Past Linearity (Homework 4)

Anuraag Girdhar

1.

Polynomial regression is used to predict egalitarin orientation

```
set.seed(123)
library(margins)
library(repr)
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(rsample)
```

```
## Loading required package: tidyr
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(purrr)
library(splines)
library(caret)
```

```
## Loading required package: lattice
```

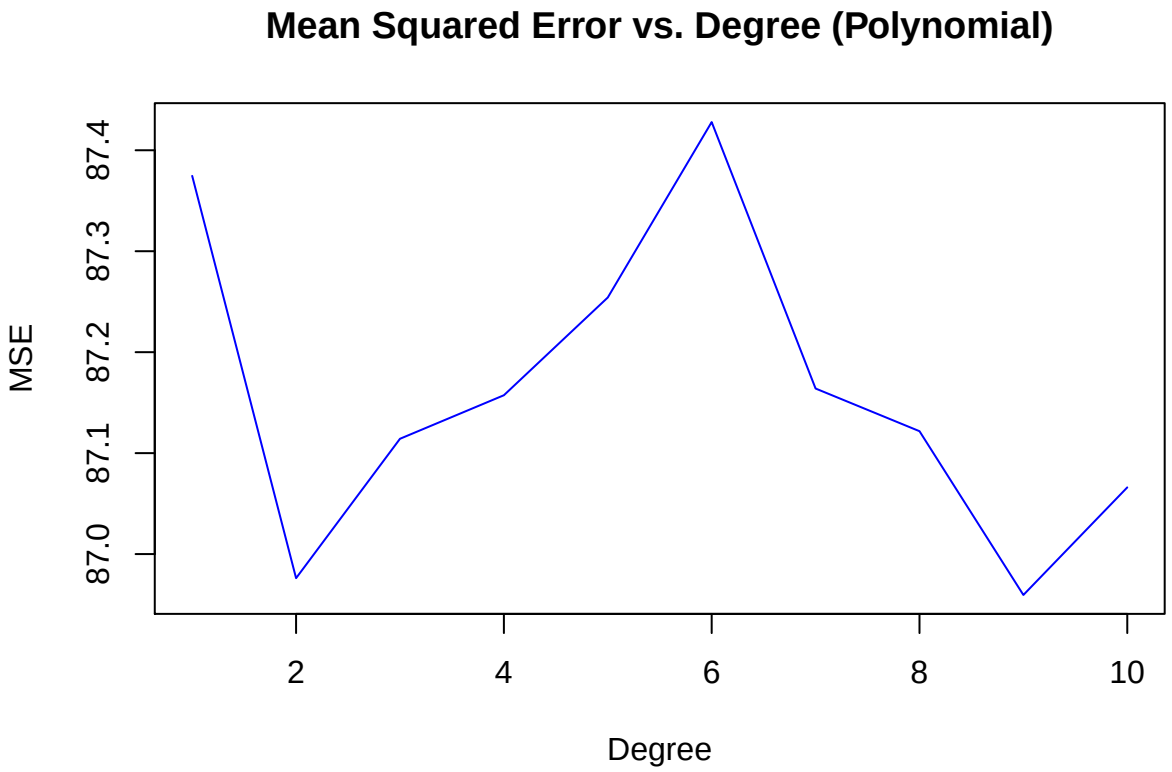
```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

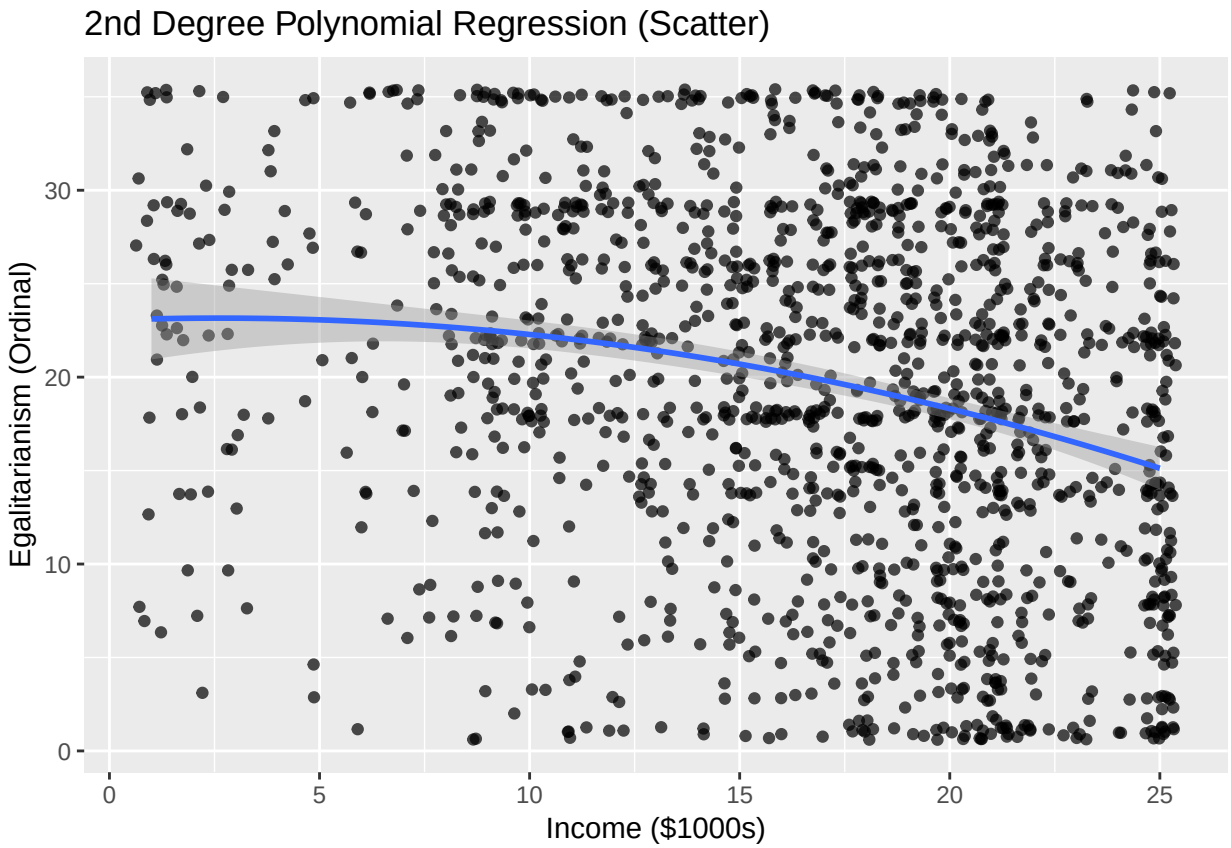
```
library(iml)
options(repr.plot.width=5, repr.plot.height=4)

gss_train = read.csv('data/gss_train.csv')
gss_test = read.csv('data/gss_test.csv')
```

```
# Fit polynomial regression model on the data

size = 10
folds = sample(size, nrow(gss_train), replace = TRUE)
```

```r
mse = c()
for(j in 1:size) {
  sub_mse = c()
  for(i in 1:size) {
    test_set = gss_train[folds==i,]
    training_set = gss_train[folds!=i,]
    lmodel = lm(egalit_scale ~ poly(income06, j), data = training_set)
    pred = predict(lmodel, test_set)
    sub_mse = c(sub_mse, mean((pred - test_set$egalit_scale)^2))
  }
  mse = c(mse, mean(sub_mse))
}

# Plot the mean squared error of the model against polynomial degree

plot(mse, type = 'l', xlab = 'Degree', ylab = 'MSE', col="Blue", main="Mean Squared Error vs. Degree (P
```



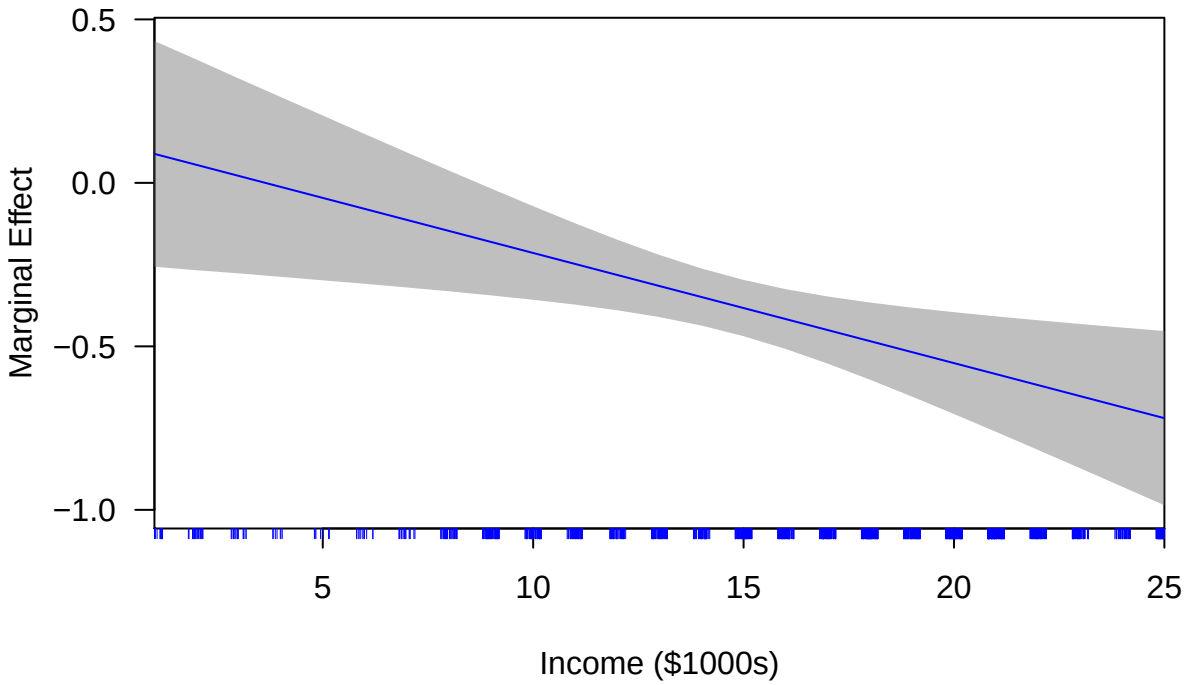
Based on mean squared error, the optimal degree for polynomial regression is 2.




```
best_model = lm(egalit_scale ~ stats::poly(income06, 2), data = training_set)
summary(margins(best_model))
```

```
##    factor     AME     SE       z      p   lower   upper
##  income06 -0.4410 0.0506 -8.7121 0.0000 -0.5402 -0.3418
```

```
# Plot Marginal Effect against Income

cplot(best_model, 'income06', what = 'effect', xlab="Income ($1000s)", ylab="Marginal Effect", col="Blu
```



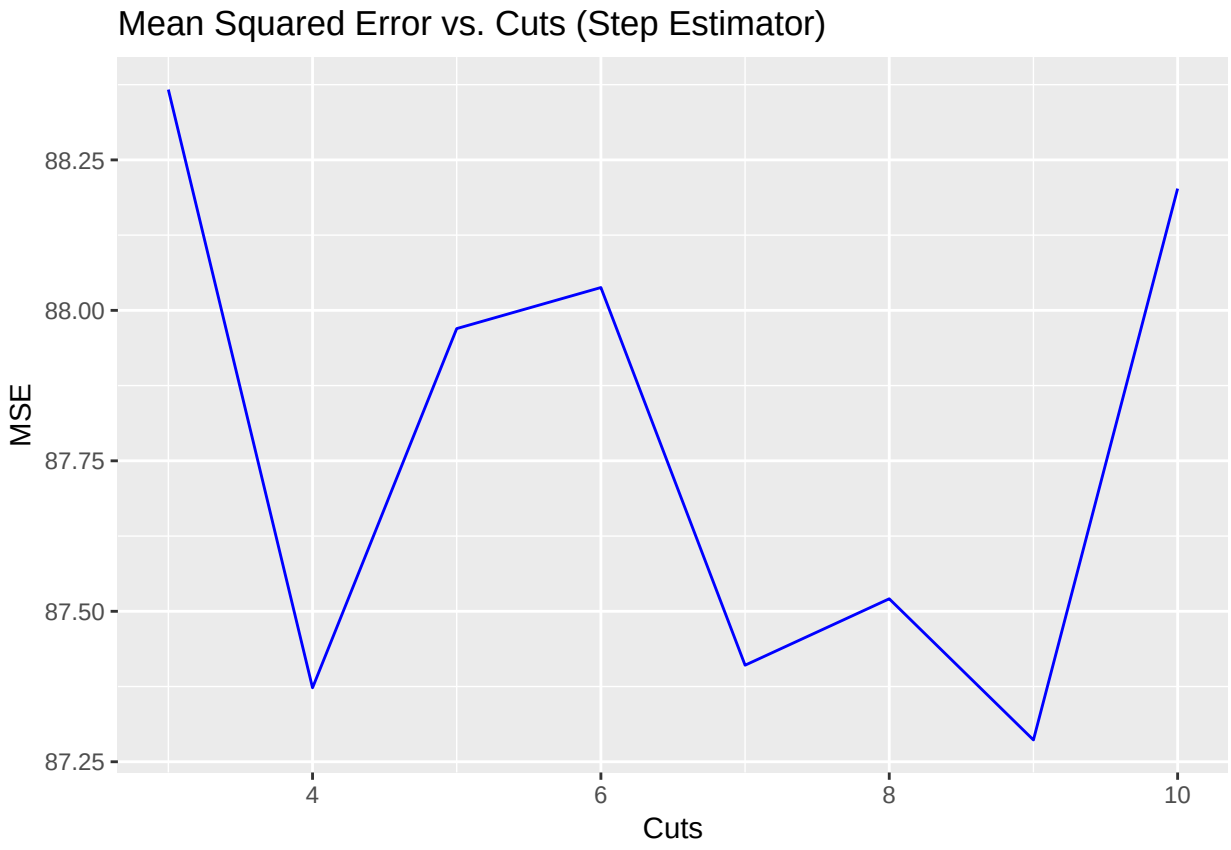
As noted before, the polynomial regression with best MSE is a 2nd order polynomial, but reaches another local minimum at a 9th order polynomial. Since the 2nd order outperforms the 9th order polynomial, questions of model complexity balance don't come into play, and the 2nd order polynomial is clearly the best choice. As income rises, the marginal effect of income falls drops from positive to negative. The AME for a 2-degree polynomial model is -0.4537, suggesting that as income rises, people's concern for egalitarianism falls. A $1000 rise in income reduces egalitarianism by about 0.45 scaled units.

2.

A step-function is used to predict egalitarian orientation

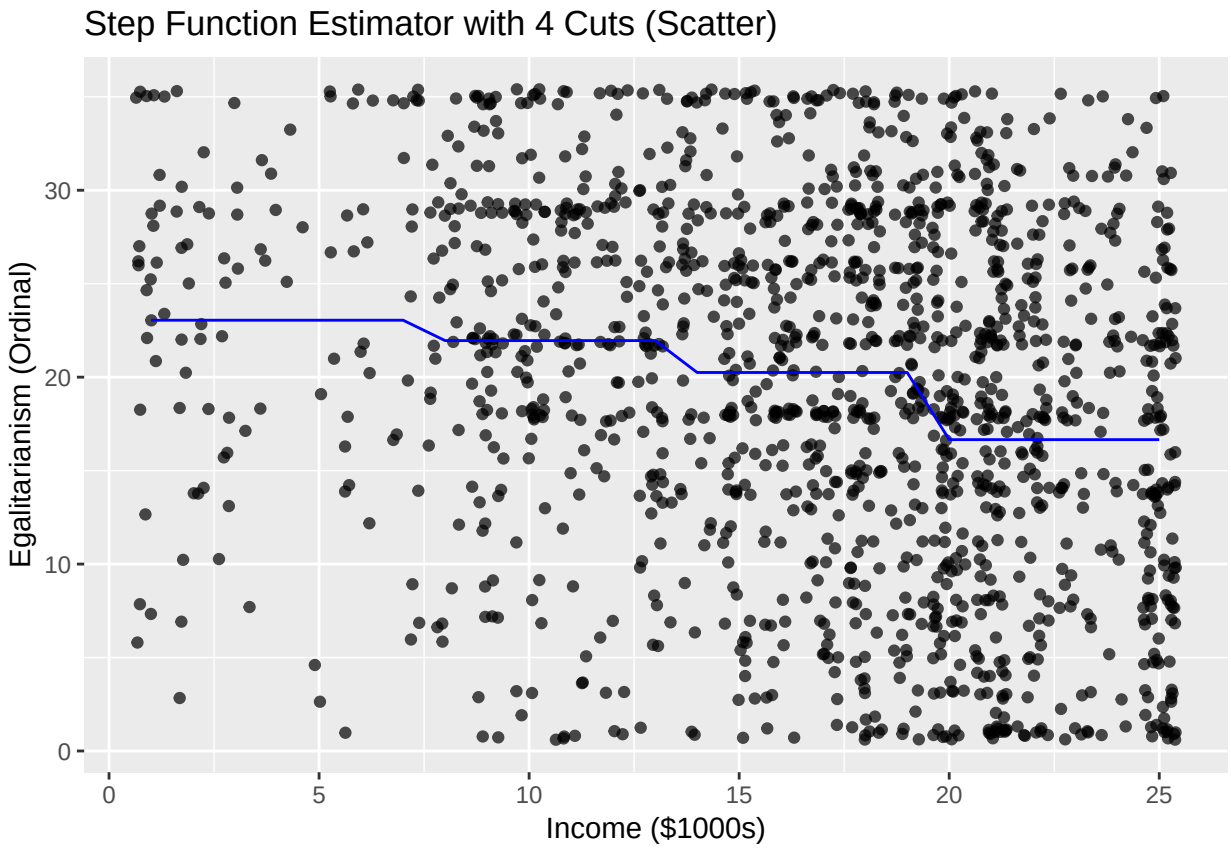
```
# Fit the step function estimator to the data

mse = c()
reg = "\\((.+),.*"
creg = "[^,]*,([^]]*)\\]"
for(j in 3:size){
  sub_mse = c()
  for(i in 1:size){
    test_set = gss_train[folds==i,]
    training_set = gss_train[folds!=i,]

    labs = levels(cut(gss_train$income06, j))
    breaks = unique(c(as.numeric(sub(reg, "\\1", labs)),
                      as.numeric(sub(creg, "\\1", labs))))
    step_model = lm(egalit_scale~cut(income06,unique(breaks)), data = training_set)
    pred = predict(step_model, test_set)

    sub_mse = c(sub_mse, mean((pred - test_set$egalit_scale)^2))
    }
  mse = c(mse, mean(sub_mse))
}
```

```r
# Plot the mean squared error of the model against step function cuts

step_sum = data.frame('Cuts' = 3:size, "MSE"=mse)
ggplot(step_sum, aes(Cuts, MSE)) +
  labs(title="Mean Squared Error vs. Cuts (Step Estimator)") +
  geom_line(color="blue")
```



Based on mean squared error, the optimum number of cuts for the step function estimator is 4.

```r
# Calculate the optimal number of cuts

labs = levels(cut(gss_train$income06, 4))
cuts = unique(c(as.numeric(sub(reg, "\\1", labs)),
                as.numeric(sub(creg, "\\1", labs))))
best_steps = lm(egalit_scale~cut(income06,unique(cuts)), data = gss_train)
pred = predict(best_steps, gss_train)
df_pred = data.frame('income06' = gss_train$income06, 'pred' = pred)

# Plot the fit of the best step function regression against egalitarianism

ggplot(gss_train, aes(income06, egalit_scale)) +
  geom_jitter(alpha = 0.7) +
  geom_line(data = df_pred, aes(income06, pred), color = 'blue') +
  labs(title = 'Step Function Estimator with 4 Cuts (Scatter)', x = 'Income ($1000s)',
       y = 'Egalitarianism (Ordinal)')
```



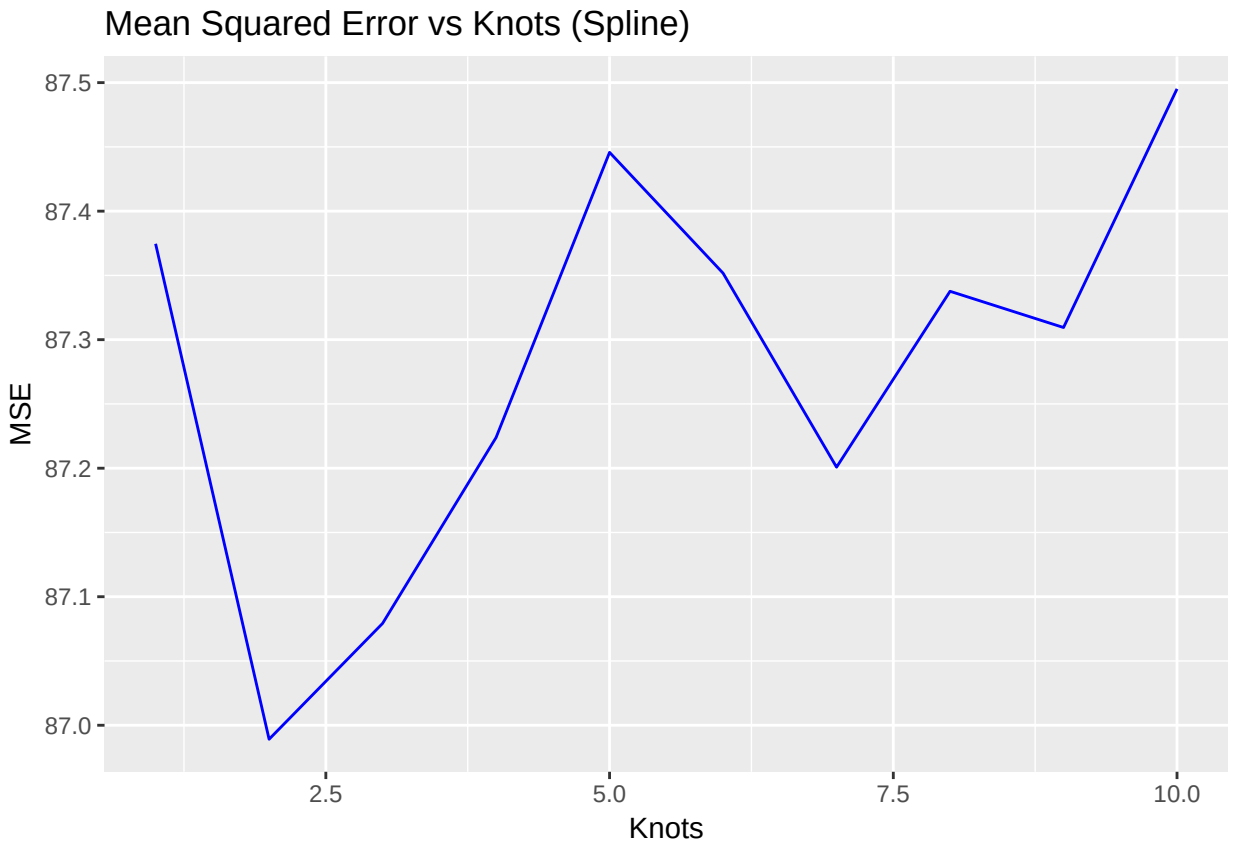
As noted before, the step function regression with best MSE is one with 4 cuts, but reaches another local minimum at 9 cuts. Since the 4 cut outperforms the 9 cut step function, questions of model complexity balance don't come into play, and the 4 cut step function is clearly the best choice. As expected from the previous polynomial regression, predicted orientation toward egalitarianism decreases as income increases

3.

A natural regression spline is used to predict egalitarian orientation

```
# Fit the natural regression spline to the data

mse = c()
for(j in 1:size){
  sub_mse = c()
  for(i in 1:size){
    test_set = gss_train[folds==i,]
    training_set = gss_train[folds!=i,]
    spline_model = lm(egalit_scale ~ ns(income06, df = j), data = training_set)
    pred = predict(spline_model, test_set)
    sub_mse = c(sub_mse, mean((pred - test_set$egalit_scale)^2))
  }
  mse = c(mse, mean(sub_mse))
}

# Plot the mean squared error of the model against cubic spline knots

step_sum = data.frame('df' = 1:size, "MSE"=mse)
ggplot(step_sum, aes(df, MSE)) + geom_line(color = "Blue") +
```

```
  labs(x = "Knots", title = "Mean Squared Error vs Knots (Spline)")
```



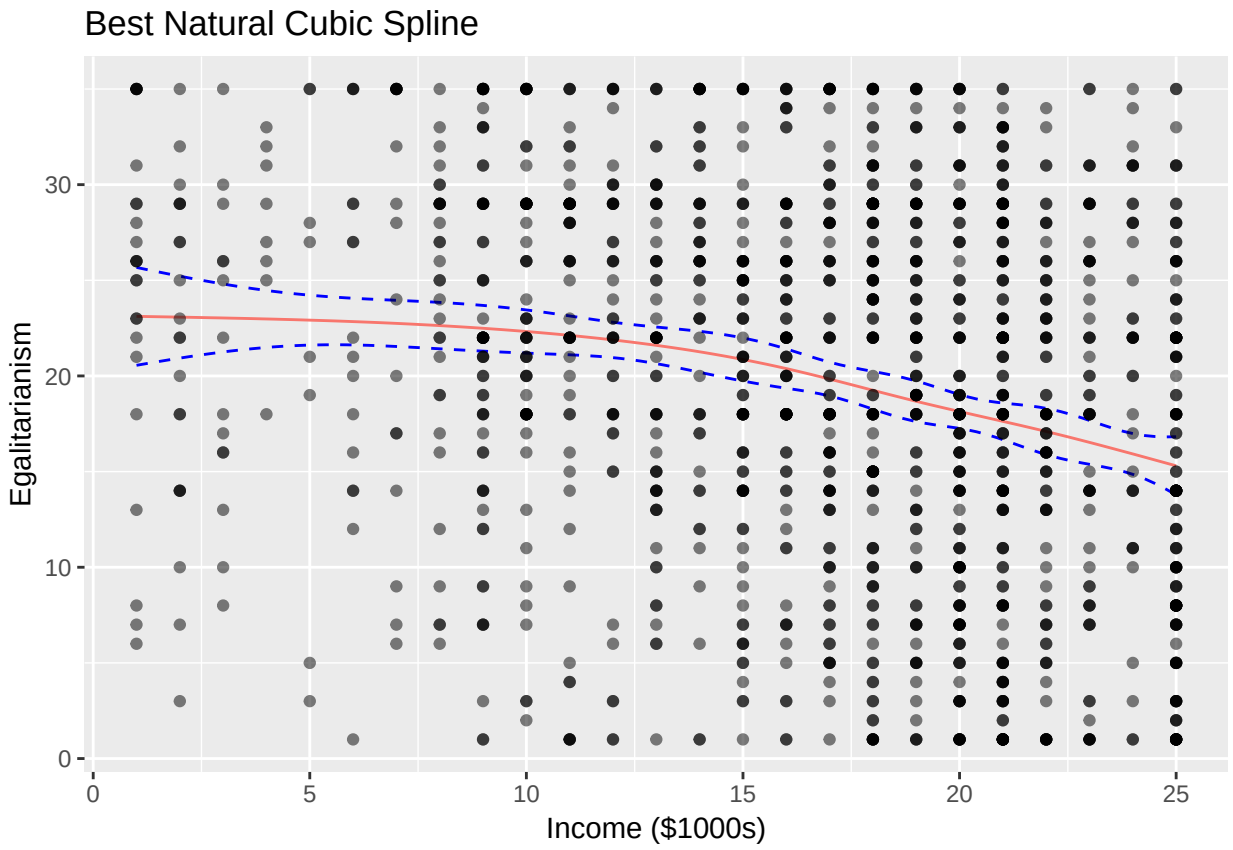
Based on mean squared error, the optimum number of knots for the natural regression spline is 2.

```
# Calculate the optimal number of knots

spline_model = glm(egalit_scale ~ ns(income06, df = 5), data = gss_train) %>%
  cplot("income06", what = "prediction", n = 100, draw = FALSE)
```

```
##       xvals    yvals    upper    lower
## 1  1.000000 23.11502 25.67307 20.55698
## 2  1.242424 23.10537 25.55909 20.65166
## 3  1.484848 23.09567 25.44693 20.74440
## 4  1.727273 23.08587 25.33690 20.83483
## 5  1.969697 23.07591 25.22930 20.92252
## 6  2.212121 23.06575 25.12444 21.00706
## 7  2.454545 23.05533 25.02265 21.08801
## 8  2.696970 23.04461 24.92429 21.16493
## 9  2.939394 23.03353 24.82970 21.23737
## 10 3.181818 23.02205 24.73923 21.30487
## 11 3.424242 23.01011 24.65324 21.36697
## 12 3.666667 22.99765 24.57204 21.42326
## 13 3.909091 22.98464 24.49595 21.47333
## 14 4.151515 22.97101 24.42520 21.51683
## 15 4.393939 22.95673 24.35998 21.55347
## 16 4.636364 22.94173 24.30039 21.58307
## 17 4.878788 22.92596 24.24639 21.60553
## 18 5.121212 22.90938 24.19785 21.62091
```

```
## 19 5.363636 22.89194 24.15450 21.62938
## 20 5.606061 22.87357 24.11591 21.63124
```

```r
# Plot the fit of the best cubic spline against egalitarianism

spline_model %>%
  ggplot(aes(x = xvals)) +
  geom_line(aes(y = yvals, color = 'red')) +
  geom_line(aes(y = upper), linetype = 2, color = 'blue') +
  geom_line(aes(y = lower), linetype = 2, color = 'blue') +
  geom_point(data = gss_train, aes(income06, egalit_scale), alpha = 0.5) +
  labs(x = "Income ($1000s)", y = "Egalitarianism", title = "Best Natural Cubic Spline") +
  theme(legend.position = "none")
```



As noted before, the cubic spline with best MSE is one with 2 knots, but reaches another local minimum at 7 knots. Since the 2 knot outperforms the 7 knot spline, questions of model complexity balance don't come into play, and the 2 knot step function is clearly the best choice. We observe a smoother fit than the step function, and a tight prediction interval around the data.

4.

Estimate linear regression, Elastic Net Regression, Principal Component Regression, and Partial Least Squares regression on all the predictors, 10-fold cross validation

```r
# Preprocessing data

standardized <- function(data){
  df <- data %>% mutate_if(is.numeric, scale) %>%
```

```
    mutate_if(is.numeric, c)
  }
gss_train <- standardized(gss_train)
gss_test <- standardized(gss_test)
#create trainControl for 10-fold cv
trControl = trainControl(method = "cv", number = 10)
```

```
# Training a linear model

lm_train <- train( egalit_scale ~ ., gss_train, method = "lm", trControl = trControl)
lm_train
```

```
## Linear Regression
##
## 1481 samples
##   44 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1334, 1333, 1332, 1332, 1333, 1333, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.8303998  0.3209638  0.6567191
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# Testing the linear model

lm_test <- train(egalit_scale ~ ., gss_test, method = "lm")
lm_test
```

```
## Linear Regression
##
## 493 samples
##  44 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 493, 493, 493, 493, 493, 493, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9933797  0.1753252  0.791766
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# Training the elastic net model

gss_elnet = train(egalit_scale ~ ., data = gss_train,
                  method = "glmnet",
                  trControl = trControl, tuneLength = 10)
```

```
# Determining the best alpha and lambda

tuned_grid <- expand.grid(alpha = gss_elnet$bestTune$alpha,
```

```
                      lambda = gss_elnet$bestTune$lambda)

# Fitting a model with the best alpha and lambda
gss_elnet_best = train(egalit_scale ~ .,
                       data = gss_test, method = "glmnet",
                       tuneGrid = tuned_grid)
gss_elnet_best
```

```
## glmnet
##
## 493 samples
##  44 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 493, 493, 493, 493, 493, 493, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.8832698  0.2434621  0.7024214
##
## Tuning parameter 'alpha' was held constant at a value of 0.6
##
## Tuning parameter 'lambda' was held constant at a value of 0.03345934
```

The best tuned alpha is 0.7, and the best tuned lambda is 0.03346. The cross-validated training MSE is 0.6889, and the test MSE from that linear model is 1.026. The final MSE obtained from the elastic net regression is 0.7653. This MSE outperforms the simple linear regression. This is along the lines of what we would expect given the fact that hyperparameter turning reduces the variance of the model.

```
# Determining the best number of principal components (ncomp)

gss_pcr = train(egalit_scale ~ ., data = gss_train, method = "pcr",
                      trControl = trControl, tuneLength = 100)
gss_pcr$bestTune$ncomp
```

```
## [1] 45
```

```
myGrid <- expand.grid(ncomp = gss_pcr$bestTune$ncomp)

# Fit a model with the best ncomp
gss_pcr_best = train(egalit_scale ~ ., data = gss_test,
                     method = "pcr", tuneGrid = myGrid)
gss_pcr_best
```

```
## Principal Component Analysis
##
## 493 samples
##  44 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 493, 493, 493, 493, 493, 493, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
```

```
##   0.8909522  0.2347152  0.7106913
##
## Tuning parameter 'ncomp' was held constant at a value of 45
```

The best number of principal components for PCR is ‘45. After tuning the model with this ncomp value, the final MSE is 0.7751. The final MSE is worse than that of elastic net but better than that of linear regression. Elastic net slightly outperforms PCR along this metric, however, PCR significantly outperforms linear regression, likely due to shared variance across predictors

```
# Determine the best ncomp

gss_pls = train(egalit_scale ~ ., data = gss_train,
                method = "pls", trControl = trControl, tuneLength = 100)
gss_pls$bestTune$ncomp
```

```
## [1] 5
```

```
myGrid <- expand.grid(ncomp = gss_pls$bestTune$ncomp)

# Fit a model with the best ncomp
gss_pls_best = train(egalit_scale ~ .,
                     data = gss_test, method = "pls",
                     tuneGrid = myGrid)
gss_pls_best
```

```
## Partial Least Squares
##
## 493 samples
##  44 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 493, 493, 493, 493, 493, 493, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9079469  0.2331896  0.7230159
##
## Tuning parameter 'ncomp' was held constant at a value of 5
```

The best number of principal components for PLS is 5. Since partial least squares is a supervised method that considers the egalitarian orientation, it requires fewer principal components to minimize training MSE. After tuning the model with this ncomp, the final MSE is 0.8203. This again outperforms linear regression but underperforms PCR, given the smaller number of principal components.

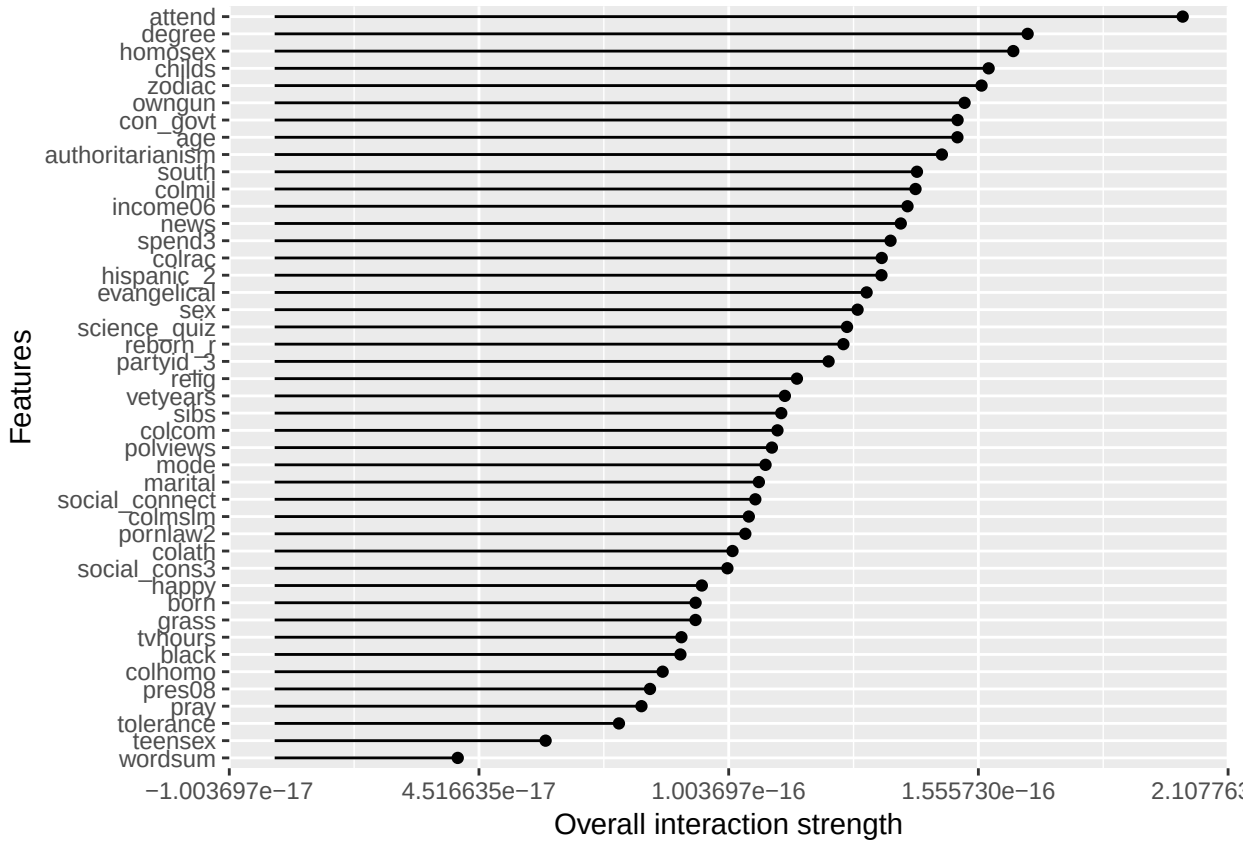
Elastic net minimizes MSE. PCR and PLS underperform because of their inflexibility with categorical variables, but all models outperform linear regression due to improvements in variance modeling.

5.

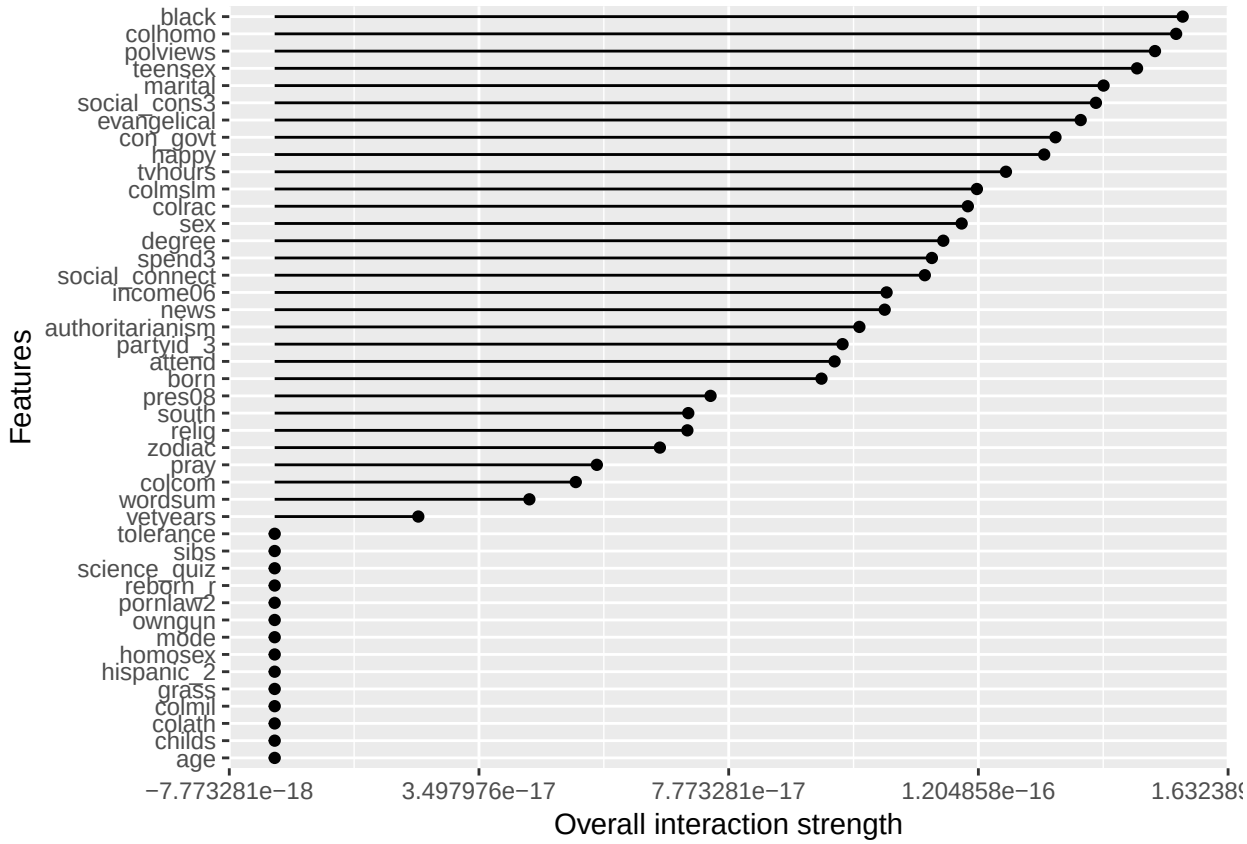
Feature interaction plots for each model fit above

```
features <- gss_test %>% dplyr::select(-egalit_scale)
response <- as.numeric(as.vector(gss_test$egalit_scale))
```

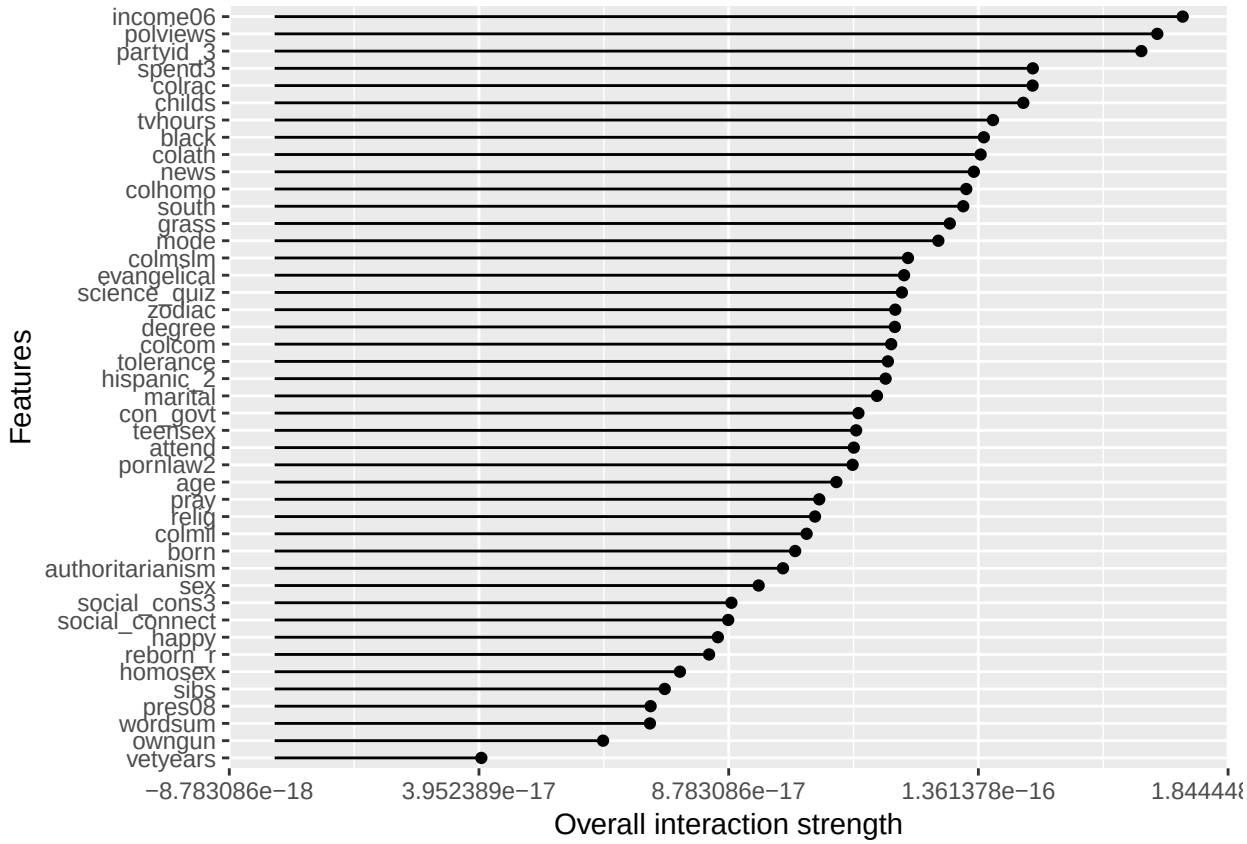
```
linear_pred = Predictor$new(model = lm_test, data = features, y = response)
plot(Interaction$new(linear_pred))
```



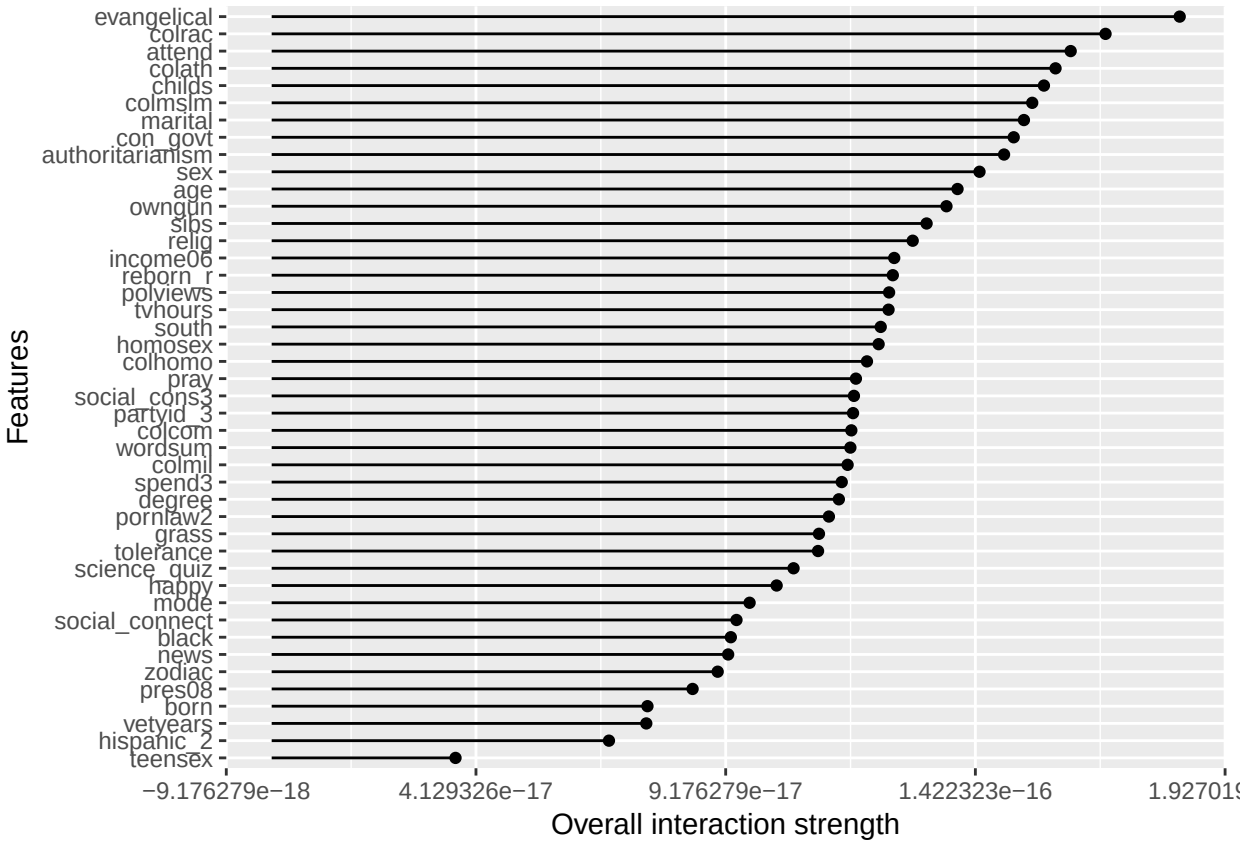

```
elastic_net_pred = Predictor$new(model = gss_elnet_best, data = features, y = response)
plot(Interaction$new(elastic_net_pred))
```




```
pcr_pred = Predictor$new(model = gss_pcr_best, data = features, y = response)
plot(Interaction$new(pcr_pred))
```




```
pls_pred = Predictor$new(model = gss_pls_best, data = features, y = response)
plot(Interaction$new(pls_pred))
```



All of the models rank the interaction strength of features differently, so it's not meaningful to enumerate and interpret them individually. Moreover, all of the effects are quite small. Authoritarianism and polviews seems to appear high many of the rankings, suggesting that these politically-motivated interactions strong, as expected. Interaction strength in the elastic net regression notably drops off to 0 after a certain point, which is expected based on what we know about elastic net eliminating features.

Otherwise, the models show very little consistency in what the penalize and reward. This is an important in terms of model selection, because although elastic net emerges as a clear winner when it comes to mean squared error, its ridge approach may make less appealing for some applications (e.g. where we want to preserve all of the features).