Modern Data Mining: Clustering

Bopei Nie

2/26/2023

Contents

1 Overview

Multiple regression is one of the most popular methods used in statistics as well as in machine learning. We use linear models as a working model for its simplicity and interpretability. It is important that we use domain knowledge as much as we could to determine the form of the response as well as the function format for the factors. Then, when we have many possible features to be included in the working model it is inevitable that we need to choose a best possible model with a sensible criterion. `Cp`, `BIC` and regularizations such as LASSO are introduced. Be aware that if a model selection is done formally or informally, the inferences obtained with the final `lm()` fit may not be valid. Some adjustment will be needed. This last step is beyond the scope of this class. Check the current research line that Linda and collaborators are working on.

This homework consists of two parts: the first one is an exercise (you will feel it being a toy example after the covid case study) to get familiar with model selection skills such as, `Cp` and `BIC`. The main job is a rather involved case study about devastating covid19 pandemic. Please read through the case study first. This project is for sure a great one listed in your CV.

For covid case study, the major time and effort would be needed in EDA portion.

1.1 Objectives

- Model building process
- Methods
  - Model selection
    - All subsets
    - Forward/Backward
  - Regularization
    - LASSO (L1 penalty)
    - Ridge (L2 penalty)
    - Elastic net
- Understand the criteria
  - `Cp`
  - Testing Errors
  - `BIC`
  - `K fold Cross Validation`
  - `LASSO`
- Packages
  - `lm()`, `Anova`
  - `regsubsets()`
  - `glmnet()` & `cv.glmnet()`

2 Review materials

- Study lecture: Model selection
- Study lecture: Regularization
- Study lecture: Multiple regression

Review the code and concepts covered during lectures: multiple regression, model selection and penalized regression through elastic net.

3 Case study 1: `ISLR::Auto` data

This will be the last part of the Auto data from ISLR. The original data contains 408 observations about cars. It has some similarity as the Cars data that we use in our lectures. To get the data, first install the package `ISLR`. The data set `Auto` should be loaded automatically. We use this case to go through methods learned so far.

```
head(Auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
Auto <- Auto[, -ncol(Auto)]
```

```
colnames(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"
```

Final modelling question: We want to explore the effects of each feature as best as possible.

1) Preparing variables:

a) You may explore the possibility of variable transformations. We normally do not suggest to transform x for the purpose of interpretation. You may consider to transform y to either correct the violation of the linear model assumptions or if you feel a transformation of y makes more sense from some theory. In this case we suggest you to look into `GPM=1/MPG`. Compare residual plots of MPG or GPM as responses and see which one might yield a more satisfactory patterns.

In addition, can you provide some background knowledge to support the notion: it makes more sense to model `GPM`?

```
origin <- as.factor(Auto$origin)
```

```
Auto$GPM <- 1/Auto$mpg

model_mpg <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + year + origin, da
summary(model_mpg)
```

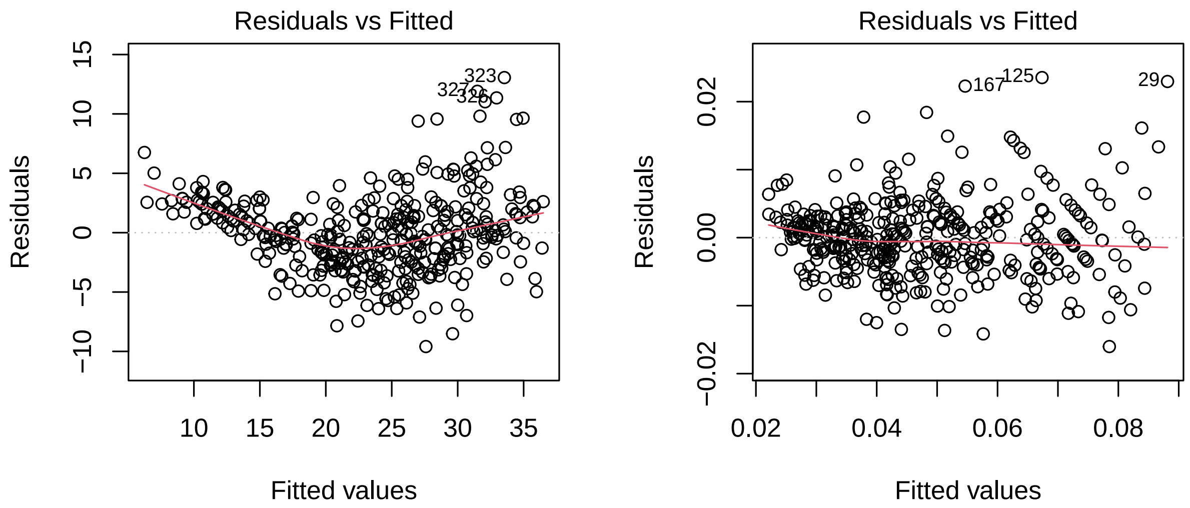
```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower + weight +
##     acceleration + year + origin, data = Auto)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.590 -2.157 -0.117  1.869 13.060
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.72e+01   4.64e+00   -3.71  0.00024 ***
## cylinders    -4.93e-01   3.23e-01   -1.53  0.12780
## displacement  1.99e-02   7.51e-03    2.65  0.00844 **
## horsepower   -1.70e-02   1.38e-02   -1.23  0.21963
## weight       -6.47e-03   6.52e-04   -9.93  < 2e-16 ***
## acceleration  8.06e-02   9.88e-02    0.82  0.41548
## year          7.51e-01   5.10e-02   14.73  < 2e-16 ***
## origin        1.43e+00   2.78e-01    5.13  4.7e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.33 on 384 degrees of freedom
## Multiple R-squared:  0.821,  Adjusted R-squared:  0.818
## F-statistic:  252 on 7 and 384 DF,  p-value: <2e-16
```

```r
model_gpm <- lm(GPM ~ cylinders + displacement + horsepower + weight + acceleration + year + origin, da
summary(model_gpm)
```

```
##
## Call:
## lm(formula = GPM ~ cylinders + displacement + horsepower + weight +
##     acceleration + year + origin, data = Auto)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.016017 -0.003348 -0.000111  0.002933  0.023540
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.14e-02   7.94e-03   11.52  < 2e-16 ***
## cylinders     1.51e-03   5.53e-04    2.73   0.0066 **
## displacement -2.57e-05   1.28e-05   -2.00   0.0461 *
## horsepower    1.26e-04   2.36e-05    5.33  1.7e-07 ***
## weight        1.09e-05   1.11e-06    9.75  < 2e-16 ***
## acceleration  3.42e-04   1.69e-04    2.03   0.0435 *
## year         -1.26e-03   8.71e-05  -14.51  < 2e-16 ***
## origin       -1.01e-03   4.75e-04   -2.13   0.0339 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00569 on 384 degrees of freedom
## Multiple R-squared:  0.885,  Adjusted R-squared:  0.883
## F-statistic:  423 on 7 and 384 DF,  p-value: <2e-16
```

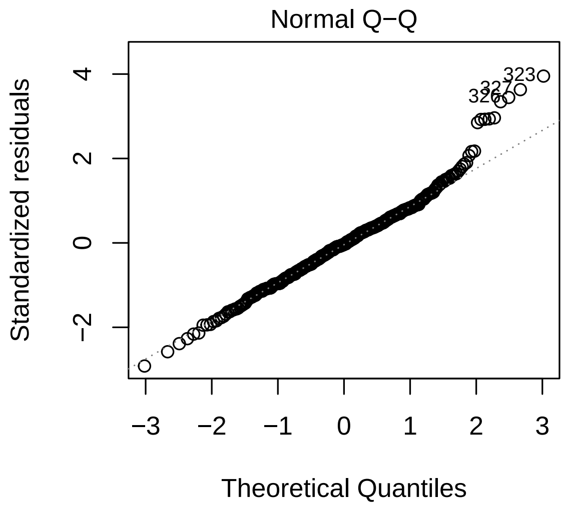
GPM has a larger R-squared than MPG does, which suggests it is more accurately captured.

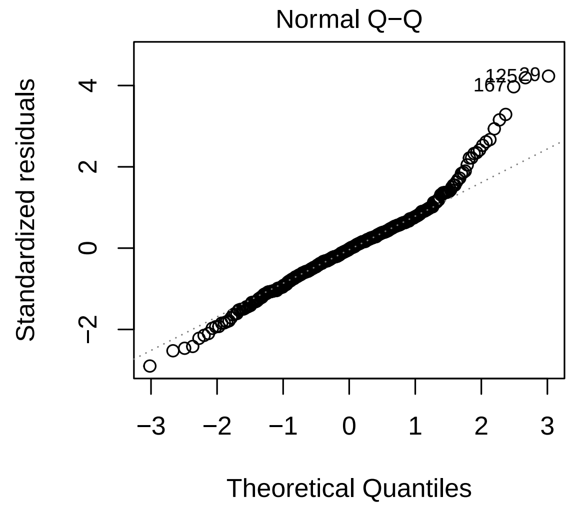
```
par(mfrow=c(1,2))
plot(model_mpg, 1)
plot(model_gpm, 1)
```



We see a better fit when using GPM The MPG Residuals vs Fitted plot has a valley as the red line, while the GPM Residuals vs Fitted plot shows that the residuals are scattered around 0. The red line is almost horizontal. This means that there is barely any relationship between the residuals and the predicted values, which is what we want to achieve.

```
par(mfrow=c(1,2))
plot(model_mpg, 2)
plot(model_gpm, 2)
```






```
shapiro.test(residuals(model_mpg))
shapiro.test(residuals(model_gpm))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model_mpg)
## W = 1, p-value = 6e-06
##
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model_gpm)
## W = 1, p-value = 4e-08
```

Both models show significant p-values for the normality test, so we reject the null hypothesis and conclude residuals for both models are not normally distributed.

Since GPM has better Residuals vs Fitted plot, we prefer using gpm. GPM stands for gallons per mile. It can measure gas consumption, which is an important feature of cars. Hence, it makes sense to model GPM.

b) You may also explore by adding interactions and higher order terms. The model(s) should be as *parsimonious* (simple) as possible, unless the gain in accuracy is significant from your point of view.

```
model_gpm1 <- lm(GPM ~ cylinders + displacement + horsepower + weight + acceleration + year + origin
                 +horsepower*acceleration+weight*year, data = Auto)
summary(model_gpm1)
```

```
##
## Call:
## lm(formula = GPM ~ cylinders + displacement + horsepower + weight +
##     acceleration + year + origin + horsepower * acceleration +
```

```
##     weight * year, data = Auto)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.016589 -0.003127 -0.000421  0.002653  0.025082
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              2.29e-03   2.32e-02    0.10   0.9214
## cylinders                9.96e-04   5.40e-04    1.84   0.0659 .
## displacement             1.08e-05   1.38e-05    0.78   0.4340
## horsepower              -1.20e-04   4.33e-05   -2.77   0.0059 **
## weight                   4.83e-05   8.13e-06    5.93  6.6e-09 ***
## acceleration            -1.04e-03   2.76e-04   -3.76   0.0002 ***
## year                     2.24e-04   3.06e-04    0.73   0.4647
## origin                  -6.67e-04   4.58e-04   -1.46   0.1460
## horsepower:acceleration  1.96e-05   3.08e-06    6.34  6.4e-10 ***
## weight:year             -5.48e-07   1.08e-07   -5.06  6.4e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00534 on 382 degrees of freedom
## Multiple R-squared:   0.9,   Adjusted R-squared:  0.897
## F-statistic:  380 on 9 and 382 DF,  p-value: <2e-16
```

I experienced with interactions and higher order terms. In the end, I decided to add two interaction terms: horsepower and acceleration, and weight and year, which increases r-squared by 0.015.

c) Use Mallow's C_p or BIC to select the model.

```r
Auto <- Auto[, -1] #remove the mpg column
```

```r
library(leaps)
```

```r
regsubsets_fit <- regsubsets(GPM ~ ., data = Auto, nvmax = 20)
summary(regsubsets_fit)
```

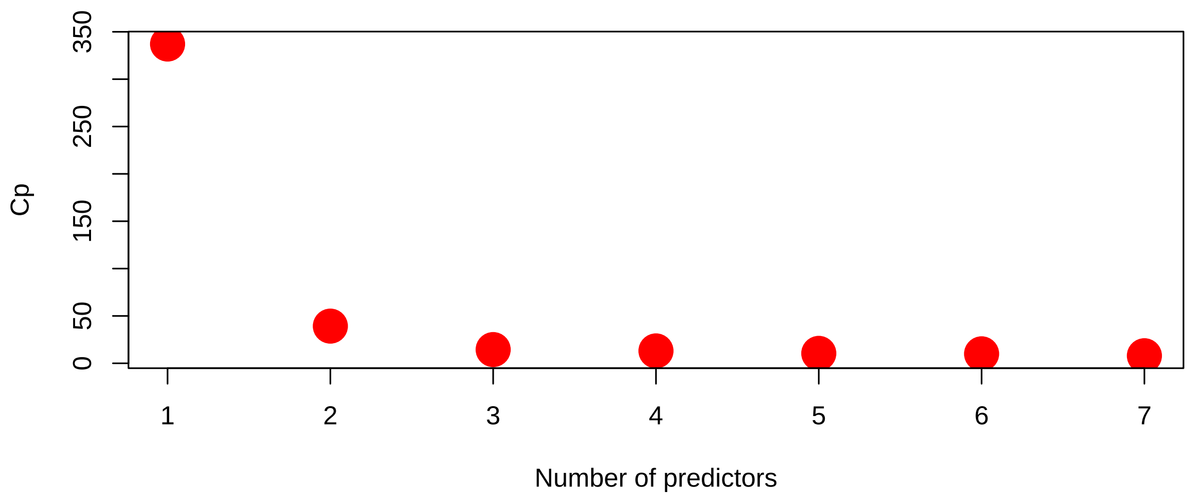
```
## Subset selection object
## Call: regsubsets.formula(GPM ~ ., data = Auto, nvmax = 20)
## 7 Variables  (and intercept)
##              Forced in Forced out
## cylinders        FALSE      FALSE
## displacement     FALSE      FALSE
## horsepower       FALSE      FALSE
## weight           FALSE      FALSE
## acceleration     FALSE      FALSE
## year             FALSE      FALSE
## origin           FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: exhaustive
##          cylinders displacement horsepower weight acceleration year origin
## 1  ( 1 ) " "       " "          " "        "*"    " "          " "  " "
```

```
## 2  ( 1 ) " "        " "          " "         "*"     " "           "*"   " "
## 3  ( 1 ) " "        " "          "*"         "*"     " "           "*"   " "
## 4  ( 1 ) " "        " "          "*"         "*"     "*"           "*"   " "
## 5  ( 1 ) "*"        " "          "*"         "*"     "*"           "*"   " "
## 6  ( 1 ) "*"        " "          "*"         "*"     "*"           "*"   "*"
## 7  ( 1 ) "*"        "*"          "*"         "*"     "*"           "*"   "*"
```

```
summary(regsubsets_fit)$cp
```

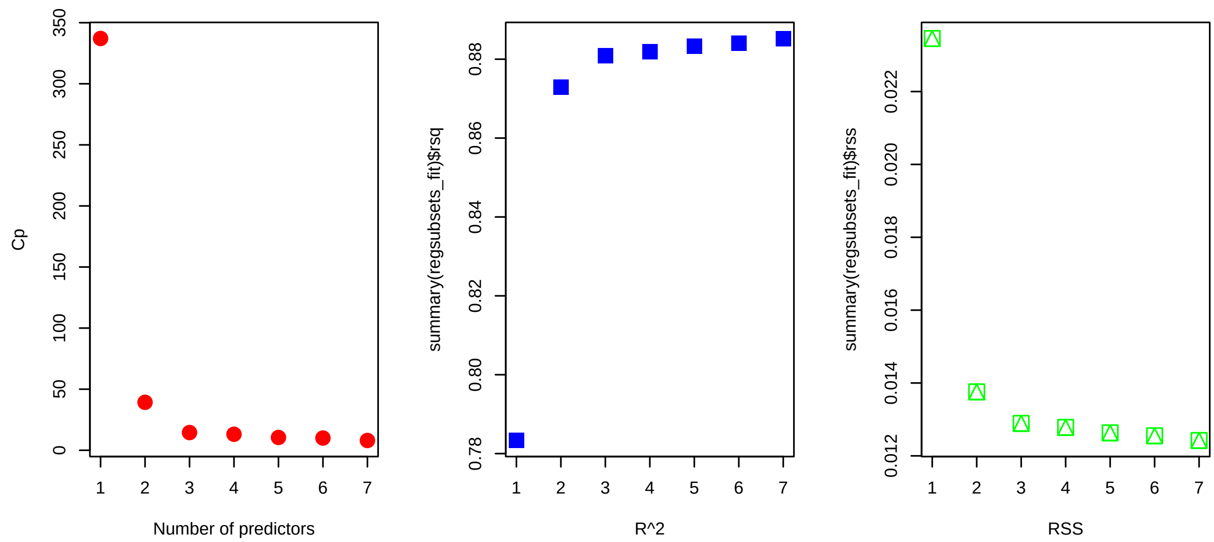
```
## [1] 337.1  39.3  14.5  13.1  10.5  10.0   8.0
```

```
plot(summary(regsubsets_fit)$cp, xlab="Number of predictors",
     ylab="Cp", col="red", pch=16, cex =3)
```



It shows that a model with 7 variables has the smaller prediction error.

```
par(mfrow=c(1, 3))  # see diff criteria
  plot(summary(regsubsets_fit)$cp, xlab="Number of predictors",
     ylab="Cp", col="red",  pch=16, cex=2)
  plot(summary(regsubsets_fit)$rsq, xlab="R^2", pch=15,  col= "blue", cex=2)
  plot(summary(regsubsets_fit)$rss, xlab="RSS", pch = 14, col="green", cex=2)
```




```
par(mfrow=c(1,1))
```

The model with 7 variables has the lowest Cp and RSS, but has the highest R^2 score.

```
opt.size <- which.min(summary(regsubsets_fit)$cp)
opt.size
```

```
## [1] 7
```

```
coef(regsubsets_fit,opt.size)
```

```
##  (Intercept)    cylinders displacement   horsepower       weight acceleration
##     9.14e-02     1.51e-03    -2.57e-05     1.26e-04     1.09e-05     3.42e-04
##         year       origin
##    -1.26e-03    -1.01e-03
```

```
fit.exh.var <- summary(regsubsets_fit)$which # logic indicators which variables are in
fit.exh.var[opt.size,]
```

```
##  (Intercept)    cylinders displacement   horsepower       weight acceleration
##         TRUE         TRUE         TRUE         TRUE         TRUE         TRUE
##         year       origin
##         TRUE         TRUE
```

```
colnames(fit.exh.var)[fit.exh.var[opt.size,]]
```

```
## [1] "(Intercept)"  "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"
```

2) Describe the final model and its accuracy. Include diagnostic plots with particular focus on the model residuals.

- Summarize the effects found.

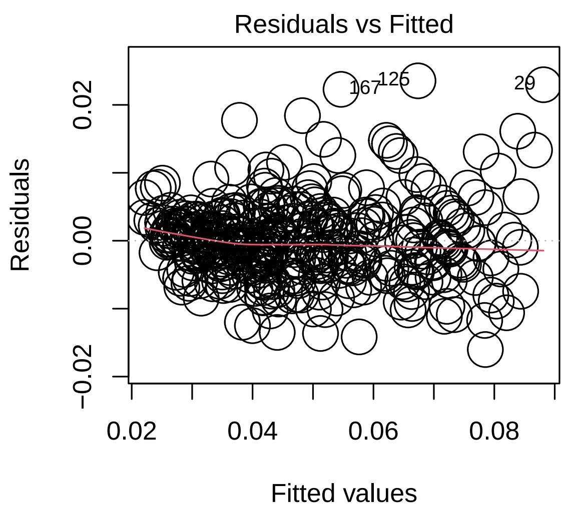
"cylinders", "displacement", "horsepower", "weight", "acceleration", "year", and "origin" are important features. We include them in our final model.

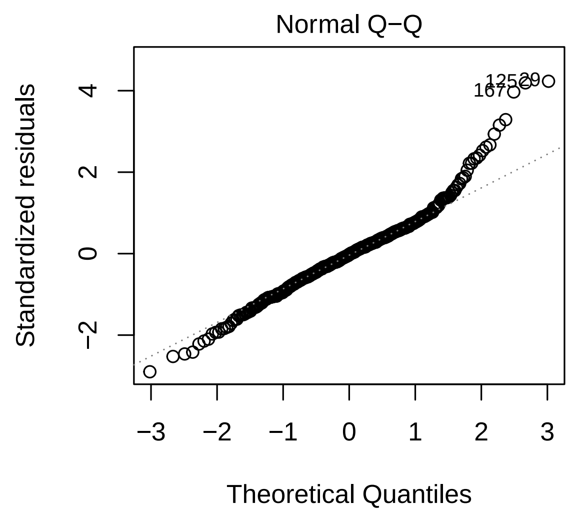
```r
fit.final <- lm(GPM ~ cylinders + displacement + horsepower + weight + acceleration + year + origin, Au
summary(fit.final)
```

```
##
## Call:
## lm(formula = GPM ~ cylinders + displacement + horsepower + weight +
##     acceleration + year + origin, data = Auto)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.016017 -0.003348 -0.000111  0.002933  0.023540
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.14e-02   7.94e-03   11.52  < 2e-16 ***
## cylinders     1.51e-03   5.53e-04    2.73   0.0066 **
## displacement -2.57e-05   1.28e-05   -2.00   0.0461 *
## horsepower    1.26e-04   2.36e-05    5.33  1.7e-07 ***
## weight        1.09e-05   1.11e-06    9.75  < 2e-16 ***
## acceleration  3.42e-04   1.69e-04    2.03   0.0435 *
## year         -1.26e-03   8.71e-05  -14.51  < 2e-16 ***
## origin       -1.01e-03   4.75e-04   -2.13   0.0339 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00569 on 384 degrees of freedom
## Multiple R-squared:  0.885,  Adjusted R-squared:  0.883
## F-statistic:  423 on 7 and 384 DF,  p-value: <2e-16
```

The final model has a Residual standard error of 0.00569, a Multiple R-squared of 0.885 and a significant p-value.

```r
par(mfrow=c(1,2))
plot(fit.final, 1, cex =3)
plot(fit.final, 2)
```





The residuals are nicely distributed along a horizontal line around 0. Just like we analyzed before, the residuals are not normally distributed.

- Predict the `mpg` of a car that is: built in 1983, in the US, red, 180 inches long, 8 cylinders, 350 displacement, 260 as horsepower, and weighs 4,000 pounds. Give a 95% CI.

```
colMeans(Auto)
```

```
##    cylinders displacement   horsepower       weight acceleration         year
##     5.47e+00     1.94e+02     1.04e+02     2.98e+03     1.55e+01     7.60e+01
##       origin          GPM
##     1.58e+00     4.78e-02
```

Since we do not have entry for acceleration, we add in the mean of acceleration from training data.

```
newcar <- Auto[1, ] # Create a new row with same structure as in Auto
newcar[1] <- 8
newcar[2] <- 350
newcar[3] <- 260
newcar[4] <- 4000
newcar[5] <- 1.55e+01
newcar[6] <- 83
newcar[7]<-1
newcar[8]<-"NA"
newcar
```

```
##   cylinders displacement horsepower weight acceleration year origin GPM
## 1         8          350        260   4000         15.5   83      1  NA
```

```
predict(fit.final, newcar, interval = "predict", se.fit = TRUE)
```

```
## $fit
##    fit    lwr    upr
## 1 0.07 0.0572 0.0827
##
## $se.fit
## [1] 0.00309
##
## $df
## [1] 384
##
## $residual.scale
## [1] 0.00569
```

The predicted GPM is 0.0647 in interval [0.0525, 0.0768] with 95% Confidence level.

```
1/(predict(fit.final, newcar, interval = "predict", se.fit = TRUE)$fit)
```

```
##    fit  lwr  upr
## 1 14.3 17.5 12.1
```

If we inverse the number, we get the predicted MPG is 15.5 in interval [13, 19] with 95% Confidence level.

We also want to try to fit new car into model_gpm1, which has the seven variables and two interaction terms and has a higher R-square.

```
predict(model_gpm1, newcar, interval = "predict", se.fit = TRUE)
```

```
## $fit
##      fit    lwr    upr
## 1 0.0748 0.0621 0.0875
##
## $se.fit
## [1] 0.00364
##
## $df
## [1] 382
##
## $residual.scale
## [1] 0.00534
```

The predicted GPM is 0.0748 in interval [0.0621, 0.0875] with 95% Confidence level.

```
1/(predict(model_gpm1, newcar, interval = "predict", se.fit = TRUE)$fit)
```

```
##    fit  lwr  upr
## 1 13.4 16.1 11.4
```

If we inverse the number, we get the predicted MPG is 13.4 in interval [11.4, 16.1] with 95% Confidence level.

- Any suggestions as to how to improve the quality of the study?
- Examine more relevant features.
- Examine more observations. Now we only have less than 400 observations.
- Explore other model selection methods.

4 Case study 2: COVID19

See a seperate file covid_case_study.Rmd for details.