hw 4

Hengle Li

2/15/2020

```
library(tidyverse)
library(tidymodels)
library(rcfss)
library(knitr)
library(splines)
library(lattice)
library(margins)
library(rsample)
library(caret)
library(glmnet)
library(leaps)
library(yardstick)
library(glmnet)
library(pls)

set.seed(666)
```

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

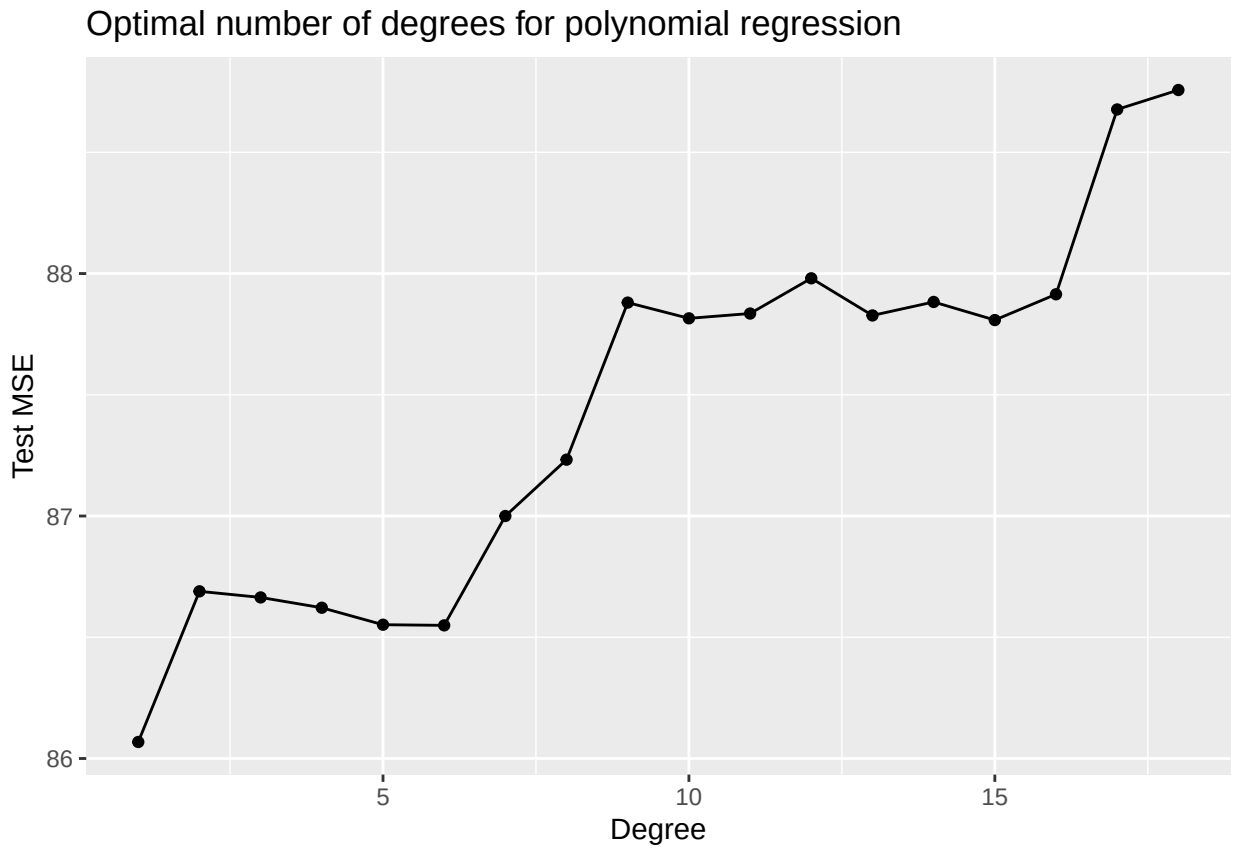
Egalitarianism and income

Polynomial regression

```
# set up function
egal_poly <- function(splits, de = NULL){
  # estimate the model on each fold
  model <- glm(egalit_scale ~ poly(income06, degree = de),
               data = analysis(splits))
  model_mse <- augment(model, newdata = gss_test) %>%
    mse(truth = egalit_scale, estimate = round(.fitted))
  model_mse$.estimate
}
```

```
#create the 10-fold CV
#since 10-fold CV is a random process
#set it apart from the functions
training_10fold <- vfold_cv(gss_train, 10)
```

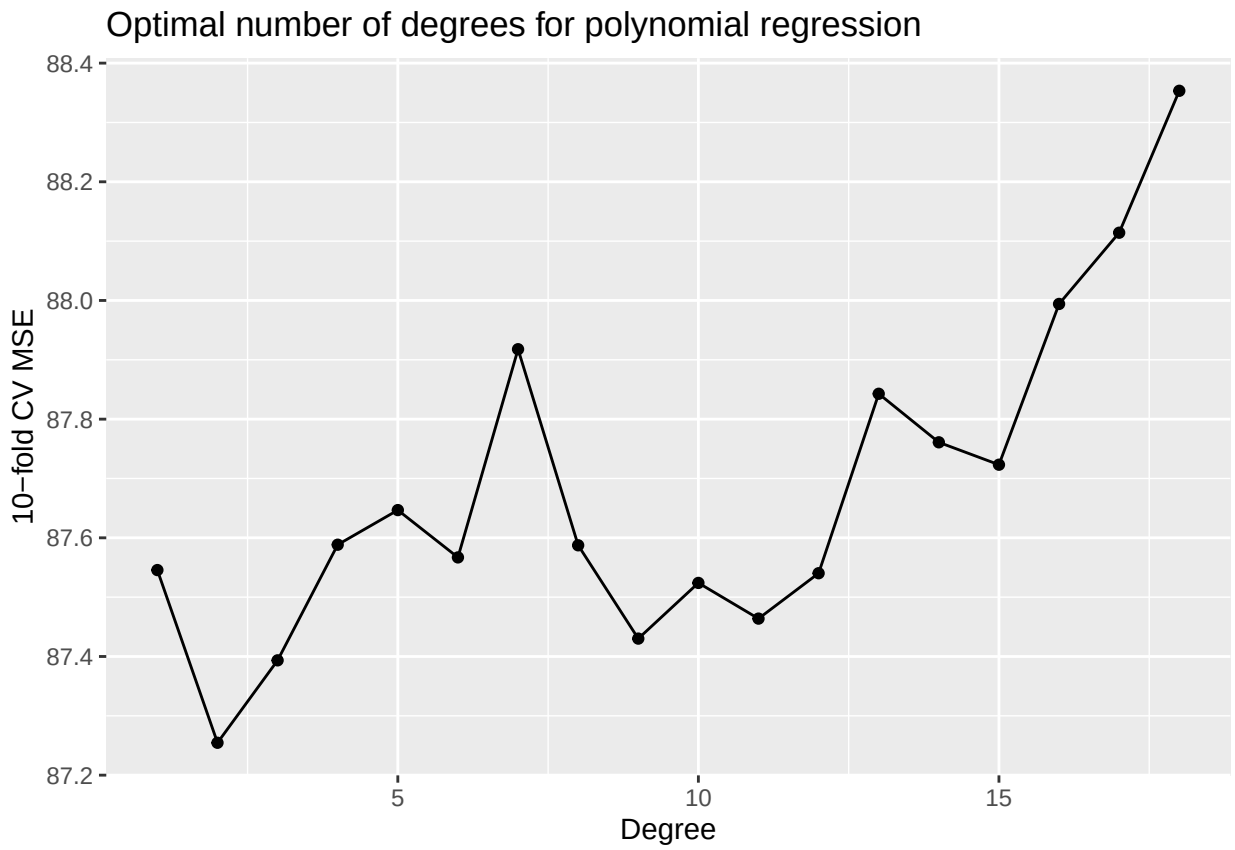
```r
#map values to the function
tidyr::expand(training_10fold, id, de = 1:18) %>%
  left_join(training_10fold) %>%
  mutate(mse = map2(splits, de, egal_poly)) %>%
  #use unlist() to extract the mse
  mutate(mse = unlist(mse)) %>%
  group_by(de) %>%
  summarise(mse = mean(mse)) %>%
  ggplot(aes(de, mse)) +
  geom_point() +
  geom_line() +
  labs(title = "Optimal number of degrees for polynomial regression",
       x = "Degree",
       y = "Test MSE")
```



- Taking 18 as the max possible degree because at 19, the MSE will be so large that the others appear almost identical
- Note that the MSE is calculated based on predictions with the testing set rather than the training set. Since the training set is already used to train models, it's pointless to see how well these models fit to the training set.
- However, if we do it this way, as can be seen in the graph here, the min MSE is at degree = 1. After degree = 1, the next lowest MSE is at degree = 5.

```r
# set up function
egal_poly_ass <- function(splits, de = NULL){
  # estimate the model on each fold
  model <- glm(egalit_scale ~ poly(income06, degree = de),
               data = analysis(splits))
  model_mse <- augment(model, newdata = assessment(splits)) %>%
    mse(truth = egalit_scale, estimate = round(.fitted))
  model_mse$.estimate
}
```

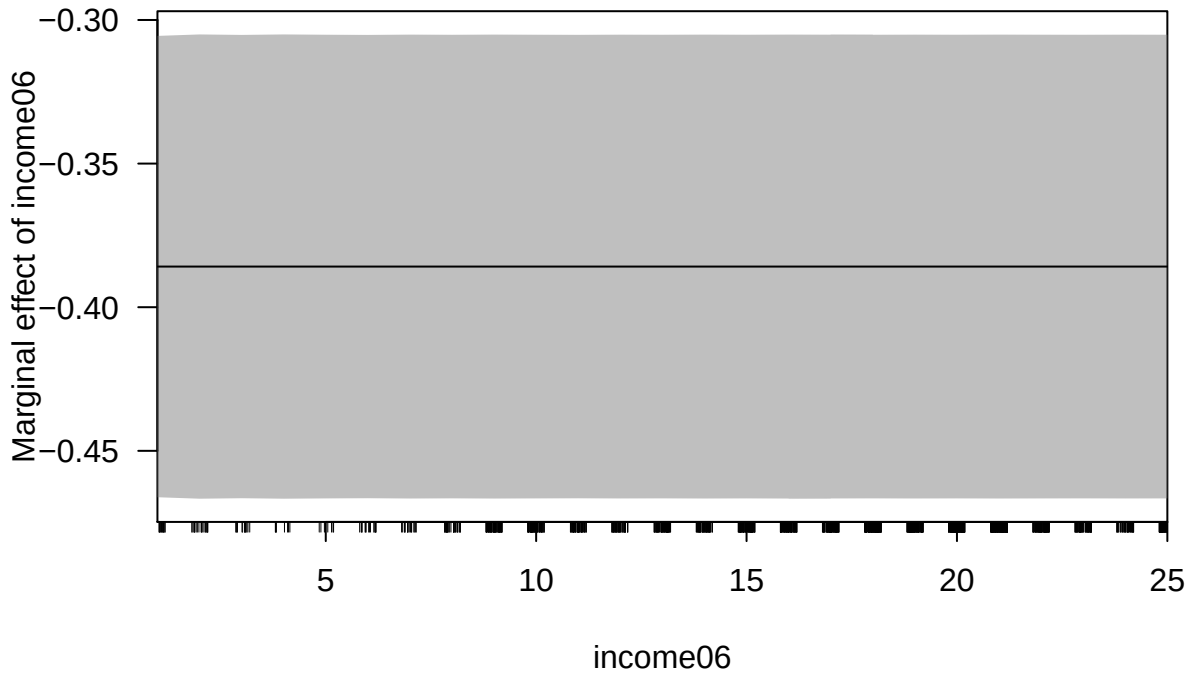
```r
#map values to the function
tidyr::expand(training_10fold, id, de = 1:18) %>%
  left_join(training_10fold) %>%
  mutate(mse = map2(splits, de, egal_poly_ass)) %>%
  #use unlist() to extract the mse
  mutate(mse = unlist(mse)) %>%
  group_by(de) %>%
  summarise(mse = mean(mse)) %>%
  ggplot(aes(de, mse)) +
  geom_point() +
  geom_line() +
  labs(title = "Optimal number of degrees for polynomial regression",
       x = "Degree",
       y = "10-fold CV MSE")
```



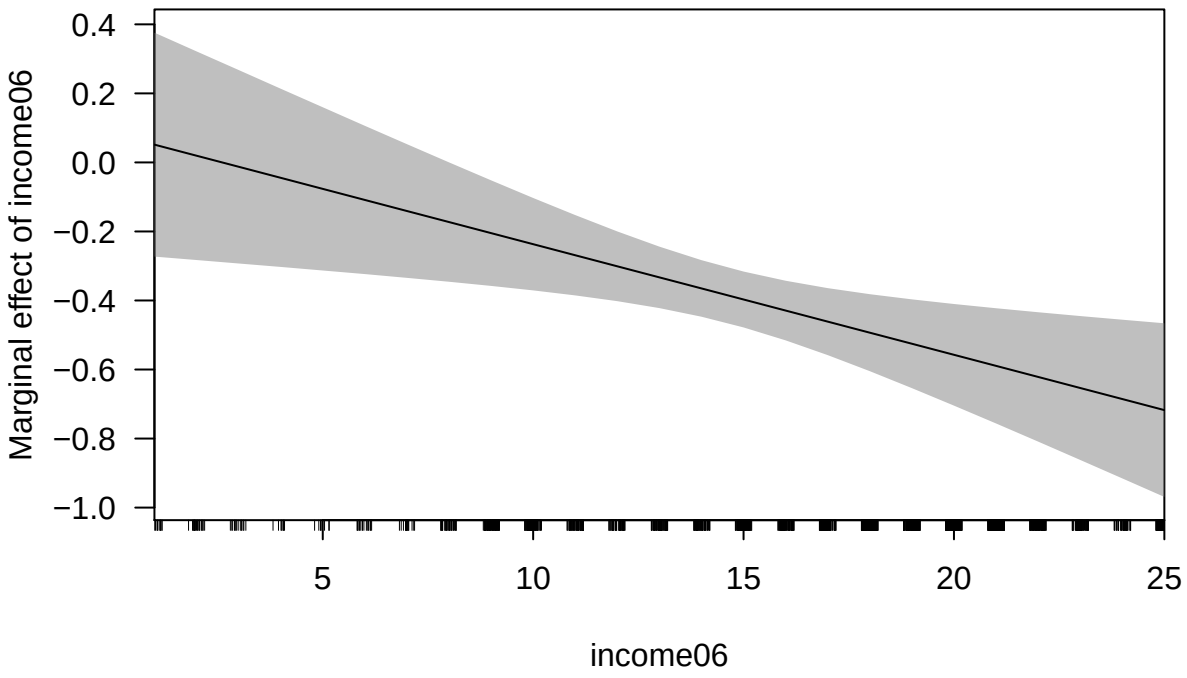
*As this graph shows, if we want to base our judgment on fitting the training set, then degree = 2 will be

hte optimal degree for the polynomial regression, although we know from the previous graph that 2 is not the best in fitting the test set.

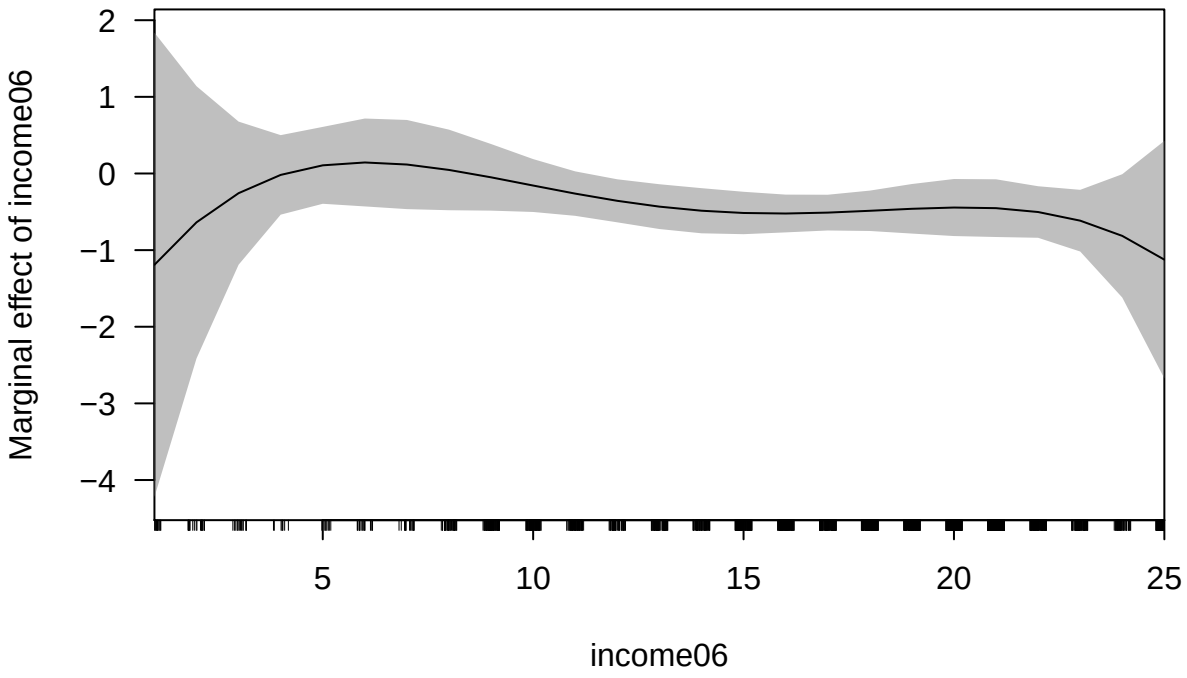
```
#set up model
#note use stats::poly, or cplot will misrecognize it
egal_poly1 <- glm(egalit_scale ~ stats::poly(income06, degree = 1),
                  data = gss_train)
cplot(egal_poly1, "income06", what = "effect")
```




```
egal_poly2 <- glm(egalit_scale ~ stats::poly(income06, degree = 2),
                  data = gss_train)
cplot(egal_poly2, "income06", what = "effect")
```

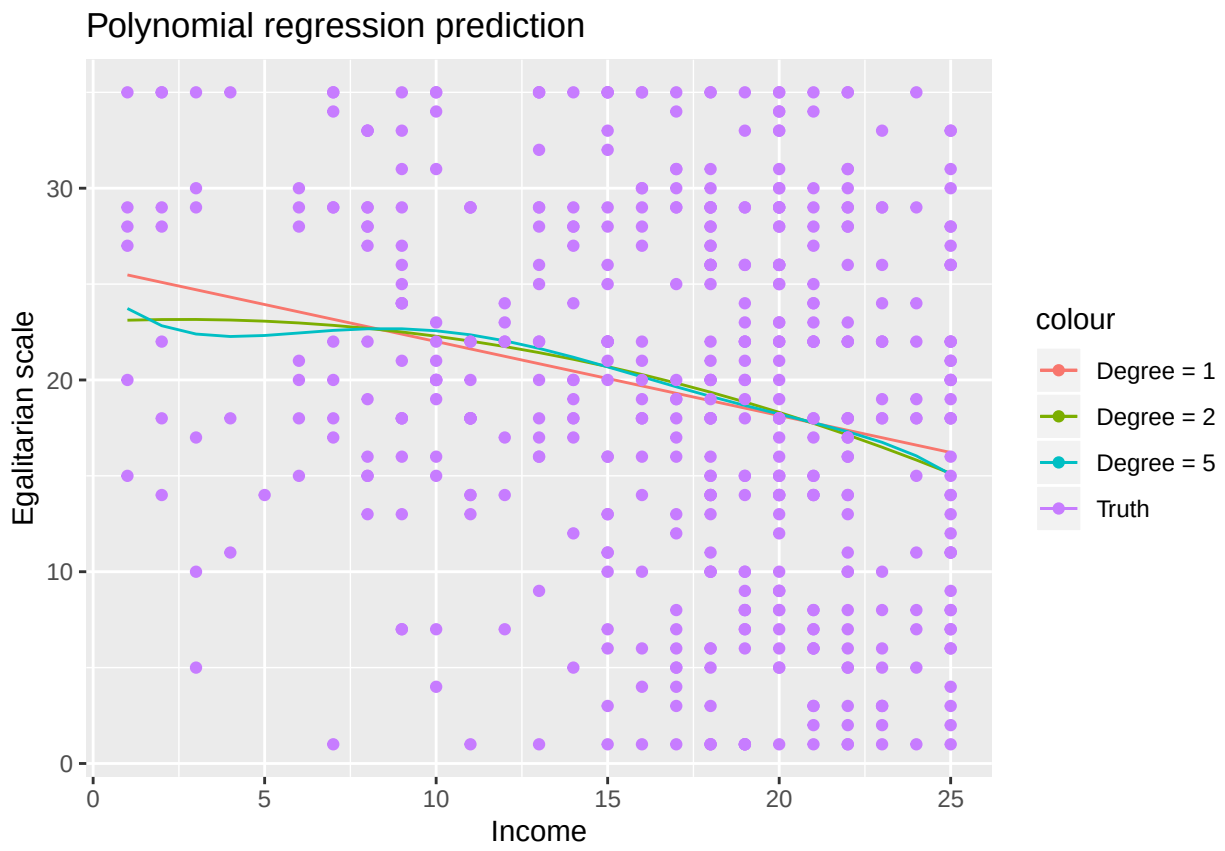



```
egal_poly5 <- glm(egalit_scale ~ stats::poly(income06, degree = 5),
                  data = gss_train)
cplot(egal_poly5, "income06", what = "effect")
```



- By comparing the AME for polynomial of degree 1 and for polynomial of degree 5, we can see that, when degree = 1, the average marginal effect of income06 on eagalit_scale is negative and constantly between -0.5 and -0.3. In contrast,
- At degree = 2, AME is relatively more various over the range of income05, ranging from -1 to 0.4 (confidence interval considered).
- At degree = 5, AME changes drastically at both ends, and it wavers between -1 and 1 most of the time.
- It's hard to determine which is better without further support, since we don't know what income06 means to egalitarian scale in reality. Perhaps degree = 1 is better than 2 and 5, because its AME is more constant. But degree = 1 may perform not as well with more data.

```
gss_test %>%
  mutate(pred1 = predict(egal_poly1, newdata = gss_test),
         pred2 = predict(egal_poly2, newdata = gss_test),
         pred5 = predict(egal_poly5, newdata = gss_test)) %>%
  ggplot(aes(x = income06)) +
  geom_line(aes(y = pred1, color = "Degree = 1")) +
  geom_line(aes(y = pred2, color = "Degree = 2")) +
  geom_line(aes(y = pred5, color = "Degree = 5")) +
  geom_point(aes(y = egalit_scale, color = "Truth")) +
  labs(title = "Polynomial regression prediction",
       x = "Income",
       y = "Egalitarian scale")
```



- As we can see by plotting the predictions against actual data, degree 2 and 5 produce almost identical curves, especially at income06 > 7. And before that point, data are too scarse to make any decision.
- All things considered, degree $= 5$ should be the optimal degree for polynomial regression here.

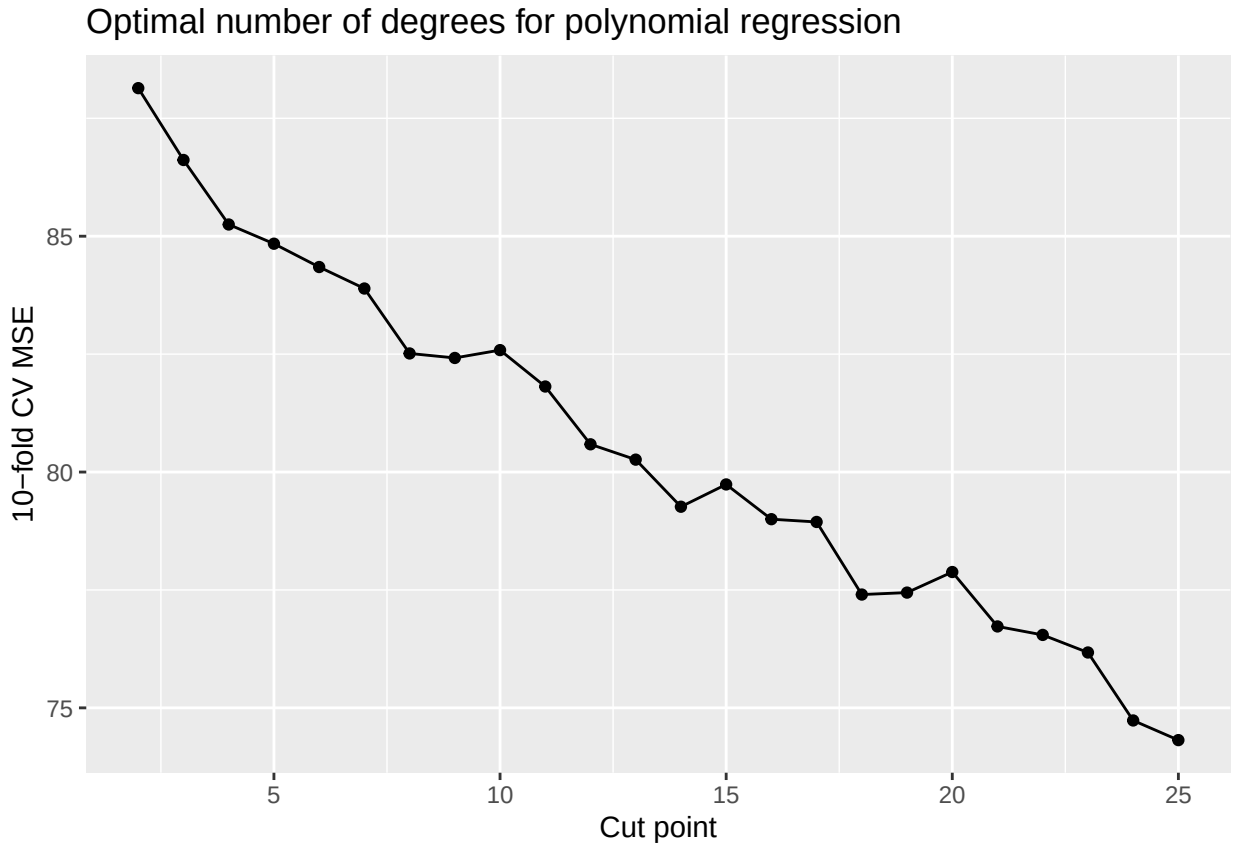
Step function

By definition of step functions, the range of x needs to be divided intervals of equal lengths. Use cut() function for this purpose.

```
#build function to calculate mse for glm
#note that we use assessment() instead of analysis()
mse_sp <- function(splits, cc){
  model <- glm(egalit_scale ~ cut(income06, cc),
               data = assessment(splits))
  model_mse <- augment(model, newdata = assessment(splits)) %>%
    mse(truth = egalit_scale, estimate = round(.fitted))
  model_mse$.estimate
}
```

```
#calculate mse
tidyr::expand(training_10fold, id, cc = 2:25) %>%
  left_join(training_10fold) %>%
  mutate(mse = map2(splits, cc, mse_sp)) %>%
  #use unlist() to extract the mse
```

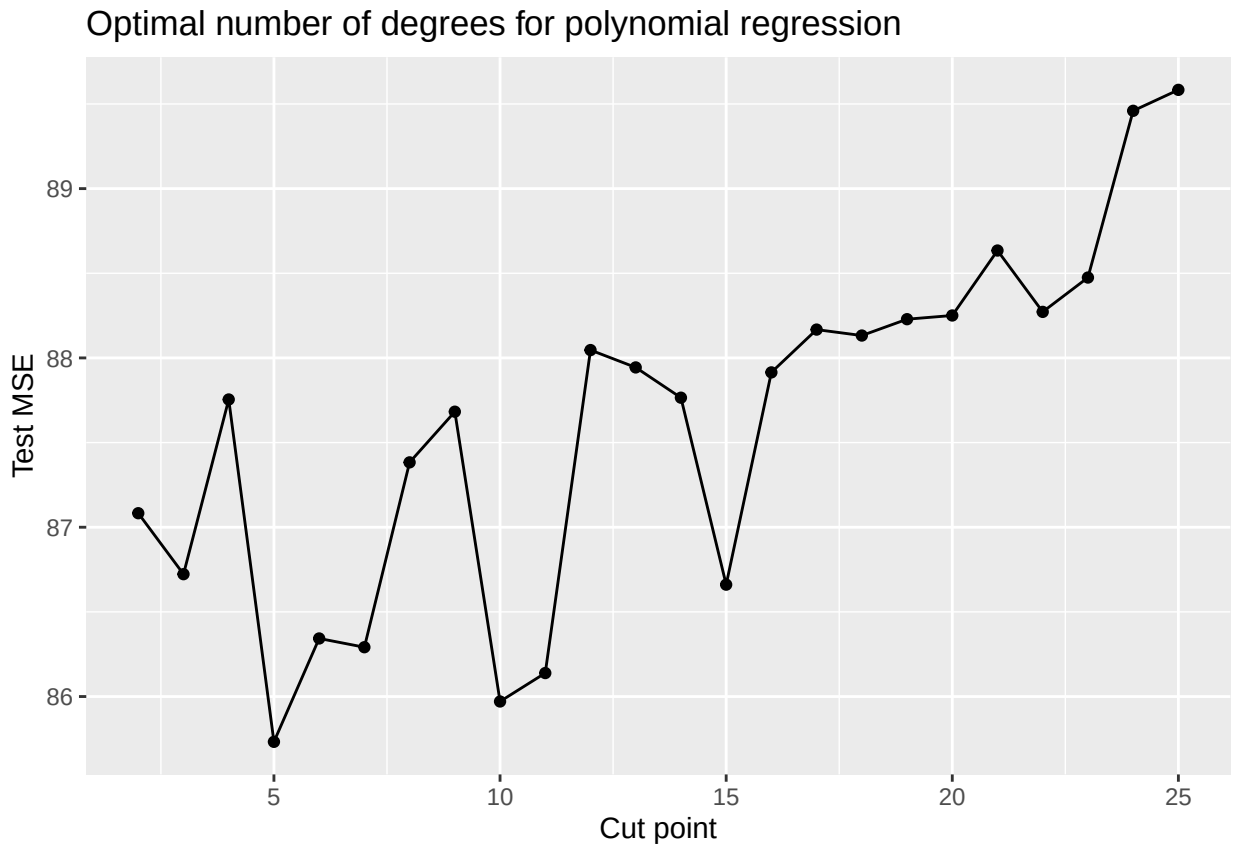
```
  mutate(mse = unlist(mse)) %>%
  group_by(cc) %>%
  summarise(mse = mean(mse))%>%
  ggplot(aes(cc, mse)) +
  geom_point() +
  geom_line() +
  labs(title = "Optimal number of degrees for polynomial regression",
       x = "Cut point",
       y = "10-fold CV MSE")
```



- As expected, in general, the more cut points there are, the lower the MSE will be. But if we choose 25, the model wil undoubtedly be overfitting.

```
#build function to calculate mse for glm
#note that we use assessment() instead of analysis()
mse_sp_test <- function(splits, cc){
  model <- glm(egalit_scale ~ cut(income06, cc),
               data = analysis(splits))
  model_mse <- augment(model, newdata = gss_test) %>%
    mse(truth = egalit_scale, estimate = round(.fitted))
  model_mse$.estimate
}
```

```
#calculate mse
tidyr::expand(training_10fold, id, cc = 2:25) %>%
  left_join(training_10fold) %>%
  mutate(mse = map2(splits, cc, mse_sp_test)) %>%
  #use unlist() to extract the mse
  mutate(mse = unlist(mse)) %>%
  group_by(cc) %>%
  summarise(mse = mean(mse))%>%
  ggplot(aes(cc, mse)) +
  geom_point() +
  geom_line() +
  labs(title = "Optimal number of degrees for polynomial regression",
       x = "Cut point",
       y = "Test MSE")
```

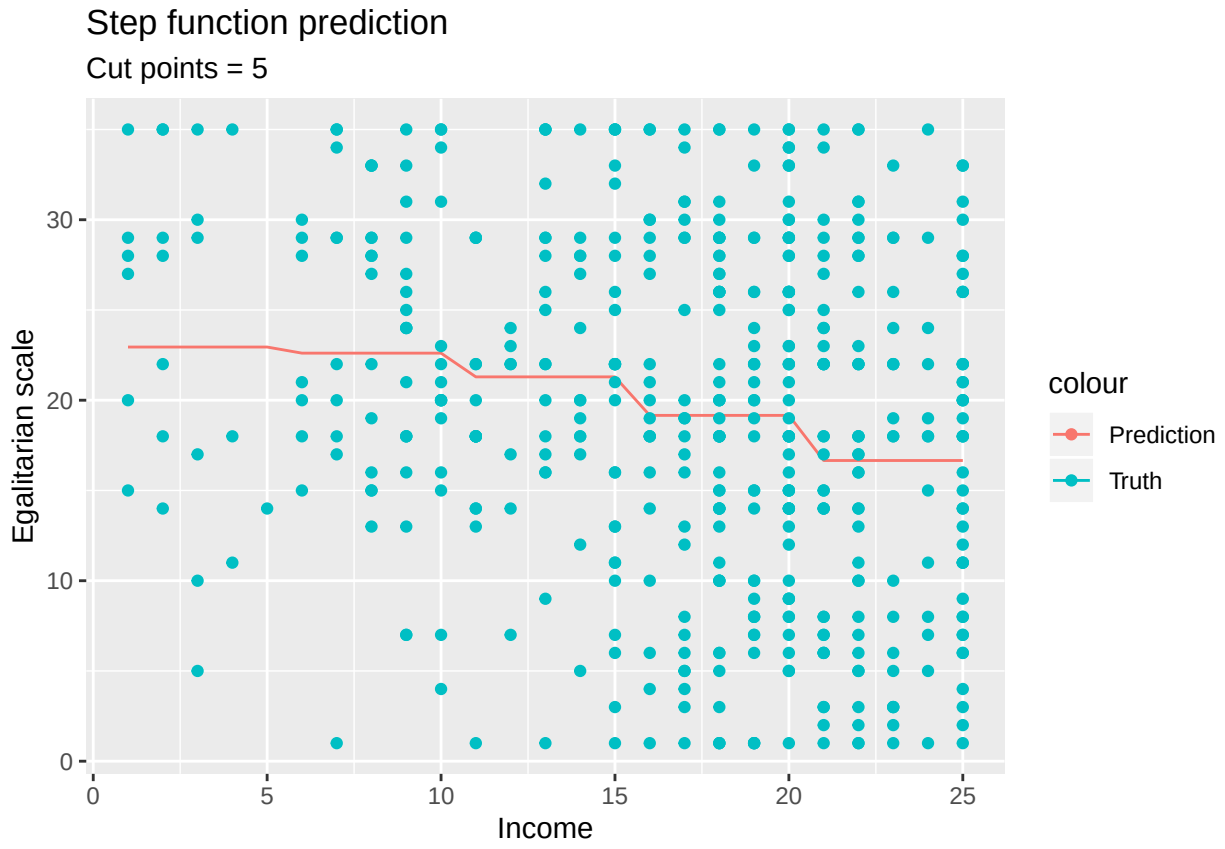


- If we repeat what is done about polynomial regression, i.e. plotting the MSE from fitting the models to the test set, we can see that 5 turns out the optimal number of cut points here.
- And of course, 25 has the highest MSE.

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

```
gss_test %>%
  mutate(pred = predict(egal_sp, newdata = gss_test)) %>%
```

```
  ggplot(aes(x = income06, y = pred)) +
  geom_line(aes(color = "Prediction")) +
  geom_point(aes(y = egalit_scale, color = "Truth")) +
  labs(title = "Step function prediction",
       subtitle = "Cut points = 5",
       x = "Income",
       y = "Egalitarian scale")
```



As can be seen by plotting predictions against actual data, the prediction curve of the setp function is similiar to that of a polynomial curve.

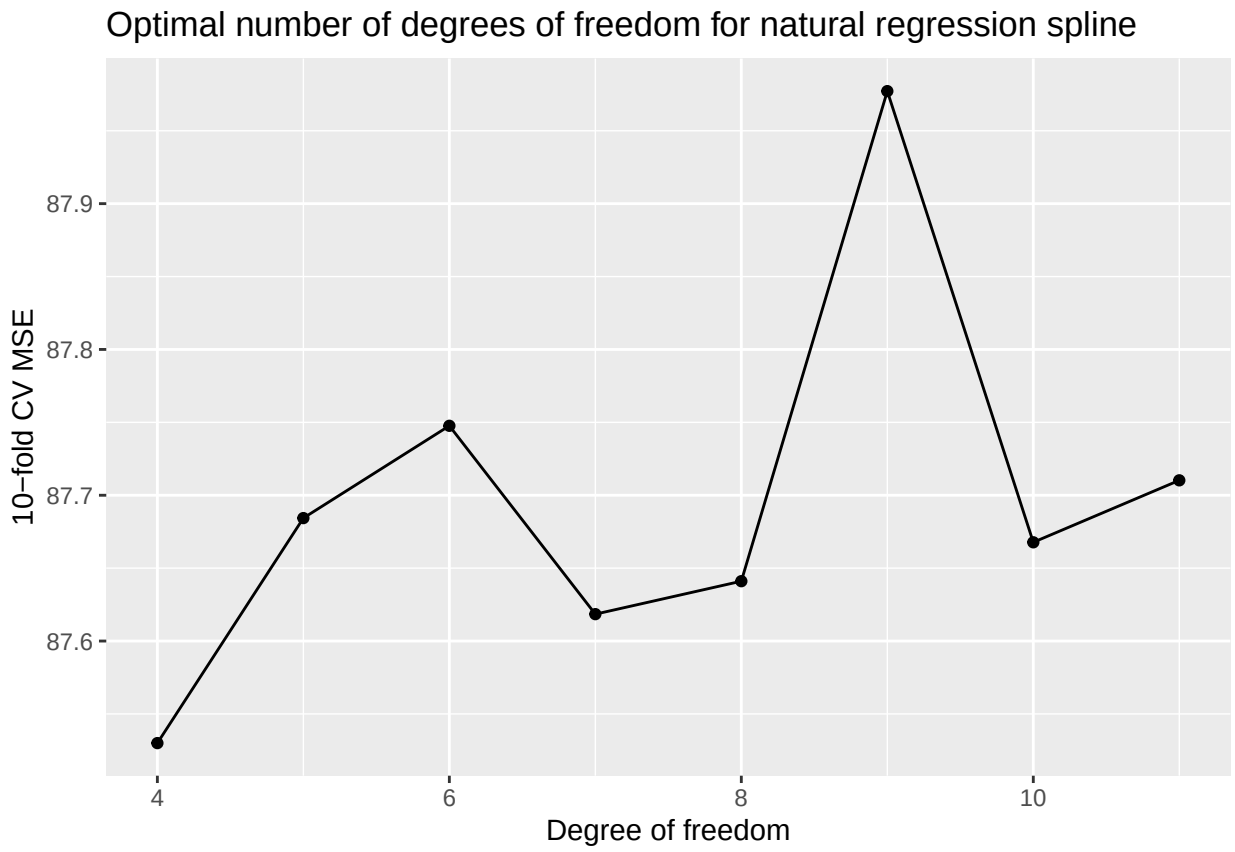
Natural regression spline

```
# set up function
egal_ns_mse <- function(splits, df = NULL){
  # estimate the model on each fold
  model <- glm(egalit_scale ~ ns(income06, df = df),
               data = analysis(splits))
  model_mse <- augment(model, newdata = assessment(splits)) %>%
    mse(truth = egalit_scale, estimate = round(.fitted))
  model_mse$.estimate
}
```

- Degree of freedom = knots + 3

- With a minimum knot of 1, the minimum df for natural regression spline is 4. Df maximum here is 11.

```
#map values to the function
tidyr::expand(training_10fold, id, df = 4:11) %>%
  left_join(training_10fold) %>%
  mutate(mse = map2(splits, df, egal_ns_mse)) %>%
  #use unlist() to extract the mse
  mutate(mse = unlist(mse)) %>%
  group_by(df) %>%
  summarise(mse = mean(mse)) %>%
  ggplot(aes(df, mse)) +
  geom_point() +
  geom_line() +
  labs(title = "Optimal number of degrees of freedom for natural regression spline",
       x = "Degree of freedom",
       y = "10-fold CV MSE")
```

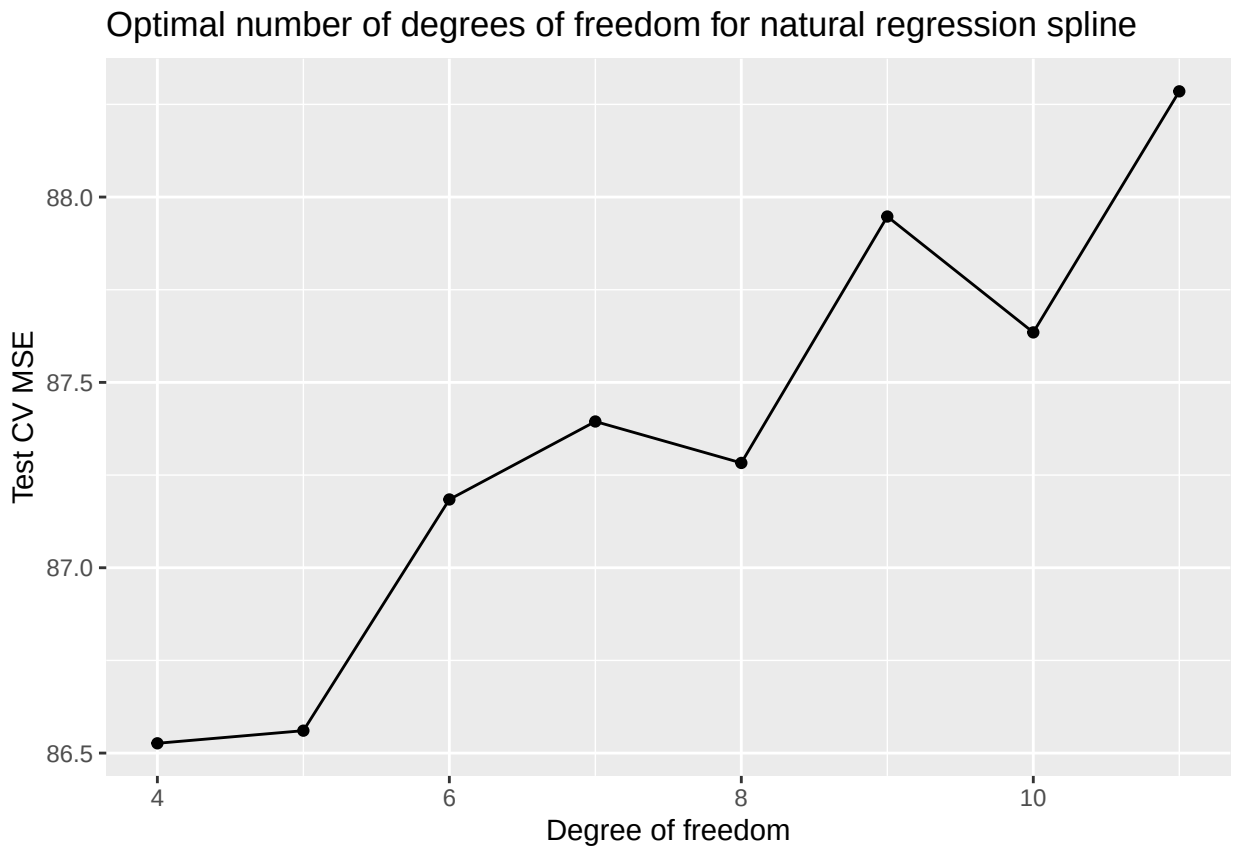


- When we plot the MSE from fitting the models to the training data, it turns out 4 is the optimal degree of freedom. Next to 4 is 7.
- To decide which one is better, we should fit the models to the test data.

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

```
                data = analysis(splits))
  model_mse <- augment(model, newdata = gss_test) %>%
    mse(truth = egalit_scale, estimate = round(.fitted))
  model_mse$.estimate
}
```

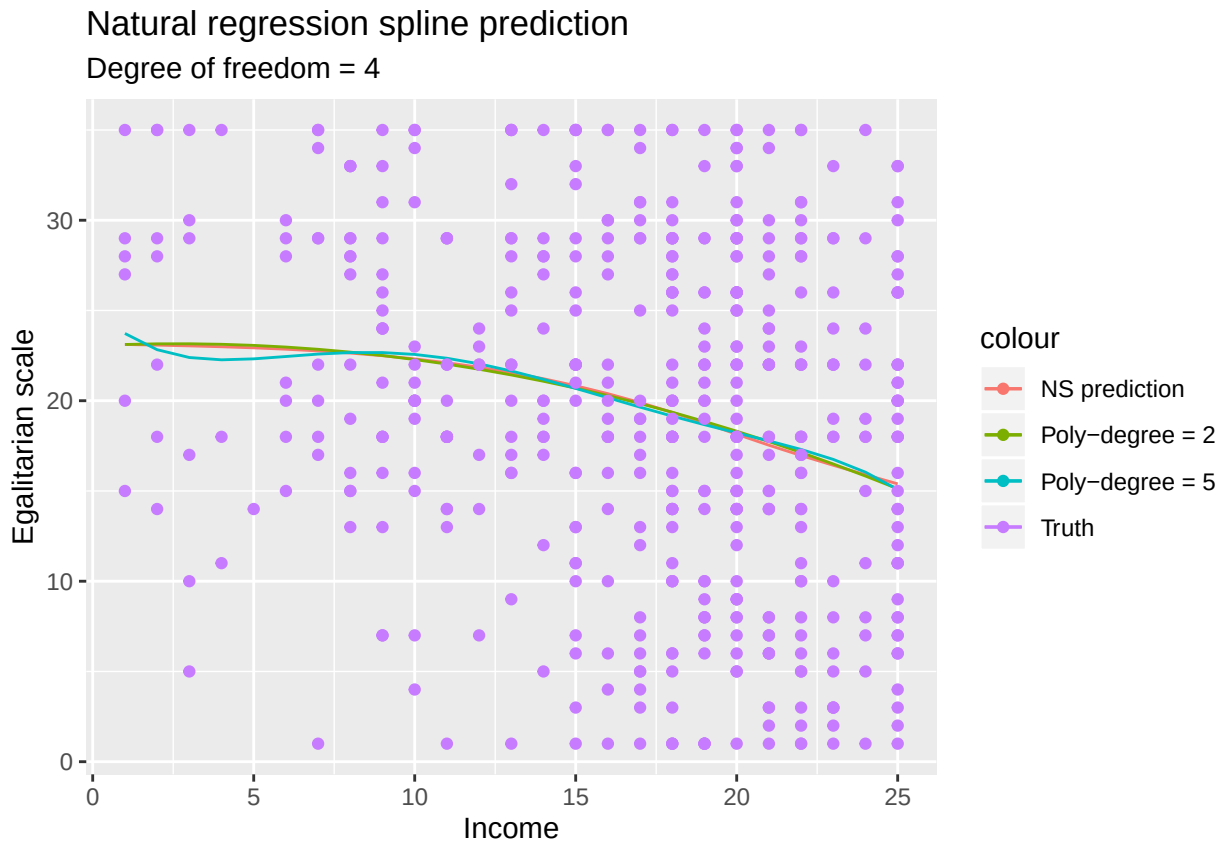
```
#map values to the function
tidyr::expand(training_10fold, id, df = 4:11) %>%
  left_join(training_10fold) %>%
  mutate(mse = map2(splits, df, egal_ns_mse_test)) %>%
  #use unlist() to extract the mse
  mutate(mse = unlist(mse)) %>%
  group_by(df) %>%
  summarise(mse = mean(mse)) %>%
  ggplot(aes(df, mse)) +
  geom_point() +
  geom_line() +
  labs(title = "Optimal number of degrees of freedom for natural regression spline",
       x = "Degree of freedom",
       y = "Test CV MSE")
```



- When the models are fitted to test data, it turns out df = 4 and df = 5 perform much better than the otheres. df = 7 has relatively high MSE here.
- Therefore, we can conclude that the optimal degree of freedom for natural spline regression is 4.

```
egal_ns <- glm(egalit_scale ~ ns(income06, df = 4),
               data = gss_train)
```

```
gss_test %>%
  mutate(pred = predict(egal_ns, newdata = gss_test),
         pred2 = predict(egal_poly2, newdata = gss_test),
         pred5 = predict(egal_poly5, newdata = gss_test)) %>%
  ggplot(aes(x = income06, y = pred)) +
  geom_line(aes(color = "NS prediction")) +
  geom_line(aes(y = pred2, color = "Poly-degree = 2")) +
  geom_line(aes(y = pred5, color = "Poly-degree = 5")) +
  geom_point(aes(y = egalit_scale, color = "Truth")) +
  labs(title = "Natural regression spline prediction",
       subtitle = "Degree of freedom = 4",
       x = "Income",
       y = "Egalitarian scale")
```



It's surprising to find that the prediction curve of degree-2 polynomial regression almost completely overlaps with that of the natural spline regression at df = 4. And naturally, all three prediction curves overlap on many place over the range of income06.

Egalitarianism and everything

```
X <- gss_train %>%
  dplyr::select(-egalit_scale)
Y <- gss_train$egalit_scale

X_cv <- model.matrix(egalit_scale ~ ., data = gss_train)[, -1]

#changed expression from hw3, gss data contained factor variables(?)
X_test <- model.matrix(~. -egalit_scale, data = gss_test)
```

- see here for more detail about the factor issue

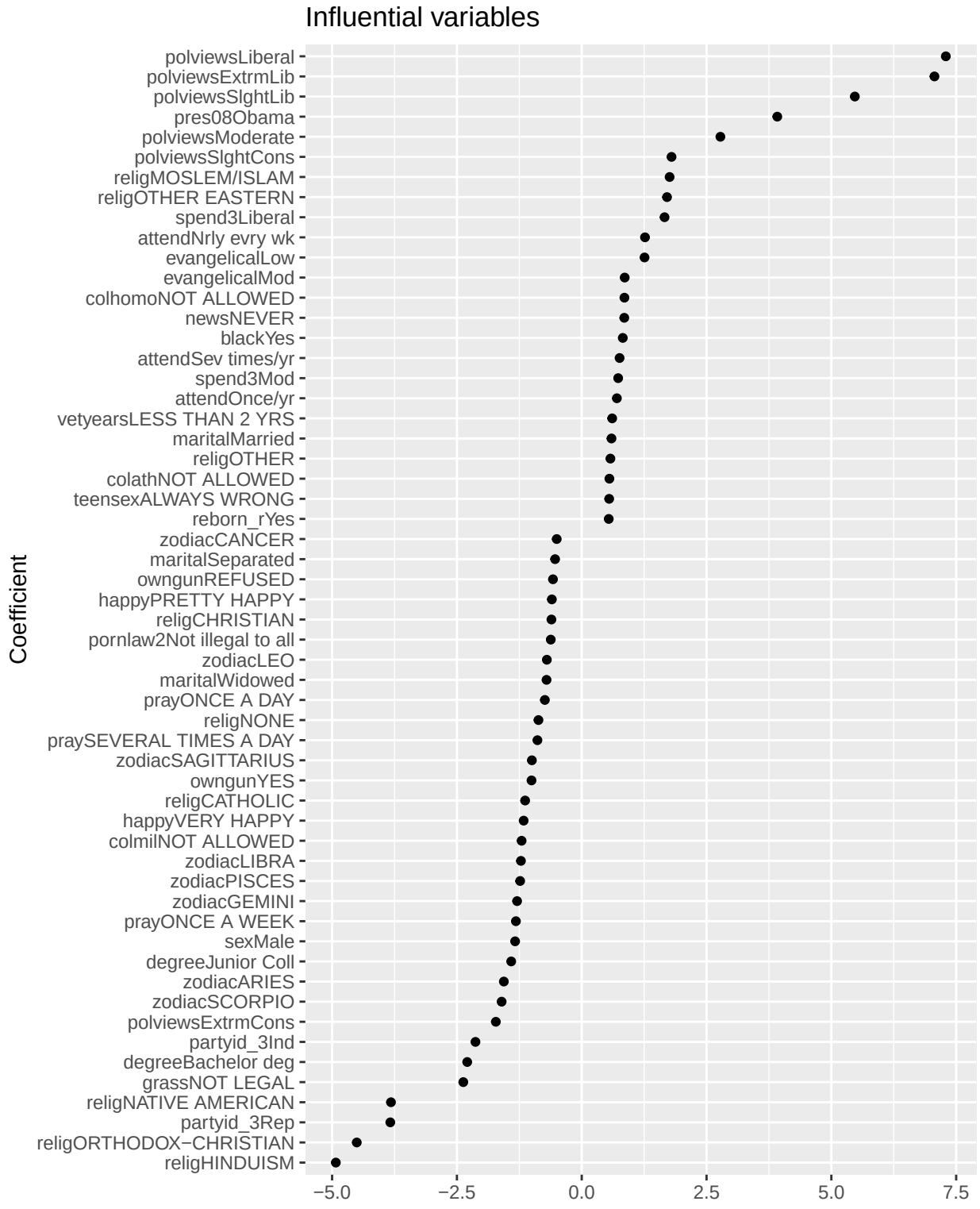
Linear regression

```
#define control method to k-fold CV
cv_10 <- trainControl(method = "cv", number = 10)

#construct model
lm_gss <- train(
  x = X,
  y = Y,
  method = "lm",
  trControl = cv_10
)

#predict
lm_result <- tibble(truth = gss_test$egalit_scale,
                    pred = predict(lm_gss, newdata = gss_test))

#calculate MSE
lm_result %>%
  mse(truth = truth, estimate = pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 mse     standard        63.4
```

```
#need to select the final model before processing with coef()
coef_lm <- coef(lm_gss$finalModel) %>%
  tidy()
#change names for better operation
colnames(coef_lm) <- c("coefficient", "coef_value")
coef_lm %>%
  #too many coefficients were included
  #drop the ones not big enough
  dplyr::filter(coef_value >= 0.5 |
                  coef_value <= -0.5) %>%
```

```
  dplyr::filter(coefficient != "(Intercept)") %>%
  ggplot(aes(x = reorder(coefficient, coef_value),
             y = coef_value)) +
  geom_point() +
  coord_flip() +
  labs(title = "Influential variables",
       x = "Coefficient",
       y = NULL)
```



- A simple linear regression (lm) would put every available feature into it, even though many of them are hardly relevant. As we can see here, for variables with categorical values, each value is taken out as a feature.
- In particular, political view (polviews) is a very strong predictor for egalitarian score. Being liberal or the extremely libereal is strongly associated with higher egalitarian score, while being extremely

conservative is strongly associated with lower egalitarian score.

- In contrast, religion turns out to be a very strong negative predictor for egalitarian score. Hinduism and Orthodox-Christianity appear to be the two strongest negative feature here, as opposed to liberal and extremely liberal being the two strongest positive.
- Interestingly, Moslem and slightly conservative political views are both positive predictors to egalitarian score, despite people's common idea that they indicate less egalitarian characters.
- On the other hand,college degree is a strong negative predictor. In common sense, college students and graduates are more liberal than those with lower degrees. How comes this result?
- It's sort of a stereotypical cliche that Republicans are less egalitarian than others. And the model agrees with it.

Elastic net

```
#randomize which fold among the 10 to use for training
folds <- sample(1:10, size = length(Y), replace = TRUE)

#create a grid for entries of lambda and MSE
#using lambda.1se here as in the previous two models
tuning <- tibble(alpha = seq(0, 1, by = .1),
                 mse_1se = NA,
                 lambda_1se = NA
                 )

#fill in the grid with values
#for each alpha, there is a corresponding lambda
for(i in seq_along(tuning$alpha)){
  filling <- cv.glmnet(x = X_cv,
                   y = Y,
                   alpha = tuning$alpha[i],
                   foldid = folds)
  tuning$mse_1se[i] <- filling$cvm[filling$lambda == filling$lambda.1se]
  tuning$lambda_1se[i] <- filling$lambda.1se
}

#create another grid for the calculated lambda and possible alpha
testing <- expand.grid(alpha = seq(0, 1, by = 0.1),
                       lambda_1se = tuning$lambda_1se)
```

```
#a grid for all models and their testing MSE
models_enet <- tibble(alpha = testing$alpha,
                      lambda_1se = testing$lambda_1se,
                      MSE = NA)

#use a function to make predictions
very_fancy <- function(al, bi){
  #reproduce the model with corresponding alpha
  dumb <- glmnet(
    x = X_cv,
    y = Y,
    alpha = al
    )
  mid_filler <- tibble(truth = gss_test$egalit_scale,
```

```
                  pred = predict(dumb,
                                 s = bi,
                                 newx = X_test)) %>%
    mutate(truth = as.numeric(truth)) %>%
    mutate(pred = as.numeric(pred))
  #extract the MSE from the resulting form
  MSE_filler <- mid_filler %>%
    mse(truth = truth,
        estimate = pred)
  MSE_filler$.estimate
}
```

```
for(i in seq_along(models_enet$alpha)) {
  models_enet$MSE[i] <- very_fancy(models_enet$alpha[i],
                                   models_enet$lambda_1se[i])
}

models_enet
```

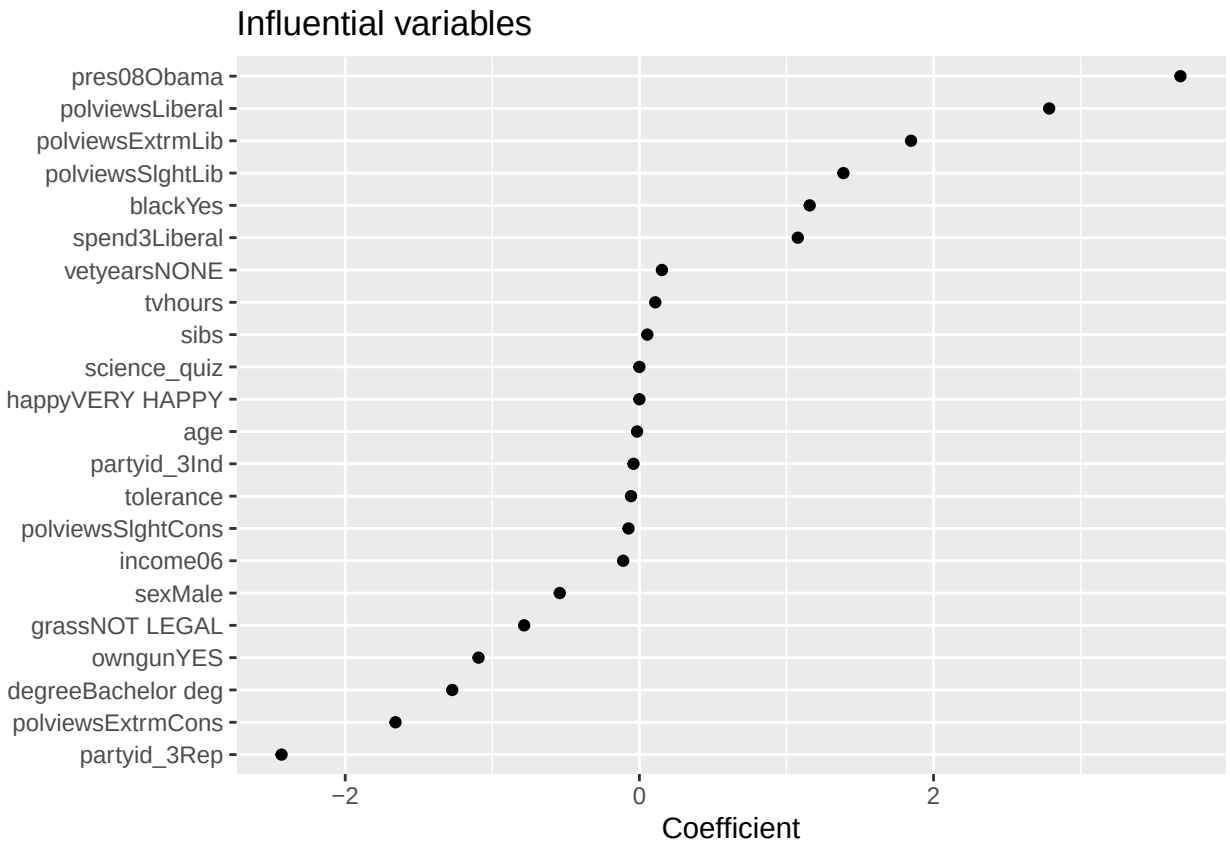
```
## # A tibble: 121 x 3
##    alpha lambda_1se   MSE
##    <dbl>      <dbl> <dbl>
##  1   0         9.88 116.
##  2   0.1       9.88  92.1
##  3   0.2       9.88  94.5
##  4   0.3       9.88  92.0
##  5   0.4       9.88  90.6
##  6   0.5       9.88  90.5
##  7   0.6       9.88  90.5
##  8   0.7       9.88  90.5
##  9   0.8       9.88  90.5
## 10   0.9       9.88  90.5
## # ... with 111 more rows
```

```
#find the combination where MSE is smallest
models_enet[which.min(models_enet$MSE),]
```

```
## # A tibble: 1 x 3
##   alpha lambda_1se   MSE
##   <dbl>      <dbl> <dbl>
## 1   0.1       2.04  90.1
```

```
#define regression model
best_enet <- glmnet(
  x = X_cv,
  y = Y,
  alpha = 0.1
)

#extract coefficients
coef(best_enet, s = 4.47891) %>%
  tidy() %>%
```

```
  dplyr::filter(row != "(Intercept)") %>%
  #draw the plot
  ggplot(aes(value, reorder(row, value))) +
  geom_point() +
  labs(title = "Influential variables",
       x = "Coefficient",
       y = NULL)
```

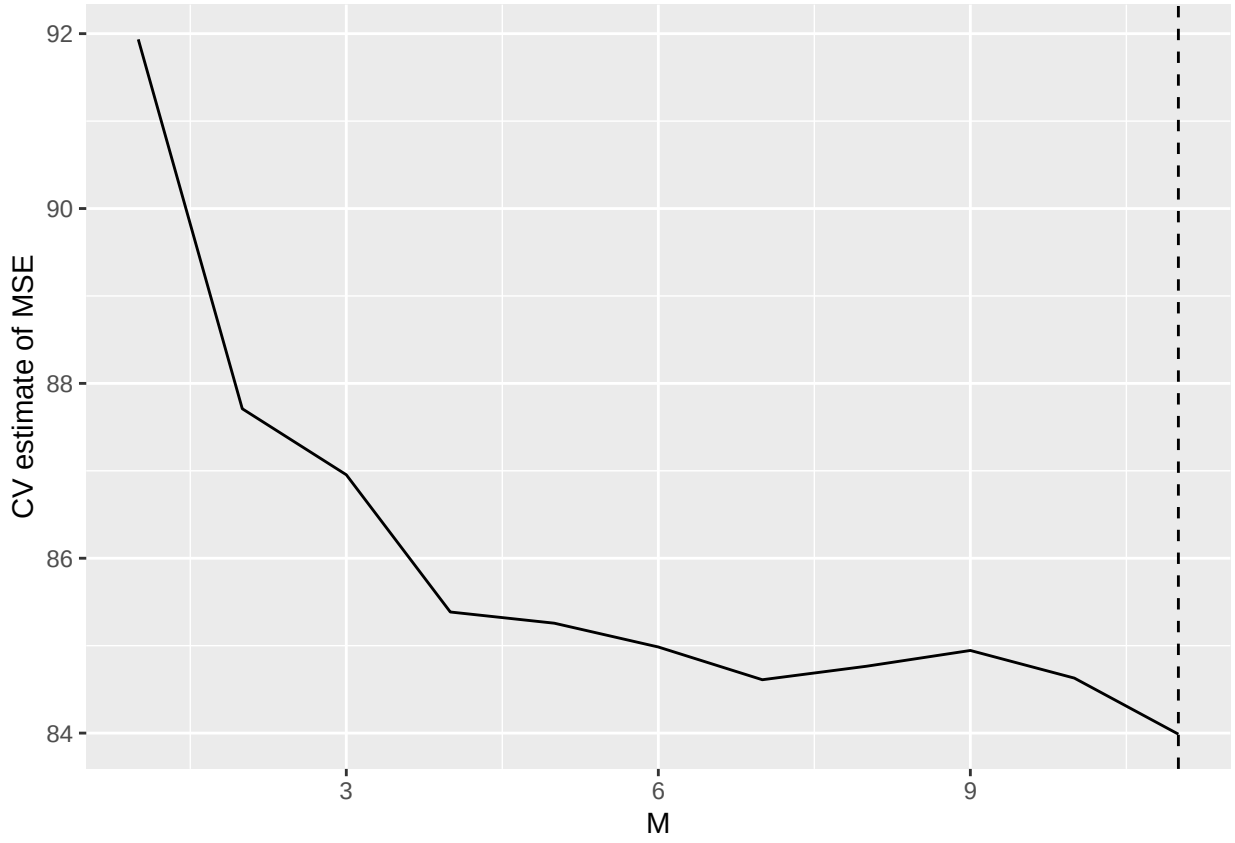


- Much fewer variables are taken in an elastic net regression model, compared with a linear regression.
- The elastic net model echoes with the linear regression model that political views, college degree, and party membership are strong indicators of egalitarian scores.
- Being liberal continues to be positively associated with egalitarian scores, while being extremely conservative, holding a bachelor's degree, and being a Republican continue to be negatively associated.
- It should be noted, however, that the absolute values of the coefficients in the elastic model are generally smaller than their counterparts in the linear regression model, even though they are in the same directions.
- Somehow voting for Obama is a strongly positive predictor both here and in the linear regression model. This is weird – we don't see Democrat as a feature in both models. Perhaps voting for Obmama is more about being liberal than about being a Democrat. If this is true, poor Republicans will have one more evidence of their being conservative.

Principal component regression

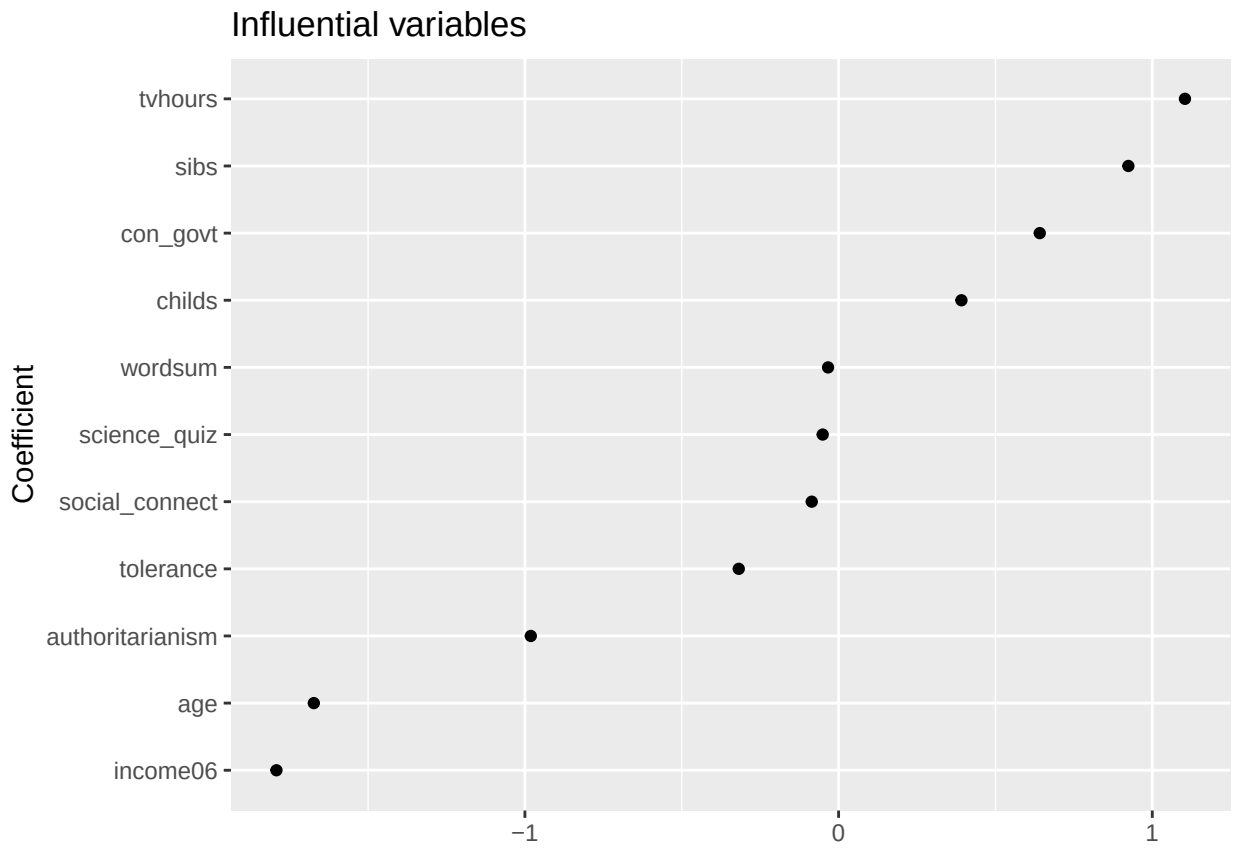
```
#construct model
gss_pcr <- pcr(egalit_scale ~ .,
               #select only numeric variables
               data = dplyr::select_if(gss_train, is.numeric),
               scale = TRUE,
               validation = "CV")
```

```
#extract stats from the model
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)

#graph the mse corresponding to each value of pc
gss_pcr_stats %>%
  ggplot(aes(x = pc, y = mse)) +
  geom_line() +
  geom_vline(xintercept = which.min(gss_pcr_stats$mse), linetype = 2) +
  labs(x = expression(M),
       y = "CV estimate of MSE")
```



The optimal number of principal components appears to be 11, according to the graph and the models' MSE.

```
#coef automatically selects pc = 11, where mse is min
#the output of coef here is an array
#use adply from plyr to transform it
pcr_coef <- plyr::adply(coef(gss_pcr), c(1,2,3))
colnames(pcr_coef) <- c("coefficient", "outcome", "pc", "coef_value")
#draw the plot
pcr_coef %>%
  ggplot(aes(x = reorder(coefficient, coef_value),
             y = coef_value)) +
  geom_point() +
  coord_flip() +
  labs(title = "Influential variables",
       x = "Coefficient",
       y = NULL)
```



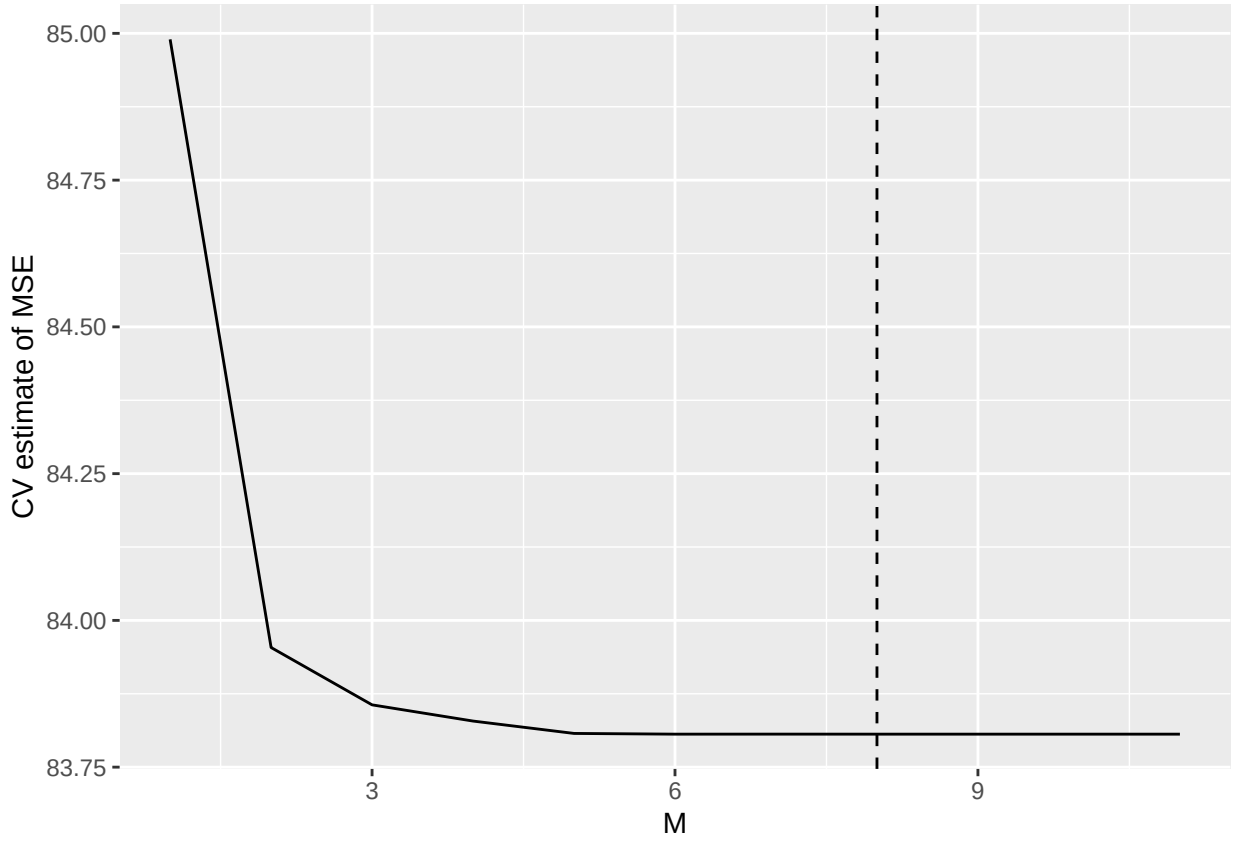
- Even fewer coefficients are included in a PCR model.
- More importantly, due to the natura of PCR, categorical variables cannot be inlcuded in the model (unless we trace back the categories to GSS original codes)
- The absolute values of these coefficients are all smaller than 2 – smaller than those from the elastic net model.
- Although it's understandable that having siblings may help one be egalitarian, it's hard to understand how the hours of one spends watching TV can be associated with egalitarianism.
- The negative coefficients makes more sense here: higher income, older age, and stronger authoritarian

characters are often associated with weaker egalitarianism in common sense. And our previous attempts with income mostly agrees with income being a negative predictor (especially high incomes).

Partial least squares regression

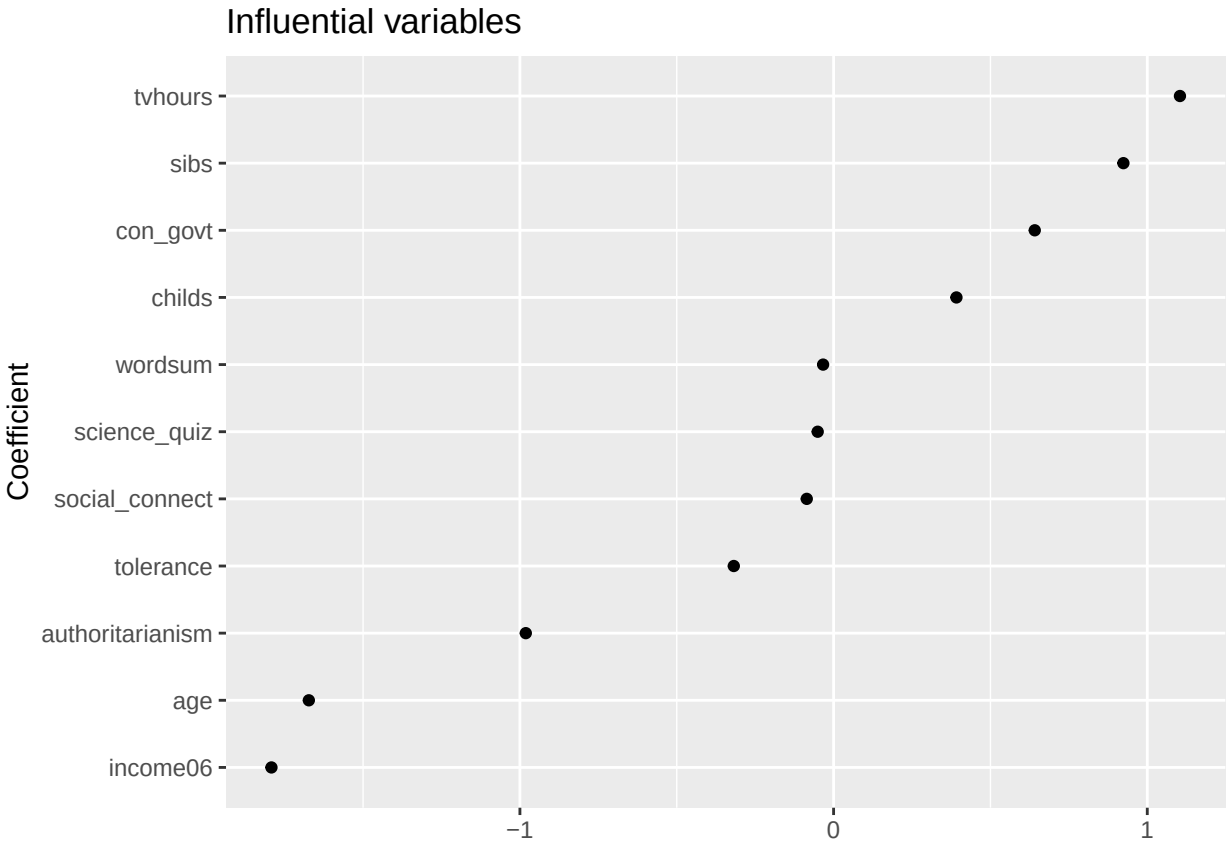
```
gss_pls <- plsr(egalit_scale ~ .,
                #select only numeric variables
                data = dplyr::select_if(gss_train, is.numeric),
                scale = TRUE,
                validation = "CV")
```

```
# 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)
) %>%
  dplyr::mutate(pc = row_number(),
         cum_exp = cumsum(pct_exp) / 100)

#graph the mse corresponding to each value of pc
gss_pls_stats %>%
  ggplot(aes(x = pc, y = mse)) +
  geom_line() +
  geom_vline(xintercept = which.min(gss_pls_stats$mse), linetype = 2) +
  labs(x = expression(M),
       y = "CV estimate of MSE")
```



For PLS, the graph indicates MSE has a minimum at pc = 8.

```
#since coef() gives only the coefficients when pc = 11
#hand-select the coefficients for pc = 8
pls_coef <- plyr::adply(gss_pls$coefficients, c(1,2,3))
colnames(pls_coef) <- c("coefficient", "outcome", "pc", "coef_value")
#draw the plot
pls_coef %>%
  dplyr::filter(pc == "8 comps") %>%
  ggplot(aes(x = reorder(coefficient, coef_value),
             y = coef_value)) +
  geom_point() +
  coord_flip() +
  labs(title = "Influential variables",
       x = "Coefficient",
       y = NULL)
```



- Again, TV hours appears to be the strongest positive indicator, followed by sibilings and confidence in the government.
- It's interesting that having confidence in the government is a positive indicator in both PRC and PLS. How can egalitarianism and confidence in the government be related? Because having faith in the government makes one more willing to hold on to the "American values"?
- And income, age, authoritarianism continue to be strongest negative predictors in the PLS model.
- In fact, if we compare this graph with the graph for PCR, we will realize they are identical A reason why neither PCR or PLS is better than the other?