Zhuang_Chu_HW4

Chu Zhuang

2020/2/16

Homework 4: Moving Beyond Linearity

by Chu Zhuang

```
library(tidyverse)
```

```
## -- 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   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

```
library(tidymodels)
```

```
## -- Attaching packages ------------------------------------------------ tidymodels 0.0.3 --
```

```
## v broom     0.5.4     v recipes   0.1.9
## v dials     0.0.4     v rsample   0.0.5
## v infer     0.5.1     v yardstick 0.0.5
## v parsnip   0.0.5
```

```
## -- Conflicts ------------------------------------------------------ tidymodels_conflicts() --
## x scales::discard()   masks purrr::discard()
## x dplyr::filter()     masks stats::filter()
## x recipes::fixed()    masks stringr::fixed()
## x dplyr::lag()        masks stats::lag()
## x dials::margin()     masks ggplot2::margin()
## x yardstick::spec()   masks readr::spec()
## x recipes::step()     masks stats::step()
## x recipes::yj_trans() masks scales::yj_trans()
```

```
library(knitr)
library(splines)
library(lattice)
library(here)
```

```
## here() starts at C:/Users/zhuangchu/Desktop/Model/problem-set-4
```

```
library(patchwork)
library(margins)
library(caret)
```

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

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

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

```
library(ggplot2)
library(magrittr)
```

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

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

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

```
library(rcfss)
library(iml)
library(devtools)
```

```
## Loading required package: usethis

##
## Attaching package: 'devtools'

## The following object is masked from 'package:recipes':
##
##     check
```

```
library(png)

options(digits=3)
set.seed(123)
```

Egalitarianism and income

First of all Load the data for training:

```
gss_train<-read_csv("data/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.
```

Then, Load the data for test:

```
gss_test<-read_csv("data/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.
```

1. Polynomial Regression

For question number one, perform polynomial regression to predict egalitarianism based on income06, and to find optimal degree by 10-fold cross-validation based on MSE.

I first try with the *polywog* package to conduct polynomial regression with cross validation; however, at last I found this package also takes lasso/ridge regression form in it. It is not purely polynomial regression, while I still put the results here for better comparison.

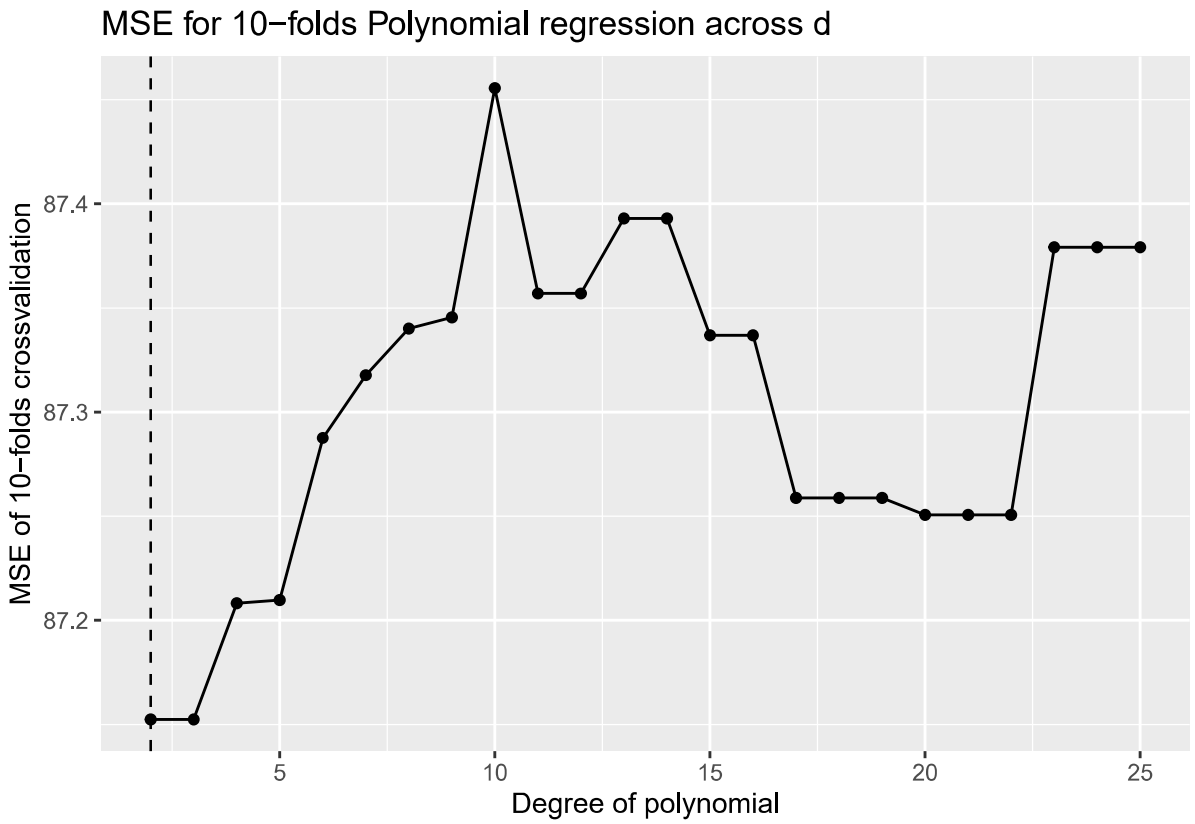
Polynomial with Lasso (not for final answer)

```
library(polywog)
```

```
## Loading required package: miscTools
```

```
gss_train0<-gss_train
#gss_train0$ones<-rep(0,nrow(gss_train0))

eg_income_poly_cv<-cv.polywog(egalit_scale ~ income06, data=gss_train, degrees.cv=2:25, nfolds=10)
eg_income_poly_mse0<-eg_income_poly_cv$results[1:24,3]

tibble(
  d=2:25,
  mse=eg_income_poly_mse0
) %>%
  ggplot(aes(d,mse))+
  geom_point()+
  geom_line()+
```

```
  geom_vline(xintercept = which.min(eg_income_poly_mse0)+1,linetype=2)+
  labs(title="MSE for 10-folds Polynomial regression across d",
       x="Degree of polynomial",
       y='MSE of 10-folds crossvalidation')
```




```
#find and save the d of smallest MSE
min_d=which.min(eg_income_poly_mse0)+1
```

Fullfill polynomial regression based on 10-fold cross validation using lm function: **Polynomial Regression** (answer for this problem)

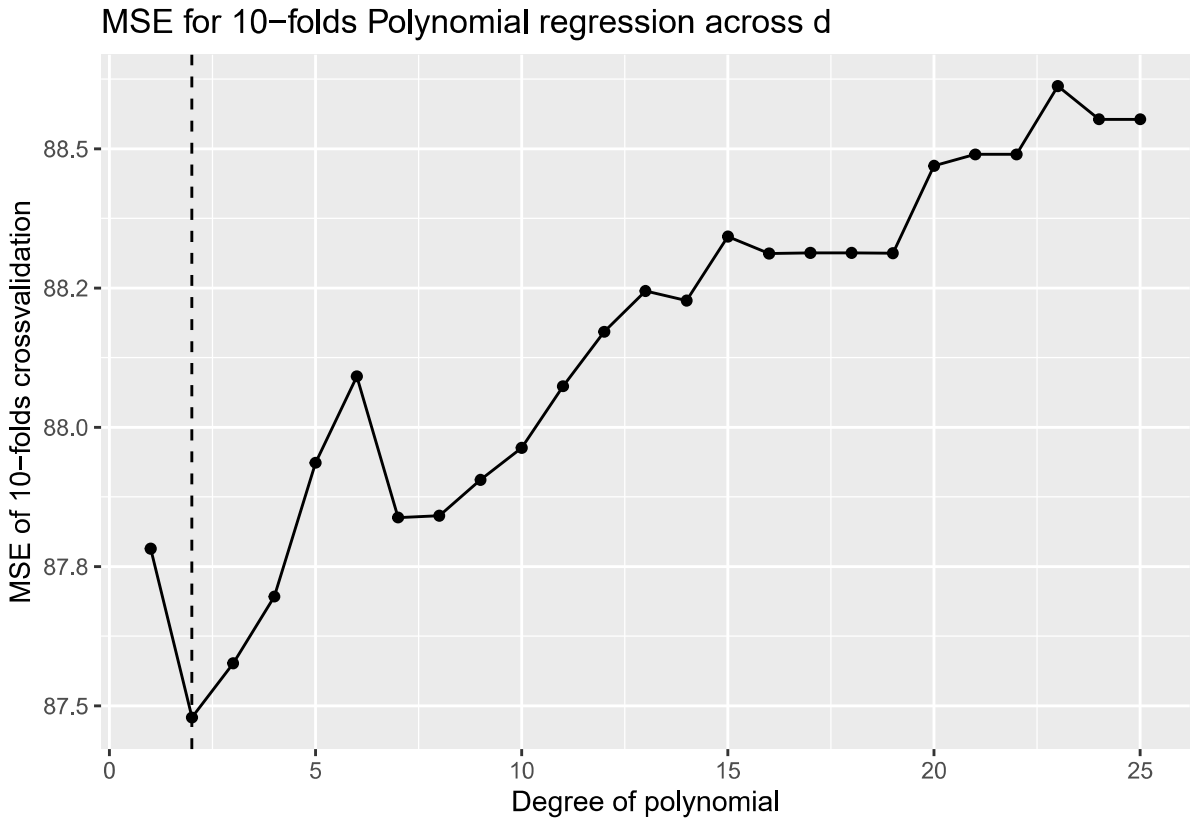
```
#build function for polynomial regression for each degree
eg_income_poly<-function(d,.data){
  #estimate model on each fold
  .data %>%
    mutate(model=map(splits,~ lm(egalit_scale ~ poly(income06, degree=d,raw=TRUE),
                data = analysis(.x))),
           truth=map(splits,~ assessment(.x)$egalit_scale),
           estimate=map2(splits,model,~ predict(.y,newdata=assessment(.x))))%>%
    unnest(truth,estimate) %>%
    group_by(id) %>%
    mse(truth=truth,estimate=estimate) %$%
    mean(.estimate)
}

#construct 10-fold dataframe
gss_cv<-vfold_cv(gss_train,v=10)
```

```r
#iterate over 10 folds and 25 degrees of freedom for poly
eg_income_poly_mse<-tibble(d=1:25)%>%
  mutate(mse=map_dbl(d,eg_income_poly,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
```

```
## 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
```

```
#plot MSE for poly regresssion across 10 folds
mse0<-eg_income_poly_mse$mse
ggplot(eg_income_poly_mse,aes(d,mse))+
  geom_point()+
  geom_line()+
  geom_vline(xintercept = which.min(mse0),linetype=2)+
  labs(title="MSE for 10-folds Polynomial regression across d",
       x="Degree of polynomial",
       y='MSE of 10-folds crossvalidation')
```

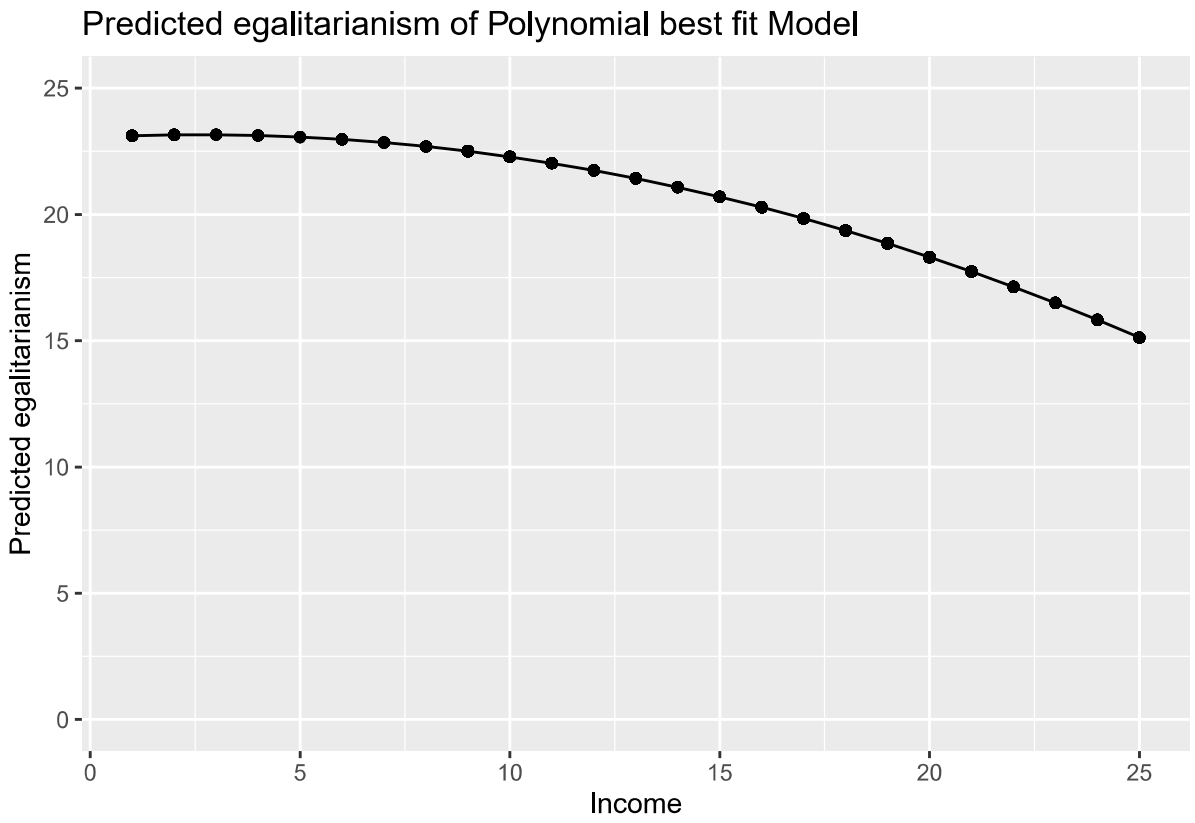



```
#find and save the degree of smallest MSE
min_d=which.min(mse0)
```

As we can see from the graph above, the degree of 2 for polynomial regression yields the smallest MSE (which also corresponds with the lasso form polynomial regression generated above).

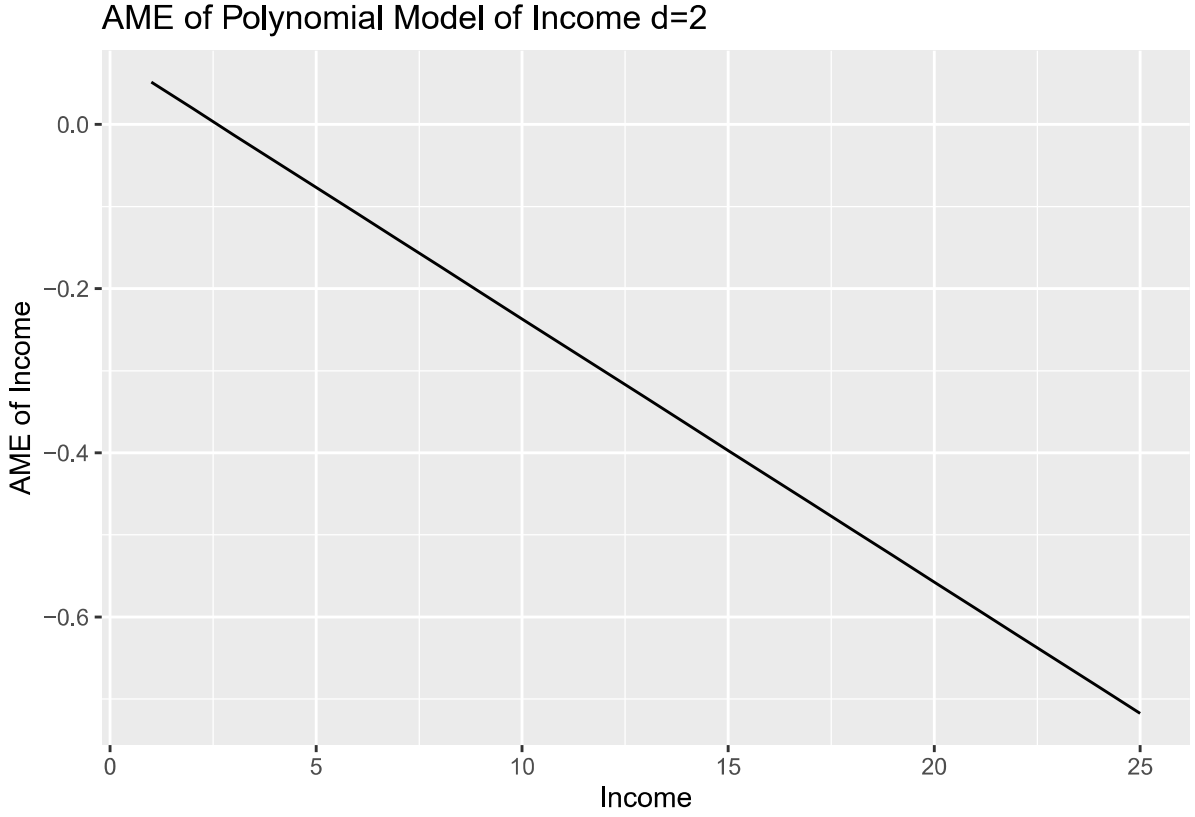
For degree of 2-the best fit polynomial regression, plot the resulting fit as below:

```
#build the power polynomial model, d=2
eg_income_poly2 <- lm(egalit_scale ~ income06 + I(income06^2),data = gss_train)

#predict egalit_scale
eg_income_ploy2_preds <- eg_income_poly2 %>%
  predict(gss_train)

#visualzation
tibble(
  income=gss_train$income06,
  eg_preds=eg_income_ploy2_preds
) %>%
  ggplot(aes(income,eg_preds))+
  geom_point()+
  geom_line()+
  ylim(0,25)+
  labs(title = "Predicted egalitarianism of Polynomial best fit Model",
       x = "Income",
       y = "Predicted egalitarianism")
```



Then, plot the averaged marginal effect (AME) of income on egalitarianism, based on this best fit polynomial model d=2:

```
cplot(eg_income_poly2, "income06", what = "effect", draw = FALSE, data=gss_train) %>%
  ggplot(aes(x = xvals)) +
  geom_line(aes(y = yvals)) +
  labs(title = "AME of Polynomial Model of Income d=2",
       x = "Income",
       y = "AME of Income")
```



Taken the above two graphs (polynomial fit and AME) all together, we could see that income has an non-linear effect on egalitarianism; with the increase of income, egalitarianism decreases (negative AME) and especially with higher income, predicted egalitarianism decreases faster, shown as a steeper line at the right end of the polynomial fit and a higher absolute value of AME with income increasing.

2. Fit a *step function* to predict egalit_scale and choose optimal number of cuts by 10-fold cross validation.

There are two versions of code for step function cross validation: here is the first one; it yields robust resutls of cut of num=4 which has the smallest MSE, while the latex does not pass this part of code (it works well in my RStudio, which is really weird)

Code which could not pass latex, but can work in RStudio -function for step function regression eg_income_step<-function(cut_num,.data){ #estimate model on each fold .data %>% mutate(model=map(splits,~ lm(egalit_scale ~ cut_interval(income06, n=cut_num,raw=TRUE), data = analysis(.x))), truth=map(splits,~ assessment(.x)$egalit_scale$), $estimate = map2(splits, model, predict(.y, newdata = assessment(.x))))$% mean(.estimate) }

-construct 10-fold dataframe gss_cv<-vfold_cv(gss_train,v=10)

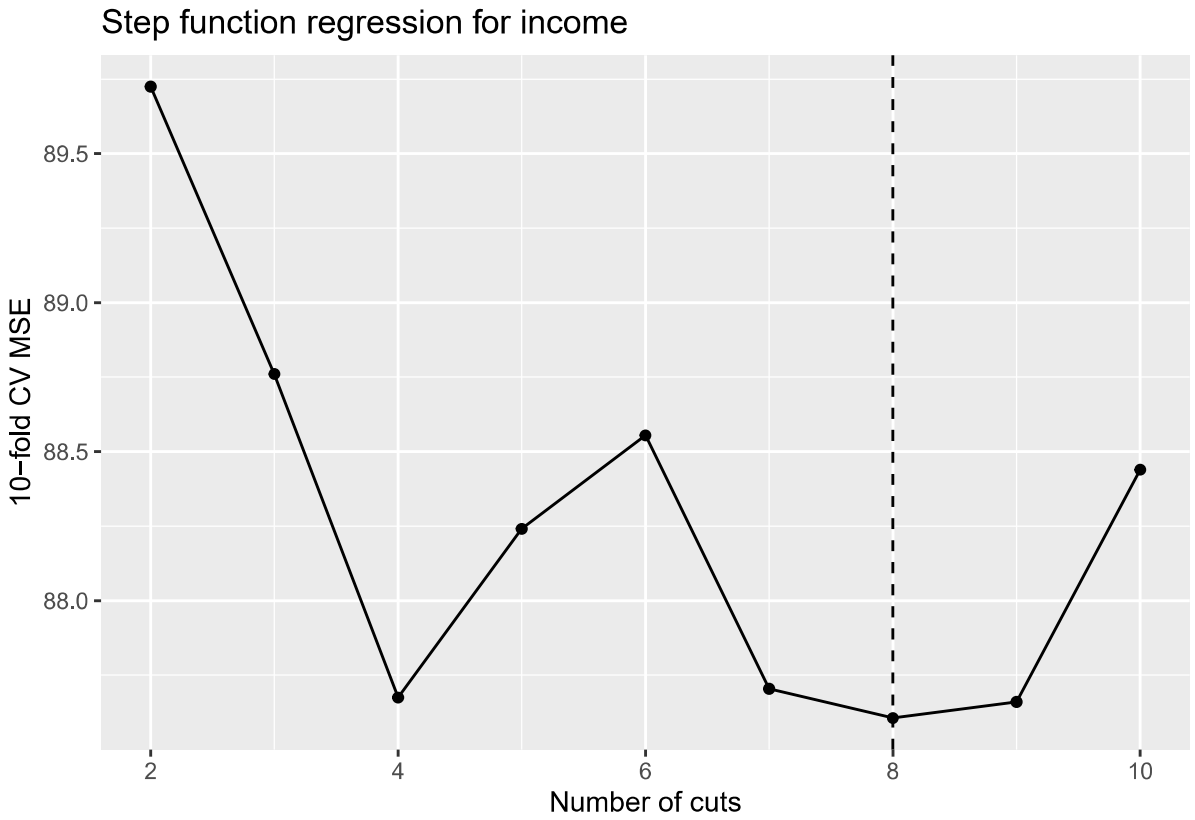
-iterate over 10 folds and 10 cuts for step function regression eg_income_step_mse<-tibble(cut_num=2:10)%>% mutate(mse=map_dbl(cut_num,eg_income_step,gss_cv))

-plot MSE for step function regresssion across 10 folds mse0=eg_income_step_mse$mse ggplot(eg_income_step_mse,aes(cut_n geom_point()+ geom_line()+ geom_vline(xintercept=which.min(mse0)+1,linetype=2)+ labs(title="MSE for 10-folds Step Function regression across 10 cuts", x="Number of Cuts", y="MSE of 10-folds cross-validation")

-Find and save the cut has smallest MSE min_cut=which.min(mse0)+1

Another version of code, while result is relative unstable for each run Then I tried another code for step function cross validation, this one latex could pass while the result is not very stable across each run (sometimes yields 9 as the best num of cut, maybe it is related with the resampling process, and I also put this version of code here)

```
eg_step_mse=vector(mode="numeric",length=9)

for (num_cut in 2:10) {
  gss_train$income06_new <- cut_interval(gss_train$income06, num_cut)
  eg_step_glm.fit <- glm(egalit_scale ~ income06_new, data = gss_train)
  eg_step_mse[num_cut - 1] <- boot::cv.glm(gss_train, eg_step_glm.fit, K = 10)$delta[1]
}

tibble(
  num_cut = 2:10,
  mse = eg_step_mse
  ) %>%
  ggplot(aes(num_cut, mse)) +
  geom_point()+
  geom_line() +
  geom_vline(xintercept = which.min(eg_step_mse) + 1, linetype = 2) +
  labs(title = "Step function regression for income",
       x = "Number of cuts",
       y = "10-fold CV MSE")
```

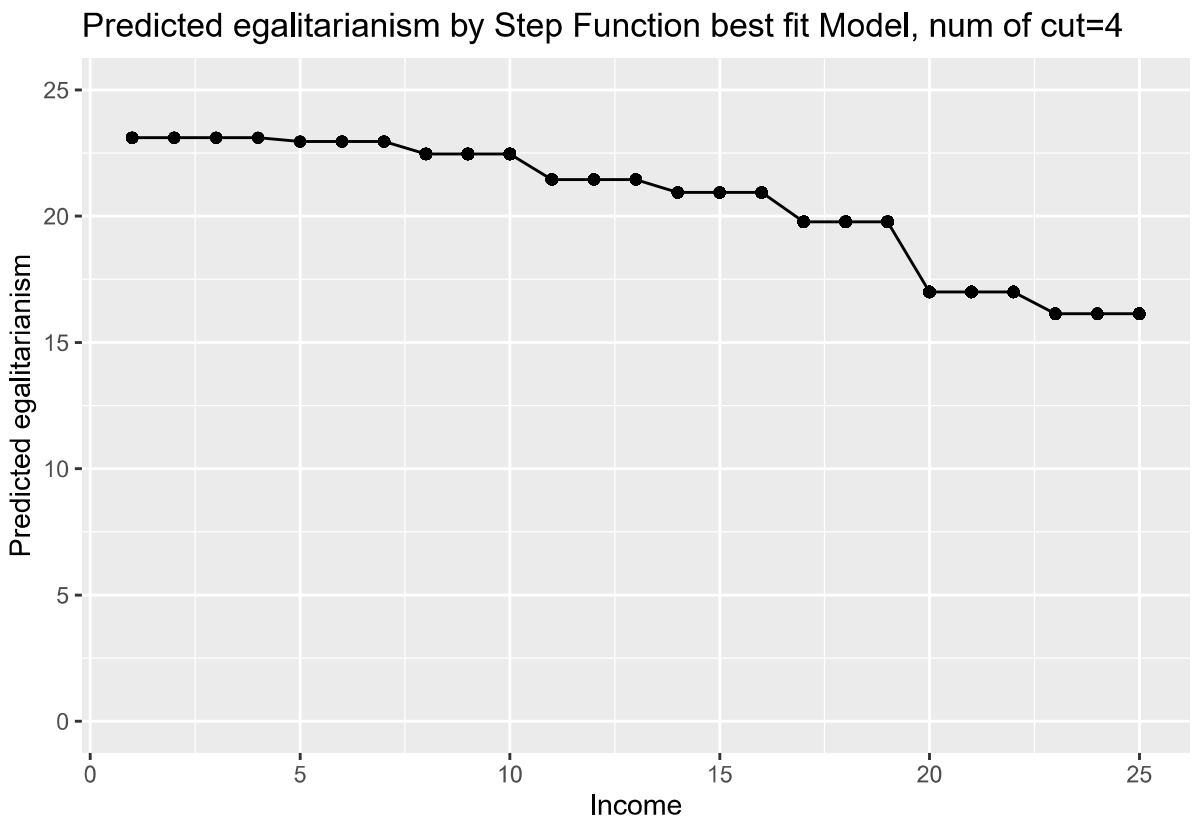



```
#Find and save the cut has smallest MSE
min_cut=which.min(eg_step_mse)+1
```

```r
#remove income06_new
gss_train <- select(gss_train, -income06_new)
```

As we could see from the above graph, when there are 4 cuts, the step function regression has the best performance-smallest MSE. Accordingly, I modeled the best step function with a cut number of 4, visualized as below:

```r
#build the best fit step function model, Num_cut=4
eg_income_step4 <- lm(egalit_scale ~ cut_interval(income06, min_cut),data = gss_train)

#Then for visualization
eg_income_step4%>%
  prediction %>%
  ggplot(aes(x = income06)) +
  geom_line(aes(y = fitted)) +
  geom_point(aes(x = income06,y = fitted))+
  ylim(0,25)+
  labs(title = "Predicted egalitarianism by Step Function best fit Model, num of cut=4",
       x = "Income",
       y = "Predicted egalitarianism")
```



As we could see from the figure, the relationship between income and predicted egalitarianism is not identical across all income groups, not a uniform linear relationship; when income increases, the predicted egalitarianism decreases more quickly, which corresponds well with the results by best fit polynomial regression.

With the best fit step function regression, we could more clearly figure out that there are *about 4 different income groups* in predicting egalitariansim. The last group-high income group is most distinctive, which has

most different attitudes towards egalitarianism relating to their income level-*higher income with much lower* egalitarianism attitudes. The influence of income on egalitarianism is more significant for high income people, which is reasonable considering that the advantaged ones hope to further maintain their benefits.

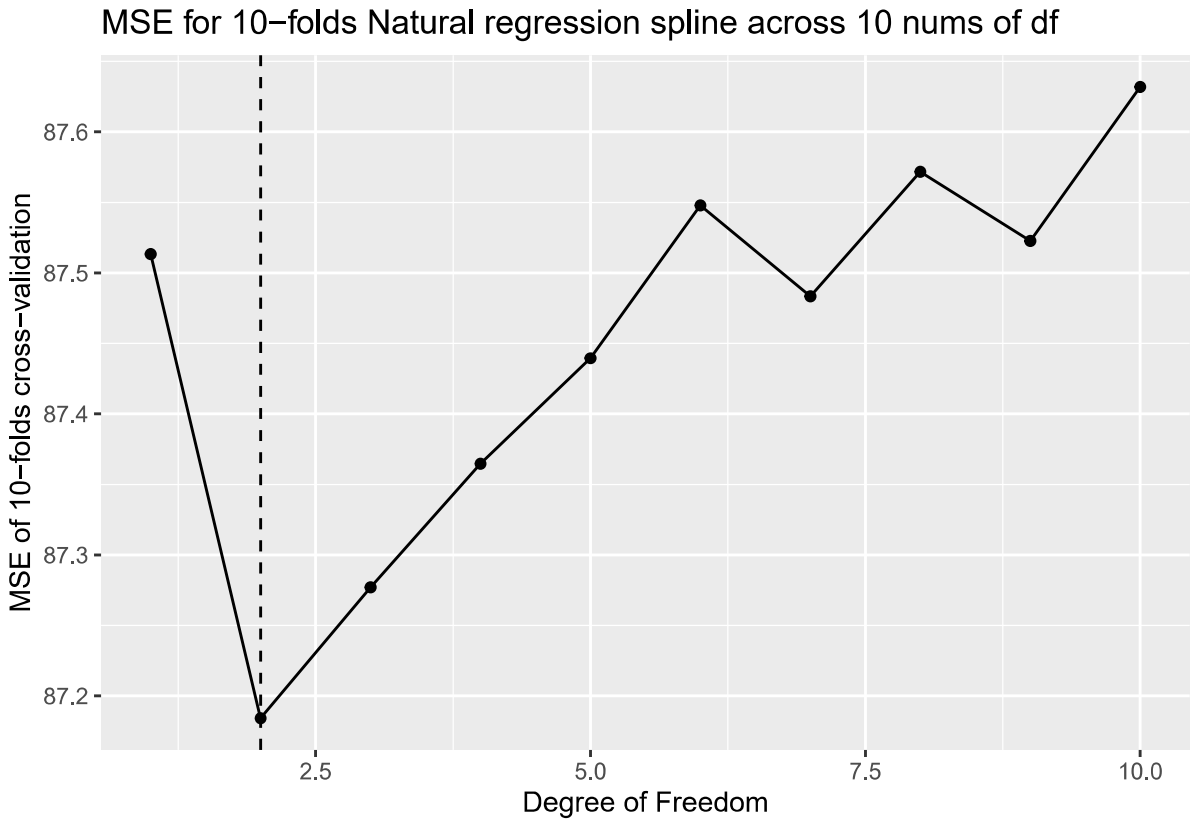
3. Fit a *natural regression spline* to predict egalit_scale by income and select optimal number of degrees of freedom by 10-fold cross validation.

```
#function for natural regression spline of each df
eg_income_ns<-function(df,.data){
  #estimate model on each fold
  .data %>%
    mutate(model=map(splits,~ glm(egalit_scale ~ ns(income06, df),data = analysis(.x))),
           truth=map(splits,~ assessment(.x)$egalit_scale),
           estimate=map2(splits,model,~ predict(.y,newdata=assessment(.x))))%>%
    unnest(truth,estimate) %>%
    group_by(id) %>%
    mse(truth=truth,estimate=estimate) %$%
    mean(.estimate)
}

#construct 10-fold dataframe
gss_cv<-vfold_cv(gss_train,v=10)

#iterate over 10 folds and 1-10 degree of freedom for natural regression spline
eg_income_ns_mse<-tibble(df=1:10) %>%
  mutate(mse=map_dbl(df,eg_income_ns,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
```

```
#plot MSE for natural regresssion spline across 10 folds
mse0=eg_income_ns_mse$mse
ggplot(eg_income_ns_mse,aes(df,mse))+
  geom_point()+
  geom_line()+
  geom_vline(xintercept=which.min(mse0),linetype=2)+
  labs(title="MSE for 10-folds Natural regression spline across 10 nums of df",
       x="Degree of Freedom",
       y="MSE of 10-folds cross-validation")
```



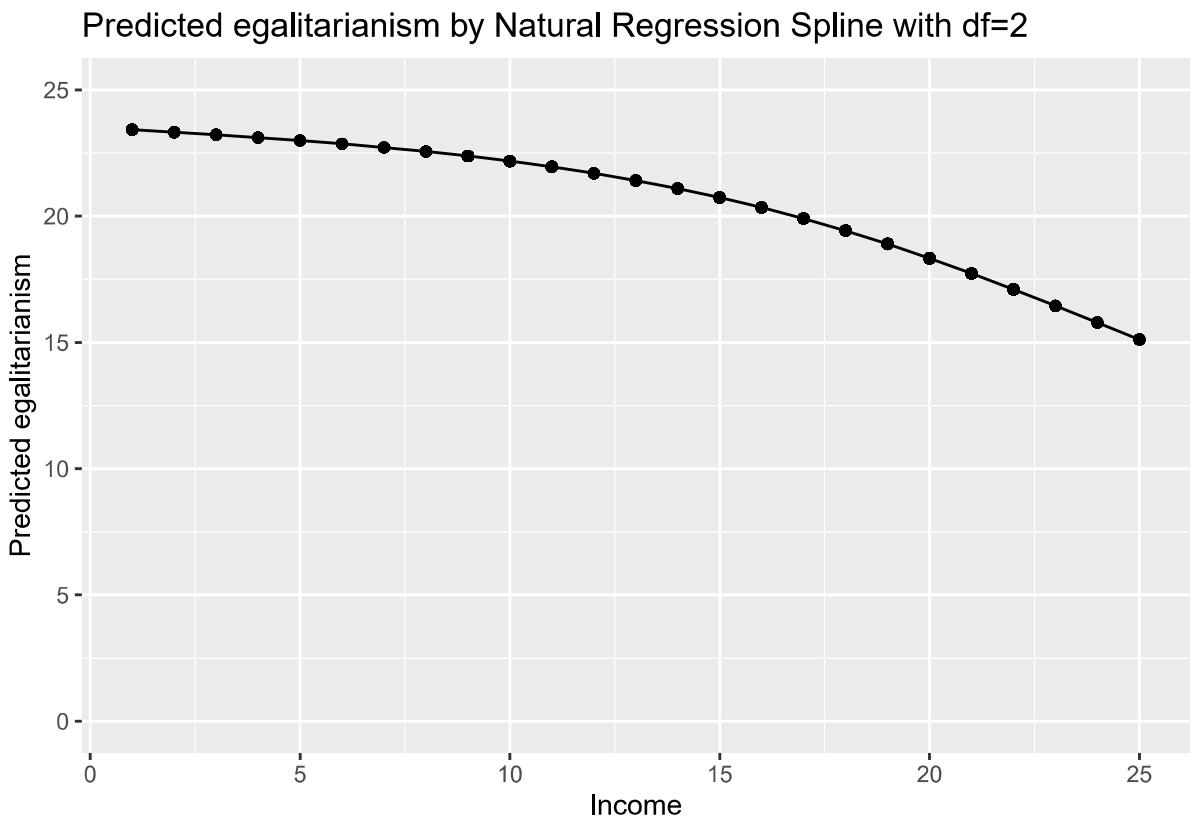

```
#find and save the df has smallest MSE
min_df=which.min(mse0)
```

The optimal number of degree of freedom is 2 for Natural regression Spline, which probably means d=2 for basis power function with no knot (taking the whole dataset) and is totally in line with the best fit polynomial regression result.

Now to visualize the best model: should be identical with the figure of polynomial regression d=2

```
#build the best fit natural regression spline, Num_df=2
eg_income_ns2 <- lm(egalit_scale ~ ns(income06, min_df),data = gss_train)

#Then for visualization
eg_income_ns2%>%
  prediction %>%
  ggplot(aes(x = income06)) +
```

```
  geom_line(aes(y = fitted)) +
  geom_point(aes(x = income06,y = fitted))+
  ylim(0,25)+
  labs(title = "Predicted egalitarianism by Natural Regression Spline with df=2",
       x = "Income",
       y = "Predicted egalitarianism")
```

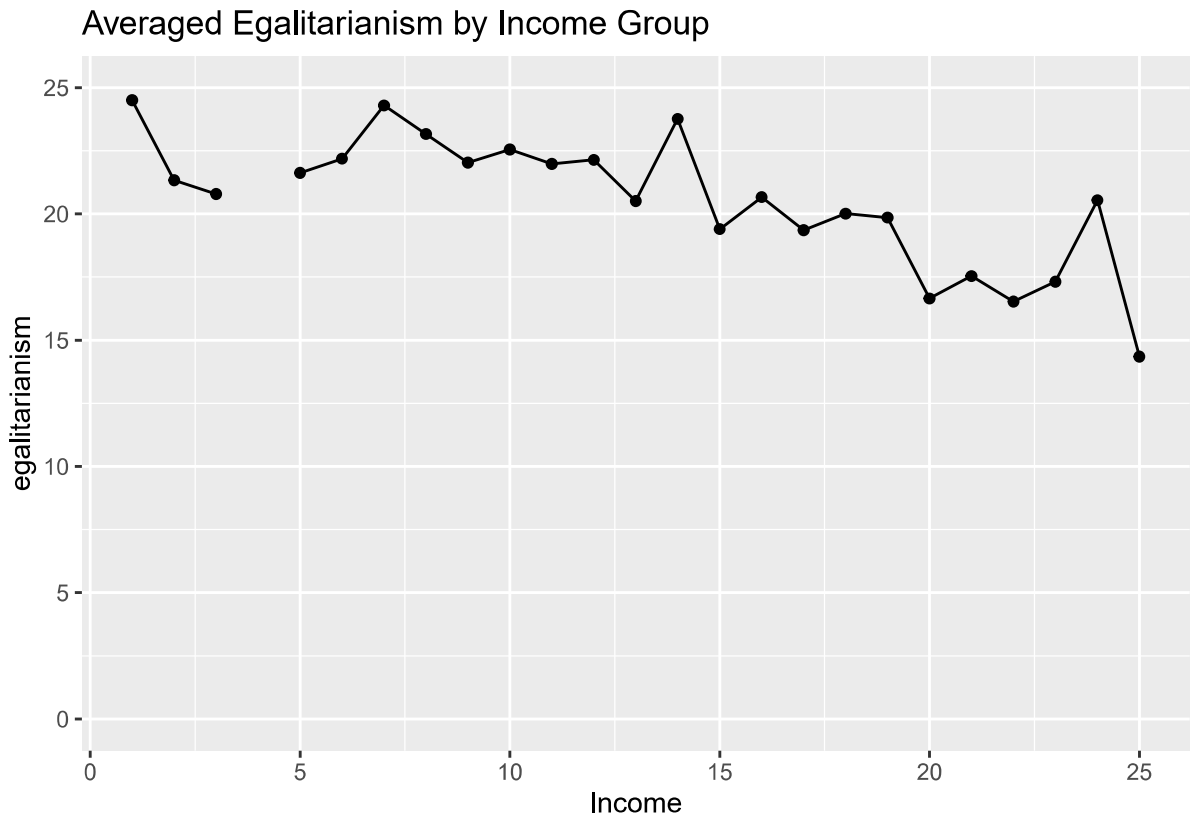


Finally, let's compare the best fit model results with the 'raw/true' relationship between income and egalitarianism: the averaged true egalitarianism score across different income groups.

As shown below, the relationship between income and egalitarianism is almost linear, while with a bit steeper at the right end of high income. This trends corresponds with the result of polynomial/natural regression spline best fit model, which capture the non-linear trend in prediciton; while for step function, the 4 cuts result is not that meaningful. According to the raw data visualization, relationship between income and egalitarianism is very continous with less difference among different income groups and the 4 number cut is less informative.

```
#averaged level of egalitarianism by income in Training dataset:
aggregate(gss_train$egalit_scale, by=list(gss_train$income06), FUN=mean)%>%
  ggplot(aes(x = Group.1))+ #for visualization
  geom_line(aes(y=x))+
  geom_point(aes(x = Group.1,y = x))+
  ylim(0,25)+
  labs(title = "Averaged Egalitarianism by Income Group",
       x = "Income",
       y = "egalitarianism")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Egalitarianism and everything

4. Estimate models based on all available predictors and optimize parameters based on 10-fold cross-validation (MSE)

a. Linear Regression

First of all construct a model for cross validation, based on that save the RMSE results across samples and develop a predictor function for visualization later.

```
#construct a linear regression model
eg_lg<-train(
  egalit_scale ~ .,
  data=gss_train,
  method='lm',
  metric='RMSE',
  trControl=trainControl(method="CV",number=10),
  preProcess=c("zv")
)

#save 10-fold cross validation results for further visualization
eg_lg_mse<-eg_lg$resample$RMSE
print("RMSE across 10 validation:")
```

```
## [1] "RMSE across 10 validation:"
```

```r
print(eg_lg_mse)
```

```
##  [1] 7.68 8.06 8.52 9.20 7.57 7.04 7.55 7.80 7.02 9.04
```

```r
#define a prediction function
predictor_eg_lg<-Predictor$new(
 model=eg_lg,
 data=select(gss_train,-egalit_scale),
 y=gss_train$egalit_scale
)

#print the traning performance of linear regression
print(eg_lg)
```

```
## Linear Regression
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1333, 1333, 1334, 1332, 1333, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   7.95  0.331     6.28
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

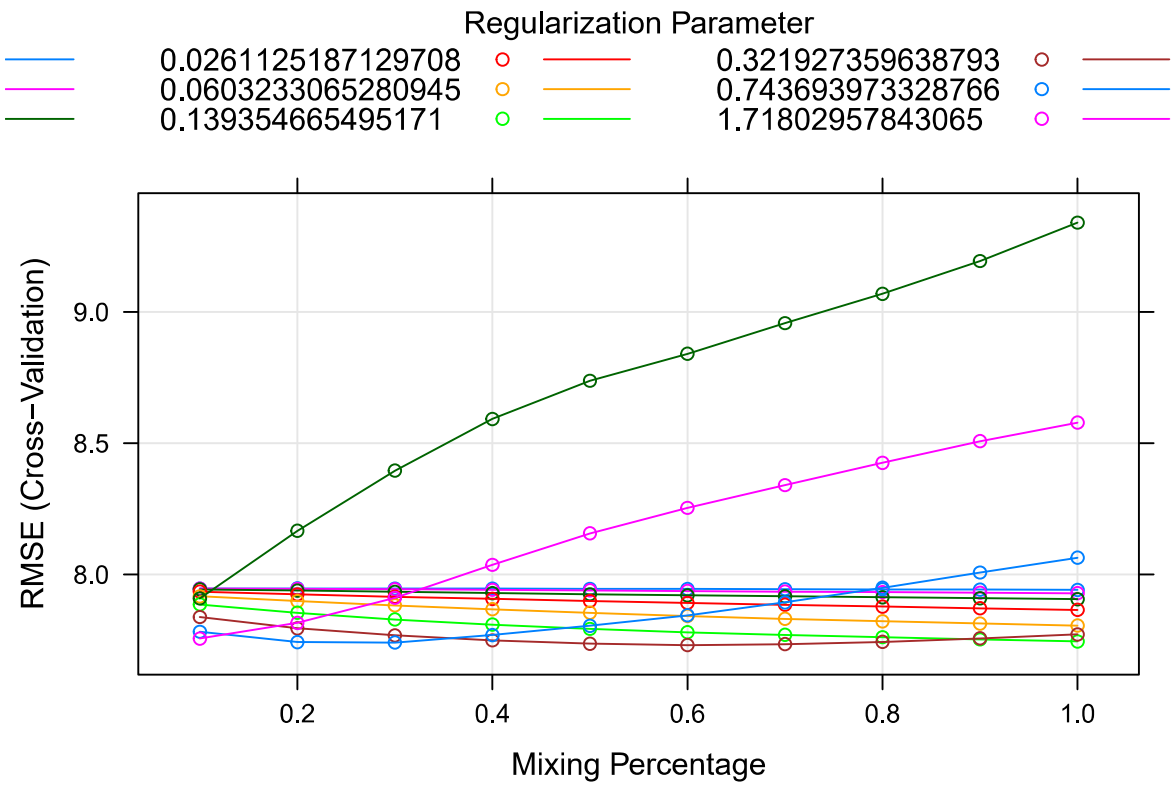
b. Elastic Net Regression

```r
#construct an elastic net regression model
eg_en<-train(
  egalit_scale ~ .,
  data=gss_train,
  method='glmnet',
  metric='RMSE',
  trControl=trainControl(method="CV",number=10),
  preProcess=c("zv","center","scale"),
  tuneLength=10
)

#save cross validation results for further visualization
eg_en_mse<-eg_en$resample$RMSE
eg_en_mse
```

```
##  [1] 8.27 8.16 6.74 7.99 7.77 7.68 7.63 7.12 8.33 7.61
```

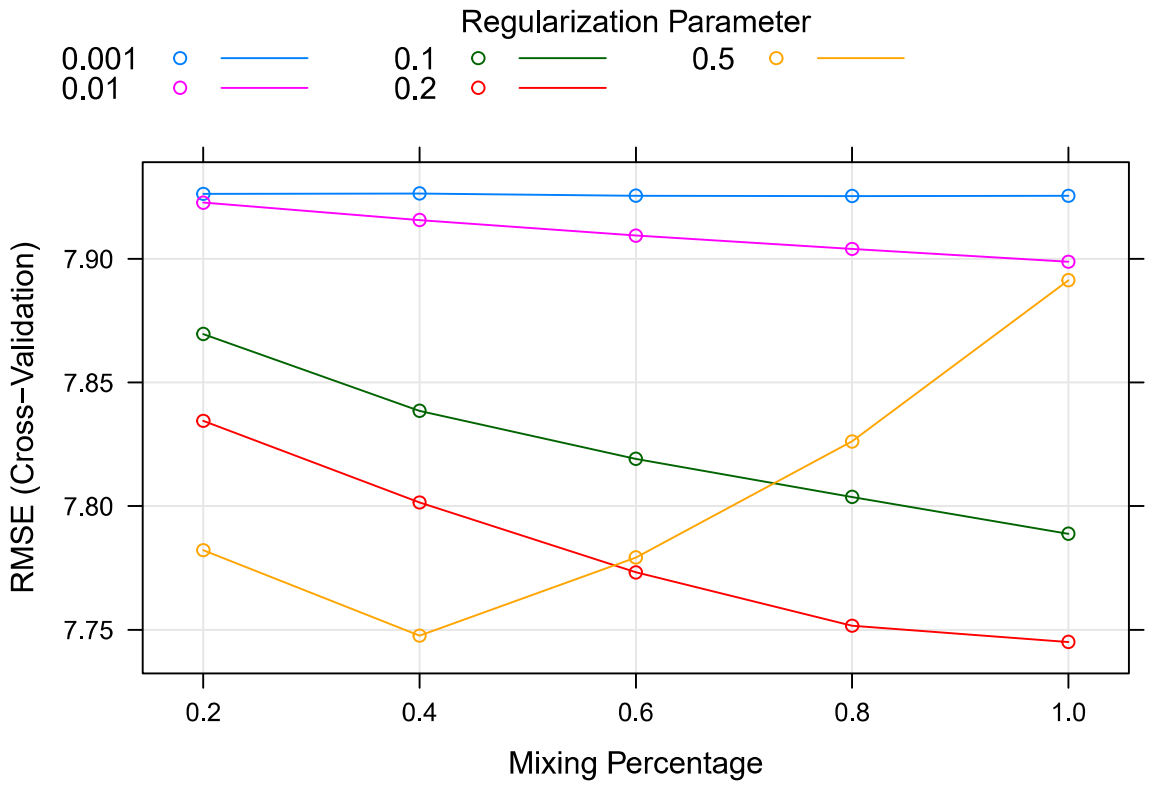
```r
#define a prediction function
predictor_eg_en<-Predictor$new(
 model=eg_en,
 data=select(gss_train,-egalit_scale),
 y=gss_train$egalit_scale
)

#plot the training performance of lambda and alpha by 10-cross validation
```

```
plot(eg_en)
```



For Elastic Net Regression, we could see above that for alpha around 0.00, the model seems to have better performance. Based on this observation, I further trained the Elastic Net Regression on the fixed set of parameters of lambda and alpha to find the optimal parameters:

```
#construct an elastic net regression model
eg_en2<-train(
  egalit_scale ~ .,
  data=gss_train,
  method='glmnet',
  metric='RMSE',
  trControl=trainControl(method="CV",number=10),
  preProcess=c("zv","center","scale"),
  tuneGrid=expand.grid(alpha=c(0.2,0.4,0.6,0.8,1.0),lambda=c(0.001,0.01,0.1,0.2,0.5))
  #training on fixed set of parameters found by visualization
)

#plot the training performance of lambda and alpha by 10-cross validation
plot(eg_en2)
```

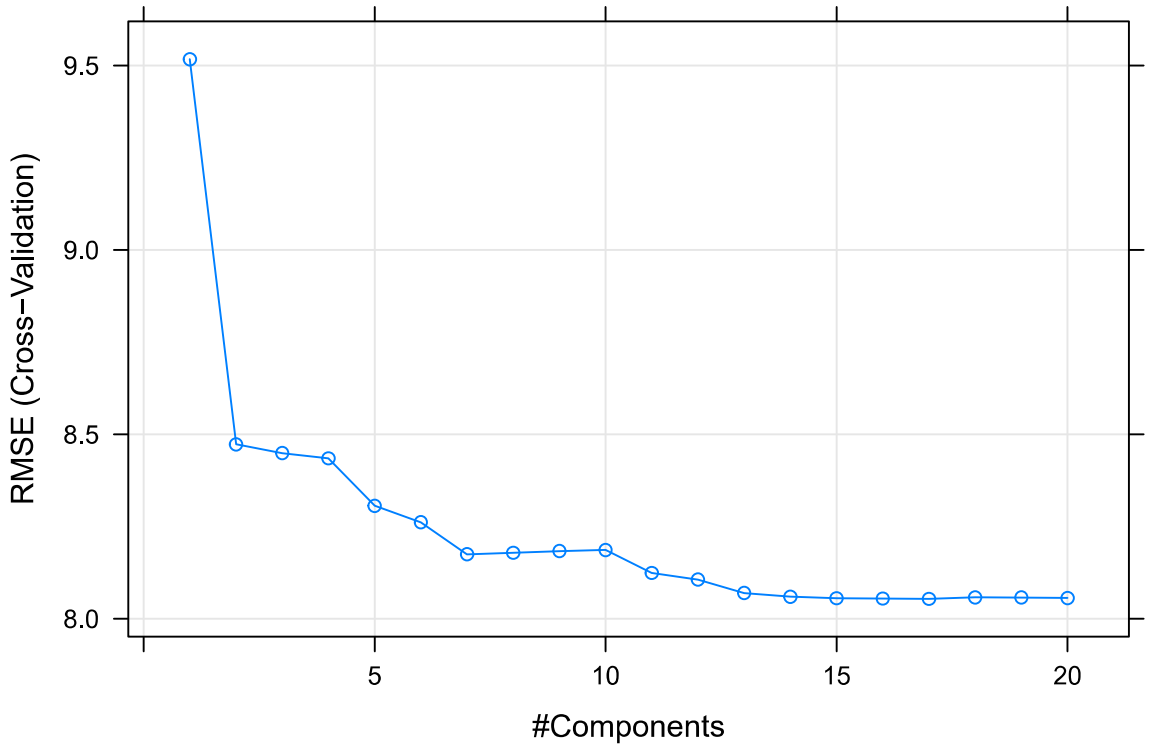


As shown above, indeed for lambda=1, alpha~0.2, the Elastic Net model has the best performance (smallest RMSE across validation) around 7.75, smaller than linear regression (7.96)

c. Principal Component Regression

```
#construct a principal component regression model
eg_pcr<-train(
  egalit_scale ~ .,
  data=gss_train,
  method='pcr',
  metric='RMSE',
  trControl=trainControl(method="CV",number=10),
  preProcess=c("zv","center","scale"),
  tuneLength=20
)

#save cross validation results for further visualization
eg_pcr_mse<-eg_pcr$resample$RMSE

#define a prediction function
predictor_eg_pcr<-Predictor$new(
 model=eg_pcr,
 data=select(gss_train,-egalit_scale),
 y=gss_train$egalit_scale
)

#plot the training performance of ncomp by 10-cross validation
```

```
plot(eg_pcr)
```

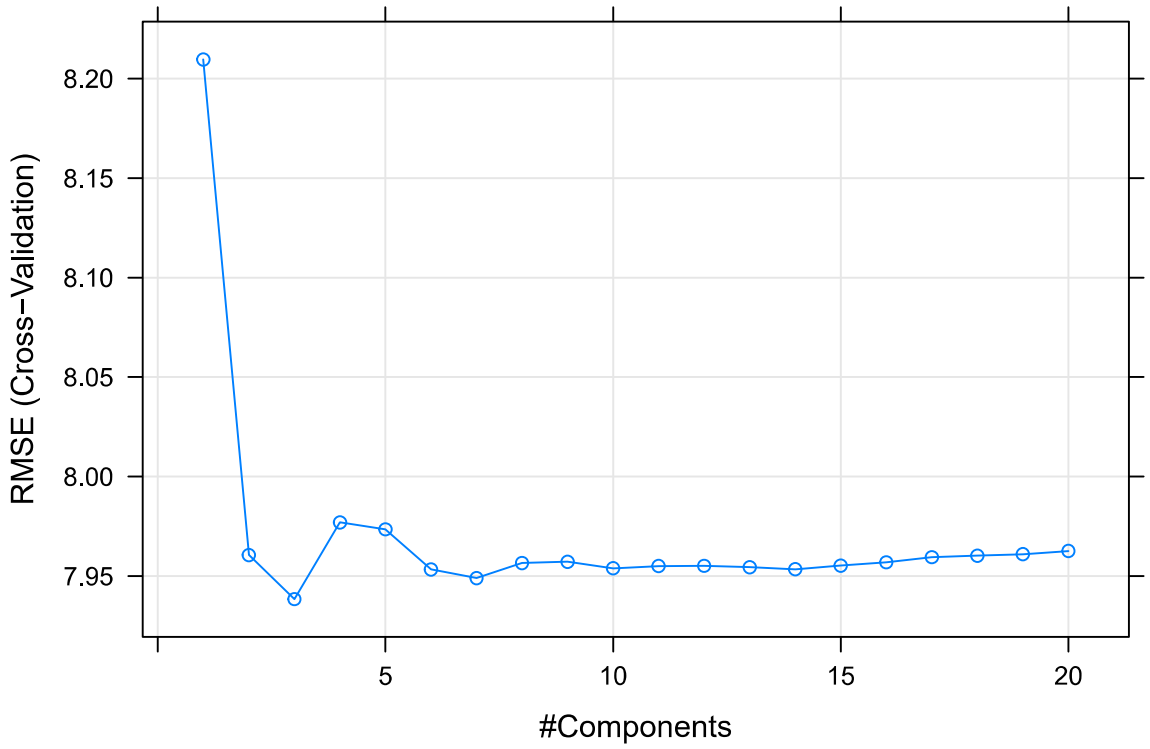


As we could see, the smallest MSE for PCR is around 8.0, with a component of around 15 (while with around 5 components, the information of the dataset is captured to a great extent), which is larger than Linear Regresion and Elastic Net Regression. PCR generallly yields worse performance compared to the previous two models (7.75/7.96).

d. Partial least squares Regression

```
#construct a partial least squares regression model
eg_pls<-train(
  egalit_scale ~ .,
  data=gss_train,
  method='pls',
  metric='RMSE',
  trControl=trainControl(method="CV",number=10),
  preProcess=c("zv","center","scale"),
  tuneLength=20
)

#save cross validation results for further visualization
eg_pls_mse<-eg_pls$resample$RMSE

#define a prediction function
predictor_eg_pls<-Predictor$new(
 model=eg_pls,
 data=select(gss_train,-egalit_scale),
```

```
 y=gss_train$egalit_scale
)

#plot the training performance of ncomp by 10-cross validation
plot(eg_pls)
```

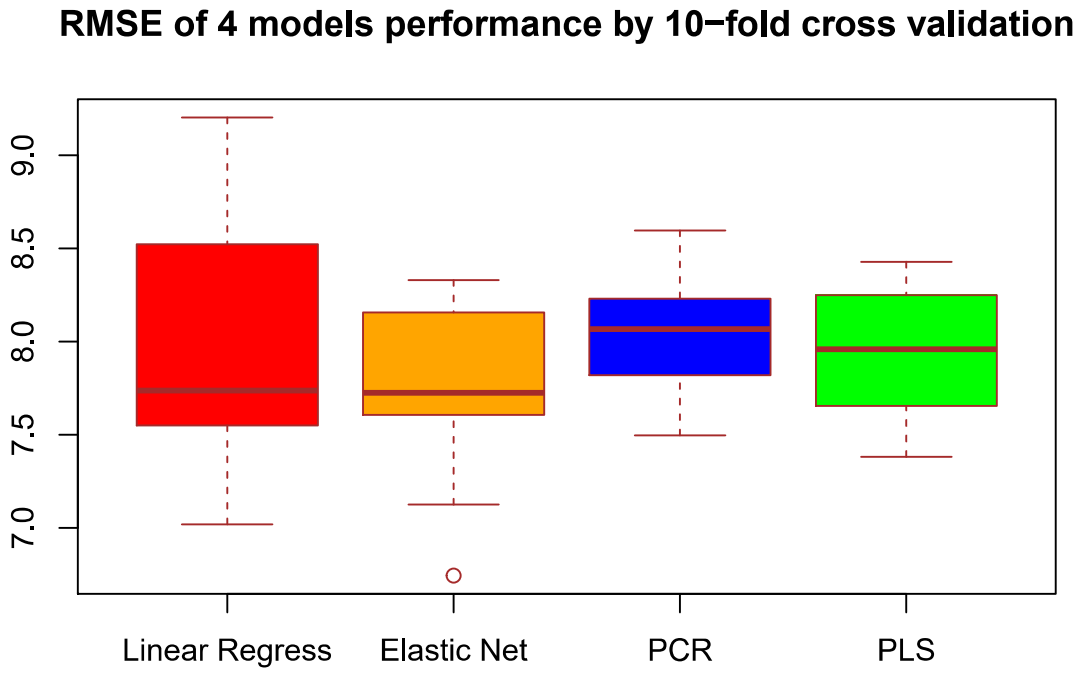


For PLS, the smallest RMSE is around 7.90~7.95 (with a component of about 7, and with 3 components the informaiton between Ega and the other variables is largely captured), generally worse than Elastic Net Regression, better than PCR, close to Linear Regression, while this comparison is just based on mean RMSE for 10-fold cross validation on the best fit model.

For better comparison, I drew a boxplot to show the distribution of the RMSE across 10-fold validations for the four models.

As shown below, Elastic Net Regression has relatively smaller RMSE, performs best, then comes PCR and PLS, finally Linear Regression. With more fined tuned model, the prediction of egalitarianism by all the other variables is more accurate.

```
#compare the training performance of 4 models by 10-fold cross validation
boxplot(eg_lg_mse,eg_en_mse,eg_pcr_mse,eg_pls_mse,
        main="RMSE of 4 models performance by 10-fold cross validation",
        at=c(1,2,3,4),
        names=c("Linear Regress","Elastic Net","PCR","PLS"),
        col=c("red","orange","blue","green"),
        border="brown")
```



Finally, I try to validate the performance of these best tuned model and test dataset and further examine their performance:

```
#compute the model perforamnce on test dataset
#for linear regression model
eg_lg_preds <- eg_lg %>%
  predict(gss_test)
eg_lg_test_rmse <- RMSE(eg_lg_preds, gss_test$egalit_scale)

#For elastic net regression
eg_en_preds <- eg_en %>%
  predict(gss_test)
eg_en_test_rmse <- RMSE(eg_en_preds, gss_test$egalit_scale)

#For PCR
eg_pcr_preds <- eg_pcr %>%
  predict(gss_test)
eg_pcr_test_rmse <- RMSE(eg_pcr_preds, gss_test$egalit_scale)

#For PLS
eg_pls_preds <- eg_pls %>%
  predict(gss_test)
eg_pls_test_rmse <- RMSE(eg_pls_preds, gss_test$egalit_scale)

#Organize in tibble
df_test_rmse<-tibble(
  Linear_Reg_RMSE=eg_lg_test_rmse,
```

```
  ElasNet_Reg_RMSE=eg_en_test_rmse,
  PCR_Reg_RMSE=eg_pcr_test_rmse,
  PLS_Reg_RMSE=eg_pls_test_rmse
)

df_test_rmse
```

```
## # A tibble: 1 x 4
##   Linear_Reg_RMSE ElasNet_Reg_RMSE PCR_Reg_RMSE PLS_Reg_RMSE
##             <dbl>            <dbl>        <dbl>        <dbl>
## 1            7.96             7.84         7.90         7.89
```

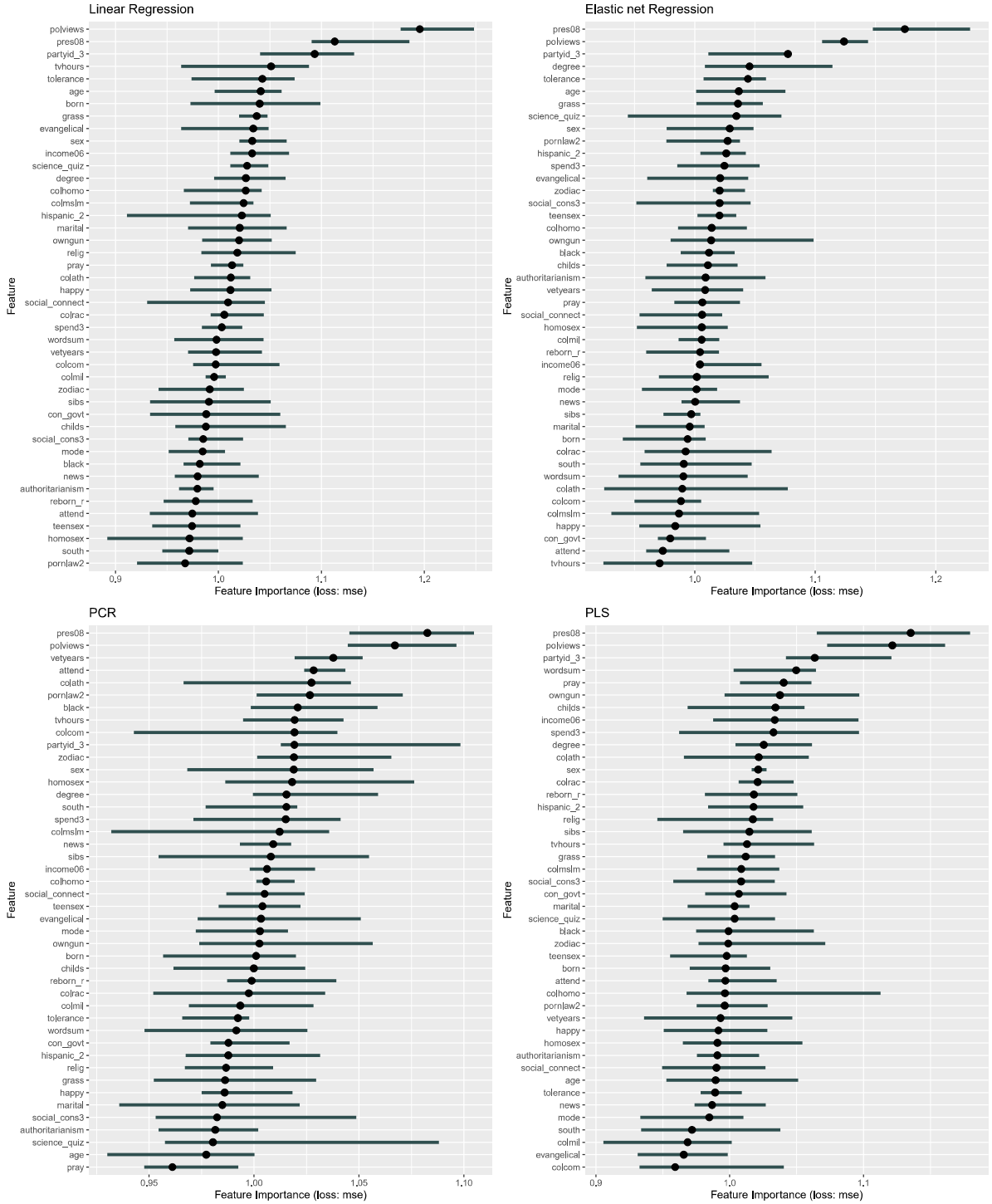
Also for the Test dataset performance, we could further validate that the performance of ElasticNet Regression is the best model among the 4, even for test dataset; then comes PCR, finally Linear Regression and PLS (which identifies significant features in a supervised way and resonably might have higher variance in perforamnce).

5. Evaluate feature importance, feature interaction, conditional effect (PDP), and compare between models, for tuned version of each model fit.

To have a deep look at how each feature/variable contribute to the prediction of egalitarianism, I computed the feature imporatnce/feature interaction for each model, and select three varaibles of greatest importance (polview, press08, partyid) for PDP visualization to futher examine their effects on egalitarianism.

a. feature importance

```
#call packages to calculate feature importance
fimp_lg <- FeatureImp$new(predictor_eg_lg, loss = "mse")
fimp_en <- FeatureImp$new(predictor_eg_en, loss = "mse")
fimp_pcr <- FeatureImp$new(predictor_eg_pcr, loss = "mse")
fimp_pls <- FeatureImp$new(predictor_eg_pls, loss = "mse")

#plot the results
p1 <- plot(fimp_lg) +
ggtitle("Linear Regression")
p2 <- plot(fimp_en) +
ggtitle("Elastic net Regression")

p3 <- plot(fimp_pcr) +
ggtitle("PCR")
p4 <- plot(fimp_pls) +
ggtitle("PLS")


#for aggregated visualization
fig.width=14
fig.height=17
p1 + p2 + p3 + p4
```



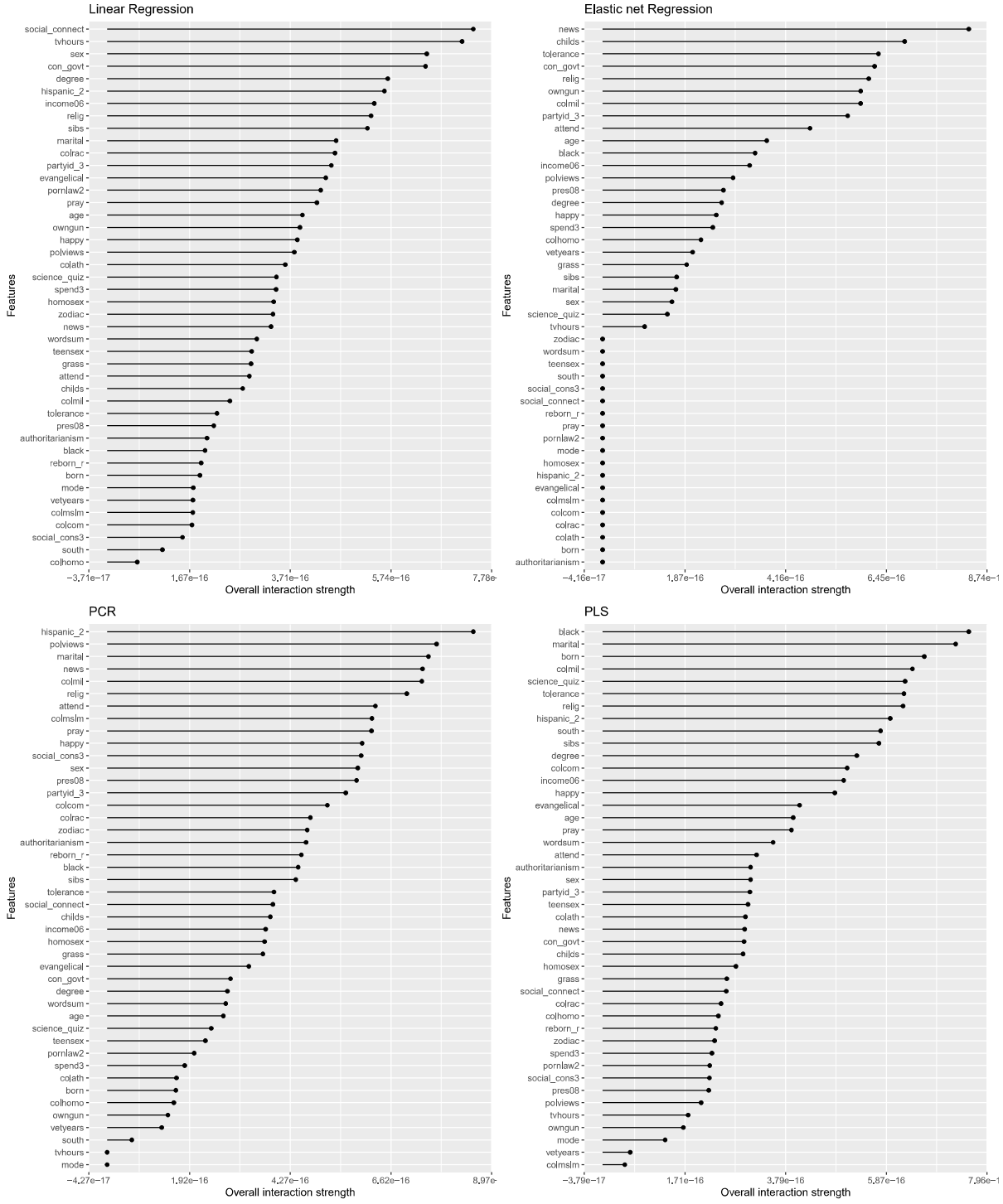
As we could see above, across 4 models, *press08* is the most distinctive and important feature in predicting egalitarianism, then comes *polviews* second important; and *partyid_3* also has a significant importance across all models. *Income06* and *grass* and *spend3* were also found as significant in linear/elastic/PLS models.

For PCR model, it yields most different significant features, probably because it is based on unsuprvised transformation of orginial features and hence assign different importance on these features after transformation; While even for this case, *press08* and *polviews* are stil robustly significant, indicating their high value in

predicting egalitarisnism.

b. feature interaction

```
#call packages to calculate pdp for variable: press08
int_lg <- Interaction$new(predictor_eg_lg)
int_en <- Interaction$new(predictor_eg_en)
int_pcr <- Interaction$new(predictor_eg_pcr)
int_pls <- Interaction$new(predictor_eg_pls)

#plot the results
p1 <- plot(int_lg) +
ggtitle("Linear Regression")
p2 <- plot(int_en) +
ggtitle("Elastic net Regression")
p3 <- plot(int_pcr) +
ggtitle("PCR")
p4 <- plot(int_pls) +
ggtitle("PLS")

#for aggregated visualization
p1 + p2 + p3 + p4
```



For Feature Interaction, the result is a little bit different across models. However, *income06*/*spend3*/*grass* generally appear to be relatively high in interaction streghth (especially for *income06* high in interaction strength across models), which are reasonably correlated with many factors related with general public and political views, social economic status and hence contributing to egalitarisnism attitudes.

For *polyview*, *press08* and *partyid_3*, results of interaction strength are not very consistent across models. In

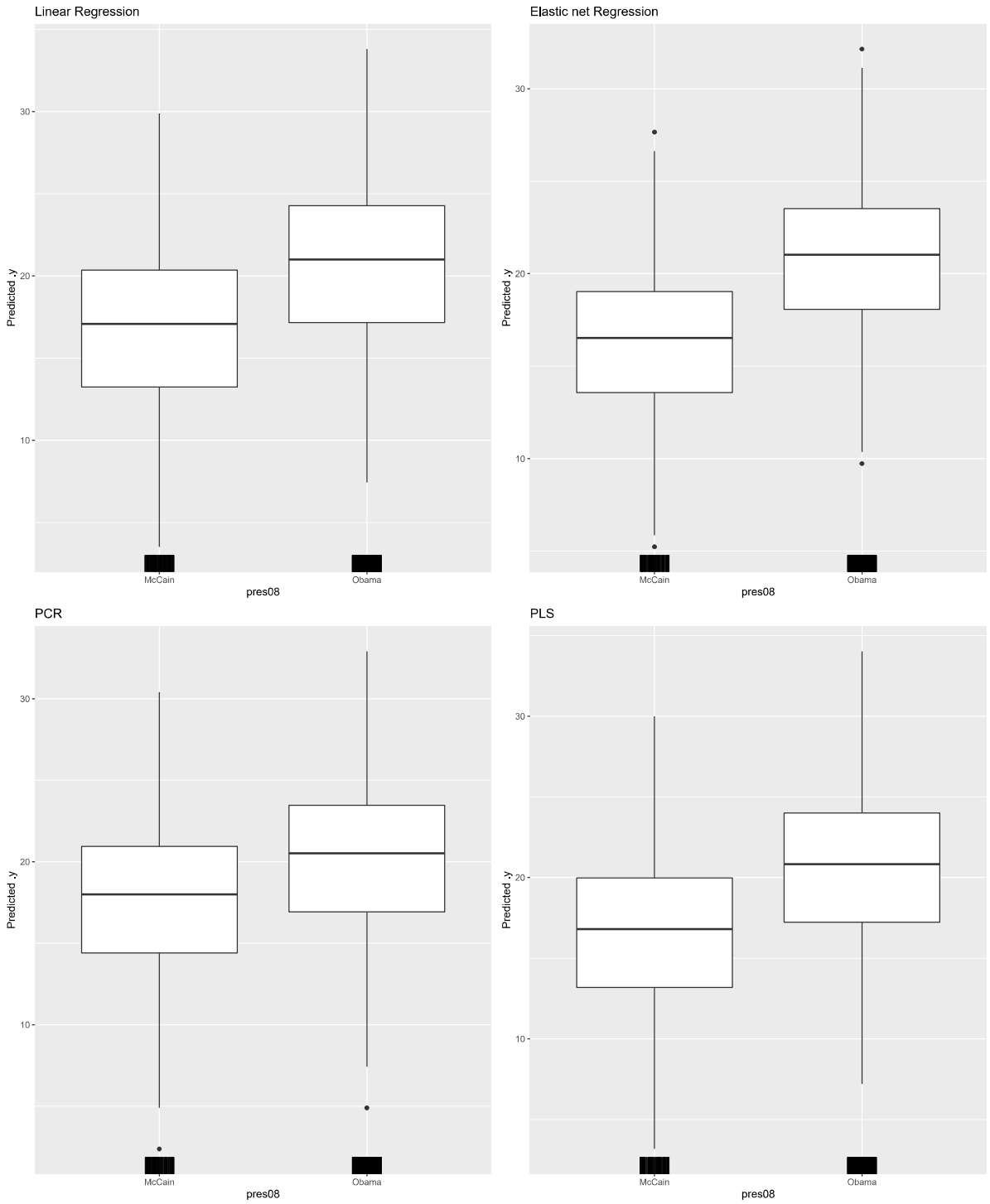
linear regression/elastic net regression model, these contributing political related model have rather low or moderate interaction strength, probably orinigating from their the correlation among these features; while for PLS and PCR, these features are relatively high in interaction strength. I guess the reason might be the supervised nature of PCR/PLS models to find the most relevant component and variables related with the response and hence the different weights assigning to them might bias the interaction strenght result.

c. PDP

Therefore, to better evaluate the perforamnce of these key contributing varaibles, we further examine their marginal/conditional effects by partial dependence plot.

First of all, plot the partial dependence plot (PDP) for *press08* variable, which is of *the highest importance* of 3 models out of 4 models:

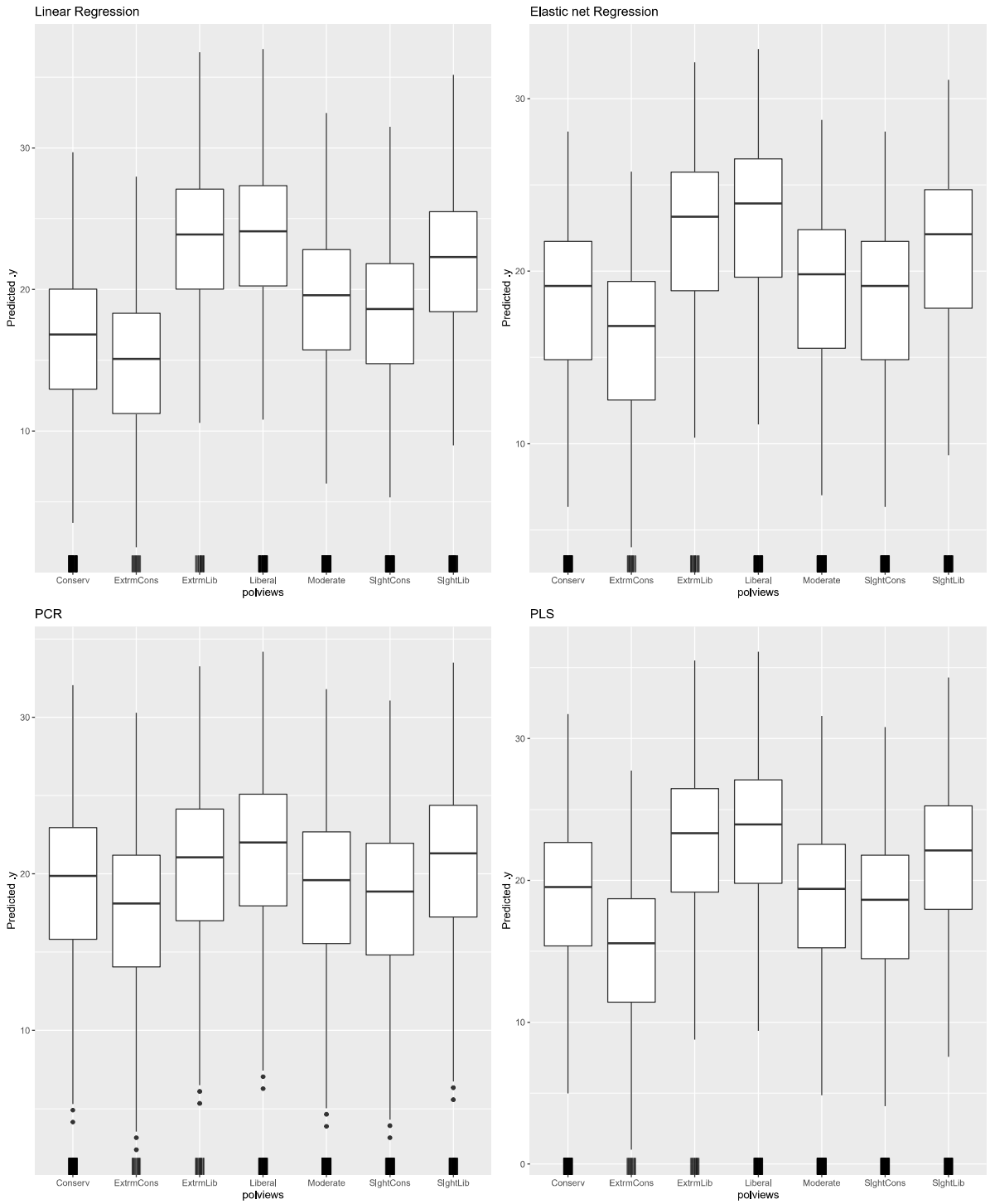
```
#call packages to calculate pdp for variable: press08
pdp_press_lg <- FeatureEffect$new(predictor_eg_lg, "pres08",method="pdp+ice")
pdp_press_en <- FeatureEffect$new(predictor_eg_en, "pres08",method="pdp+ice")
pdp_press_pcr <- FeatureEffect$new(predictor_eg_pcr, "pres08",method="pdp+ice")
pdp_press_pls <- FeatureEffect$new(predictor_eg_pls, "pres08",method="pdp+ice")

#plot the results
p1 <- plot(pdp_press_lg) +
ggtitle("Linear Regression")
p2 <- plot(pdp_press_en) +
ggtitle("Elastic net Regression")
p3 <- plot(pdp_press_pcr) +
ggtitle("PCR")
p4 <- plot(pdp_press_pls) +
ggtitle("PLS")

#for aggregated visualization
p1 + p2 + p3 + p4
```



Pres08 indicates whether people vote for 'Obama' or 'McCain'. As shown above, people who vote for 'Obama', which might indicate the people to be more liberal and bear a more positive attitude towards minority, are more inclined to have higher egalitarianism attitude. The result is robust across all 4 models.

Then, plot PDP for *polviews* variable, which is of second highest importance of all models:

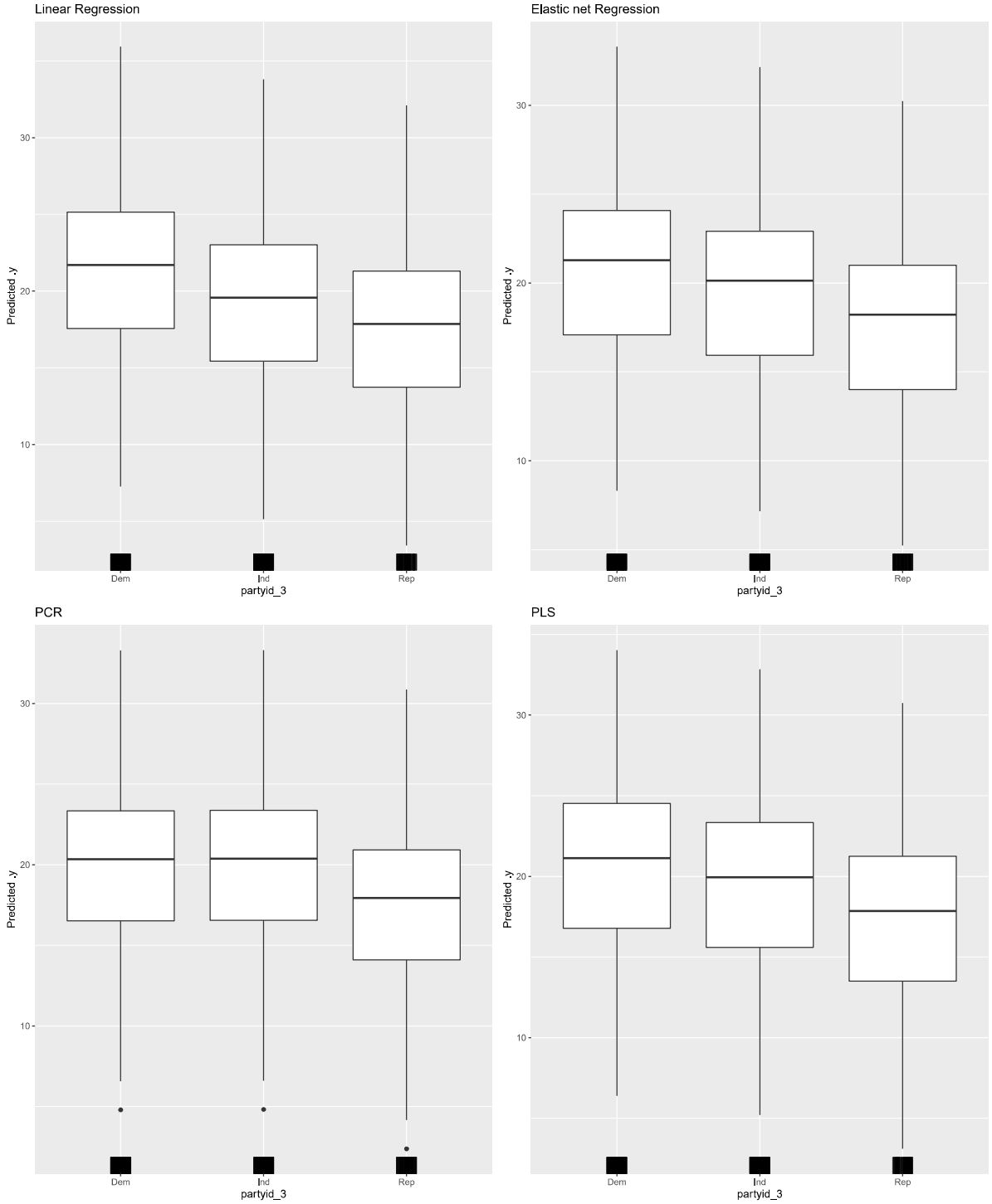
```r
#call packages to calculate pdp for variable: polviews
pdp_polvi_lg <- FeatureEffect$new(predictor_eg_lg, "polviews",method="pdp+ice")
pdp_polvi_en <- FeatureEffect$new(predictor_eg_en, "polviews",method="pdp+ice")
pdp_polvi_pcr <- FeatureEffect$new(predictor_eg_pcr, "polviews",method="pdp+ice")
pdp_polvi_pls <- FeatureEffect$new(predictor_eg_pls, "polviews",method="pdp+ice")

#plot the results
p1 <- plot(pdp_polvi_lg) +
ggtitle("Linear Regression")
p2 <- plot(pdp_polvi_en) +
ggtitle("Elastic net Regression")
p3 <- plot(pdp_polvi_pcr) +
ggtitle("PCR")
p4 <- plot(pdp_polvi_pls) +
ggtitle("PLS")

#for aggregated visualization
p1 + p2 + p3 + p4
```



Polview is a direct measure of people being 'liberal' or 'conservative'. As shwon above, people who claim to be 'liberal', even 'slight liberal', tend to be higher in egalitarianism, compared with claimed 'conservative', which is also robust across all models, indicating political viewpoints is highly correlated with attitudes towards egalitarianism.

Then, try with the *partyid_3* variable, which is also of high importance in 4 models, except for in PCR:

```r
#call packages to calculate pdp for variable: partyid_3
pdp_party_lg <- FeatureEffect$new(predictor_eg_lg, "partyid_3",method="pdp+ice")
pdp_party_en <- FeatureEffect$new(predictor_eg_en, "partyid_3",method="pdp+ice")
pdp_party_pcr <- FeatureEffect$new(predictor_eg_pcr, "partyid_3",method="pdp+ice")
pdp_party_pls <- FeatureEffect$new(predictor_eg_pls, "partyid_3",method="pdp+ice")

#plot the results
p1 <- plot(pdp_party_lg) +
ggtitle("Linear Regression")
p2 <- plot(pdp_party_en) +
ggtitle("Elastic net Regression")
p3 <- plot(pdp_party_pcr) +
ggtitle("PCR")
p4 <- plot(pdp_party_pls) +
ggtitle("PLS")

#for aggregated visualization
p1 + p2 + p3 + p4
```



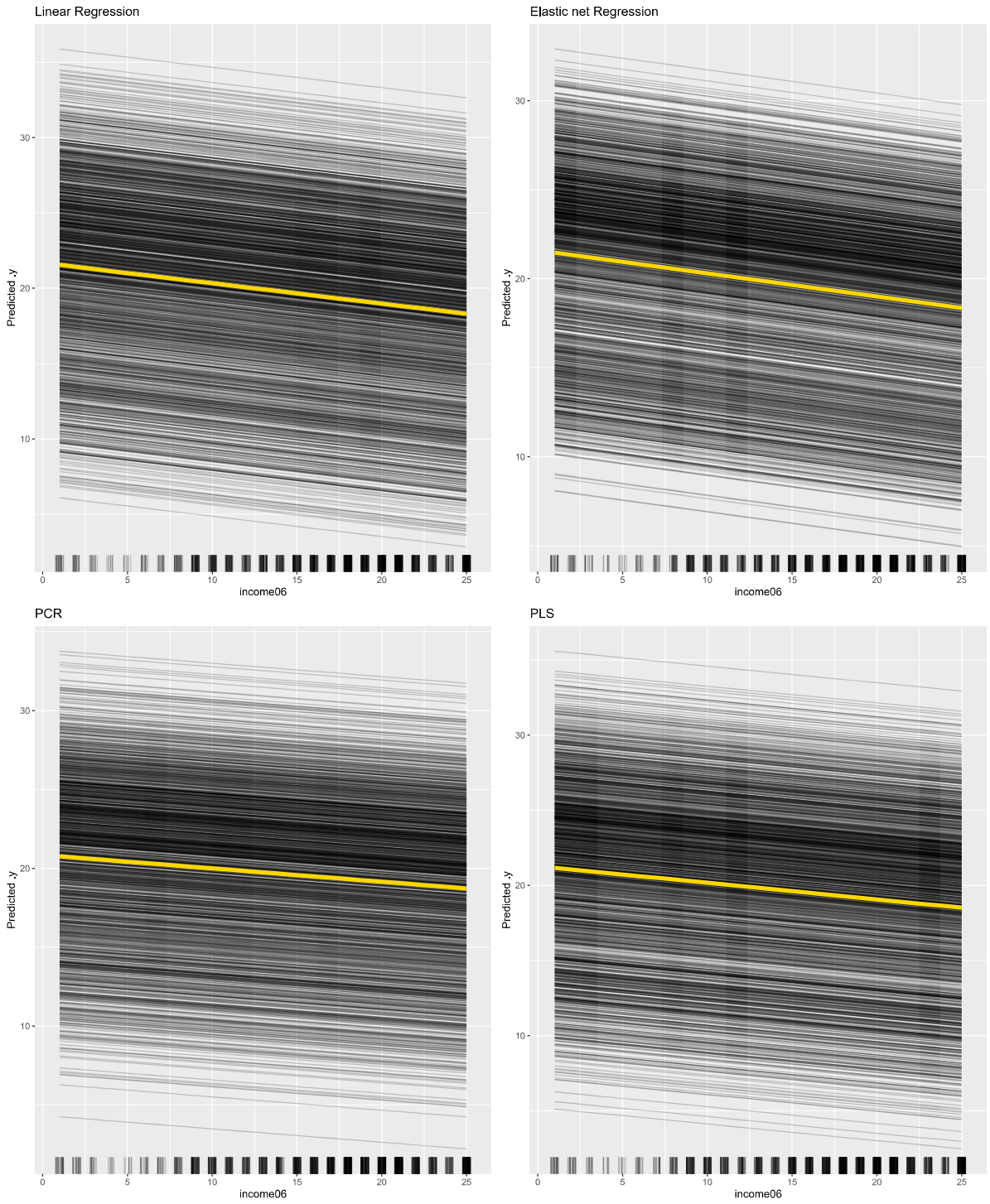
Partyid_3 indicates one of their friends to be Democrat, a Republican, or an Independent. WIth friend as Democrat, more liberal, people are more tended to have high egalitarianism attitudes, which might indicate that people wiht similar political view are more probably being friends or versa, political view of friends might influence one's own opinion, while we could not tell it from the current results.

However, *partyid_3* is not very significant important with PCR model(ranked low in feature importance

graph), which might be resulted from the unseparable between independent and Democrat and more deeply coming from the different transformation of these information into components which might 'distort' the orginial information.

Finally, plot the PDP of *income06* variable, which is also significant but have high interaction contribution to egalitarianism:

```
#call packages to calculate pdp for variable: partyid_3
pdp_income_lg <- FeatureEffect$new(predictor_eg_lg, "income06",method="pdp+ice")
pdp_income_en <- FeatureEffect$new(predictor_eg_en, "income06",method="pdp+ice")
pdp_income_pcr <- FeatureEffect$new(predictor_eg_pcr, "income06",method="pdp+ice")
pdp_income_pls <- FeatureEffect$new(predictor_eg_pls, "income06",method="pdp+ice")

#plot the results
p1 <- plot(pdp_income_lg) +
ggtitle("Linear Regression")
p2 <- plot(pdp_income_en) +
ggtitle("Elastic net Regression")
p3 <- plot(pdp_income_pcr) +
ggtitle("PCR")
p4 <- plot(pdp_income_pls) +
ggtitle("PLS")

#for aggregated visualization
p1 + p2 + p3 + p4
```



The marginal effect of income shows a clear negative non-linear relationship between income and egalitarianism, robustly across all 4 models, and well corresponds with the results of quesiton one in this homework- with increasing income, people tend to have *much lower* attitudes towards egalitarianism.