Solution_HW1

MSBA7002: Business Statistics

Abstract

The homework solutions contain a continued effort from the MSBA7002 TAs and instructors[1]. You should compare the solutions with your own homework submission to see how you can better improve your analytical skills. Please **do not redistribute the document or put it online** to hurt other students' learning experience.

Contents

[1] We would like to thank Jianlong Shao for providing the first version of the solution and for his great contributions to the class

1 Manager Rating

Sol:

$$\begin{cases} \beta_0 & = \alpha_0 - \alpha_1 \\ \beta_1 & = 2\alpha_1 \\ \beta_2 & = \alpha_2 - \alpha_3 \\ \beta_3 & = 2\alpha_3 \end{cases}$$

$$\begin{cases} \alpha_0 & = \beta_0 + \frac{1}{2}\beta_1 \\ \alpha_1 & = \frac{1}{2}\beta_1 \\ \alpha_2 & = \beta_2 + \frac{1}{2}\beta_3 \\ \alpha_3 & = \frac{1}{2}\beta_3 \end{cases}$$

2 Production Time Run

ProdTime.dat contains information about 20 production runs supervised by each of three managers. Each observation gives the time (in minutes) to complete the task, Time for Run, as well as the number of units produced, Run Size, and the manager involved, Manager.

```
df.ptr <- read.csv('ProdTime.dat')
str(df.ptr)
```

```
## 'data.frame':    61 obs. of  3 variables:
##  $ Time.for.Run: int  252 215 238 261 297 236 282 264 254 223 ...
##  $ Manager     : chr  "a" "a" "a" "a" ...
##  $ Run.Size    : int  204 103 143 210 334 102 261 118 198 87 ...
```

```
# delete missing value
df.ptr <- na.omit(df.ptr)
```

Open Answer, Either Manager B or C perform the best is correct.

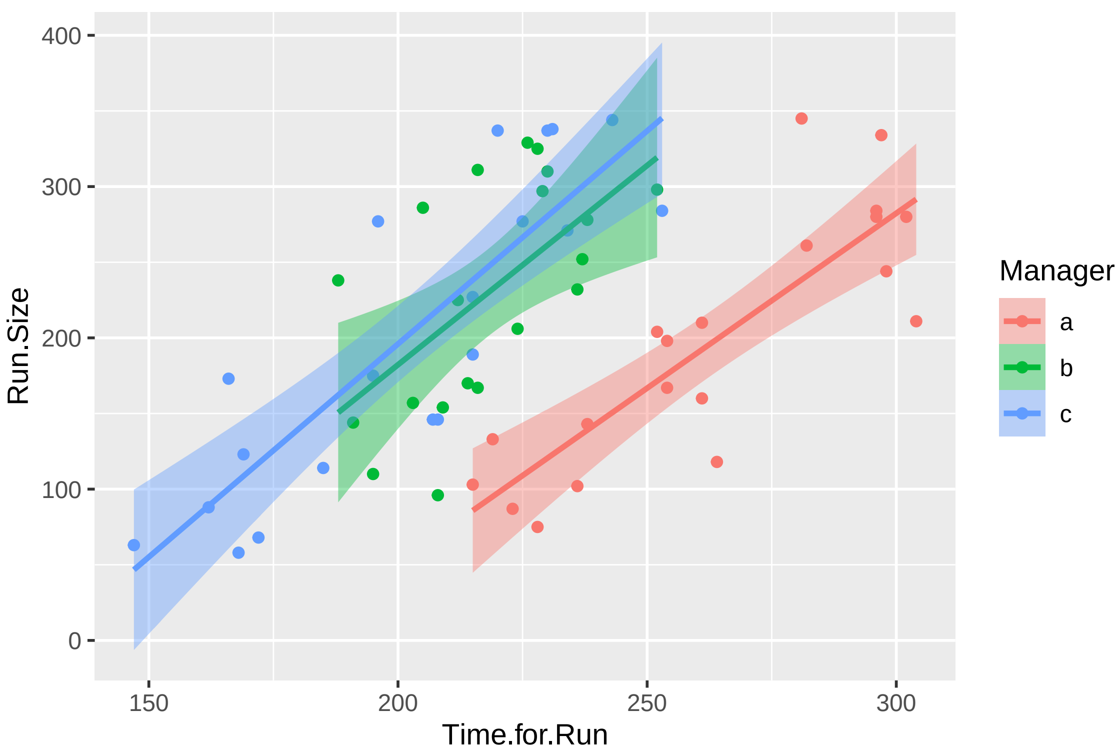
We could compare the performance of manager from either aspect below. But the second one seems better.

(i) Compare the performance of manager from `Run.size ~ Time.for.run`

```
fit.ptr1 <- lm(Run.Size ~ Time.for.Run + Manager, data = df.ptr)
summary(fit.ptr1)
```

```
##
## Call:
## lm(formula = Run.Size ~ Time.for.Run + Manager, data = df.ptr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -107.867  -43.213   -6.687   37.298  101.794
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -480.9069    70.8539  -6.787 7.77e-09 ***
## Time.for.Run   2.5769     0.2655   9.705 1.34e-13 ***
## Managerb     148.7765    20.6882   7.191 1.67e-09 ***
## Managerc     161.9917    23.3730   6.931 4.50e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 53.29 on 56 degrees of freedom
## Multiple R-squared:  0.6375, Adjusted R-squared:  0.618
## F-statistic: 32.82 on 3 and 56 DF,  p-value: 2.238e-12
```

```
ggplot(data = df.ptr, aes(x = Time.for.Run, y = Run.Size, color = Manager, fill = Manager)) +
  geom_point() +
  geom_smooth(method = "lm", formula = "y~x")
```

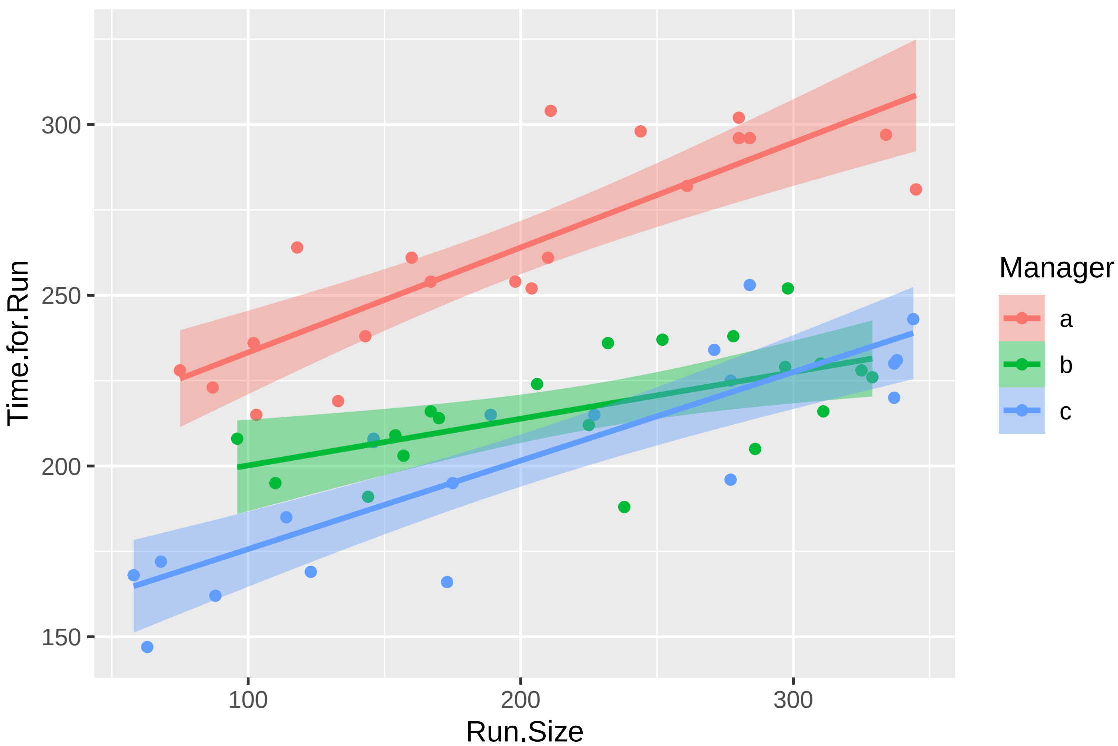


(ii) Compare the performance of manager from `Time.for.run` ~ `Run.size`. This model seems making more sense as we hope to know given a task, i.e. the number of units produced, how long does it take for the manager to complete the task.

```
fit.ptr2 <- lm(Time.for.Run ~ Run.Size + Manager, data = df.ptr)
summary(fit.ptr2)
```

```
##
## Call:
## lm(formula = Time.for.Run ~ Run.Size + Manager, data = df.ptr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.979 -12.467   0.765  12.041  37.531
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 215.11848    6.14820  34.989  < 2e-16 ***
## Run.Size      0.24337    0.02508   9.705 1.34e-13 ***
## Managerb    -53.06082    5.24159 -10.123 2.93e-14 ***
## Managerc    -62.16817    5.18003 -12.002  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.38 on 56 degrees of freedom
## Multiple R-squared:  0.8131, Adjusted R-squared:  0.8031
## F-statistic: 81.23 on 3 and 56 DF,  p-value: < 2.2e-16
```

```
ggplot(data = df.ptr, aes(x = Run.Size, y = Time.for.Run, color = Manager, fill = Manager)) +
  geom_point() +
  geom_smooth(method = "lm", formula = "y~x")
```



From the above plot, the slope looks different for Manager b and c. So we can also consider adding the interaction term.

```
fit.ptr3 <- lm(Time.for.Run ~ Run.Size * Manager, data = df.ptr)
summary(fit.ptr3)
```

```
##
## Call:
## lm(formula = Time.for.Run ~ Run.Size * Manager, data = df.ptr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.047  -8.698   1.274   8.608  36.633
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       202.53415    9.26933  21.850  < 2e-16 ***
## Run.Size            0.30727    0.04358   7.051 3.41e-09 ***
## Managerb          -16.04029   14.82719  -1.082   0.2841
## Managerc          -52.78645   12.25963  -4.306 7.06e-05 ***
## Run.Size:Managerb  -0.17049    0.06492  -2.626   0.0112 *
## Run.Size:Managerc  -0.04802    0.05639  -0.852   0.3982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.66 on 54 degrees of freedom
## Multiple R-squared:  0.8353, Adjusted R-squared:   0.82
## F-statistic: 54.76 on 5 and 54 DF,  p-value: < 2.2e-16
```

With the above model, we conclude that when `Run.Size` is small, Manager c is better than both Manager a and b (since $0 > -16.04 > -52.79$). As `Run.Size` becomes larger, the effect of the dummy variable `Managerb` decreases with slope -0.17. However, since the effect of the dummy variable `Managerc` barely change with `Run.Size` (the interaction of `Run.Size` and `Managerc` is not significant). Therefore, when `Run.Size` is large, the advantage of Manager c over Manager b becomes less significant, although they both still perform better than Manager a.

3 Auto Data from ISLR

3.1

Explore the data, with particular focus on pairwise plots and summary statistics. Briefly summarize your findings and any peculiarities in the data.

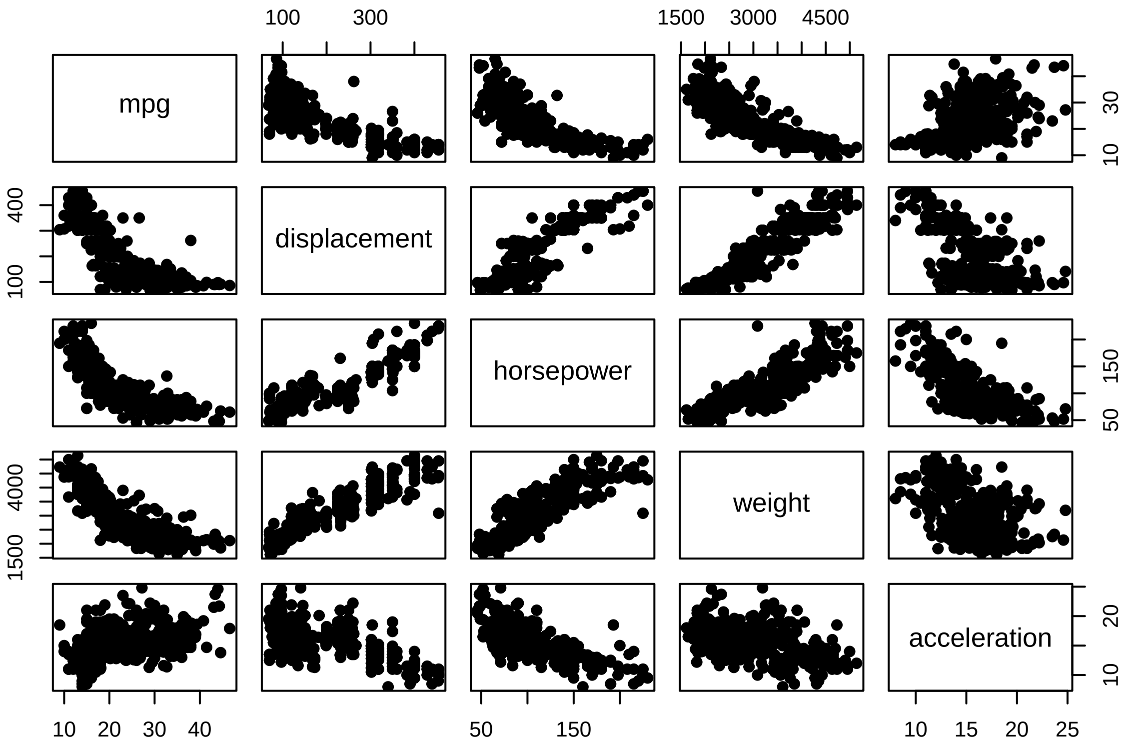
```
auto_data<-ISLR::Auto
str(auto_data)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 
```

```
summary(auto_data)
```

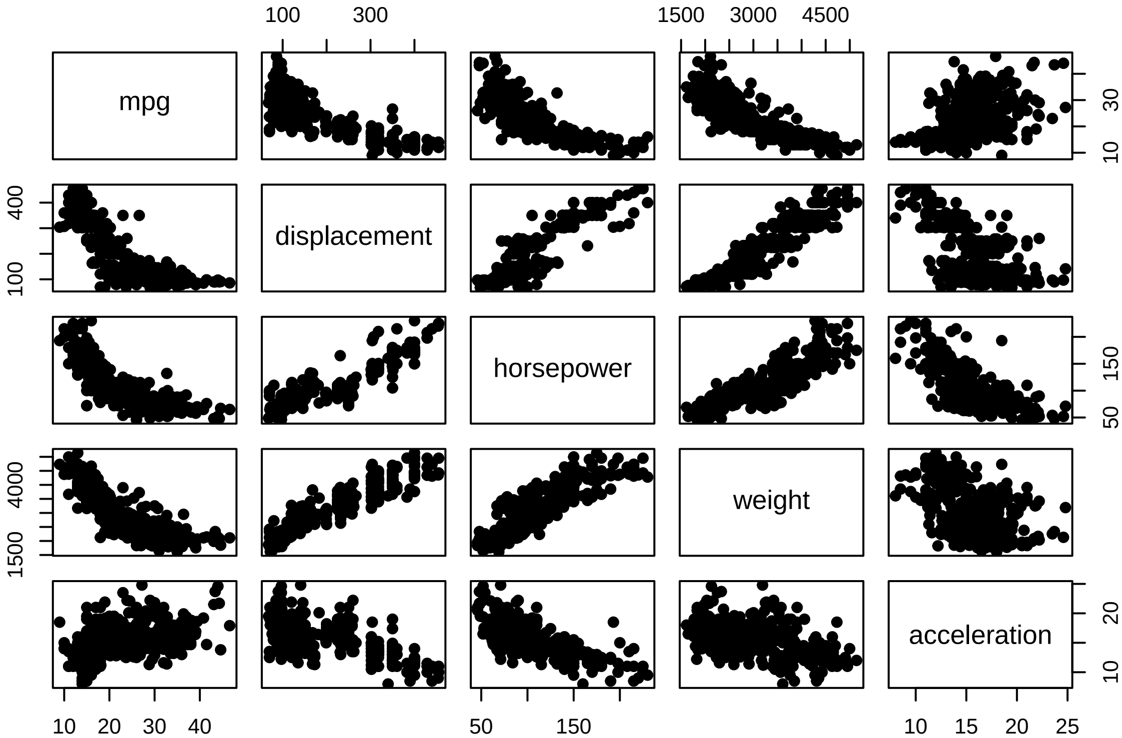
```
##       mpg          cylinders      displacement     horsepower        weight    
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613  
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225  
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804  
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978  
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615  
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140  
##                                                                               
##   acceleration        year           origin                      name    
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5  
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5  
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5  
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4  
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4  
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4  
##                                                  (Other)           :365  
```

```
pairs(auto_data[,-c(2,7,8,9)], pch = 19)
```



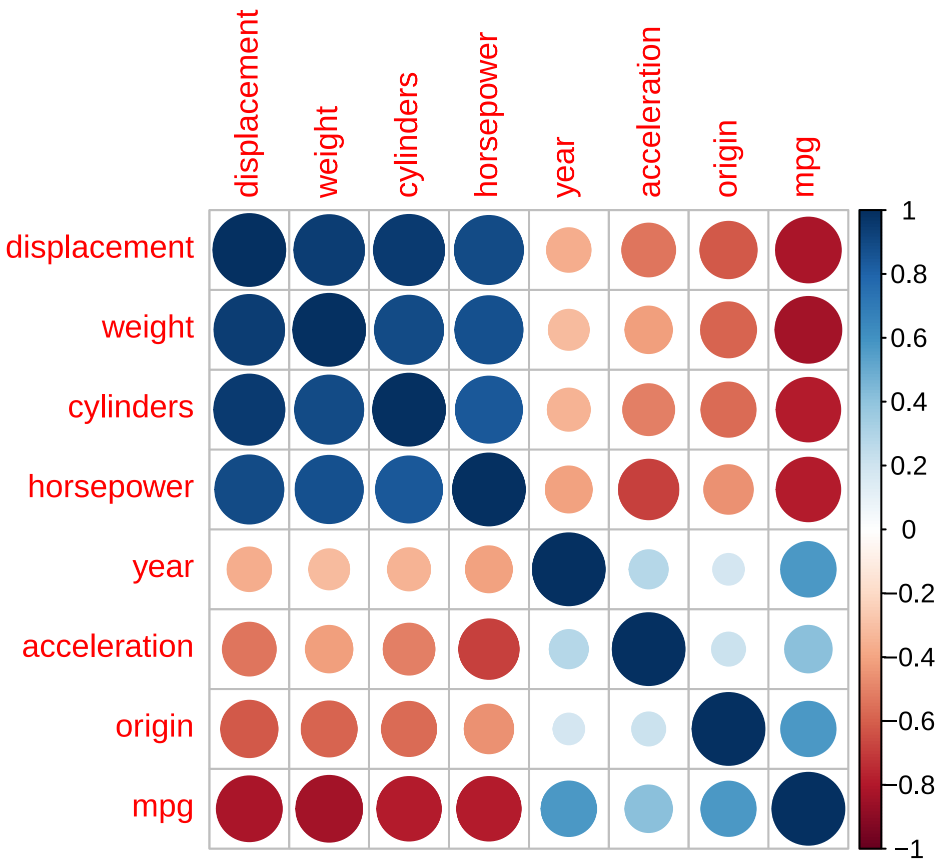
There may be potential outliers, let us remove them.

```
# remove the potential outlier
auto_data = auto_data[!((auto_data$horsepower>200) & (auto_data$weight<3500)),]
auto_data = auto_data[!((auto_data$mpg>30) & (auto_data$displacement>250)),]
pairs(auto_data[,-c(2,7,8,9)], pch = 19)
```



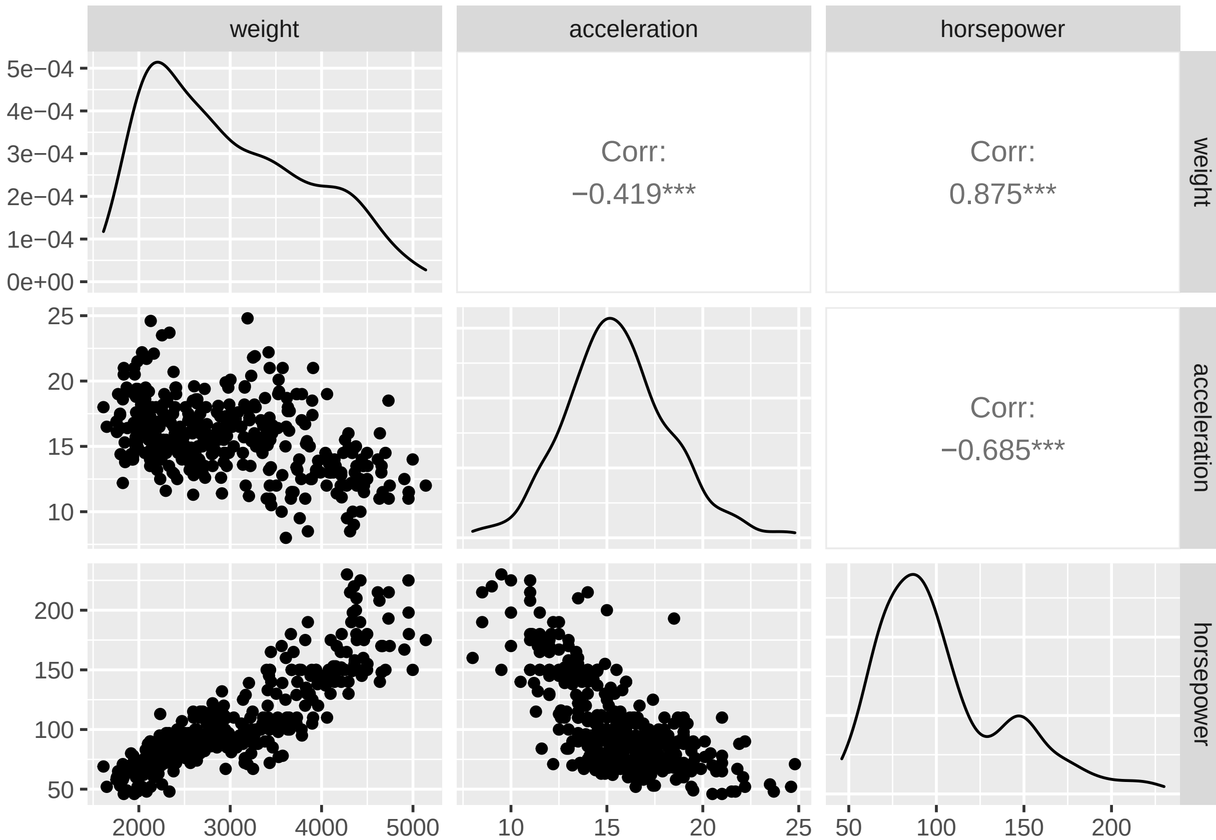
Horsepower, weight and acceleration have strong correlation. Since fewer accelerate time and larger vehicle weight will need more horsepower to run, it might be horsepower ~ weight/acceleration.

```
auto_cor <- cor(auto_data[sapply(auto_data, is.numeric)])
corrplot(auto_cor, order = "FPC")
```



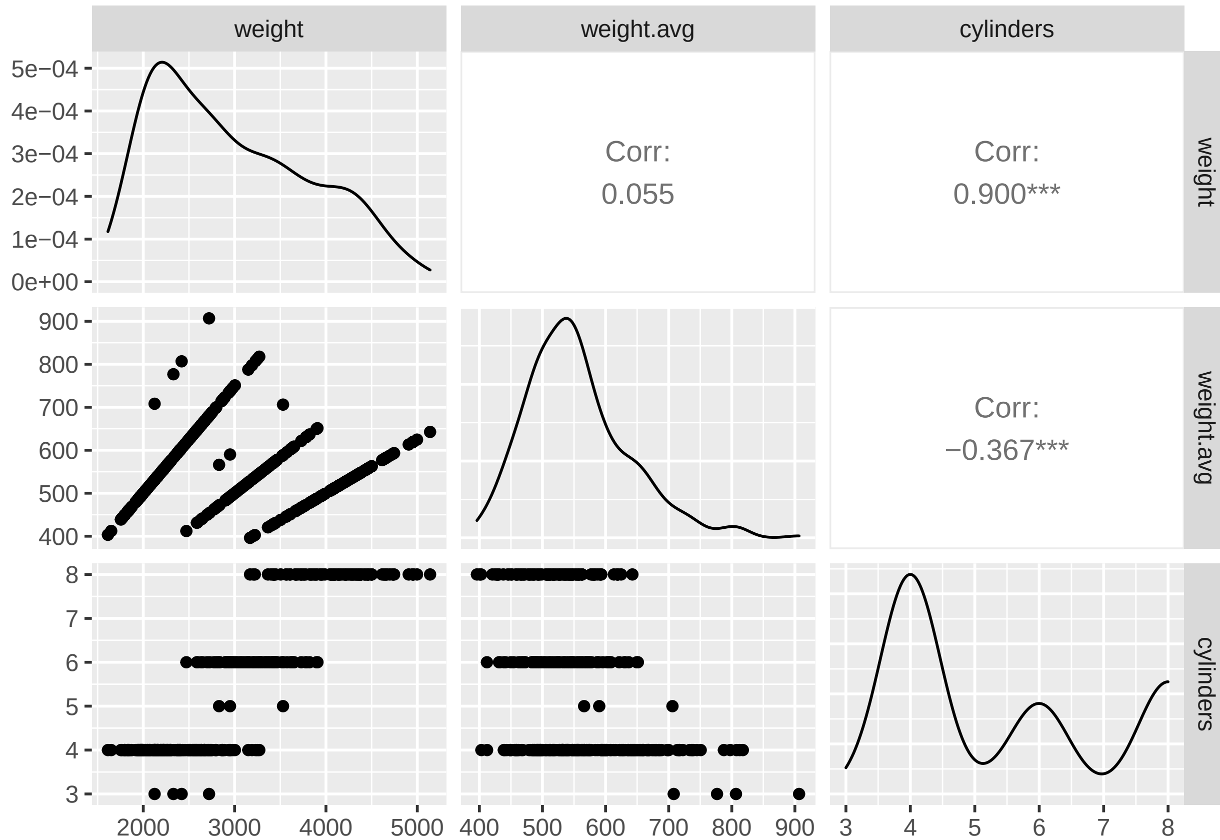
From the plot below, we can see this presumption holds true. Hence, further adjustment is needed if a linear regression model contains all of them.

```
auto_data %>%
  select(weight, acceleration, horsepower) %>%
  ggpairs()
```



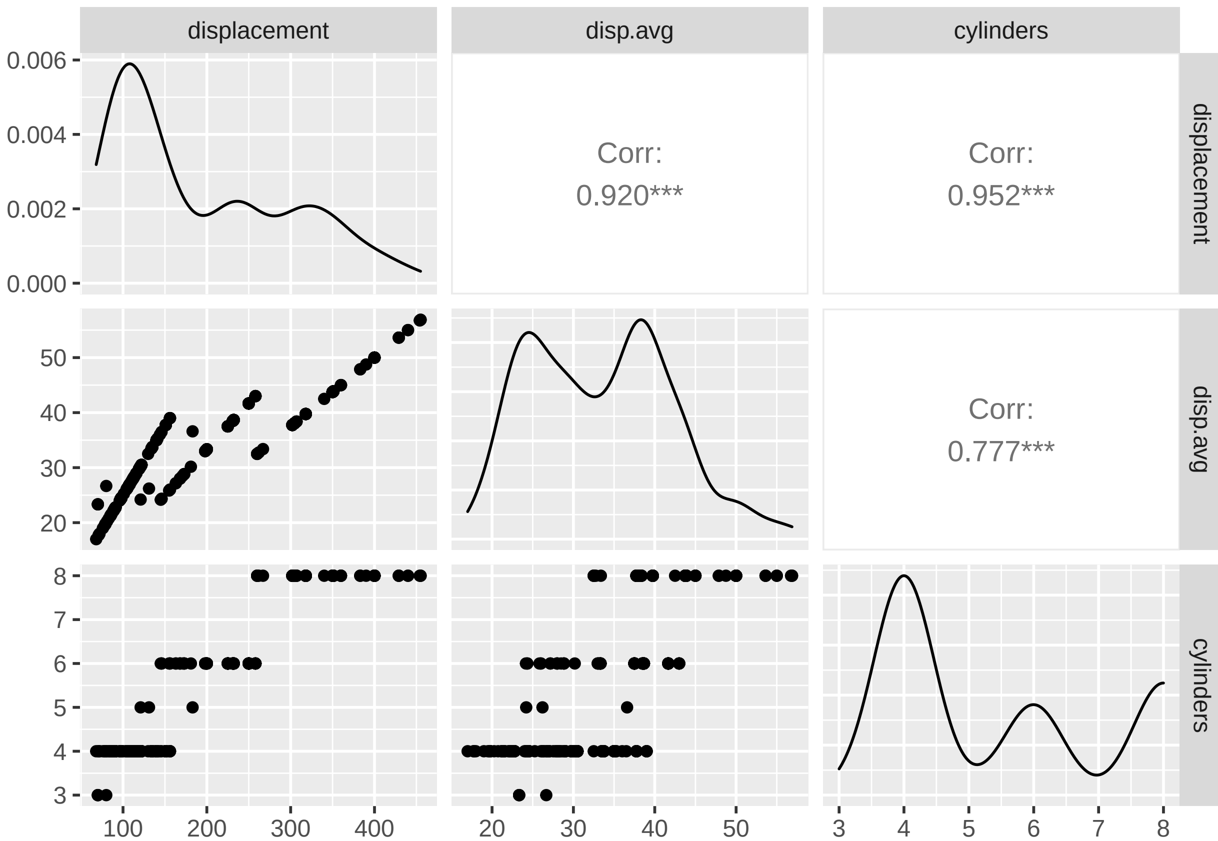
Weight has positive correlation with cylinders and displacement. Obviously, a vehicle with more cylinders weighs heavier. Hence, it makes more sense to use weight.avg to represent the vehicle weight per cylinder. To justify this transformation, the corrplot of weight, cylinders and weight.avg is shown below. The correlation still exists, but not as significant as before.

```
auto.tf <- data.frame(auto_data)
auto.tf['weight.avg'] <- with(auto.tf, weight/cylinders)
auto.tf %>%
  select(weight, weight.avg, cylinders) %>%
  ggpairs()
```



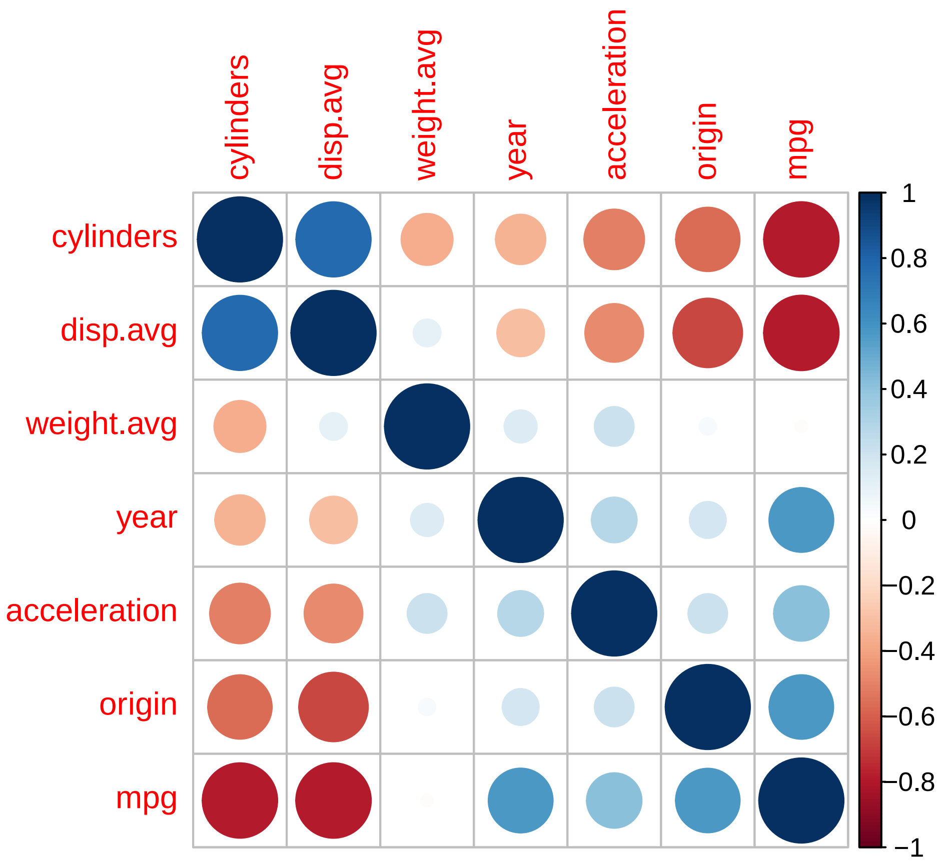
Displacement is correlatetd to cylinders postively. displacement represents the total swept volume inside cylinders. Hence, it makes more sense to use "disp.avg" to represent the swept volumn per cylinder. From the plot below, we can see the correlationship still exists, hence we may want to condition on cylinders to see the categorical effect of the number of the cylinders.

```
auto.tf['disp.avg'] <- with(auto_data, displacement/cylinders)
auto.tf %>%
  select(displacement, disp.avg, cylinders) %>%
  ggpairs()
```



The corrplot with transformed variables is presented below

```
auto.tf %>%
  select(mpg, cylinders, disp.avg, weight.avg, acceleration,
year, origin) %>%
  cor() %>%
  corrplot(order = "FPC")
```



Except for mpg, the coliearity among all other variables seems improved. year, origin and cylinders are going to be conditioned on later, because they are discrete values and have strong correlationship with other factors.

3.2

What effect does time have on MPG?

3.2.1

Start with a simple regression of ‘mpg‘ vs. ‘year‘ and report R’s ‘summary‘ output. Is year a significant variable at the .05 level? State what effect year has on mpg, if any, according to this model.

```
fit_3.2.1 <- lm(mpg~year, data = auto_data)
summary(fit_3.2.1)
```

```
##
## Call:
## lm(formula = mpg ~ year, data = auto_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9550  -5.4425  -0.3981   4.9504  18.2645
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -69.21835    6.69593  -10.34   <2e-16 ***
## year          1.21942    0.08803   13.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.368 on 388 degrees of freedom
## Multiple R-squared:  0.3309, Adjusted R-squared:  0.3292
## F-statistic: 191.9 on 1 and 388 DF,  p-value: < 2.2e-16
```

That the coefficient of year is 1.22, **significant** at 0.05 level. This coefficient shows that year has a positive marginal effect on mpg. R2 of regressing mpg on year is 33.1%.

3.2.2

Add horsepower on top of the variable year. Is year still a significant variable at the .05 level? Give a precise interpretation of the year effect found here.

```
fit_3.2.2 <- lm(mpg ~ year + horsepower, data = auto_data)
summary(fit_3.2.2)
```

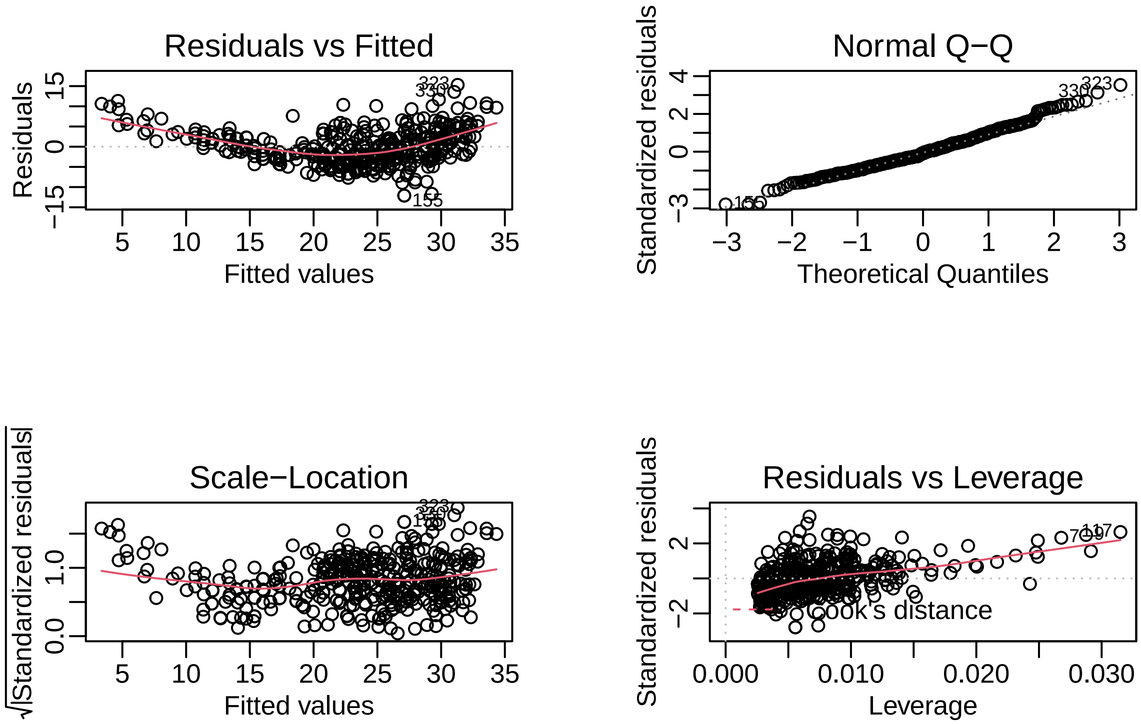
```
##
## Call:
## lm(formula = mpg ~ year + horsepower, data = auto_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.1075  -3.0076  -0.3572   2.5787  15.3031
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -12.038503   5.314352  -2.265    0.024 *
## year          0.650458   0.065871   9.875   <2e-16 ***
## horsepower   -0.133865   0.006343 -21.103   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.348 on 387 degrees of freedom
## Multiple R-squared:  0.6889, Adjusted R-squared:  0.6873
## F-statistic: 428.5 on 2 and 387 DF,  p-value: < 2.2e-16
```

The coefficient of year in this model is 0.65, **significant** at 0.05 level. The coefficient of year show that year has a positive **partial effect** on mpg, given a constant horsepower.

Interpretation: Given a constant horsepower, the expectation of mpg will increase 0.65 unit if the year increases one unit,

Diagnose

```
par(mfrow = c(2,2),mgp = c(1.5,0.5,0))
plot(fit_3.2.2)
```



3.2.3

The two 95% CI's for the coefficient of year differ among i) and ii). How would you explain the difference to a non-statistician?

```
CI_3.2.3 <- data.frame(a = c(confint.lm(fit_3.2.1)[2,1],confint.lm(fit_3.2.2)[2,1]),
                       b = c(confint.lm(fit_3.2.1)[2,2],confint.lm(fit_3.2.2)[2,2]),
                       c = c('Marginal effect','Partial effect'))
colnames(CI_3.2.3) <- c('2.5% CI','97.5% CI','Reason for difference')
rownames(CI_3.2.3) <- c('Model without horsepower','Model with horsepower')
CI_3.2.3
```

```
##                            2.5% CI  97.5% CI Reason for difference
## Model without horsepower 1.0463562 1.3924909       Marginal effect
## Model with horsepower    0.5209479 0.7799677        Partial effect
```

The coefficients of year in two model should be different, since the coefficient of model one measures marginal effect of year on mpg and the coefficient of model measures partial effect.

3.2.4

Do a model with interaction by fitting ‘lm(mpg year * horsepower)‘. Is the interaction effect significant at .05 level? Explain the year effect (if any).

```
fit_3.2.4 <- lm(mpg ~ year * horsepower, data = auto_data)
summary(fit_3.2.4)
```

```
##
## Call:
## lm(formula = mpg ~ year * horsepower, data = auto_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.2856  -2.4253  -0.4458   2.3781  14.4431
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.238e+02  1.227e+01 -10.090   <2e-16 ***
## year             2.155e+00  1.632e-01  13.199   <2e-16 ***
## horsepower       1.022e+00  1.171e-01   8.726   <2e-16 ***
## year:horsepower -1.565e-02  1.584e-03  -9.881   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.889 on 386 degrees of freedom
## Multiple R-squared:  0.7517, Adjusted R-squared:  0.7498
## F-statistic: 389.5 on 3 and 386 DF,  p-value: < 2.2e-16
```

The coefficient of interaction term is $-1.565 * 10^{-2}$, significant at 0.05 level. The marginal effect of year on mpg is $2.155 - 1.565 * 10^{-2} * horsepower$.

3.3

Note that the same variable can play different roles! Take a quick look at the variable ‘cylinders‘, try to use this variable in the following analyses wisely. We all agree that larger number of cylinder will lower mpg. However, we can interpret ‘cylinders‘ as either a continuous (numeric) variable or a categorical variable.

3.3.1

i. Fit a model, that treats ‘cylinders‘ as a continuous/numeric variable: ‘lm(mpg horsepower + cylinders, ISLR::Auto)‘. Is ‘cylinders‘ significant at the 0.01 level? What effect does ‘cylinders‘ play in this model?

```
fit_3.3.1 <- lm(mpg ~ horsepower + cylinders, data = ISLR::Auto)
summary(fit_3.3.1)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + cylinders, data = ISLR::Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.4378  -3.2422  -0.3721   2.3532  16.9289
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.94842    0.77880  55.147  < 2e-16 ***
## horsepower  -0.08612    0.01119  -7.693 1.19e-13 ***
## cylinders   -1.91982    0.25261  -7.600 2.24e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.584 on 389 degrees of freedom
## Multiple R-squared:  0.6569, Adjusted R-squared:  0.6551
## F-statistic: 372.4 on 2 and 389 DF,  p-value: < 2.2e-16
```

Cylinders is significant at the 0.01 level. Given horsepower is the same, `cylinders` plays a negative effect on `mpg`.

3.3.2

Fit a model that treats ‘cylinders‘ as a categorical/factor variable: ‘lm(mpg horsepower + as.factor(cylinders), ISLR::Auto)‘. Is ‘cylinders‘ significant at the .01 level? What is the effect of ‘cylinders‘ in this model?

```r
fit_3.3.2 <- lm(mpg ~ horsepower + factor(cylinders), ISLR::Auto)
summary(fit_3.3.2)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + factor(cylinders), data = ISLR::Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5917 -2.7067 -0.6102  1.9001 16.3258
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        30.77614    2.41283  12.755  < 2e-16 ***
## horsepower         -0.10303    0.01133  -9.095  < 2e-16 ***
## factor(cylinders)4  6.57344    2.16921   3.030  0.00261 **
## factor(cylinders)5  5.07367    3.26661   1.553  0.12120
## factor(cylinders)6 -0.34406    2.18580  -0.157  0.87501
## factor(cylinders)8  0.49738    2.27639   0.218  0.82716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.27 on 386 degrees of freedom
## Multiple R-squared:  0.7046, Adjusted R-squared:  0.7008
## F-statistic: 184.1 on 5 and 386 DF,  p-value: < 2.2e-16
```

The effect of cylinders in this model: cylinders4 has a significant estimator coeffiecient (6.573) at 0.01 level, which means that cylinders 4 has a positive effect on mpg compared with situation cylinders is equal to 3. However, cylinders5, cylinders6, cylinders8 don’t own a significant estimator.

3.3.3

What are the fundamental differences between treating cylinders as a numeric or a factor? Use ‘anova(fit1, fit2)‘ to help gauge the effect. Explain their difference.

```
fit_3.3.3 <- lm(mpg ~ horsepower, ISLR::Auto)
```

```
anova(fit_3.3.3,fit_3.3.1)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ horsepower
## Model 2: mpg ~ horsepower + cylinders
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    390 9385.9
## 2    389 8172.5  1    1213.4 57.758 2.239e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(fit_3.3.3,fit_3.3.2)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ horsepower
## Model 2: mpg ~ horsepower + factor(cylinders)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    390 9385.9
## 2    386 7036.7  4    2349.2 32.217 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we can see adding cylinders as a categorical variable gives extra 3 degrees of freedom, which means the model treat as.factor(cylinders) as 4 different variables (cylinders = 4, 5, 6, 8). From the result, we can see this improves the model fit significantly. Hence, we can interpret the second model lm.cl.cat includes 4 variables representing 5 types of cylinders besides year. Each kind of cylinder will have different effect on the fuel economy performance (mpg). The model fits better since the number of cylinders is not related to mpg monotonely.

4 Crime Data

We use the crime data to study the prediction of the number of violent crimes (per population). We are going to mainly look at Florida and California. Note the following code:

```
crime <- read.csv("CrimeData_sub.csv", stringsAsFactors = F, na.strings = c("?"))
crime <- na.omit(crime)
```

Our goal is to find the factors/variables which relate to violent crime. This variable is included in crime as crime$violentcrimes.perpop.

4.1

Divide your data into 80% training and 20% testing. Run the ordinary least square regression with all the variables and with the training data. Get RMSE and R2 for both the training and testing data and see if there is a difference.

```
set.seed(20211001)
ind <- sample(2, nrow(crime), replace = T, prob = c(0.8, 0.2))
crime.train <- crime[ind==1,]
crime.test <- crime[ind==2,]

# this gives each data 80% probability to be in the training
# but we do not exactly have 80% of data in the training
# you can use a more exact way of selection 80% data as training, for example,
# train_ind <- sample(nrow(crime), size = round(nrow(crime)*0.8))
# crime.train <- crime[train_ind, ]
# crime.test <- crime[-train_ind, ]

# Dump everything in the model
fit.all <- lm(violentcrimes.perpop~., data=crime.train)
# in-sample performance
predictions <- predict(fit.all, crime.train)
data.frame(
  RMSE = RMSE(predictions, crime.train$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.train$violentcrimes.perpop)
)
```

```
##       RMSE   Rsquare
## 1 282.1736 0.8263237
```

```
# out-of-sample performance
predictions <- predict(fit.all, crime.test)
data.frame(
  RMSE = RMSE(predictions, crime.test$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.test$violentcrimes.perpop)
)
```

```
##     RMSE   Rsquare
## 1 373.37 0.6362164
```

The testing performance is much worse than the training performance, which indicates possible overfitting.

4.2

Use LASSO to choose a reasonable, small model. Fit an OLS model with the variables obtained. The final model should only include variables with p-values < 0.05. Note: you may choose to use lambda 1se or lambda min to answer the following questions where apply.

4.2.1

What is the model reported by LASSO? Use 5-fold cross-validation to select the tuning parameter.

Let us first try LASSO with a set of λ's and look at the Lasso path.

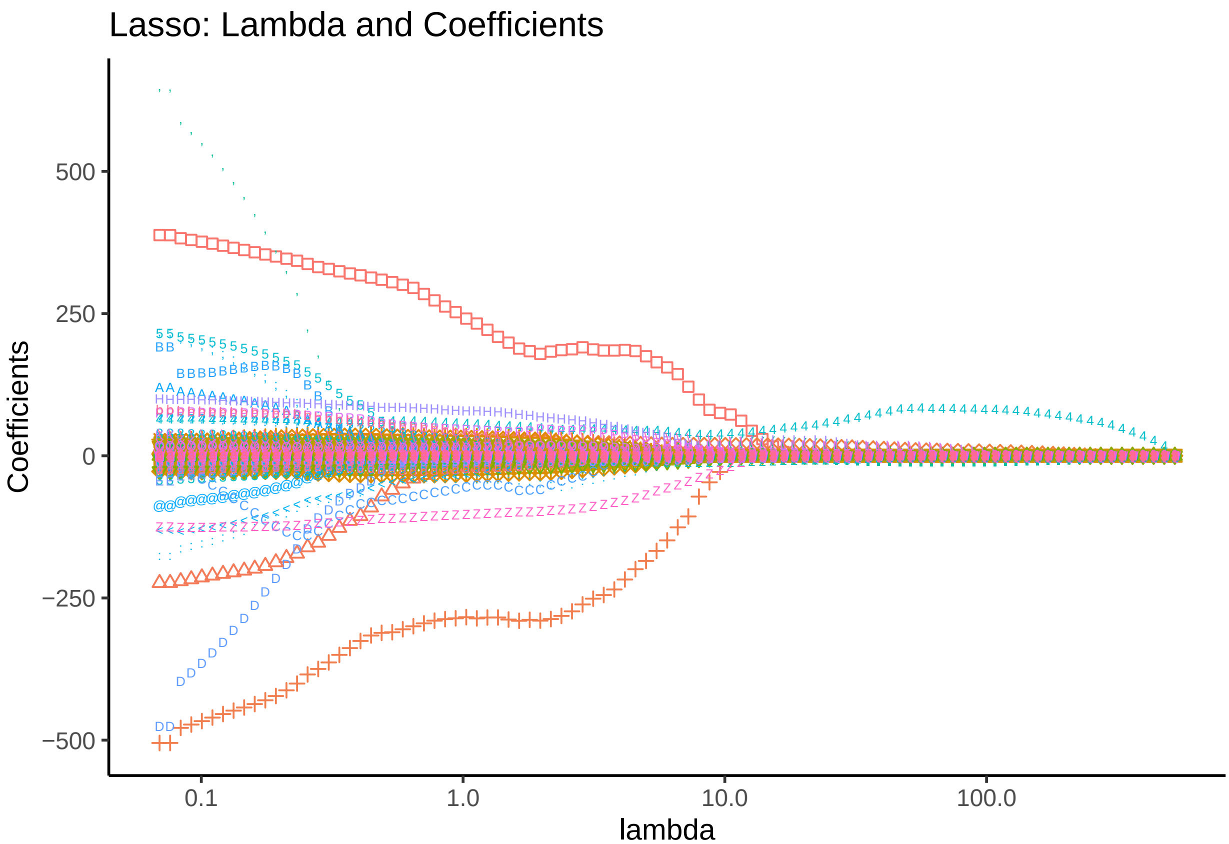
```
# create a matrix contain the whole predictors
x.matrix <- model.matrix(violentcrimes.perpop~., data=crime.train)[, -1]
violentcrimes.perpop <- crime.train[,103]

fit.lasso.path <- glmnet(x.matrix, violentcrimes.perpop, alpha=1)
```

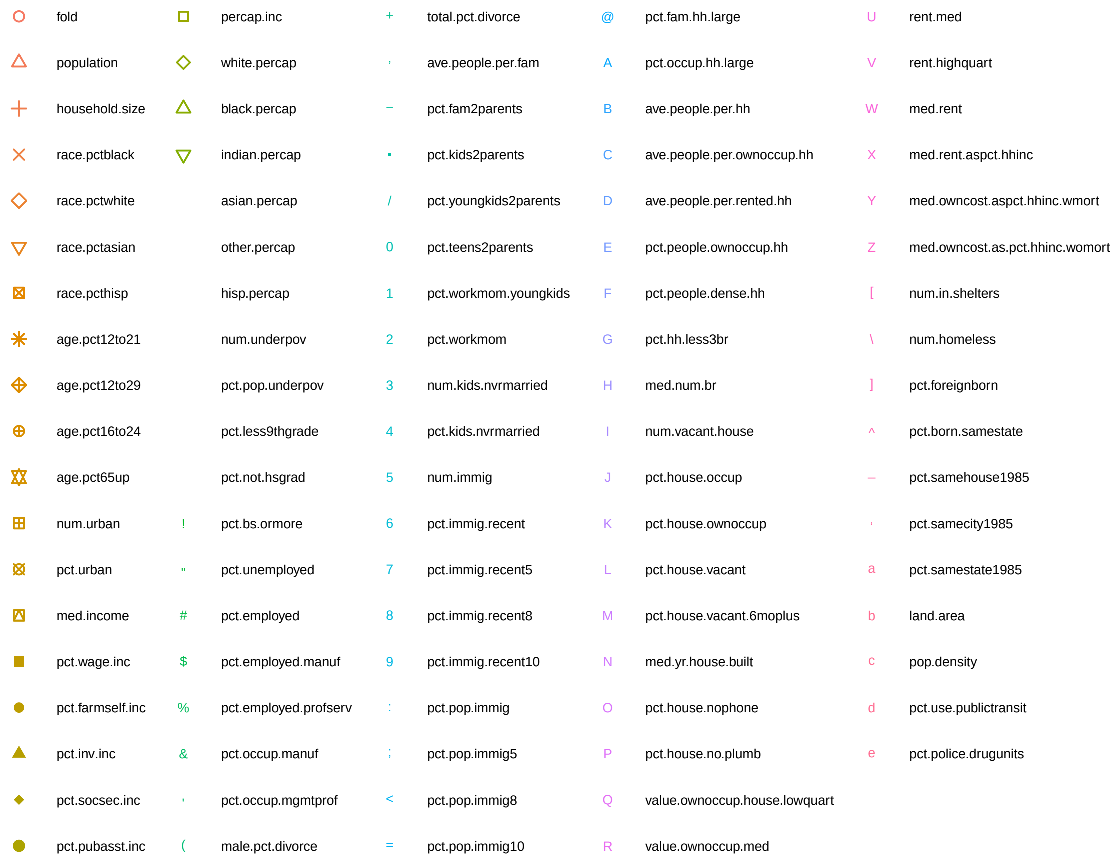
```
df.lasso <- fit.lasso.path$beta %>% as.matrix()
colnames(df.lasso) <- fit.lasso.path$lambda
df.lasso <- df.lasso %>% reshape2::melt()
colnames(df.lasso) <- c("Variables","lambda","Coefficients")

p.lasso <- df.lasso %>%
  ggplot(aes(x = lambda, y = Coefficients, color = Variables, shape = Variables)) +
  geom_point() + scale_x_log10() +
  scale_shape_manual(values=seq(0,dim(x.matrix)[2])) +
  ggtitle("Lasso: Lambda and Coefficients") + theme_classic() +
  theme(legend.position="bottom") + theme(legend.title = element_blank()) +
  theme(legend.text=element_text(size=5))

p.lasso + theme(legend.position = "none")
```




```
as_ggplot(get_legend(p.lasso))
```



We then use 5-fold CV to select the best λ. Note that given different seed, model selected will be different as CV folds will be defined differently.

```
set.seed(20211001)
fit.lasso <- cv.glmnet(x.matrix,violentcrimes.perpop,alpha = 1,nfolds = 5)

coef.min <- coef(fit.lasso, s="lambda.min")
coef.min  <- coef.min[which(coef.min!=0),]

var.min <- rownames(as.matrix(coef.min))
lm.input <- as.formula(paste("violentcrimes.perpop", "~", paste(var.min[-1], collapse = "+")))
lm.input
```

```
## violentcrimes.perpop ~ race.pctblack + pct.inv.inc + male.pct.divorce +
##     pct.kids2parents + pct.youngkids2parents + pct.workmom +
##     pct.kids.nvrmarried + pct.english.only + num.vacant.house +
##     pct.house.vacant + med.yr.house.built + pct.house.nophone +
##     pct.house.no.plumb + num.homeless + pct.samecity1985 + pop.density
```

4.2.2

What is the model after refitting OLS with the selected variables? What are RMSE and R2 for the training and testing data? Compare them with results in Q4.2.

```
fit <- lm(lm.input, data = data.frame(violentcrimes.perpop =
                crime.train$violentcrimes.perpop, x.matrix))
summary(fit)
```

```
##
## Call:
## lm(formula = lm.input, data = data.frame(violentcrimes.perpop = crime.train$violentcrimes.perpop,
##     x.matrix))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1101.16  -200.69   -42.98   158.08  1870.28
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            5.427e+03  7.376e+03   0.736  0.46250
## race.pctblack          2.618e+01  3.780e+00   6.926 2.83e-11 ***
## pct.inv.inc           -1.498e+00  3.198e+00  -0.468  0.63983
## male.pct.divorce       3.694e+01  1.390e+01   2.658  0.00829 **
## pct.kids2parents      -6.898e+00  6.915e+00  -0.997  0.31937
## pct.youngkids2parents -2.801e+00  6.063e+00  -0.462  0.64445
## pct.workmom           -1.132e+01  4.116e+00  -2.750  0.00634 **
## pct.kids.nvrmarried    2.729e+01  1.986e+01   1.374  0.17043
## pct.english.only      -5.550e+00  2.694e+00  -2.060  0.04029 *
## num.vacant.house       5.190e+01  2.568e+01   2.021  0.04424 *
## pct.house.vacant       1.728e+01  1.317e+01   1.312  0.19072
## med.yr.house.built    -2.016e+00  3.754e+00  -0.537  0.59169
## pct.house.nophone      1.652e+01  1.307e+01   1.264  0.20731
## pct.house.no.plumb    -1.677e+01  6.994e+01  -0.240  0.81062
## num.homeless           1.928e+01  1.480e+01   1.302  0.19379
## pct.samecity1985       2.675e+00  2.586e+00   1.034  0.30187
## pop.density            8.350e-03  9.768e-03   0.855  0.39336
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 359.7 on 288 degrees of freedom
## Multiple R-squared:  0.7335, Adjusted R-squared:  0.7187
## F-statistic: 49.55 on 16 and 288 DF,  p-value: < 2.2e-16
```

Let us apply the model for training and testing performance evaluation using RMSE and R2.

```
x.test <- model.matrix(violentcrimes.perpop~., data=crime.test)[, -1]
# in-sample performance
predictions <- predict(fit, data.frame(violentcrimes.perpop =
                crime.train$violentcrimes.perpop, x.matrix))
data.frame(
  RMSE = RMSE(predictions, crime.train$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.train$violentcrimes.perpop)
)
```

```
##       RMSE   Rsquare
## 1 349.5212 0.7335261
```

```
# out-of-sample performance
predictions <- predict(fit, data.frame(violentcrimes.perpop =
                crime.test$violentcrimes.perpop, x.test))
data.frame(
  RMSE = RMSE(predictions, crime.test$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.test$violentcrimes.perpop)
)
```

```
##       RMSE   Rsquare
## 1 351.4334 0.6543754
```

We see that although the in-sample/training RMSE and R2 are worse than OLS without variable selection, the out-of-sample/testing RMSE and R2 are actually better. This indicates that the new model is more reliable.

4.2.3

What is your final model, after excluding high p-value variables? You will need to use model selection method to obtain this final model. Make it clear what criterion/criteria you have used and justify why they are appropriate.

We adopt best subset selection to select variables.

```
regfit_exh <- regsubsets(lm.input,method = 'exhaustive', nvmax = length(var.min),
                         data = data.frame(violentcrimes.perpop = crime.train$violentcrimes.perpop,
                                           x.matrix))

f.e <- summary(regfit_exh)
var.min2 <- names(coef(regfit_exh,which.min(f.e$bic)))
lm.input2 <- as.formula(paste("violentcrimes.perpop", "~", paste(var.min2[-1], collapse = "+")))
```

```
fit2 <- lm(lm.input2,data = data.frame(violentcrimes.perpop = crime.train$violentcrimes.perpop,
                                       x.matrix))
summary(fit2)
```

```
##
## Call:
## lm(formula = lm.input2, data = data.frame(violentcrimes.perpop = crime.train$violentcrimes.perpop,
##     x.matrix))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1017.08  -200.76   -32.24   150.78  1955.64
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3098.246    329.069   9.415  < 2e-16 ***
## race.pctblack       31.678      2.972  10.658  < 2e-16 ***
## male.pct.divorce    48.283     12.528   3.854 0.000142 ***
## pct.kids2parents   -17.288      3.737  -4.626 5.56e-06 ***
## pct.workmom        -16.506      3.541  -4.661 4.75e-06 ***
## pct.english.only    -9.315      1.531  -6.086 3.56e-09 ***
## num.homeless        31.807     12.484   2.548 0.011338 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 361.3 on 298 degrees of freedom
## Multiple R-squared:  0.7218, Adjusted R-squared:  0.7162
## F-statistic: 128.8 on 6 and 298 DF,  p-value: < 2.2e-16
```

Let us apply the model for training and testing performance evaluation using RMSE and R2.

```
# in-sample performance
predictions <- predict(fit2, data.frame(violentcrimes.perpop =
                crime.train$violentcrimes.perpop, x.matrix))
data.frame(
  RMSE = RMSE(predictions, crime.train$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.train$violentcrimes.perpop)
)
```

```
##       RMSE   Rsquare
## 1 357.1548 0.7217592
```

```
# out-of-sample performance
predictions <- predict(fit2, data.frame(violentcrimes.perpop =
                crime.test$violentcrimes.perpop, x.test))
data.frame(
  RMSE = RMSE(predictions, crime.test$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.test$violentcrimes.perpop)
)
```

```
##       RMSE   Rsquare
## 1 356.4376 0.6430036
```

So both the training and testing RMSE and R2 are slightly worse compared to the refitted LASSO model. However, the model now is much more simple and parsimonious. By reducing a lot of unimportant variables, the model fit only gets worse a little bit.

4.2.4

Try Ridge regression with 5-fold CV to select the tuning parameter. Compare its training and testing RMSE and R2 with the previous models.

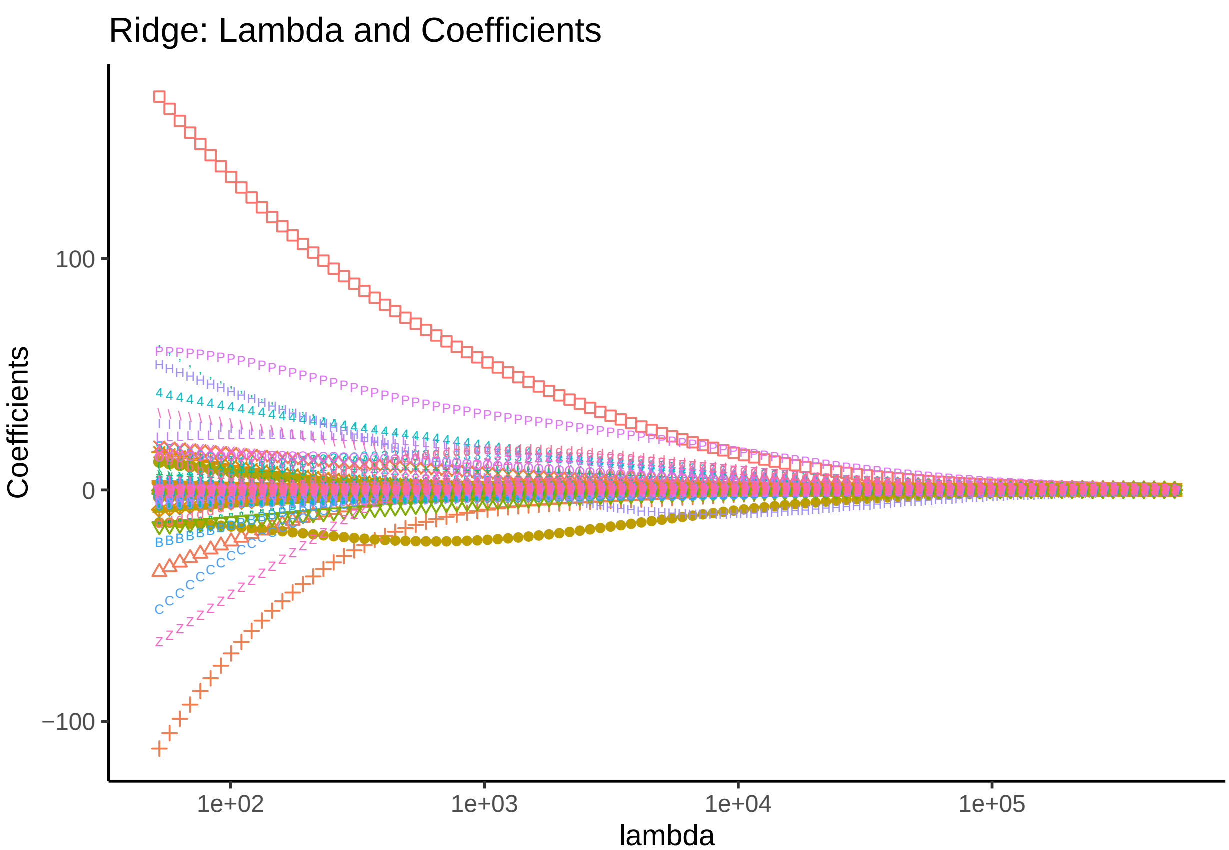
Let us first try Ridge with a set of λ's and look at the Ridge path.

```
fit.ridge.path <- glmnet(x.matrix, violentcrimes.perpop, alpha=0)
```

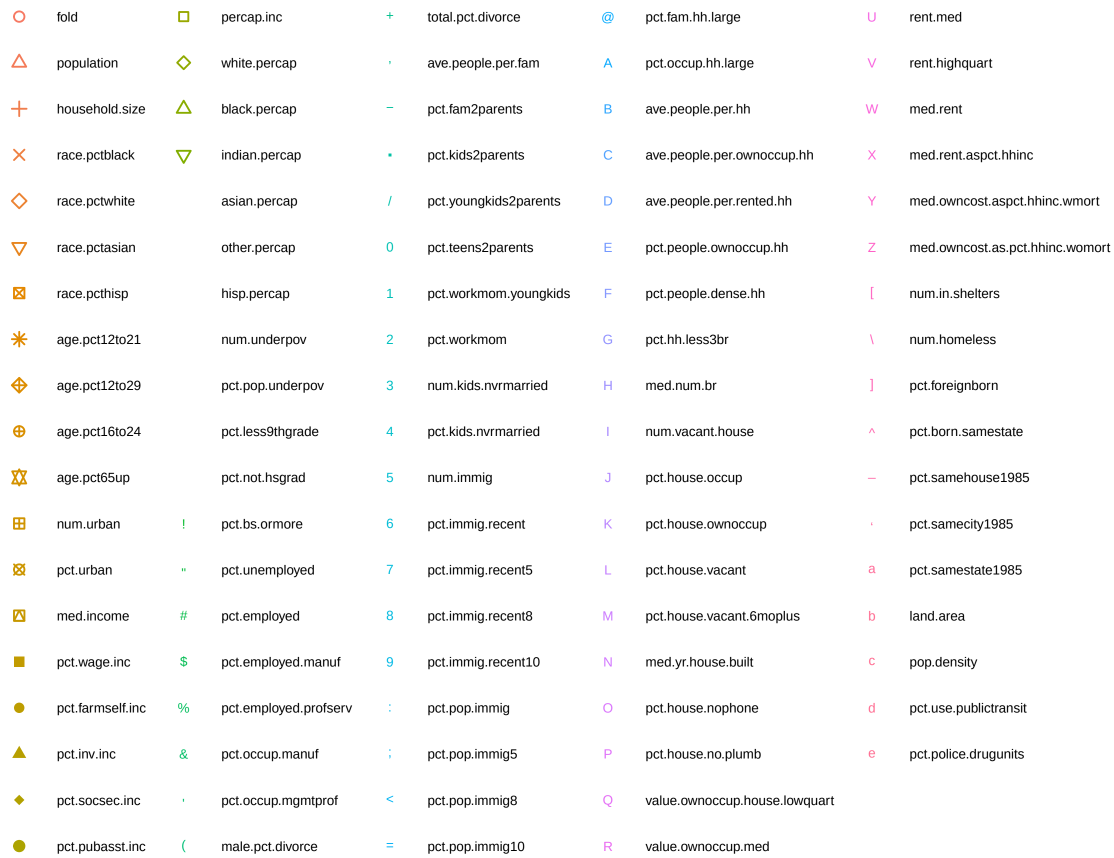
```
df.lasso <- fit.ridge.path$beta %>% as.matrix()
colnames(df.lasso) <- fit.ridge.path$lambda
df.lasso <- df.lasso %>% reshape2::melt()
colnames(df.lasso) <- c("Variables","lambda","Coefficients")

p.lasso <- df.lasso %>%
  ggplot(aes(x = lambda, y = Coefficients, color = Variables, shape = Variables)) +
  geom_point() + scale_x_log10() +
  scale_shape_manual(values=seq(0,dim(x.matrix)[2])) +
  ggtitle("Ridge: Lambda and Coefficients") + theme_classic() +
  theme(legend.position="bottom") + theme(legend.title = element_blank()) +
  theme(legend.text=element_text(size=5))

p.lasso + theme(legend.position = "none")
```




```
as_ggplot(get_legend(p.lasso))
```



We then use 5-fold CV to select the best λ.

```
set.seed(20211001)
fit.ridge.cv <- cv.glmnet(x.matrix,violentcrimes.perpop, alpha = 0,nfolds = 5)

lambda.ridge <- fit.ridge.cv$lambda.min  # fit.ridge$lambda.1se

fit.ridge <- glmnet(x.matrix, violentcrimes.perpop,
                    alpha=0,lambda=lambda.ridge)
```

Let us apply the model for training and testing performance evaluation using RMSE and R2.

```
# in-sample performance
predictions <- predict(fit.ridge, x.matrix) %>% as.vector()
data.frame(
  RMSE = RMSE(predictions, crime.train$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.train$violentcrimes.perpop)
)
```

```
##       RMSE   Rsquare
## 1 354.6953 0.7312004
```

```
# out-of-sample performance
predictions <- predict(fit.ridge, x.test) %>% as.vector()
data.frame(
  RMSE = RMSE(predictions, crime.test$violentcrimes.perpop),
  Rsquare = R2(predictions, crime.test$violentcrimes.perpop)
)
```

```
##       RMSE   Rsquare
## 1 330.1644 0.6957485
```

The testing performance is the best among all the above models. However the model is quite dense and hard to interpret. Whether Ridge and LASSO should be used in practice is a case-by-case decision. Elastic net which combines Ridge and LASSO could be another great choice.