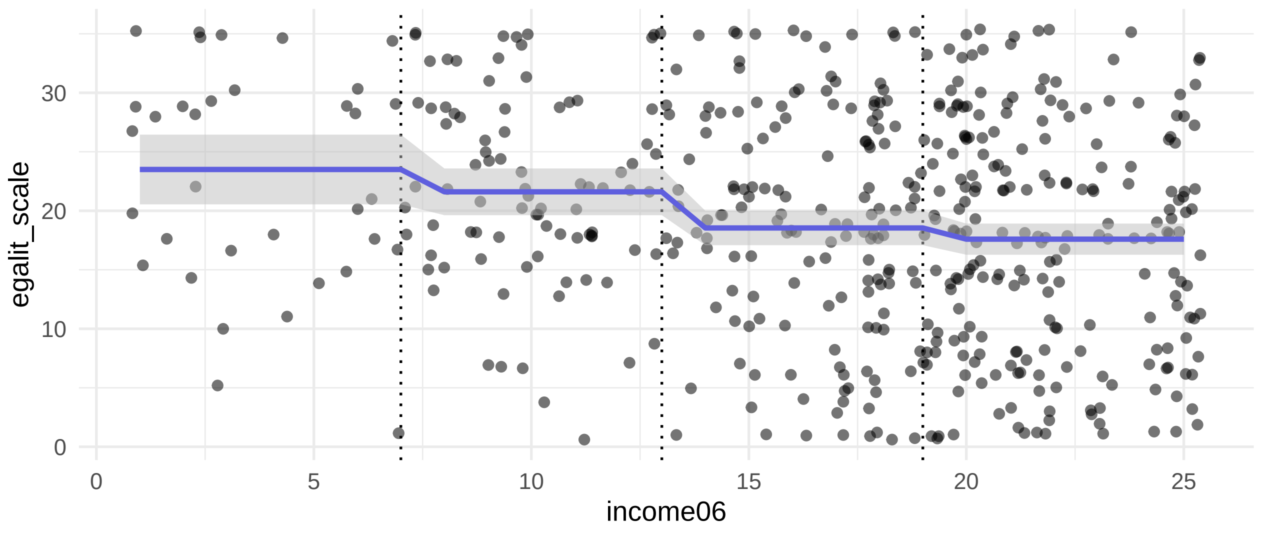
```r
set.seed(0) #Ensure the reproducibility

# Fit the train data to step functions and perform the CV by each number of cuts
cuts <- rep(NA, 7)
error_steps <- rep(NA, 7)
for (i in 2:8) {
  gss_train$income_cut <- cut(gss_train$income06, i)
  step.fit <- glm(egalit_scale ~ income_cut, data = gss_train)
  cuts[i] <- i
  error_steps[i] <- cv.glm(gss_train, step.fit, K = 10)$delta[2]
  steps <- data.frame(cuts, error_steps)
}

# Plot the CV error by each number of cuts
steps <- steps[-1, ]
steps %>%
  ggplot(aes(x = cuts, y = error_steps)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = seq(2, 8, 1)) +
  labs(x = "Number of cuts", y = "CV error")
```



```r
# Show the steps when the CV error is minimum
steps %>%
  filter(error_steps == min(error_steps))
```

```
##   cuts error_steps
## 1    4    87.55596
```

```r
# Fit the test data and predict the egalit scale
gss_test$income_cut <- cut(gss_test$income06, 4)
steps_fit <- glm(egalit_scale ~ income_cut, data = gss_test)
steps_pred <- augment(steps_fit, newdata = gss_test)

# Plot the income and egalit scale by jitter and prediction line
steps_pred %>%
  ggplot(aes(x = income06)) +
  geom_vline(xintercept = c(7, 13, 19), linetype = "dotted") +
  geom_jitter(aes(y = egalit_scale, alpha = 0.3)) +
  geom_line(aes(y = .fitted), data = steps_pred, color = "blue", size = 1) +
  geom_ribbon(aes(ymin = .fitted - 2*.se.fit, ymax = .fitted + 2*.se.fit),
              fill = "gray", alpha = 0.5) +
  theme(legend.position = 'none')
```

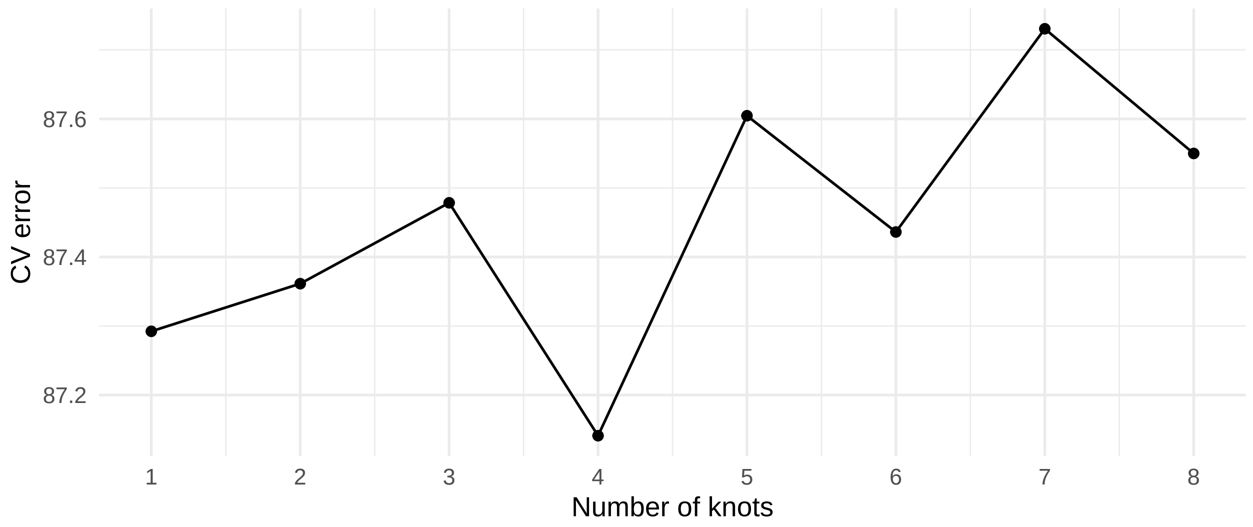


```r
# Eliminate the income_cut
gss_train[ ,46]  <- NULL
gss_test[ ,46] <- NULL
```

- Fit a step function to predict egalit_scale as a function of income06, and chose the optimal number of cuts: 4 cuts and plot the data as above.
- The interpretation is that the categories of the people with more income can be predicted to have lower feelings towards egalitarianism. The standard errors are larger in low income categories partly because of the fewer sample size.

Question 3

```r
set.seed(1) #Ensure the reproducibility

# Fit the train data to spline functions and perform the CV by each number of knots
knots <- 1:8
error_spline <- rep(NA, 8)
for (i in 1:8) {
    spline.fit <- glm(egalit_scale ~ ns(income06, df = i + 3), data = gss_train)
    error_spline[i] <- cv.glm(gss_train, spline.fit, K = 10)$delta[2]
}
spline <- data.frame(knots, error_spline)

# Plot the CV error by each number of knots
spline %>%
  ggplot(aes(x = knots, y = error_spline)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = seq(1, 8, 1)) +
  labs(x = "Number of knots", y = "CV error")
```

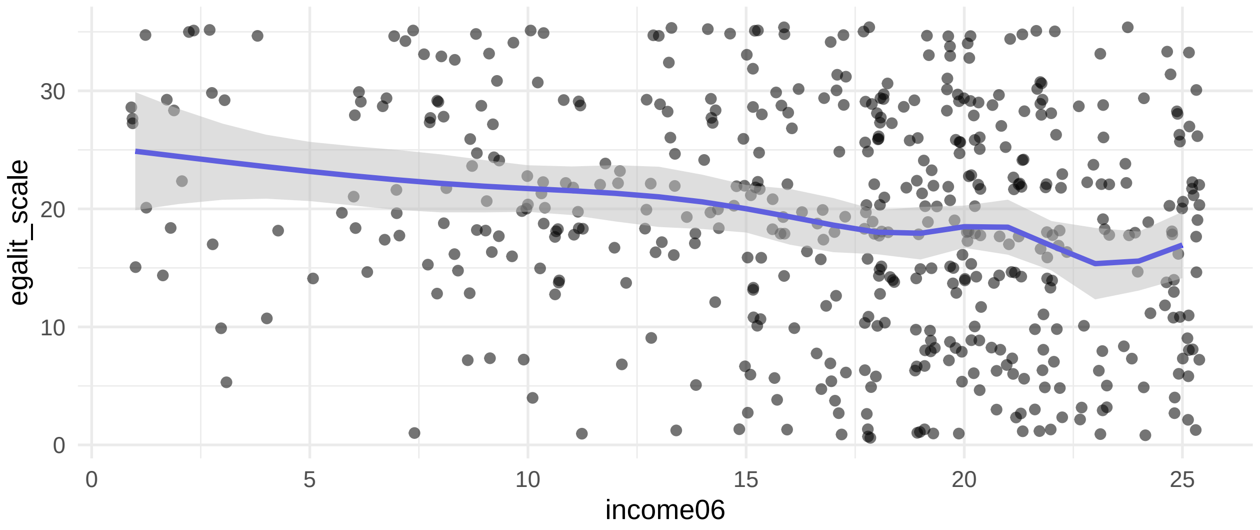


```r
# Show the knots when the CV error is minimum
spline %>%
  filter(error_spline == min(error_spline))
```

```
##   knots error_spline
## 1     4     87.14114
```

```r
# Fit the test data and predict the egalit scale
spline_fit <- glm(egalit_scale ~ splines::ns(income06, df = 4 + 3), data = gss_test)
spline_pred <- augment(spline_fit, newdata = gss_test)

# Plot the income and egalit scale by jitter and prediction line
```

```r
spline_pred %>%
  ggplot(aes(x = income06)) +
  geom_jitter(aes(y = egalit_scale, alpha = 0.3)) +
  geom_line(aes(y = .fitted), data = spline_pred, color = "blue", size = 1) +
  geom_ribbon(aes(ymin = .fitted - 2*.se.fit, ymax = .fitted + 2*.se.fit),
              fill = "gray", alpha = 0.5) +
  theme(legend.position = 'none')
```



- I fit a natural regression spline to predict egalit_scale as a function of income06. I selected the optimal number of degrees of freedom: 7, and plot the results of the optimal model.
- The interpretation is that the people with more income can be predicted to have lower feelings towards egalitarianism. Looking at the standard errors (gray zone), this is a better fit to the data than the step function.

Egalitarianism and everything

Question 4

data pre-processing: feature standardization

I converted category variables of the data to numeric variables, and then standardized features (except egalit_scale). By doing this, all features' mean became 0 and ones' standard deviation became 1.

```r
# Convert category variable to numeric variable
train_numeric <- data.matrix(data.frame(unclass(gss_train)))
test_numeric <- data.matrix(data.frame(unclass(gss_test)))

# Standardize the features except egalit_scale (mean:0, sd:1)
train_scale <- scale(data.frame(train_numeric) %>% select(!egalit_scale))
train_scale <- as.data.frame(train_scale)
train_scale$egalit_scale <- data.frame(train_numeric)$egalit_scale
```

```r
test_scale <- scale(data.frame(test_numeric) %>% select(!egalit_scale))
test_scale <- as.data.frame(test_scale)
test_scale$egalit_scale <- data.frame(test_numeric)$egalit_scale

# Check the feature standardization only in test data just in case
test_scale %>% summarise_each(mean) # mean = 0 (except egalit_scale)
```

```
##             age        attend authoritarianism        black          born
## 1 2.009071e-16 -1.642366e-16     1.308494e-16 4.982492e-17 -1.790799e-16
##          childs        colath        colrac        colcom       colmil
## 1 6.401359e-18 -3.838538e-17 1.427727e-16 -2.198206e-16 8.909393e-17
##         colhomo       colmslm      con_govt        degree  evangelical
## 1 1.781258e-16 -7.823849e-18 -8.394435e-17 9.985348e-17 -6.562733e-17
##           grass         happy  hispanic_2       homosex     income06
## 1 -1.283555e-16 -9.681453e-18 8.04919e-17 -2.837702e-16 -3.883775e-18
##         marital          mode          news        owngun     partyid_3
## 1 7.119492e-17 6.112239e-17 -5.958318e-17 -1.201182e-16 -9.921093e-17
##        polviews      pornlaw2          pray        pres08      reborn_r
## 1 2.484182e-16 -1.169628e-16 -5.530443e-17 3.054107e-17 -1.549949e-16
##           relig  science_quiz           sex          sibs social_connect
## 1 7.230397e-18 -2.274123e-16 2.180443e-16 5.410103e-17  -7.284431e-17
##    social_cons3         south        spend3       teensex     tolerance
## 1 1.678754e-16 -1.570639e-16 1.640343e-16 -3.341476e-17 -1.770606e-16
##         tvhours vetyears      wordsum       zodiac egalit_scale
## 1 5.409179e-17        0 -2.34579e-16 -8.67784e-17     19.11359
```

```r
test_scale %>% summarise_each(sd) # sd = 1 (except egalit_scale)
```

```
##   age attend authoritarianism black born childs colath colrac colcom colmil
## 1   1      1                1     1    1      1      1      1      1      1
##   colhomo colmslm con_govt degree evangelical grass happy hispanic_2 homosex
## 1       1       1        1      1           1     1     1          1       1
##   income06 marital mode news owngun partyid_3 polviews pornlaw2 pray pres08
## 1        1       1    1    1      1         1        1        1    1      1
##   reborn_r relig science_quiz sex sibs social_connect social_cons3 south spend3
## 1        1     1            1   1    1              1            1     1      1
##   teensex tolerance tvhours vetyears wordsum zodiac egalit_scale
## 1       1         1       1        1       1      1     9.515674
```

(a) Linear regression

```r
# Fit the linear regression using 10 fold CV to minimize MSE
(glm <- train(egalit_scale~., data = train_scale,
              method = 'lm', metric = 'RMSE',
              trControl = trainControl(method = 'cv', number = 10)))
```

```
## Linear Regression
##
## 1481 samples
##   44 predictor
```

```
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1334, 1333, 1332, 1332, 1334, 1333, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   8.004338  0.3137795  6.394753
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# Calculate the MSE (= RMSE^2)
(8.002394)^2
```

```
## [1] 64.03831
```

In case of the linear regression, the MSE is 64.03831.

(b) Elastic net regression

```
set.seed(0) # Ensure the reproducibility

# Fit the elastic net regression using 10 fold CV to minimize MSE
(en <- train(egalit_scale~., data = train_scale,
             method = 'glmnet', metric = 'RMSE',
             trControl = trainControl(method = 'cv', number = 10),
             tuneLength = 10))
```

```
## glmnet
##
## 1481 samples
##   44 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1333, 1332, 1334, 1333, 1333, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda       RMSE      Rsquared   MAE
##   0.1    0.002118067  7.971155  0.3153620  6.354929
##   0.1    0.004893009  7.971155  0.3153620  6.354929
##   0.1    0.011303486  7.971155  0.3153620  6.354929
##   0.1    0.026112519  7.971155  0.3153620  6.354929
##   0.1    0.060323307  7.968860  0.3156240  6.353541
##   0.1    0.139354665  7.962791  0.3163555  6.349738
##   0.1    0.321927360  7.950305  0.3180480  6.343337
##   0.1    0.743693973  7.932957  0.3210822  6.346044
##   0.1    1.718029578  7.953600  0.3210112  6.405191
##   0.1    3.968871254  8.071438  0.3160035  6.573949
##   0.2    0.002118067  7.971198  0.3153667  6.354904
##   0.2    0.004893009  7.971198  0.3153667  6.354904
##   0.2    0.011303486  7.971198  0.3153667  6.354904
##   0.2    0.026112519  7.970364  0.3154650  6.354395
##   0.2    0.060323307  7.965828  0.3160170  6.351466
##   0.2    0.139354665  7.956189  0.3172800  6.345091
##   0.2    0.321927360  7.937803  0.3200559  6.339387
##   0.2    0.743693973  7.938018  0.3210721  6.371257
##   0.2    1.718029578  7.999136  0.3172993  6.473632
##   0.2    3.968871254  8.251110  0.2973389  6.773910
##   0.3    0.002118067  7.971220  0.3153703  6.354890
##   0.3    0.004893009  7.971220  0.3153703  6.354890
##   0.3    0.011303486  7.971220  0.3153703  6.354890
##   0.3    0.026112519  7.968981  0.3156404  6.353450
##   0.3    0.060323307  7.962938  0.3164007  6.349444
##   0.3    0.139354665  7.949279  0.3183070  6.341149
##   0.3    0.321927360  7.932004  0.3211066  6.343692
##   0.3    0.743693973  7.952675  0.3196806  6.397348
##   0.3    1.718029578  8.069018  0.3094130  6.564938
##   0.3    3.968871254  8.413989  0.2798254  6.937287
##   0.4    0.002118067  7.971275  0.3153641  6.354929
##   0.4    0.004893009  7.971275  0.3153641  6.354929
##   0.4    0.011303486  7.971189  0.3153752  6.354880
##   0.4    0.026112519  7.967591  0.3158199  6.352496
##   0.4    0.060323307  7.959959  0.3168086  6.347187
##   0.4    0.139354665  7.943281  0.3192203  6.338244
##   0.4    0.321927360  7.933329  0.3210813  6.354252
##   0.4    0.743693973  7.968756  0.3182112  6.424344
##   0.4    1.718029578  8.150972  0.2988712  6.661247
##   0.4    3.968871254  8.545275  0.2685114  7.069786
##   0.5    0.002118067  7.971311  0.3153624  6.354932
##   0.5    0.004893009  7.971311  0.3153624  6.354932
##   0.5    0.011303486  7.970624  0.3154467  6.354495
##   0.5    0.026112519  7.966267  0.3159932  6.351588
##   0.5    0.060323307  7.956945  0.3172355  6.345076
##   0.5    0.139354665  7.938223  0.3200186  6.337530
##   0.5    0.321927360  7.938103  0.3205741  6.365777
##   0.5    0.743693973  7.992129  0.3155617  6.457879
##   0.5    1.718029578  8.229123  0.2887109  6.743469
##   0.5    3.968871254  8.678108  0.2552305  7.204141
##   0.6    0.002118067  7.971314  0.3153636  6.354890
##   0.6    0.004893009  7.971314  0.3153636  6.354890
##   0.6    0.011303486  7.970007  0.3155249  6.354060
##   0.6    0.026112519  7.964968  0.3161630  6.350715
##   0.6    0.060323307  7.953927  0.3176716  6.343317
##   0.6    0.139354665  7.934492  0.3206236  6.338157
##   0.6    0.321927360  7.945162  0.3197017  6.378577
##   0.6    0.743693973  8.020505  0.3121024  6.496059
##   0.6    1.718029578  8.294651  0.2806336  6.813347
##   0.6    3.968871254  8.784309  0.2533486  7.309072
##   0.7    0.002118067  7.971300  0.3153665  6.354913
##   0.7    0.004893009  7.971300  0.3153665  6.354913
##   0.7    0.011303486  7.969391  0.3156038  6.353648
##   0.7    0.026112519  7.963747  0.3163235  6.349861
##   0.7    0.060323307  7.950781  0.3181357  6.341310
##   0.7    0.139354665  7.932875  0.3209039  6.341217
##   0.7    0.321927360  7.950959  0.3190766  6.389521
##   0.7    0.743693973  8.055092  0.3073967  6.538365
##   0.7    1.718029578  8.344313  0.2761476  6.865130
##   0.7    3.968871254  8.907687  0.2493906  7.426060
##   0.8    0.002118067  7.971288  0.3153699  6.354902
##   0.8    0.004893009  7.971288  0.3153699  6.354902
##   0.8    0.011303486  7.968779  0.3156824  6.353235
##   0.8    0.026112519  7.962483  0.3164923  6.348925
##   0.8    0.060323307  7.947884  0.3185664  6.339542
##   0.8    0.139354665  7.932719  0.3209636  6.345325
##   0.8    0.321927360  7.955653  0.3187149  6.398892
##   0.8    0.743693973  8.093841  0.3017719  6.584624
##   0.8    1.718029578  8.397186  0.2709978  6.917660
##   0.8    3.968871254  9.050237  0.2395210  7.554349
##   0.9    0.002118067  7.971308  0.3153682  6.354910
##   0.9    0.004893009  7.971308  0.3153682  6.354910
##   0.9    0.011303486  7.968172  0.3157607  6.352823
##   0.9    0.026112519  7.961180  0.3166710  6.347920
##   0.9    0.060323307  7.945302  0.3189515  6.338320
##   0.9    0.139354665  7.933090  0.3209596  6.349541
##   0.9    0.321927360  7.962066  0.3180807  6.409852
##   0.9    0.743693973  8.128323  0.2970259  6.625935
##   0.9    1.718029578  8.454165  0.2647376  6.973821
##   0.9    3.968871254  9.194544  0.2295790  7.674570
##   1.0    0.002118067  7.971324  0.3153667  6.354918
##   1.0    0.004893009  7.971225  0.3153787  6.354863
##   1.0    0.011303486  7.967567  0.3158393  6.352399
##   1.0    0.026112519  7.959860  0.3168538  6.346903
##   1.0    0.060323307  7.942848  0.3193218  6.337524
##   1.0    0.139354665  7.934275  0.3208405  6.354033
##   1.0    0.321927360  7.970328  0.3171393  6.422626
##   1.0    0.743693973  8.161933  0.2923939  6.664329
##   1.0    1.718029578  8.511690  0.2581813  7.030251
##   1.0    3.968871254  9.341309  0.2291685  7.812307
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.3 and lambda = 0.3219274.
```

```r
# Calculate the MSE (= RMSE^2)
(7.965442)^2
```

```
## [1] 63.44827
```

In case of the elastic net regression, the MSE is 63.44827 when alpha = 0.3 and lambda = 0.3219274.

(c) Principal component regression

```r
set.seed(0) # Ensure the reproducibility

# Fit the elastic net regression using 10 fold CV to minimize MSE
(pcr <- train(egalit_scale~., data = train_scale,
              method = 'pcr', metric = 'RMSE',
              trControl = trainControl(method = 'cv', number = 10),
              tuneLength = 25))
```

```
## Principal Component Analysis
##
## 1481 samples
##   44 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1333, 1332, 1334, 1333, 1333, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared     MAE
##    1     9.606186  0.006361916  8.040998
##    2     8.328120  0.252193770  6.751873
##    3     8.186049  0.277047321  6.566935
##    4     8.156189  0.282728286  6.541748
##    5     8.156475  0.282689107  6.541171
##    6     8.109372  0.290257530  6.503810
##    7     8.118816  0.288664486  6.511752
##    8     8.119589  0.288645500  6.512461
##    9     8.119517  0.288646874  6.512472
##   10     8.116325  0.289363604  6.511273
##   11     8.110378  0.290473587  6.501737
##   12     8.115127  0.289607235  6.508659
##   13     8.118796  0.288895620  6.512840
##   14     8.100607  0.291851875  6.484557
##   15     8.098285  0.292251624  6.484410
##   16     8.100139  0.292014483  6.477554
##   17     8.098603  0.292486398  6.477943
##   18     8.058409  0.299536327  6.444167
##   19     8.063595  0.298488470  6.443831
##   20     8.026826  0.305005690  6.412105
##   21     8.034788  0.303371277  6.411667
##   22     8.040898  0.302350809  6.420260
##   23     8.047291  0.301420489  6.426302
##   24     8.053313  0.300474965  6.433934
##   25     8.055754  0.300030545  6.441273
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 20.
```

```
# Calculate the MSE (= RMSE^2)
(8.052263)^2
```

```
## [1] 64.83894
```

In case of the principal component regression, the MSE is 64.83894 when the number of component is 20.

(d) Partial least squares regression

```r
set.seed(0) # Ensure the reproducibility

# Fit the elastic net regression using 10 fold CV to minimize MSE
(pls <- train(egalit_scale~., data = train_scale,
              method = 'pls', metric = 'RMSE',
              trControl = trainControl(method = 'cv', number = 10),
              tuneLength = 20))
```

```
## Partial Least Squares
##
## 1481 samples
##   44 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1333, 1332, 1334, 1333, 1333, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared   MAE
##    1     8.066524  0.2977267  6.434851
##    2     8.001162  0.3096623  6.395425
##    3     7.980793  0.3132387  6.363744
##    4     7.982828  0.3134459  6.360445
##    5     7.977556  0.3144310  6.356143
##    6     7.975842  0.3147450  6.358435
##    7     7.975199  0.3148393  6.358046
##    8     7.974292  0.3150010  6.356897
##    9     7.973964  0.3150458  6.356680
##   10     7.974097  0.3150265  6.356763
##   11     7.974065  0.3150328  6.356692
##   12     7.974038  0.3150353  6.356649
##   13     7.974047  0.3150344  6.356649
##   14     7.974043  0.3150348  6.356648
##   15     7.974047  0.3150343  6.356650
##   16     7.974045  0.3150344  6.356649
##   17     7.974045  0.3150345  6.356649
##   18     7.974045  0.3150345  6.356649
##   19     7.974045  0.3150345  6.356649
##   20     7.974045  0.3150345  6.356649
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 9.
```

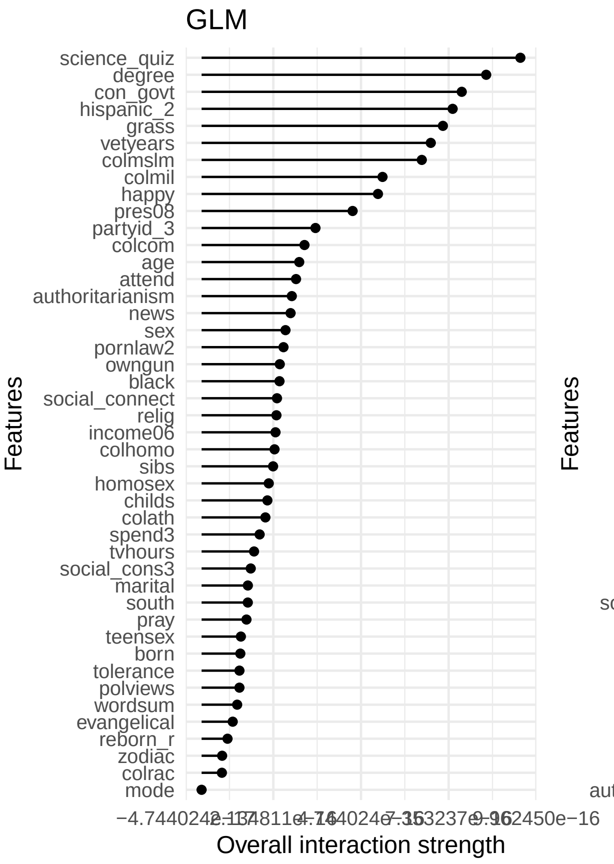
```r
# Calculate the MSE (= RMSE^2)
(7.973964)^2
```

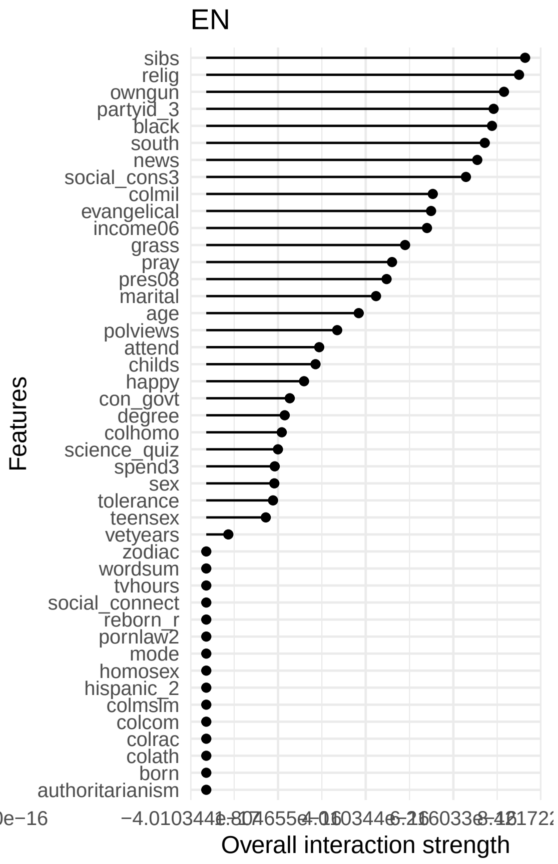
```
## [1] 63.5841
```

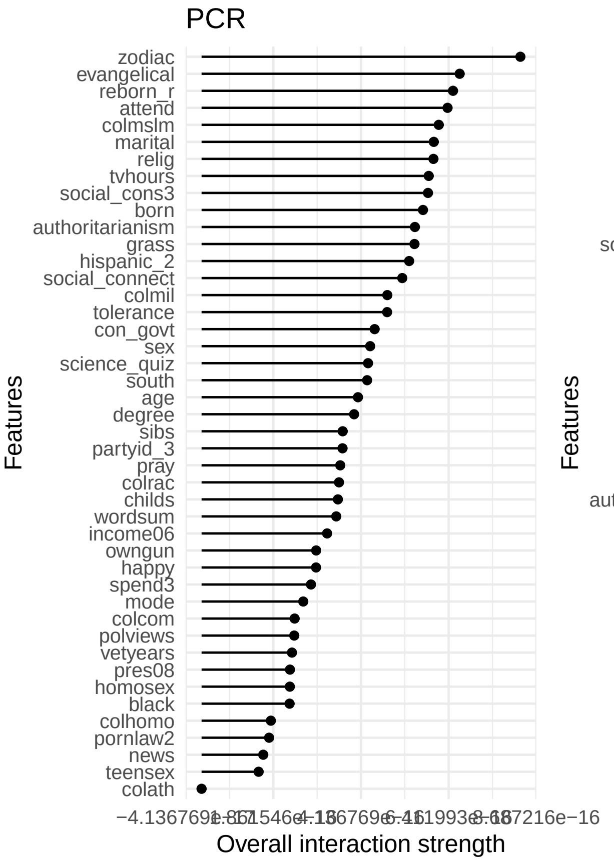
In case of the principal component regression, the MSE is 63.5841 when the number of component is 9.

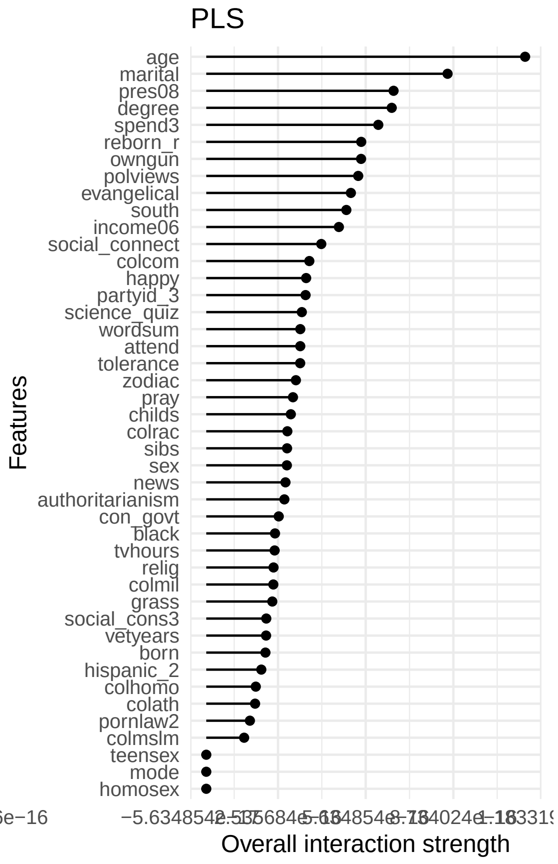
Question 5

```r
set.seed(0) #Ensure the reproducibility

# Prepare features and response of test data
features <- test_scale %>% select(-egalit_scale)
response <- test_scale %>% select(egalit_scale)

# Create predictor object to pass to explainer functions
predictor.glm <- Predictor$new(
  model = glm, data = features, y = response, predict.fun = predict)

predictor.en <- Predictor$new(
  model = en, data = features, y = response, predict.fun = predict)

predictor.pcr <- Predictor$new(
  model = pcr, data = features, y = response, predict.fun = predict)

predictor.pls <- Predictor$new(
  model = pls, data = features, y = response, predict.fun = predict)

# Measure INXNs: identify features with large interaction effects in each model
interact.glm <- Interaction$new(predictor.glm) %>%
  plot() + ggtitle("GLM")

interact.en  <- Interaction$new(predictor.en) %>%
  plot() + ggtitle("EN")

interact.pcr <- Interaction$new(predictor.pcr) %>%
  plot() + ggtitle("PCR")

interact.pls <- Interaction$new(predictor.pls) %>%
  plot() + ggtitle("PLS")

# Plot INXNs in each model
grid.arrange(interact.glm, interact.en, interact.pcr, interact.pls, nrow = 2, ncol = 2)
```


GLM
Features
Overall interaction strength
science_quiz
degree
con_govt
hispanic_2
grass
vetyears
colmslm
colmil
happy
pres08
partyid_3
colcom
age
attend
authoritarianism
news
sex
pornlaw2
owngun
black
social_connect
relig
income06
colhomo
sibs
homosex
childs
colath
spend3
tvhours
social_cons3
marital
south
pray
teensex
born
tolerance
polviews
wordsum
evangelical
reborn_r
zodiac
colrac
mode


EN
Features
Overall interaction strength
sibs
relig
owngun
partyid_3
black
south
news
social_cons3
colmil
evangelical
income06
grass
pray
pres08
marital
age
polviews
attend
childs
happy
con_govt
degree
colhomo
science_quiz
spend3
sex
tolerance
teensex
vetyears
zodiac
wordsum
tvhours
social_connect
reborn_r
pornlaw2
mode
homosex
hispanic_2
colmslm
colcom
colrac
colath
born
authoritarianism


PCR
Features
Overall interaction strength
zodiac
evangelical
reborn_r
attend
colmslm
marital
relig
tvhours
social_cons3
born
authoritarianism
grass
hispanic_2
social_connect
colmil
tolerance
con_govt
sex
science_quiz
south
age
degree
sibs
partyid_3
pray
colrac
childs
wordsum
income06
owngun
happy
spend3
mode
colcom
polviews
vetyears
pres08
homosex
black
colhomo
pornlaw2
news
teensex
colath


PLS
Features
Overall interaction strength
age
marital
pres08
degree
spend3
reborn_r
owngun
polviews
evangelical
south
income06
social_connect
colcom
happy
partyid_3
science_quiz
wordsum
attend
tolerance
zodiac
pray
childs
colrac
sibs
sex
news
authoritarianism
con_govt
black
tvhours
relig
colmil
grass
social_cons3
vetyears
born
hispanic_2
colhomo
colath
pornlaw2
colmslm
teensex
mode
homosex

- The value of feature importance is how much the predicted outcome changes when the term is omitted. Looking at “the overall interaction strength”, they are almost zero in every model (e-16 or e-17). Especially in elastic net regression, many features (about 15) have low interaction strengths.
- Basically, the feature importances are ranked differently across different models. For example, “reborn_r” is higher rank in PCR and PLS while it is lower rank in GLM and EN. On the other hand, some features are in similar ranks. For example, “attend” are a little above the middle. However, since there seems to be no feature in the top rank of every model, we cannot surely say “this feature is important”.