Problem Set 4

Pete Cuppernull

2/16/2020

Load packages

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(tidymodels)
library(rcfss)
library(knitr)
library(splines)
library(lattice)
library(here)
library(patchwork)
library(margins)
```

Load Data

```
gss_test <- read_csv(here("data/gss_test.csv"))
gss_train <- read_csv(here("data/gss_train.csv"))
```

1. Polynomial Regression

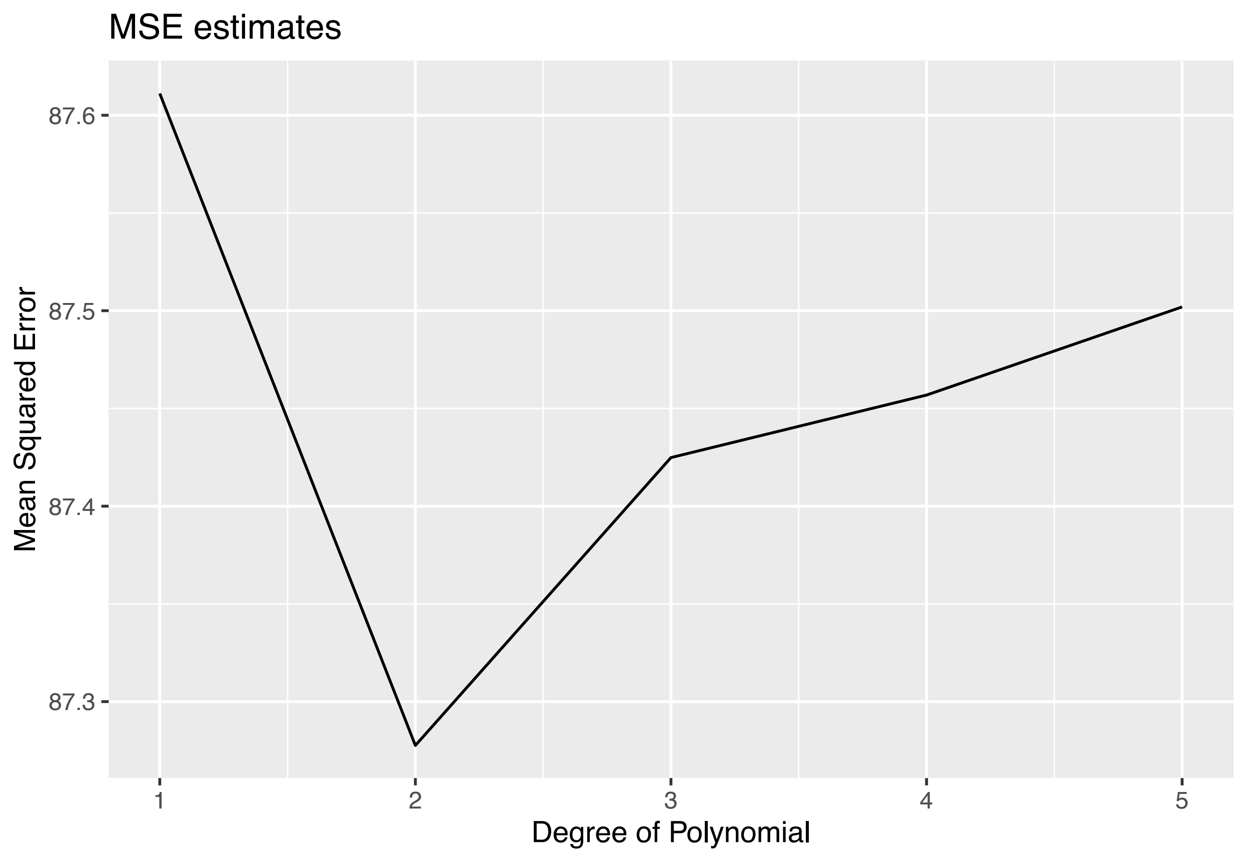
```
egalit_poly <- glm(egalit_scale ~ poly(income06, degree = 2), data = gss_train, family = gaussian)

holdout_results <- function(splits, i) {
  # Fit the model to the training set
  mod <- glm(egalit_scale ~ poly(income06, i, raw = TRUE), data = analysis(splits))

  # Save the heldout observations
  holdout <- assessment(splits)

  # `augment` will save the predictions with the holdout data set
  res <- augment(mod, newdata = holdout) %>%
    # calculate the metric
    mse(truth = egalit_scale, estimate = .fitted)

  # Return the assessment data set with the additional columns
  res
}

# function to return MSE for a specific fit
poly_mse <- function(i, vfold_data){
  vfold_mod <- vfold_data %>%
    mutate(results = map(splits, holdout_results, i)) %>%
    unnest(results) %>%
    spread(.metric, .estimate)

  mean(vfold_mod$mse)
}

# split Auto into 10 folds
gss_cv10 <- vfold_cv(data = gss_train,
                     v = 10)

cv_mse <- tibble(terms = seq(from = 1,
                             to = 5),
                 mse_vfold = map_dbl(terms, poly_mse, gss_cv10))
cv_mse
```

```
## # A tibble: 5 x 2
##   terms mse_vfold
##   <int>     <dbl>
## 1     1      87.6
## 2     2      87.3
## 3     3      87.4
## 4     4      87.5
## 5     5      87.5
```

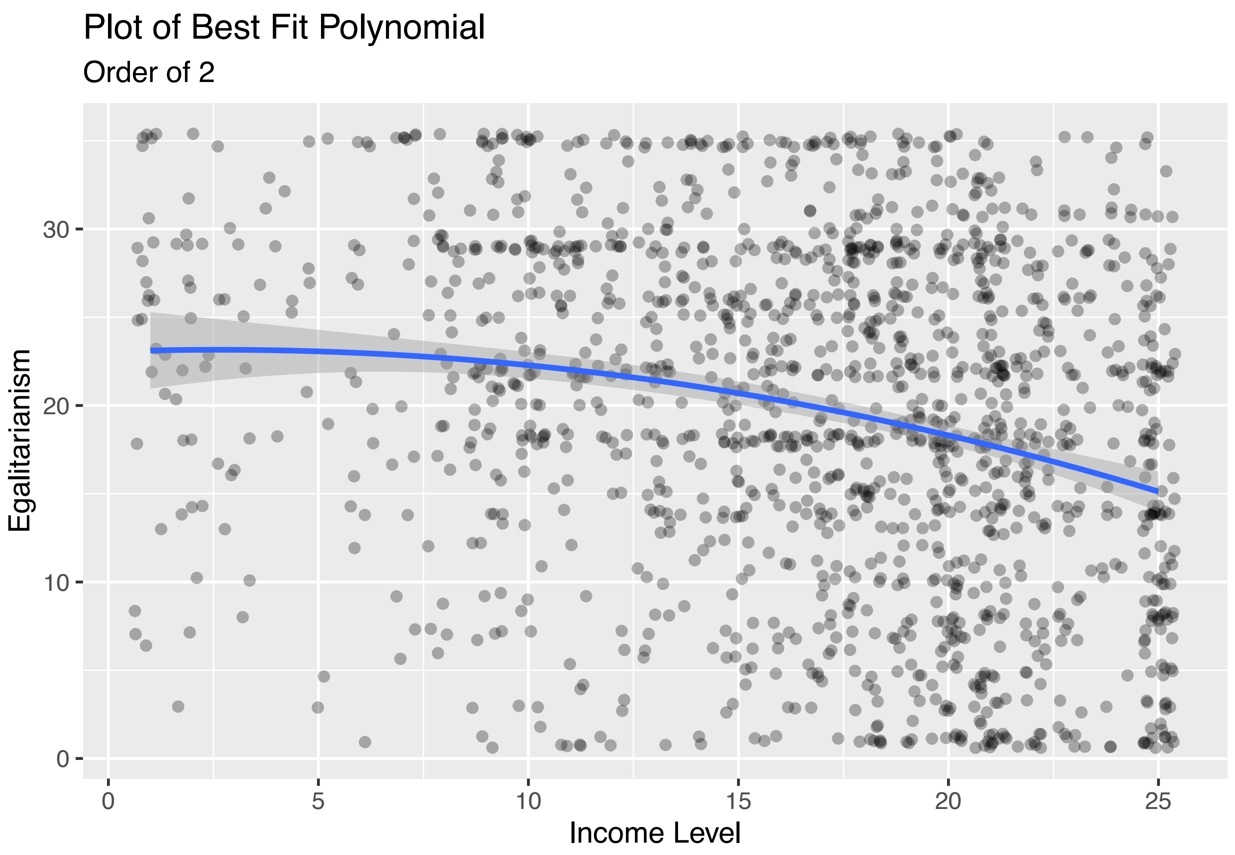
```
which.min(cv_mse$mse_vfold)
```

```
## [1] 2
```

```
ggplot(cv_mse, aes(terms, mse_vfold)) +
  geom_line() +
  scale_color_brewer(type = "qual") +
  labs(title = "MSE estimates",
       x = "Degree of Polynomial",
       y = "Mean Squared Error",
       color = "CV Method")
```



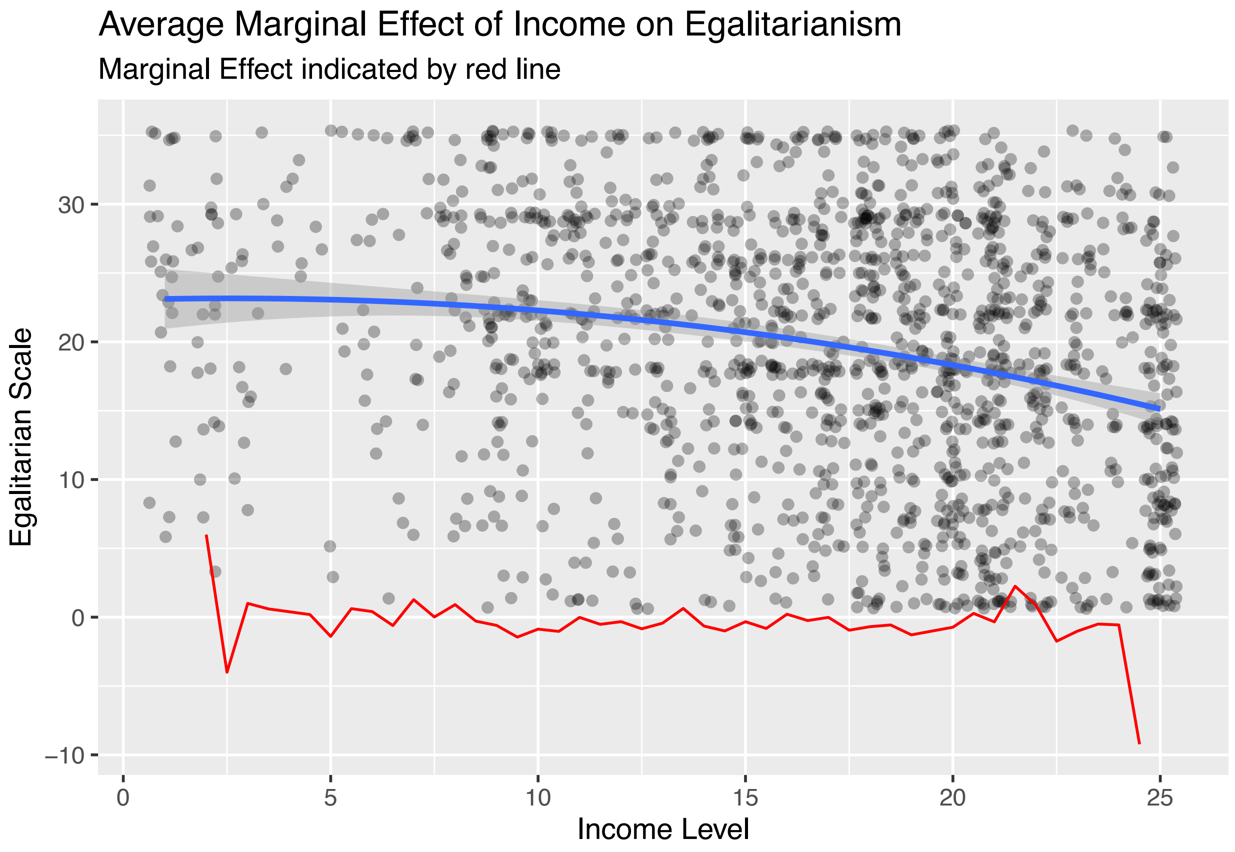
```
##plot the fit to the data

ggplot(gss_train, aes(income06, egalit_scale)) +
  geom_jitter(alpha = .3) +
  stat_smooth(method='lm', formula = y ~ poly(x, degree = 2)) +
  labs(title = "Plot of Best Fit Polynomial",
       subtitle = "Order of 2",
       x = "Income Level",
       y = "Egalitarianism")
```



The optimal degree of the polynomial is 2.

Average Marginal Effects

```
dY <- diff(gss_train$egalit_scale)/diff(gss_train$income06)
dX <- rowMeans(embed(gss_train$income06,2))
originalData <- data.frame(X = gss_train$income06, Y = gss_train$egalit_scale, type = "Original")
derivativeData <- data.frame(X = dX, Y = dY, type = "Derivative")


derivate_averages <- derivativeData %>%
  group_by(X) %>%
  filter(Y != -Inf) %>%
  filter(Y != Inf) %>%
  summarize(marginal_effect = mean(Y))
  #mutate(x2 = (X*2),
       #  is.whole = if_else(x2 %% 2 == 0, TRUE, FALSE)) %>%
  #filter(is.whole == TRUE)

ggplot() +
  geom_jitter(data = gss_train, mapping = aes(income06, egalit_scale), alpha = .3) +
  stat_smooth(data = gss_train,
              mapping = aes(income06, egalit_scale),
              method='lm',
              formula = y ~ poly(x, degree = 2)) +
  geom_line(data = derivate_averages, mapping = aes(X, marginal_effect), color = "red") +
  labs(title = "Average Marginal Effect of Income on Egalitarianism",
       subtitle = "Marginal Effect indicated by red line",
       x = "Income Level",
       y = "Egalitarian Scale")
```

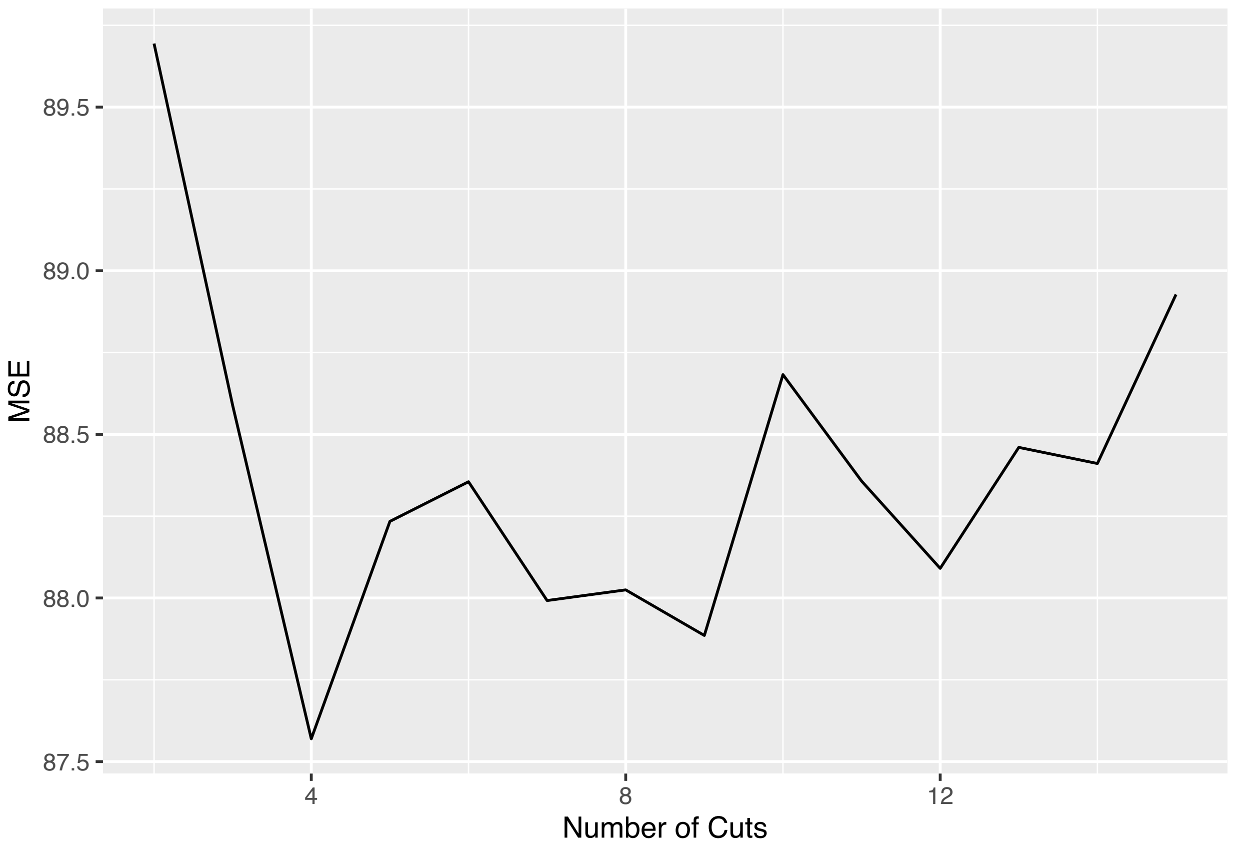


The average marginal effect of income level on egalitarianism remains largely negative throughout the range of income levels. This indicates that as individuals make more money, they are less likely to have egalitarian preferences. We can observe, however, that from income levels 6-8, egalitarianism increases as income increases – this would suggest a possible anomaly for this income range where individuals become more egalitarian as their income increases.

2. Step Function

```
egalit_step <- glm(egalit_scale ~ cut(income06, 5), data = gss_train)


holdout_results_step <- function(splits, i) {
  # Fit the model to the training set
  mod <- glm(egalit_scale ~ cut(income06, i),
             data = analysis(splits))

  # Save the heldout observations
  holdout <- assessment(splits)

  # `augment` will save the predictions with the holdout data set
  res <- augment(mod, newdata = holdout) %>%
    # calculate the metric
    mse(truth = egalit_scale, estimate = .fitted)
```

```r
  # Return the assessment data set with the additional columns
  res
}

# function to return MSE for a specific fit
step_mse <- function(i, vfold_data){
  vfold_mod <- vfold_data %>%
    mutate(results = map(splits, holdout_results_step, i)) %>%
    unnest(results) %>%
    spread(.metric, .estimate)

  mean(vfold_mod$mse)
}

# split Auto into 10 folds
gss_cv10_step <- vfold_cv(data = gss_train,
                      v = 10)

cv_mse <- tibble(terms = seq(from = 2,
                             to = 15),
                 mse_vfold = map_dbl(terms, step_mse, gss_cv10_step))
cv_mse %>%
  ggplot(aes(terms, mse_vfold)) +
  geom_line() +
  labs(Title = "MSEs of Step Functions with Varying Cuts",
       y = "MSE",
       x = "Number of Cuts")
```

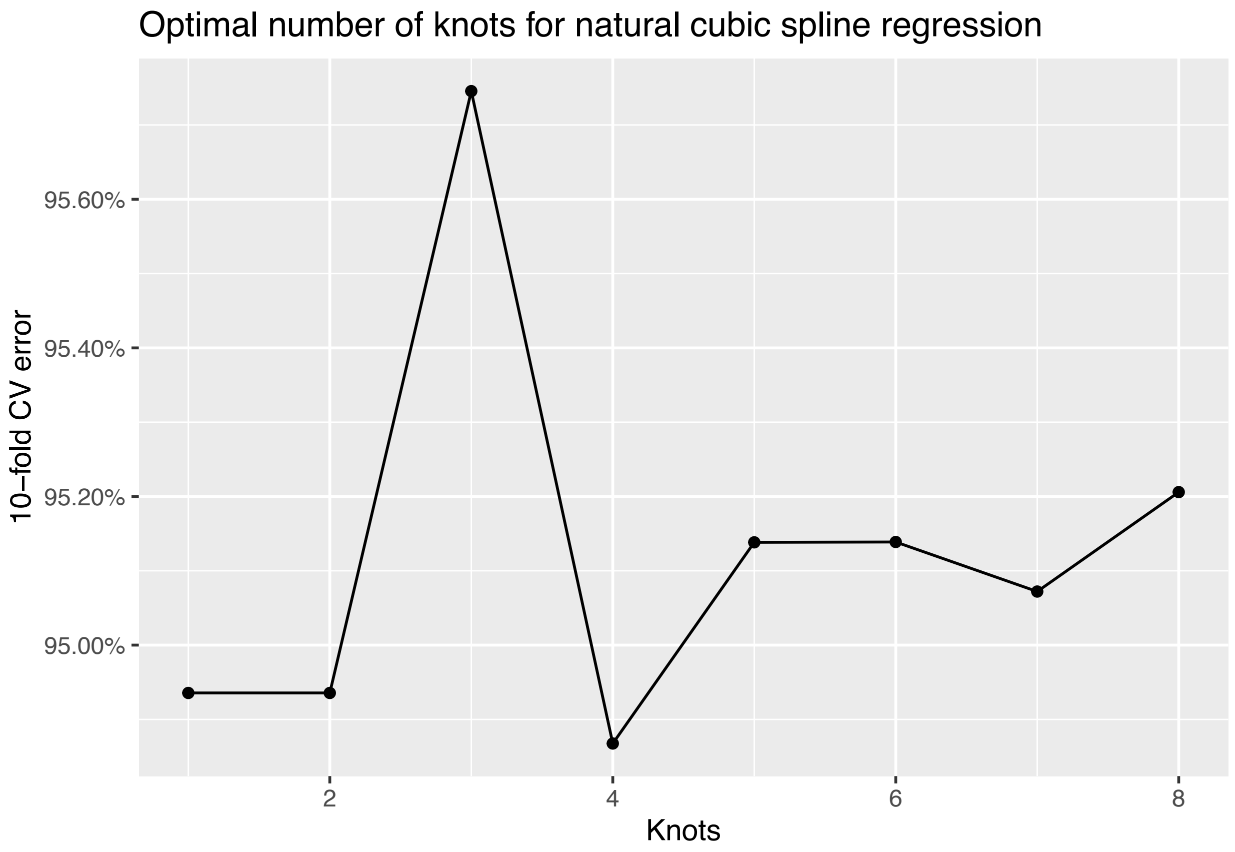


The optimum number of cuts basic on the cross-validation MSEs is 4.

3. Natural Regression Spline

```
gss_spline <- function(splits, df = NULL){
  # estimate the model on each fold
  model <- glm(egalit_scale ~ ns(income06, df = df),
               data = analysis(splits))


  model_acc <- augment(model, newdata = assessment(splits)) %>%
    accuracy(truth = factor(egalit_scale, levels = 1:35), estimate = factor(round(.fitted), levels = 1:

  mean(model_acc$.estimate)
}



tune_over_knots <- function(splits, knots){
  gss_spline(splits, df = knots + 3)
}

# estimate CV error for knots in 0:25
results <- vfold_cv(gss_train, v = 10)
```

```r
expand(results, id, knots = 1:8) %>%
  left_join(results) %>%
  mutate(acc = map2_dbl(splits, knots, tune_over_knots)) %>%
  group_by(knots) %>%
  summarize(acc = mean(acc)) %>%
  mutate(err = 1 - acc) %>%
  ggplot(aes(knots, err)) +
  geom_point() +
  geom_line() +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Optimal number of knots for natural cubic spline regression",
       x = "Knots",
       y = "10-fold CV error")
```



```r
best_model <- glm(egalit_scale ~ ns(income06, df = 10),
                  data = gss_train)
summary(best_model)
```

```
##
## Call:
## glm(formula = egalit_scale ~ ns(income06, df = 10), data = gss_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -22.0935   -6.4923    0.2215    7.1448   20.5398
##
## Coefficients:
```

```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               22.9419     1.4240  16.111  < 2e-16 ***
## ns(income06, df = 10)1   -1.4324     1.7890  -0.801  0.42345
## ns(income06, df = 10)2   -0.3516     2.3945  -0.147  0.88330
## ns(income06, df = 10)3   -4.1090     2.0934  -1.963  0.04985 *
## ns(income06, df = 10)4   -2.0648     2.0490  -1.008  0.31376
## ns(income06, df = 10)5   -4.7438     2.1182  -2.240  0.02527 *
## ns(income06, df = 10)6   -5.4505     2.0994  -2.596  0.00952 **
## ns(income06, df = 10)7   -7.8180     2.4016  -3.255  0.00116 **
## ns(income06, df = 10)8   -0.6552     2.5216  -0.260  0.79503
## ns(income06, df = 10)9   -6.0536     3.6497  -1.659  0.09740 .
## ns(income06, df = 10)10  -8.8145     1.3828  -6.374 2.45e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 86.87606)
##
##     Null deviance: 137007  on 1480  degrees of freedom
## Residual deviance: 127708  on 1470  degrees of freedom
## AIC: 10828
##
## Number of Fisher Scoring iterations: 2
```

The optimum number of knots is 7.

Problem Set 4 - Part 2

Pete Cuppernull

2/16/2020

4. Estimate variety of models

a. Linear Regression

```
#Set Train Control
library(glmnet)
library(caret)

train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(egalit_scale ~., data = gss_train, method = "lm",
               trControl = train.control)
# Summarize the results
print(model)
```

```
## Linear Regression
##
## 1481 samples
##   44 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1333, 1332, 1333, 1334, 1332, 1332, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   7.962045  0.3227389  6.300663
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

b. Elastic net regression

```
library(glmnet)
gss_train_x <- model.matrix(egalit_scale ~ ., gss_train)[, -1]
gss_train_y <- log(gss_train$egalit_scale)

gss_test_x <- model.matrix(egalit_scale ~ ., gss_test)[, -1]
gss_test_y <- log(gss_test$egalit_scale)

fold_id <- sample(1:10, size = length(gss_train_y), replace = TRUE)

tuning_grid <- tibble::tibble(
  alpha      = seq(0, 1, by = .1),
  mse_min    = NA,
```

```r
  mse_1se    = NA,
  lambda_min = NA,
  lambda_1se = NA
)

for(i in seq_along(tuning_grid$alpha)) {
  # fit CV model for each alpha value
  fit <- cv.glmnet(gss_train_x,
                   gss_train_y,
                   alpha = tuning_grid$alpha[i],
                   foldid = fold_id)
    # extract MSE and lambda values
  tuning_grid$mse_min[i]    <- fit$cvm[fit$lambda == fit$lambda.min]
  tuning_grid$mse_1se[i]    <- fit$cvm[fit$lambda == fit$lambda.1se]
  tuning_grid$lambda_min[i] <- fit$lambda.min
  tuning_grid$lambda_1se[i] <- fit$lambda.1se
}

en_mse <- min(tuning_grid$mse_min)
```

The MSE of the Elastic Net approach is 0.4930639.

c. Principal component regression

```r
library(pls)
# PCR first
gss_pcr <- pcr(egalit_scale ~ .,
                data = select_if(gss_train, is.numeric),
                center = TRUE,
                scale = TRUE,
                validation = "CV")

# extract needed stats
gss_pcr_stats <- tibble(
  pct_exp = loadings(gss_pcr) %>%
    attr("explvar"),
  mse = as.vector(MSEP(gss_pcr, estimate = "CV", intercept = FALSE)$val)
) %>%
  mutate(pc = row_number(),
         cum_exp = cumsum(pct_exp) / 100)

pcr_mse <- min(gss_pcr_stats$mse)
```

The MSE of the PCR approach is 83.6521953.

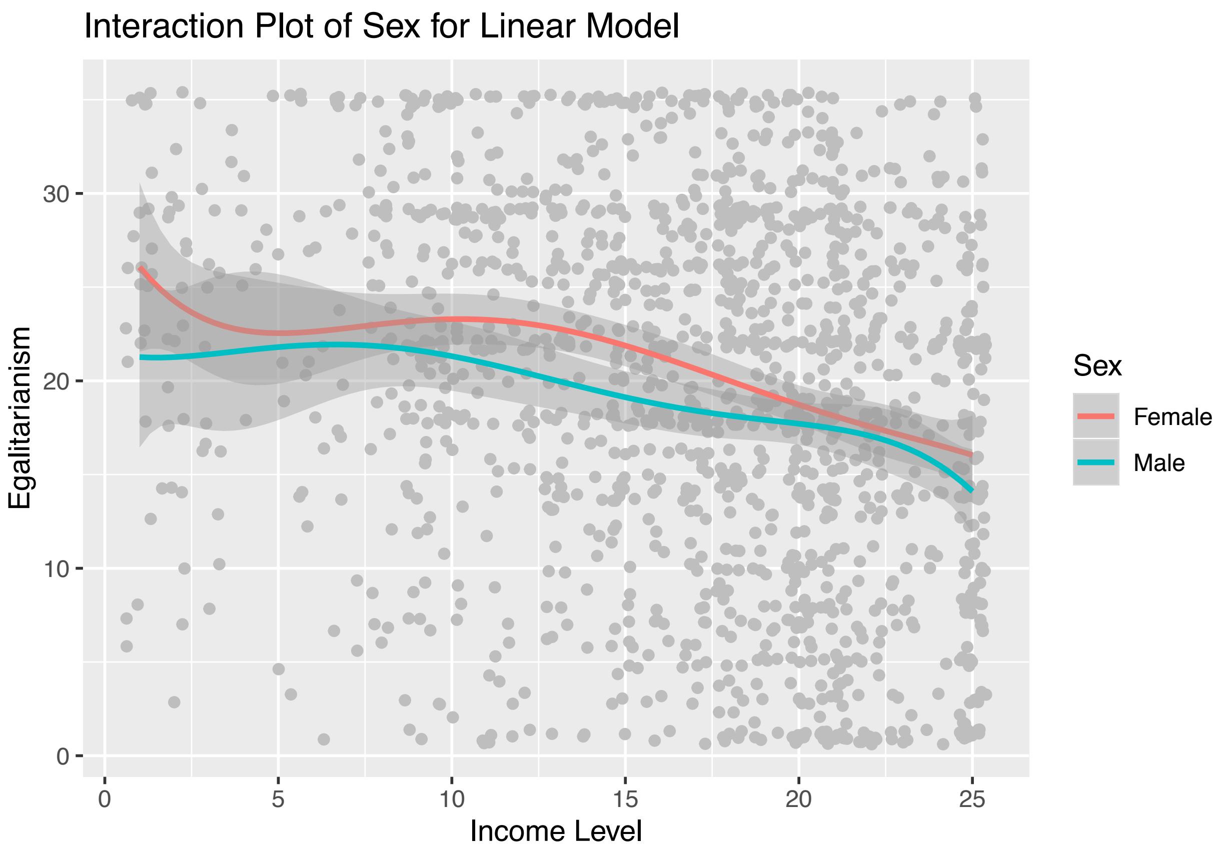
d. Partial least squares regression

```r
gss_pls <- plsr(egalit_scale ~ .,
                 data = select_if(gss_train, is.numeric),
                 center = TRUE,
                 scale = TRUE,
                 validation = "CV")
# extract needed stats
gss_pls_stats <- tibble(
  pct_exp = loadings(gss_pls) %>% attr("explvar"),
  mse = as.vector(MSEP(gss_pls, estimate = "CV", intercept = FALSE)$val)
) %>%
  mutate(pc = row_number(),
         cum_exp = cumsum(pct_exp) / 100)
pls_mse <- min(gss_pls_stats$mse)
```

The MSE of the Elastic Net approach is 83.8332953.

Question 5 - Discussion

```
##Interaction Plot for LM
ggplot(gss_train, aes(income06, egalit_scale)) +
  geom_jitter(color = "gray") +
  stat_smooth(method='lm', formula = y ~ poly(x, degree = 5), mapping = aes(color = as.factor(sex))) +
  labs(color = "Sex",
       title = "Interaction Plot of Sex for Linear Model",
       x = "Income Level",
       y = "Egalitarianism")
```



As an example, we can considering the interaction plot above to see that there is a disparate effect of income on egalitarianism between males and females. Considering all of our modeling approaches though, the elastic net appproach appears to be the best given its low MSE of 0.4930639.