Problem Set 4

MACS 30100 Winter 2020

Nak Won Rim

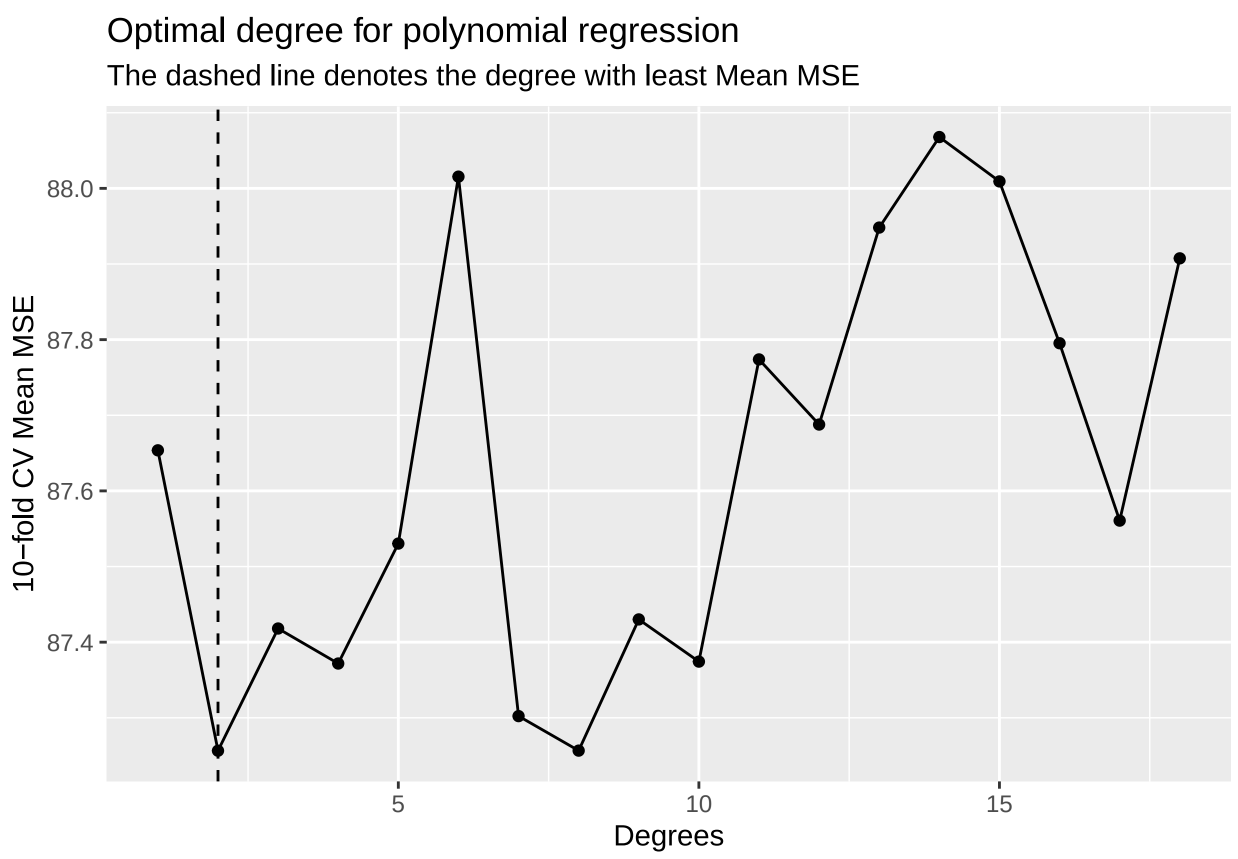
Egalitarianism and income

1. Perform polynomial regression to predict `egalit_scale` as a function of `income06`. Use and plot 10-fold cross-validation to select the optimal degree d for the polynomial based on the MSE. Plot the resulting polynomial fit to the data, and also graph the average marginal effect (AME) of `income06` across its potential values. Be sure to provide substantive interpretation of the results.

```r
set.seed(23)
gss_train <- read.csv("data/gss_train.csv")
gss_test <- read.csv("data/gss_test.csv")

tuning_df1 <- data.frame(matrix(NA, nrow=18, ncol=2))
colnames(tuning_df1) <- c("degree", "CV_MSE")
for (i in 1:18){
  tuning_df1$degree[i] <- i
  # using boot pakcage's cv.glm to get the CV MSE
  mod <- glm(egalit_scale ~ poly(income06, i), data = gss_train)
  # 10-fold CV
  tuning_df1$CV_MSE[i] <- boot::cv.glm(gss_train, mod, K=10)$delta[1]
}

ggplot(tuning_df1, aes(degree, CV_MSE)) +
  geom_point() +
  geom_line()+
  geom_vline(xintercept = which.min(tuning_df1$CV_MSE), linetype = 2) +
  labs(title = "Optimal degree for polynomial regression",
       subtitle = "The dashed line denotes the degree with least Mean MSE",
       x = "Degrees",
       y = "10-fold CV Mean MSE")
```



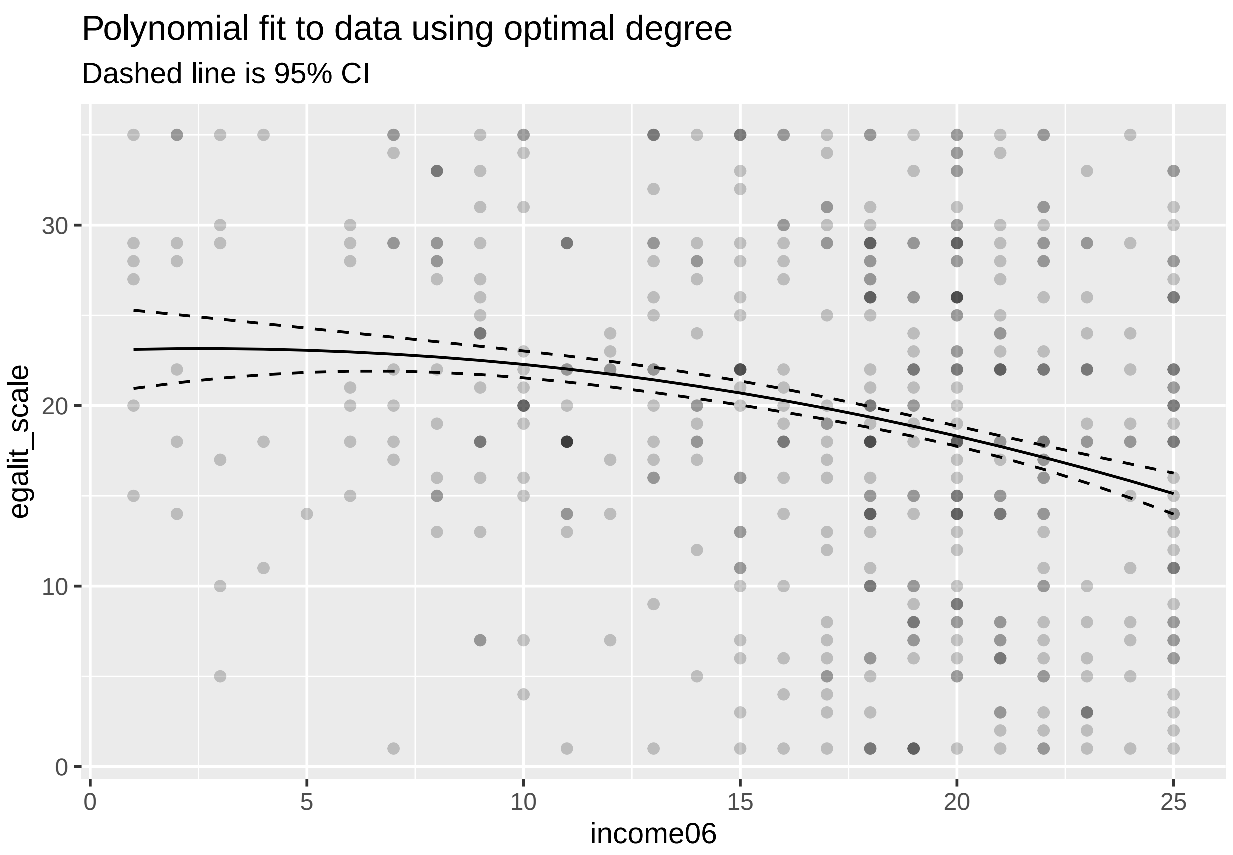
```r
min(tuning_df1$CV_MSE) ^ 0.5
```

```
## [1] 9.341118
```

From the above plot, we can see that the optimal degree for the polynomial regression based on the MSE is 2. Following is the polynomial fit to the data using $d = 2$.

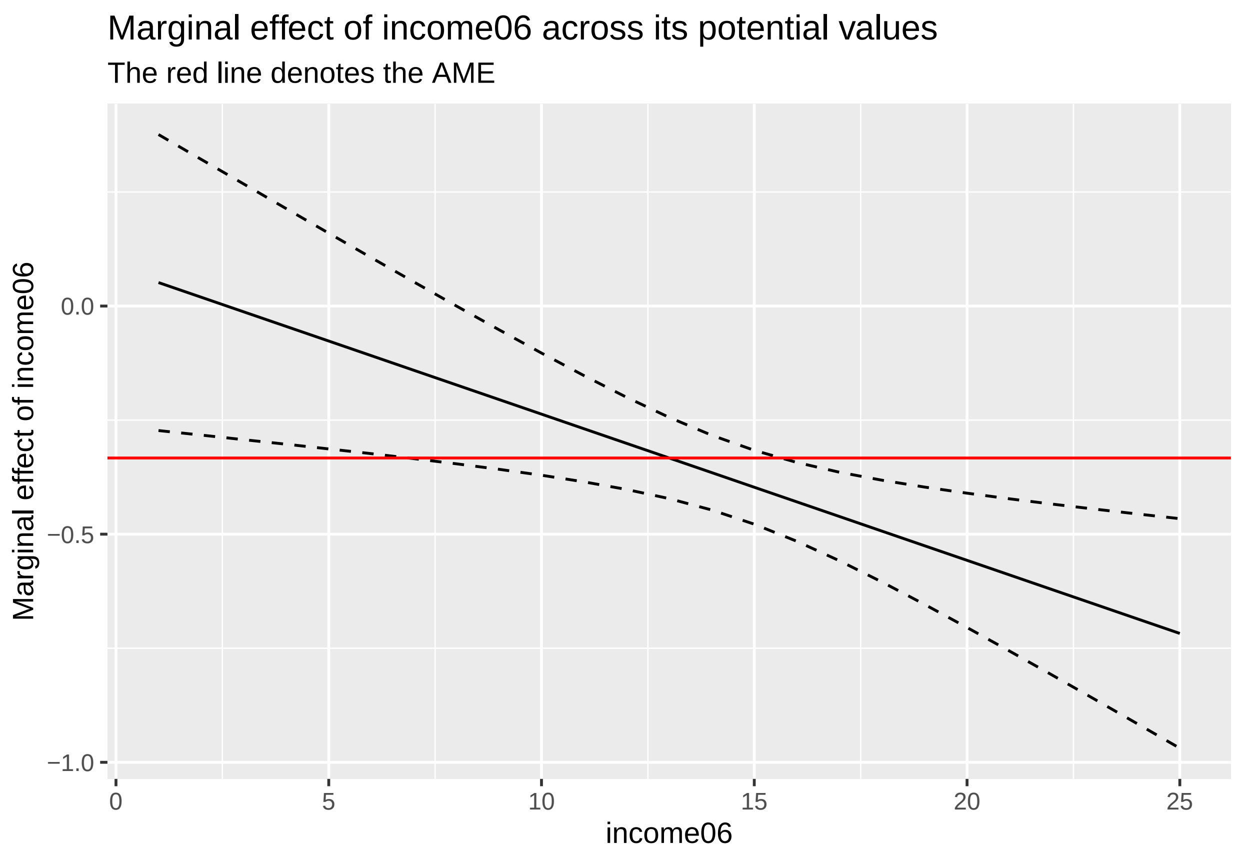
```r
glm_egalit <- glm(egalit_scale ~ stats::poly(income06, 2), data = gss_train)
margins::cplot(glm_egalit, "income06", what = "prediction", draw = FALSE) %>%
  ggplot(aes(x = xvals)) +
  geom_line(aes(y = yvals)) +
  geom_line(aes(y = upper), linetype = 2) +
  geom_line(aes(y = lower), linetype = 2) +
  geom_point(aes(x=income06, y=egalit_scale), gss_test, alpha=0.2) +
  labs(title = "Polynomial fit to data using optimal degree",
       subtitle = "Dashed line is 95% CI",
       x = "income06",
       y = "egalit_scale")
```

```
##    xvals    yvals    upper    lower
## 1      1 23.11631 25.28371 20.94890
## 2      2 23.15180 25.04001 21.26359
## 3      3 23.15524 24.79232 21.51817
## 4      4 23.12665 24.54195 21.71134
## 5      5 23.06600 24.29036 21.84165
## 6      6 22.97331 24.03899 21.90764
## 7      7 22.84858 23.78870 21.90846
## 8      8 22.69180 23.53894 21.84467
## 9      9 22.50298 23.28678 21.71917
## 10    10 22.28211 23.02670 21.53752
## 11    11 22.02920 22.75132 21.30708
## 12    12 21.74424 22.45297 21.03552
## 13    13 21.42724 22.12502 20.72946
## 14    14 21.07819 21.76262 20.39376
## 15    15 20.69710 21.36290 20.03130
## 16    16 20.28396 20.92501 19.64291
## 17    17 19.83878 20.45044 19.22712
## 18    18 19.36155 19.94356 18.77955
## 19    19 18.85228 19.41247 18.29210
## 20    20 18.31096 18.86919 17.75274
```



Following plot shows the marginal effect of income06 when using polynomial regression with $d = 2$:

```
margin_plot <- margins::cplot(glm_egalit, "income06", what = "effect", draw = FALSE)
ggplot(margin_plot, aes(x = xvals)) +
  geom_line(aes(y = yvals)) +
  geom_line(aes(y = upper), linetype = 2) +
  geom_line(aes(y = lower), linetype = 2) +
  geom_hline(yintercept= mean(margin_plot$yvals), color="red") +
  labs(title = "Marginal effect of income06 across its potential values",
       subtitle = "The red line denotes the AME",
       x = "income06",
       y = "Marginal effect of income06")
```



```
mean(margin_plot$yvals)
```

```
## [1] -0.3330249
```

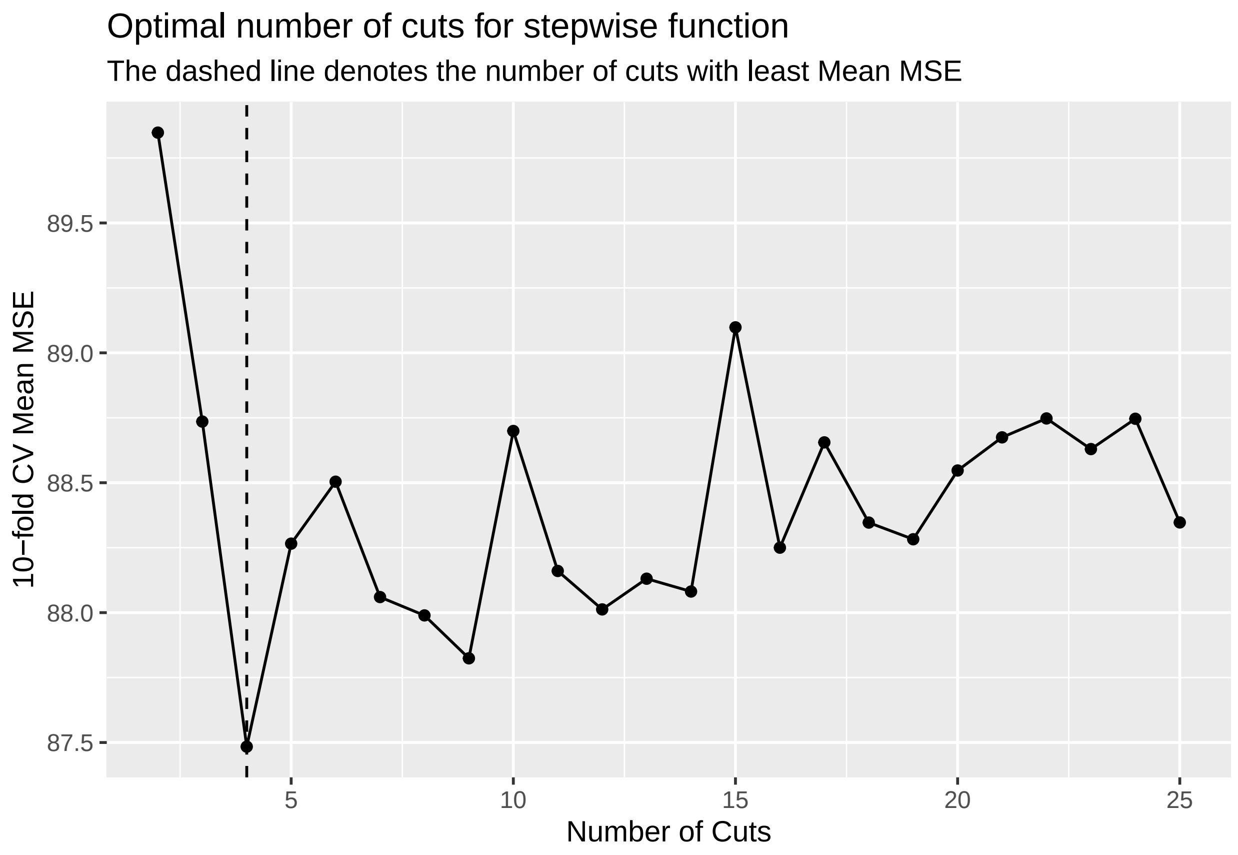
The optimal degree d is 2 for the polynomial regression. Therefore, when we fit the polynomial regression model predicting `egalit_scale` using `income06`, the shape of the fit is curvilinear. We can see from the second plot that the predicted `egalit_scale` becomes smaller as `income06` gets larger. One thing to notice is that the confidence interval of the prediction seems to become wider as it gets to the edge of the `income06`. Also, we can see that the slope of the function seems to get bigger as the `income06` gets larger. This also nicely shown in the third plot, where the marginal effect of `income06` gets bigger in absolute values. Overall, `income06` is showing a negative marginal effect to `egalit_scale` in almost all values, with average marginal effect (AME) being about - 0.33 (shown as the red line in the third plot)

Although we found the optimal degree using cross-validation and used that degree to fit the polynomial regression, the CV RMSE is around 9.34. I do not think this is not great considering that `egalit_scale` ranges from 1 to 35. The reason behind this could be that `income06` does not explain `egalit_scale` a lot.

2. Fit a step function to predict `egalit_scale` as a function of `income06`, and perform 10-fold cross-validation to choose the optimal number of cuts. Plot the fit and interpret the results.

```r
set.seed(72)
tuning_df2 <- data.frame(matrix(NA, nrow=24, ncol=2))
colnames(tuning_df2) <- c("num_cut", "CV_MSE")
for (i in 2:25){
  tuning_df2$num_cut[i - 1] <- i
  # add a column to gss_train to fix the interval
  gss_train$income06_interval <- cut_interval(gss_train$income06, i)
  # using boot pakcage's cv.glm to get the CV MSE
  mod <- glm(egalit_scale ~ income06_interval, data = gss_train)
  # 10-fold CV
  tuning_df2$CV_MSE[i-1] <- boot::cv.glm(gss_train, mod, K=10)$delta[1]
}

ggplot(tuning_df2, aes(num_cut, CV_MSE)) +
  geom_point() +
  geom_line() +
  # + 1 because the cuts starts at 2 not 1
  geom_vline(xintercept = which.min(tuning_df2$CV_MSE) + 1, linetype = 2) +
  labs(title = "Optimal number of cuts for stepwise function",
       subtitle = "The dashed line denotes the number of cuts with least Mean MSE",
       x = "Number of Cuts",
       y = "10-fold CV Mean MSE")
```



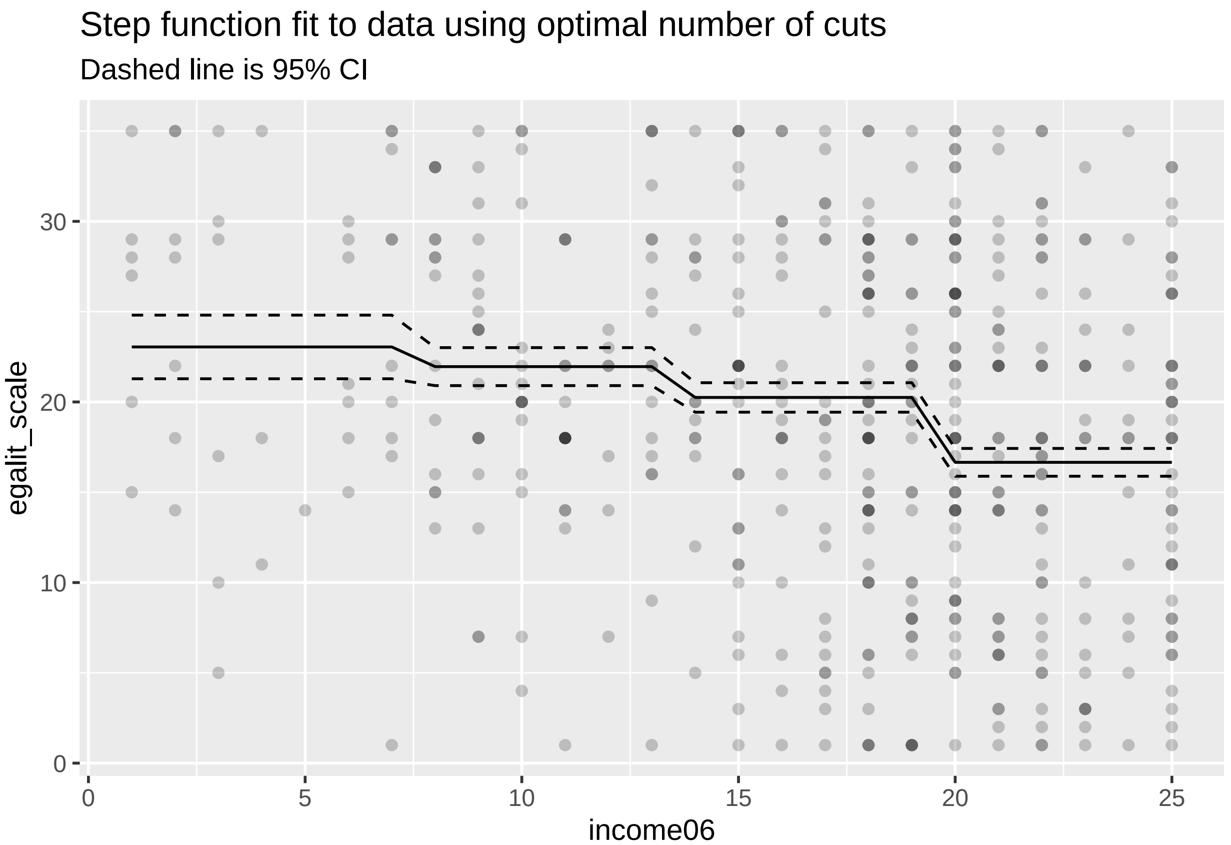
```r
min(tuning_df2$CV_MSE) ^ 0.5
```

```
## [1] 9.353291
```

From the above plot, we can see that the optimal number of cuts for the step function in terms of MSE is 4. Following is the step function fit to the data using 4 as the number of cuts:

```r
step_egalit <- glm(egalit_scale ~ cut_interval(as.numeric(income06), 4), data = gss_train)
margins::cplot(step_egalit, "income06", what = "prediction", draw = FALSE) %>%
  ggplot(aes(x = as.numeric(xvals))) +
  geom_line(aes(y = yvals)) +
  geom_line(aes(y = upper), linetype = 2) +
  geom_line(aes(y = lower), linetype = 2) +
  geom_point(aes(x=income06, y=egalit_scale), gss_test, alpha=0.2) +
  labs(title = "Step function fit to data using optimal number of cuts",
       subtitle = "Dashed line is 95% CI",
       x = "income06",
       y = "egalit_scale")
```

```
##    xvals    yvals    upper    lower
## 1     25 16.65724 17.42707 15.88742
## 2     23 16.65724 17.42707 15.88742
## 3     19 20.25000 21.06580 19.43420
## 4     16 20.25000 21.06580 19.43420
## 5      5 23.04630 24.80863 21.28396
## 6      1 23.04630 24.80863 21.28396
## 7      8 21.95710 23.00925 20.90494
## 8     14 20.25000 21.06580 19.43420
## 9     18 20.25000 21.06580 19.43420
## 10    10 21.95710 23.00925 20.90494
## 11    22 16.65724 17.42707 15.88742
## 12    24 16.65724 17.42707 15.88742
## 13    21 16.65724 17.42707 15.88742
## 14    20 16.65724 17.42707 15.88742
## 15    17 20.25000 21.06580 19.43420
## 16    15 20.25000 21.06580 19.43420
## 17    13 21.95710 23.00925 20.90494
## 18     6 23.04630 24.80863 21.28396
## 19    12 21.95710 23.00925 20.90494
## 20     9 21.95710 23.00925 20.90494
```



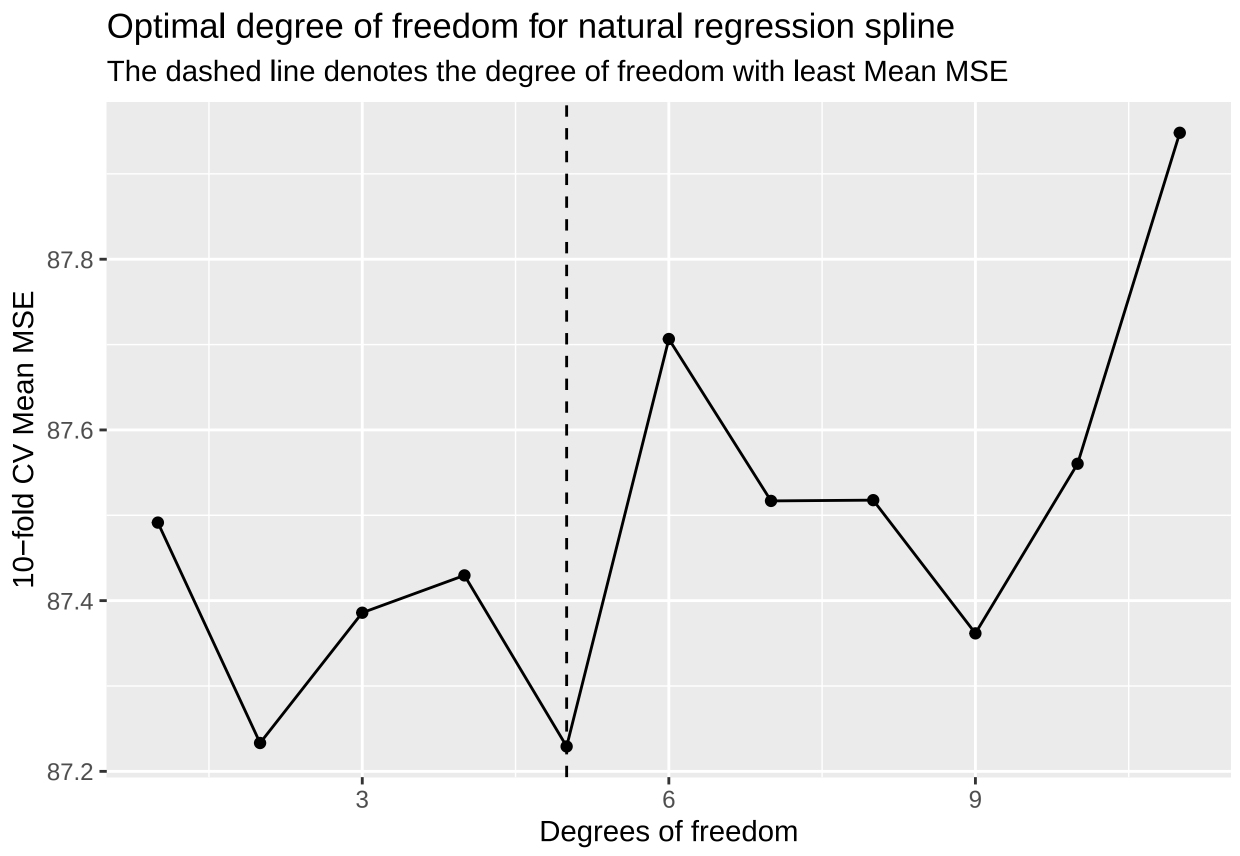
The optimal number of cuts for the step function is 4. As we can see from the second plot, the general trend is the same as the polynomial regression plot - as `income06` increases, `egalit_scale` decreases. However, we can see that the function is much less smooth than the polynomial regression since we are basically plotting four constant functions over `income06`. Also, another thing to note is that the confidence interval stays constant across one cut, but it differs between cuts.

Analogous to the polynomial regression, the CV RMSE (9.35) is quite high even though we optimized the number of cuts. This could be another clue that `income06` does not explain `egalit_scale` that much.

3. Fit a natural regression spline to predict `egalit_scale` as a function of `income06`. Use 10-fold cross-validation to select the optimal number of degrees of freedom, and present the results of the optimal model.

```r
set.seed(119)
tuning_df3 <- data.frame(matrix(NA, nrow=11, ncol=2))
colnames(tuning_df3) <- c("df", "CV_MSE")
for (i in 1:11){
  tuning_df3$df[i] <- i
  # using boot pakcage's cv.glm to get the CV MSE
  mod <- glm(egalit_scale ~ ns(income06, i), data = gss_train)
  # 10-fold CV
  tuning_df3$CV_MSE[i] <- boot::cv.glm(gss_train, mod, K=10)$delta[1]
}

ggplot(tuning_df3, aes(df, CV_MSE)) +
  geom_point() +
  geom_line()+
  geom_vline(xintercept = which.min(tuning_df3$CV_MSE), linetype = 2) +
  labs(title = "Optimal degree of freedom for natural regression spline",
       subtitle = "The dashed line denotes the degree of freedom with least Mean MSE",
       x = "Degrees of freedom",
       y = "10-fold CV Mean MSE")
```



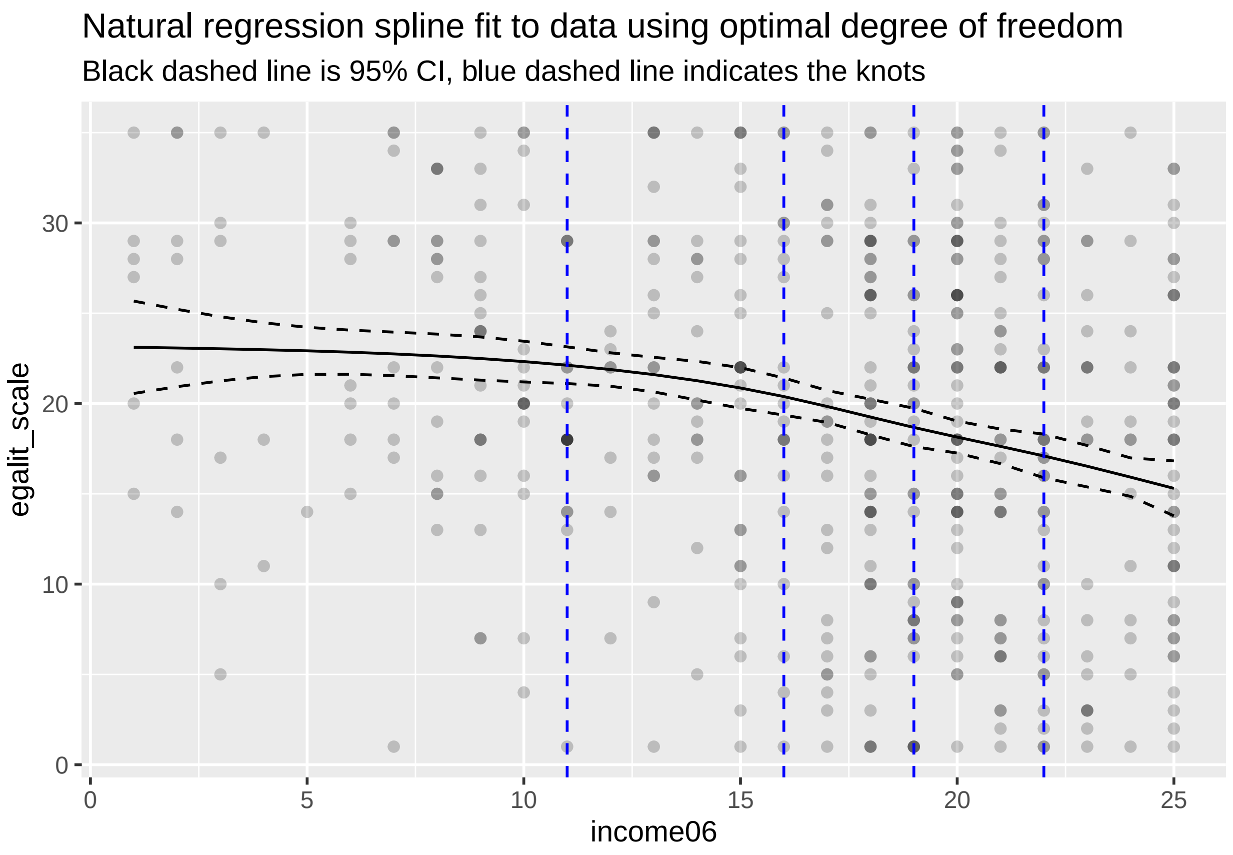
```r
min(tuning_df3$CV_MSE) ^ 0.5
```

```
## [1] 9.339655
```

From the above plot, we can see that the optimal degrees of freedom for the natural regression spline based on MSE is 5. Following is the natural regression spline fit to the data using 5 as degrees of freedom:

```r
ns_egalit <- glm(egalit_scale ~ ns(income06, df=5), data = gss_train)
margins::cplot(ns_egalit, "income06", what = "prediction", draw = FALSE) %>%
  ggplot(aes(x = as.numeric(xvals))) +
  geom_line(aes(y = yvals)) +
  geom_line(aes(y = upper), linetype = 2) +
  geom_line(aes(y = lower), linetype = 2) +
  geom_point(aes(x=income06, y=egalit_scale), gss_test, alpha=0.2) +
  geom_vline(xintercept = attr(ns(gss_train$income06, df = 5), "knots"),
             linetype = 2, color = "blue") +
  labs(title = "Natural regression spline fit to data using optimal degree of freedom",
       subtitle = "Black dashed line is 95% CI, blue dashed line indicates the knots",
       x = "income06",
       y = "egalit_scale")
```

```
##    xvals    yvals    upper    lower
## 1      1 23.11502 25.67307 20.55698
## 2      2 23.07465 25.21603 20.93327
## 3      3 23.03070 24.80668 21.25472
## 4      4 22.97960 24.46878 21.49043
## 5      5 22.91778 24.22145 21.61410
## 6      6 22.84165 24.06196 21.62134
## 7      7 22.74764 23.95448 21.54080
## 8      8 22.63218 23.84529 21.41907
## 9      9 22.49169 23.68688 21.29650
## 10    10 22.32259 23.45054 21.19465
## 11    11 22.12132 23.13708 21.10556
## 12    12 21.88331 22.80874 20.95788
## 13    13 21.60014 22.55723 20.64304
## 14    14 21.26238 22.33299 20.19177
## 15    15 20.86063 21.98750 19.73375
## 16    16 20.38547 21.41224 19.35869
## 17    17 19.83674 20.72022 18.95325
## 18    18 19.25126 20.23740 18.26512
## 19    19 18.67512 19.73664 17.61359
## 20    20 18.14083 19.02482 17.25685
```



According to the 10-fold cross-validation, the optimal degree of freedom is 5 for a natural regression spline. The second plot shows the fitted natural spline for the data, One thing to notice is that the fitted natural regression spline looks very similar to the polynomial regression with $d = 2$, although we are fitting the regression by the knots now. This could be interpreted that there is not much difference in the fitted model between the splines. It might be better to just use the polynomial regression since that is less complex and more interpretable. Another thing to notice is that the confidence interval gets wider at the edges, similar to the fitted polynomial regression.

Also, the CV RMSE was around 9.34, also very close to the CV RMSE of polynomial regression. This intuitively makes sense because the fitted natural regression spline looks very much like the 2nd-degree polynomial regression fit. This lack of performance enhancement could be another reason to just use simpler polynomial regression.

Egalitarianism and everything

4. Estimate the following models using all the available predictors (be sure to perform appropriate data pre-processing (e.g., feature standardization) and hyperparameter tuning (e.g. lambda for PCR/PLS, lambda and alpha for elastic net). Also use 10-fold cross-validation for each model to estimate the model's performance using MSE:

```
## reloading gss_train because I added a column in problem 2
gss_train <- read.csv("data/gss_train.csv")

# save 10-fold CV train control
tr_ctrl <- caret::trainControl(method="cv", number=10)
```

Note that the pre-processing and hyperparameter tuning is all done inside the caret::train function. I call caret::preProcess inside the caret::train function, with methods c("zv", "center", "scale"). "zv" excludes the zero-variance features (based on prof. Waggoner's answer in piazza post @79 - "Feel free to drop any features with zero variance (so called, "single value" predictors)."),"scale" divides the feature by the SD of that feature, and "center" subtracts the feature by the mean of the feature. Also note that this does not standardize the response variable, `egalit_scale` in this case. Furthermore, although not explicitly specified, the caret::train function will create dummy variables for the categorical variables. I will demonstrate this by showing the coefficient of the final model of the linear regression (a).

The hyperparameter tuning is also done inside the caret::train function. I will report the final hyperparameters afterward. Also note that tr_ctrl fed into "trControl" inside the caret::train function makes sures that the 10-fold CV is being used.

a. Linear regression

Note that feature standardization is not really necessary for linear regression, but I am doing it for uniformity between models.

```r
set.seed(61)
# Note that the pre-processing is done inside the caret::train function
lm_gss <- caret::train(egalit_scale ~ ., data=gss_train, method="lm", metric="RMSE",
                       trControl = tr_ctrl, preProcess=c("zv", "scale", "center"))
results <- data.frame(matrix(NA, nrow=1, ncol=4))
colnames(results) <- c("CV_MSE", "Test_MSE", "CV_R2", "CV_RMSE")
results$CV_MSE <- lm_gss$results[ ,"RMSE"] ^ 2
results$Test_MSE <- rcfss::mse_vec(gss_test$egalit_scale, predict(lm_gss, newdata=gss_test))
results$CV_R2 <- lm_gss$results[, "Rsquared"]
results$CV_RMSE <- lm_gss$results[ ,"RMSE"]
knitr::kable(results, align="cccc",
             col.names=c("CV MSE", "Test MSE", "CV R^2", "CV RMSE"),
             caption = "MSE and R2 for Linear Regression")
```

Table 1: MSE and R2 for Linear Regression

CV MSE	Test MSE	CV R^2	CV RMSE
62.70211	63.37644	0.330663	7.918466

As seen in the above table, CV predicted MSE is around 62.70, and Test MSE is around 63.38. CV predicted R^2 is around 0.33.

```r
names(lm_gss$finalModel$coefficients)
```

```
##   [1] "(Intercept)"                     "age"
##   [3] "`attend>Once/wk`"                "`attend2-3 times /mo`"
##   [5] "`attendEvery wk`"                "attendNever"
##   [7] "`attendNrly evry wk`"            "`attendOnce/mo`"
##   [9] "`attendOnce/yr`"                 "`attendSev times/yr`"
##  [11] "authoritarianism"                "blackYes"
##  [13] "bornYES"                         "childs"
##  [15] "`colathNOT ALLOWED`"             "`colracNOT ALLOWED`"
##  [17] "`colcomNOT FIRED`"               "`colmilNOT ALLOWED`"
##  [19] "`colhomoNOT ALLOWED`"           "`colmslmYes, allowed`"
##  [21] "con_govt"                       "`degreeBachelor deg`"
##  [23] "`degreeGraduate deg`"           "degreeHS"
##  [25] "`degreeJunior Coll`"            "evangelicalLow"
##  [27] "evangelicalMod"                 "`grassNOT LEGAL`"
##  [29] "`happyPRETTY HAPPY`"            "`happyVERY HAPPY`"
##  [31] "hispanic_2Yes"                  "`homosexALWAYS WRONG`"
##  [33] "`homosexNOT WRONG AT ALL`"      "`homosexSOMETIMES WRONG`"
##  [35] "income06"                       "maritalMarried"
##  [37] "`maritalNever married`"         "maritalSeparated"
##  [39] "maritalWidowed"                 "`modeOVER THE PHONE`"
##  [41] "`newsFEW TIMES A WEEK`"         "`newsLESS THAN ONCE WK`"
##  [43] "newsNEVER"                      "`newsONCE A WEEK`"
##  [45] "owngunREFUSED"                  "owngunYES"
##  [47] "partyid_3Ind"                   "partyid_3Rep"
##  [49] "polviewsExtrmCons"              "polviewsExtrmLib"
##  [51] "polviewsLiberal"                "polviewsModerate"
##  [53] "polviewsSlghtCons"              "polviewsSlghtLib"
##  [55] "`pornlaw2Not illegal to all`"   "prayNEVER"
##  [57] "`prayONCE A DAY`"               "`prayONCE A WEEK`"
##  [59] "`praySEVERAL TIMES A DAY`"      "`praySEVERAL TIMES A WEEK`"
##  [61] "pres08Obama"                    "reborn_rYes"
##  [63] "religCATHOLIC"                  "religCHRISTIAN"
##  [65] "religHINDUISM"                  "`religINTER-NONDENOMINATIONAL`"
##  [67] "religJEWISH"                    "`religMOSLEM/ISLAM`"
##  [69] "`religNATIVE AMERICAN`"         "religNONE"
##  [71] "`religORTHODOX-CHRISTIAN`"      "religOTHER"
##  [73] "`religOTHER EASTERN`"           "religPROTESTANT"
##  [75] "science_quiz"                   "sexMale"
##  [77] "sibs"                           "social_connect"
##  [79] "social_cons3Liberal"            "social_cons3Mod"
##  [81] "southSouth"                     "spend3Liberal"
##  [83] "spend3Mod"                      "`teensexALWAYS WRONG`"
##  [85] "`teensexNOT WRONG AT ALL`"      "`teensexSOMETIMES WRONG`"
##  [87] "tolerance"                      "tvhours"
##  [89] "`vetyearsLESS THAN 2 YRS`"      "`vetyearsMORE THAN 4 YRS`"
##  [91] "vetyearsNONE"                   "wordsum"
##  [93] "zodiacARIES"                    "zodiacCANCER"
##  [95] "zodiacCAPRICORN"                "zodiacGEMINI"
##  [97] "zodiacLEO"                      "zodiacLIBRA"
##  [99] "zodiacPISCES"                   "zodiacSAGITTARIUS"
## [101] "zodiacSCORPIO"                  "zodiacTAURUS"
## [103] "zodiacVIRGO"
```

The above list is to show that caret::train is creating the dummy variables, as mentioned in the previous section.

b. Elastic net regression

```
set.seed(61)
# Note that the pre-processing is done inside the caret::train function
elnet_gss <- caret::train(egalit_scale ~ ., data=gss_train, method="glmnet", metric="RMSE",
```

```r
                          trControl = tr_ctrl, preProcess=c("zv", "scale", "center"), tuneLength=10)
results2 <- data.frame(matrix(NA, nrow=1, ncol=6))
colnames(results2) <- c("alpha", "lambda", "CV_MSE", "Test_MSE", "CV_R2", "CV_RMSE")
results2$alpha <- elnet_gss$results[rownames(elnet_gss$bestTune), "alpha"]
results2$lambda <- elnet_gss$results[rownames(elnet_gss$bestTune), "lambda"]
results2$CV_MSE <- elnet_gss$results[rownames(elnet_gss$bestTune), "RMSE"] ^ 2
results2$Test_MSE <- rcfss::mse_vec(gss_test$egalit_scale, predict(elnet_gss, newdata=gss_test))
results2$CV_R2 <- elnet_gss$results[rownames(elnet_gss$bestTune), "Rsquared"]
results2$CV_RMSE <- elnet_gss$results[rownames(elnet_gss$bestTune), "RMSE"]
knitr::kable(results2, align="cccccc",
             col.names=c("alpha", "lambda","CV MSE", "Test MSE", "CV R^2", "CV RMSE"),
             caption = "alpha, lambda, MSE and R2 for best Elastic Net Regression")
```

Table 2: alpha, lambda, MSE and R2 for best Elastic Net Regression

alpha	lambda	CV MSE	Test MSE	CV R^2	CV RMSE
0.6	0.3219274	59.84686	61.4736	0.3574777	7.736075

As seen in the above table, the optimal hyperparameter alpha for the elastic net regression is 0.6. The optimal lambda is about 0.32. CV predicted MSE is around 59.84, and Test MSE is around 61.47. CV predicted R^2 is around 0.36.

c. Principal component regression

```r
set.seed(61)
# Note that the pre-processing is done inside the caret::train function
pcr_gss <- caret::train(egalit_scale ~ ., data=gss_train, method="pcr", metric="RMSE",
                        trControl = tr_ctrl, preProcess=c("zv", "scale", "center"), tuneLength=44)
results3 <- data.frame(matrix(NA, nrow=1, ncol=5))
colnames(results3) <- c("ncomp", "CV_MSE", "Test_MSE", "CV_R2", "CV_RMSE")
results3$ncomp <- pcr_gss$results[rownames(pcr_gss$bestTune), "ncomp"]
results3$CV_MSE <- pcr_gss$results[rownames(pcr_gss$bestTune), "RMSE"] ^ 2
results3$Test_MSE <- rcfss::mse_vec(gss_test$egalit_scale, predict(pcr_gss, newdata=gss_test))
results3$CV_R2 <- pcr_gss$results[rownames(pcr_gss$bestTune), "Rsquared"]
results3$CV_RMSE <- pcr_gss$results[rownames(pcr_gss$bestTune), "RMSE"]
knitr::kable(results3, align="ccccc",
             col.names=c("Number of Components","CV MSE", "Test MSE", "CV R^2", "CV RMSE"),
             caption = "Number of components, MSE and R2 for best Principal Component Regression")
```

Table 3: Number of components, MSE and R2 for best Principal Component Regression

Number of Components	CV MSE	Test MSE	CV R^2	CV RMSE
26	64.17851	62.16889	0.3102192	8.011149

As seen in the above table, the optimal number of components for the principal component regression is 26. CV predicted MSE is around 64.18, and Test MSE is around 62.17. CV predicted R^2 is around 0.31.

d. Partial least squares regression

```
set.seed(61)
# Note that the pre-processing is done inside the caret::train function
pls_gss <- caret::train(egalit_scale ~ ., data=gss_train, method="pls", metric="RMSE",
                      trControl = tr_ctrl, preProcess=c("zv", "scale", "center"), tuneLength=44)
results4 <- data.frame(matrix(NA, nrow=1, ncol=5))
colnames(results4) <- c("ncomp", "CV_MSE", "Test_MSE", "CV_R2", "CV_RMSE")
results4$ncomp <- pls_gss$results[rownames(pls_gss$bestTune), "ncomp"]
results4$CV_MSE <- pls_gss$results[rownames(pls_gss$bestTune), "RMSE"] ^ 2
results4$Test_MSE <- rcfss::mse_vec(gss_test$egalit_scale, predict(pls_gss, newdata=gss_test))
results4$CV_R2 <- pls_gss$results[rownames(pls_gss$bestTune), "Rsquared"]
results4$CV_RMSE <- pls_gss$results[rownames(pls_gss$bestTune), "RMSE"]
knitr::kable(results4, align="ccccc",
             col.names=c("Number of Components","CV MSE", "Test MSE", "CV R^2", "CV RMSE"),
             caption = "Number of components, MSE and R2 for best Partial Least Squares Regression")
```

Table 4: Number of components, MSE and R2 for best Partial Least Squares Regression

Number of Components	CV MSE	Test MSE	CV R^2	CV RMSE
14	62.47102	63.38703	0.3328514	7.903861

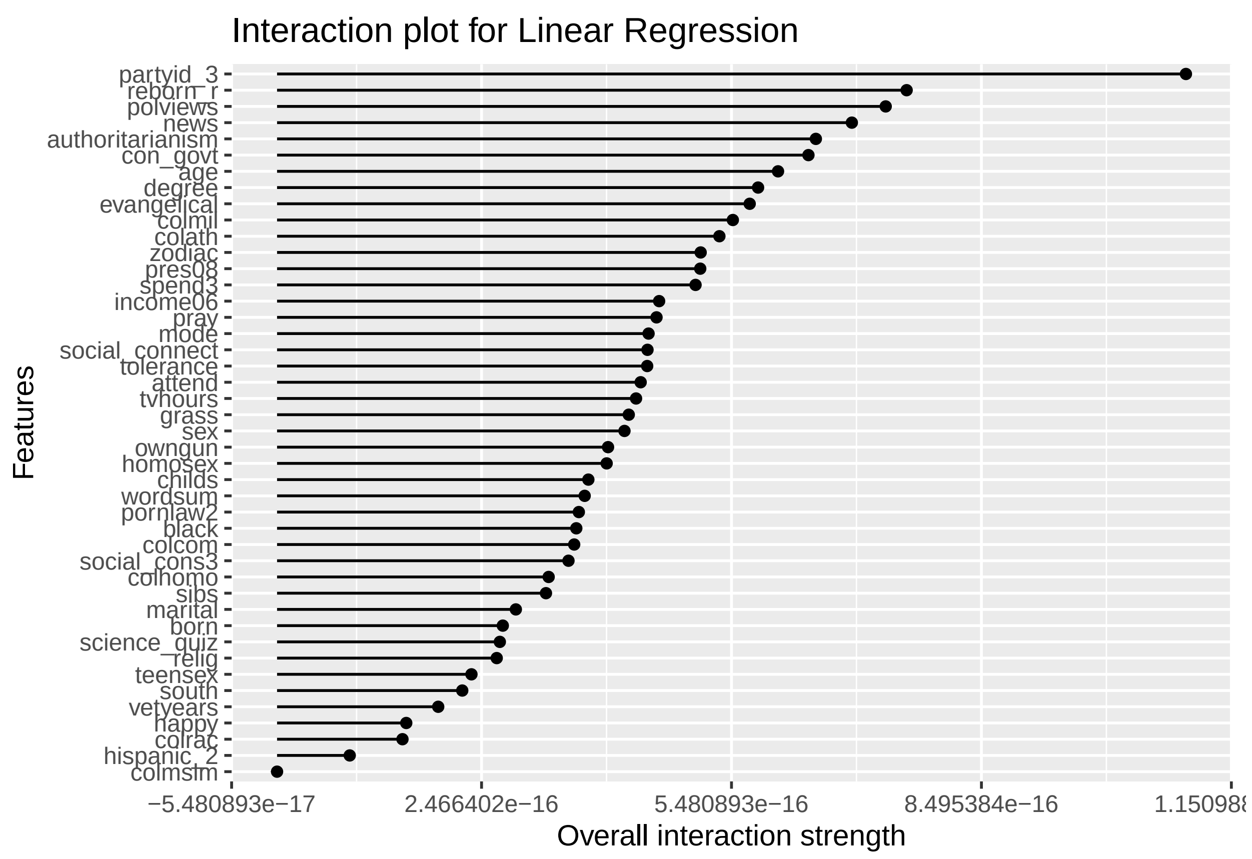
As seen in the above table, the optimal number of components for the partial least regression is 14. CV predicted MSE is around 62.47, and Test MSE is around 63.39. CV predicted R^2 is around 0.33.

Overall, I think all the models are doing similar jobs in terms of MSE or R^2. Their overall RMSE is around 7.9, which is slightly better than models using only one variable `income06`, which showed overall RMSE around 9.3. I do note that elastic net regression seems to be doing a little better in terms of MSE and R2 than other models.

5. For each final tuned version of each model fit, evaluate feature importance by generating feature interaction plots. Upon visual presentation, be sure to discuss the substantive results for these models and in comparison to each other (e.g., talk about feature importance, conditional effects, how these are ranked differently across different models, etc.).

Linear Regression

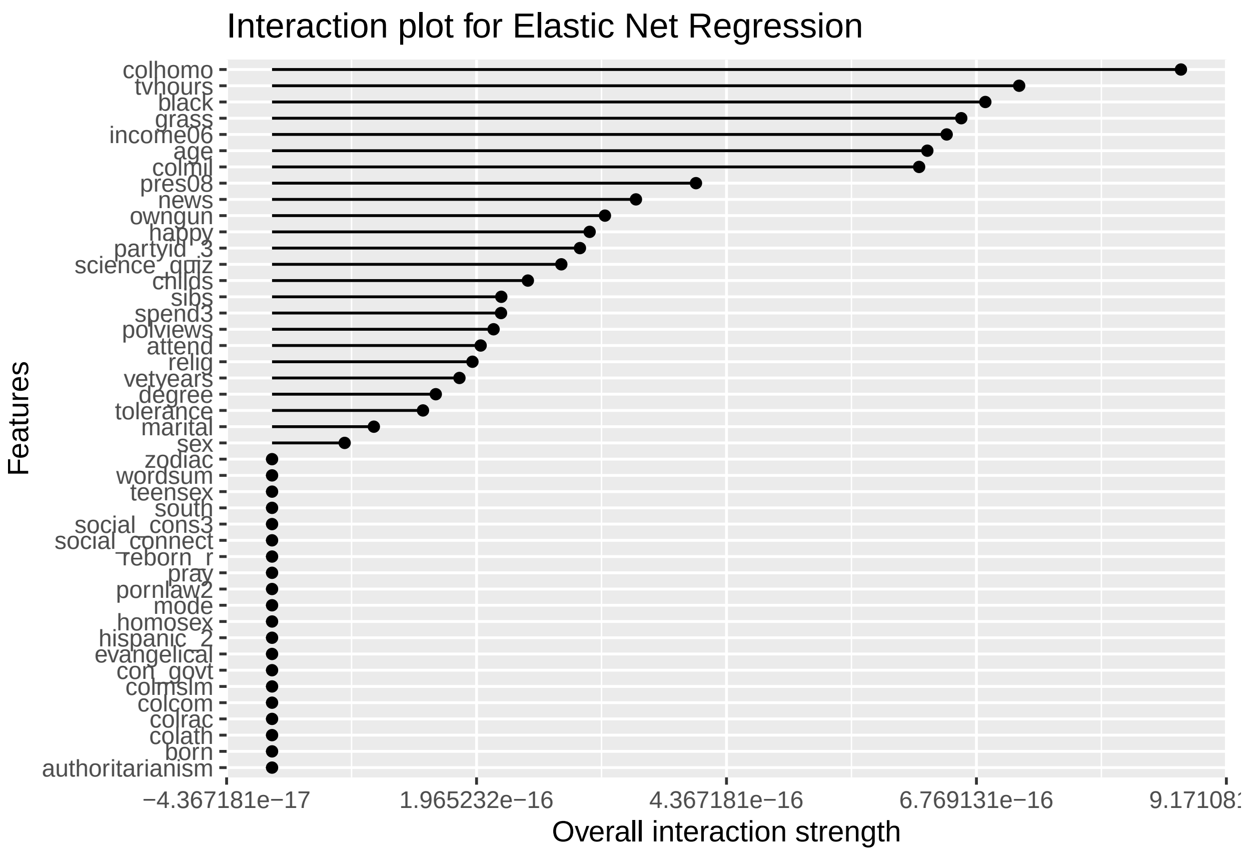
```
set.seed(61)
predictor_lm <- iml::Predictor$new(
  model = lm_gss,
  data = gss_test[-which(names(gss_test) == "egalit_scale")],
  y = gss_test$egalit_scale,
)
interact_glm <- iml::Interaction$new(predictor_lm)
plot(interact_glm) +
  ggtitle("Interaction plot for Linear Regression")
```



In the finally tuned and fitted linear regression model, `partyid_3`, `reborn_r`, `polviews`, `news`, and `authoritarianism` are the top 5 features in terms of feature importance evaluated by overall interaction strength. `partyid_3` seems to have especially strong overall interaction strength, and `colmstm` seems to have very little overall interaction strength.

Elastic net regression

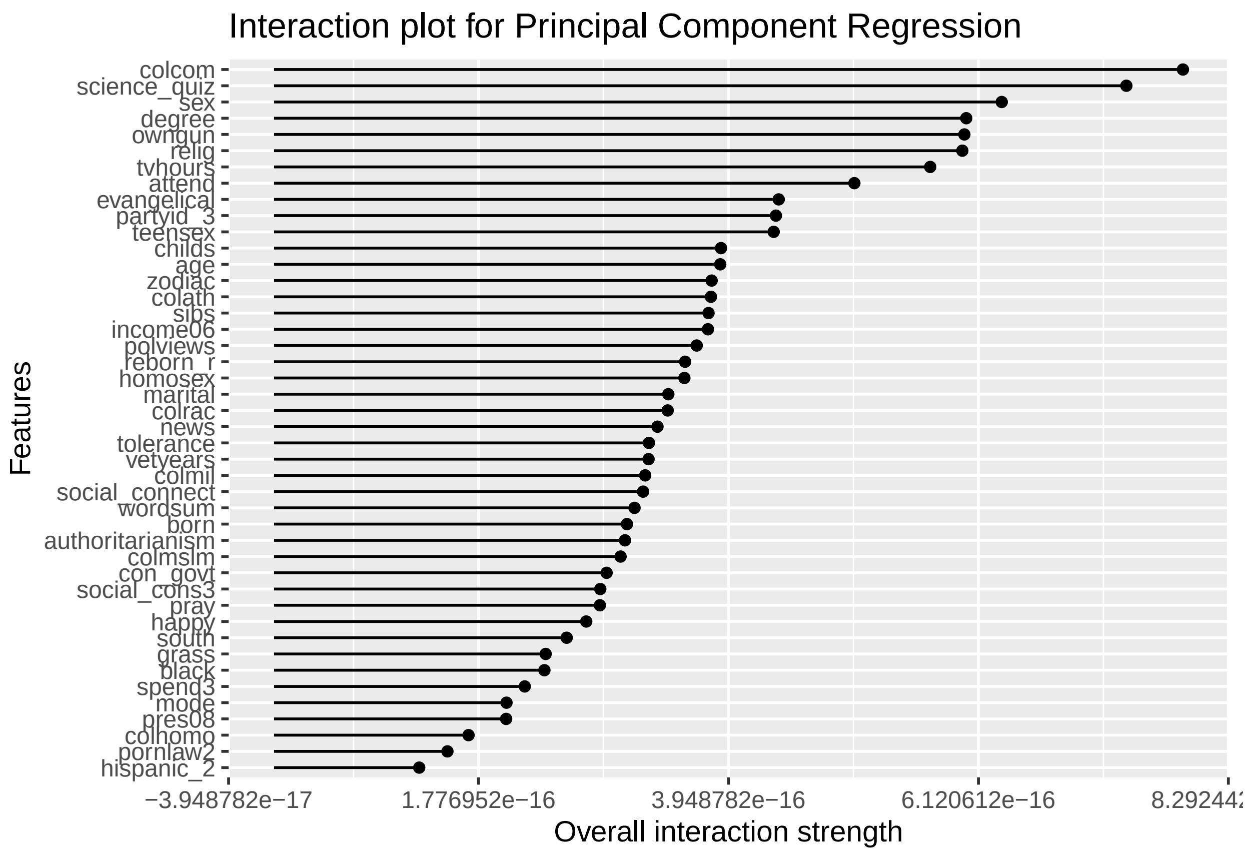
```
set.seed(61)
predictor_elnet <- iml::Predictor$new(
  model = elnet_gss,
  data = gss_test[-which(names(gss_test) == "egalit_scale")],
  y = gss_test$egalit_scale,
)
interact_elnet <- iml::Interaction$new(predictor_elnet)
plot(interact_elnet) +
  ggtitle("Interaction plot for Elastic Net Regression")
```



In the finally tuned and fitted elastic net regression model, `colhomo`, `tvhours`, `black`, `grass`, `income06` and `age` are the top 6 features in terms of feature importance evaluated by overall interaction strength. `colhomo` seems to have especially strong overall interaction strength even within these top 6. Other features seem to be having much less overall interaction strength, and around 20 features have very little interaction strength. This could be because elastic net regression is a regularization method. The redundant features are regularized in this model, so it makes sense that there are a lot of features that have little interaction strength.

Principal component regression

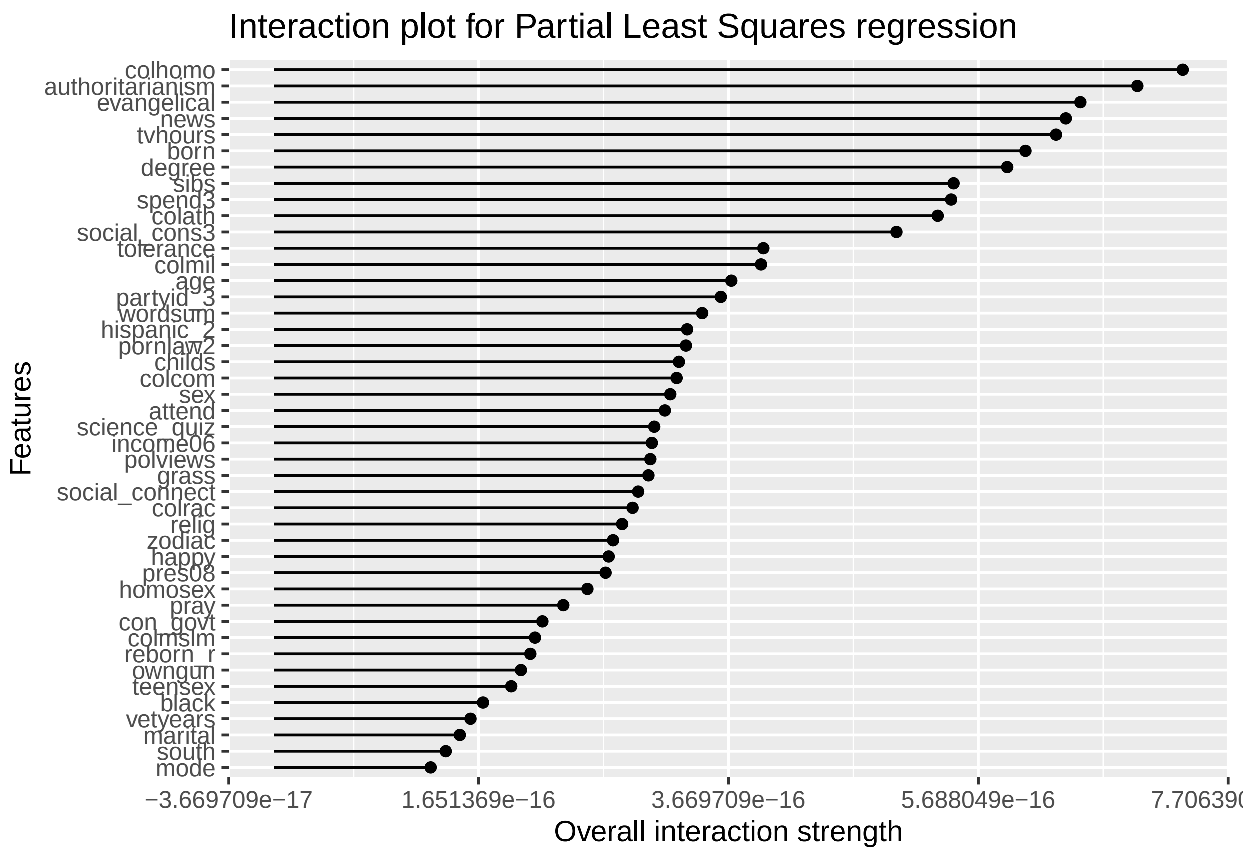
```r
set.seed(61)
predictor_pcr <- iml::Predictor$new(
  model = pcr_gss,
  data = gss_test[-which(names(gss_test) == "egalit_scale")],
  y = gss_test$egalit_scale,
)
interact_pcr <- iml::Interaction$new(predictor_pcr)
plot(interact_pcr) +
  ggtitle("Interaction plot for Principal Component Regression")
```



In the finally tuned and fitted principal component regression model, `colcom`, `science_quiz`, `sex`, `degree`, and `owngun`, `relig`, `tvhours` are the top 7 features in terms of feature importance evaluated by overall interaction strength. `colcom` and `science_quiz` seems to have relatively stronger overall interaction strength. Other features outside the mentioned 7 features seem to have quite less overall interaction strength compared to these 7 features.

Partial least squares regression

```
set.seed(61)
predictor_pls <- iml::Predictor$new(
  model = pls_gss,
  data = gss_test[-which(names(gss_test) == "egalit_scale")],
  y = gss_test$egalit_scale,
)
interact_pls <- iml::Interaction$new(predictor_pls)
plot(interact_pls) +
  ggtitle("Interaction plot for Partial Least Squares regression")
```



In the finally tuned and fitted partial least squares regression model, `colhomo`, `authoritarianism`, `evangelical`, `news`, and `tvhours` are the top 5 features in terms of feature importance evaluated by overall interaction strength.

```r
compare_df <- data.frame(matrix(NA, nrow=15, ncol=4))
colnames(compare_df) <- c("lm", "elnet", "pcr", "pls")
compare_df$lm <- interact_glm$results[order(-interact_glm$results$.interaction), ]$.feature[1:15]
compare_df$elnet <- interact_elnet$results[order(-interact_elnet$results$.interaction), ]$.feature[1:15
compare_df$pcr <- interact_pcr$results[order(-interact_pcr$results$.interaction), ]$.feature[1:15]
compare_df$pls <- interact_pls$results[order(-interact_pls$results$.interaction), ]$.feature[1:15]
knitr::kable(compare_df, align="cccc",
             col.names=c("Linear Regression", "Elastic Net Regression",
                         "Principal Component Regression","Partial Least Squares Regression"),
             caption = "Top 15 features by feature interaction strength for each model")
```

Table 5: Top 15 features by feature interaction strength for each model

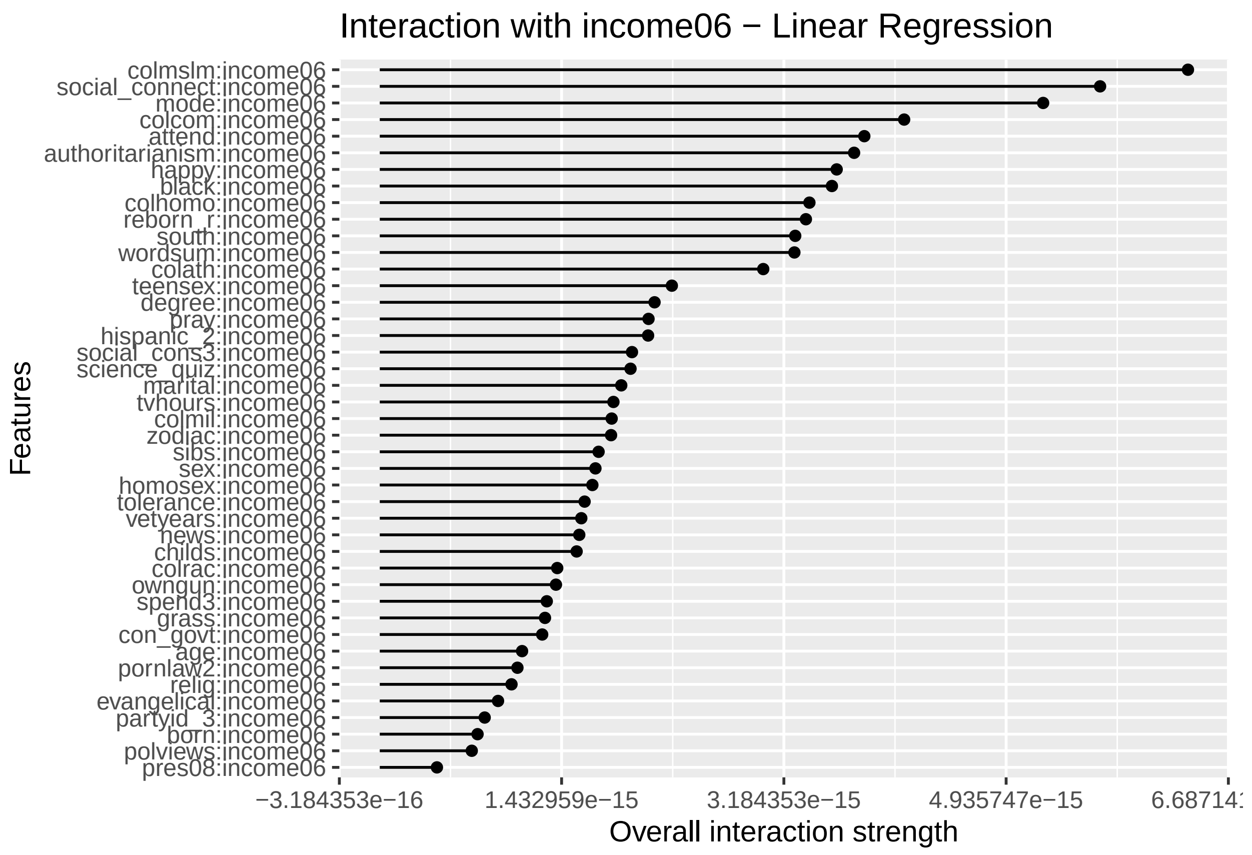
Linear Regression	Elastic Net Regression	Principal Component Regression	Partial Least Squares Regression
partyid_3	colhomo	colcom	colhomo
reborn_r	tvhours	science_quiz	authoritarianism
polviews	black	sex	evangelical
news	grass	degree	news
authoritarianism	income06	owngun	tvhours
con_govt	age	relig	born

Linear Regression	Elastic Net Regression	Principal Component Regression	Partial Least Squares Regression
age	colmil	tvhours	degree
degree	pres08	attend	sibs
evangelical	news	evangelical	spend3
colmil	owngun	partyid_3	colath
colath	happy	teensex	social_cons3
zodiac	partyid_3	childs	tolerance
pres08	science_quiz	age	colmil
spend3	childs	zodiac	age
income06	sibs	colath	partyid_3

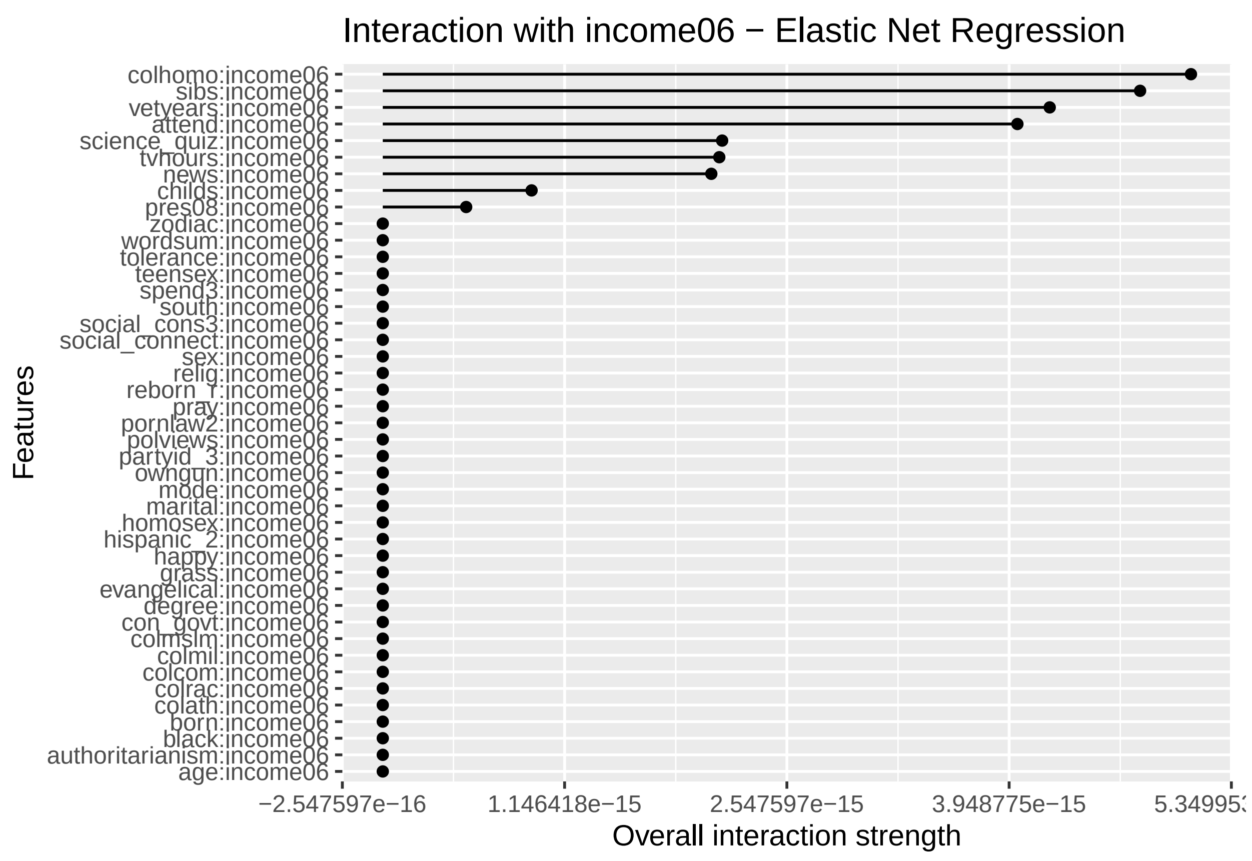
The above table shows top 15 features in terms of feature importance evaluated by overall interaction strength for each models. Although some features like `partyid_3`, `age` appears in all 4 models and features like `news`, `degree`, `evangelical`, `tvhours`, `colmil` appears in 3 of the models, we can see that there are striking differences in the feature importance of each feature across models. I think this difference comes from the fact that there is no single feature that predicts `egalit_scale` quite well in the GSS dataset, and different models utilize different methods, different assumptions, and different weights to the features to estimate the response feature.

Finally, building on that we examined `income06` a lot in problem 1 ~ 3, I will see what feature shows the largest interactions with `income06` in each model.

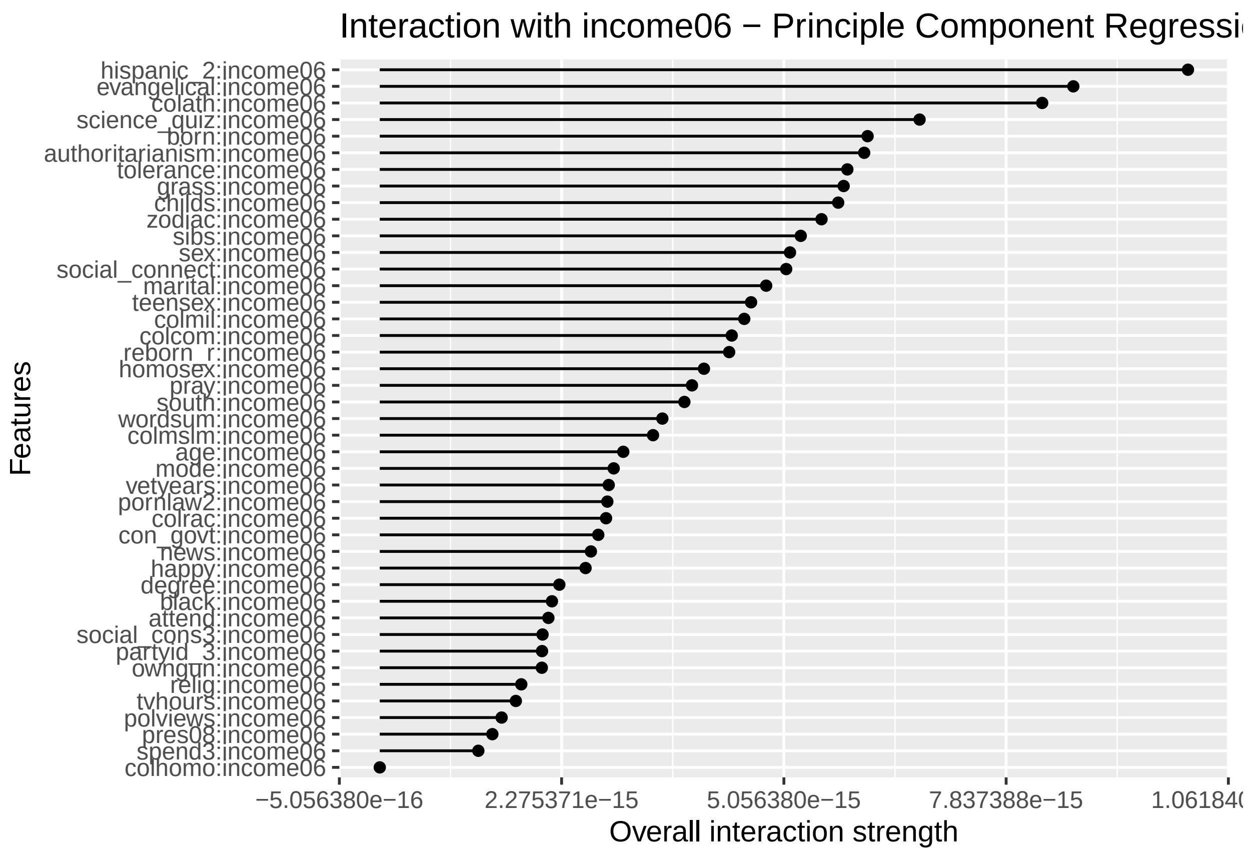
```r
set.seed(61)
Interaction$new(predictor_lm, feature = "income06") %>%
  plot() +
  ggtitle("Interaction with income06 - Linear Regression")
```



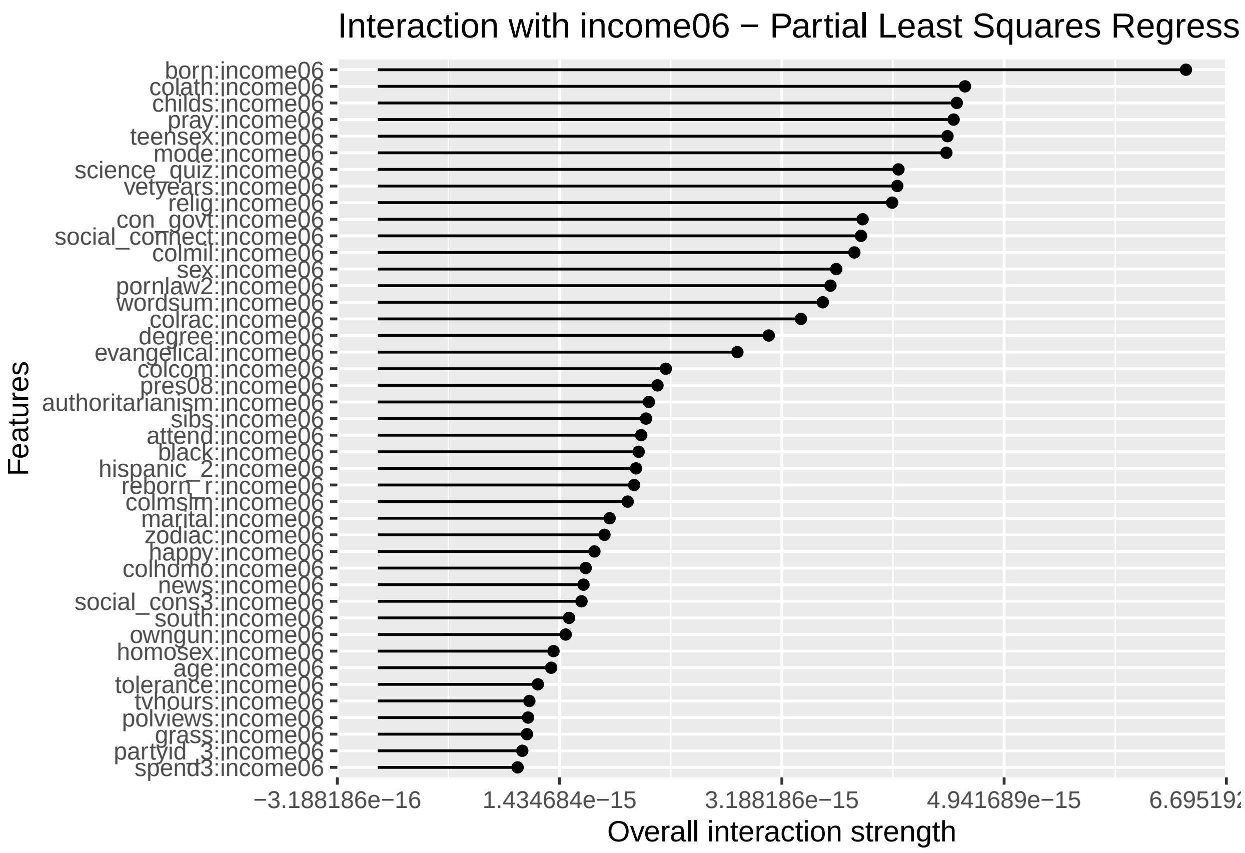
```r
Interaction$new(predictor_elnet, feature = "income06") %>%
  plot() +
  ggtitle("Interaction with income06 - Elastic Net Regression")
```



```r
Interaction$new(predictor_pcr, feature = "income06") %>%
  plot() +
  ggtitle("Interaction with income06 - Principle Component Regression")
```



```r
Interaction$new(predictor_pls, feature = "income06") %>%
  plot() +
  ggtitle("Interaction with income06 - Partial Least Squares Regression")
```



From the above plots, we can see that the features that show the largest interactions with `income06` also vary a lot across models. This is analogous to the interaction plot of all features shown above.

This article was written using R Markdown and was knitted to pdf using the tinytex package