Problem Set 4

Akira Masuda

2020/2/16

```
Course: MACS30100 Perspectives on Computational Modeling (Winter 2020)
Author: Akira Masuda (ID: alakira)
```

```
knitr::opts_chunk$set(message = FALSE, warning = FALSE)
knitr::opts_chunk$set(fig.width=6,fig.height=3.4,fig.align='center')
```

```
library(knitr)
library(ggplot2)
library(tidyverse)
library(splines)
library(leaps)
library(glmnet)
library(caret)
library(DT)
# options(width=1000)
rm(list=ls())
set.seed(1100)
```

Non-linear regression

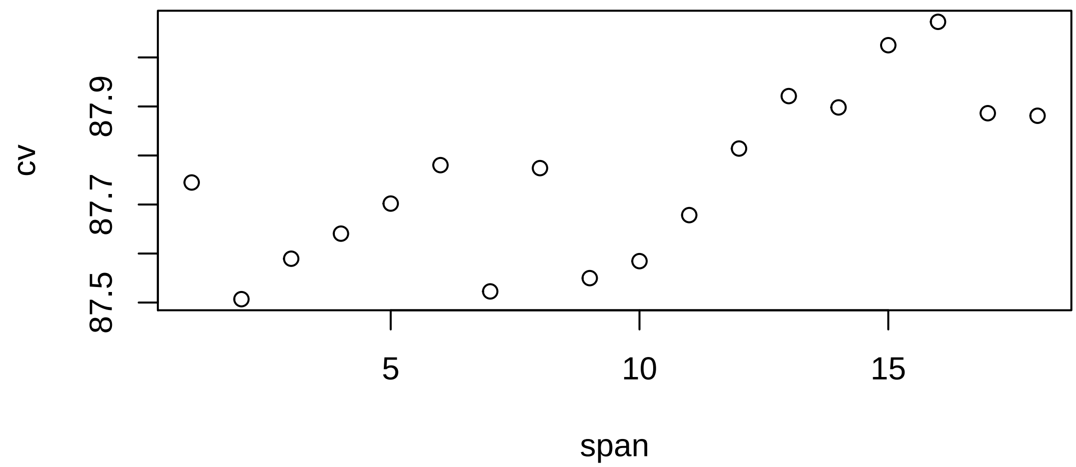
Egalitarianism and income

1.

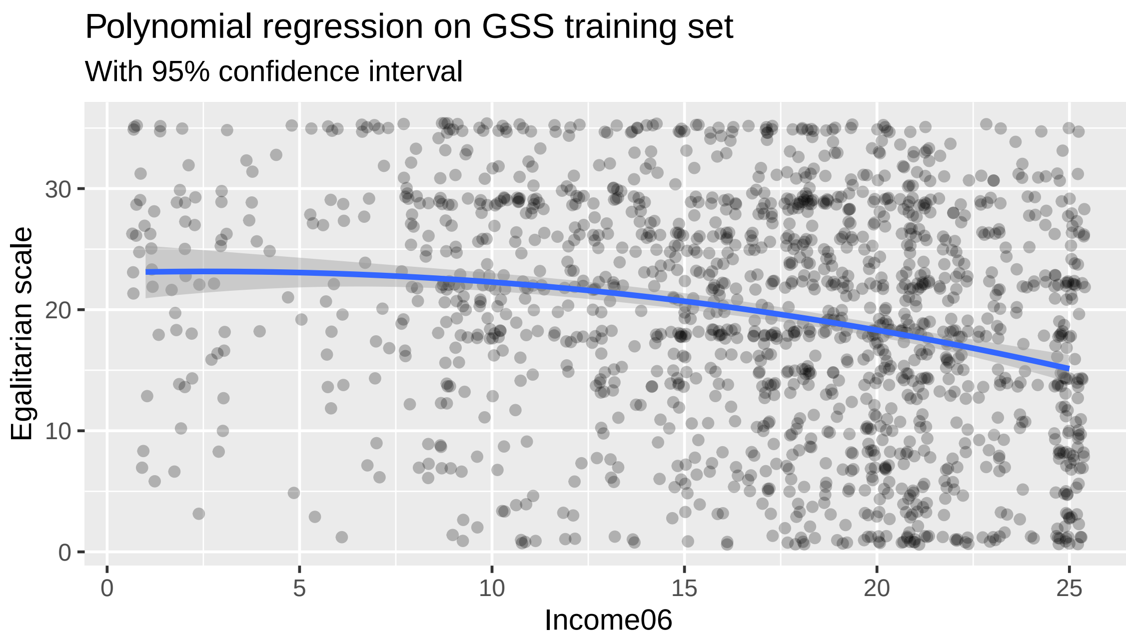
```
gss_train <- read.csv('data/gss_train.csv')
gss_test <- read.csv('data/gss_test.csv')
```

```
k <- 10
fold <- sample(k, nrow(gss_train), replace = TRUE)

## For each span from 1 to 10 we can calculate the CV test error:
mse <- numeric(k)
span <- seq(1, 18, by = 1)
cv <- numeric(length(span))

for (j in span) {
  for (i in seq_len(k)) {
    take <- fold==i
    foldi <- gss_train[take, ]
    foldOther <- gss_train[!take, ]
    f <- lm(egalit_scale ~ poly(x=income06, degree=j), data=foldOther)
    pred <- predict(f, foldi)
```

```
    mse[i] <- mean((pred - foldi$egalit_scale)^2, na.rm=TRUE)
  }
  cv[j]<- mean(mse)
}

plot(span, cv)
```



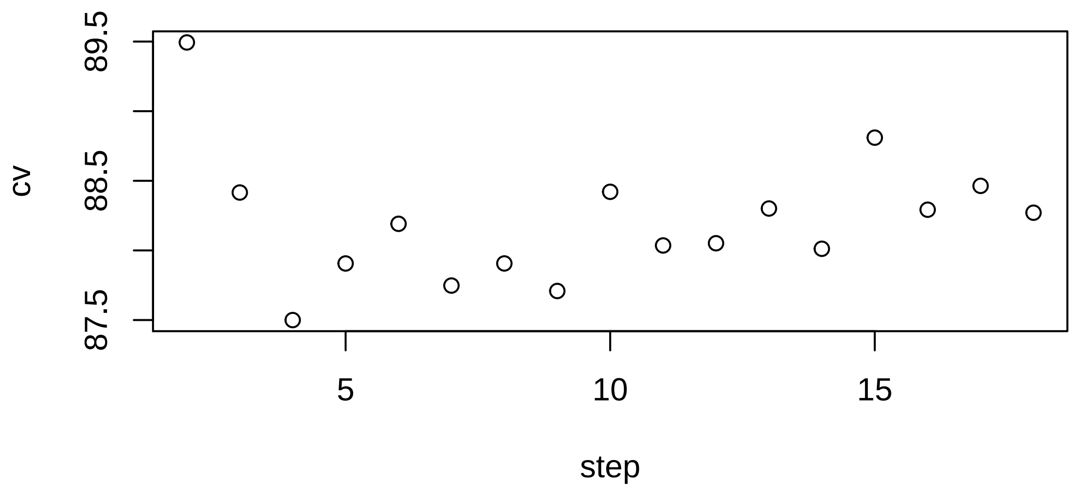

```
# plot the model results
ggplot(gss_train, aes(income06, egalit_scale)) +
  geom_jitter(alpha = .25) +
  geom_smooth(method = lm, formula = y ~ poly(x = x, degree = 2)) +
  labs(title = "Polynomial regression on GSS training set",
       subtitle = "With 95% confidence interval",
       x = "Income06",
       y = "Egalitarian scale")
```



2. Step function

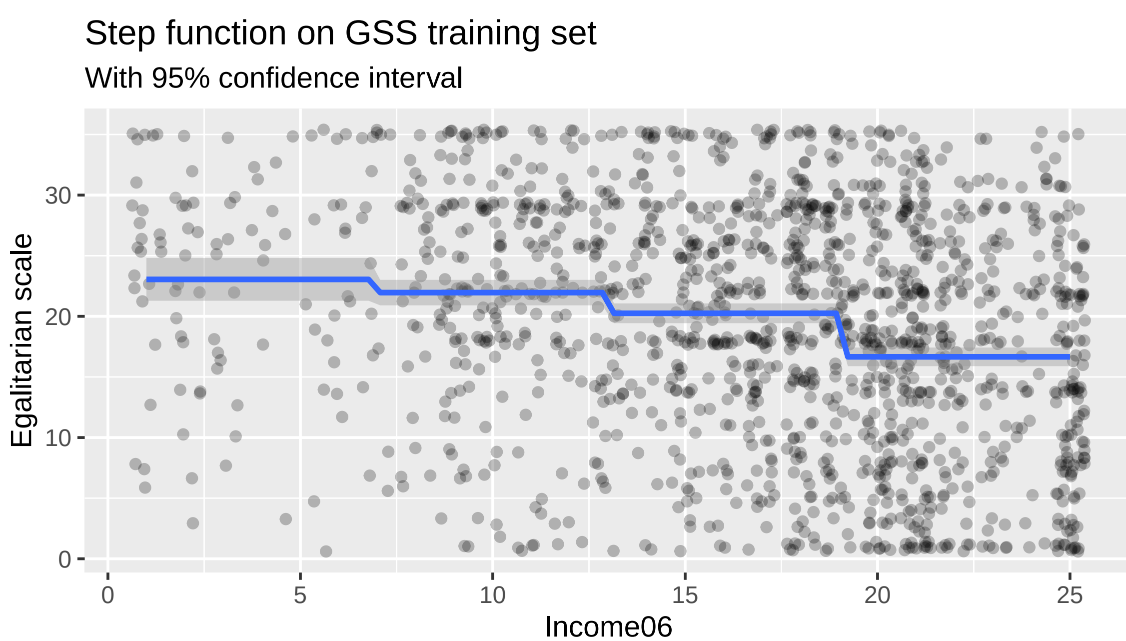
When use step function, we first have to assign each income06 a category by cut() function. Then, run similar code as above with that category variables.

```
k <- 10
fold <- sample(k, nrow(gss_train), replace = TRUE)

## For each span from 1 to 10 we can calculate the CV test error:
mse <- numeric(k)
step <- seq(2, 18, by = 1)
step.err <- rep(NA, length(step))
cv <- numeric(length(step))

for (j in step) {
  gss_train$inc_cut <- cut_interval(gss_train$income06, j)
  for (i in seq_len(k)) {
    take <- fold == i
    foldi <- gss_train[take, ]
    foldOther <- gss_train[!take, ]
    f <- lm(egalit_scale ~ inc_cut, data = foldOther)
    pred <- predict(f, foldi)
    mse[i] <- mean((pred - foldi$egalit_scale)^2, na.rm = TRUE)
  }
  cv[j-1]<- mean(mse)
}

plot(step, cv)
```



The results above show that four-break has the lowerst MSE.

```
# plot the model results
ggplot(gss_train, aes(income06, egalit_scale)) +
  geom_jitter(alpha = .25) +
  geom_smooth(method = glm, formula = y ~ cut(x = x, breaks=4)) +
  labs(title = "Step function on GSS training set",
       subtitle = "With 95% confidence interval",
       x = "Income06",
       y = "Egalitarian scale")
```

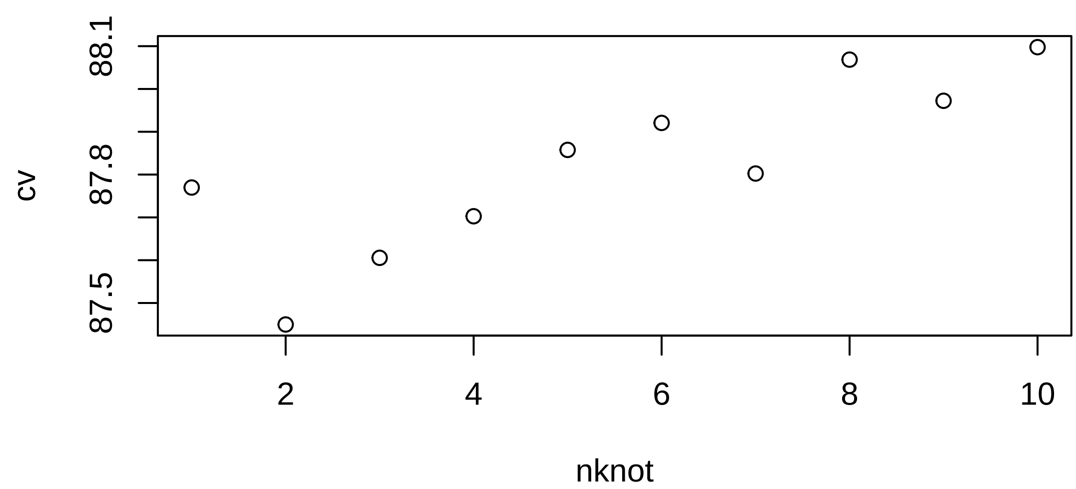


3. Spline

```
k <- 10
fold <- sample(k, nrow(gss_train), replace = TRUE)

## For each span from 1 to 10 we can calculate the CV test error:
mse <- numeric(k)
nknot <- seq(1, 10, by = 1)
cv <- numeric(length(nknot))
cv
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

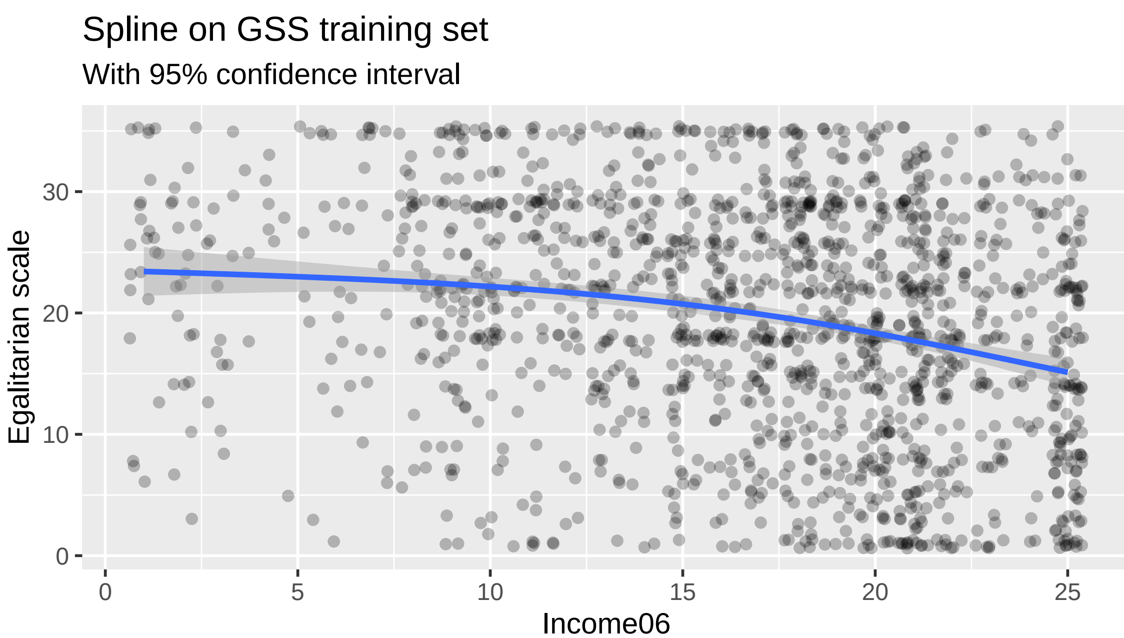
```
for (j in nknot) {
  for (i in seq_len(k)) {
    take <- fold==i
    foldi <- gss_train[take, ]
    foldOther <- gss_train[!take, ]
    f <- lm(egalit_scale ~ ns(x=income06, df=j), data=foldOther)
    pred <- predict(f, foldi)
    mse[i] <- mean((pred - foldi$egalit_scale)^2, na.rm=TRUE)
  }
  cv[j]<- mean(mse)
}

plot(nknot, cv)
```



The results above show that two-knots has the lowerst MSE.

```
# plot the model results
ggplot(gss_train, aes(income06, egalit_scale)) +
  geom_jitter(alpha = .25) +
  geom_smooth(method = glm, formula = y ~ ns(x = x, df=2)) +
  labs(title = "Spline on GSS training set",
       subtitle = "With 95% confidence interval",
```

```r
       x = "Income06",
       y = "Egalitarian scale")
```



Egalitarianism and everything

4.

Pre-preocessing

```r
# Reload the data.
gss_train <- read.csv('data/gss_train.csv')
gss_test <- read.csv('data/gss_test.csv')

# For the categorical variables, I changed the variables as follows.
#  - If the variable is binary, I assigned 0 or 1.
#  - If the variable is multi-category, I generate each columns and assigned
#    0 or 1 for each column.
#gss_train <- gss_train %>%
#  select(-c(attend, degree))
#gss_test <- gss_test %>%
#  select(-c(attend,degree))


gss_train$black <- as.integer(gss_train$black == 'Yes')
gss_train$born <- as.integer(gss_train$born == 'YES')
gss_train$colath <- as.integer(gss_train$colath == 'ALLOWED')
gss_train$colrac <- as.integer(gss_train$colrac == 'ALLOWED')
gss_train$colcom <- as.integer(gss_train$colcom == 'FIRED')
gss_train$colmil <- as.integer(gss_train$colmil == 'ALLOWED')
gss_train$colhomo <- as.integer(gss_train$colhomo == 'ALLOWED')
gss_train$colmslm <- as.integer(gss_train$colmslm == 'Yes, allowed')
gss_train$grass <- as.integer(gss_train$grass == 'LEGAL')
gss_train$hispanic_2 <- as.integer(gss_train$hispanic_2 == 'Yes')
```

```r
gss_train$mode <- as.integer(gss_train$mode == 'IN-PERSON')
gss_train$pornlaw2 <- as.integer(gss_train$pornlaw2 == 'Illegal to all')
gss_train$pres08 <- as.integer(gss_train$pres08 == 'Obama')
gss_train$reborn_r <- as.integer(gss_train$reborn_r == 'Yes')
gss_train$sex <- as.integer(gss_train$sex == 'Male')
gss_train$south <- as.integer(gss_train$south == 'South')

#tmp <- dummyVars(~., data=gss_train, sep='_')
#gss_train.dummy <- as.data.frame(predict(tmp, gss_train))

gss_test$black <- as.integer(gss_test$black == 'Yes')
gss_test$born <- as.integer(gss_test$born == 'YES')
gss_test$colath <- as.integer(gss_test$colath == 'ALLOWED')
gss_test$colrac <- as.integer(gss_test$colrac == 'ALLOWED')
gss_test$colcom <- as.integer(gss_test$colcom == 'FIRED')
gss_test$colmil <- as.integer(gss_test$colmil == 'ALLOWED')
gss_test$colhomo <- as.integer(gss_test$colhomo == 'ALLOWED')
gss_test$colmslm <- as.integer(gss_test$colmslm == 'Yes, allowed')
gss_test$grass <- as.integer(gss_test$grass == 'LEGAL')
gss_test$hispanic_2 <- as.integer(gss_test$hispanic_2 == 'Yes')
gss_test$mode <- as.integer(gss_test$mode == 'IN-PERSON')
gss_test$pornlaw2 <- as.integer(gss_test$pornlaw2 == 'Illegal to all')
gss_test$pres08 <- as.integer(gss_test$pres08 == 'Obama')
gss_test$reborn_r <- as.integer(gss_test$reborn_r == 'Yes')
gss_test$sex <- as.integer(gss_test$sex == 'Male')
gss_test$south <- as.integer(gss_test$south == 'South')

#tmp <- dummyVars(~., data=gss_test, sep='_')
#gss_test.dummy <- as.data.frame(predict(tmp, gss_test))


pgss_train <- gss_train
pgss_test <- gss_test
#pgss_train <- select_if(gss_train, is.numeric) %>%
#  merge(gss_train.dummy)
#pgss_test <- select_if(gss_test, is.numeric) %>%
#  merge(gss_test.dummy)
```

a. Linear regression

```r
options(warn=-1)
lr_model <- train(egalit_scale~., data=pgss_train, method='lm',
                  metric='RMSE', preProcess='zv',
                  trControl=trainControl(method='cv', number=10))
options(warn=1)
lr_model
```

```
## Linear Regression
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
```

```
## Summary of sample sizes: 1332, 1333, 1332, 1333, 1332, 1334, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.950629  0.3265621  6.27887
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

b. Elastic net regression

```r
options(warn=-1)
eln_model <- train(egalit_scale~., data=pgss_train,
                   method='glmnet', metric='RMSE',
                   preProcess='zv',
                   trControl=trainControl(method='cv', number=10),
                   tuneLength=10)
options(warn=1)
eln_model
```

```
## glmnet
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1334, 1333, 1333, 1333, 1333, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda       RMSE      Rsquared   MAE
##   0.1    0.002118067  7.964210  0.3227054  6.270176
##   0.1    0.004893009  7.964108  0.3227165  6.270150
##   0.1    0.011303486  7.962187  0.3229285  6.269584
##   0.1    0.026112519  7.954050  0.3238994  6.264896
##   0.1    0.060323307  7.937225  0.3259241  6.253116
##   0.1    0.139354665  7.906412  0.3296868  6.233430
##   0.1    0.321927360  7.859773  0.3357329  6.206486
##   0.1    0.743693973  7.807979  0.3435286  6.195172
##   0.1    1.718029578  7.772850  0.3535846  6.224765
##   0.1    3.968871254  7.918851  0.3481464  6.415563
##   0.2    0.002118067  7.964493  0.3227011  6.270089
##   0.2    0.004893009  7.963259  0.3228357  6.269759
##   0.2    0.011303486  7.958109  0.3234525  6.267370
##   0.2    0.026112519  7.944575  0.3251026  6.257720
##   0.2    0.060323307  7.918750  0.3283268  6.239914
##   0.2    0.139354665  7.876499  0.3337745  6.213827
##   0.2    0.321927360  7.821185  0.3414935  6.191254
##   0.2    0.743693973  7.760894  0.3524970  6.186039
##   0.2    1.718029578  7.822317  0.3519652  6.301093
##   0.2    3.968871254  8.170214  0.3200573  6.683522
##   0.3    0.002118067  7.964283  0.3227228  6.270085
##   0.3    0.004893009  7.962272  0.3229526  6.269499
##   0.3    0.011303486  7.953541  0.3240218  6.264234
##   0.3    0.026112519  7.935732  0.3262550  6.251093
##   0.3    0.060323307  7.903075  0.3303919  6.229222
```

```
##   0.3   0.139354665  7.852102  0.3372062  6.199757
##   0.3   0.321927360  7.795698  0.3455891  6.184444
##   0.3   0.743693973  7.753571  0.3557771  6.202124
##   0.3   1.718029578  7.918853  0.3418429  6.410402
##   0.3   3.968871254  8.399723  0.2898412  6.920348
##   0.4   0.002118067  7.964152  0.3227443  6.270101
##   0.4   0.004893009  7.960770  0.3231475  6.268792
##   0.4   0.011303486  7.949439  0.3245398  6.261032
##   0.4   0.026112519  7.927287  0.3273488  6.244857
##   0.4   0.060323307  7.889733  0.3321811  6.220807
##   0.4   0.139354665  7.834506  0.3397235  6.192031
##   0.4   0.321927360  7.772155  0.3497322  6.177877
##   0.4   0.743693973  7.775361  0.3544620  6.238915
##   0.4   1.718029578  8.041385  0.3253262  6.540432
##   0.4   3.968871254  8.590893  0.2629115  7.108130
##   0.5   0.002118067  7.964013  0.3227561  6.270166
##   0.5   0.004893009  7.958665  0.3234103  6.267524
##   0.5   0.011303486  7.945358  0.3250680  6.257825
##   0.5   0.026112519  7.919428  0.3283626  6.239372
##   0.5   0.060323307  7.876866  0.3339427  6.212729
##   0.5   0.139354665  7.819576  0.3419329  6.187203
##   0.5   0.321927360  7.754575  0.3531322  6.175221
##   0.5   0.743693973  7.809022  0.3511330  6.279163
##   0.5   1.718029578  8.161201  0.3077417  6.671414
##   0.5   3.968871254  8.736407  0.2437572  7.253704
##   0.6   0.002118067  7.962947  0.3228824  6.269682
##   0.6   0.004893009  7.956698  0.3236550  6.266234
##   0.6   0.011303486  7.941449  0.3255680  6.254825
##   0.6   0.026112519  7.912391  0.3292771  6.234697
##   0.6   0.060323307  7.865135  0.3355606  6.205812
##   0.6   0.139354665  7.808153  0.3436399  6.184418
##   0.6   0.321927360  7.746313  0.3551706  6.178455
##   0.6   0.743693973  7.851734  0.3462233  6.328907
##   0.6   1.718029578  8.258841  0.2937440  6.777200
##   0.6   3.968871254  8.840007  0.2400358  7.360452
##   0.7   0.002118067  7.962623  0.3229254  6.269643
##   0.7   0.004893009  7.954571  0.3239198  6.264771
##   0.7   0.011303486  7.937593  0.3260775  6.251935
##   0.7   0.026112519  7.905653  0.3301664  6.230157
##   0.7   0.060323307  7.855139  0.3369516  6.200263
##   0.7   0.139354665  7.798140  0.3452160  6.181743
##   0.7   0.321927360  7.746795  0.3558639  6.187810
##   0.7   0.743693973  7.900496  0.3399965  6.382939
##   0.7   1.718029578  8.343274  0.2817445  6.860606
##   0.7   3.968871254  8.958914  0.2318815  7.473825
##   0.8   0.002118067  7.962089  0.3229951  6.269422
##   0.8   0.004893009  7.952669  0.3241586  6.263355
##   0.8   0.011303486  7.933487  0.3266104  6.248938
##   0.8   0.026112519  7.899410  0.3309954  6.226050
##   0.8   0.060323307  7.846495  0.3381632  6.195556
##   0.8   0.139354665  7.787850  0.3469284  6.179364
##   0.8   0.321927360  7.754126  0.3554767  6.201892
##   0.8   0.743693973  7.955834  0.3321369  6.442067
##   0.8   1.718029578  8.423688  0.2698657  6.936682
```

```
##   0.8    3.968871254  9.070438  0.2298209  7.572827
##   0.9    0.002118067  7.961513  0.3230675  6.269122
##   0.9    0.004893009  7.950888  0.3243823  6.261943
##   0.9    0.011303486  7.929835  0.3270797  6.246232
##   0.9    0.026112519  7.893833  0.3317365  6.222603
##   0.9    0.060323307  7.838805  0.3392474  6.192293
##   0.9    0.139354665  7.777453  0.3487109  6.176736
##   0.9    0.321927360  7.764354  0.3546474  6.218821
##   0.9    0.743693973  8.011733  0.3238545  6.503062
##   0.9    1.718029578  8.502223  0.2571182  7.011716
##   0.9    3.968871254  9.194927  0.2298203  7.676352
##   1.0    0.002118067  7.960752  0.3231624  6.268696
##   1.0    0.004893009  7.949135  0.3246063  6.260591
##   1.0    0.011303486  7.926367  0.3275274  6.243762
##   1.0    0.026112519  7.887963  0.3325316  6.219029
##   1.0    0.060323307  7.831583  0.3402814  6.189551
##   1.0    0.139354665  7.767485  0.3504835  6.173851
##   1.0    0.321927360  7.776475  0.3535293  6.235748
##   1.0    0.743693973  8.065744  0.3156565  6.562986
##   1.0    1.718029578  8.573670  0.2450649  7.084794
##   1.0    3.968871254  9.341005  0.2298203  7.813151
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.6 and lambda = 0.3219274.
```

c. Principal component regression

```r
options(warn=-1)
pcr_model <- train(egalit_scale~., data=pgss_train,
                   method='pcr', metric='RMSE',
                   preProcess='zv',
                   trControl=trainControl(method='cv', number=10),
                   tuneLength=20)
options(warn=1)
pcr_model
```

```
## Principal Component Analysis
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1333, 1333, 1334, 1332, 1333, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared    MAE
##    1     9.525899  0.02721140  7.931877
##    2     9.228729  0.08572719  7.647135
##    3     9.230454  0.08528292  7.649239
##    4     9.203476  0.09024627  7.646430
##    5     9.209540  0.08877866  7.652256
##    6     9.168751  0.09734971  7.602856
##    7     9.142704  0.10294387  7.561974
##    8     9.152719  0.10191040  7.565761
```

```
##    9     9.092708  0.11553771  7.527345
##   10     9.093087  0.11476262  7.517855
##   11     8.984593  0.13244848  7.414024
##   12     8.752982  0.17692173  7.219303
##   13     8.424912  0.23518738  6.794826
##   14     8.426466  0.23496855  6.798845
##   15     8.337500  0.25214177  6.710089
##   16     8.326139  0.25432049  6.697809
##   17     8.330039  0.25344169  6.699085
##   18     8.325895  0.25483803  6.687950
##   19     8.302115  0.25895992  6.681031
##   20     8.260761  0.26582062  6.631605
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 20.
```

d. Partial least squares regression

```r
options(warn=-1)
pls_model <- train(egalit_scale~., data=pgss_train,
                   method='pls', metric='RMSE',
                   preProcess='zv',
                   trControl=trainControl(method='cv', number=10),
                   tuneLength=10)
options(warn=1)
pls_model
```

```
## Partial Least Squares
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1333, 1333, 1332, 1334, 1332, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared    MAE
##    1     9.435365  0.04226966  7.840704
##    2     9.150064  0.10181908  7.581108
##    3     8.809655  0.16440627  7.242349
##    4     8.407210  0.23970653  6.795827
##    5     8.234429  0.27302298  6.563375
##    6     8.102314  0.29626119  6.455934
##    7     8.012863  0.31040602  6.380223
##    8     7.946237  0.32301173  6.294170
##    9     7.935873  0.32413685  6.299611
##   10     7.932951  0.32515810  6.292732
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 10.
```

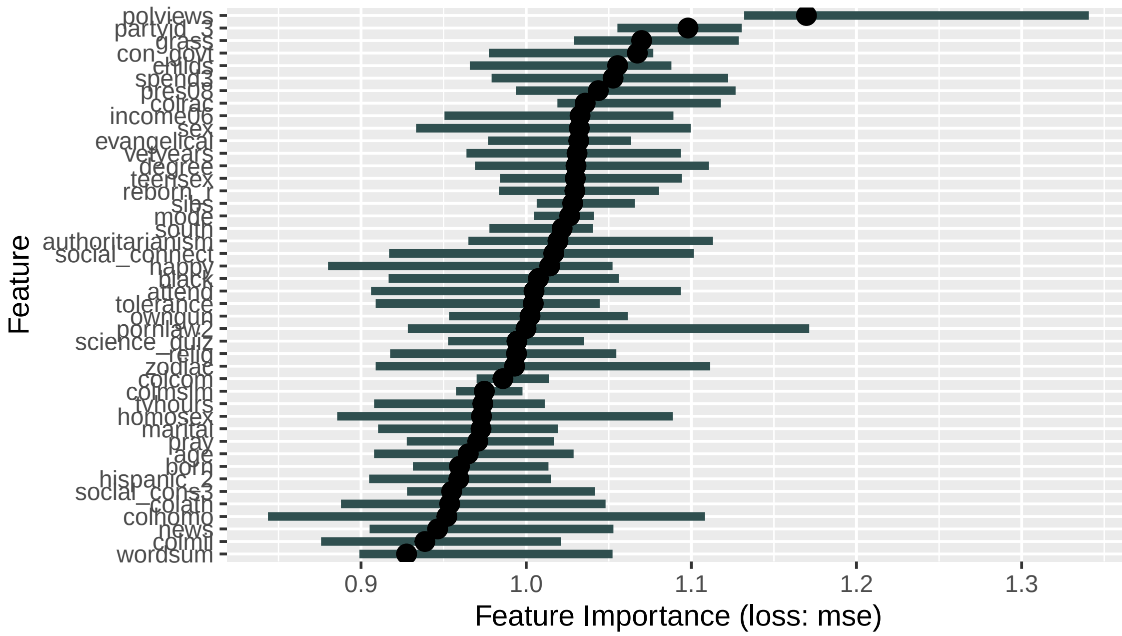
```r
res <- list(Linear_Regression = lr_model,
            Elastic_Net = eln_model,
            Principal_Component = pcr_model,
```

```r
            Partial_Least_Squares = pls_model) %>%
  resamples %>%
  summary

res$statistics$RMSE^2
```

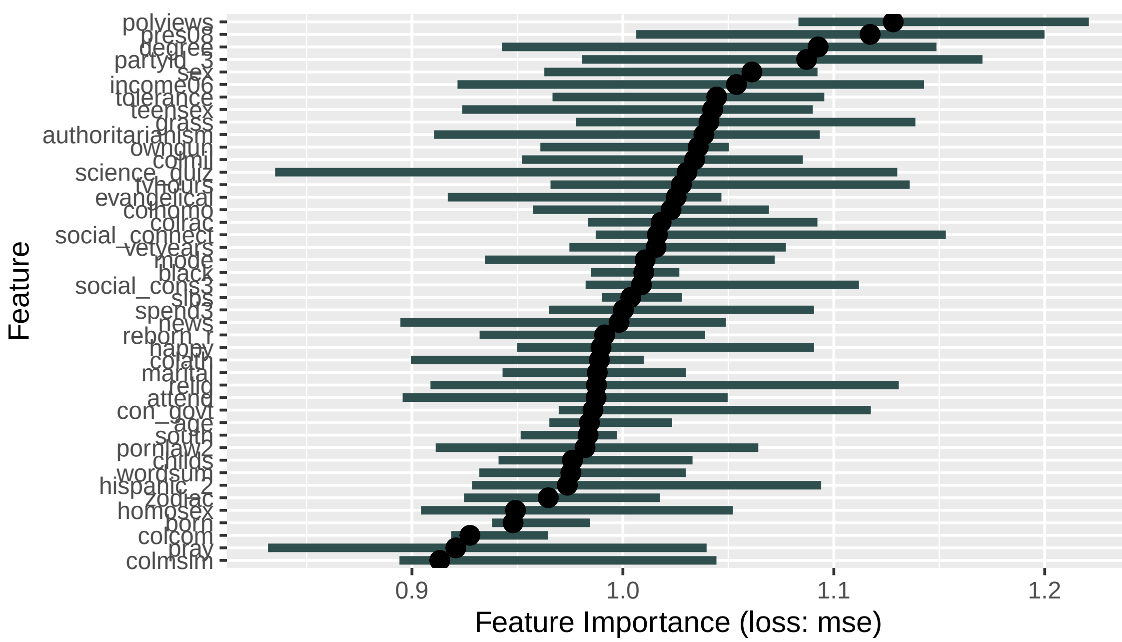
```
##                             Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## Linear_Regression     51.75887 58.07882 64.04166 63.21251 67.07119 82.98490
## Elastic_Net           47.20925 56.39438 60.25883 60.00536 64.73023 73.12946
## Principal_Component   57.10472 65.45311 68.21701 68.24017 72.79180 75.61596
## Partial_Least_Squares 56.14826 61.19333 62.45039 62.93172 65.86019 72.00093
##                       NA's
## Linear_Regression        0
## Elastic_Net              0
## Principal_Component      0
## Partial_Least_Squares    0
```

5.

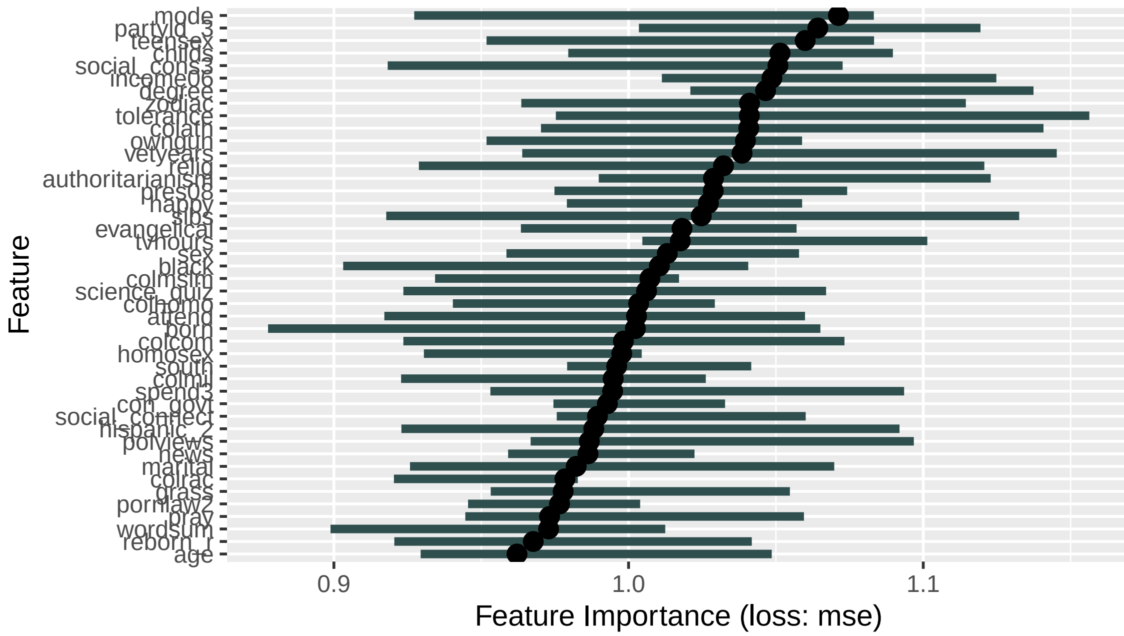
```r
library(iml)
pgss_test_x = select(pgss_test, -egalit_scale)
pgss_test_y = pgss_test$egalit_scale

lr_pred <- Predictor$new(model=lr_model,
                         data=pgss_test_x,
                         y=pgss_test_y)
eln_pred <- Predictor$new(model=eln_model,
                         data=pgss_test_x,
                         y=pgss_test_y)
pcr_pred <- Predictor$new(model=pcr_model,
                         data=pgss_test_x,
                         y=pgss_test_y)
pls_pred <- Predictor$new(model=pls_model,
                         data=pgss_test_x,
                         y=pgss_test_y)


lr_fea <- FeatureImp$new(lr_pred, loss='mse')
eln_fea <- FeatureImp$new(eln_pred, loss='mse')
pcr_fea <- FeatureImp$new(pcr_pred, loss='mse')
pls_fea <- FeatureImp$new(pls_pred, loss='mse')

plot(lr_fea)
```



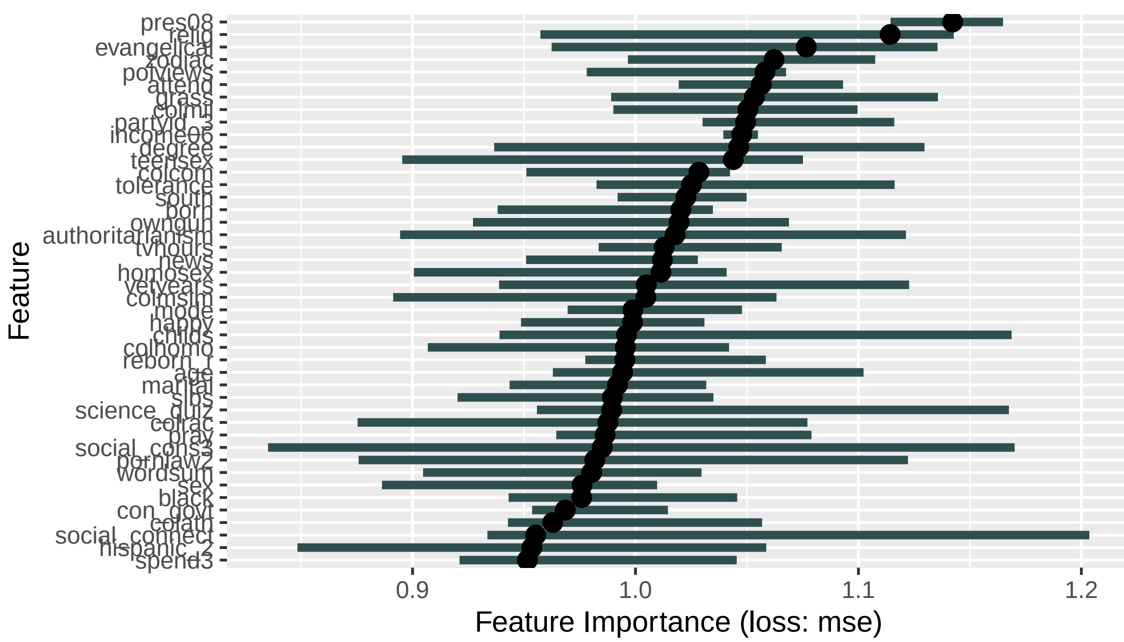

```r
plot(eln_fea)
```




```r
plot(pcr_fea)
```


Feature
mode
partyid_3
teensex
childs
social_cons3
income06
degree
zodiac
tolerance
colath
owngun
vetyears
relig
authoritarianism
pres08
happy
sibs
evangelical
tvhours
sex
black
colmslm
science_quiz
colhomo
attend
born
colcom
homosex
south
colmil
spend3
con_govt
social_connect
hispanic_2
polviews
news
marital
colrac
grass
pornlaw2
pray
wordsum
reborn_r
age
0.9
1.0
1.1
Feature Importance (loss: mse)

```
plot(pls_fea)
```


Feature
pres08
relig
evangelical
zodiac
polviews
attend
grass
colmil
partyid_3
income06
degree
teensex
colcom
tolerance
south
born
owngun
authoritarianism
tvhours
news
homosex
vetyears
colmslm
mode
happy
childs
colhomo
reborn_r
age
marital
sibs
science_quiz
colrac
pray
social_cons3
pornlaw2
wordsum
sex
black
con_govt
colath
social_connect
hispanic_2
spend3
0.9
1.0
1.1
1.2
Feature Importance (loss: mse)