# Xu_Yilun_HW04

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2

## -- Attaching packages ------------------------------ tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'dplyr' was built under R version 3.6.2

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

```r
library(rcfss)
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 3.6.2
```

```r
library(margins)
```

```
## Warning: package 'margins' was built under R version 3.6.2
```

```r
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library(splines)
library(tibble)
library(dplyr)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 3.6.2
```

```r
library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 3.6.2

## For binary classification, the first factor level is assumed to be the event.
## Set the global option `yardstick.event_first` to `FALSE` to change this.

##
```

```
## Attaching package: 'yardstick'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.2
```

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

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

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall
```

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

```r
library(dvmisc)
```

```
## Warning: package 'dvmisc' was built under R version 3.6.2
```

```
## Loading required package: rbenchmark
```

```
##
## Attaching package: 'dvmisc'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand_grid
```

```r
library(inum)
```

```
## Warning: package 'inum' was built under R version 3.6.2
```

```r
library(iml)
```

```
## Warning: package 'iml' was built under R version 3.6.2
```

```r
set.seed(1234)
```

Egalitarianism and income 1

```r
g_train <- read_csv("C:/Users/mac/Desktop/hw04/gss_train.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   age = col_double(),
##   authoritarianism = col_double(),
##   childs = col_double(),
##   con_govt = col_double(),
##   egalit_scale = col_double(),
##   income06 = col_double(),
##   science_quiz = col_double(),
##   sibs = col_double(),
##   social_connect = col_double(),
##   tolerance = col_double(),
##   tvhours = col_double(),
##   wordsum = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
g_test <- read_csv("C:/Users/mac/Desktop/hw04/gss_test.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   age = col_double(),
##   authoritarianism = col_double(),
##   childs = col_double(),
##   con_govt = col_double(),
##   egalit_scale = col_double(),
##   income06 = col_double(),
##   science_quiz = col_double(),
##   sibs = col_double(),
##   social_connect = col_double(),
##   tolerance = col_double(),
##   tvhours = col_double(),
##   wordsum = col_double()
## )
## See spec(...) for full column specifications.
```

```
i06_pred <- function(d, .data){.data %>%
    mutate(model = map(splits, ~ lm(egalit_scale ~
                                      poly(income06, degree = d,
                                           raw = TRUE),
                                    data = analysis(.x))),
           estimate = map2(splits, model,
                           ~ predict(.y, newdata = assessment(.x))),
           truth = map(splits, ~ assessment(.x)$egalit_scale)) %>%
    unnest(truth, estimate) %>%
    group_by(id) %>%
    mse(truth = truth, estimate = estimate) %$%
    mean(.estimate)}
gss_cv <- vfold_cv(g_train, v = 10)

i06_mse <- tibble(d = 1:15) %>%
  mutate(mse = map_dbl(d, i06_pred, gss_cv))
```

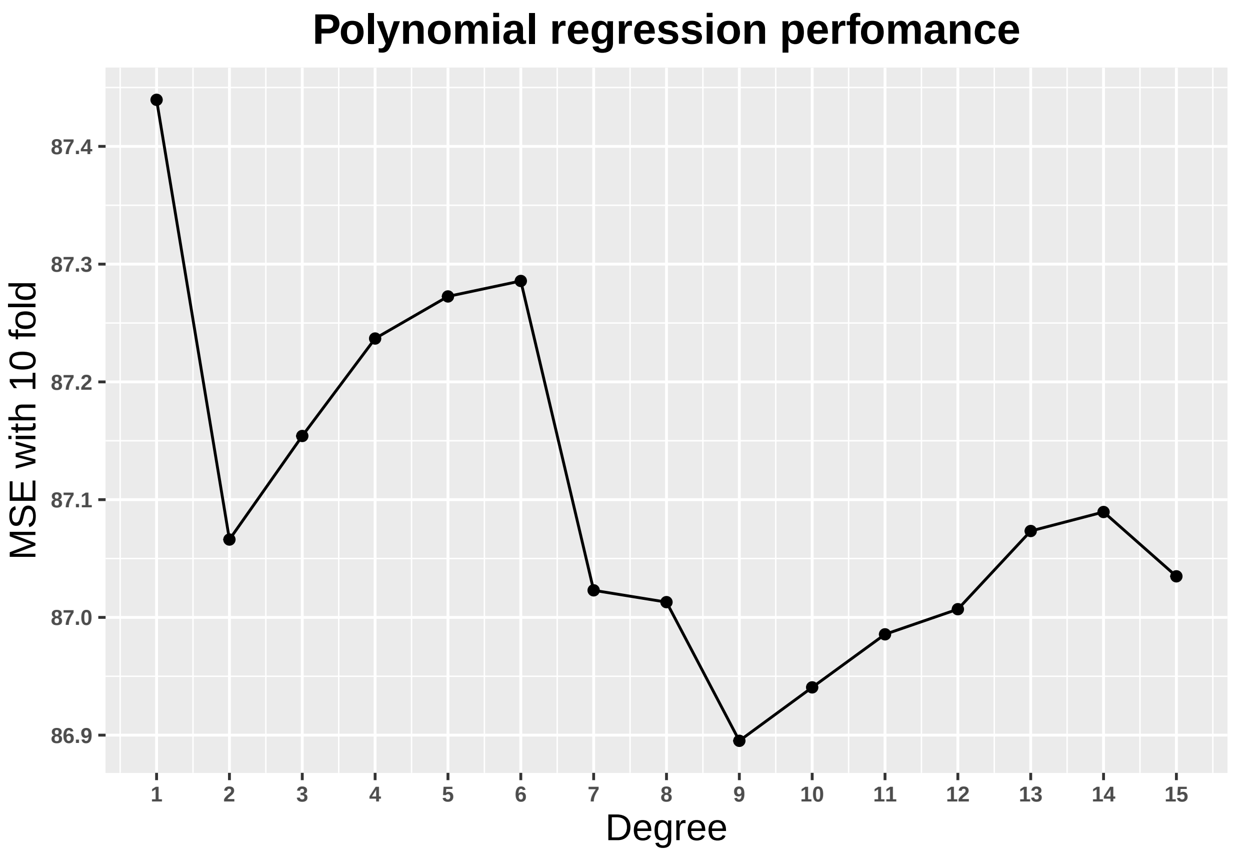
```
## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed
## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed
```

```
## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading
```

```
## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed
```

```
## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading
## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed
## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading
## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed
## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading
## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(.y, newdata = assessment(.x)): prediction from a rank-
## deficient fit may be misleading

## Warning: unnest() has a new interface. See ?unnest for details.
## Try `df %>% unnest(c(truth, estimate))`, with `mutate()` if needed
```

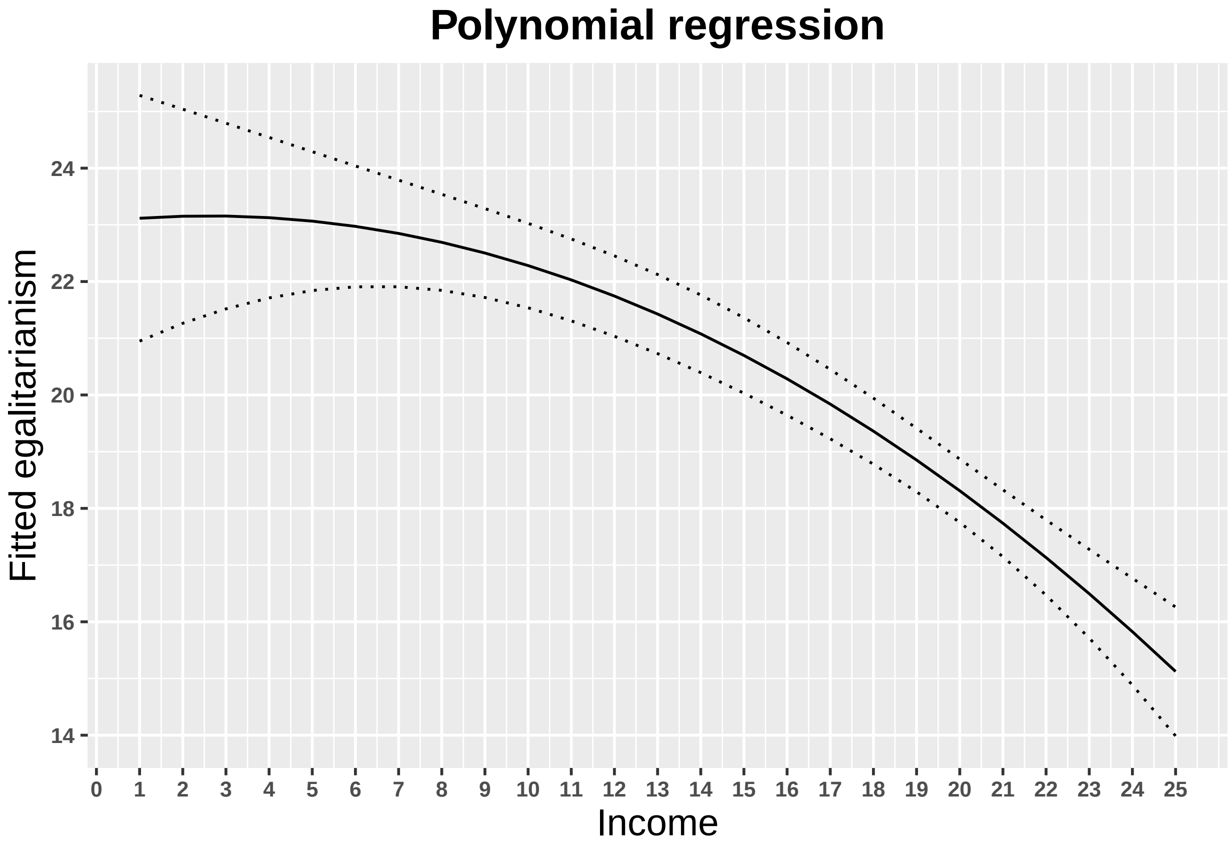
```
ggplot(i06_mse, aes(d, mse)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = seq(0, 15, 1)) +
  labs(title = "Polynomial regression perfomance",
       x = "Degree", y = "MSE with 10 fold") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 8, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



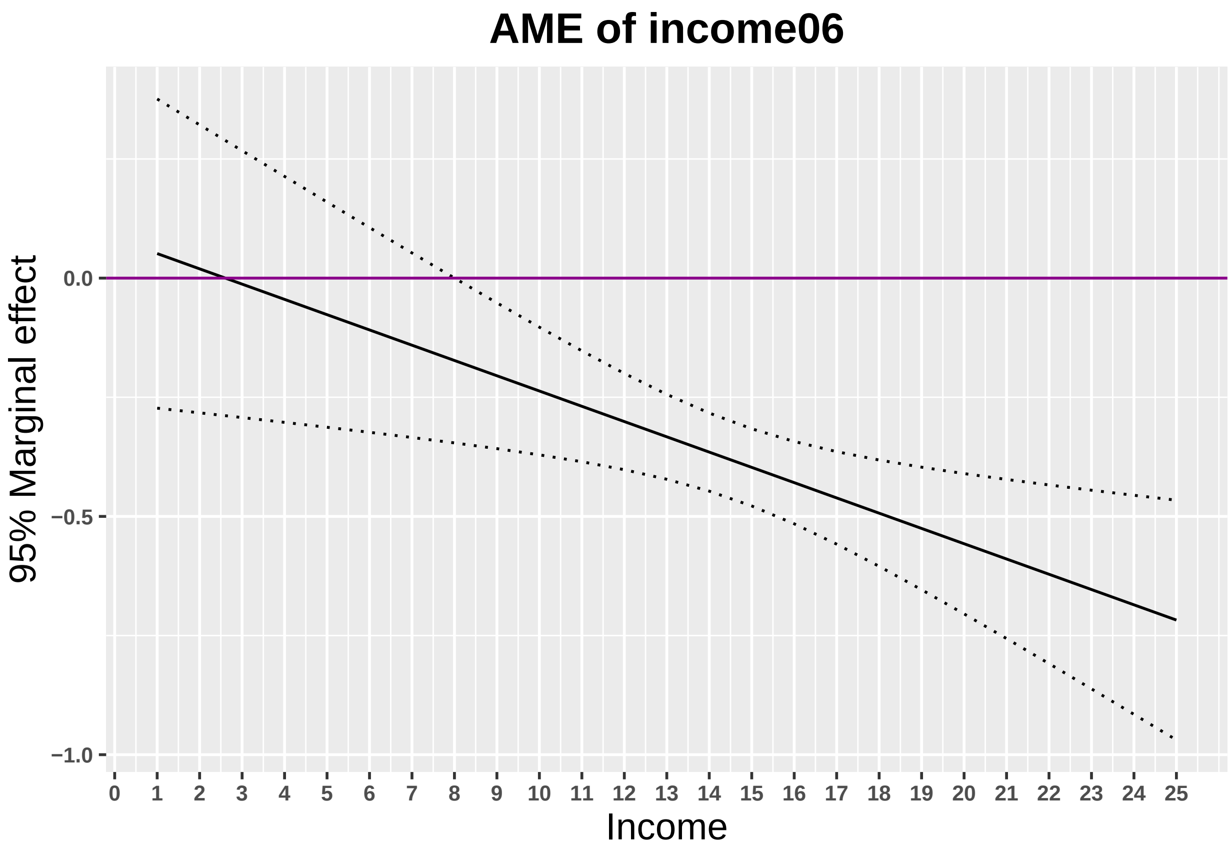
```
i06_egalit <- lm(egalit_scale ~ income06 + I(income06^2),
                    data = g_train)

for_plot_egalit <- cplot(i06_egalit, "income06", what =
                           "prediction", 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(for_plot_egalit, aes(x = xvals)) +
  scale_x_continuous(breaks = seq(0, 25, 1)) +
  geom_line(aes(y = yvals)) +
  geom_line(aes(y = upper), linetype = 3) +
  geom_line(aes(y = lower), linetype = 3) +
  labs(x = "Income", y = "Fitted egalitarianism",
       title = "Polynomial regression") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 8, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```

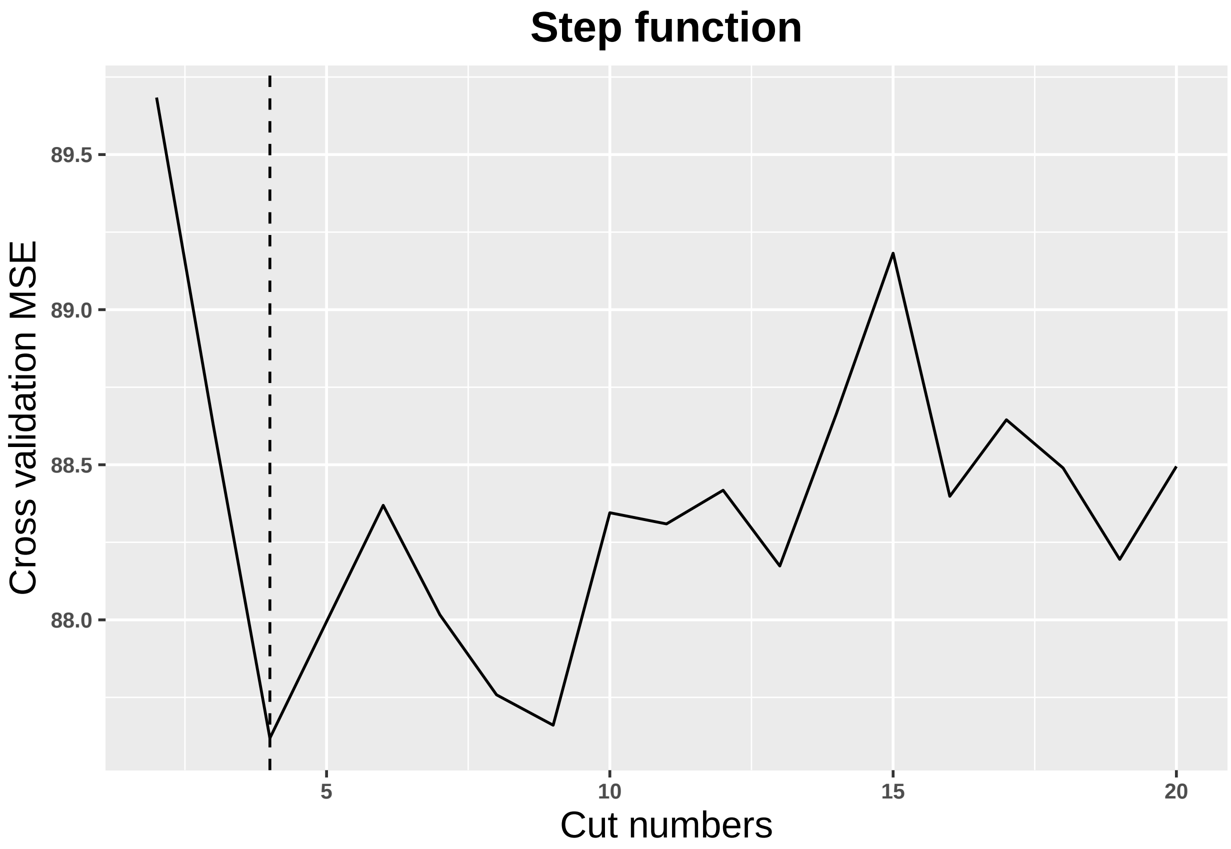


```
for_plot_effect <- cplot(i06_egalit, "income06", what = "effect",
                         draw = FALSE)

ggplot(for_plot_effect, aes(x = xvals)) +
  geom_line(aes(y = yvals)) +
  scale_x_continuous(breaks = seq(0, 25, 1)) +
  geom_line(aes(y = upper), linetype = 3) +
  geom_line(aes(y = lower), linetype = 3) +
  geom_hline(yintercept = 0, color = 'darkmagenta') +
  labs(title = "AME of income06",
       x = "Income", y = "95% Marginal effect") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                face = "bold"),
        axis.text = element_text(size = 8, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```

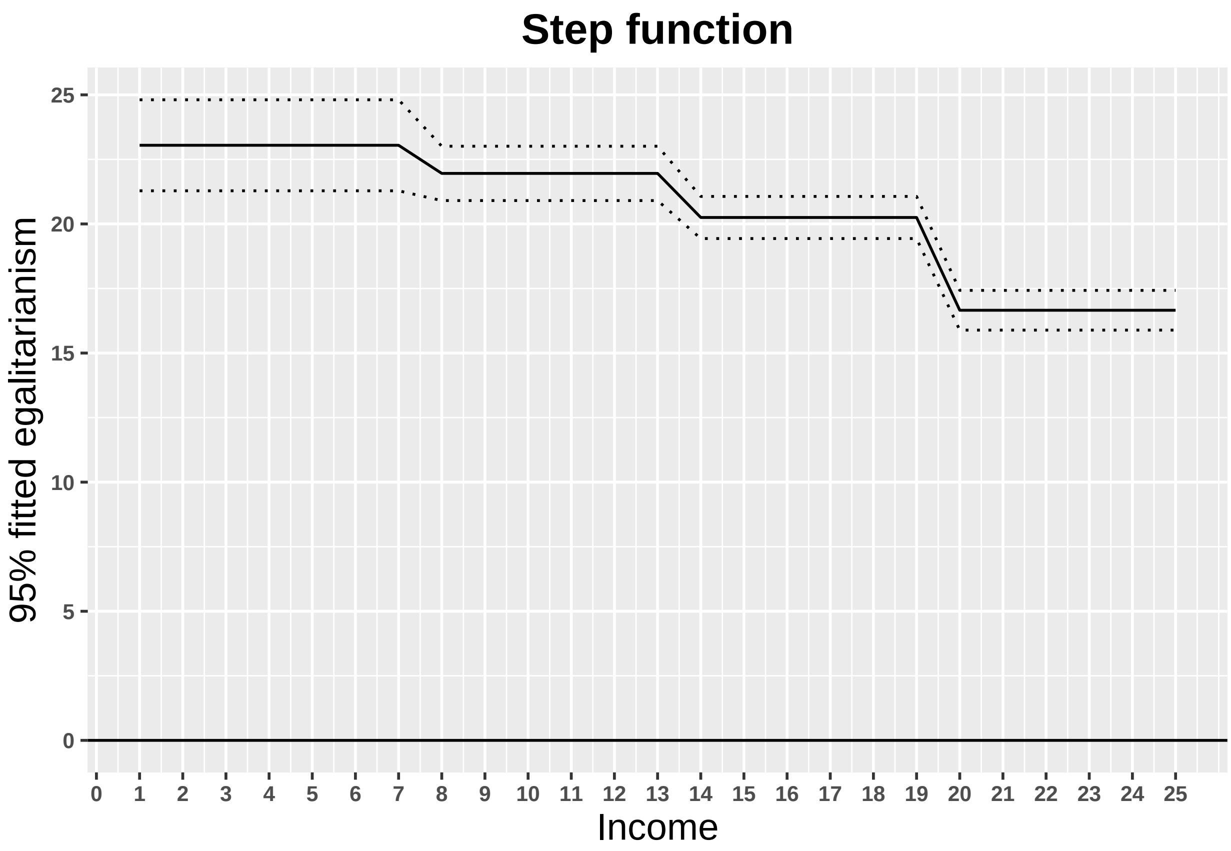


2

```
step_error <- vector(length = 19, mode = "numeric")

for (i in 2:20) {g_train$income06_cut <-
  cut_interval(g_train$income06, i)

fit_model <- glm(egalit_scale ~ income06_cut, data = g_train)

step_error[i - 1] <- boot::cv.glm(g_train, fit_model, K =
                                    10)$delta[1]}

tibble(n_cut = 2:20, mse = step_error) %>%
  ggplot(aes(n_cut, mse)) +
  geom_line() +
  geom_vline(xintercept = which.min(step_error) + 1,
             linetype = 2) +
  labs(title = "Step function",
       x = "Cut numbers", y = "Cross validation MSE") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                face = "bold"),
        axis.text = element_text(size = 8, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```

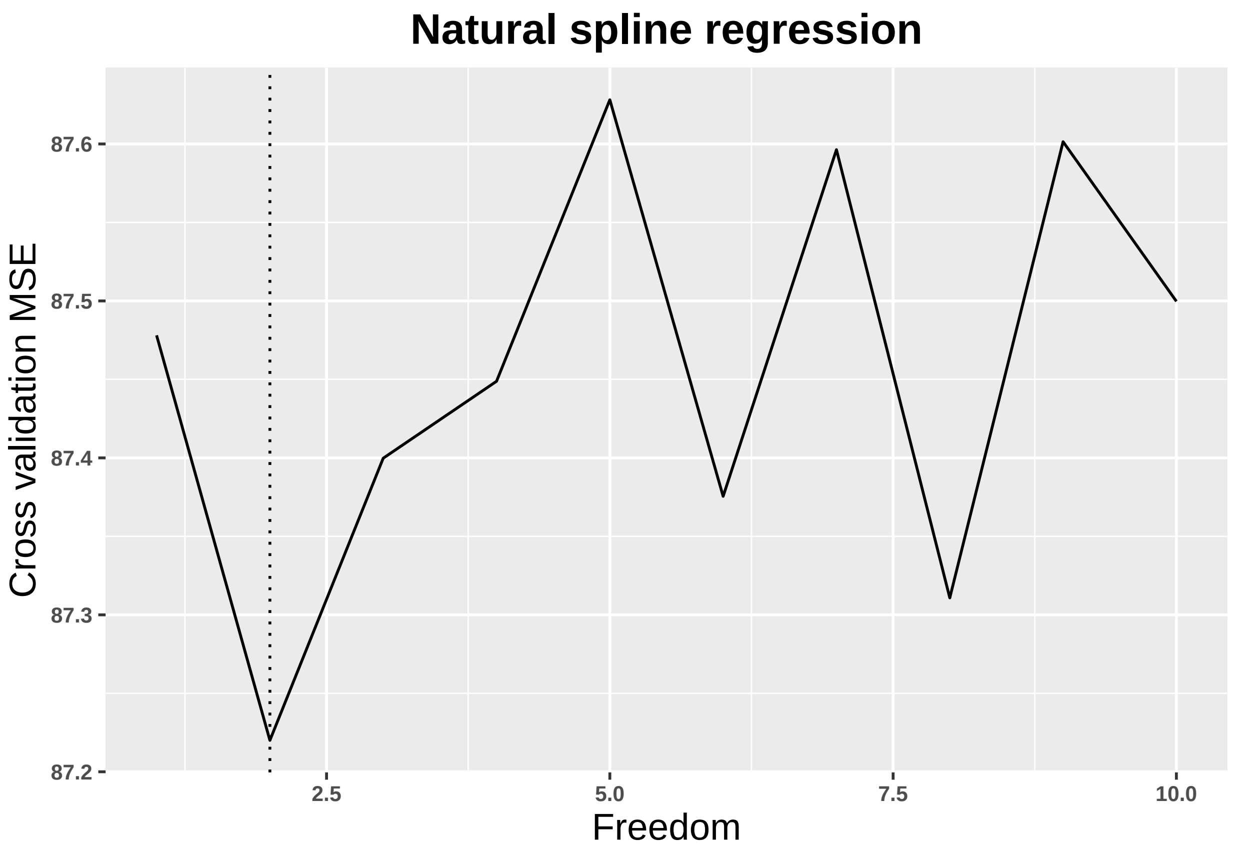


```
i06_step <- lm(egalit_scale ~
                     cut_interval(income06, which.min(step_error)
                                  + 1), data = g_train)

i06_step %>% prediction %>% ggplot(aes(x = income06)) +
  geom_line(aes(y = fitted)) +
  geom_line(aes(y = fitted + 1.96 * se.fitted), linetype = 3) +
  geom_line(aes(y = fitted - 1.96 * se.fitted), linetype = 3) +
  geom_hline(yintercept = 0, linetype = 1) +
  scale_x_continuous(breaks = seq(0, 25, 1)) +
  labs(title = "Step function",
       x = "Income", y = "95% fitted egalitarianism") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 8, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```

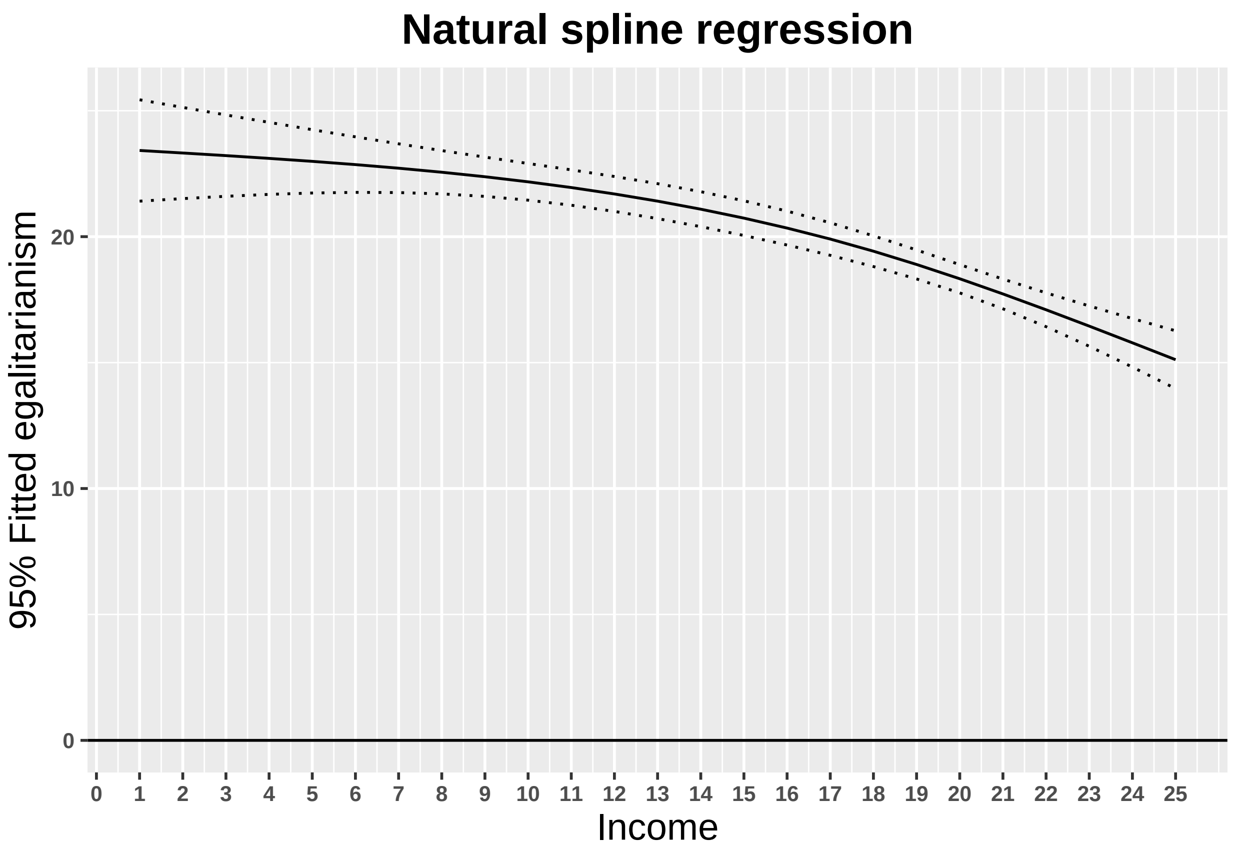


3

```
n_error <- vector(mode = "numeric", length = 10)

for (i in 1:10) {glm.fit <- glm(egalit_scale ~
                                  ns(income06, df = i), data =
                                  g_train)

n_error[i] <- boot::cv.glm(g_train, glm.fit, K = 10)$delta[1]}

tibble(df = 1:10, mse = n_error) %>%
  ggplot(aes(df, mse)) + geom_line() +
  geom_vline(xintercept = which.min(n_error),
             linetype = 3) +
  labs(title = "Natural spline regression", x = "Freedom",
       y = "Cross validation MSE") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 8, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



```
i06_natural <- lm(egalit_scale ~
                    ns(income06, df = which.min(n_error)),
                  data = g_train)
i06_natural %>% prediction %>%
  ggplot(aes(x = income06)) +
  geom_line(aes(y = fitted)) +
  geom_line(aes(y = fitted + 1.96 * se.fitted), linetype = 3) +
  geom_line(aes(y = fitted - 1.96 * se.fitted), linetype = 3) +
  geom_hline(yintercept = 0, linetype = 1) +
  scale_x_continuous(breaks = seq(0, 25, 1)) +
  labs(title = "Natural spline regression",
       x = "Income", y = "95% Fitted egalitarianism") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 8, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



4 Linear Regression

```
g_train <- read_csv("C:/Users/mac/Desktop/hw04/gss_train.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   age = col_double(),
##   authoritarianism = col_double(),
##   childs = col_double(),
##   con_govt = col_double(),
##   egalit_scale = col_double(),
##   income06 = col_double(),
##   science_quiz = col_double(),
##   sibs = col_double(),
##   social_connect = col_double(),
##   tolerance = col_double(),
##   tvhours = col_double(),
##   wordsum = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
g_test <- read_csv("C:/Users/mac/Desktop/hw04/gss_test.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   age = col_double(),
##   authoritarianism = col_double(),
##   childs = col_double(),
##   con_govt = col_double(),
##   egalit_scale = col_double(),
##   income06 = col_double(),
##   science_quiz = col_double(),
##   sibs = col_double(),
##   social_connect = col_double(),
##   tolerance = col_double(),
##   tvhours = col_double(),
##   wordsum = col_double()
## )
## See spec(...) for full column specifications.
```

```
g_lm <- train(egalit_scale ~ ., data = g_train, method = "lm",
              metric = "RMSE", preProcess = c("zv"),
              trControl = trainControl(method = "cv",
                                       number = 10))
g_lm
```

```
## Linear Regression
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1333, 1334, 1333, 1332, 1332, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.957693  0.3276001  6.297669
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Elastic net regression

```
g_elastic <- train(egalit_scale ~ ., data = g_train,
                   method = "glmnet", metric = "RMSE",
                   trControl = trainControl(method = "cv",
                                            number = 10),
                   preProcess = c("zv", "center", "scale"),
                   tuneLength = 10)
```

Principal component regression

```
g_pcr <- train(egalit_scale ~ ., data = g_train, method = "pcr",
             trControl = trainControl(method = "cv", number = 10),
             metric = "RMSE", tuneLength = 20,
             preProcess = c("zv", "center", "scale"))
```

Partial least squares regression

```
g_pls <- train(egalit_scale ~ ., data = g_train, method = "pls",
             tuneLength = 20, metric = "RMSE",
             trControl = trainControl(method = "cv", number = 10),
```

```r
                preProcess = c("zv", "center", "scale"))
```

MSE of different models

```r
(models_performance <- list(Linear_regression = g_lm,
                            Elastic_net = g_elastic,
                            Principle_component = g_pcr,
                            Partial_least_squares = g_pls) %>%
   resamples %>% summary %$% (statistics$RMSE)^2)
```

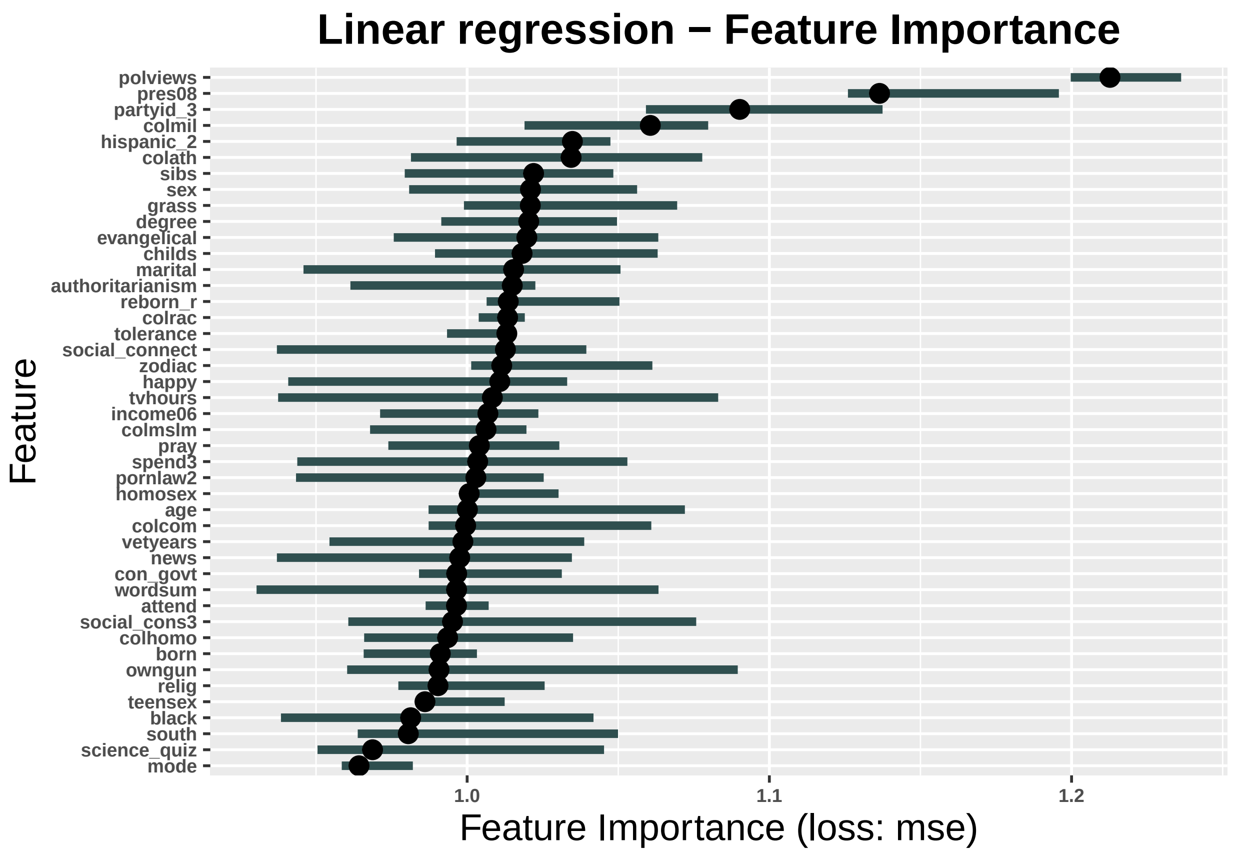
```
##                           Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## Linear_regression     52.11634 58.61215 64.55210 63.32488 69.48664 74.73067
## Elastic_net           54.34838 55.71985 59.31238 60.41362 61.97279 70.91825
## Principle_component   54.03026 61.07557 65.31945 64.71798 70.31291 73.16048
## Partial_least_squares 54.24157 61.29937 63.38260 63.32856 65.41169 69.40480
##                       NA's
## Linear_regression        0
## Elastic_net              0
## Principle_component      0
## Partial_least_squares    0
```

5

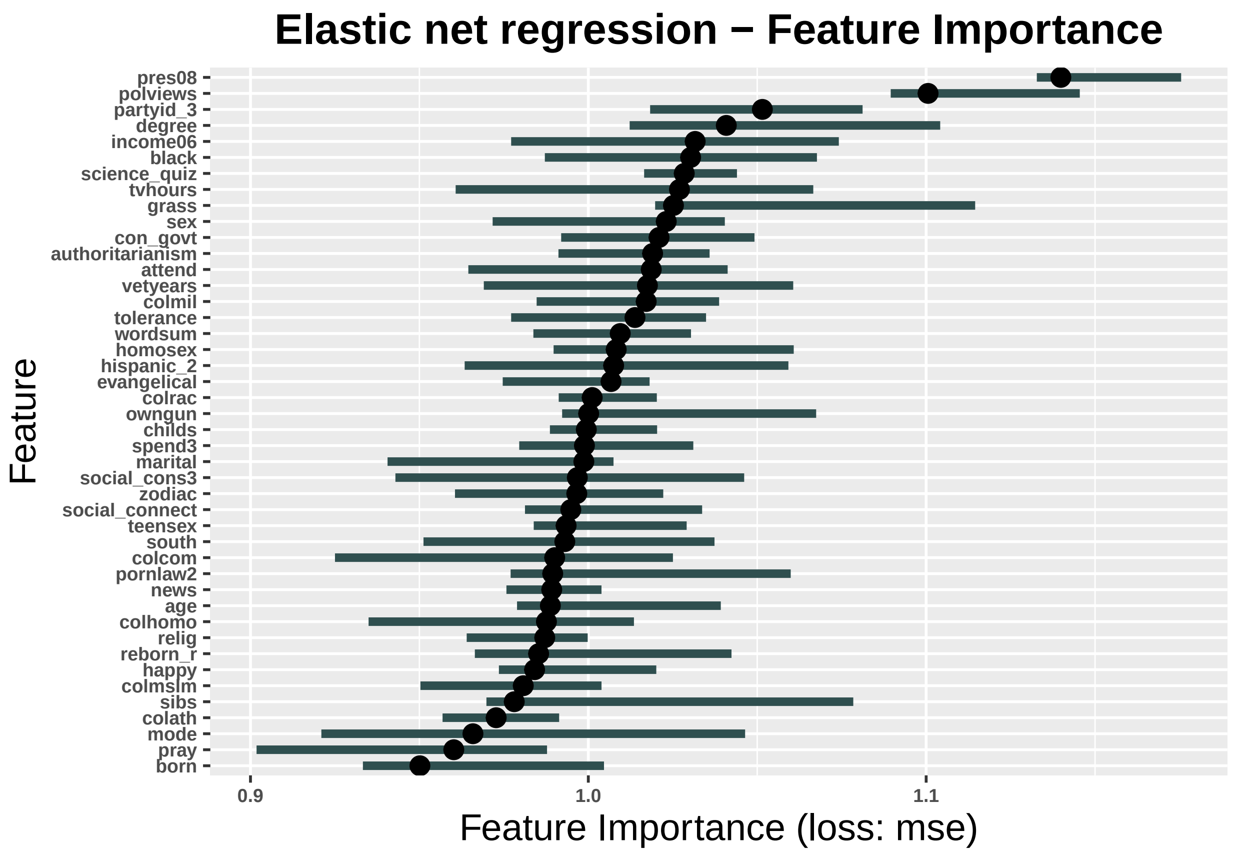
```r
g_train_part <- select(g_train, -egalit_scale)
test_lm <- Predictor$new(model = g_lm, data = g_train_part,
                              y = g_train$egalit_scale)

test_elastic <- Predictor$new(model = g_elastic,
                              data = g_train_part,
                              y = g_train$egalit_scale)

test_pcr <- Predictor$new(model = g_pcr, data = g_train_part,
                          y = g_train$egalit_scale)

test_pls <- Predictor$new(model = g_pls, data = g_train_part,
                          y = g_train$egalit_scale)
```

Feature importance

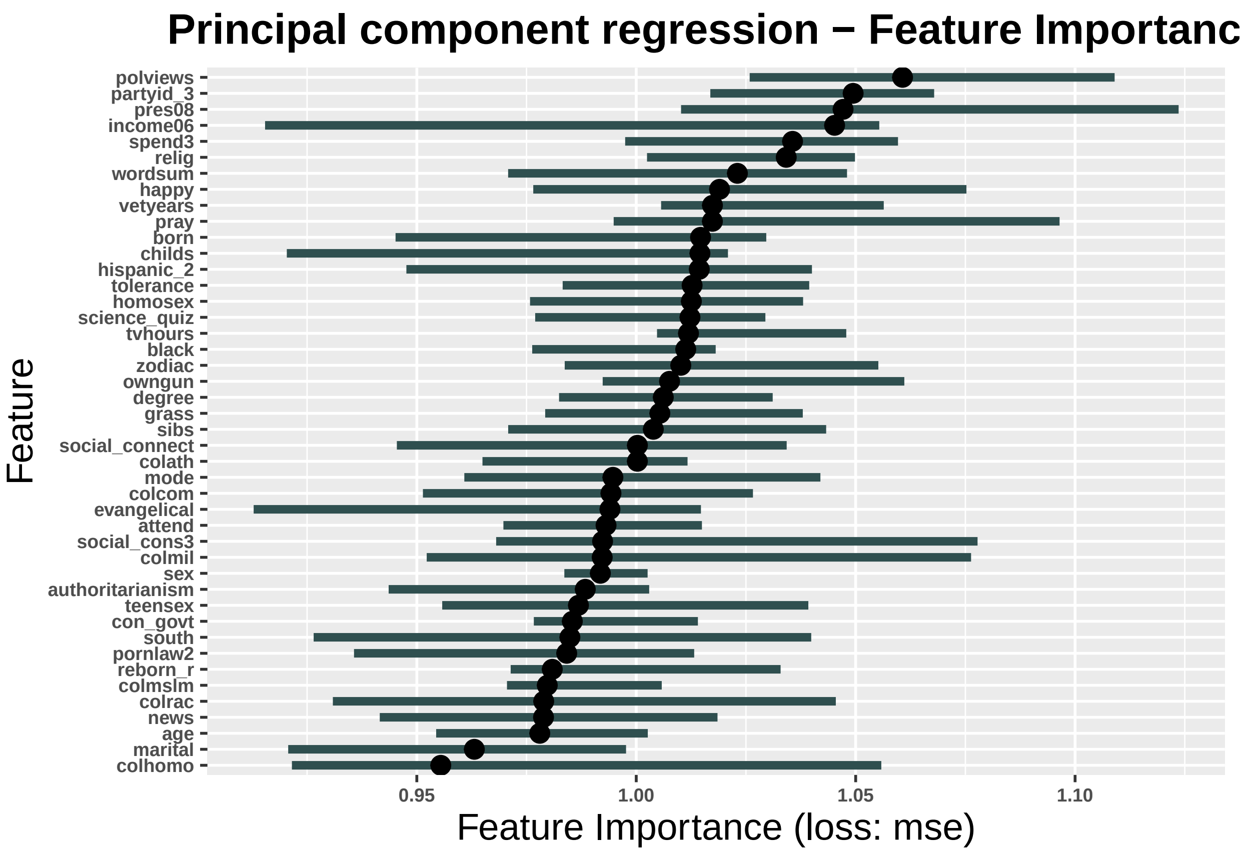
```r
imp_lm <- FeatureImp$new(test_lm, loss = "mse")
imp_pcr <- FeatureImp$new(test_pcr, loss = "mse")
imp_pls <- FeatureImp$new(test_pls, loss = "mse")
imp_elastic <- FeatureImp$new(test_elastic, loss = "mse")

#Linear regression
plot(imp_lm) + ggtitle("Linear regression - Feature Importance") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



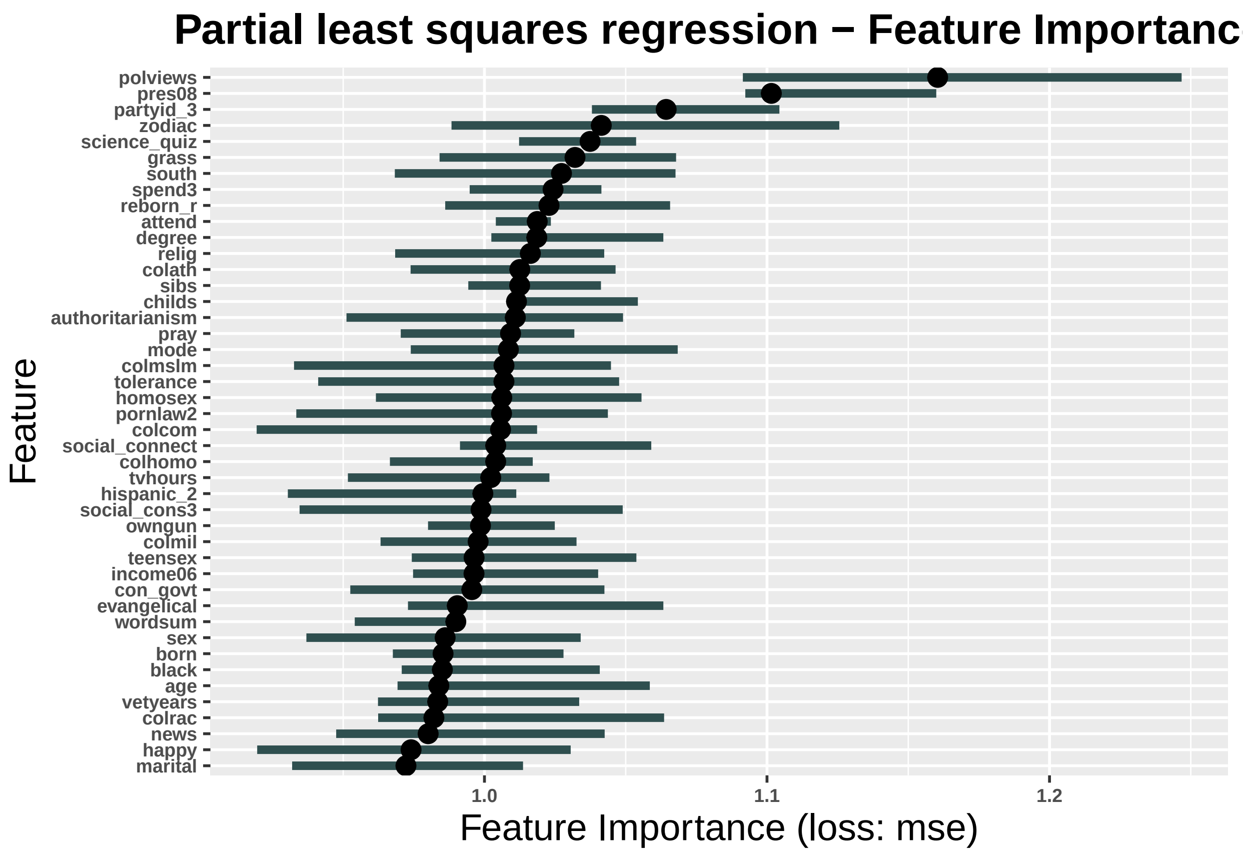
```r
#Elastic net regression
plot(imp_elastic) + ggtitle("Elastic net regression - Feature Importance") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



```r
#Principal component regression
plot(imp_pcr) + ggtitle("Principal component regression - Feature Importance") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```

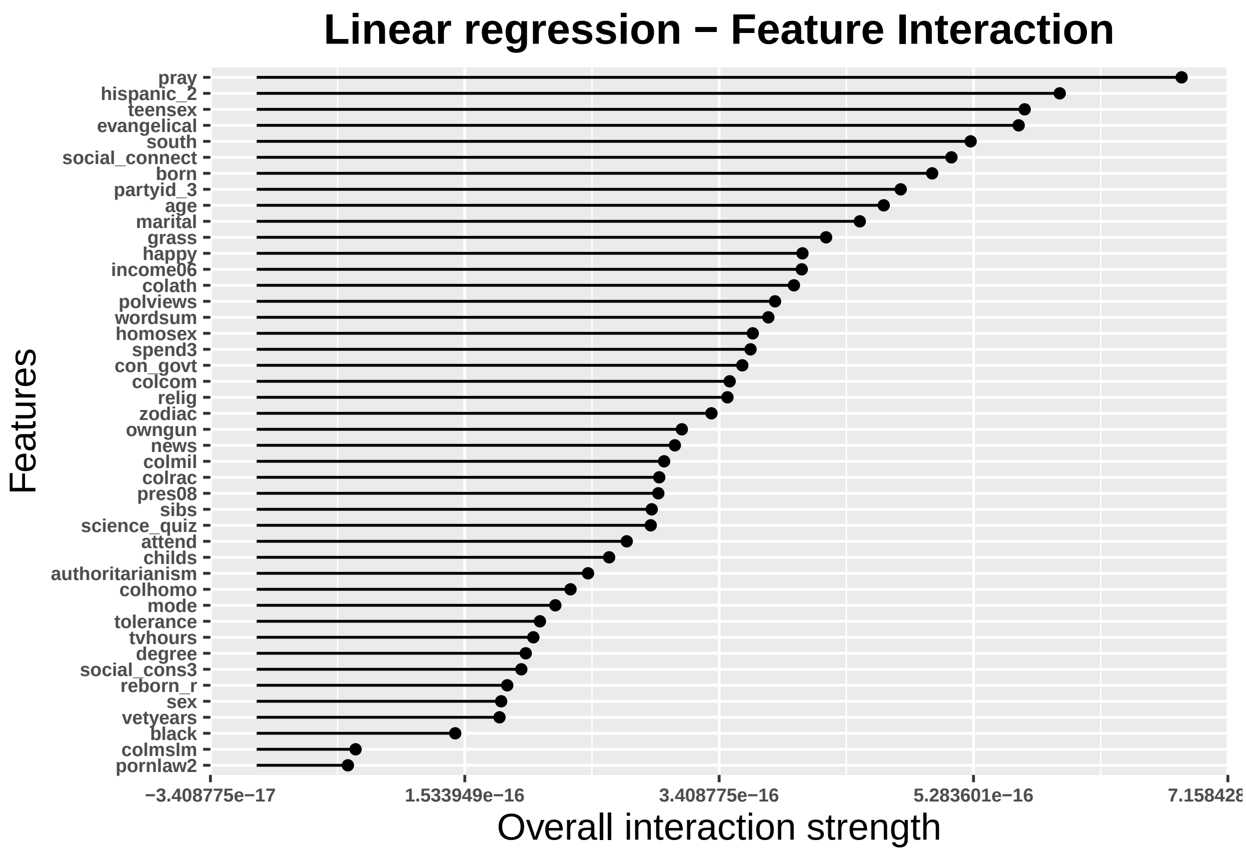


```r
#Partial least squares regression
plot(imp_pls) + ggtitle("Partial least squares regression - Feature Importance") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```

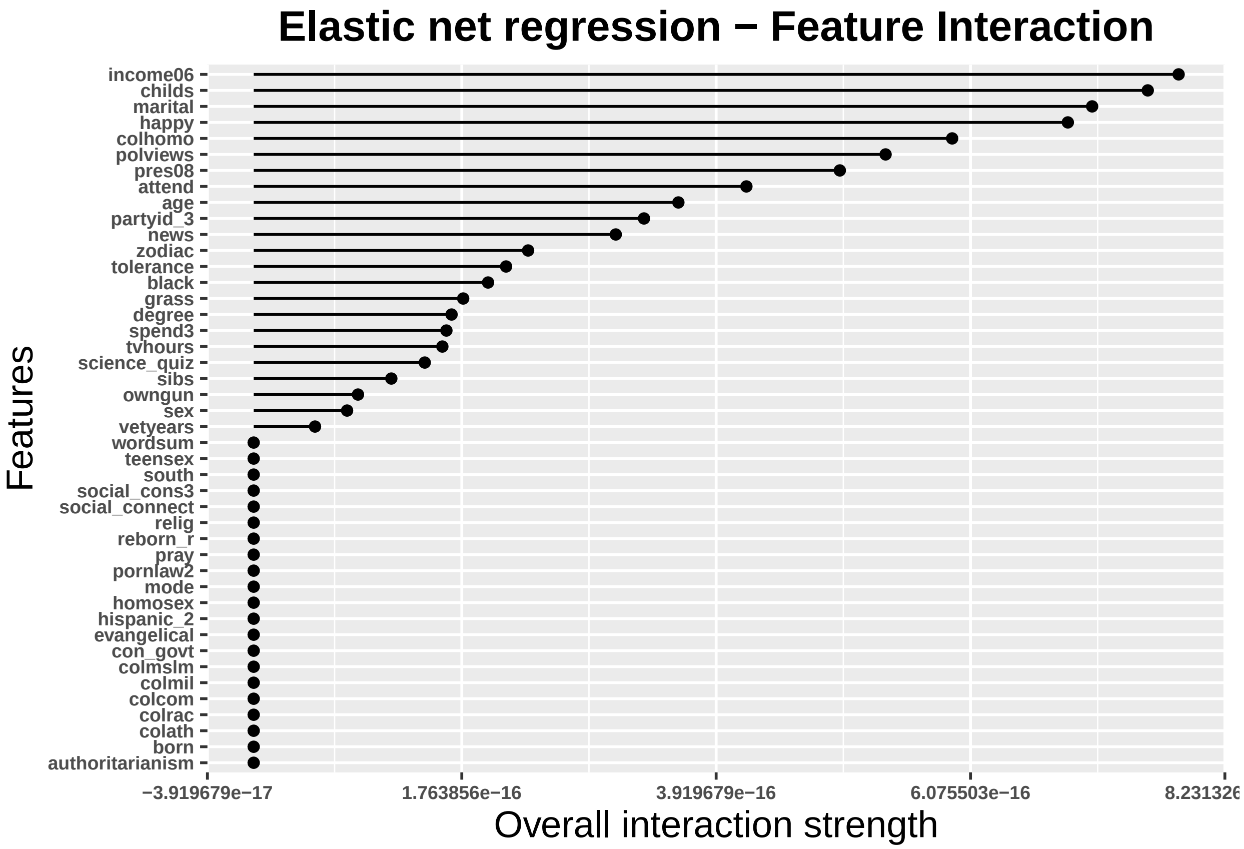


Feature Interaction

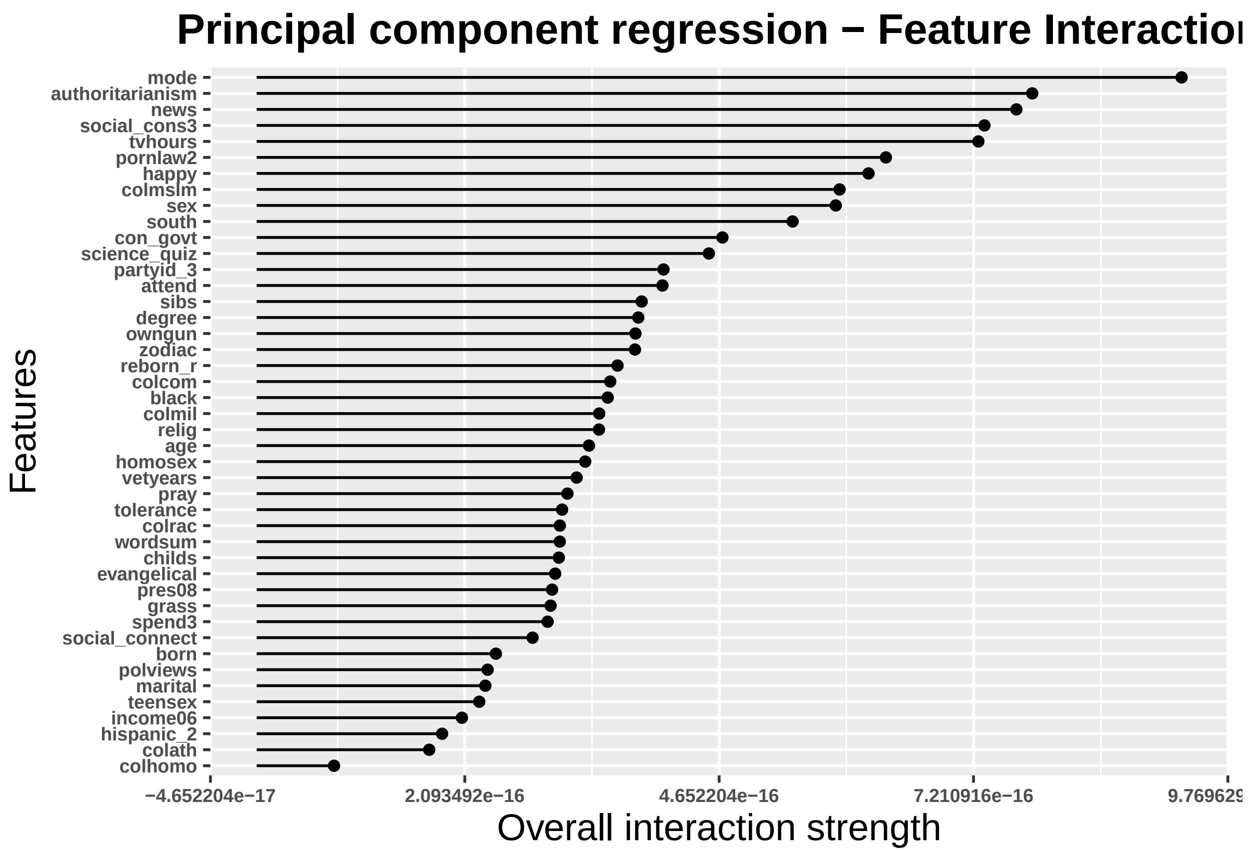
```
fi_lm <- Interaction$new(test_lm)
fi_pcr <- Interaction$new(test_pcr)
fi_pls <- Interaction$new(test_pls)
fi_elastic <- Interaction$new(test_elastic)

#Linear regression
plot(fi_lm) + ggtitle("Linear regression - Feature Interaction") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



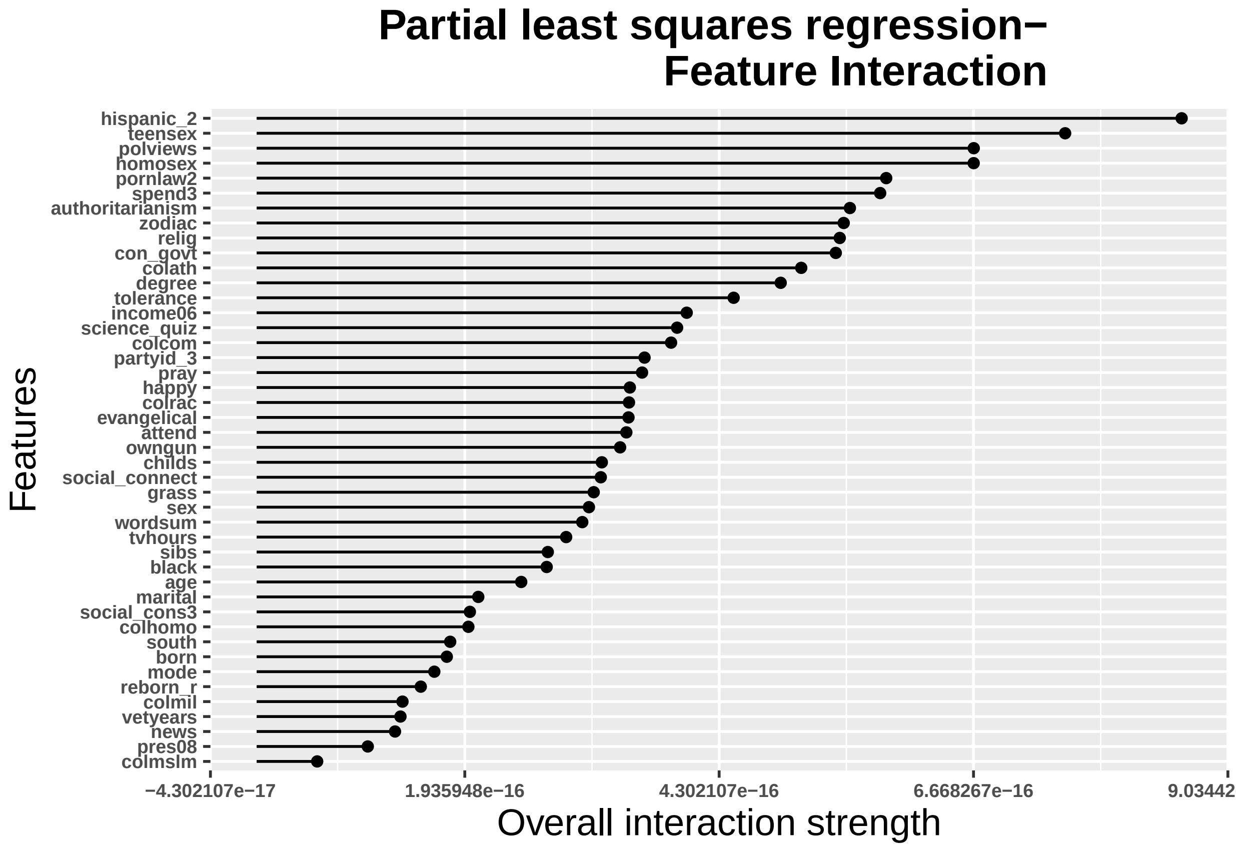
```r
#Elastic net regression
plot(fi_elastic) + ggtitle("Elastic net regression - Feature Interaction") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



```
#Principal component regression
plot(fi_pcr) + ggtitle("Principal component regression - Feature Interaction") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



```
#Partial least squares regression
plot(fi_pls) + ggtitle("Partial least squares regression-
                       Feature Interaction") +
  theme(plot.title = element_text(hjust = 0.5,size = 16,
                                  face = "bold"),
        axis.text = element_text(size = 7, face = "bold"),
        axis.title.x = element_text(size = 14),
        axis.title.y = element_text(size = 14))
```



We can see that in different models, the importance of different variables are different. However, we can also see that the feature importances of different variables are almost around 1%. There are no variables that account for a significantly large proportion in a specific model. In addition, according to the above feature interaction plots, we can see that the interaction effects between the features are very weak (below very small proportions of variance explained per feature).