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)
library(iml)
library(margins)
rm(list=ls())
set.seed(1100)
```

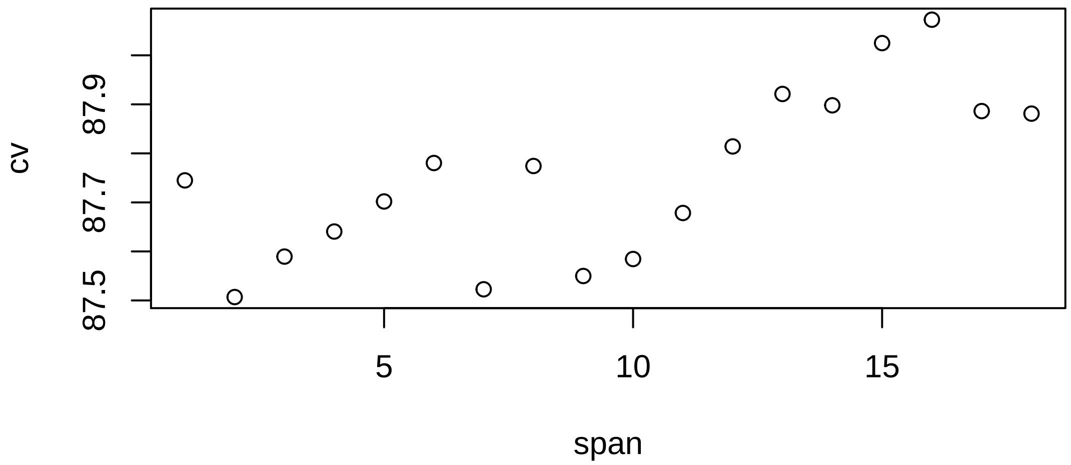
Non-linear regression

Egalitarianism and income

1.

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

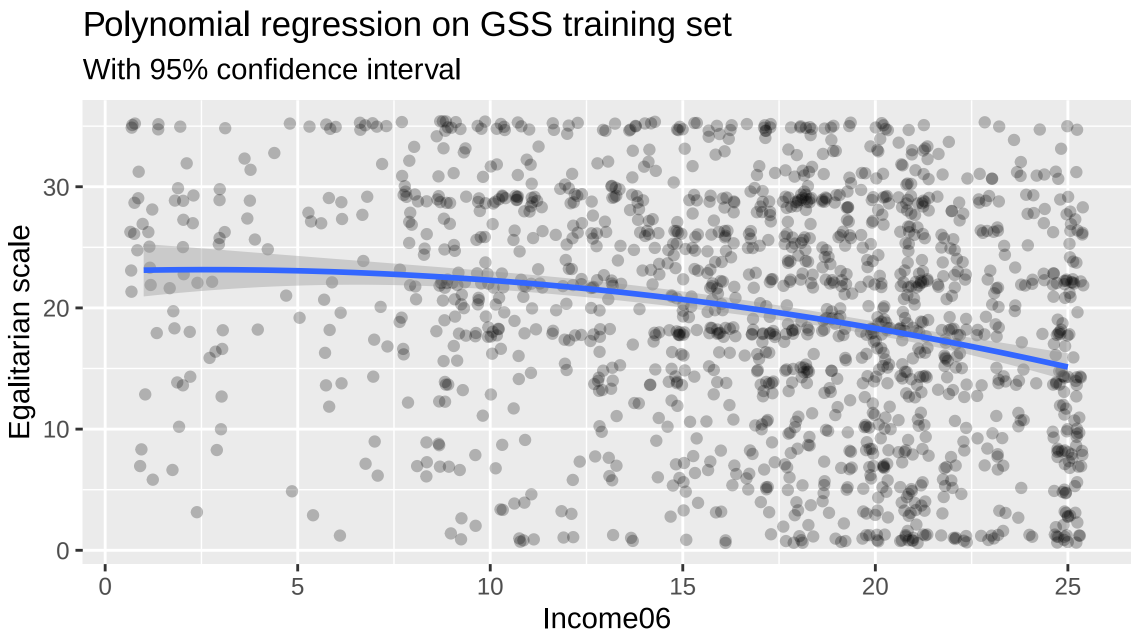
```
k <- 10
fold <- sample(k, nrow(gss_train), replace = TRUE)

## 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)
```



Above 10-fold cross-validation plot shows degree of 2 yields the lowest MSE. This result is susceptible to the random choice of cv; when I run this several times, the result might change. However, there is a strong tendency that MSE increases when the degree increases. This is due to the overfitting.

The next plot shows the resulting polynomial fit to the data with degree of 2 and also the jittering points of training data set.

```
# 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")
```

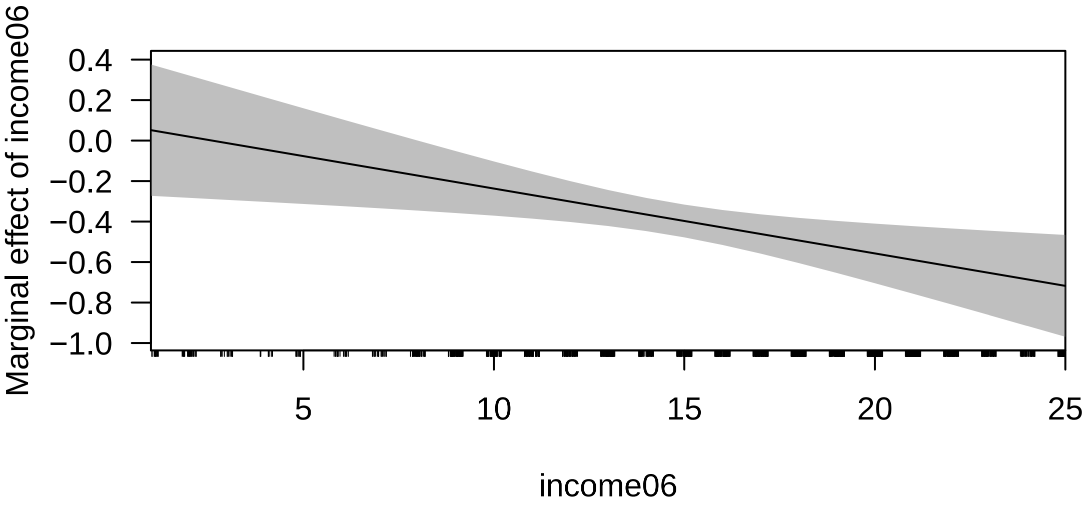


The next plot shows the average marginal effect (AME) of income06 across its potential values.

```
poly_model <- lm(egalit_scale ~ stats::poly(income06, 2), data=gss_train)
summary(margins(poly_model))
```

factor	AME	SE	z	p	lower	upper
income06	-0.4507319	0.047479	-9.49329	0	-0.543789	-0.3576748

```
cplot(poly_model, "income06", what="effect")
```



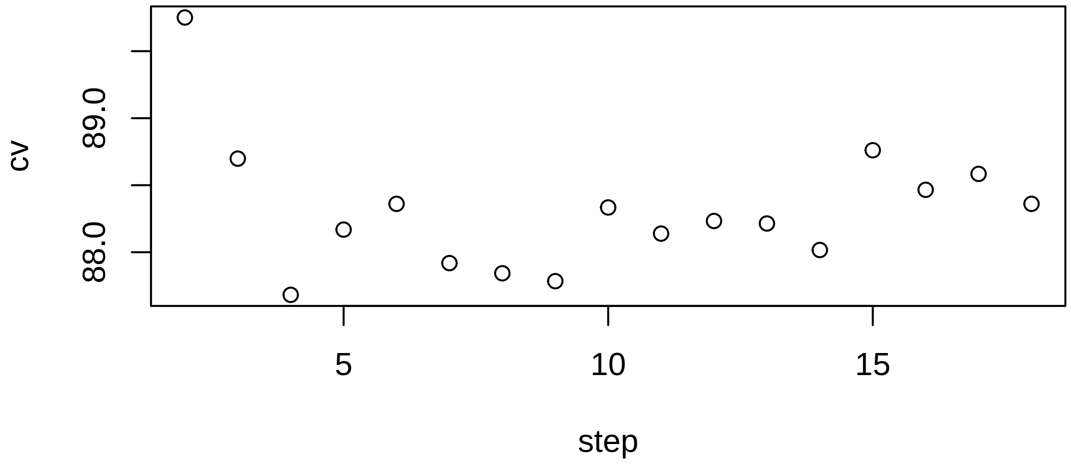
This plot shows that the marginal effect of income06 decreases when income06 increases. Also, most of the

AME are below 0, meaning that egalit_scale has a declining trend on income06. On average, one increase of income06 decreases egalit_scale by -0.4507

2. Step function

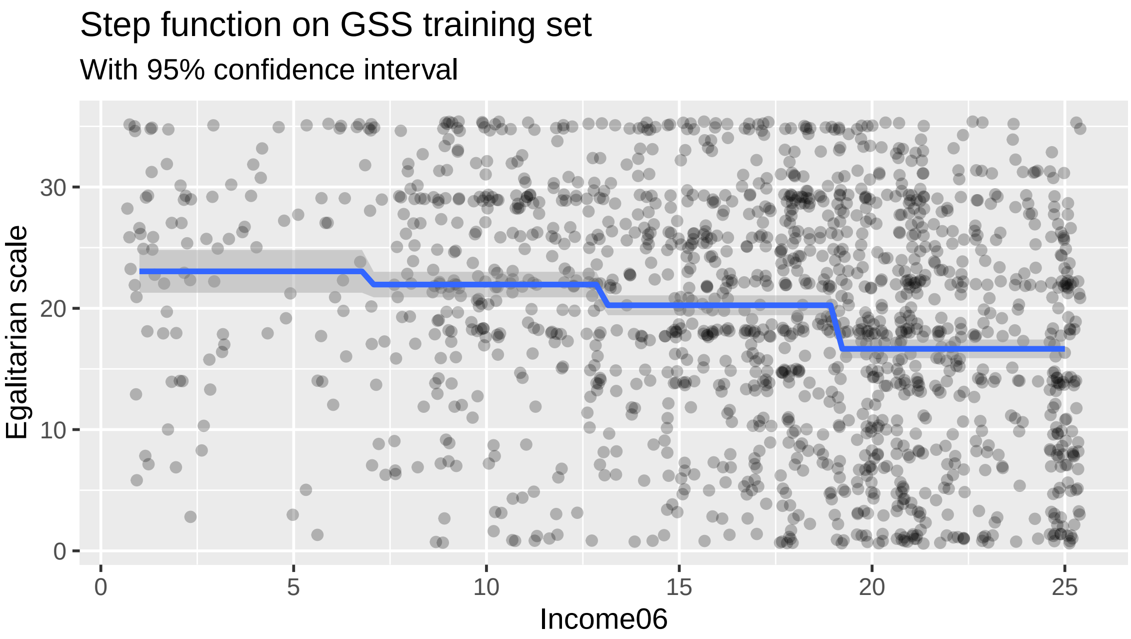
When use step function, we first have to assign each income06 a category by cut() function. Then, run similar code as above with that category variables.

```r
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 4-break has the lowest MSE.

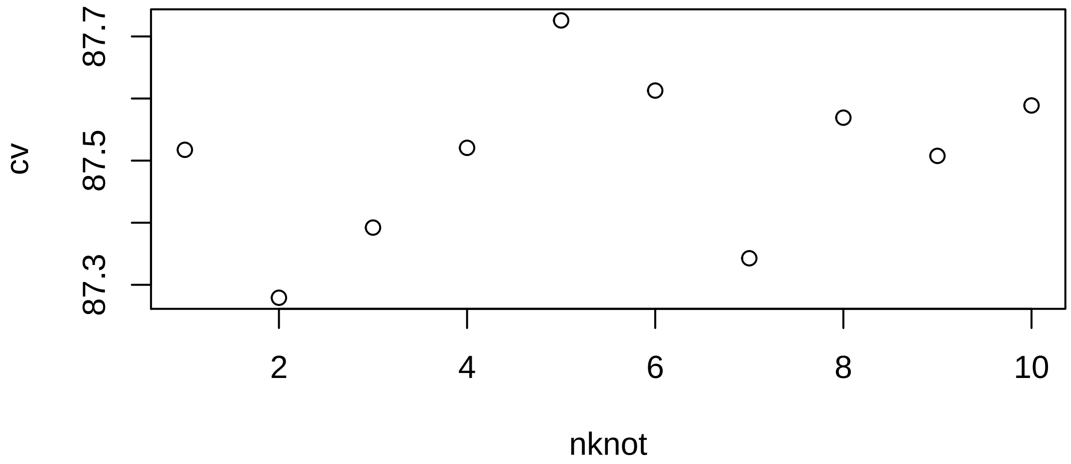
```
# plot the model results
ggplot(gss_train, aes(income06, egalit_scale)) +
  geom_jitter(alpha = .25) +
  geom_smooth(method = glm, formula = y ~ cut(x = x, breaks=4)) +
  labs(title = "Step function on GSS training set",
       subtitle = "With 95% confidence interval",
       x = "Income06",
  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



The fit plot above shows that higher incoem06 group has lower egalit_scale. Also, it suggests that the highest income06 group has statistically lower egalit_scale than the other groups. Also, the lowest income06 group has high variance due to the small sample size in that group. This would lowering the prediction accuracy in that group.

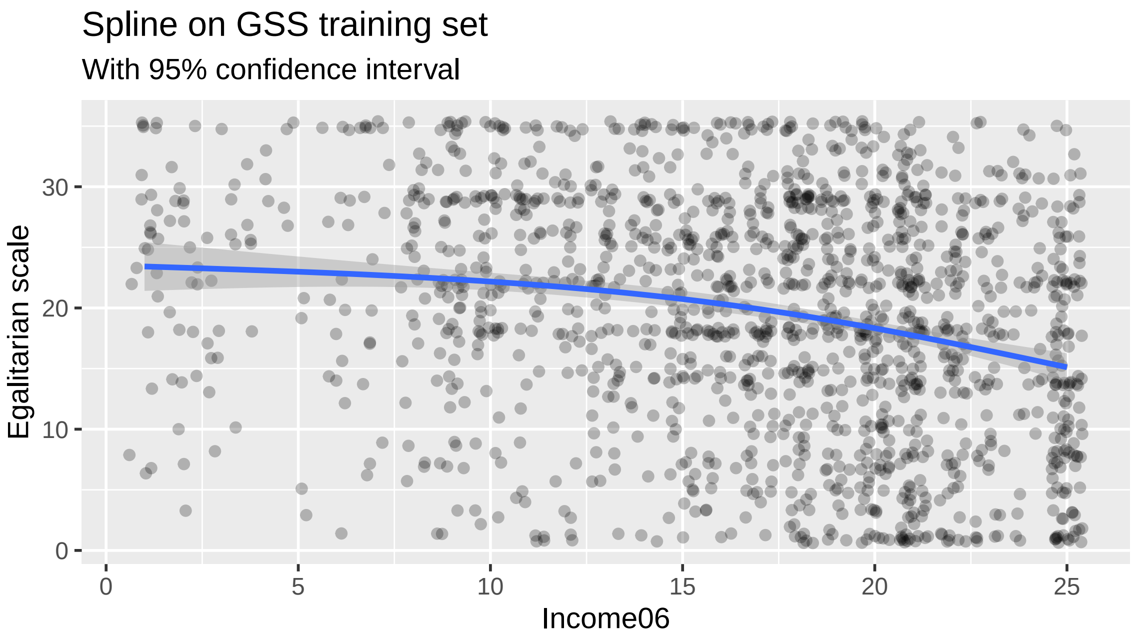
3. Spline

```
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))

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 2-knots has the lowest 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",
       x = "Income06",
       y = "Egalitarian scale")
```



Since it has only two knots, the line resembles to that of the polynomial regression. We can observe the same properties as (1).

Egalitarianism and everything

4.

Pre-processing

```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$black <- as.integer(gss_train$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')
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')
```

```r
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')

pgss_train <- gss_train
pgss_test <- gss_test
```

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: 1333, 1333, 1333, 1333, 1332, 1334, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.939936  0.3267291  6.277334
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

With linear regression, the RMSE is 7.923 and R-squared is 0.332

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)
```

```r
options(warn=1)
eln_model
```

```
## glmnet
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1333, 1334, 1332, 1333, 1333, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda       RMSE      Rsquared   MAE
##   0.1    0.002118067  7.974377  0.3231938  6.296325
##   0.1    0.004893009  7.974294  0.3232047  6.296281
##   0.1    0.011303486  7.972000  0.3235351  6.294859
##   0.1    0.026112519  7.962817  0.3246891  6.288753
##   0.1    0.060323307  7.945905  0.3267625  6.278295
##   0.1    0.139354665  7.916021  0.3304164  6.259951
##   0.1    0.321927360  7.866689  0.3367833  6.231687
##   0.1    0.743693973  7.815772  0.3441974  6.218941
##   0.1    1.718029578  7.776486  0.3553251  6.239966
##   0.1    3.968871254  7.916904  0.3507962  6.423568
##   0.2    0.002118067  7.974546  0.3232051  6.296193
##   0.2    0.004893009  7.973208  0.3233944  6.295393
##   0.2    0.011303486  7.967168  0.3241795  6.291180
##   0.2    0.026112519  7.952842  0.3259973  6.282258
##   0.2    0.060323307  7.928750  0.3290006  6.266744
##   0.2    0.139354665  7.884593  0.3346736  6.239008
##   0.2    0.321927360  7.826111  0.3426891  6.211108
##   0.2    0.743693973  7.765334  0.3538687  6.203375
##   0.2    1.718029578  7.820943  0.3544022  6.310306
##   0.2    3.968871254  8.167771  0.3214559  6.690823
##   0.3    0.002118067  7.974301  0.3232403  6.295831
##   0.3    0.004893009  7.971922  0.3235825  6.294334
##   0.3    0.011303486  7.962363  0.3248066  6.288117
##   0.3    0.026112519  7.944370  0.3271040  6.276572
##   0.3    0.060323307  7.913133  0.3310439  6.256122
##   0.3    0.139354665  7.857912  0.3383915  6.222453
##   0.3    0.321927360  7.799573  0.3468765  6.203097
##   0.3    0.743693973  7.748175  0.3589564  6.209894
##   0.3    1.718029578  7.920003  0.3430554  6.420994
##   0.3    3.968871254  8.394315  0.2914921  6.923154
##   0.4    0.002118067  7.973835  0.3233103  6.295543
##   0.4    0.004893009  7.970171  0.3238166  6.292987
##   0.4    0.011303486  7.958122  0.3253555  6.285319
##   0.4    0.026112519  7.936886  0.3280724  6.271434
##   0.4    0.060323307  7.898747  0.3329674  6.246589
##   0.4    0.139354665  7.838372  0.3411440  6.212001
##   0.4    0.321927360  7.774399  0.3513158  6.194095
##   0.4    0.743693973  7.773168  0.3569603  6.245421
##   0.4    1.718029578  8.042620  0.3257688  6.554805
##   0.4    3.968871254  8.590188  0.2623748  7.112356
##   0.5   0.002118067  7.973395  0.3233764  6.295256
##   0.5   0.004893009  7.967709  0.3241347  6.291350
##   0.5   0.011303486  7.953447  0.3259795  6.282161
##   0.5   0.026112519  7.929705  0.3290003  6.266450
##   0.5   0.060323307  7.884744  0.3348702  6.237180
##   0.5   0.139354665  7.823423  0.3432767  6.205480
##   0.5   0.321927360  7.755721  0.3549889  6.188049
##   0.5   0.743693973  7.809360  0.3529988  6.287738
##   0.5   1.718029578  8.158322  0.3084190  6.680476
##   0.5   3.968871254  8.738797  0.2422273  7.258537
##   0.6   0.002118067  7.972848  0.3234565  6.294829
##   0.6   0.004893009  7.965347  0.3244477  6.289869
##   0.6   0.011303486  7.949680  0.3264710  6.279650
##   0.6   0.026112519  7.922499  0.3299404  6.261492
##   0.6   0.060323307  7.872395  0.3365521  6.229491
##   0.6   0.139354665  7.811084  0.3451192  6.201296
##   0.6   0.321927360  7.744038  0.3576596  6.188007
##   0.6   0.743693973  7.852865  0.3475145  6.338898
##   0.6   1.718029578  8.253087  0.2949351  6.781412
##   0.6   3.968871254  8.842802  0.2383921  7.364443
##   0.7   0.002118067  7.971959  0.3235614  6.294225
##   0.7   0.004893009  7.963322  0.3247108  6.288554
##   0.7   0.011303486  7.946003  0.3269487  6.277304
##   0.7   0.026112519  7.915918  0.3307927  6.257103
##   0.7   0.060323307  7.860948  0.3381356  6.222285
##   0.7   0.139354665  7.799555  0.3469234  6.197540
##   0.7   0.321927360  7.740160  0.3591552  6.193936
##   0.7   0.743693973  7.904294  0.3403567  6.394905
##   0.7   1.718029578  8.339702  0.2823604  6.865586
##   0.7   3.968871254  8.959656  0.2313305  7.475713
##   0.8   0.002118067  7.971397  0.3236404  6.293887
##   0.8   0.004893009  7.961478  0.3249448  6.287363
##   0.8   0.011303486  7.942567  0.3273961  6.274940
##   0.8   0.026112519  7.909505  0.3316344  6.252837
##   0.8   0.060323307  7.851078  0.3395114  6.216391
##   0.8   0.139354665  7.789261  0.3486053  6.195628
##   0.8   0.321927360  7.746392  0.3589458  6.206630
##   0.8   0.743693973  7.959150  0.3323356  6.455640
##   0.8   1.718029578  8.423917  0.2691716  6.942053
##   0.8   3.968871254  9.069010  0.2306229  7.572923
##   0.9   0.002118067  7.970717  0.3237384  6.293313
##   0.9   0.004893009  7.959667  0.3251796  6.286157
##   0.9   0.011303486  7.939377  0.3278080  6.272758
##   0.9   0.026112519  7.903065  0.3324953  6.248498
##   0.9   0.060323307  7.842583  0.3406903  6.211938
##   0.9   0.139354665  7.777894  0.3505599  6.191327
##   0.9   0.321927360  7.759100  0.3576277  6.223506
##   0.9   0.743693973  8.014979  0.3237025  6.518699
##   0.9   1.718029578  8.505854  0.2555583  7.019871
##   0.9   3.968871254  9.193587  0.2306281  7.676318
##   1.0   0.002118067  7.969835  0.3238586  6.292692
##   1.0   0.004893009  7.957611  0.3254535  6.284810
##   1.0   0.011303486  7.936195  0.3282159  6.270573
##   1.0   0.026112519  7.896776  0.3333433  6.244252
##   1.0    0.060323307  7.834961  0.3417455  6.208272
##   1.0    0.139354665  7.767513  0.3524230  6.186476
##   1.0    0.321927360  7.774773  0.3558061  6.242308
##   1.0    0.743693973  8.067361  0.3154963  6.578103
##   1.0    1.718029578  8.577450  0.2437046  7.091572
##   1.0    3.968871254  9.339766  0.2306281  7.812059
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.7 and lambda = 0.3219274.
```

With elastic net regression, the lowest RMSE is 7.744 and R-squared is 0.358 when alpha is 0.6 and lambda is 0.322

c. Principal component regression

```
options(warn=-1)
pcr_model <- train(egalit_scale~., data=pgss_train,
                   method='pcr', metric='RMSE',
                   preProcess='zv',
                   trControl=trainControl(method='cv', number=10),
                   tuneLength=30)
options(warn=1)
pcr_model
```

```
## Principal Component Analysis
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1332, 1332, 1333, 1334, 1333, 1333, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared    MAE
##    1     9.517696  0.02519338  7.927240
##    2     9.222561  0.08754259  7.647983
##    3     9.229796  0.08600651  7.654838
##    4     9.210694  0.08932142  7.657699
##    5     9.212685  0.08882465  7.659698
##    6     9.166695  0.09815247  7.604822
##    7     9.135766  0.10336869  7.556764
##    8     9.138062  0.10282694  7.559692
##    9     9.109567  0.10931999  7.549724
##   10     9.106051  0.11100937  7.536400
##   11     8.995659  0.13276852  7.424423
##   12     8.748090  0.17959043  7.217273
##   13     8.441168  0.23676583  6.807480
##   14     8.442047  0.23673522  6.807955
##   15     8.360418  0.25253129  6.734868
##   16     8.352685  0.25384451  6.721692
##   17     8.352999  0.25387647  6.716122
##   18     8.343319  0.25521038  6.705622
##   19     8.307144  0.26146736  6.686754
##   20     8.267716  0.26800421  6.638650
##   21     8.201317  0.27911879  6.583222
##   22     8.200290  0.27909528  6.564241
##   23     8.159652  0.28545625  6.538609
##   24     8.132790  0.28950805  6.504752
##   25     8.133024  0.28955147  6.507078
##   26     8.116959  0.29239277  6.483244
##   27     8.100463  0.29553313  6.472440
##   28     8.101787  0.29575493  6.469279
##   29     8.048628  0.30471020  6.415037
##   30     8.051843  0.30441179  6.414292
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 29.
```

With principal component regression, the lowest RMSE is 8.049 and R-squared is 0.305 when the number of component is 29.

d. Partial least squares regression

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

```
## Partial Least Squares
##
## 1481 samples
##   44 predictor
##
## Pre-processing:  (None)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1334, 1333, 1332, 1333, 1333, 1332, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE      Rsquared    MAE
##    1     9.436809  0.04018549  7.840085
##    2     9.158901  0.09567850  7.581239
##    3     8.845914  0.15844725  7.268008
##    4     8.436468  0.23756739  6.803926
##    5     8.238178  0.27147160  6.585812
##    6     8.085391  0.29884756  6.443828
##    7     8.010792  0.31138249  6.363087
##    8     7.938723  0.32432084  6.304226
##    9     7.933378  0.32723654  6.278954
##   10     7.930422  0.32831475  6.288807
##   11     7.939757  0.32781600  6.300926
##   12     7.956360  0.32584540  6.323882
##   13     7.974440  0.32390854  6.318715
##   14     7.994415  0.32143799  6.323090
##   15     8.033088  0.31614307  6.352912
##   16     8.026877  0.31816668  6.342698
##   17     8.016840  0.32010404  6.328898
##   18     7.995132  0.32375121  6.311505
##   19     7.983230  0.32539229  6.293195
##   20     7.984817  0.32516034  6.294217
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 10.
```

With partial squares regression, the RMSE is 7.930 and R-squared is 0.328 when the number of component is 10.

The results above show that the Elastic Net regression yields the lowest MSE.

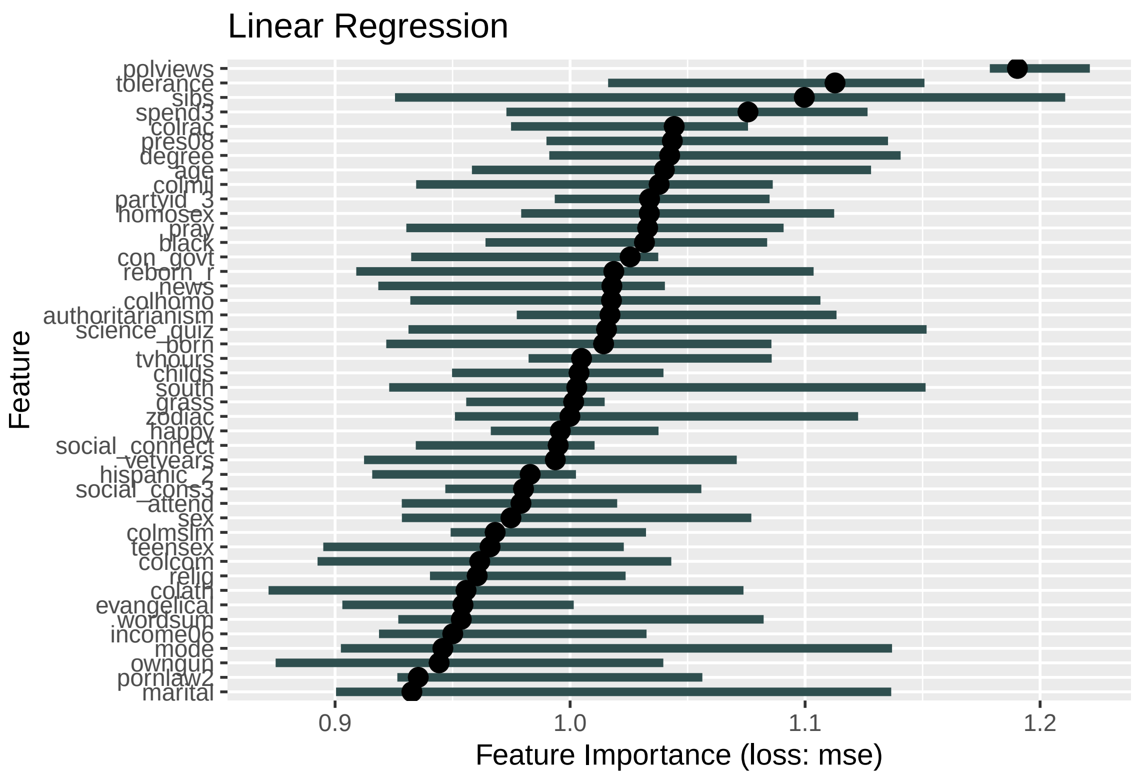
5.

Followings are some findings from the feature importance plots below.

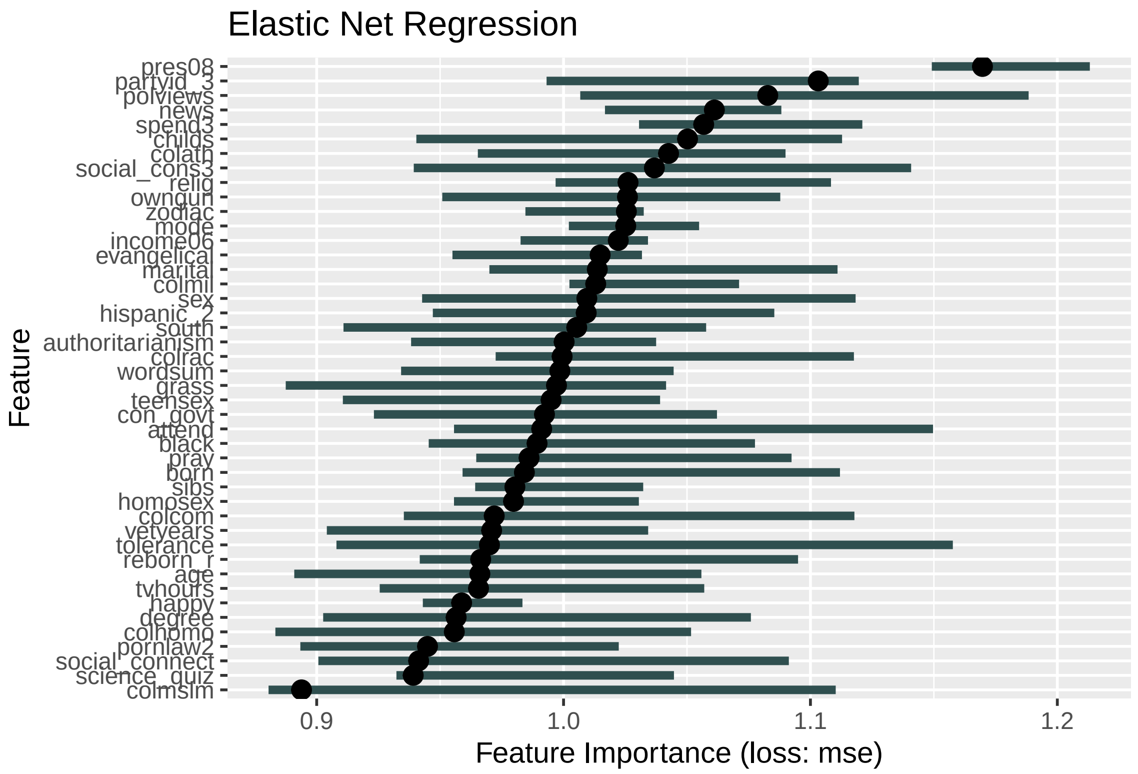
- The order of features according to feature importance is different among models. For example, income06 is relatively important feature in Elastic Net, PCR, and PLS, it is less important in Linear regression.
- Most of the feature importance is around 0.9 ~ 1.1, except pres08 and polviews. This means there is not outstanding feature for prediction in this dataset except pres08 and polviews.
- pres08 and polviews have high feature importance in every model. pres08 contains whether *Vote obama or mccain*, and polviews is a question of political views, from Extremely liberal to Extremely conservative. These two variables, potentially having high correlations, suggest that political view point or affiliation would be a good predictor for egalitarianism.

```r
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')
```

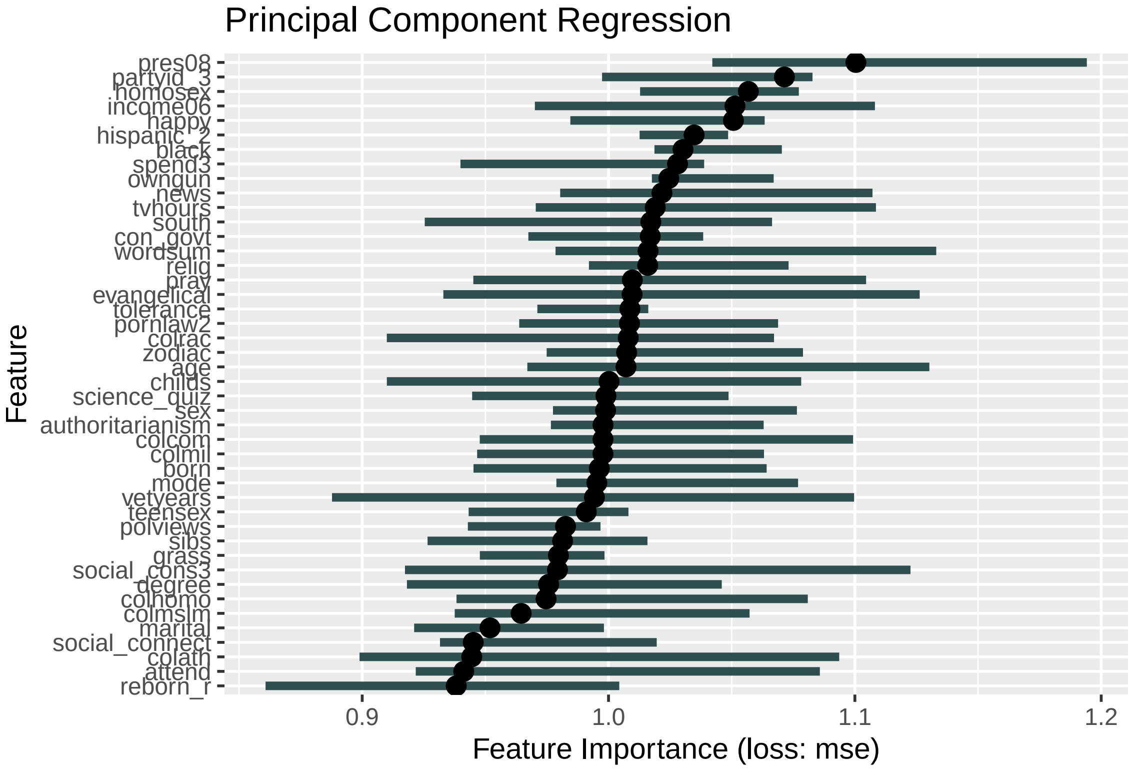
```r
plot(lr_fea) + ggtitle('Linear Regression')
```



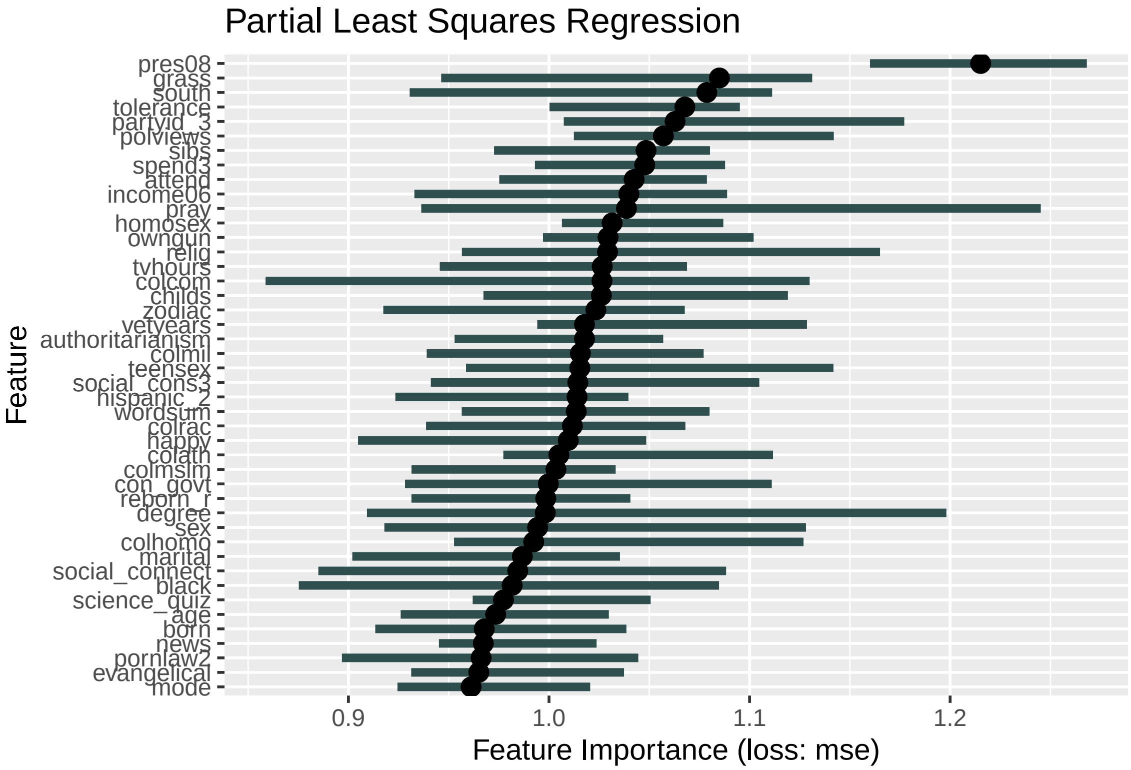
```
plot(eln_fea) + ggtitle('Elastic Net Regression')
```



```r
plot(pcr_fea) + ggtitle('Principal Component Regression')
```



```r
plot(pls_fea) + ggtitle('Partial Least Squares Regression')
```


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