Mingtao_Gao_HW4

Mingtao Gao

2/16/2020

```
# Packages used in this assignment
library(ISLR)
library(boot)
library(caret)
```

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

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
## Loading required package: ggplot2
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
library(leaps)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("margins")
```

Non-linear regression Egalitarianism and income 0.

```
# Load the dataset from file
test <- read.csv('data/gss_test.csv')
train <- read.csv('data/gss_train.csv')
set.seed(1234)
```

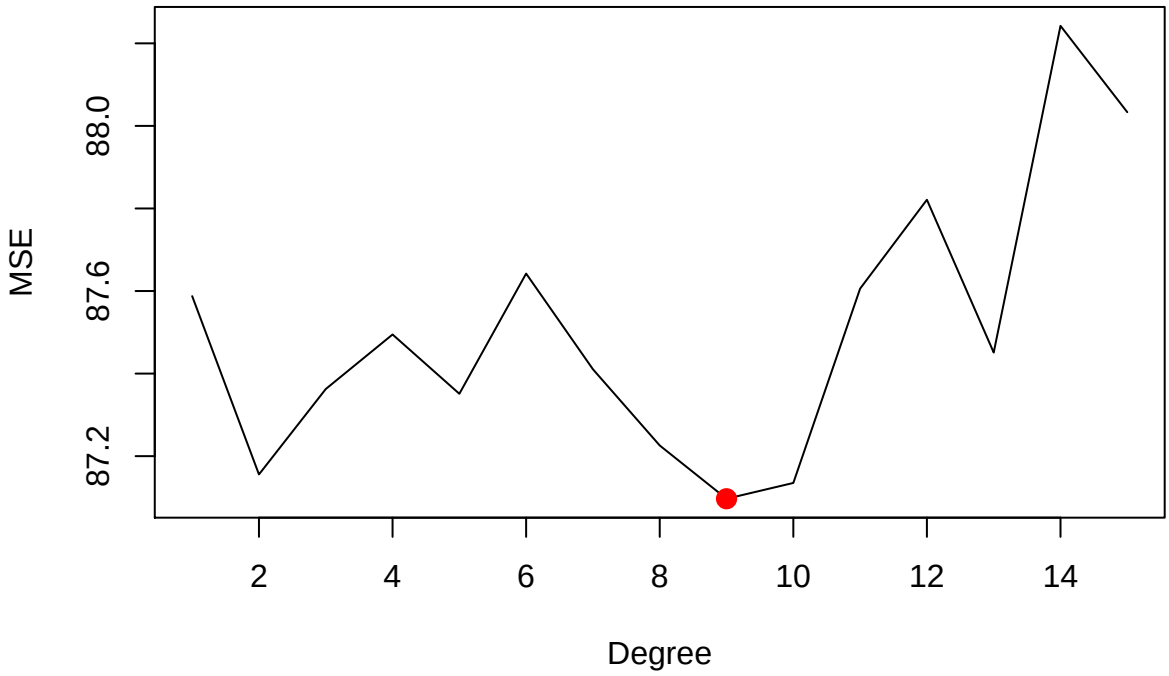
1. Polynomial regression

```
errs <- rep(NA, 10)
# Loop through powers of income
```

```r
for (i in 1:15) {
  fit <- glm(egalit_scale ~ poly(income06, i), data=train)
  errs[i] <- cv.glm(train, fit, K=10)$delta[1]
}
errs
```

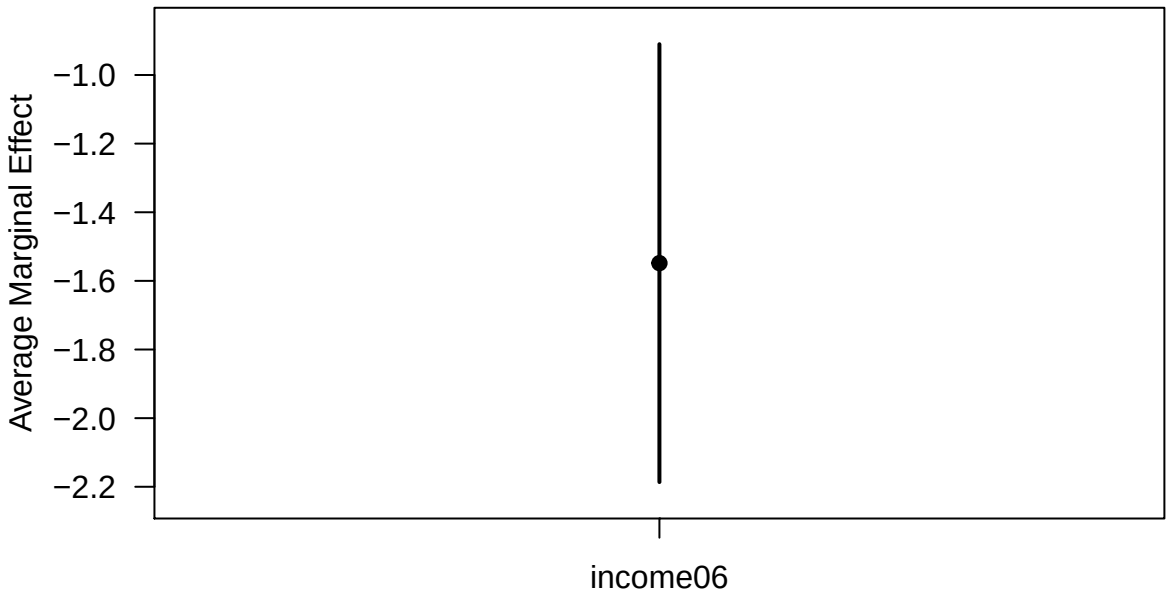
```
##  [1] 87.58758 87.15571 87.36262 87.49474 87.35113 87.64233 87.41045 87.22589
##  [9] 87.09682 87.13513 87.60574 87.82109 87.45106 88.24233 88.03326
```

```r
# Plot the data to illustrate MSE
plot(x=1:15, y=errs, xlab="Degree", ylab="MSE", type="l")
d.min <- which.min(errs)
points(which.min(errs), errs[which.min(errs)], col="red", cex=2, pch=20)
```



We may see that d = 9 is the optimal degree for the polynomial. In fact, 2nd degree polynomial also shows a comparably low MSE value. If we examine the p values of 2nd and 9th degree polynomial model, we might find both models appear to be a good fit to this data. But, based on the graph, I will use 9 as the degree for polynomial regression.

```r
# Fit a 9th degree polynomial model to training data and plot AME of income06
fit <- lm(egalit_scale ~ stats::poly(income06, 9), data = train)
plot(margins(fit))
```

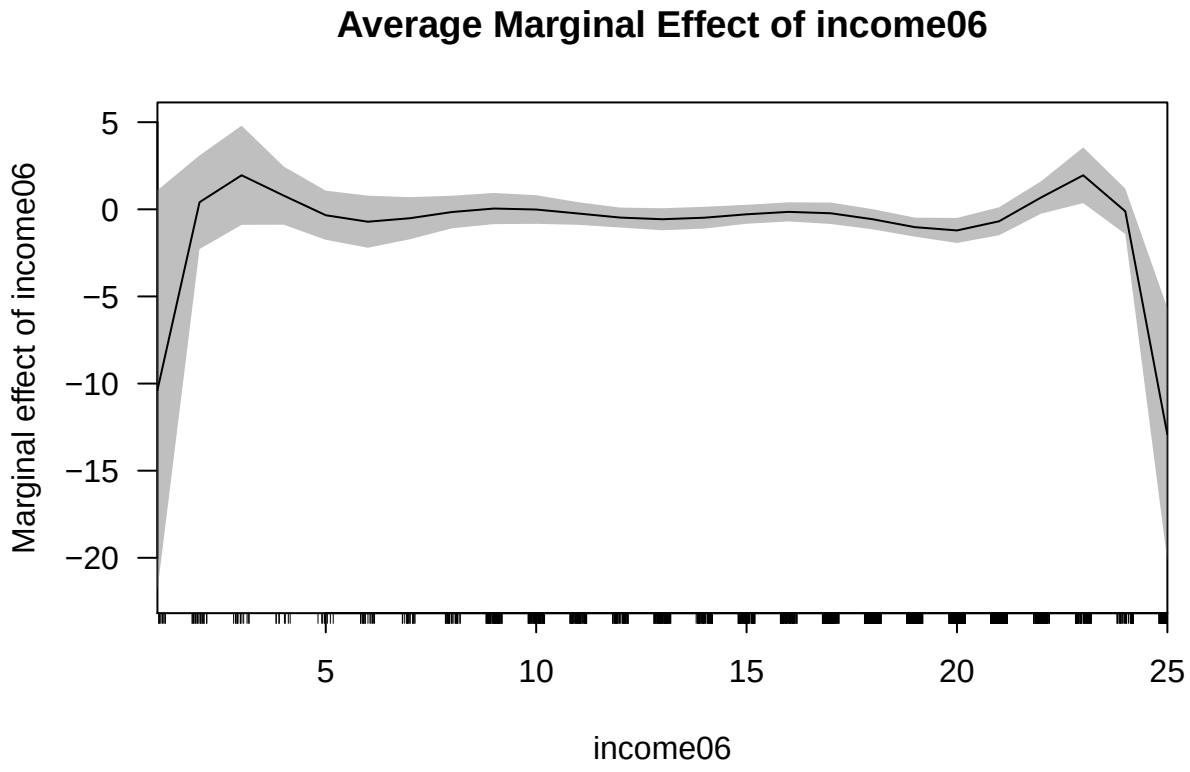


Since we are only using income06 as one predictors for response variable, egalit_scale. The AME is calculate as -1.548284, which means that, on average, a one-unit increase in income decreases the probability of respondent egalit_scale by 1.55 percentage points.

```
margins(fit, at = list(income06 = fivenum(train$income06)))
```

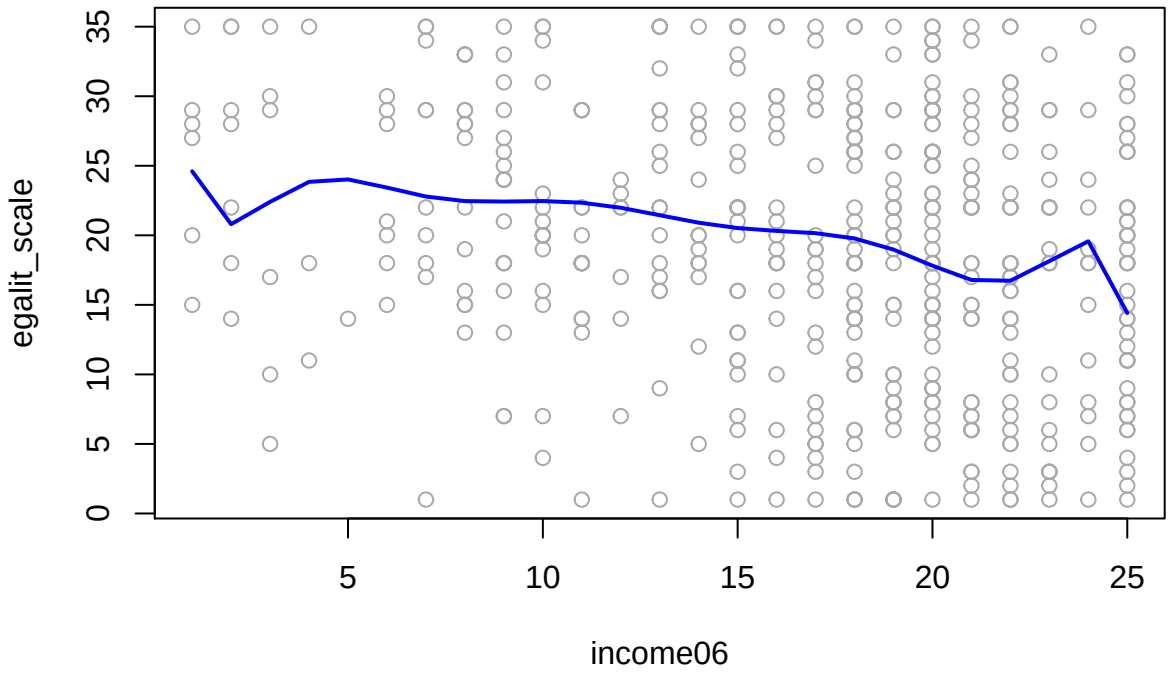
```
## Average marginal effects at specified values

## lm(formula = egalit_scale ~ stats::poly(income06, 9), data = train)

##  at(income06) income06
##             1 -10.3748
##            13  -0.5688
##            18  -0.5708
##            21  -0.6864
##            25 -12.9298
```

```
# Plot the marginal effects of values across all possible values of income06
cplot(fit, "income06", what="effect", main="Average Marginal Effect of income06")
```



We can see maginal effects of income06 across all possible values and its confidence intervals. From the table printed above and the graph, we can find that the average marginal effect of income06 is large and negative when income06 is extremely large or small. It means that when income06 is extremely large or small, it has greater negative marginal effects on egalit_scale.

```
# Plot the resulting polynomial fit to the data
plot(egalit_scale ~ income06, data=test, col="darkgrey")
incomelims <-  range(test$income06)
income.grid <-  seq(from = incomelims[1], to = incomelims[2])
lm.pred <- predict(fit, data.frame(income06=income.grid))
lines(income.grid, lm.pred, col="blue", lwd=2)
```

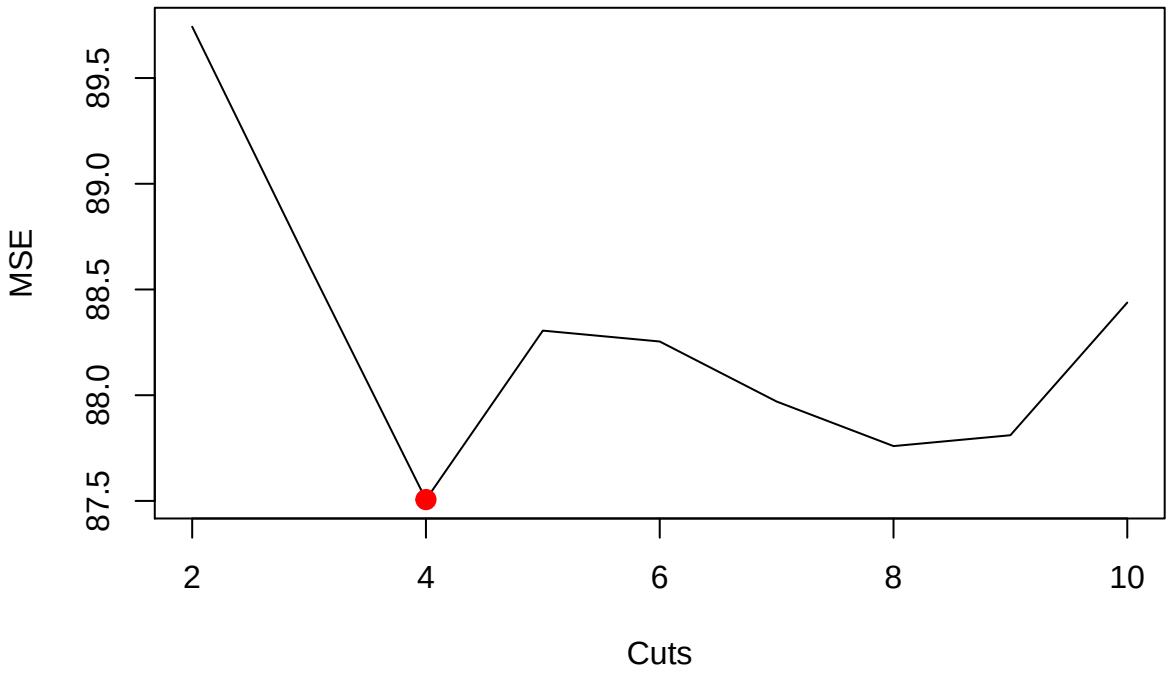



```
# Evaluate model performance by calculating MSE
preds = predict(fit, test)
mean((preds - test$egalit_scale)^2)
```

```
## [1] 87.85106
```

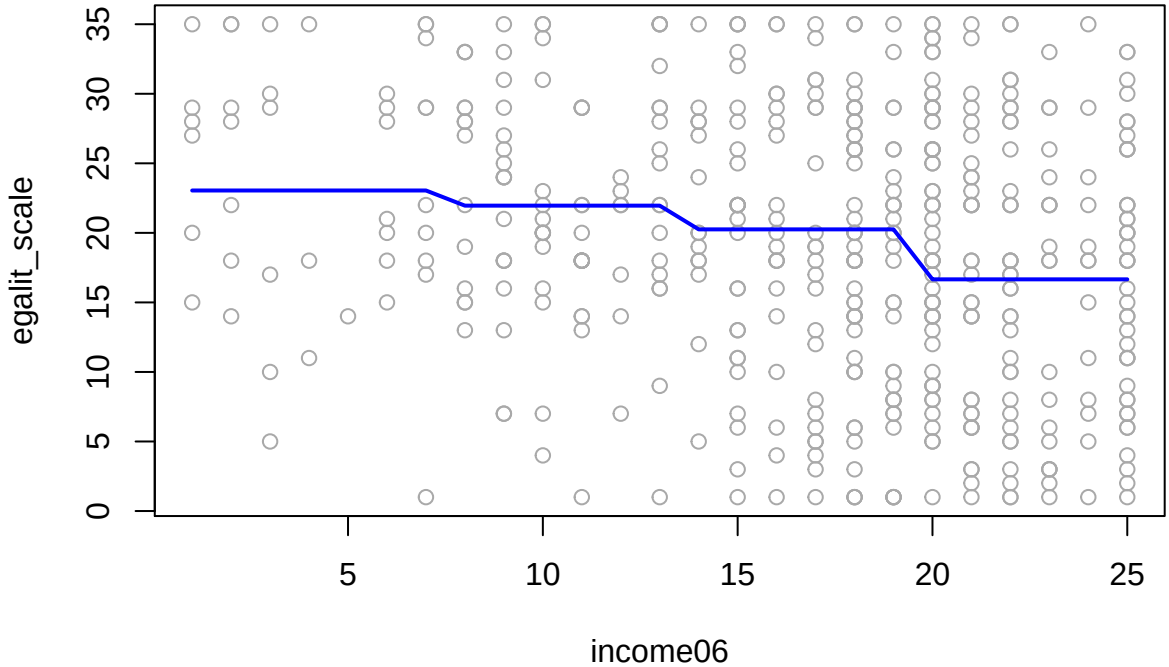
2. Step function

```
# For each cut perform 10-fold cross-validation
errs <- rep(NA, 10)
for (i in 2:10) {
    train$income.cut <- cut(train$income06, i)
    fit <- glm(egalit_scale ~ income.cut, data = train)
    errs[i] <- cv.glm(train, fit, K=10)$delta[1]
}

# Plot the fit and highlight the minimum
plot(2:10, errs[-1], xlab="Cuts", ylab="MSE", type="l")
c.min <- which.min(errs)
points(which.min(errs), errs[which.min(errs)], col="red", cex=2, pch=20)
```



We may see that the error is minimum for 4 cuts, which means if we break the range of income06 into 4 cuts and fit a different constant in each cut, we can prevent the global structure imposed by using polynomial functions.

```
# Plot the resulting step function fit to the data
plot(egalit_scale ~ income06, data=test, col="darkgrey")
incomelims <- range(test$income06)
income.grid <- seq(from = incomelims[1], to = incomelims[2])
fit <- lm(egalit_scale ~ cut(income06, 4), data=train)
lm.pred <- predict(fit, data.frame(income06=income.grid))
lines(income.grid, lm.pred, col="blue", lwd=2)
```



The solid curve displays the fitted value from a least squares regression of egalit_scale using step functions of

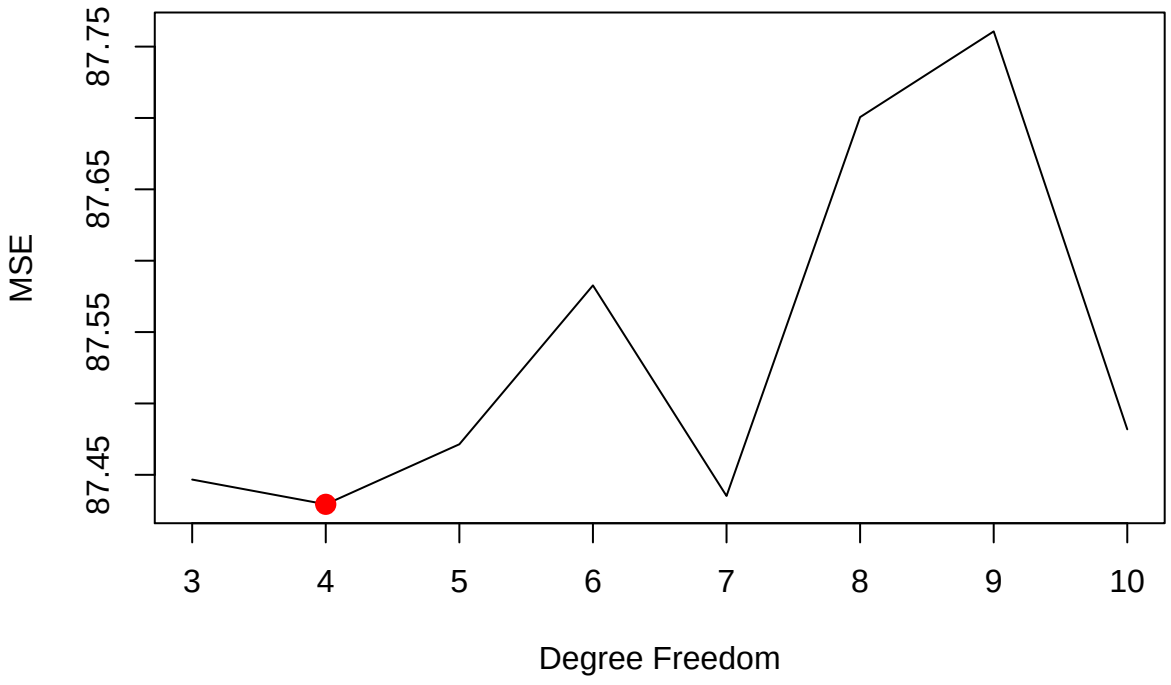
income06. We can see the range of income06 was broken into 4 cuts to fit the data.

```
# Evaluate model performance by calculating MSE
preds = predict(fit, test)
mean((preds - test$egalit_scale)^2)
```

```
## [1] 88.00247
```

3. Natural regression spline

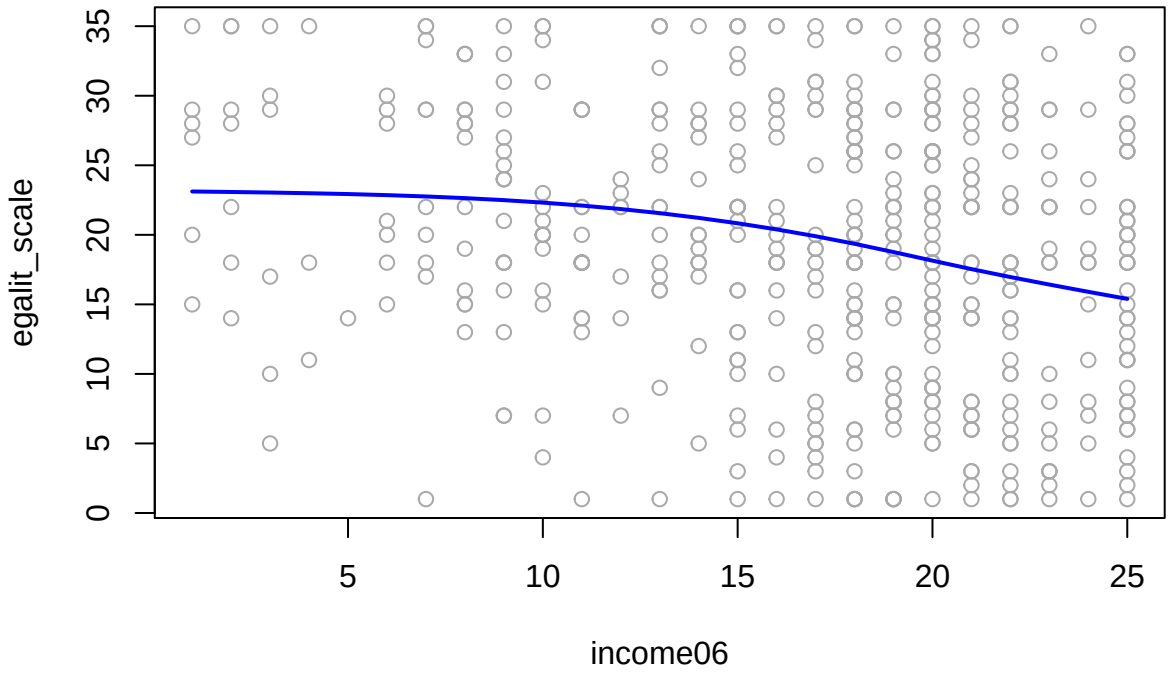
```
library(splines)

# For each degree freedom perform 10-fold cross-validation
errs <- rep(NA, 10)
for (i in 3:10) {
    fit <- glm(egalit_scale ~ ns(income06, df=i), data=train)
    errs[i] <- cv.glm(train, fit, K=10)$delta[1]
}

# Plot the fit and highlight the minimum
plot(3:10, errs[-c(1, 2)], xlab="Degree Freedom", ylab="MSE", type="l")
df.min <- which.min(errs)
points(which.min(errs), errs[which.min(errs)], col="red", cex=2, pch=20)
```



The graph shows 10-fold cross-validated mean squared errors for selecting the degrees of freedom when fitting splines to the GSS data. We may see that the MSE is minimum for 4 degrees of freedom. It means regression splines divides the range of income06 into 4 distinct regions and fit the data with a polynomial function in each region.

```
# Plot the resulting regression spline fit to the data
plot(egalit_scale ~ income06, data=test, col="darkgrey")
incomelims <- range(test$income06)
income.grid <- seq(from = incomelims[1], to = incomelims[2])
fit <- lm(egalit_scale ~ ns(income06, df=4), data=train)
lm.pred <- predict(fit, data.frame(income06=income.grid))
```

```
lines(income.grid, lm.pred, col="blue", lwd=2)
```




```
# Evaluate model performance by calculating MSE
preds = predict(fit, test)
mean((preds - test$egalit_scale)^2)
```

```
## [1] 86.30499
```

We can find that the MSE of using regression splines is smaller than MSE of both step functions and polynominal regression, because splines introduce flexibility by increasing the number of knots but keeping the degree fixed, which produces more stable estimates. Compared to polynominal regression, which increasing degrees of the function, regression splines often give better results to polynomial regression.

Egalitarianism and everything 4.

```
# Load the dataset from file
test <- read.csv('data/gss_test.csv')
train <- read.csv('data/gss_train.csv')
```

```
# Check missing values and zero values
nzv <- nearZeroVar(train, saveMetrics = TRUE)
nzv
```

```
##                 freqRatio percentUnique zeroVar   nzv
## age              1.029412     4.8615800   FALSE FALSE
## attend           1.279863     0.6076975   FALSE FALSE
## authoritarianism 1.233766     0.5401756   FALSE FALSE
## black            5.524229     0.1350439   FALSE FALSE
## born             5.985849     0.1350439   FALSE FALSE
## childs           1.004808     0.6076975   FALSE FALSE
## colath           2.104822     0.1350439   FALSE FALSE
## colrac           1.106686     0.1350439   FALSE FALSE
## colcom           2.226580     0.1350439   FALSE FALSE
## colmil           1.611993     0.1350439   FALSE FALSE
## colhomo          7.921687     0.1350439   FALSE FALSE
```

```
## colmslm           2.004057     0.1350439   FALSE FALSE
## con_govt          1.006173     0.2700878   FALSE FALSE
## degree            2.761364     0.3376097   FALSE FALSE
## egalit_scale      1.026786     2.3632681   FALSE FALSE
## evangelical       1.434694     0.2025658   FALSE FALSE
## grass             1.054092     0.1350439   FALSE FALSE
## happy             1.810268     0.2025658   FALSE FALSE
## hispanic_2        6.442211     0.1350439   FALSE FALSE
## homosex           1.059970     0.2700878   FALSE FALSE
## income06          1.124031     1.6880486   FALSE FALSE
## marital           1.604878     0.3376097   FALSE FALSE
## mode              6.753927     0.1350439   FALSE FALSE
## news              1.228571     0.3376097   FALSE FALSE
## owngun            1.977597     0.2025658   FALSE FALSE
## partyid_3         1.184261     0.2025658   FALSE FALSE
## polviews          2.678733     0.4726536   FALSE FALSE
## pornlaw2          2.226580     0.1350439   FALSE FALSE
## pray              1.088095     0.4051317   FALSE FALSE
## pres08            2.111345     0.1350439   FALSE FALSE
## reborn_r          1.358280     0.1350439   FALSE FALSE
## relig             2.026866     0.8777853   FALSE FALSE
## science_quiz      1.080537     0.7427414   FALSE FALSE
## sex               1.200594     0.1350439   FALSE FALSE
## sibs              1.112727     1.4179608   FALSE FALSE
## social_connect    1.096774     0.8777853   FALSE FALSE
## social_cons3      1.490741     0.2025658   FALSE FALSE
## south             1.682971     0.1350439   FALSE FALSE
## spend3            1.072727     0.2025658   FALSE FALSE
## teensex           5.863388     0.2700878   FALSE FALSE
## tolerance         1.538462     1.0803511   FALSE FALSE
## tvhours           1.220000     1.2829169   FALSE FALSE
## vetyears         15.963415     0.2700878   FALSE FALSE
## wordsum           1.308772     0.7427414   FALSE FALSE
## zodiac            1.118519     0.8102633   FALSE FALSE
```

There are no non-zero variance predictors or target. There is no need to delete non-zero variance predictors.

```
# Data pre-processing (standarization)
preprocessParams <- preProcess(select_if(train, is.numeric),
                               method=c("center", "scale"))
train_trans <- predict(preprocessParams, select_if(train, is.numeric))
preprocessParams <- preProcess(select_if(test, is.numeric),
                               method=c("center", "scale"))
test_trans <- predict(preprocessParams, select_if(test, is.numeric))

# Separate dataset into predictors and response variable
xtrain <- model.matrix (egalit_scale~., train_trans)[,-1]
ytrain <- train_trans$egalit_scale
xtest <- model.matrix (egalit_scale~., test_trans)[,-1]
ytest <- test_trans$egalit_scale
```

a. Linear regression

```
# Fit the data with linear regression model
train_control <- trainControl(method="cv", number=10)
lm.fit <- train(egalit_scale ~ .,
```

```
                data=train_trans,
                trControl=train_control,
                method="lm")
```

```
# Evaluate model performance by calculating MSE
preds = predict(lm.fit, test_trans)
mean((preds - test_trans$egalit_scale)^2)
```

```
## [1] 0.934211
```

b. Elastic net regression

```
# Fit the data with elastic net regression
elasnet.fit <- train(
  x = xtrain,
  y = ytrain,
  method = "glmnet",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10)

# Display the best lambda and alpha combination
elasnet.fit$bestTune
```

```
##    alpha     lambda
## 37   0.5 0.01661469
```

```
# Use Elastic Net regression with best lambda and alpha and calculate test MSE
elasnet.fit$results %>%
  filter(alpha == elasnet.fit$bestTune$alpha, lambda == elasnet.fit$bestTune$lambda)
```

```
##   alpha     lambda      RMSE  Rsquared       MAE     RMSESD RsquaredSD
## 1   0.5 0.01661469 0.9494164 0.1034468 0.7855902 0.02763477 0.03627152
##        MAESD
## 1 0.01896721
```

```
# Evaluate model performance by calculating MSE
preds = predict(elasnet.fit, test_trans)
mean((preds - test_trans$egalit_scale)^2)
```

```
## [1] 0.9292995
```

c. Principal component regression

```
# Fit the data with PCR
library(pls)
```

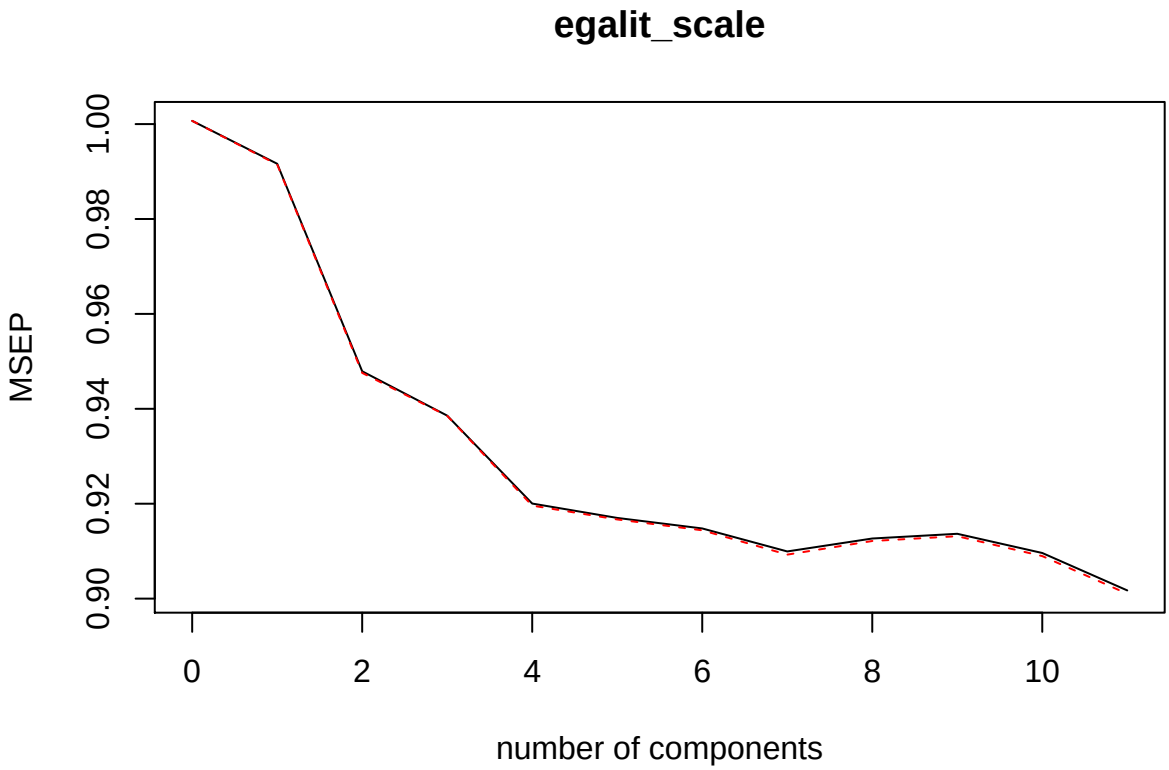
```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
pcr.fit <- pcr(egalit_scale ~ ., data=train_trans,
               scale=TRUE, validation="CV")
```

```
validationplot(pcr.fit, val.type = "MSEP")
```



Based on the graph, we may see the MSE minimizes when we use 10 components for PCR. However, to make sure the result is correct, we need to take a detailed look at the results.

```
# Find the number of components with lowest cross validation error
cv = RMSEP(pcr.fit)
pcr.ncomps = which.min(cv$val[estimate = "adjCV", , ]) - 1
pcr.ncomps
```

```
## 11 comps
##       11
```

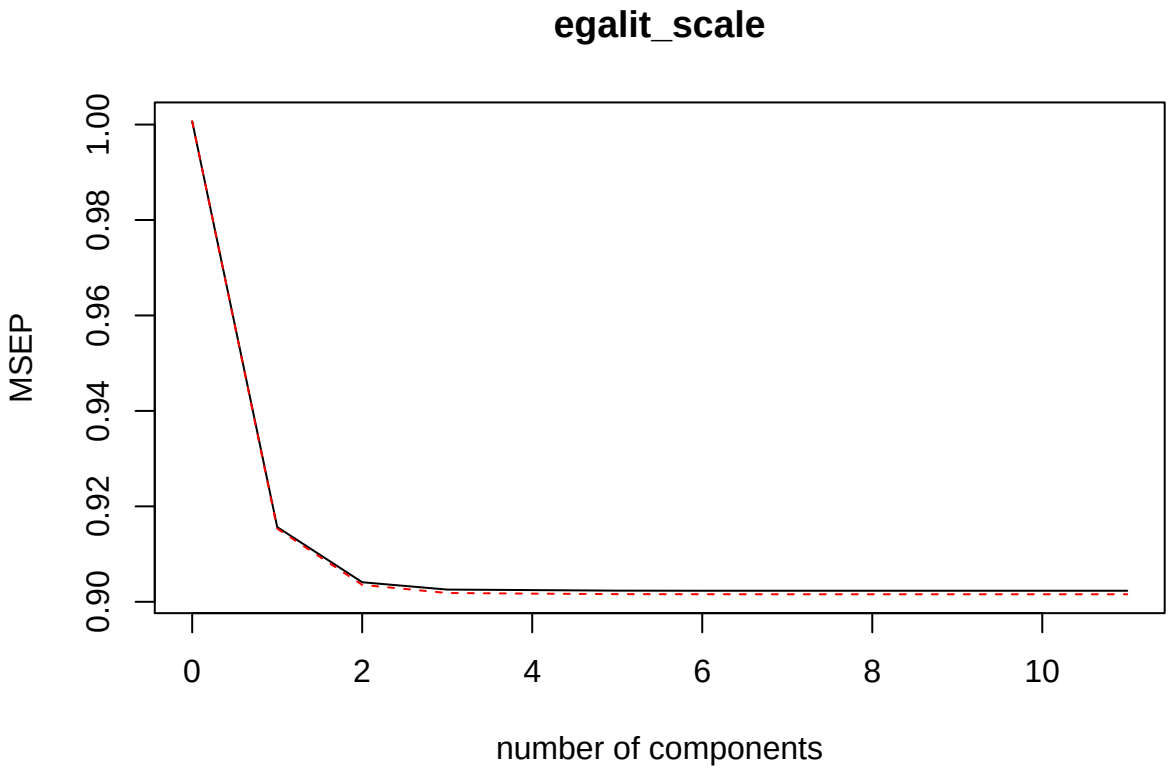
When number of components is 11, error is minimized. We will use 11 components for PCR training and testing.

```
pcr.fit <- pcr(egalit_scale ~ ., data=train_trans, ncomp=pcr.ncomps)

# Evaluate model performance by calculating MSE
preds = predict(pcr.fit, test_trans)
mean((preds - test_trans$egalit_scale)^2)
```

```
## [1] 0.9383643
```

d. Partial least squares regression

```
pls.fit <- plsr(egalit_scale ~ ., data=train_trans,
                scale=TRUE, validation="CV")
validationplot(pls.fit, val.type = "MSEP")
```



Based on the graph, we may see the MSE decreases first and remain stable after we use 2 components for PLS. However, we still need to take a detailed look at the results.

```
# Find the number of components with lowest cross validation error
cv = RMSEP(pls.fit)
pls.ncomps = which.min(cv$val[estimate = "adjCV", , ]) - 1
pls.ncomps
```

```
## 7 comps
##       7
```

When number of components is 8, error is minimized. We will use 8 components for PLS training and testing.

```
# Fit the model with best number of components
pls.fit = plsr(egalit_scale ~ ., data=train_trans, ncomp=pls.ncomps)

# Evaluate model performance by calculating MSE
preds = predict(pls.fit, test_trans)
mean((preds - test_trans$egalit_scale)^2)
```

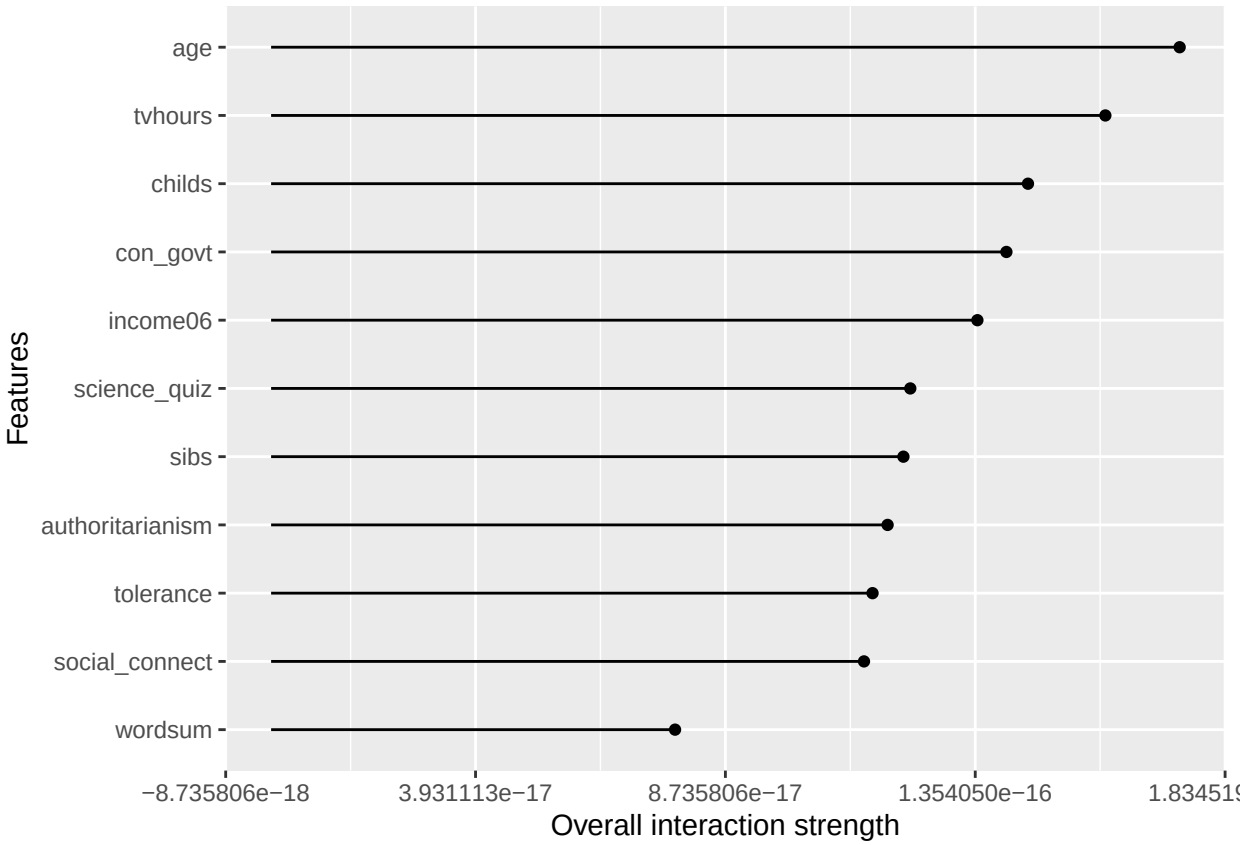
```
## [1] 0.9332759
```

2. Feature importance

a. Linear regression

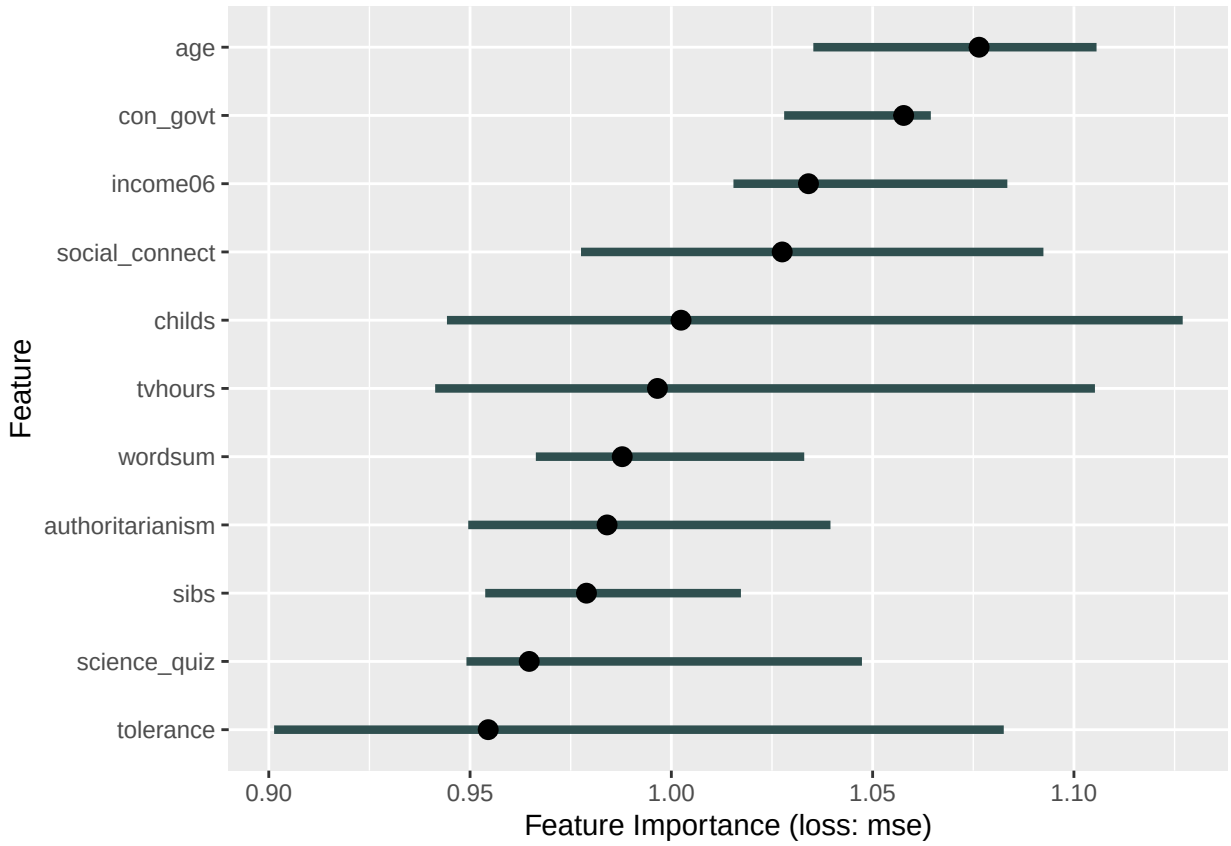
```
# I used iml packages to analyze feature importance for linear regression model
library(iml)
# Create predictor object
lm.predictor = Predictor$new(lm.fit, data=test_trans, y="egalit_scale")
```

```
# Identify features with largest interaction effects in each model
interact = Interaction$new(lm.predictor)
plot(interact)
```



The plot displays how strongly features interact with any other features. We can see that social_connect displays the highest interaction strength. In general, all features display a certain level of interaction with each other.

```
# Compute feature importance based on specified loss metric
imp = FeatureImp$new(lm.predictor, loss = "mse")
plot(imp)
```

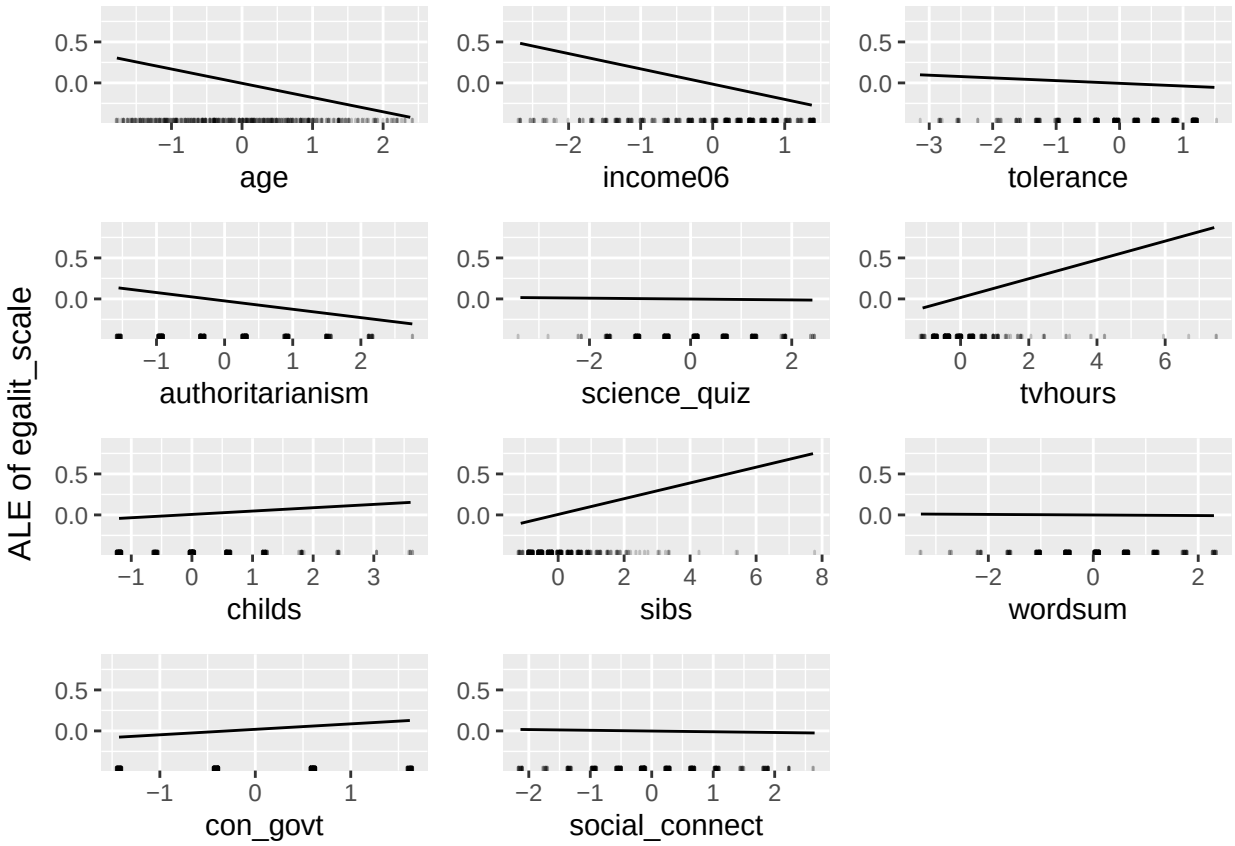



```
imp$results
```

```
##             feature importance.05 importance importance.95 permutation.error
## 1               age     1.0352705  1.0764416      1.105621         1.0056236
## 2          con_govt     1.0280203  1.0577096      1.064456         0.9881240
## 3          income06     1.0154255  1.0340621      1.083459         0.9660323
## 4    social_connect     0.9775638  1.0275378      1.092424         0.9599371
## 5            childs     0.9442588  1.0024116      1.127021         0.9364640
## 6           tvhours     0.9413355  0.9965426      1.105220         0.9309811
## 7           wordsum     0.9663376  0.9878021      1.033006         0.9228156
## 8  authoritarianism     0.9495469  0.9840370      1.039526         0.9192982
## 9              sibs     0.9537660  0.9789162      1.017283         0.9145144
## 10     science_quiz     0.9491017  0.9646945      1.047346         0.9012283
## 11        tolerance     0.9013277  0.9545055      1.082576         0.8917096
```

The feature importance measure works by shuffling each feature and measuring how much the performance drops. Based on the graph, we find that in linear regression model, age is the most important feature in affecting the response variable egalit_scale.

```
# Generate Partial Dependence Plot
effs = FeatureEffects$new(lm.predictor)
plot(effs)
```



Besides knowing which features were important, we are interested in how the features influence the predicted outcome. Since we used linear regression model, all feature effects are linear. From above graphs, we can see predictors with low slopes are those ranked low at importance, including tolerance, wordsum, social_connect, con_govt. Yet, age, income06 and tvhours all display comparably high slopes, either negative or positive.

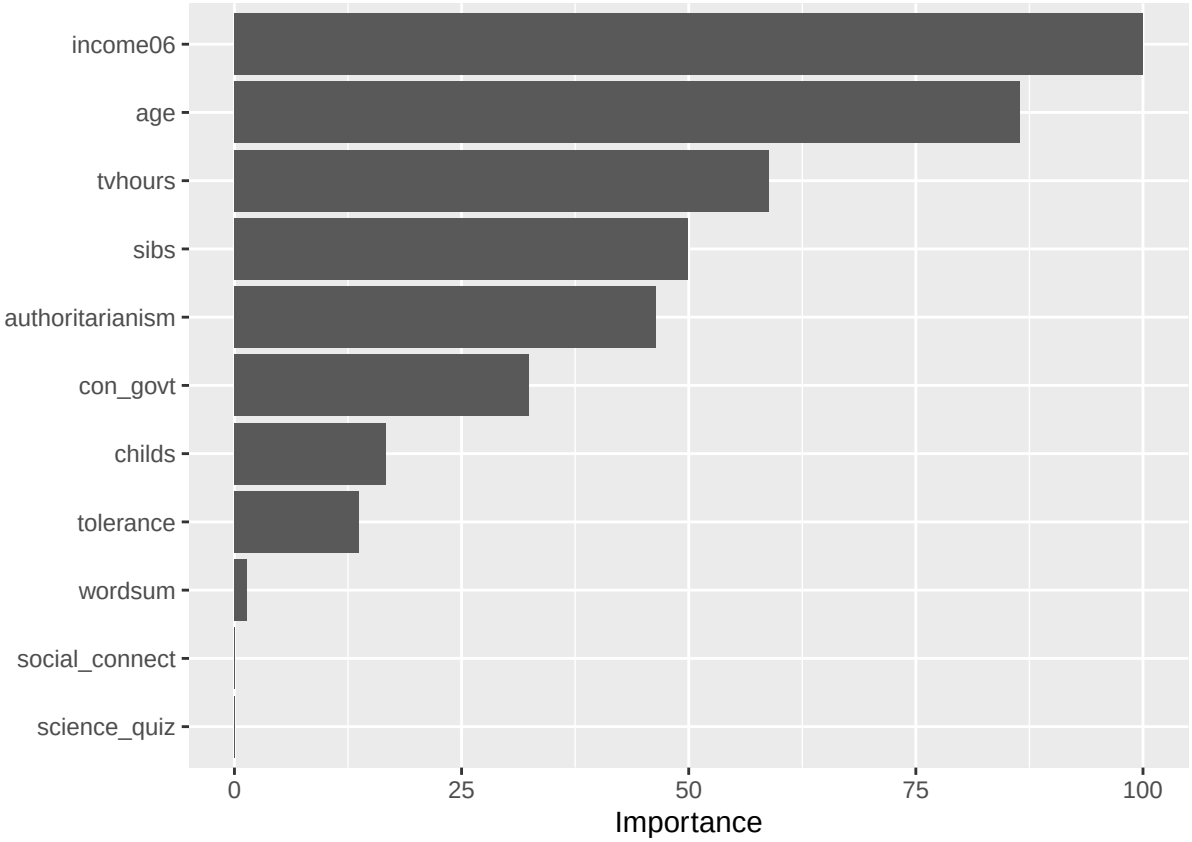
b. Elastic net regression

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
vip(elasnet.fit, num_features = 11)
```

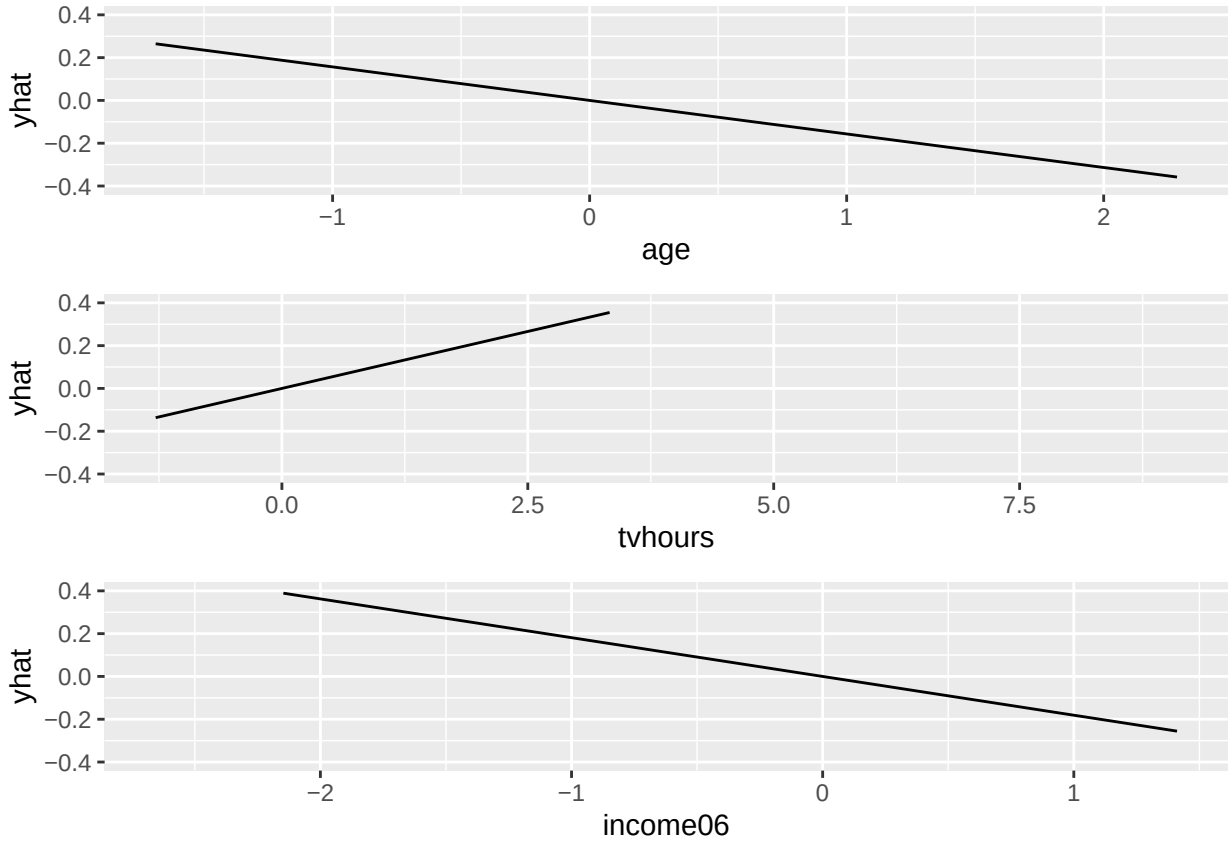


We can see that using elastic net regression, the most three important features are income06, age, and tvhours. The results are very similar to linear regression model.

```
library(pdp)
library(ggplot2)
# Plot partial dependence plot of top three most important variables
par.age <- partial(elasnet.fit, pred.var = c("age"), chull = TRUE)
plot.age <- autoplot(par.age) + ylim(c(-0.4, 0.4))

par.tvhours  <- partial(elasnet.fit, pred.var = c("tvhours"), chull = TRUE)
plot.tvhours  <- autoplot(par.tvhours) + ylim(c(-0.4, 0.4))

par.income  <- partial(elasnet.fit, pred.var = c("income06"), chull = TRUE)
plot.income  <- autoplot(par.income) + ylim(c(-0.4, 0.4))

grid.arrange(plot.age, plot.tvhours, plot.income)
```

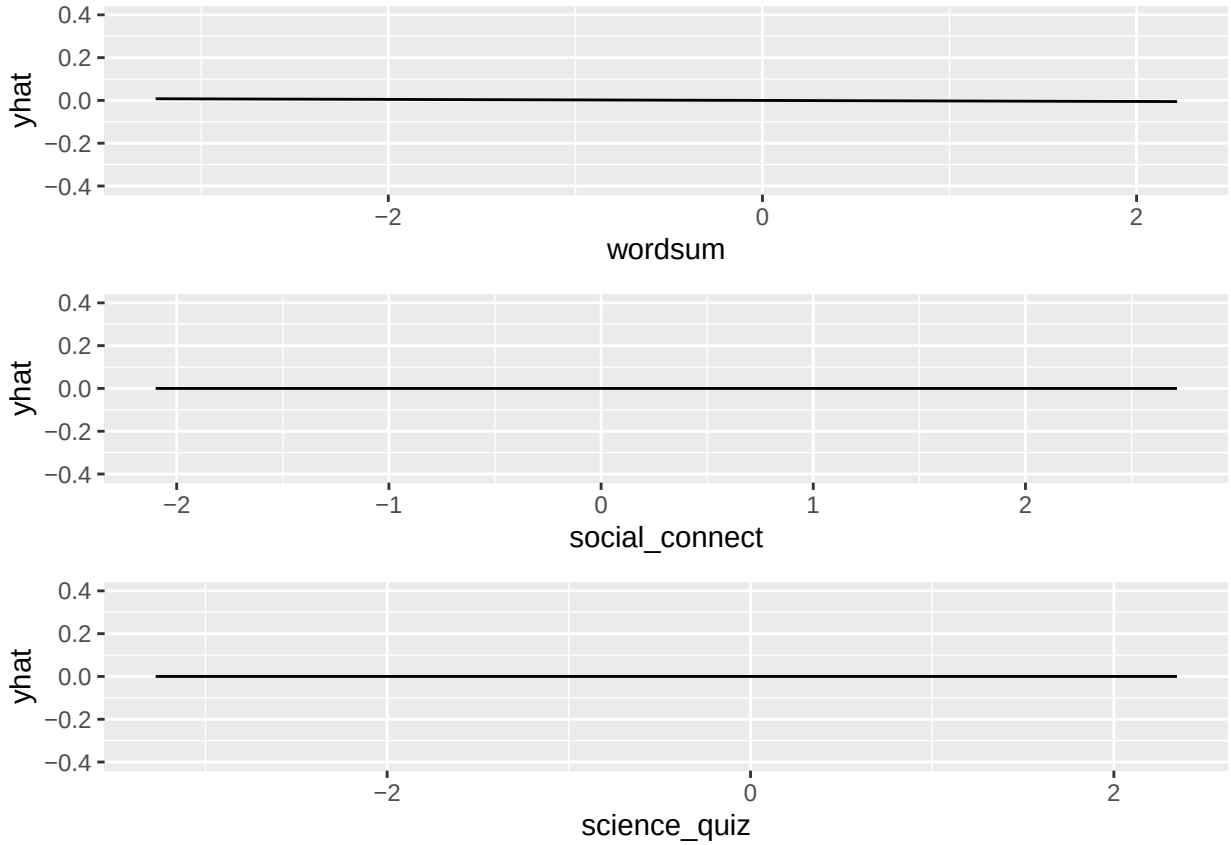
```
## Warning: Removed 10 rows containing missing values (geom_path).
```

```
## Warning: Removed 3 rows containing missing values (geom_path).
```



We can see the first three important variables have strong either or negative relationship with predicted value of egalit_scale.

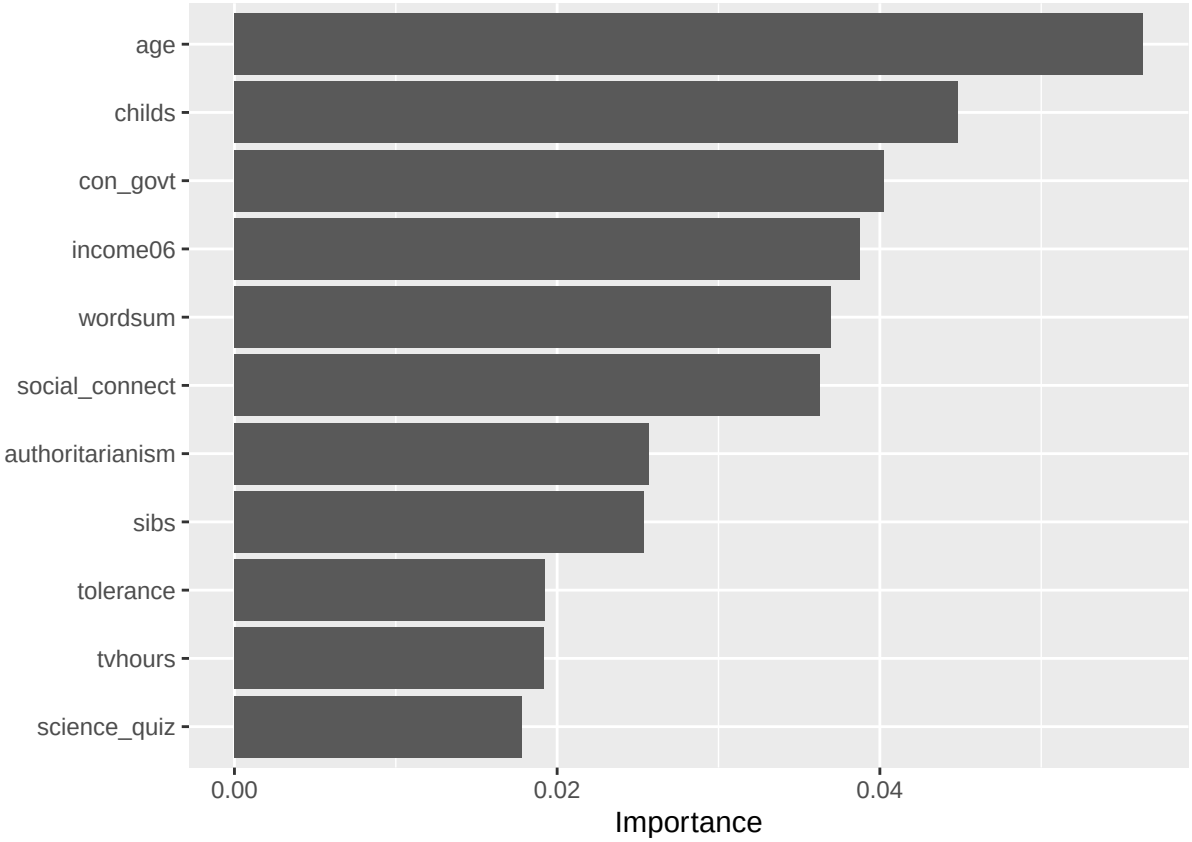
```
# Plot partial dependence plot of three least important variables
par.ws <- partial(elasnet.fit, pred.var = c("wordsum"), chull = TRUE)
plot.ws <- autoplot(par.ws) + ylim(c(-0.4, 0.4))

par.sc  <- partial(elasnet.fit, pred.var = c("social_connect"), chull = TRUE)
plot.sc  <- autoplot(par.sc) + ylim(c(-0.4, 0.4))

par.sq  <- partial(elasnet.fit, pred.var = c("science_quiz"), chull = TRUE)
plot.sq  <- autoplot(par.sq) + ylim(c(-0.4, 0.4))

grid.arrange(plot.ws, plot.sc, plot.sq)
```



We can see the last three important variables have no relationship with predicted value of egalit_scale, because all graphs show a flat line.

c. Principal component regression

```
vip(pcr.fit, num_features=pcr.ncomps, method="model")
```



For a PLS model, variable importance can be computed using the weighted sums of the absolute regression coefficients. We can use vip::vip() to extract and plot the most important variables. The importance measure is normalized from 100 (most important) to 0 (least important). We can see using principle component regression, the most important four features are age, childs, con_govt, income06.

```
# Plot partial dependence plot of top four most important variables
par.age <- partial(pcr.fit, pred.var = c("age"), chull = TRUE)
```

```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.age <- autoplot(par.age) + ylim(c(-0.4, 0.4))

par.childs  <- partial(pcr.fit, pred.var = c("childs"), chull = TRUE)
```

```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.childs  <- autoplot(par.childs) + ylim(c(-0.4, 0.4))

par.govt <- partial(pcr.fit, pred.var = c("con_govt"), chull = TRUE)
```

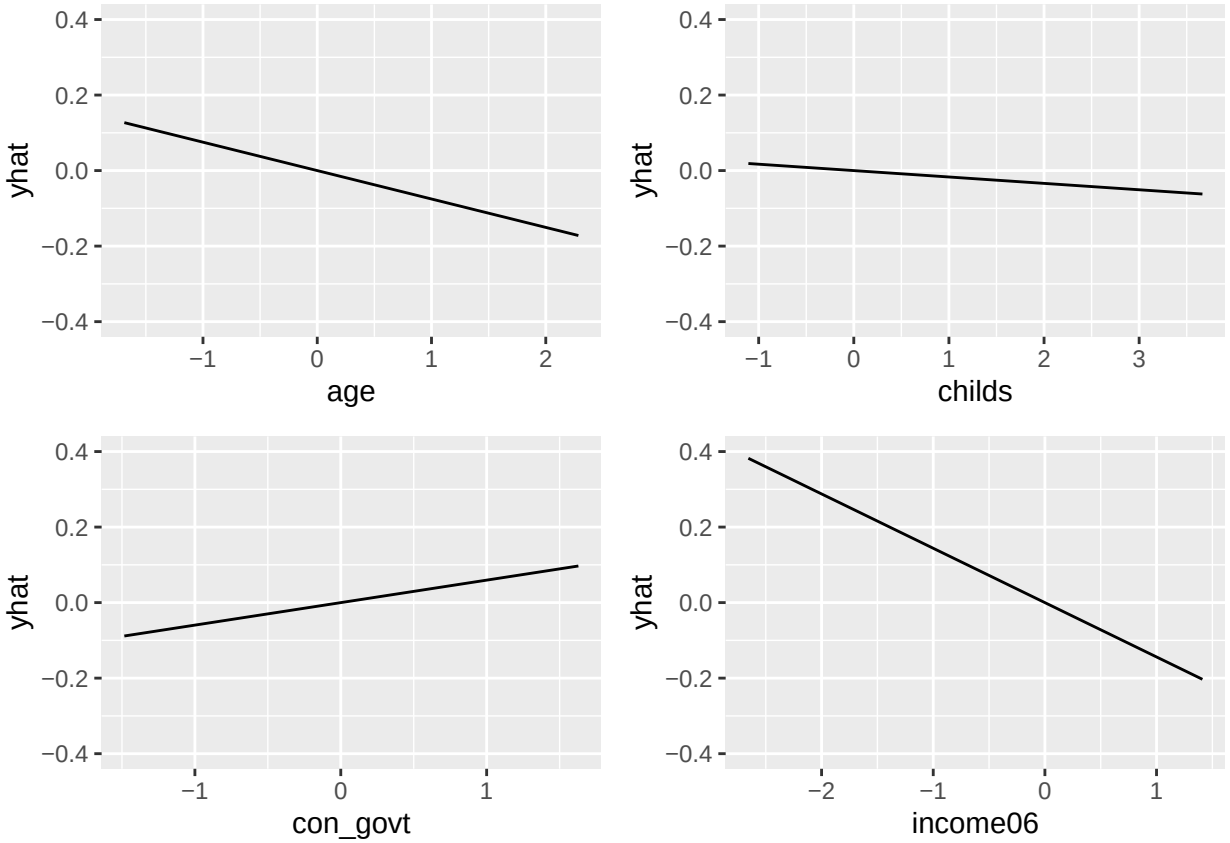
```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.govt <- autoplot(par.govt) + ylim(c(-0.4, 0.4))

par.income  <- partial(pcr.fit, pred.var = c("income06"), chull = TRUE)
```

```
## Warning in super_type.default(object): `type` could not be determined; assuming
```

```
## `type = "regression"`
```

```
plot.income  <- autoplot(par.income) + ylim(c(-0.4, 0.4))

grid.arrange(plot.age, plot.childs, plot.govt, plot.income)
```



We can see from the graphs that three out of four of the most important predictors have a negative relationship with egalitarian scale.

```
# Plot partial dependence plot of four least important variables
par.sibs <- partial(pcr.fit, pred.var = c("sibs"), chull = TRUE)

## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
plot.sibs <- autoplot(par.sibs) + ylim(c(-0.05, 0.8))

par.tolerance  <- partial(pcr.fit, pred.var = c("tolerance"), chull = TRUE)

## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
plot.tolerance  <- autoplot(par.tolerance) + ylim(c(-0.05, 0.8))

par.tvhours <- partial(pcr.fit, pred.var = c("tvhours"), chull = TRUE)

## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
plot.tvhours <- autoplot(par.tvhours) + ylim(c(-0.05, 0.8))
```

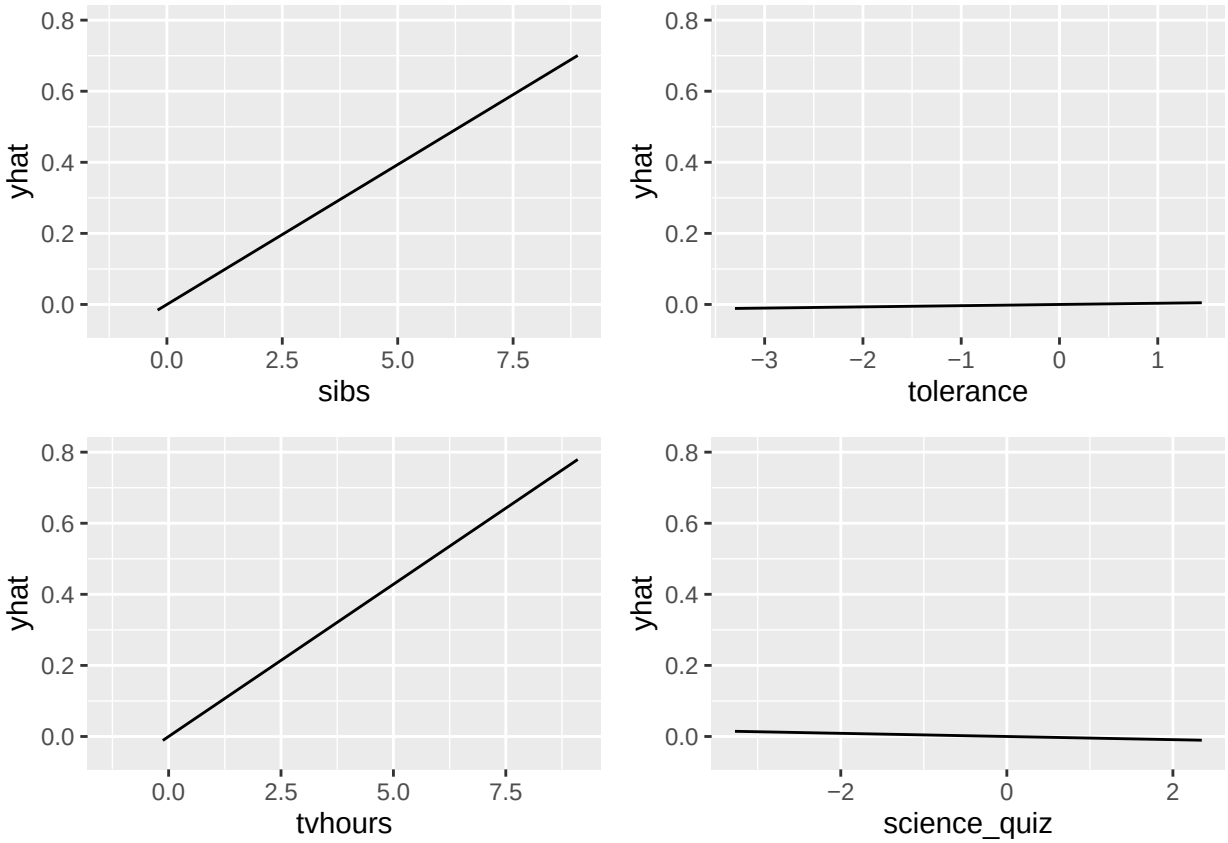
```
par.sq  <- partial(pcr.fit, pred.var = c("science_quiz"), chull = TRUE)
```

```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.sq  <- autoplot(par.sq) + ylim(c(-0.05, 0.8))

grid.arrange(plot.sibs, plot.tolerance, plot.tvhours, plot.sq)
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```

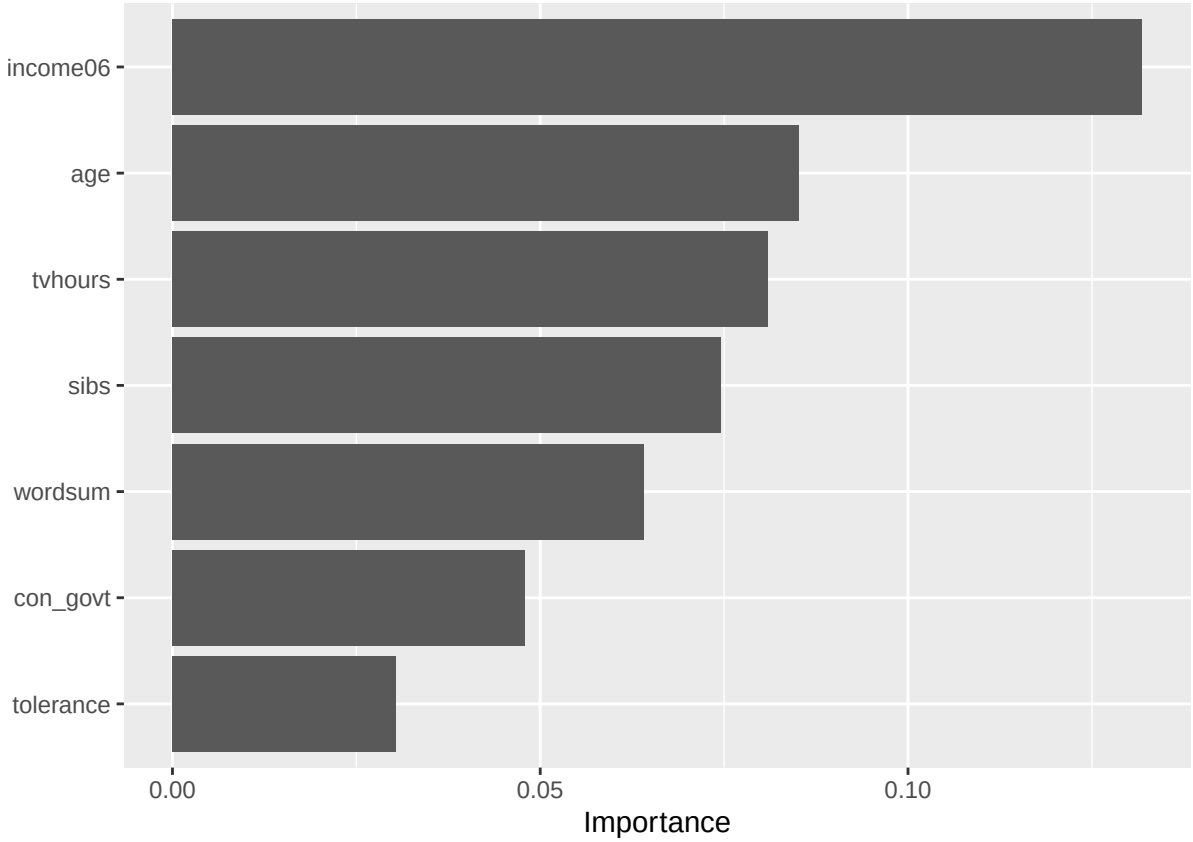
```
## Warning: Removed 2 rows containing missing values (geom_path).
```



We can see from the graphs that two of least important predictors have a positive relationship with egalitarian scale and the other two has no strong correlation with egalit_scale.

d. Partial least squares regression

```
vip(pls.fit, num_features=pls.ncomps, method="model")
```



The PLS model only takes 8 predicators to train the model. The three most important variables are income06, age, and tvhours. The three least important variables are con_govt, tolerance, and science_quiz.

```
# Plot Partial Dependence Plot of top two most important variable
par.income  <- partial(pls.fit, pred.var = c("income06"), chull = TRUE)
```

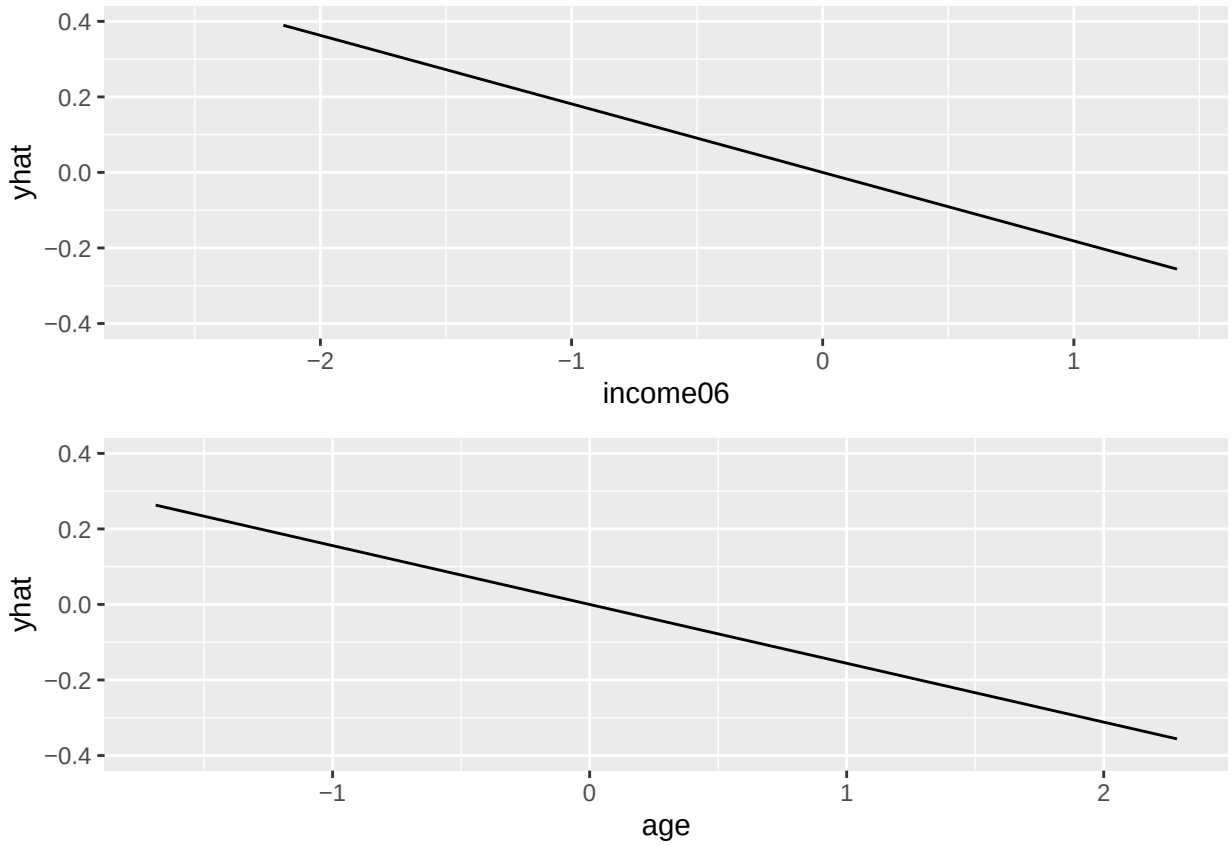
```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.income  <- autoplot(par.income) + ylim(c(-0.4, 0.4))

par.age <- partial(pls.fit, pred.var = c("age"), chull = TRUE)
```

```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.age <- autoplot(par.age) + ylim(c(-0.4, 0.4))

grid.arrange(plot.income, plot.age)
```

```
## Warning: Removed 3 rows containing missing values (geom_path).
```



From the above graphs, age and income06 both appear to be most important features for PCR and PLS models and in both models, they appear to have strong negative relationship with predicated value for egalit_scale.

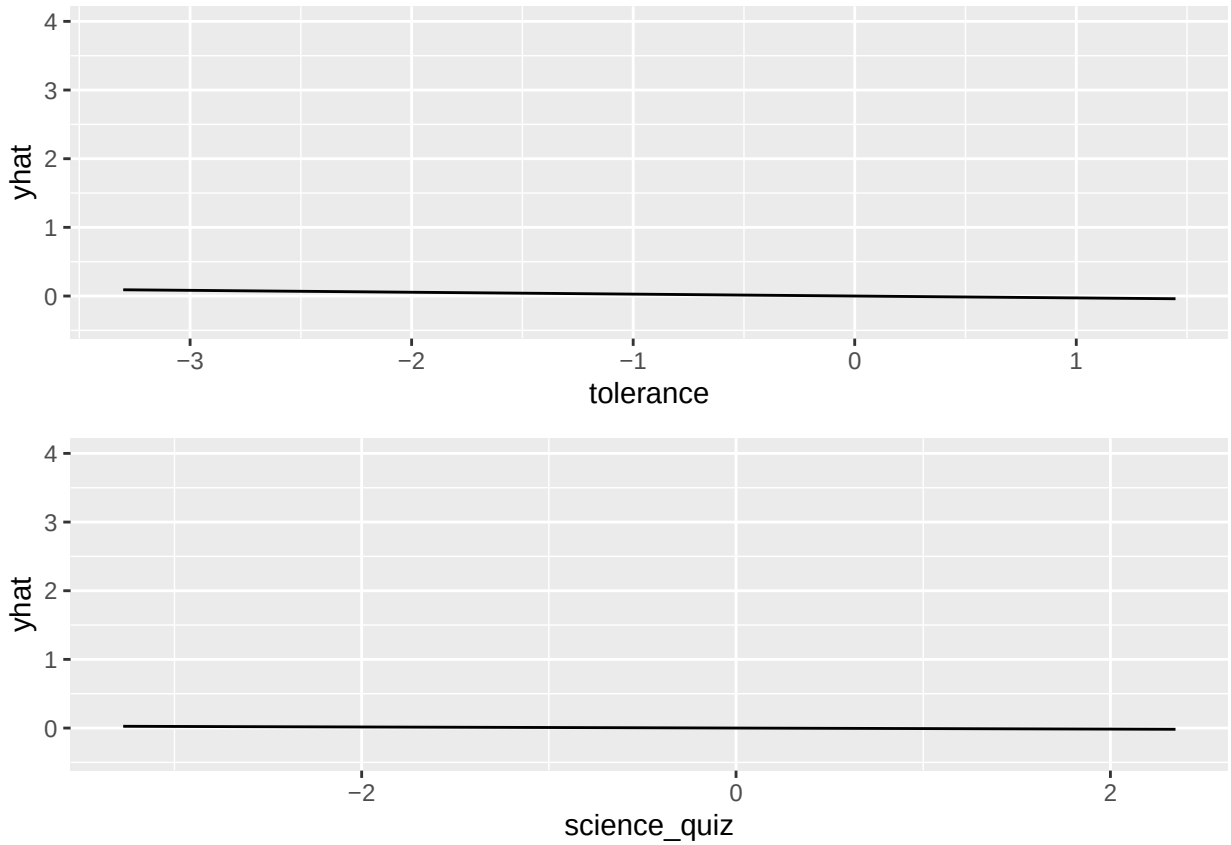
```
# Plot Partial Dependence Plot of three least important variable
par.tole  <- partial(pls.fit, pred.var = c("tolerance"), chull = TRUE)
```

```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.tole  <- autoplot(par.tole) + ylim(c(-0.4, 4))

par.sq <- partial(pls.fit, pred.var = c("science_quiz"), chull = TRUE)
```

```
## Warning in super_type.default(object): `type` could not be determined; assuming
## `type = "regression"`
```

```
plot.sq <- autoplot(par.sq) + ylim(c(-0.4, 4))

grid.arrange(plot.tole, plot.sq)
```



From the above graphs, tolerance and science_quiz both appear to be least important features for PCR and PLS models. In both models, they appear to have no relationship with predicated value for egalit_scale, since lines are nearly flat in two graphs.

Comparing different models used in question 2. We can find that age and income06 appear to be the most important features in those models and least important features vary across different models. The linear regression model demonstrates the basic linear model and how each features play a role in the model. Elastic net, PCR and PLS are also linear models but they apply different methods to reduce dimension to achieve better performance. Thus, we can see the features that lead to most of the variability in the data, as well as the relationship with the response are more important and least important features generally display no correlation with the predicated values of egalit_scale. Due to difference in those methods, the importance of features they display slightly varies from each other.