# LASSO Regression

Jaime Davila

4/6/2022

## Previously on ADM

Let's recall the steps to set-up our ridge regression model

### Dataset loading and training/testing split

We can load our dataset by doing:

```
library(ISLR2)
data(Credit)
credit.tbl <- as_tibble(Credit)
```

And create our training/testing datasets by:

```
library(tidymodels)
tidymodels_prefer()

set.seed(654321)
credit.split <- initial_split(credit.tbl, prop=0.8)
credit.train.tbl <- training(credit.split)
credit.test.tbl <- testing(credit.split)
```

### Setting up the ridge regression model

First we set-up our ridge model below. Notice that `mixture=0` and that set-up our `penalty` (or $\lambda$ ) to be tuned later on:

```
ridge.model <-
  linear_reg(mixture = 0, penalty=tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")
```

Below we create our recipe and our workflow. Notice that we need to use `step_normalize()` to make sure all the variables are standarized.

```
ridge.recipe <-
  recipe(formula = Balance ~ ., data = credit.train.tbl) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

ridge.wf <- workflow() %>%
```

```
  add_recipe(ridge.recipe) %>%
  add_model(ridge.model)
```

### Setting up the parameters for the optimization

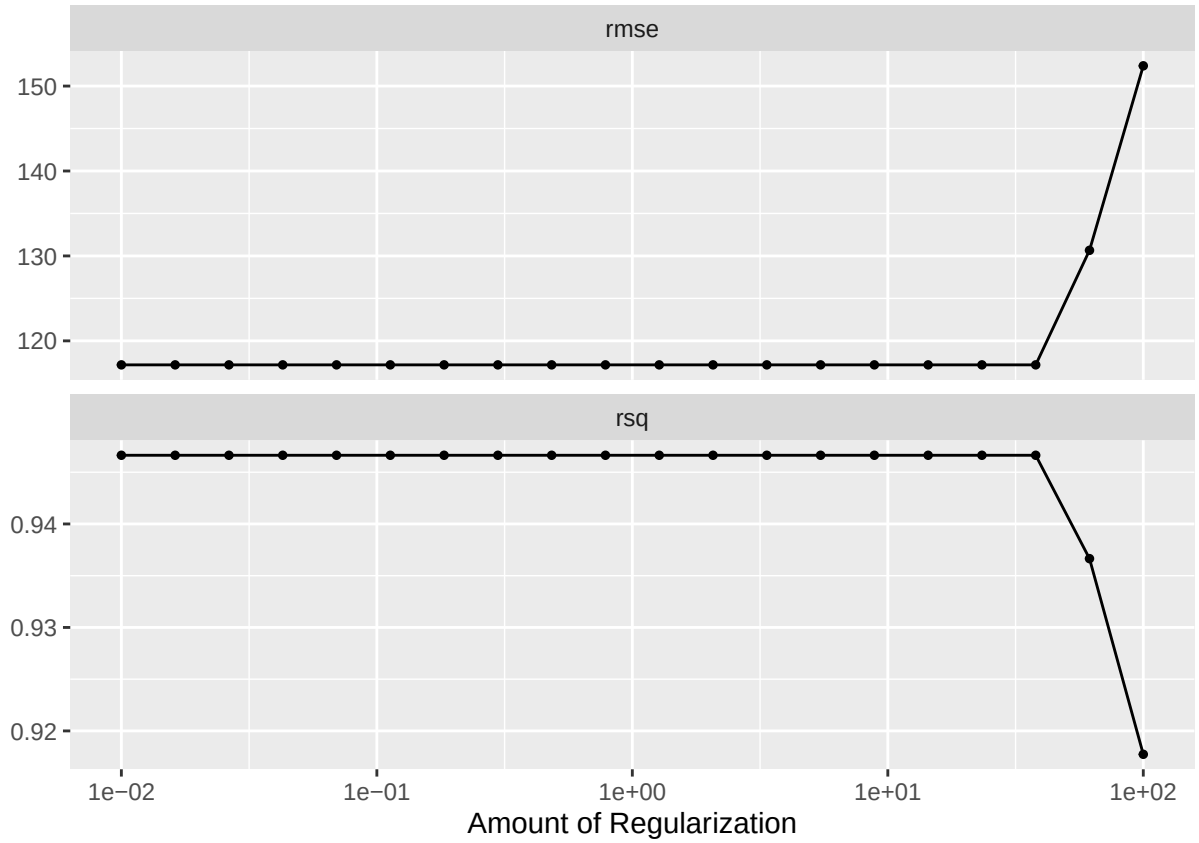
Below we create our 10-fold cross-validation dataset using `vfold_cv()` and we create a logarithmic grid using `penalty()`

```
credit.fold <- vfold_cv(credit.train.tbl, v = 10)

penalty.grid <-
  grid_regular(penalty(range = c(-2, 2)), levels = 20)
```

### Optimizing the penalty ($\lambda$)

The following code allows to optimize the penalty parameter

```
tune.res <- tune_grid(
  ridge.wf,
  resamples = credit.fold,
  grid = penalty.grid
)
autoplot(tune.res)
```



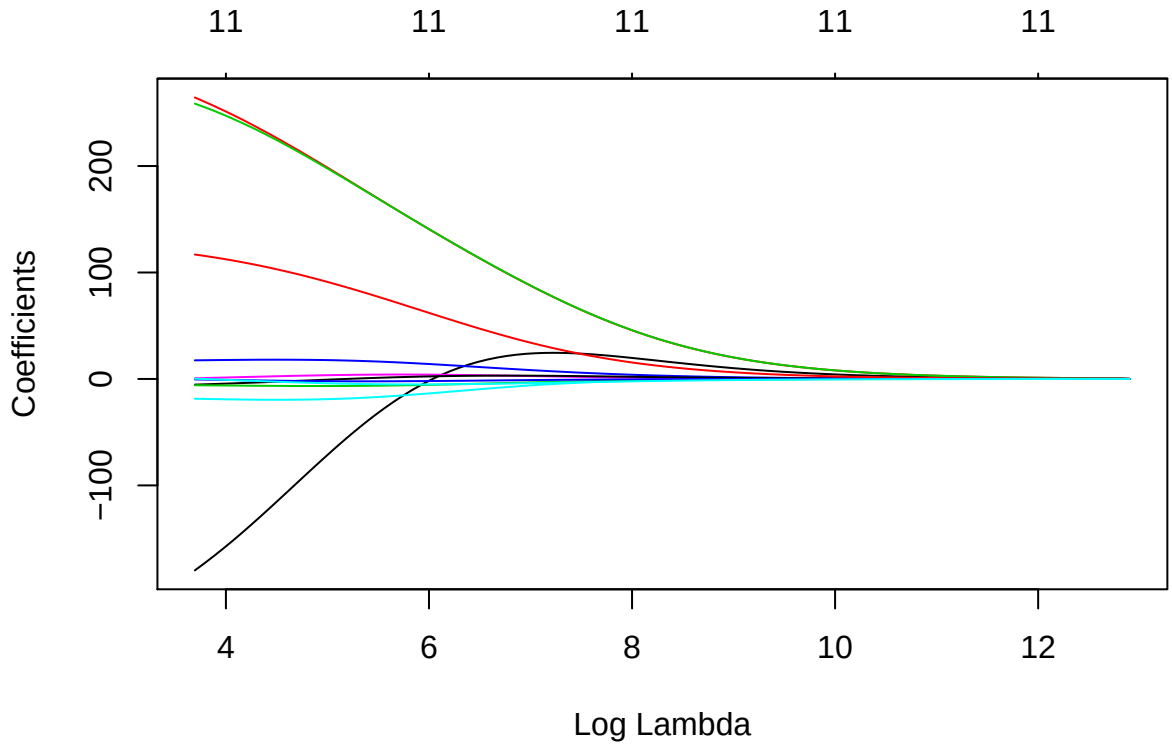

And finally we can:

- Select the best parameter
- Explore the influence of lambda on the coefficients
- Calculate our $R^2$ using the testing dataset
- Show the importance of the variables

```
(best.penalty <- select_best(tune.res, metric = "rsq"))
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <fct>
## 1    0.01 Preprocessor1_Model01
```

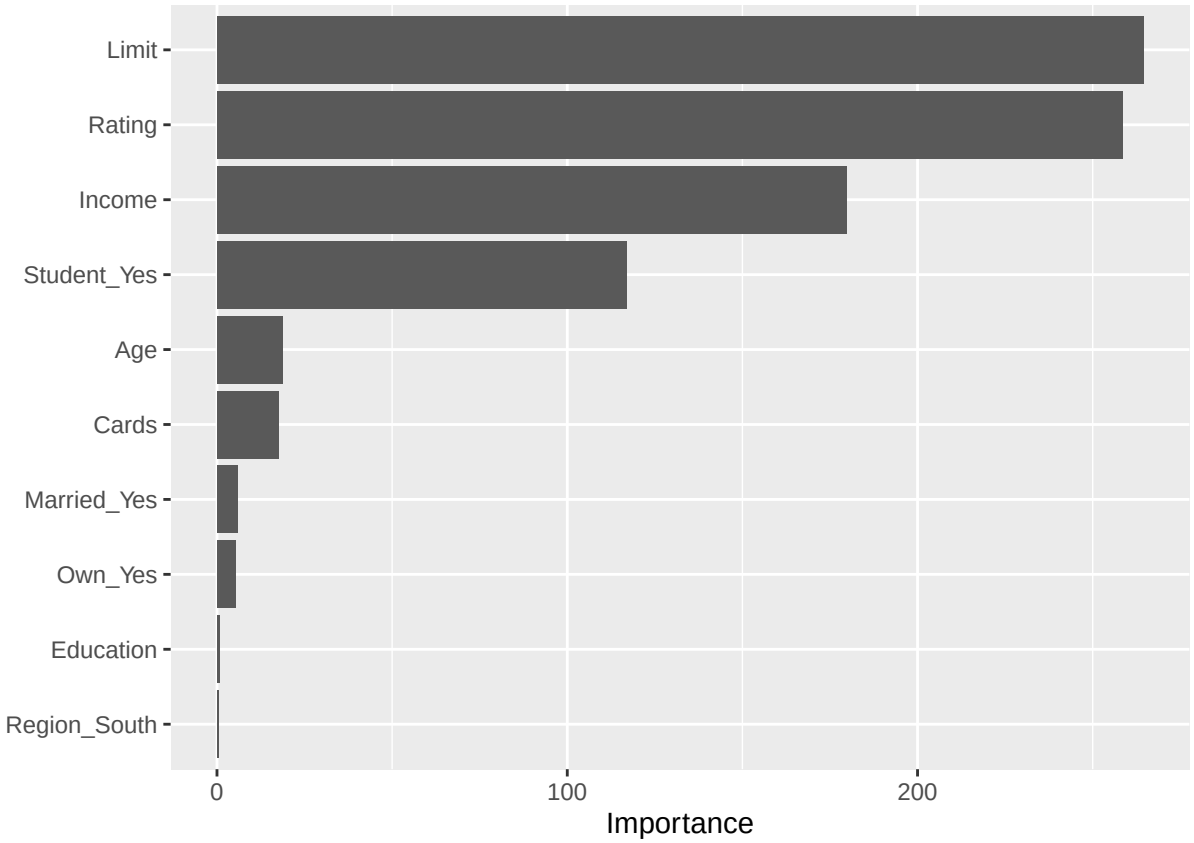
```
ridge.final.wf <- finalize_workflow(ridge.wf, best.penalty)
ridge.final.fit <- fit(ridge.final.wf, data = credit.train.tbl)

ridge.final.fit %>%
  extract_fit_engine() %>%
  plot(xvar = "lambda")
```




```
augment(ridge.final.fit, new_data = credit.test.tbl) %>%
  rsq(truth = Balance, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.932
```

```
library(vip)
extract_fit_parsnip(ridge.final.fit) %>%
  vip()
```




## LASSO regression

When we were dealing with ridge regression we minimized the following function:

$$RSS(\beta_0, \ldots, \beta_p) + \lambda \sum_{i=1}^{p} \beta_j^2$$

The key thing to realize is that we are optimizing over the choice of our coefficients $\beta_0, \ldots, \beta_p$ and that $\lambda$ is a fixed value which will be finding later on.

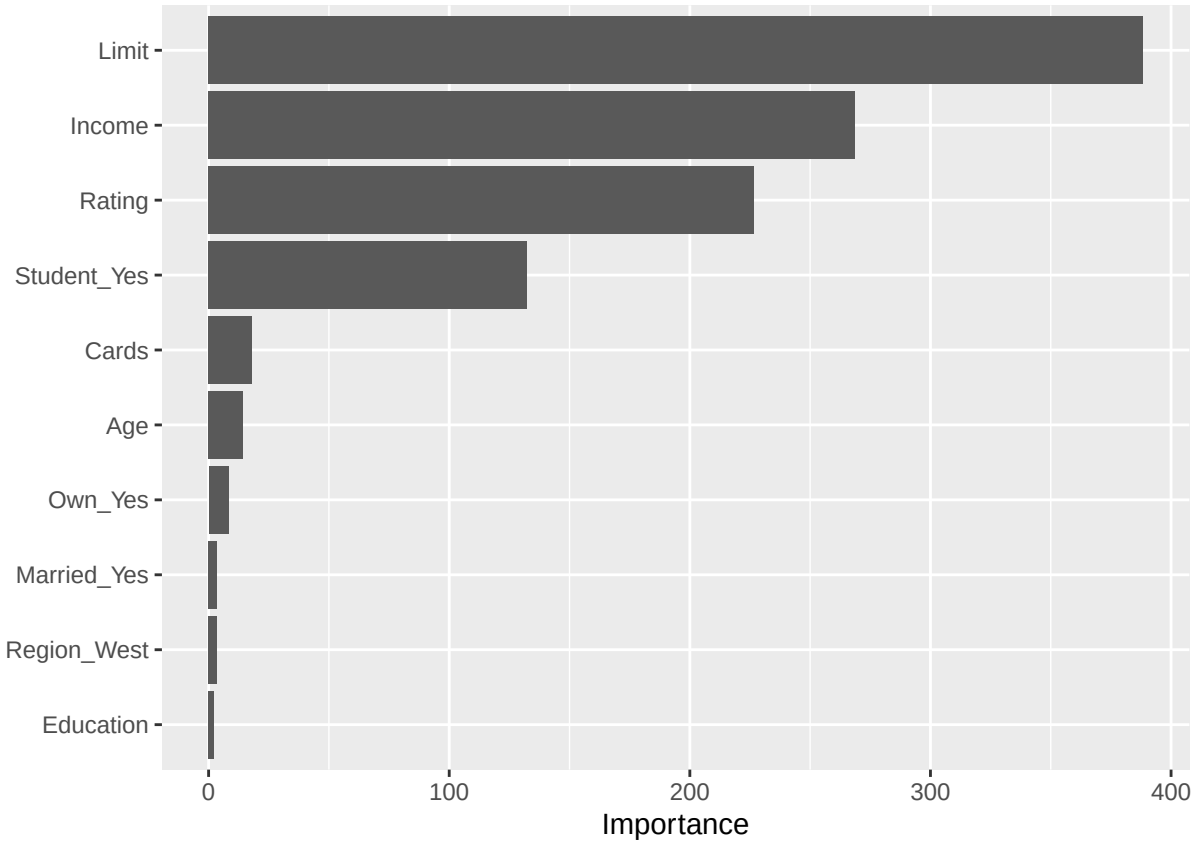
In LASSO we minimize a slightly different optimization function with profound consequences on our model. The function is:

$$RSS(\beta_0, \ldots, \beta_p) + \lambda \sum_{i=1}^{p} |\beta_j|$$

The following exercises will walk you through the setting up of the LASSO model and how the result differs from the ridge regression

1. Set up a LASSO regression model by using `tidymodels()` making sure you use the parameter `mixture=1`. For your $\lambda$ use a value of 1. Calculate the $R^2$ and the order of importance of variables

```
lasso.model <-
  linear_reg(mixture = 1, penalty=1) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

lasso.recipe <-
  recipe(formula = Balance ~ ., data = credit.train.tbl) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

lasso.wf <- workflow() %>%
  add_recipe(lasso.recipe) %>%
  add_model(lasso.model)

lasso.fit <- fit(lasso.wf, credit.train.tbl)

extract_fit_parsnip(lasso.fit)%>%
  vip()
```




```
augment(lasso.fit, credit.test.tbl)%>%
  rsq(truth = Balance, estimate = .pred)
```

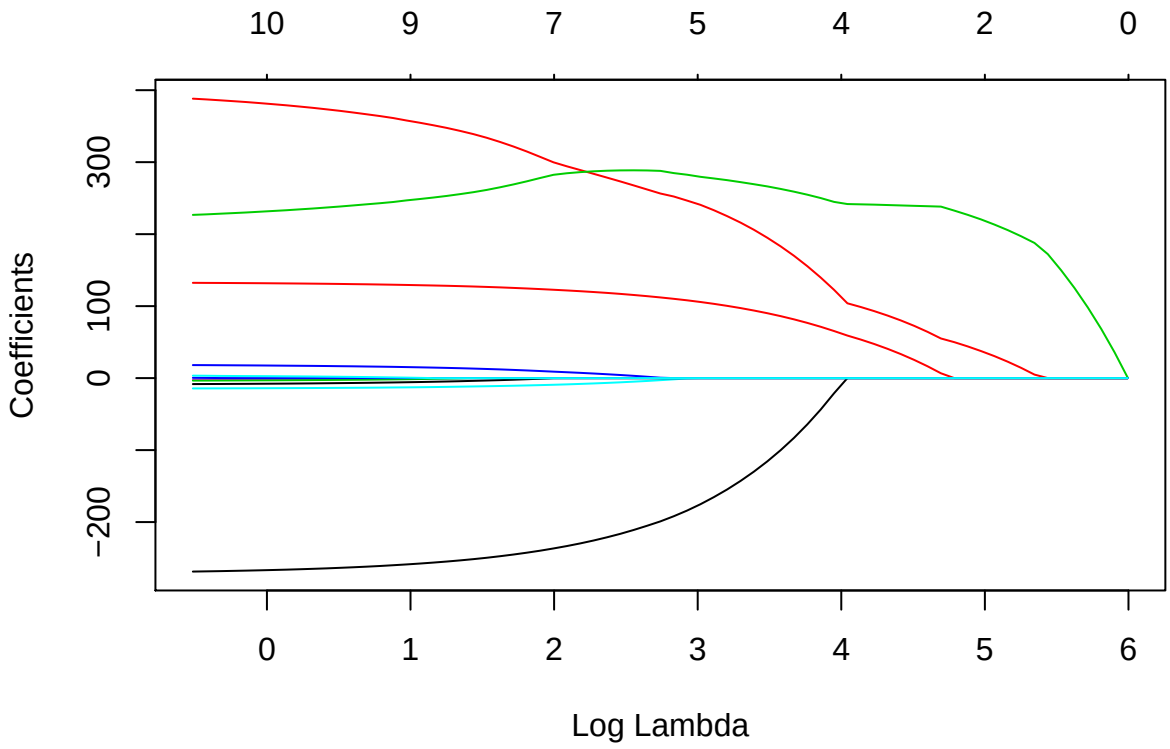
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
```

```
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.946
```

We see that Limit, Income, Rating, and Student_Yes have the highest importance, followed by a pretty large gap before the next predictor. We see an $R^2$ of 0.946 when we fit the model on the testing dataset.

2. Compare the following plot to fig 6.6 from ISLR (left panel). How do you interpret this plot? What do you think the numbers on the upper side of the plot mean (Notice they go like 10,9,7,5,4,2,0)? How is this plot different from the corresponding plot from the ridge regression?

```
lasso.fit %>%
  extract_fit_engine() %>%
  plot(xvar = "lambda")
```



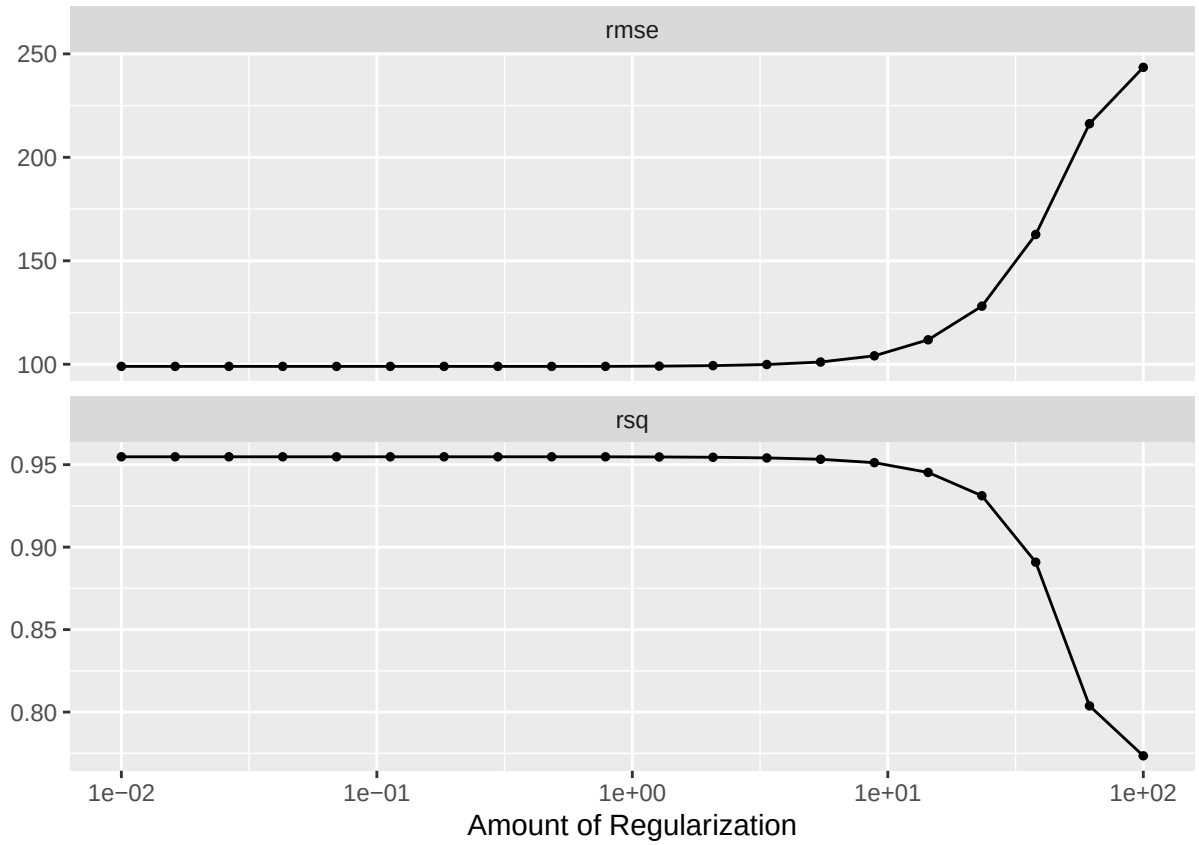

This plot shows the values of different variables' coefficients as our penalty value (lambda) increases. As lambda increases, the coefficient of the variables decreases. The numbers along the top of the graph seem to represent the number of non-zero predictors at the given value of lambda.

This plot differs from the corresponding ridge regression plot because in ridge regression no coefficients reach exactly 0. In ridge, they get very close to 0, but never become 0. On the other hand, the lasso plot shows coefficients reaching 0 exactly and thus serves as a method variable selection as well as shrinkage.

3. Experiment with your penalty grid and use cross-validation optimize the parameter $\lambda$. Calculate your $R^2$ on your testing dataset and determine the importance of the features. Compare this to your results using ridge regression.

```
lasso.model <-
  linear_reg(mixture = 1, penalty=tune()) %>%
  set_mode("regression") %>%
```

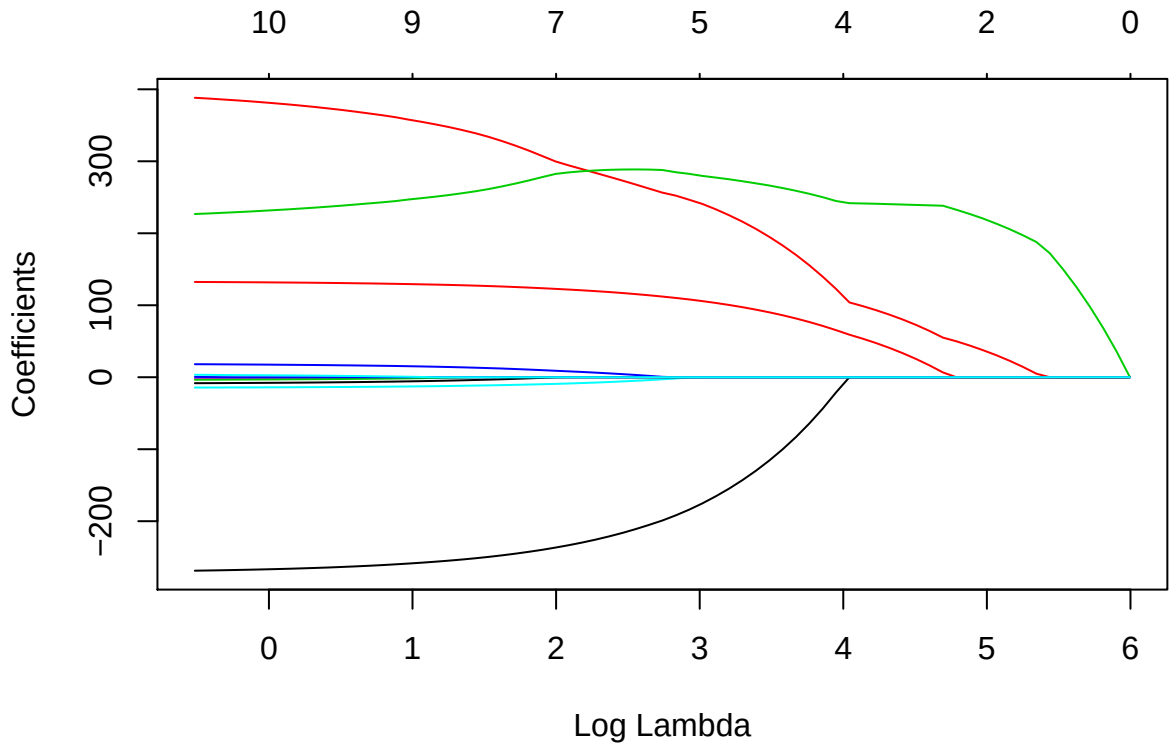
```
  set_engine("glmnet")

lasso.recipe <-
  recipe(formula = Balance ~ ., data = credit.train.tbl) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

lasso.wf <- workflow() %>%
  add_recipe(lasso.recipe) %>%
  add_model(lasso.model)


credit.fold <- vfold_cv(credit.train.tbl, v = 10)

penalty.grid <-
  grid_regular(penalty(range = c(-2, 2)), levels = 20)

tune.res <- tune_grid(
  lasso.wf,
  resamples = credit.fold,
  grid = penalty.grid
)
autoplot(tune.res)
```




```
#show_best(tune.res, metric = "rsq")
(best.penalty <- select_best(tune.res, metric = "rsq"))
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <fct>
## 1    0.01 Preprocessor1_Model01
```

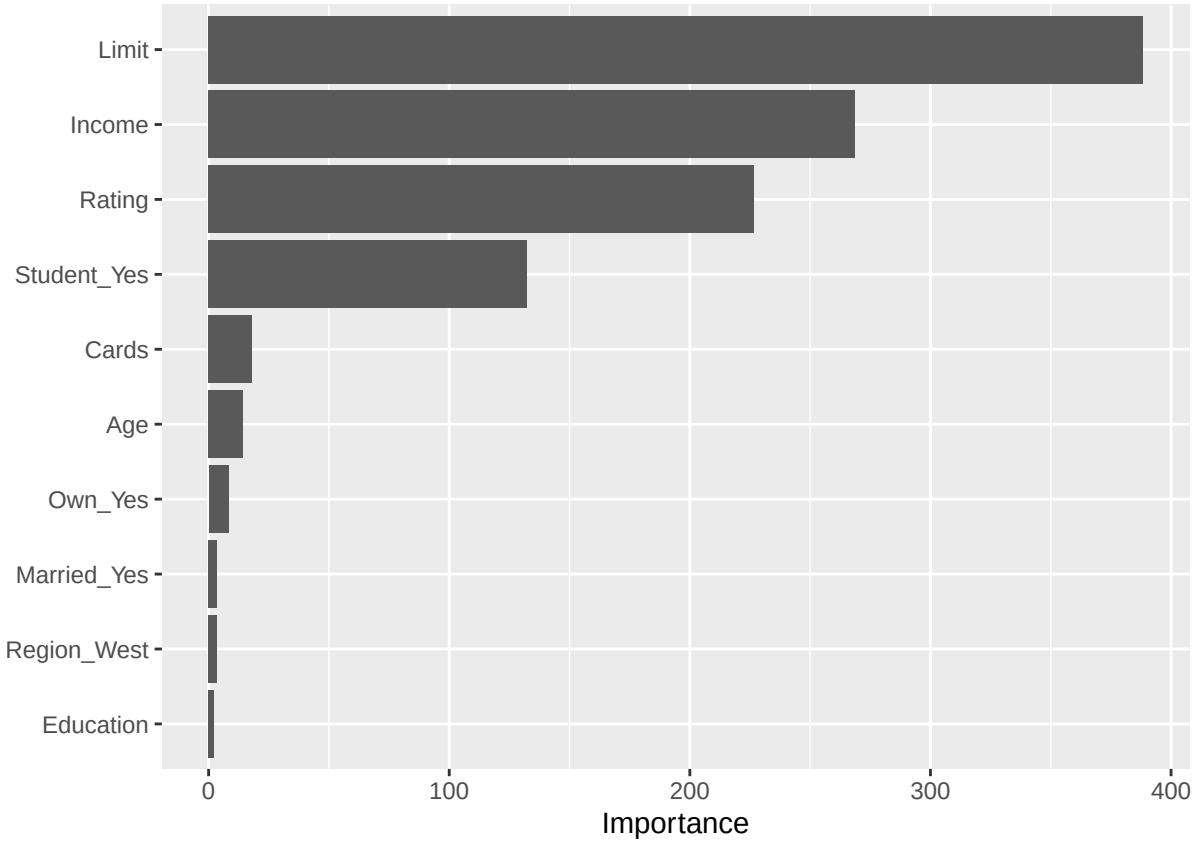
```
lasso.final.wf <- finalize_workflow(lasso.wf, best.penalty)
lasso.final.fit <- fit(lasso.final.wf, data = credit.train.tbl)

lasso.final.fit %>%
  extract_fit_engine() %>%
  plot(xvar = "lambda")
```




```
#R^2
augment(lasso.final.fit, new_data = credit.test.tbl) %>%
  rsq(truth = Balance, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.946
```

```
#Importance
extract_fit_parsnip(lasso.final.fit) %>%
  vip()
```




Note: For some reason, the optimal value of lambda keeps changing (I think it is because the values are so close than when we restart the cross validation process we get slightly different values, but the $R^2$ stays constant at 0.945-0.946).

When we go through and use cross-validation to optimize lambda, we get an optimal lambda value of between 0.01 and 1.5 (see note above), which gives a corresponding $R^2$ estimate of 0.946. We see that this is a slight improvement over the ridge regression, which gave an $R^2$ of 0.932. We see that the importance of variables did slightly change - with lasso income is the second most important variable (followed by rating and student), while with ridge regression rating was the second most important (followed by income and student). In both cases, limit was the most important predictor.

## A word on the geometry of LASSO and ridge

Ridge and LASSO regression can be interpreted as minimizing $RSS(\beta_0, \ldots, \beta_p)$ subject to the constraint (in this case note that $s$ is a fixed parameter)

- For ridge regression, $\sum_{j=1}^{p} (\beta_j)^2 \leq s$
- For LASSO regression, $\sum_{j=1}^{p} |\beta_j| \leq s$

One way to think about this is that we are allowed a budget of $s$ that we are spending across the sum of the $\beta_j^2$ for ridge and the sum of $|\beta_j|$ for LASSO. Notice that if $s$ tends to infinity we end up with just linear regression and that if $s$ is very small we force all of the coefficients to go to zero (Implying that $s$ and $\lambda$ from our first formulation are inversely related)

Furthermore notice that if $p = 2$ the constraint for ridge consists of the unit disc with radius $\sqrt{(s)}$, while the constraint for LASSO is a diamond with diagonal $\frac{s}{2}$. Check figure 6.7 from ISLR from an illustration of how this observation implies that in LASSO we tend to choose values on the corners of the diamond, hence we are preferring solutions were the input variables are equal to zero.

## The College dataset

In the following set of exercises we will be exploring the `College` dataset from the `ISLR2` package. For more information on this dataset please consult `?College`. Let's start by loading our dataset and creating a training/testing dataset

```
data(College)
college.tbl <- as_tibble(College) %>%
  select(-c(Accept, Enroll))

set.seed(123456)
college.split <- initial_split(college.tbl, prop=0.8)
college.train.tbl <- training(college.split)
college.test.tbl <- testing(college.split)
```

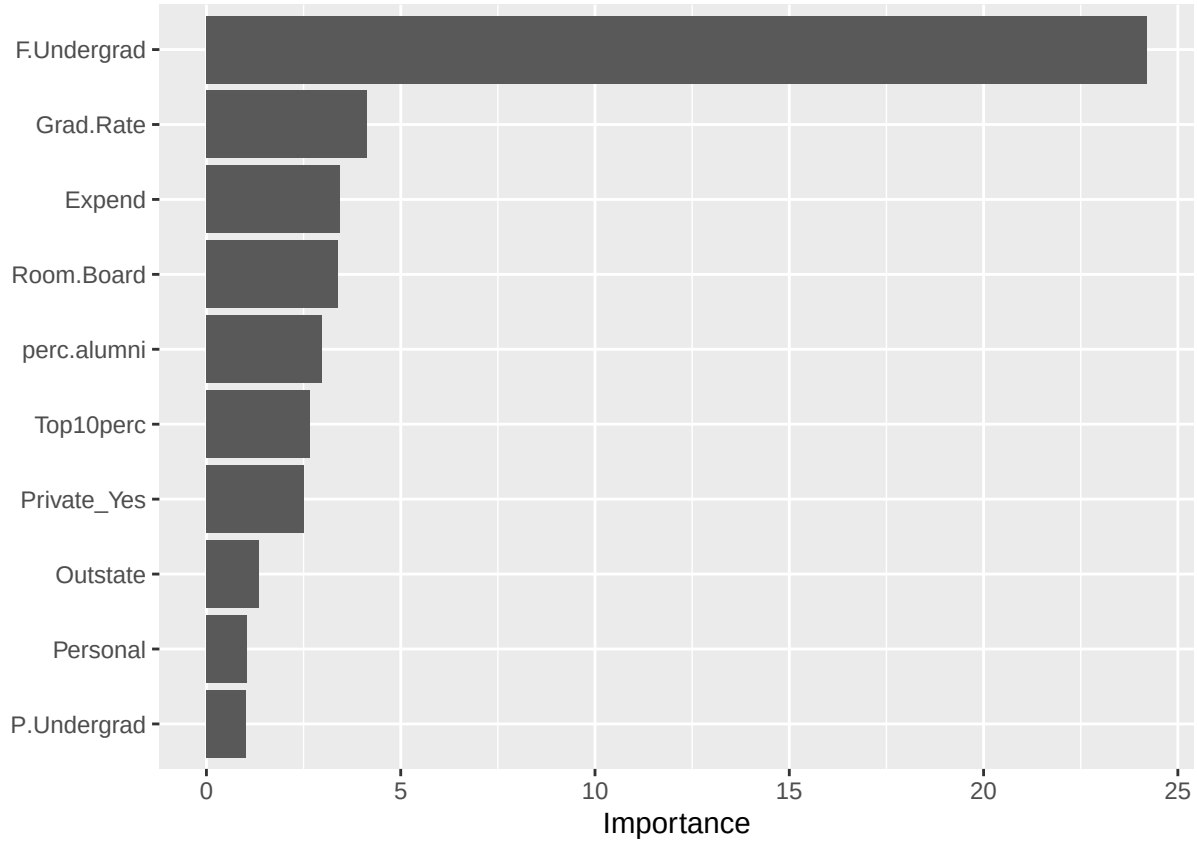
We are interested in predicting the percent of acceptance (`pct.accept`) based on the other variables.

4. Create a linear model to predict `Apps` based on the other variables. What is the $R^2$ on the testing dataset? Plot the input variable importance

```
college.recipe <-
  recipe(formula= Apps ~ ., data=college.train.tbl) %>%
  step_dummy(all_nominal_predictors())

lm.model <- linear_reg()%>%
  set_mode("regression")%>%
  set_engine("lm")

lm.wflow <- workflow()%>%
  add_recipe(college.recipe) %>%
  add_model(lm.model)

lm.fit <- fit(lm.wflow, college.train.tbl)

#R^2
augment(lm.fit, college.test.tbl)%>%
  rsq(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.839
```

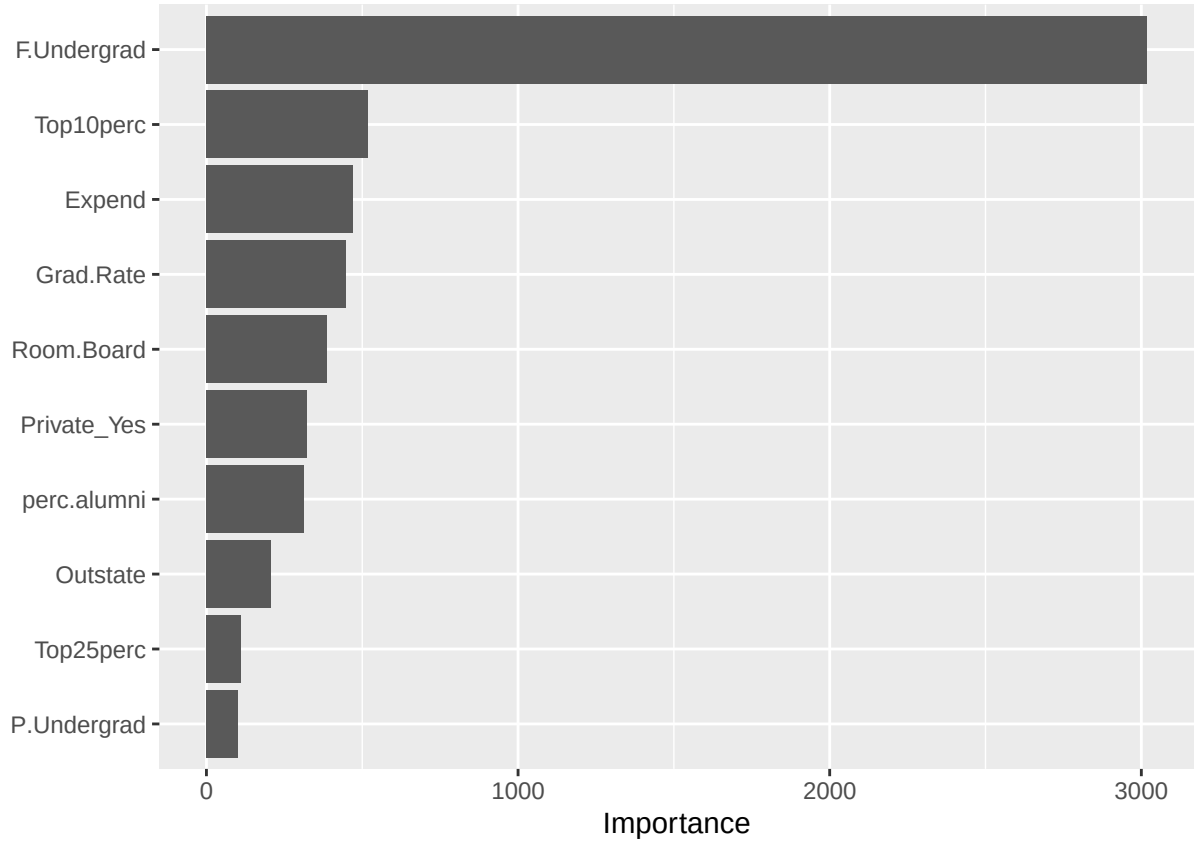
```
#Importance
extract_fit_parsnip(lm.fit)%>%
  vip()
```




Here, we see an $R^2$ value of 0.839. From the importance plot, we see that F.Undergrad (full time undergrads) is by far the most important predictor, followed by Grad.Rate and Expend.

5. Create a LASSO model to predict `Apps` based on the other variables. What is the $R^2$ on the testing dataset? Plot the input variable importance

```
#ASSUMING lambda = 1 given no specification and no cross-validation in this problem
lasso.model <-
  linear_reg(mixture = 1, penalty=1) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

lasso.recipe <-
  recipe(formula = Apps ~ ., data = college.train.tbl) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())

lasso.wf <- workflow() %>%
  add_recipe(lasso.recipe) %>%
  add_model(lasso.model)

lasso.fit <- fit(lasso.wf, college.train.tbl)

#Importance
extract_fit_parsnip(lasso.fit)%>%
  vip()
```




```
#R^2
augment(lasso.fit, college.test.tbl)%>%
  rsq(truth = Apps, estimate = .pred)
```

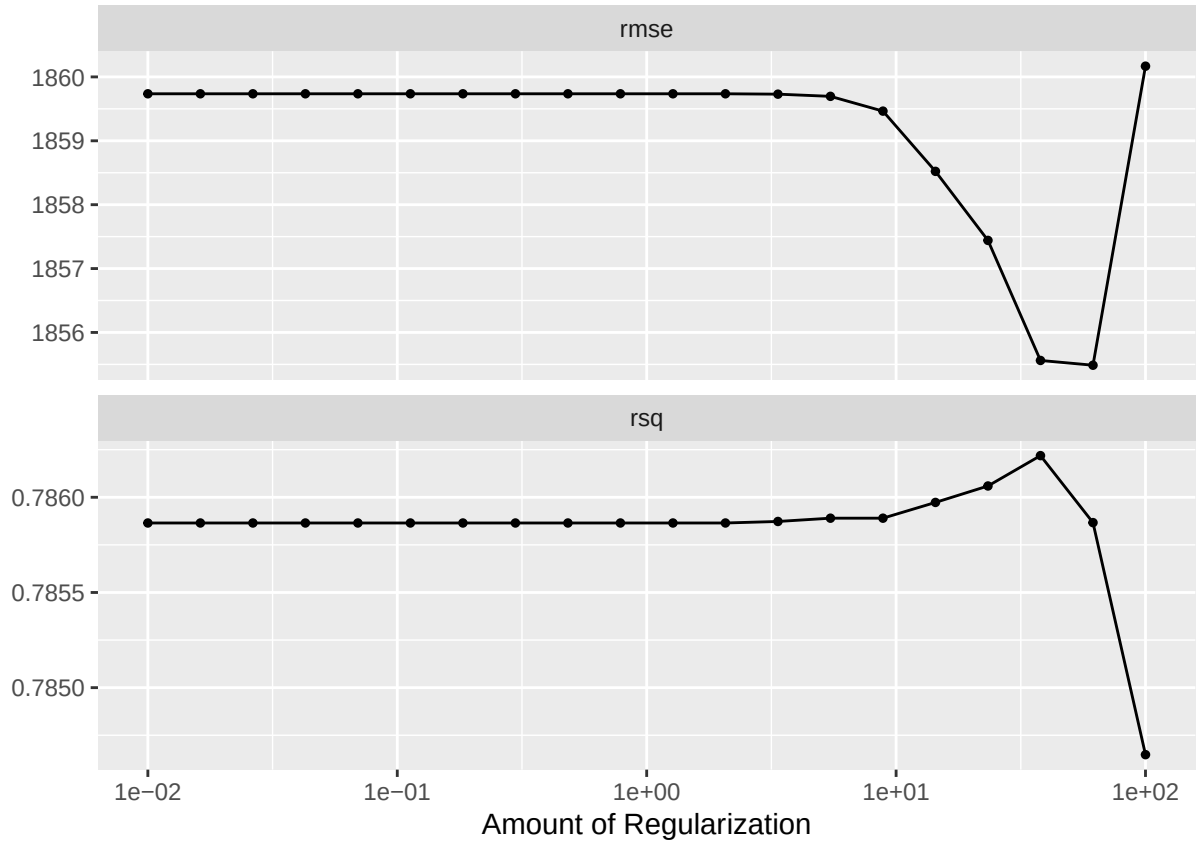
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.839
```

When we use $\lambda = 1$, we get the same $R^2$ value as we did before: 0.839. However, there is some change in the importance of variables. Number of full time undergrads is still far and away the most important, but when we do the lasso importance we see the Top10perc variable move into second most important, while the Expend predictor is third most important.

I wasn't sure if we should cross-validate to optimize lambda, so that's what I did below. However, similar to what happened in #3, we have a similar issue where the optimal lambda/penalty keeps changing due to how close all the different $R^2$ values are.

```
lasso.model <-
  linear_reg(mixture = 1, penalty=tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

lasso.recipe <-
  recipe(formula = Apps ~ ., data = college.train.tbl) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors())
```

```
lasso.wf <- workflow() %>%
  add_recipe(lasso.recipe) %>%
  add_model(lasso.model)


college.fold <- vfold_cv(college.train.tbl, v = 10)

penalty.grid <-
  grid_regular(penalty(range = c(-2, 2)), levels = 20)

tune.res <- tune_grid(
  lasso.wf,
  resamples = college.fold,
  grid = penalty.grid
)
autoplot(tune.res)
```

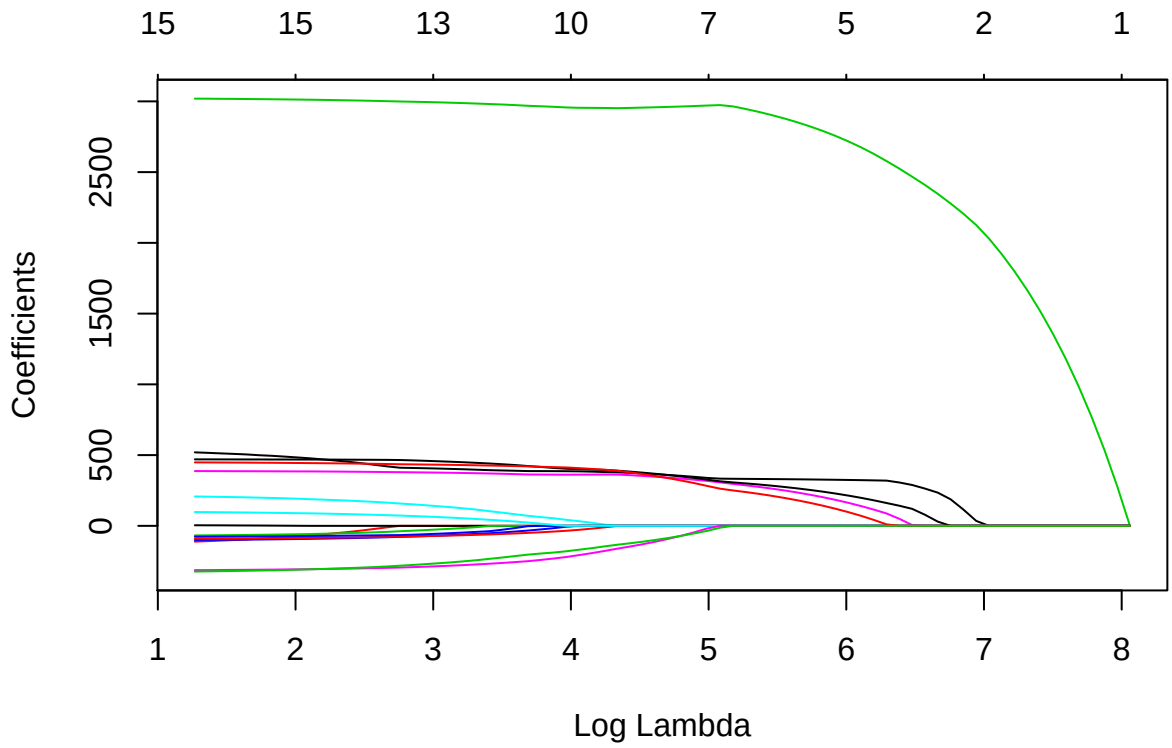



```
(best.penalty <- select_best(tune.res, metric = "rsq"))
```

```
## # A tibble: 1 x 2
##   penalty .config
##     <dbl> <fct>
## 1    37.9 Preprocessor1_Model18
```

```
lasso.final.wf <- finalize_workflow(lasso.wf, best.penalty)
lasso.final.fit <- fit(lasso.final.wf, data = college.train.tbl)
```

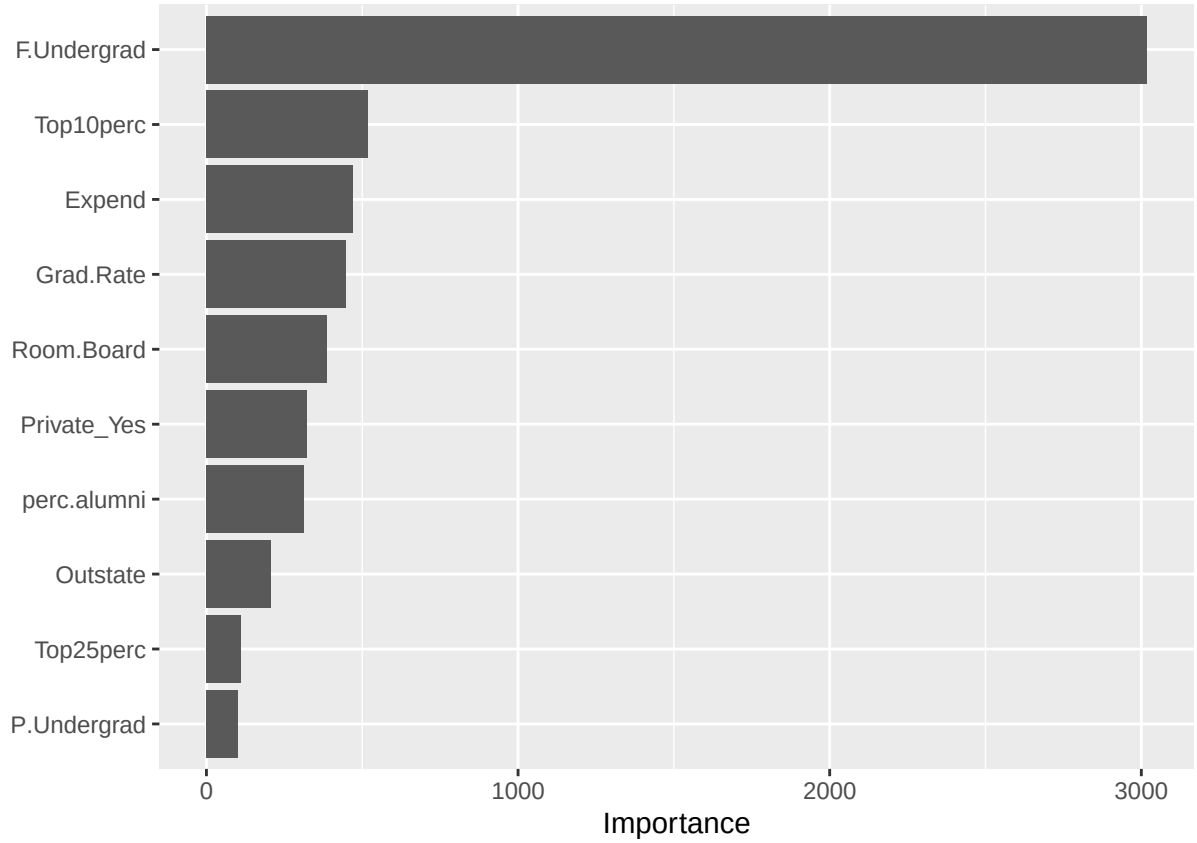
```
lasso.final.fit %>%
  extract_fit_engine() %>%
  plot(xvar = "lambda")
```




```
#R^2
augment(lasso.final.fit, new_data = college.test.tbl) %>%
  rsq(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.838
```

```
#Importance
extract_fit_parsnip(lasso.final.fit) %>%
  vip()
```




See note above, but when I run this now I get an optimal lambda of 37.9 (before I got an optimal lambda of 3.36 and also 61.6, which is why this is a little confusing), but both gave $R^2$ estimates of around 0.837-0.838, which is essentially identical to when we had the penalty set to 1 and very similar to the full linear model. The variable importance remains the same.

6. Select the top-5 most important variables from LASSO and create a linear model based on only these features. How does the $R^2$ on the testing dataset compare with the full linear model?

```
college2.recipe <-
  recipe(formula= Apps ~ F.Undergrad + Top10perc + Expend + Grad.Rate + Room.Board,
         data=college.train.tbl) %>%
  step_dummy(all_nominal_predictors())

lm2.model <- linear_reg()%>%
  set_mode("regression")%>%
  set_engine("lm")

lm2.wflow <- workflow()%>%
  add_recipe(college2.recipe) %>%
  add_model(lm2.model)

lm2.fit <- fit(lm2.wflow, college.train.tbl)

#R^2 of 5 variable lm
augment(lm2.fit, college.test.tbl)%>%
  rsq(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.835
```

```
#R^2 of full lm
augment(lm.fit, college.test.tbl)%>%
  rsq(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.839
```

When using the 5 most important variables in a linear model, we get an $R^2$ of 0.835, a decrease of only 0.004 when compared to the full linear model. This suggests that it was probably a good idea to cut these variables, given the tiny difference in variability explained by the model.

7. Construct a KNN model using the top-5 most important variables from LASSO. Based on the $R^2$ on the testing dataset, is this a better model than 6?

```
#WHAT IS K?
library(kknn)
knn.model <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>%
  set_mode("regression")

  knn.recipe <- recipe(Apps ~ F.Undergrad + Top10perc + Expend + Grad.Rate + Room.Board,
                       data=college.train.tbl)

  knn.wflow <- workflow() %>%
    add_recipe(knn.recipe) %>%
    add_model(knn.model)

  knn.fit <- fit(knn.wflow, college.train.tbl)
  set.seed(123456)


  college.folds <- vfold_cv(college.train.tbl, v = 10)
  neighbors.tbl <- grid_regular(neighbors(range = c(1, 51)), levels = 11)

  tune.results <- tune_grid(
  object = knn.wflow,
  resamples = college.folds,
  grid = neighbors.tbl
)

show_best(tune.results, metric = "rsq")
```

```
## # A tibble: 5 x 7
##   neighbors .metric .estimator  mean     n std_err .config
##       <int> <chr>   <chr>      <dbl> <int>   <dbl> <fct>
## 1        21 rsq     standard   0.794    10  0.0287 Preprocessor1_Model05
## 2        26 rsq     standard   0.793    10  0.0303 Preprocessor1_Model06
## 3         6 rsq     standard   0.791    10  0.0220 Preprocessor1_Model02
## 4        16 rsq     standard   0.791    10  0.0262 Preprocessor1_Model04
## 5        11 rsq     standard   0.790    10  0.0241 Preprocessor1_Model03
```

```
(best.neighbor <- select_best(tune.results, metric = "rsq"))
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <fct>
## 1        21 Preprocessor1_Model05
```

```
knn.final.wf <- finalize_workflow(knn.wflow, best.neighbor)
knn.final.fit <- fit(knn.final.wf, college.train.tbl)

#Final R^2
augment(knn.final.fit, college.test.tbl)%>%
  rsq(truth = Apps, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.803
```

When we cross-validate to optimize k and then run a knn model using that optimal k value (turned out to be 21), we see an $R^2$ of 0.803 on the testing dataset. This is worse than our model in #6, when we saw an $R^2$ of 0.835.

8. What will be the topic for your Challenge 2 spotlight? Write a paragraph explaining the topic, dataset or question that you will be addressing.

I am planning to look at convolutional neural networks. I found a couple of examples online that work through how to train CNNs on simple datasets, so I'm planning on doing that. My plan will be to use their dataset (most likely the one described here: https://tensorflow.rstudio.com/tutorials/advanced/images/cnn/), which involves trying to classify an animal based on an image.

If I can figure it out, I think it'd be good to apply that to the mnist dataset, but if I can't do that I still think having the animal exmaple will work pretty well.