HW3 - Regularization Methods [MACS 301000]

Adarsh Mathew

2/7/2020

Part 1: Conceptual Exercises

Q1: Generate data

```
n_ft <- 20
n_obs <- 1000
set.seed(03022020)

x_sim <- bind_cols(lapply(c(1:(n_ft/2)), function(x) {rnorm(n_obs, 0, 1)}),
                   lapply(c((1+(n_ft/2)):(n_ft - 3)), function(x) {runif(n_obs, -2, 2)}),
                   list(V18 = runif(n_obs, 0, 1)),
                  expand_grid(V19 = seq(-1,1, length.out = 40), V20 = seq(-1, 1, length.out = 25)))
```

```
beta_sim <- rnorm(n_ft, 0, 5)
beta_sim[which(between(beta_sim, -1.5, 1.5))] <- 0
names(beta_sim) <- paste("V", 1:20, sep = '')
```

```
y_sim <- list(y_sim = as.matrix(x_sim) %*% beta_sim + rnorm(n_obs, 0 , 1)) %>% as_tibble()
df_sim <- bind_cols(y_sim, x_sim)
rm(x_sim, y_sim)
```

Q2: Split data

```
df_split <- initial_split(df_sim, prop = 0.1)
df_train <- rsample::training(df_split)
df_test <- rsample::testing(df_split)
```

Q3: Best Subset Selection on Training set

```
## does caret have this?

best_subset <- regsubsets(y_sim ~ ., data = df_train,
                          nvmax = 20)
summ_best_subset <- summary(best_subset)
```

```
# source for this function: https://lagunita.stanford.edu/c4x/HumanitiesSciences/StatLearning/asset/ch6

predict.regsubsets = function(object,newdata,id,...){
      form <- as.formula(object$call[[2]]) # Extract the formula used when we called regsubsets()
      mat <- model.matrix(form, newdata)    # Build the model matrix
      coeffs <- coef(object,id=id)          # Extract the coefficiants of the ith model
```

```r
      xvars = names(coeffs)                # Pull out the names of the predictors used in the ith model
      mat[,xvars]%*%coeffs               # Make predictions using matrix multiplication
}

best_subset_mse <- tibble(mod_no = c(1:(best_subset$np - 1)))

for (i in 1:(best_subset$np - 1)) {
  best_subset_mse[i,"mse_test"] <- mean((predict.regsubsets(best_subset, df_test, i) - df_test$y_sim)^2
  best_subset_mse[i,"mse_train"] <- mean((predict.regsubsets(best_subset, df_train, i) - df_train$y_sim

}

# best_subset_mse <- bind_cols(best_subset_mse, train_rss_mse = best_subset$rss[2:21]/best_subset$nn)
```

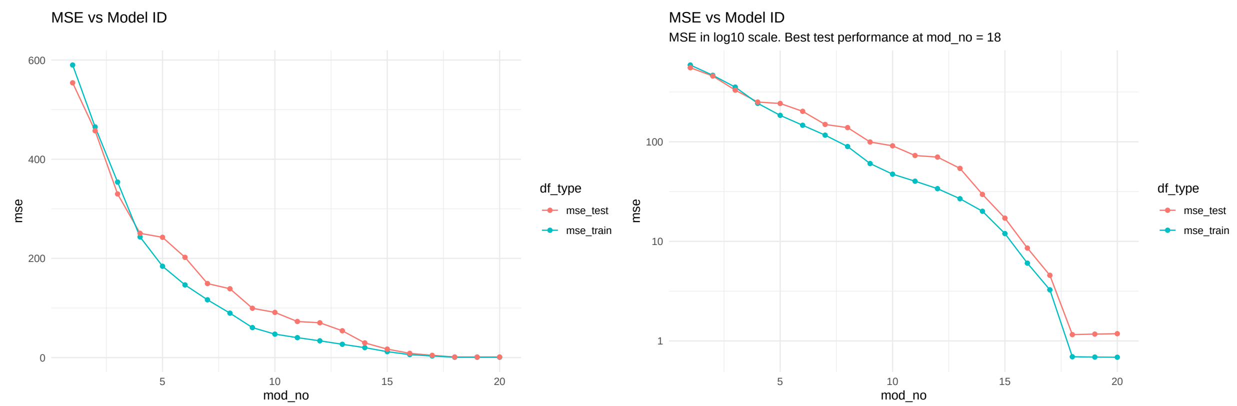
```r
best_subset_mse
```

```
## # A tibble: 20 x 3
##    mod_no mse_test mse_train
##     <int>    <dbl>     <dbl>
##  1      1   554.     590.
##  2      2   457.     465.
##  3      3   330.     354.
##  4      4   251.     243.
##  5      5   243.     184.
##  6      6   202.     147.
##  7      7   149.     117.
##  8      8   139.      89.6
##  9      9    99.5     60.5
## 10     10    91.1     47.3
## 11     11    72.8     40.2
## 12     12    70.2     33.8
## 13     13    54.1     26.8
## 14     14    29.6     20.1
## 15     15    17.1     12.0
## 16     16     8.56     6.04
## 17     17     4.56     3.26
## 18     18     1.16     0.694
## 19     19     1.17     0.689
## 20     20     1.18     0.686
```

Q4, 6: MSE Results

```r
mse_plot <- best_subset_mse %>%
  pivot_longer(cols = c("mse_train", "mse_test"),
               names_to = "df_type", values_to = "mse") %>%
  ggplot() +
  geom_line(aes(x = mod_no, y = mse, colour = df_type)) +
  geom_point(aes(x = mod_no, y = mse, colour = df_type)) +
  ggtitle("MSE vs Model ID") +
  theme_minimal()

mse_plot_log <- best_subset_mse %>%
  pivot_longer(cols = c("mse_train", "mse_test"),
               names_to = "df_type", values_to = "mse") %>%
```

```r
  ggplot() +
  geom_line(aes(x = mod_no, y = mse, colour = df_type)) +
  geom_point(aes(x = mod_no, y = mse, colour = df_type)) +
  scale_y_log10() +
  ggtitle("MSE vs Model ID",
          subtitle = "MSE in log10 scale. Best test performance at mod_no = 18") +
  theme_minimal()

mse_plot + mse_plot_log
```



- Best training performance: Model 20
- Best test performance: Model 18

```r
coef(best_subset, 18)
```

```
## (Intercept)          V1          V2          V4          V5          V6 
## -0.07329022 -6.93569040 -1.67002267 -6.31411337  9.23065476 -5.22499516 
##          V7          V8          V9         V10         V11         V12 
##  3.53404260 11.24065844  3.19542919 -2.92576186  9.38312055  8.08932615 
##         V13         V14         V15         V17         V18         V19 
##  3.38707320  5.16574055  1.54070973  3.34360494 11.05714457  6.50561108 
##         V20 
##  5.03731589
```

```r
beta_sim
```

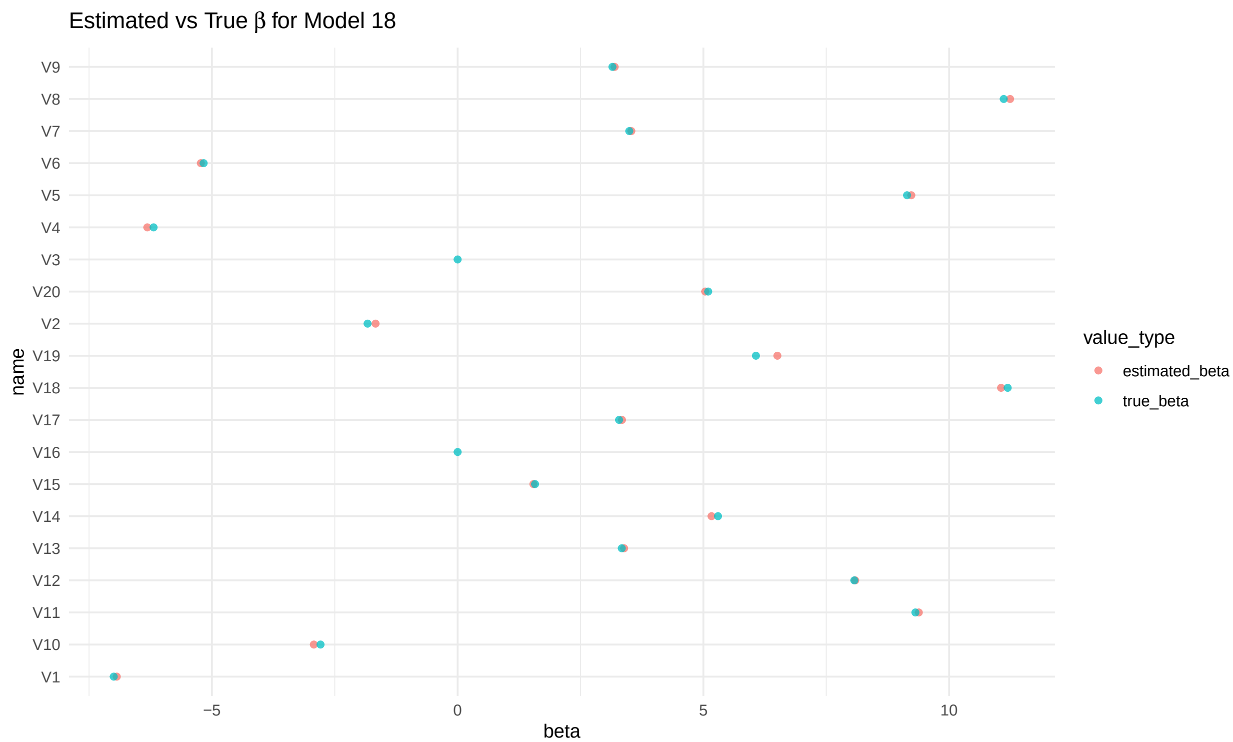
```
##        V1        V2        V3        V4        V5        V6        V7        V8 
## -6.997265 -1.832203  0.000000 -6.186461  9.144247 -5.167956  3.492206 11.111396 
##        V9       V10       V11       V12       V13       V14       V15       V16 
##  3.149478 -2.789873  9.313544  8.070247  3.339392  5.297600  1.575552  0.000000 
##       V17       V18       V19       V20 
##  3.284364 11.191552  6.069124  5.097983
```

```r
beta_sim %>% enframe() %>%
  rename(true_beta = value) %>%
  left_join(coef(best_subset, 18) %>%
              enframe() %>%
              rename(estimated_beta = value)) %>%
  pivot_longer(cols = c("estimated_beta", "true_beta"),
               names_to = "value_type",
               values_to = "beta") %>%
```

```r
  #filter(name != "(Intercept)") %>%
  ggplot() +
  geom_point(aes(x = name, colour = value_type, y = beta), alpha = 0.75) +
  theme_minimal() + coord_flip() +
  ggtitle(bquote("Estimated vs True" ~ beta ~ "for Model 18"))
```

```
## Joining, by = "name"
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



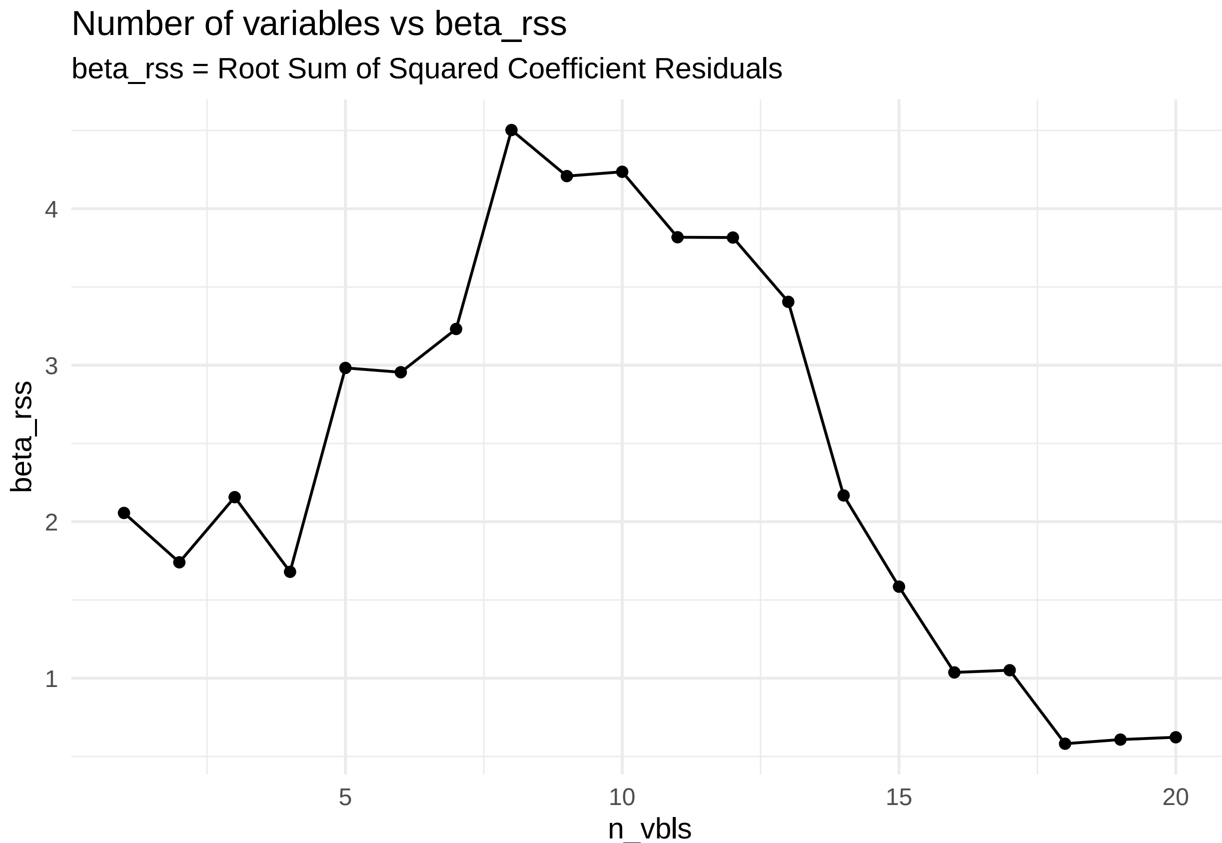
We see that the estimated and true βs for Model 18 overlap for almost all variables, and that variables with $\beta = 0$ are dropped.

```r
beta_diff_sqrt <- function(mod_id) {

  xx_temp <- coef(best_subset, mod_id) %>%
             enframe() %>%
             rename(estimated_beta = value) %>%
            inner_join(beta_sim %>% enframe() %>% rename(true_beta = value),
                       by = "name") %>%
            mutate(beta_diff = (true_beta - estimated_beta)^2)

  return(sqrt(sum(xx_temp$beta_diff)))

}


coef_diff <- tibble(mod_no = c(1:(best_subset$np - 1))) %>%
  mutate(n_vbls = map_dbl(mod_no, ~sum(summ_best_subset$which[.x,])) - 1,
         beta_rss = map_dbl(mod_no, ~beta_diff_sqrt(.x)))
```

```r
coef_diff
```

```
## # A tibble: 20 x 3
##    mod_no n_vbls beta_rss
##     <int>  <dbl>    <dbl>
##  1      1      1    2.06
##  2      2      2    1.74
##  3      3      3    2.16
##  4      4      4    1.68
##  5      5      5    2.98
##  6      6      6    2.96
##  7      7      7    3.23
##  8      8      8    4.50
##  9      9      9    4.21
## 10     10     10    4.24
## 11     11     11    3.82
## 12     12     12    3.82
## 13     13     13    3.41
## 14     14     14    2.17
## 15     15     15    1.58
## 16     16     16    1.04
## 17     17     17    1.05
## 18     18     18    0.581
## 19     19     19    0.608
## 20     20     20    0.623
```

```r
coef_diff %>%
  ggplot() +
  geom_line(aes(x = n_vbls, y = beta_rss)) +
  geom_point(aes(x = n_vbls, y = beta_rss)) +
  ggtitle("Number of variables vs beta_rss",
          subtitle = "beta_rss = Root Sum of Squared Coefficient Residuals") +
  theme_minimal()
```



We see that the minimum value is at `n_vbls = 18`, which matches our result from the test MSE metric.

Part 2: Application Exercises

```
gss_train <- read_csv("../data/gss_train.csv",
    col_types = cols(childs = col_integer(),
        egalit_scale = col_integer()))

View(gss_train)

gss_test <- read_csv("../data/gss_test.csv",
    col_types = cols(childs = col_integer(),
        egalit_scale = col_integer()))

set.seed(03022020)

View(gss_test)
```

Linear Model

```
model_linear <- train(form = egalit_scale ~ ., data = gss_train,
                      method = "lm")
```

```
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit
## may be misleading
```

```
print("Training MSE for lm:")
```

```
## [1] "Training MSE for lm:"
```

```
mean((predict(model_linear, newdata = gss_train) - gss_train$egalit_scale)^2)
```

```
## [1] 55.12264
```

```r
print("Testing MSE for lm:")
```

```
## [1] "Testing MSE for lm:"
```

```r
mean((predict(model_linear, newdata = gss_test) - gss_test$egalit_scale)^2)
```

```
## [1] 63.21363
```

Ridge Regression

```r
model_ridge <- cv.glmnet(x = gss_train %>% select(-egalit_scale) %>% as.matrix(),
                         y = gss_train$egalit_scale,
                         type.measure = "mse",
                         alpha = 0,
                         nfolds = 10)

print("Training MSE for ridge:")
```

```
## [1] "Training MSE for ridge:"
```

```r
mean((predict(model_ridge, newx = gss_train %>%
                select(-egalit_scale) %>%
                as.matrix()) - gss_train$egalit_scale)^2)
```

```
## [1] 59.43662
```

```r
print("Testing MSE for ridge:")
```

```
## [1] "Testing MSE for ridge:"
```

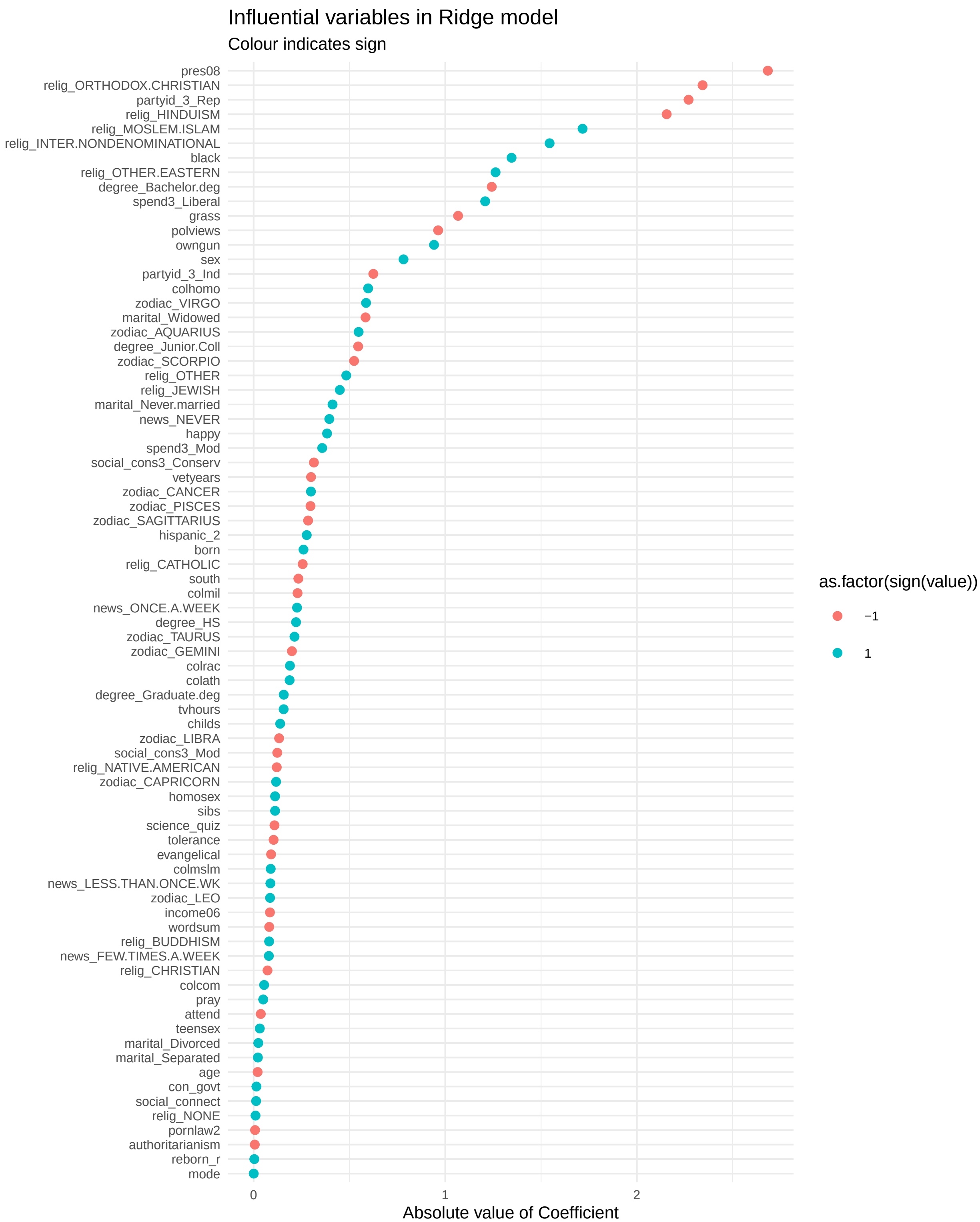
```r
mean((predict(model_ridge, newx = gss_test %>%
                select(-egalit_scale) %>%
                as.matrix()) - gss_test$egalit_scale)^2)
```

```
## [1] 61.44488
```

```r
# model_ridge$results
```

```r
coef(model_ridge, s = "lambda.1se") %>%
  broom::tidy() %>%
  filter(row != "(Intercept)") %>%
  ggplot(aes(abs(value), reorder(row, abs(value)), colour = as.factor(sign(value)))) +
  geom_point() +
  labs(title = "Influential variables in Ridge model",
       subtitle = "Colour indicates sign",
       x = "Absolute value of Coefficient",
       y = NULL) +
  theme_minimal(base_size = 8)
```

```
## Warning: 'tidy.dgCMatrix' is deprecated.
## See help("Deprecated")

## Warning: 'tidy.dgTMatrix' is deprecated.
## See help("Deprecated")
```



Lasso Regression

```
model_lasso <-  cv.glmnet(x = gss_train %>% select(-egalit_scale) %>% as.matrix(),
                          y = gss_train$egalit_scale,
```

```r
                         type.measure = "mse",
                         alpha = 1, nfolds = 10)

print("Training MSE for Lasso:")
```

```
## [1] "Training MSE for Lasso:"
```

```r
mean((predict(model_lasso, newx = gss_train %>%
                select(-egalit_scale) %>%
                as.matrix()) - gss_train$egalit_scale)^2)
```

```
## [1] 60.35195
```

```r
print("Testing MSE for Lasso:")
```

```
## [1] "Testing MSE for Lasso:"
```

```r
mean((predict(model_lasso, newx = gss_test %>%
                select(-egalit_scale) %>%
                as.matrix()) - gss_test$egalit_scale)^2)
```

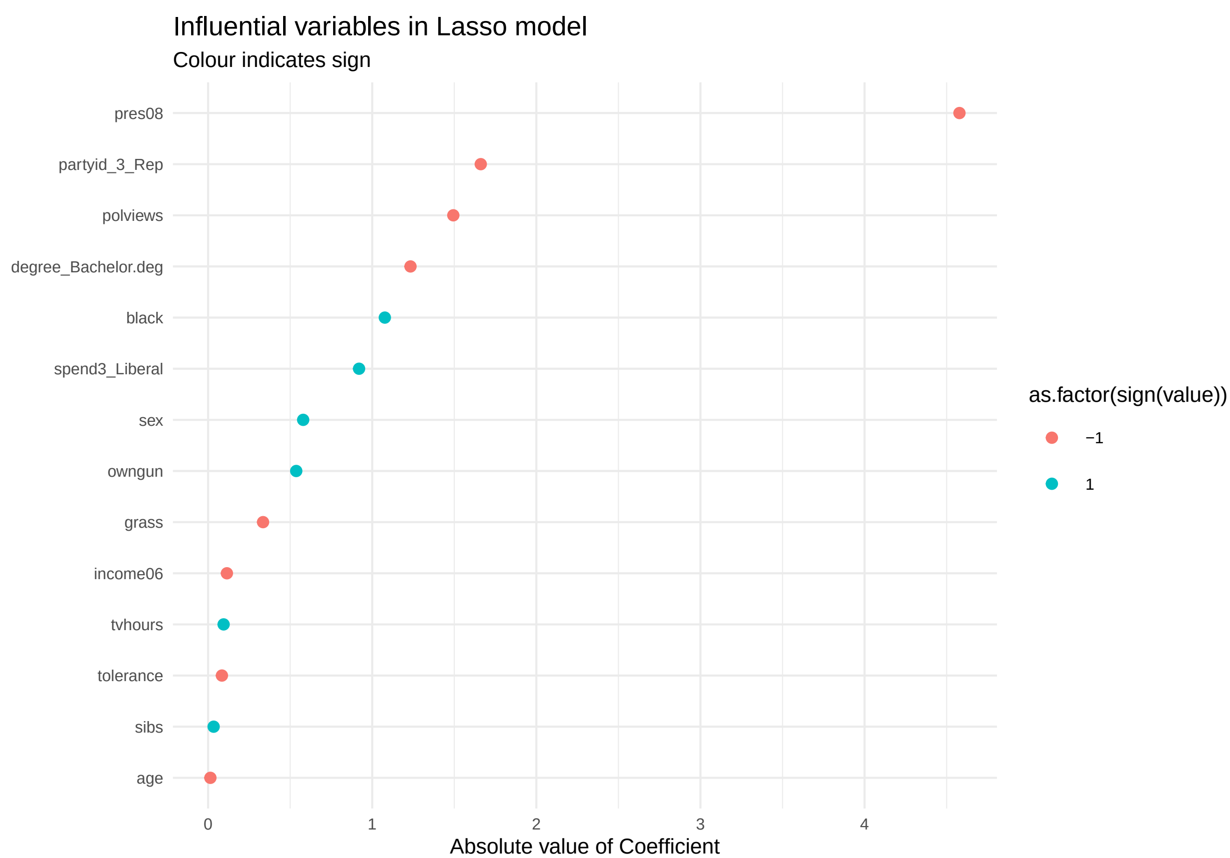
```
## [1] 62.09828
```

```r
model_lasso$nzero[which(model_lasso$lambda == model_lasso$lambda.1se)]
```

```
## s23
##  14
```

```r
coef(model_lasso, s = "lambda.1se") %>%
  broom::tidy() %>%
  filter(row != "(Intercept)") %>%
  ggplot(aes(abs(value), reorder(row, abs(value)), colour = as.factor(sign(value)))) +
  geom_point() +
  labs(title = "Influential variables in Lasso model",
       subtitle = "Colour indicates sign",
       x = "Absolute value of Coefficient",
       y = NULL) +
  theme_minimal(base_size = 8)
```

```
## Warning: 'tidy.dgCMatrix' is deprecated.
## See help("Deprecated")

## Warning: 'tidy.dgTMatrix' is deprecated.
## See help("Deprecated")
```



Thus, there are 14 non-zero cofficients for the model with optimal lamda = lamda.1se.

Elastic Net

```
glmnet_out_fun <- function(alpha_value){
  set.seed(03022020)
  cv_model_obj <- cv.glmnet(x = gss_train %>% select(-egalit_scale) %>% as.matrix(),
                            y = gss_train$egalit_scale,
                            alpha = alpha_value,
                            type.measure = "mse", nfolds = 10)

  xx_out <- tibble(lambda_min = cv_model_obj$lambda.min,
                   lambda_1se = cv_model_obj$lambda.1se,
                   mse_min = cv_model_obj$cvm[which(cv_model_obj$lambda == cv_model_obj$lambda.min)],
                   mse_1se = cv_model_obj$cvm[which(cv_model_obj$lambda == cv_model_obj$lambda.1se)]
                   )

  return(xx_out)
}

tuning_grid <- tibble(alpha = seq(0, 1, by = 0.1))

enet_df <- tuning_grid %>%
  mutate(out_pm = map(alpha, ~glmnet_out_fun(.x))) %>%
  unnest(cols = c(out_pm))
```

```r
enet_df %>% arrange(mse_min)
```

```
## # A tibble: 11 x 5
##    alpha lambda_min lambda_1se mse_min mse_1se
##    <dbl>      <dbl>      <dbl>   <dbl>   <dbl>
##  1   0.6      0.323      0.819    59.9    61.6
##  2   0.5      0.388      0.983    59.9    61.6
##  3   0.7      0.277      0.702    59.9    61.6
##  4   0.8      0.242      0.614    59.9    61.5
##  5   0.9      0.215      0.546    59.9    61.5
##  6   1        0.194      0.492    59.9    61.5
##  7   0.4      0.485      1.23     59.9    61.7
##  8   0.3      0.646      1.49     59.9    61.5
##  9   0.2      0.883      2.24     59.9    61.7
## 10   0.1      1.47       3.72     60.1    61.8
## 11   0        2.45       8.20     61.1    62.5
```

Thus, minimum MSE is at $\alpha = 0.6$.

```r
model_elastic_final <- cv.glmnet(x = gss_train %>% select(-egalit_scale) %>% as.matrix(),
                          y = gss_train$egalit_scale, alpha = 0.6,
                          type.measure = "mse", nfolds = 10)

print("Testing MSE for Elastic Net:")
```

```
## [1] "Testing MSE for Elastic Net:"
```

```r
mean((predict(model_elastic_final, newx = gss_test %>%
                select(-egalit_scale) %>%
                as.matrix()) - gss_test$egalit_scale)^2)
```

```
## [1] 62.33619
```

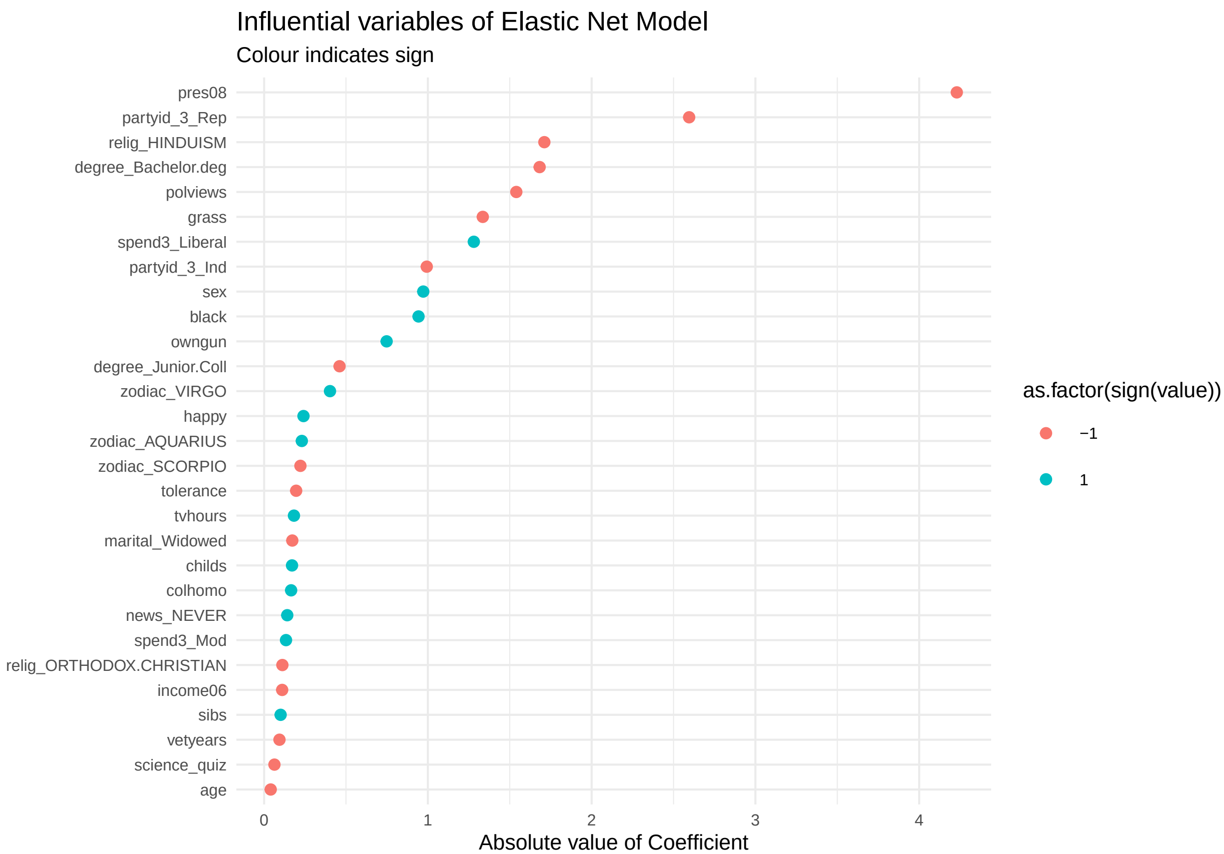
```r
model_elastic_final$nzero[which(model_elastic_final$lambda == model_elastic_final$lambda.min)]
```

```
## s33
##  29
```

29 non-zero coefficients for $\alpha = 0.6$.

```r
coef(model_elastic_final, s = "lambda.min") %>%
  broom::tidy() %>%
  filter(row != "(Intercept)") %>%
  ggplot(aes(abs(value), reorder(row, abs(value)), colour = as.factor(sign(value)))) +
  geom_point() +
  labs(title = "Influential variables of Elastic Net Model",
       subtitle = "Colour indicates sign",
       x = "Absolute value of Coefficient",
       y = NULL) +
  theme_minimal(base_size = 8)
```

```
## Warning: 'tidy.dgCMatrix' is deprecated.
## See help("Deprecated")

## Warning: 'tidy.dgTMatrix' is deprecated.
## See help("Deprecated")
```



Model Selection

Testing MSE is largely comparable across models. Ridge Regression gives us the best test-MSE, but the marginal improvement could be attributed to the model retaining all variables. Given the challenges in interpreting the coefficients, it wouldn't be my preferred choice. Elastic Net – which has the next best test-MSE and gives us a sparser model with 29 variables – would be the better choice here. That said, we're still running variants of the linear regression model here. We haven't evaluated our data for its adherence to the assumptions of the linear model. If we wanted to improve our fit, that would be the first place to begin.