Problem Set 3: Least Squares Regression

Stat 154, Fall 2017, Prof. Sanchez

Self grade due date: Tu Oct-10 (before midnight)

Problem 1 (10 pts)

Consider a simple linear regression model with OLS fitted values given by:

$$\hat{y}_i = b_0 + b_1 x_i$$

Show that the sum of residuals equals zero, that is: $\sum_{i=1}^{n} e_i = 0$

Recall that $b_0 = \bar{y} - b_1 \bar{x}$

$$\begin{aligned}
\sum_{i=1}^{n} e_i &= \sum_{i=1}^{n} (y_i - \hat{y}_i) \\
&= \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i) \\
&= \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} b_0 - \sum_{i=1}^{n} b_1 x_i \\
&= n\bar{y} - nb_0 - b_1 \bar{x} \\
&= n(\bar{y} - b_0 - b_1 \bar{x}) \\
&= n(\bar{y} - (\bar{y} - b_1 \bar{x}) - b_1 \bar{x}) \\
&= n(\bar{y} - \bar{y} + b_1 \bar{x} - b_1 \bar{x}) \\
&= 0
\end{aligned}$$

Problem 2 (30 pts)

We examine a response variable Y in terms of two predictors X and Z. There are n observations. Let $\mathbf{X}$ be a matrix formed by a constant term of $\mathbf{1}$, and the vectors $\mathbf{x}$ and $\mathbf{z}$. Consider the cross-product matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ given below:

$$\mathbf{X}^\mathsf{T}\mathbf{X} = \begin{bmatrix} 30 & 0 & 0 \\ ? & 10 & 7 \\ ? & ? & 15 \end{bmatrix}$$

a. Complete the missing values denoted by "?", and determine the value of n? (10 pts)

The matrix $\mathbf{X}^\top\mathbf{X}$ is symmetric, so the entries are:

$$\mathbf{X}^\top\mathbf{X} = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 10 & 7 \\ 0 & 7 & 15 \end{bmatrix}$$

From the first entry of $\mathbf{X}^\top\mathbf{X}$, the value of $n = 30$

b. Calculate the linear correlation coefficient between X and Z. (10 pts)

Also from $\mathbf{X}^\top\mathbf{X}$, we can deduce that $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = 0$ and $\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i = 0$.

$$\begin{aligned} cor(X,Y) &= \frac{\sum_{i=1}^{30}(x_i - \bar{x})(z_i - \bar{z})}{\sqrt{\sum_{i=1}^{30}(x_i - \bar{x})^2 \sum_{i=1}^{30}(z_i - \bar{z})^2}} \\ &= \frac{\sum_{i=1}^{30} x_i z_i}{\sqrt{\sum_{i=1}^{30} x_i^2 \sum_{i=1}^{30} z_i^2}} \\ &= \frac{7}{\sqrt{150}} = 0.57 \end{aligned}$$

c. If the OLS regression equation is: $\hat{y}_i = -2 + x_i + 2z_i$, What is the value of $\bar{y}$? (5 pts)

From the regression equation we have that $y_i = -2 + x_i + 2z_i + e_i$.

$$\begin{aligned} \bar{y} &= \frac{1}{n}\sum_{i=1}^{30} y_i \\ &= \frac{1}{n}\sum_{i=1}^{30}(-2 + x_i + 2z_i + e_i) \\ &= \frac{1}{n}(-2n + n\bar{x} + 2n\bar{z} + \sum_{i=1}^{30} e_i) \end{aligned}$$

Since $\bar{x} = \bar{z} = 0$, we get that $\bar{y} = -2$. Remember that the sum of residuals is zero.

d. If the residual sum of squares (RSS) is 12, What is the value of R^2? (5 pts)

Consider the decomposition of the total sum of squares (TSS):

$$\underbrace{\sum_{i=1}^{n}(y_i - \bar{y})^2}_{\text{TSS}} = \underbrace{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}_{\text{RSS}} + \underbrace{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}_{\text{ESS}}$$

where: RSS is the residuals sum of squares, and ESS is the explained sum of squares.

Equivalently:

$$1 = \frac{\text{RSS}}{\text{TSS}} + \frac{\text{ESS}}{\text{TSS}}$$

The coefficient of determination R^2 can then be expressed in terms of the TSS decomposition:

$$\begin{aligned} R^2 &= \frac{\|\hat{\mathbf{y}}\|^2}{\|\mathbf{y}\|^2} \\ &= \frac{\text{ESS}}{\text{TSS}} \\ &= 1 - \frac{\text{RSS}}{\text{TSS}} \\ &= 1 - \frac{\|\mathbf{e}\|^2}{\|\mathbf{y} - \bar{y}\mathbf{1}\|^2} \end{aligned}$$

The explained sum of squares (ESS) can be obtained as:

$$\begin{aligned} \|\hat{\mathbf{y}} - \bar{y}\mathbf{1}\|^2 &= \sum_{i=1}^{30}(-2 + x_i + 2z_i - (-2))^2 \\ &= \sum_{i=1}^{30}(x_i^2 + 4x_i z_i + 4z_i^2) \\ &= 10 + 4(7) + 4(15) \\ &= 98. \end{aligned}$$

We know that RSS is 12, and that the Total Sum of Squares is: $98 + 12 = 110$.

Thus:

$$R^2 = 1 - \frac{12}{110} = 0.891.$$

Problem 3 (30 pts)

In this exercise you will create some simulated data and will fit simple linear regression models to it. Make sure to use `set.seed(1)` prior to starting part (a) to ensure consistent results.

- Parts (a) - (g) are worth 10 pts
- Part (h) is worth 10 pts
- Part (i) is worth 10 pts

a. Using the `rnorm()` function, create a vector, `x`, containing 100 observations drawn from a $N(0, 1)$ distribution. This represents a feature, X.

```
set.seed(1)
n <- 100

# predictor
x <- rnorm(n, mean = 0, sd = 1)
```

b. Using the `rnorm()` function, create a vector, `eps`, containing 100 observations drawn from a $N(0, 0.25)$ distribution i.e. a normal distribution with mean zero and variance 0.25.

Notice that the problem specifies a normal distribution with variance = 0.25. However, the function `rnorm()` uses the argument `sd` (standard deviation). Consequently, you need to use `sd = sqrt(0.25)` to generate the correct `eps`

```
# keep in mind that rnorm() uses sd
eps <- rnorm(n, mean = 0, sd = sqrt(0.25))
```

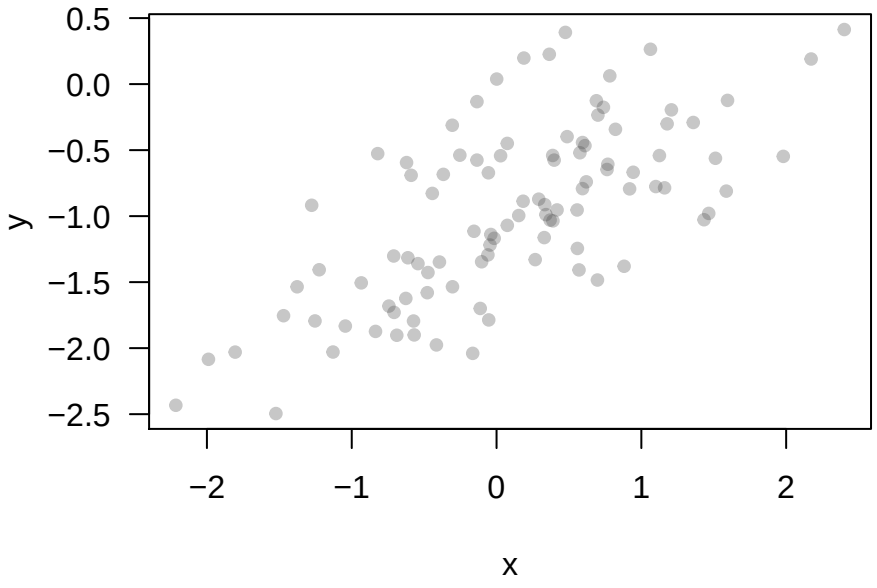
c. Using `x` and `eps`, generate a vector `y` according to the model:

$$Y = -1 + 0.5X + \epsilon$$

```
# response
y <- -1 + (0.5 * x) + eps
```

d. Create a scatterplot displaying the relationship between `x` and `y`. Comment on what you observe.

```
plot(x, y, las = 1, pch = 19, cex = 0.8, col = "#55555555")
```



From the scatterplot, you can say that the cloud of points has a positive linear trend, or simply that X and Y are positively correlated. This plot is just a display to check that the generated data makes sense with the way the were generated.

e. Fit a least squares linear model to predict `y` using `x`. Comment on the model obtained. How do $\hat{\beta}_0$ and $\hat{\beta}_1$ compare to β_0 and β_1?

To fit the OLS model you can use `lm()` or calculate the results with your own matrix operations.

```
# simple linear regression
model1 <- lm(y ~ x)
model1_sum <- summary(model1)
model1_sum
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93842 -0.30688 -0.06975  0.26970  1.17309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
## x            0.49947    0.05386   9.273 4.58e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 98 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4619
## F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

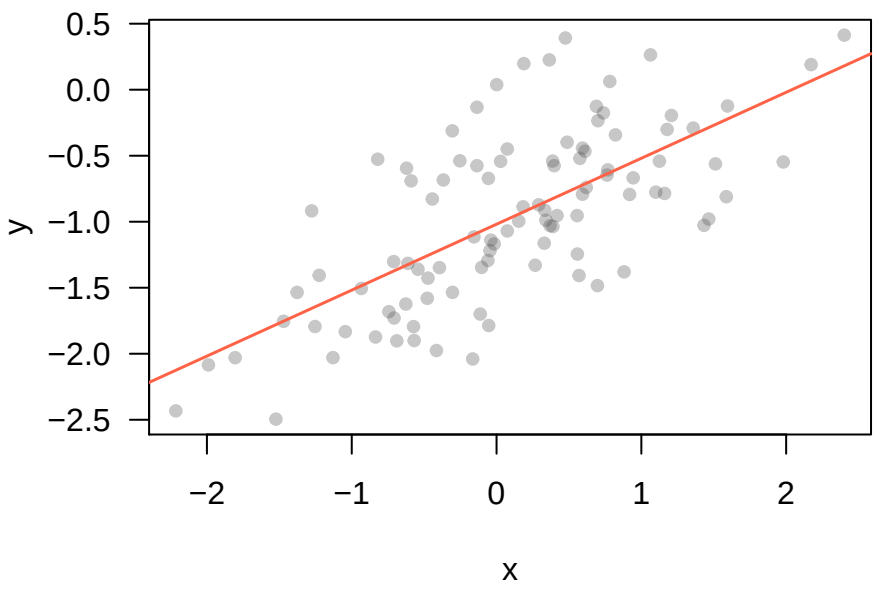
The estimated coefficients $\hat{\beta}_1 = 0.499$, and $\hat{\beta}_0 = -1.019$ are close to the theoretical ones $\beta_1 = 0.5$ and $\beta_0 = -1$. If you obtained coefficients that seem to be "considerably" different from the theoretical ones, you are probably doing something wrong.

f. Display the least squares line on the scatterplot obtained in (d). Draw the theoretical regression line on the plot, in a different color. Use the `legend()` command to create an appropriate legend.

Since the problem does not specify what plotting functions should be used, you can use R base `plot()`, or `ggplot()` from the package `"ggplot2"`, or something else.

Although the problem requires using the `legend()` function, we can take this element as an optinal thing. No need to penalize if your plot lacks a legend.

```r
plot(x, y, las = 1, pch = 19, cex = 0.8, col = "#55555555")
abline(model1, col = "tomato", lwd = 1.5)
```



g. Now fit a polynomial regression model that predicts `y` using `x` and `x^2`. Is there evidence that the quadratic term improves the model fit? Explain your answer.

```r
# polynomial regression
model2 <- lm(y ~ x + I(x^2))
model2_sum <- summary(model2)
model2_sum
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98252 -0.31270 -0.06441  0.29014  1.13500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.97164    0.05883 -16.517  < 2e-16 ***
## x            0.50858    0.05399   9.420  2.4e-15 ***
## I(x^2)      -0.05946    0.04238  -1.403    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.479 on 97 degrees of freedom
## Multiple R-squared:  0.4779, Adjusted R-squared:  0.4672
## F-statistic:  44.4 on 2 and 97 DF,  p-value: 2.038e-14
```

To compare the fits between `model1` and `model2`, we can examine the R^2_{adj} and the Mean Square Errors (MSE). Keep in mind that if you just simply compare the (unadjusted) R^2, this comparison won't be fair; because R^2 depends on the number of predictors. As the number of predictors increases, so does R^2; that's why you must use the adjusted version.

```
# compare R2-adjusted values
c(model1_sum$adj.r.squared, model2_sum$adj.r.squared)
```

```
## [1] 0.4619164 0.4671806
```

```
# compare MSEs
mse1 <- mean(model1$residuals^2)
mse2 <- mean(model2$residuals^2)
c(mse1, mse2)
```

```
## [1] 0.2270890 0.2225728
```

In both cases, the polynomial regression seems to do a better job of fitting the data. It has a larger R^2_{adj}, and a smaller MSE compared to the simple linear model.

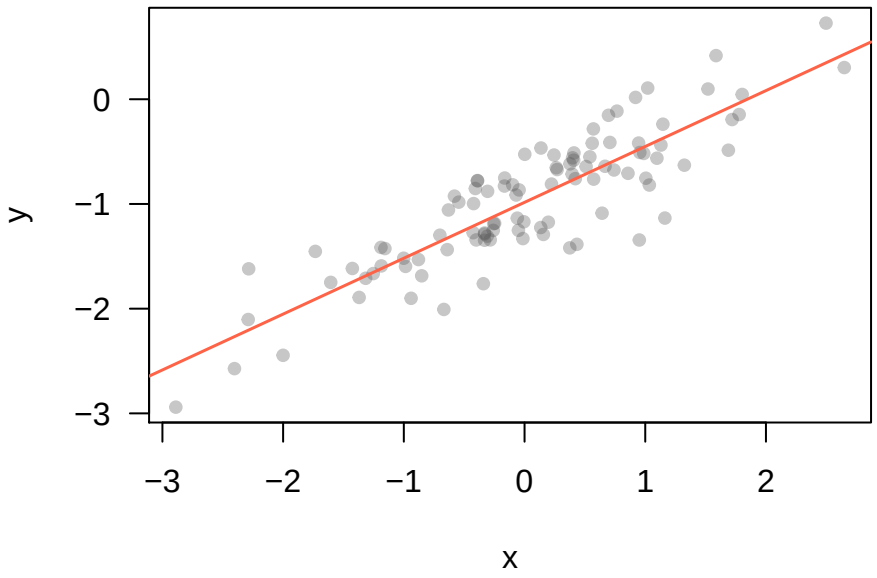
h. Repeat (a)-(f) after modifying the data generation process in such a way that there is *less* noise in the data. The theoretical model $Y = -1 + 0.5X + \epsilon$ should remain the same. Describe your results. (10 pts)

```
# predictor, error (less noise), and response
x <- rnorm(n, 0, 1)
eps <- rnorm(n, 0, sqrt(0.1))
y <- -1 + 0.5 * x + eps

# linear regression
model3 <- lm(y ~ x)
model3
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##     -0.9847       0.5336
```

```
plot(x, y, las = 1, pch = 19, cex = 0.8, col = "#55555555")
abline(model3, col = "tomato", lwd = 1.5)
```



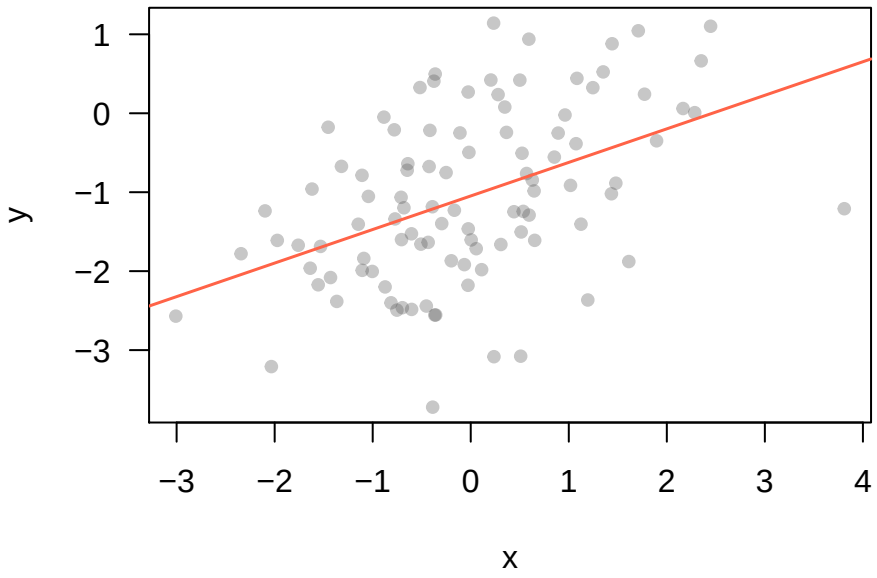
i. Repeat (a)-(f) after modifying the data generation process in such a way that there is *more* noise in the data. The theoretical model $Y = -1 + 0.5X + \epsilon$ should remain the same. Describe your results. (10 pts)

```
# predictor, error (more noise), and response
x <- rnorm(100, mean = 0, sd = 1)
eps <- rnorm(100, mean = 0, sd = 1)
y <- -1 + (0.5 * x) + eps

# linear regression
model4 <- lm(y ~ x)
model4
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##     -1.0475       0.4251
```

```
plot(x, y, las = 1, pch = 19, cex = 0.8, col = "#55555555")
abline(model4, col = "tomato", lwd = 1.5)
```



Problem 4 (10 pts)

Write a function `ols_fit` that computes the (ordinary) least squares solution, via QR decomposition, given an input model matrix $\mathbf{X}$ and a response vector $\mathbf{y}$. The function `ols_fit` should take two arguments:

- `X`: a model (or design) matrix of predictors
- `y`: a vector for the response variable

The output should be a `list` with elements:

- `coefficients`: estimated regression coefficients
- `y_values`: vector of observed values
- `fitted_values`: vector of fitted (hat) values
- `residuals`: vector of residuals
- `n`: number of observations n, and
- `q`: number of columns in the model matrix `X`

```r
# write a function ols_fit() to compute the OLS solution
# via QR decomposition
ols_fit <- function(X, y) {
  QR <- qr(X)
  Q <- qr.Q(QR)
  R <- qr.R(QR)
  Qty <- t(Q) %*% y
  b <- backsolve(R, Qty)
  y_hat <- as.vector(X %*% b)
  # output
  list(
    coefficients = b,
    y_values = y,
    fitted_values = y_hat,
    residuals = y - y_hat,
    n = nrow(X),
    q = ncol(X)
  )
}
```

Consider the data frame `mtcars` that comes in R. Assume a vector `y` for the response variable `mpg`, and the following matrix `X` formed by a column of 1's (intercept), `disp`, and `hp`:

	intercept	disp	hp
Mazda RX4	1	160	110
Mazda RX4 Wag	1	160	110
Datsun 710	1	108	93
⋮	⋮	⋮	⋮
Maserati Bora	1	301	335
Volvo 142E	1	121	109

Unless all the variables (i.e. the predictors and response) are mean-centered, `X` will be a matrix of dimension $n \times (p+1)$, where n is the number of observations, and p is the number of predictors. In other words, `X` should include the column of 1's to account for the intercept term, except for when all variables are mean-centered.

Test your function with the example above (using the `mtcars` data set), and confirm you get the same `coefficients`, and the same `summary()` statistics for `fitted_values` and `residuals`.

```r
# example with mtcars
X <- as.matrix(cbind(1, mtcars[ ,c('disp', 'hp')]))
y <- mtcars$mpg
```

You should be able to call `ols_fit` as follows:

```r
fit <- ols_fit(X, y)
names(fit)
```

```
## [1] "coefficients"  "y_values"      "fitted_values" "residuals"
## [5] "n"             "q"
```

```r
fit$coefficients
```

```
##             [,1]
## [1,] 30.73590425
## [2,] -0.03034628
## [3,] -0.02484008
```

```r
summary(fit$fitted_values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.32   15.16   21.64   20.09   24.70   27.15
```

```r
summary(fit$residuals)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.7950 -2.3040 -0.8246  0.0000  1.8580  6.9360
```

Problem 5 (10 pts)

Write auxiliary functions `R2()` and `RSE()` to compute the coefficient of determination R^2, and the Residual Standard Error (RSE), respectively.

a. The function `R2` computes the coefficient of determination R^2:

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

```r
R2 <- function(obj) {
  y <- obj$y_values
  y_hat <- obj$fitted_values
  y_mean <- mean(y)
  # Explained and Total Sums of Squares
  ESS <- sum((y_hat - y_mean)^2)
  TSS <- sum((y - y_mean)^2)
  # coefficient of determination
  R2 <- ESS / TSS
  R2
}
```

The function `R2()` should take the output of `ols_fit()` as the only input. This means that you should be able to call `R2()` like:

```
fit <- ols_fit(X, y)
R2(fit)
```

```
## [1] 0.7482402
```

b. The function `RSE` computes the Residual Standard Error given by the formula:

$$\text{RSE} = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n - p - 1}}$$

where p is the number of predictors.

Instead of using p, you can use:

$$\text{RSE} = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n - q}}$$

where q is the number of columns in the model matrix `X`, returned by `ols_fit()`.

```
RSE <- function(obj) {
  RSS = sum(obj$residuals^2)
  sqrt(RSS / (obj$n - obj$q))
}
```

The function `RSE()` should take the output of `ols_fit()` as the only input. This means that you should be able to call `RSE()` like:

```
fit <- ols_fit(X, y)
RSE(fit)
```

```
## [1] 3.126601
```

Problem 6 (20 pts)

Consider the dataset `prostate` available in the `data/` folder of the course github repository. This dataset comes from a study on 97 men with prostate cancer who were due to receive a radical prostatectomy. Use your `ols_fit()` and `R2()` functions to:

```
dat <- read.table('prostate.txt')
```

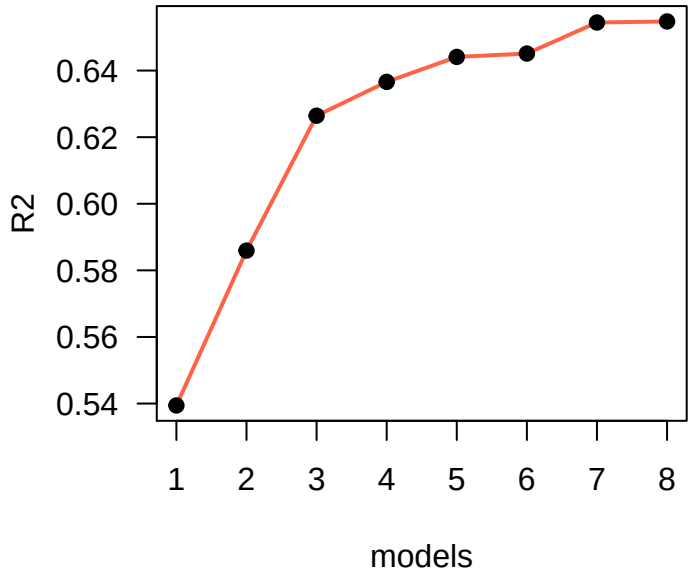
- Fit a model with `lpsa` as the response and `lcavol`as the predictor. Record the residual standard error and the R^2.
- Add `lweight`, `svi`, `lbph`, `age`, `lcp`, `pgg45` and `gleason` to the model one at a time. For each model record the residual standard error and the R^2.

- Make plots of the trends for each of these two statistics.

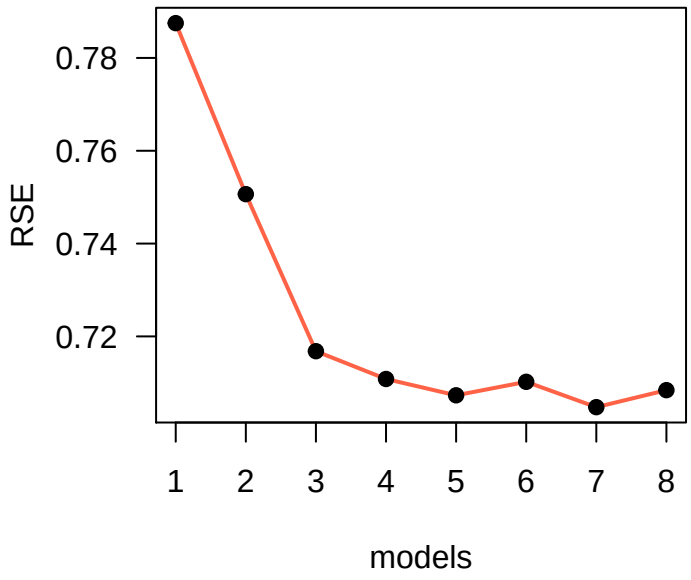
```
preds <- c('lcavol', 'lweight', 'svi', 'lbph', 'age', 'lcp', 'pgg45', 'gleason')
y <- dat$lpsa
r2 <- rep(0, length(preds))
rse <- rep(0, length(preds))

for (j in 1:length(preds)) {
  X <- as.matrix(cbind(1, dat[ ,preds[1:j]]))
  fit <- ols_fit(X, y)
  rse[j] <- RSE(fit)
  r2[j] <- R2(fit)
}
```

```
# Trend of R2 (data prostate)
plot(1:length(preds), r2, type = 'l', col = 'tomato', lwd = 2,
     las = 1, xlab = 'models', ylab = 'R2')
points(1:length(preds), r2, pch = 19)
```




```
# Trend of RSE (data prostate)
plot(1:length(preds), rse, type = 'l', col = 'tomato', lwd = 2,
     las = 1, xlab = 'models', ylab = 'RSE')
points(1:length(preds), rse, pch = 19)
```



Problem 7 (20 pts)

This question involves the use of multiple linear regression on the `Auto` data set. The associated file is in the website of the textbook "An Introduction to Statistical Learning":

http://www-bcf.usc.edu/~gareth/ISL/Auto.data

- Parts (a) - (d) are worth 10 pts
- Parts (e) - (f) are worth 10 pts

The data in `Auto.data` codifies missing values with question marks `"?"`. The code below assumes that there is a copy of `Auto.data` in your working directory. To have more control on the data type for each column, I'm using the argument `colClasses`. Of course, you can read this data in R in other ways.

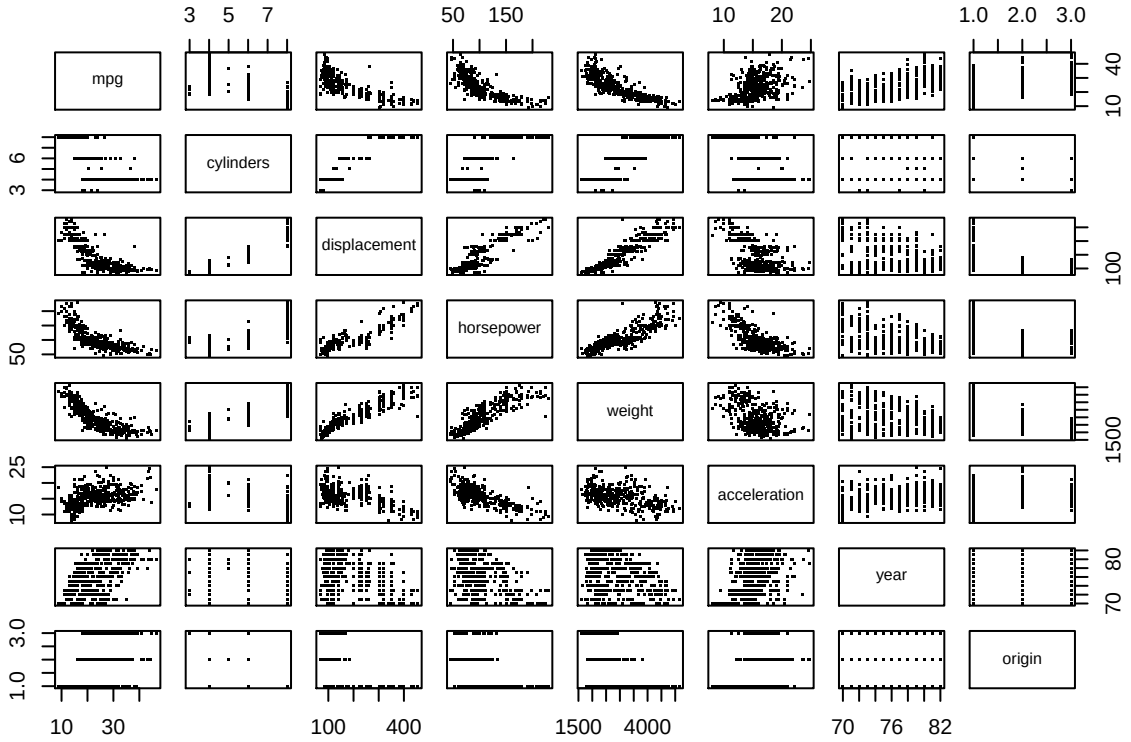
```
# read data
Auto <- read.table(
  "Auto.data",
  header = TRUE,
  na.strings = "?",
  colClasses = c(
    mpg = "real",
    cylinders = "real",
    displacement = "real",
    horsepower = "real",
    weight = "real",
    acceleration = "real",
    year = "integer",
    origin = "integer",
```

```
    name = "character"
  ))
```

a. Produce a scatterplot matrix which inlcudes all of the variables in the data set.

The function `pairs()` allows you to get a scatterplot matrix. Another interesting option is the function `ggpairs()` from the package `"GGally"`.

```
# scatterplot matrix
pairs(Auto[, -9], pch = ".")
```



b. Compute the matrix of correlations between the variables using the function `cor()`. You will need to exclude the `name` variable, which is qualitative.

The `cor()` function has the `use =` argument which allows you to remove those observations containing missing values. You can also use `na.omit()` to explicitly remove missing values, and then pass your data to `cor()`

```
# matrix of correlations (excluding NAs)
R <- cor(Auto[ ,-9], use = "complete.obs")
round(R, 3)
```

```
##                mpg cylinders displacement horsepower weight acceleration
## mpg          1.000    -0.778       -0.805     -0.778 -0.832        0.423
## cylinders   -0.778     1.000        0.951      0.843  0.898       -0.505
## displacement -0.805     0.951        1.000      0.897  0.933       -0.544
## horsepower  -0.778     0.843        0.897      1.000  0.865       -0.689
```

```
## weight       -0.832     0.898        0.933      0.865  1.000       -0.417
## acceleration  0.423    -0.505       -0.544     -0.689 -0.417        1.000
## year          0.581    -0.346       -0.370     -0.416 -0.309        0.290
## origin        0.565    -0.569       -0.615     -0.455 -0.585        0.213
##                year origin
## mpg           0.581  0.565
## cylinders    -0.346 -0.569
## displacement -0.370 -0.615
## horsepower   -0.416 -0.455
## weight       -0.309 -0.585
## acceleration  0.290  0.213
## year          1.000  0.182
## origin        0.182  1.000
```

c. Use the `lm()` function to perform a multiple linear regression with `mpg` as the response and all other variables except `name` as the predictors.

```
fit1 <- lm(mpg ~ . - name, data = Auto)
fit1
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Coefficients:
##  (Intercept)     cylinders  displacement    horsepower        weight
##   -17.218435     -0.493376      0.019896     -0.016951     -0.006474
## acceleration          year        origin
##     0.080576      0.750773      1.426140
```

Use the `summary()` function to print the results.

```
# summary
fit1_sum <- summary(fit1)
fit1_sum
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
```

```
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

Comment on the output. For instance:

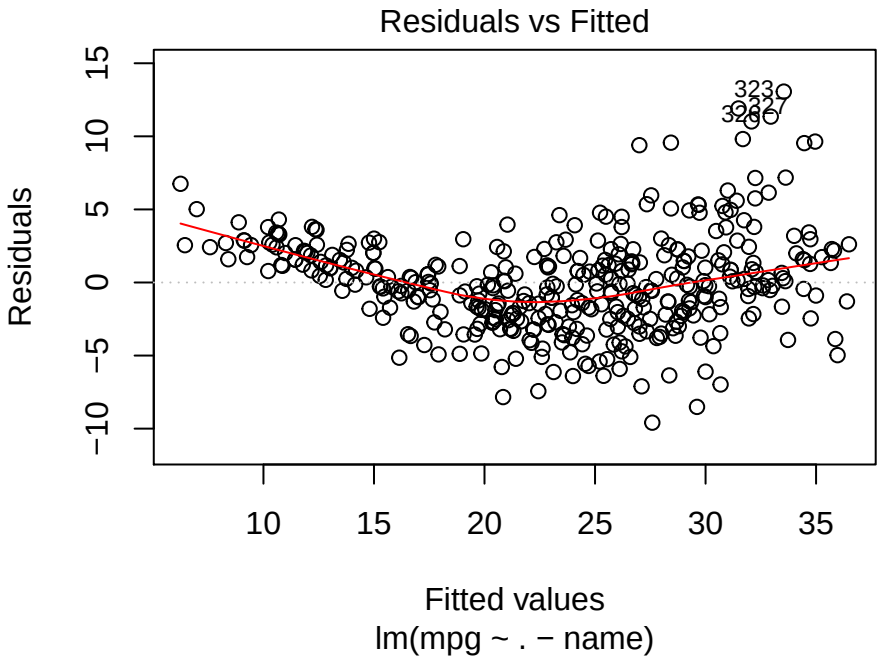
- Is there a relationship between the predictors and the response?
- Which predictors appear to have a statistically significant relationship to the response?
- What does the coefficient for the `year` variable suggest?

The predictors that appear to have a statistically significant relationship are `displacement`, `weight`, `year`, and `origin`.

The coefficient for the `year`, suggests that for every extra year, the fuel consumption of the cars went up 0.751 mpg by on average, assuming the rest of predictors remained constant.

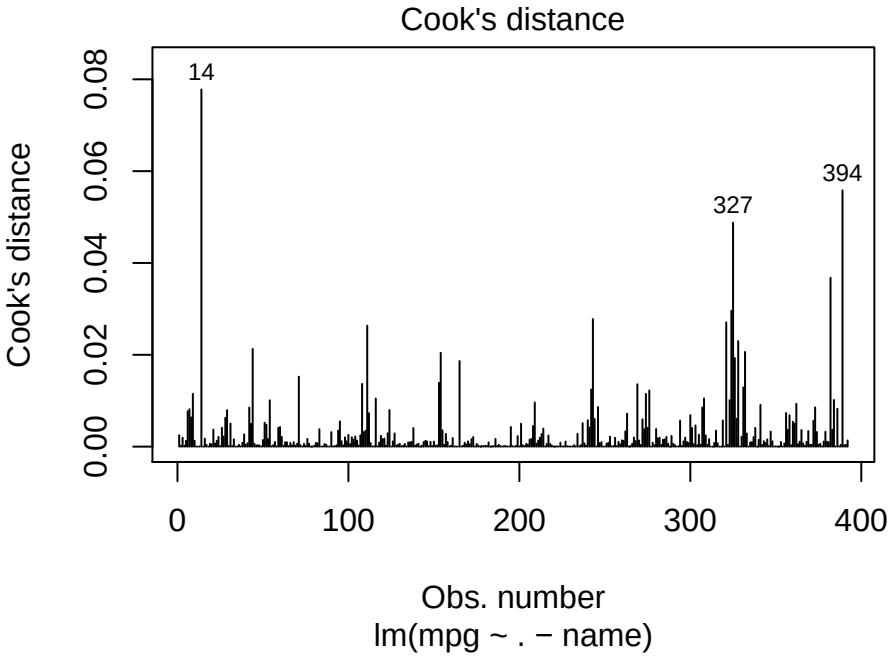
d. Use the `plot()` function to produce diagnostic plots of the linear regression fit. Comment on any problems you see with the fit. Do the residual plots suggest any unusually large outliers? Does the leverage plot identify any observations with unusual high leverage?

```
# residual plot
plot(fit1, which = 1)
```



The residual plot shows a lack of homoscedasticity; notice how the cloud of points widdens for larger values of `mpg`.

```
# residual plot
plot(fit1, which = 4)
```



e. Use the * and : symbols to fit a linear regression model with an interaction effect. Does the interaction that you choose appear to be statistically significant?

Part e) is admittedly open-ended, and it is more intended as an exercise to play with interaction terms, and experiment with the `lm()` function. Depending on what variables you play with, you may get statistically significant results or not.

There are two ways to include interaction terms in a linear model using the `lm()` function. The syntax `displacement:cylinders` tells R to include an interaction term between `displacement` and `cylinders`. The syntax `displacement*cylinders` simultaneously includes `displacement`, `cylinders`, and the interaction term `displacement` × `cylinders` as predictors; it is a shorthand for `displacement + cylinders + displacement:cylinders`

```
# interaction with *
fit_star <- lm(mpg ~ displacement*cylinders, data = Auto)
fit_star
```

```
##
## Call:
## lm(formula = mpg ~ displacement * cylinders, data = Auto)
##
## Coefficients:
##            (Intercept)            displacement               cylinders
##               48.43553                -0.13682                -2.42262
## displacement:cylinders
##                0.01207
```

```
# interaction with :
fit_colon <- lm(mpg ~ displacement:cylinders, data = Auto)
fit_colon
```

```
##
## Call:
## lm(formula = mpg ~ displacement:cylinders, data = Auto)
##
## Coefficients:
##            (Intercept)  displacement:cylinders
##              31.039795               -0.006143
```

f. Fit another linear regression model by trying a few different transformations of the variables, such as $log(X)$, $\sqrt{X}$, X^2. Comment on your findings.

This is another exercise (from ISL) in which the idea is to play with `lm()` and spend some time experimenting with various transformations. Because each of you may come up with your own model, we will grade this problem based on effort.