APPLIED REGRESSION ANALYSIS

Ch2: R Lab

Gengxin li

Department of Mathematics and Statistics
University of Michigan Dearborn

Exercise: **The gasoline mileage performance of 32 different automobiles.**

TABLE B.3 **Gasoline Mileage Performance for 32 Antomobiles**

Automobile	y	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
Apollo	18.90	350	165	260	8.0:1	2.56:1	4	3	200.3	69.9	3910	A
Omega	17.00	350	170	275	8.5:1	2.56:1	4	3	199.6	72.9	2860	A
Nova	20.00	250	105	185	8.25:1	2.73:1	1	3	196.7	72.2	3510	A
Monarch	18.25	351	143	255	8.0:1	3.00:1	2	3	199.9	74.0	3890	A
Duster	20.07	225	95	170	8.4:1	2.76:1	1	3	194.1	71.8	3365	M
Jenson Conv.	11.2	440	215	330	8.2:1	2.88:1	4	3	184.5	69	4215	A
Skyhawk	22.12	231	110	175	8.0:1	2.56:1	2	3	179.3	65.4	3020	A
Monza	21.47	262	110	200	8.5:1	2.56:1	2	3	179.3	65.4	3180	A
Scirocco	34.70	89.7	70	81	8.2:1	3.90:1	2	4	155.7	64	1905	M
Corolla SR-5	30.40	96.9	75	83	9.0:1	4.30:1	2	5	165.2	65	2320	M
Camaro	16.50	350	155	250	8.5:1	3.08:1	4	3	195.4	74.4	3885	A
Datsun B210	36.50	85.3	80	83	8.5:1	3.89:1	2	4	160.6	62.2	2009	M
Capri II	21.50	171	109	146	8.2:1	3.22:1	2	4	170.4	66.9	2655	M
Pacer	19.70	258	110	195	8.0:1	3.08:1	1	3	171.5	77	3375	A
Babcat	20.30	140	83	109	8.4:1	3.40:1	2	4	168.8	69.4	2700	M
Granada	17.80	302	129	220	8.0:1	3.0:1	2	3	199.9	74	3890	A
Eldorado	14.39	500	190	360	8.5:1	2.73:1	4	3	224.1	79.8	5290	A
Imperial	14.89	440	215	330	8.2:1	2.71:1	4	3	231.0	79.7	5185	A
Nova LN	17.80	350	155	250	8.5:1	3.08:1	4	3	196.7	72.2	3910	A
Valiant	16.41	318	145	255	8.5:1	2.45:1	2	3	197.6	71	3660	A
Starfire	23.54	231	110	175	8.0:1	2.56:1	2	3	179.3	65.4	3050	A
Cordoba	21.47	360	180	290	8.4:1	2.45:1	2	3	214.2	76.3	4250	A
Trans AM	16.59	400	185	NA	7.6:1	3.08:1	4	3	196	73	3850	A
Corolla E-5	31.90	96.9	75	83	9.0:1	4.30:1	2	5	165.2	61.8	2275	M
Astre	29.40	140	86	NA	8.0:1	2.92:1	2	4	176.4	65.4	2150	M
Mark IV	13.27	460	223	366	8.0:1	3.00:1	4	3	228	79.8	5430	A
Celica GT	23.90	133.6	96	120	8.4:1	3.91:1	2	5	171.5	63.4	2535	M
Charger SE	19.73	318	140	255	8.5:1	2.71:1	2	3	215.3	76.3	4370	A
Cougar	13.90	351	148	243	8.0:1	3.25:1	2	3	215.5	78.5	4540	A
Elite	13.27	351	148	243	8.0:1	3.26:1	2	3	216.1	78.5	4715	A
Matador	13.77	360	195	295	8.25:1	3.15:1	4	3	209.3	77.4	4215	A
Corvette	16.50	350	165	255	8.5:1	2.73:1	4	3	185.2	69	3660	A

y: Miles/gallon
x_1: Displacement (cubic in.)
x_2: Horsepower (ft-lb)
x_3: Torqne (ft-lb)
x_4: Compression ratio
x_5: Rear axle ratio
x_6: Carburetor (barrels)
x_7: No. of transmission speeds
x_8: Overall length (in.)
x_9: Width (in.)
x_{10}: Weight (lb)
x_{11}: Type of transmission (A automatic; M manual)
Source: *Motor Trend*, 1975.

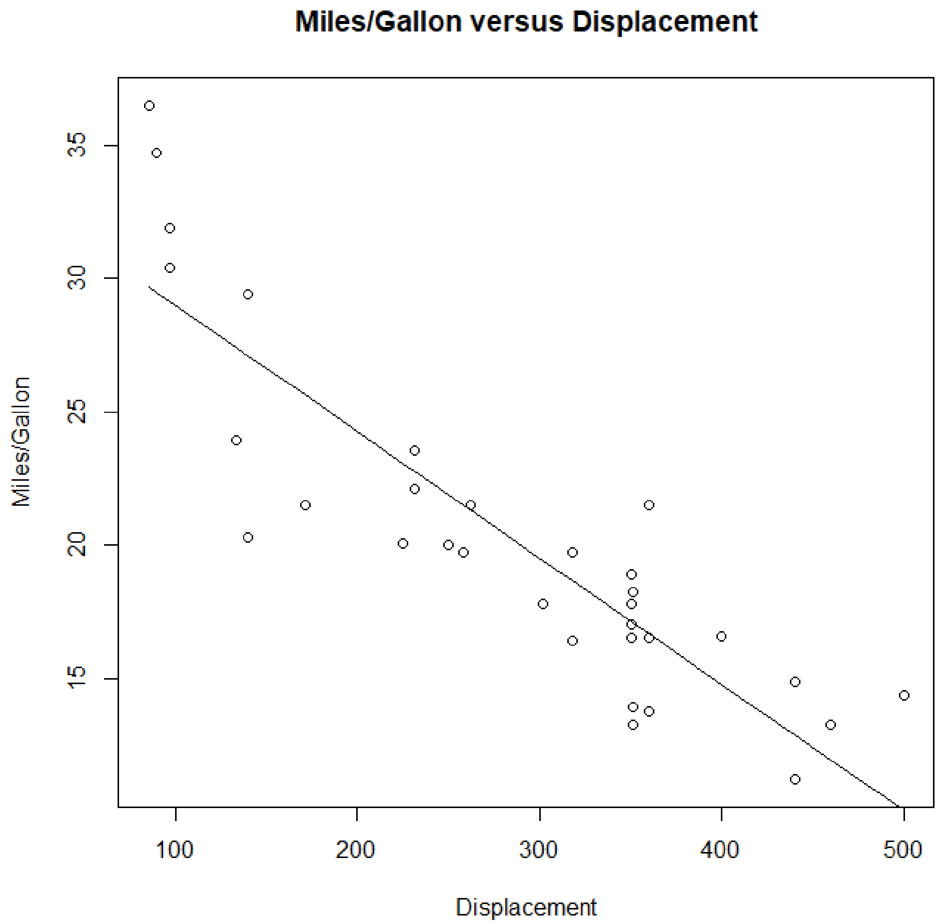
```
> ### 32 automobiles
> ### load the antomobile data in R
> setwd('D:\\bf\\UMD\\teaching\\STT530\\exercise')
> dir()
[1] "chap2_exercise_1.csv"   "Exercise_chapter2.docx"
> auto <- read.csv("chap2_exercise_1.csv", header=T)
> attach(auto)
> dim(auto)
[1] 32 12
> auto
       y    x1  x2  x3   x4   x5 x6 x7    x8   x9  x10 x11
1  18.90 350.0 165 260 8.00 2.56  4  3 200.3 69.9 3910   1
2  17.00 350.0 170 275 8.50 2.56  4  3 199.6 72.9 3860   1
3  20.00 250.0 105 185 8.25 2.73  1  3 196.7 72.2 3510   1
4  18.25 351.0 143 255 8.00 3.00  2  3 199.9 74.0 3890   1
5  20.07 225.0  95 170 8.40 2.76  1  3 194.1 71.8 3365   0
6  11.20 440.0 215 330 8.20 2.88  4  3 184.5 69.0 4215   1
7  22.12 231.0 110 175 8.00 2.56  2  3 179.3 65.4 3020   1
8  21.47 262.0 110 200 8.50 2.56  2  3 179.3 65.4 3180   1
9  34.70  89.7  70  81 8.20 3.90  2  4 155.7 64.0 1905   0
10 30.40  96.9  75  83 9.00 4.30  2  5 165.2 65.0 2320   0
11 16.50 350.0 155 250 8.50 3.08  4  3 195.4 74.4 3885   1
12 36.50  85.3  80  83 8.50 3.89  2  4 160.6 62.2 2009   0
13 21.50 171.0 109 146 8.20 3.22  2  4 170.4 66.9 2655   0
14 19.70 258.0 110 195 8.00 3.08  1  3 171.5 77.0 3375   1
15 20.30 140.0  83 109 8.40 3.40  2  4 168.8 69.4 2700   0
16 17.80 302.0 129 220 8.00 3.00  2  3 199.9 74.0 3890   1
17 14.39 500.0 190 360 8.50 2.73  4  3 224.1 79.8 5290   1
18 14.89 440.0 215 330 8.20 2.71  4  3 231.0 79.7 5185   1
19 17.80 350.0 155 250 8.50 3.08  4  3 196.7 72.2 3910   1
20 16.41 318.0 145 255 8.50 2.45  2  3 197.6 71.0 3660   1
21 23.54 231.0 110 175 8.00 2.56  2  3 179.3 65.4 3050   1
22 21.47 360.0 180 290 8.40 2.45  2  3 214.2 76.3 4250   1
23 16.59 400.0 185  NA 7.60 3.08  4  3 196.0 73.0 3850   1
24 31.90  96.9  75  83 9.00 4.30  2  5 165.2 61.8 2275   0
25 29.40 140.0  86  NA 8.00 2.92  2  4 176.4 65.4 2150   0
26 13.27 460.0 223 366 8.00 3.00  4  3 228.0 79.8 5430   1
27 23.90 133.6  96 120 8.40 3.91  2  5 171.5 63.4 2535   0
28 19.73 318.0 140 255 8.50 2.71  2  3 215.3 76.3 4370   1
29 13.90 351.0 148 243 8.00 3.25  2  3 215.5 78.5 4540   1
30 13.27 351.0 148 243 8.00 3.26  2  3 216.1 78.5 4715   1
31 13.77 360.0 195 295 8.25 3.15  4  3 209.3 77.4 4215   1
32 16.50 360.0 165 255 8.50 2.73  4  3 185.2 69.0 3660   1
```

The gasoline mileage performance of 32 different automobiles.

a. Fit a simple linear regression model relating gasoline mileage y (miles per gallon) to engine displacement x l (cubic inches).

b. Draw the scatter plot between gasoline mileage y and engine displacement x1, and draw the fitted the simple linear regression line

```
> ### a. Fit a simple linear regression model relating gasoline mileage y (miles per gallon) to
> ###    engine displacement x 1 (cubic inches).
> model <- lm(y ~ x1, data=auto)
> summary(model)

Call:
lm(formula = y ~ x1, data = auto)

Residuals:
    Min      1Q  Median      3Q     Max
-6.7923 -1.9752  0.0044  1.7677  6.8171

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 33.722677   1.443903   23.36  < 2e-16 ***
x1          -0.047360   0.004695  -10.09 3.74e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.065 on 30 degrees of freedom
Multiple R-squared:  0.7723,    Adjusted R-squared:  0.7647
F-statistic: 101.7 on 1 and 30 DF,  p-value: 3.743e-11

>
> ### b. Draw the scatter plot between gasoline mileage y and engine displacement x1,
> ###    and draw the fitted the simple linear regression line
> plot(x1, y,
+      xlab="Displacement", ylab="Miles/Gallon",
+      main="Miles/Gallon versus Displacement",
+      panel.last = lines(sort(x1), fitted(model)[order(x1)]))
```



$$\hat{y} = 33.72 - 0.047x1$$

c. Test the hypothesis H 0 : β 1 = 0.

d. Find 95% Confidence Interval for β_0 *and* β_1

```
> ### c. Test the hypothesis H 0 : ? 1 = 0.
> model <- lm(y ~ x1, data=auto)
> summary(model)

Call:
lm(formula = y ~ x1, data = auto)

Residuals:
    Min      1Q  Median      3Q     Max
-6.7923 -1.9752  0.0044  1.7677  6.8171

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 33.722677   1.443903   23.36  < 2e-16 ***
x1          -0.047360   0.004695  -10.09 3.74e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.065 on 30 degrees of freedom
Multiple R-squared:  0.7723,	Adjusted R-squared:  0.7647
F-statistic: 101.7 on 1 and 30 DF,  p-value: 3.743e-11

> ### rejection region
> qt(1-0.05/2,30)
[1] 2.042272
>
>
> ### d. Find 95% Confidence Interval for ? 0 and ? 1. Find 95% C.I for ? 1.
> confint(model)
                  2.5 %      97.5 %
(Intercept) 30.77383383 36.67151954
x1          -0.05694883 -0.03777032
> confint(model,"x1")
         2.5 %      97.5 %
x1 -0.05694883 -0.03777032
>
```

Conclusion:

c. Hypothesis test

i. hypothesis $\begin{cases} H_0\colon \beta_1 = 0 \\ H_a\colon \beta_1 \neq 0 \end{cases}$.

ii. Test statistic: $t_{obs} = \dfrac{b_1 - \beta_{10}}{\sqrt{\dfrac{s^2}{S_{xx}}}} = \dfrac{-0.04736 - 0}{0.004695} = -10.09$

iii. Rejection region: $t > t_{\frac{\alpha}{2},n-2} = 2.042$ *or* $t < -t_{\frac{\alpha}{2},n-2} = -2.042$

iv. Since t_{obs} is in the rejection region, we reject the H_0. It indicates that x1 (displacement) is negatively linear related to y (miles/gallon)

d. Find 95% Confidence Interval for β_0 *and* β_1, and 95% Confidence Interval for β_1

95% CI for β_0 is (30.77, 36.67);

95% CI for β_1 is (-0.057, -0.038);

Since 0 is outside of C.I, it indicates that we reject the H_0

e. Find **80%** Confidence Interval for β_0 *and* β_1

```
> ### e. Find 80% Confidence Interval for β 0 and β 1
> help(confint)
> confint(model, level=0.8)
                   10 %        90 %
(Intercept) 31.83056480 35.61478857
x1          -0.05351248 -0.04120667
```

80% Confidence Interval for β_0 is (31.83, 35.61)
80% Confidence Interval for β_1 is (-0.054, -0.041)

confint {stats} R Documentation

Confidence Intervals for Model Parameters

Description

Computes confidence intervals for one or more parameters in a fitted model. There is a default and a method for objects inheriting from class "lm".

Usage

```
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`
: a fitted model object.

`parm`
: a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.

`level`
: the confidence level required.

`...`
: additional argument(s) for methods.

Details

`confint` is a generic function. The default method assumes asymptotic normality, and needs suitable coef and vcov methods to be available. The default method can be called directly for comparison with other methods.

For objects of class "lm" the direct formulae based on *t* values are used.

There are stub methods in package **stats** for classes "glm" and "nls" which call those in package MASS (if installed): if the MASS namespace has been loaded, its methods will be used directly. (Those methods are based on profile likelihood.)

f. Calculate R^2, and calculate the **correlation** between miles/gallon (y) and engine displacement (x1).

```
> ### f. Calculate R^2 and calculate the correlation between y and x1.
> model <- lm(y ~ x1, data=auto)
> summary(model)

Call:
lm(formula = y ~ x1, data = auto)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.7923 -1.9752  0.0044  1.7677  6.8171 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 33.722677   1.443903   23.36  < 2e-16 ***
x1          -0.047360   0.004695  -10.09 3.74e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.065 on 30 degrees of freedom
Multiple R-squared:  0.7723,	Adjusted R-squared:  0.7647 
F-statistic: 101.7 on 1 and 30 DF,  p-value: 3.743e-11

> 
> cor(y,x1)
[1] -0.8787896
```

Conclusion: R^2=0.7723, and the correlation between y and x1 is r=-0.8788

```
> cor(y,x1)^2
[1] 0.7722712
```

$$r^2 = -0.8788^2 = 0.7723 = R^2$$

g. Calculate **sum of squared errors (SSE)**, **mean square error (MSE)**, and **regression (or residual) standard error (s)**.

```
> # g. find SSE, MSE and s
> sum(residuals(model)^2) ##SSE=281.42
[1] 281.8244
>
> n=32
> sum(residuals(model)^2)/(n-2) ##MSE=s^2=9.39
[1] 9.394146
>
> n=32
> sqrt(sum(residuals(model)^2)/(n-2)) ## s=3.06
[1] 3.064987
```

```
> model <- lm(y ~ x1, data=auto)
> summary(model)

Call:
lm(formula = y ~ x1, data = auto)

Residuals:
    Min      1Q  Median      3Q     Max
-6.7923 -1.9752  0.0044  1.7677  6.8171

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 33.722677   1.443903   23.36  < 2e-16 ***
x1          -0.047360   0.004695  -10.09 3.74e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.065 on 30 degrees of freedom
Multiple R-squared:  0.7723,	Adjusted R-squared:  0.7647
F-statistic: 101.7 on 1 and 30 DF,  p-value: 3.743e-11
```

The LSE estimator of σ^2 is $s^2 = MSE = 3.065^2 = 9.39$

h. Find a 95% PI on the **mean (average) gasoline mileage** if the engine displacement is 275 in.

i. Suppose that we wish to predict **the gasoline mileage** obtained from a car with a 275 in engine. Give a point estimate of mileage. Find a 95% prediction interval on the mileage.

j. Compare the two intervals obtained in parts h and i. Explain the difference between them. Which one is wider, and why?

```
> ### h.Find a 95% CI on the mean gasoline mileage if the engine displacement is 275 in.
> predict(model, newdata=data.frame(x1=275), interval="confidence")# CI for E(y|x0)
       fit      lwr      upr
1 20.69879 19.58807 21.80952
>
>
> ### i.Suppose that we wish to predict the gasoline mileage obtained from a car with a 275 in engine
> ### displacement. Give a point estimate of mileage. Find a 95% prediction interval on the mileage.
> predict(model, newdata=data.frame(x1=275), interval="prediction")# PI for y|x0
       fit      lwr      upr
1 20.69879 14.34147 27.05611
```

Conclusion:

h. The 95% confidence interval on the mean mileage is (19.59, 21.81).

i. The point estimate of mileage is 20.69879.

The 95% prediction interval on the mileage is (14.34, 27.06).

j. Compare the above two intervals obtained in parts h and i, the interval of mean mileage gives us a narrower interval. The PI of mileage gives a wider interval.

ii. predictive variance. $\hat{Y}_0 = b_0 + b_1 x_0$. $Y_0 = \beta_0 + \beta_1 x_0 + \varepsilon = Y | x_0$

$$Var(\hat{y}_0) = E(\hat{Y}_0 - Y_0)^2 = E[(b_0 - \beta_0) + (b_1 - \beta_1)x_0 - \varepsilon]^2$$

$$= \cancel{Var(b_0)+} E[(b_0 - \beta_0) + (x_0 - \bar{x})(b_1 - \beta_1) + \bar{x}(b_1 - \beta_1) - \varepsilon]^2$$

$$= Var(b_0) + (x_0 - \bar{x})^2 Var(b_1) + \bar{x}^2 Var(b_1) + Var(\varepsilon) + 2(x_0 - \bar{x}) Cov(b_0, b_1)$$
$$+ 2\bar{x}\, Cov(b_0, b_1) + \cancel{2\bar{x}}\, 2(x_0 - \bar{x})\, Var(b_1) \cdot \bar{x}$$

= rest of the terms are 0 assuming the independence of ε_0 with $\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n$.

$$= Var(b_0) + [(x_0 - \bar{x})^2 + \bar{x}^2 + 2(x_0 - \bar{x})\bar{x}] Var(b_1) + Var(\varepsilon) + [2(x_0 - \bar{x}) + 2\bar{x}] Cov(b_0,$$

$$= Var(b_0) + [(x_0 - \bar{x} + \bar{x})^2] Var(b_1) + Var(\varepsilon) + 2x_0\, Cov(b_0, b_1)$$

$$= Var(b_0) + x_0^2 Var(b_1) + Var(\varepsilon) + 2x_0\, Cov(b_0, b_1)$$

$$= \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}}\right] + \frac{x_0^2}{S_{xx}} \cdot \sigma^2 + \sigma^2 + 2x_0\, Cov(\bar{y} - b_1\bar{x}, b_1)$$

$$= \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} + \frac{x_0^2}{S_{xx}} + 1\right] - 2x_0\bar{x}\, Var(b_1)$$

$$= \sigma^2 \left[1 + \frac{1}{n} + \frac{\bar{x}^2 + x_0^2}{S_{xx}}\right] - 2x_0\bar{x} \cdot \frac{\sigma^2}{S_{xx}}$$

$$= \sigma^2 \left[1 + \frac{1}{n} + \frac{x_0^2 - 2x_0\bar{x} + \bar{x}^2}{S_{xx}}\right]$$

$$= \sigma^2 \left[1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{S_{xx}}\right]$$

Python version

The gasoline mileage performance of 32 different automobiles.

TABLE B.3 Gasoline Mileage Performance for 32 Antomobiles

Automobile	y	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
Apollo	18.90	350	165	260	8.0:1	2.56:1	4	3	200.3	69.9	3910	A
Omega	17.00	350	170	275	8.5:1	2.56:1	4	3	199.6	72.9	2860	A
Nova	20.00	250	105	185	8.25:1	2.73:1	1	3	196.7	72.2	3510	A
Monarch	18.25	351	143	255	8.0:1	3.00:1	2	3	199.9	74.0	3890	A
Duster	20.07	225	95	170	8.4:1	2.76:1	1	3	194.1	71.8	3365	M
Jenson Conv.	11.2	440	215	330	8.2:1	2.88:1	4	3	184.5	69	4215	A
Skyhawk	22.12	231	110	175	8.0:1	2.56:1	2	3	179.3	65.4	3020	A
Monza	21.47	262	110	200	8.5:1	2.56:1	2	3	179.3	65.4	3180	A
Scirocco	34.70	89.7	70	81	8.2:1	3.90:1	2	4	155.7	64	1905	M
Corolla SR-5	30.40	96.9	75	83	9.0:1	4.30:1	2	5	165.2	65	2320	M
Camaro	16.50	350	155	250	8.5:1	3.08:1	4	3	195.4	74.4	3885	A
Datsun B210	36.50	85.3	80	83	8.5:1	3.89:1	2	4	160.6	62.2	2009	M
Capri II	21.50	171	109	146	8.2:1	3.22:1	2	4	170.4	66.9	2655	M
Pacer	19.70	258	110	195	8.0:1	3.08:1	1	3	171.5	77	3375	A
Babcat	20.30	140	83	109	8.4:1	3.40:1	2	4	168.8	69.4	2700	M
Granada	17.80	302	129	220	8.0:1	3.0:1	2	3	199.9	74	3890	A
Eldorado	14.39	500	190	360	8.5:1	2.73:1	4	3	224.1	79.8	5290	A
Imperial	14.89	440	215	330	8.2:1	2.71:1	4	3	231.0	79.7	5185	A
Nova LN	17.80	350	155	250	8.5:1	3.08:1	4	3	196.7	72.2	3910	A
Valiant	16.41	318	145	255	8.5:1	2.45:1	2	3	197.6	71	3660	A
Starfire	23.54	231	110	175	8.0:1	2.56:1	2	3	179.3	65.4	3050	A
Cordoba	21.47	360	180	290	8.4:1	2.45:1	2	3	214.2	76.3	4250	A
Trans AM	16.59	400	185	NA	7.6:1	3.08:1	4	3	196	73	3850	A
Corolla E-5	31.90	96.9	75	83	9.0:1	4.30:1	2	5	165.2	61.8	2275	M
Astre	29.40	140	86	NA	8.0:1	2.92:1	2	4	176.4	65.4	2150	M
Mark IV	13.27	460	223	366	8.0:1	3.00:1	4	3	228	79.8	5430	A
Celica GT	23.90	133.6	96	120	8.4:1	3.91:1	2	5	171.5	63.4	2535	M
Charger SE	19.73	318	140	255	8.5:1	2.71:1	2	3	215.3	76.3	4370	A
Cougar	13.90	351	148	243	8.0:1	3.25:1	2	3	215.5	78.5	4540	A
Elite	13.27	351	148	243	8.0:1	3.26:1	2	3	216.1	78.5	4715	A
Matador	13.77	360	195	295	8.25:1	3.15:1	4	3	209.3	77.4	4215	A
Corvette	16.50	350	165	255	8.5:1	2.73:1	4	3	185.2	69	3660	A

y: Miles/gallon
x_1: Displacement (cubic in.)
x_2: Horsepower (ft-lb)
x_3: Torqne (ft-lb)
x_4: Compression ratio
x_5: Rear axle ratio
x_6: Carburetor (barrels)
x_7: No. of transmission speeds
x_8: Overall length (in.)
x_9: Width (in.)
x_{10}: Weight (lb)
x_{11}: Type of transmission (A automatic; M manual)

Source: *Motor Trend*, 1975.

```python
# supp_lecture3
# load chapter 2 exercise data set
auto = pd.read_csv('chap2_exercise_1.csv')
print(auto.shape)
print(auto)
```

```
        y     x1   x2     x3    x4    x5  x6  x7     x8    x9   x10  x11
0   18.90  350.0  165  260.0  8.00  2.56   4   3  200.3  69.9  3910    1
1   17.00  350.0  170  275.0  8.50  2.56   4   3  199.6  72.9  3860    1
2   20.00  250.0  105  185.0  8.25  2.73   1   3  196.7  72.2  3510    1
3   18.25  351.0  143  255.0  8.00  3.00   2   3  199.9  74.0  3890    1
4   20.07  225.0   95  170.0  8.40  2.76   1   3  194.1  71.8  3365    0
5   11.20  440.0  215  330.0  8.20  2.88   4   3  184.5  69.0  4215    1
6   22.12  231.0  110  175.0  8.00  2.56   2   3  179.3  65.4  3020    1
7   21.47  262.0  110  200.0  8.50  2.56   2   3  179.3  65.4  3180    1
8   34.70   89.7   70   81.0  8.20  3.90   2   4  155.7  64.0  1905    0
9   30.40   96.9   75   83.0  9.00  4.30   2   5  165.2  65.0  2320    0
10  16.50  350.0  155  250.0  8.50  3.08   4   3  195.4  74.4  3885    1
11  36.50   85.3   80   83.0  8.50  3.89   2   4  160.6  62.2  2009    0
12  21.50  171.0  109  146.0  8.20  3.22   2   4  170.4  66.9  2655    0
13  19.70  258.0  110  195.0  8.00  3.08   1   3  171.5  77.0  3375    1
14  20.30  140.0   83  109.0  8.40  3.40   2   4  168.8  69.4  2700    0
15  17.80  302.0  129  220.0  8.00  3.00   2   3  199.9  74.0  3890    1
16  14.39  500.0  190  360.0  8.50  2.73   4   3  224.1  79.8  5290    1
```

The gasoline mileage performance of 32 different automobiles.

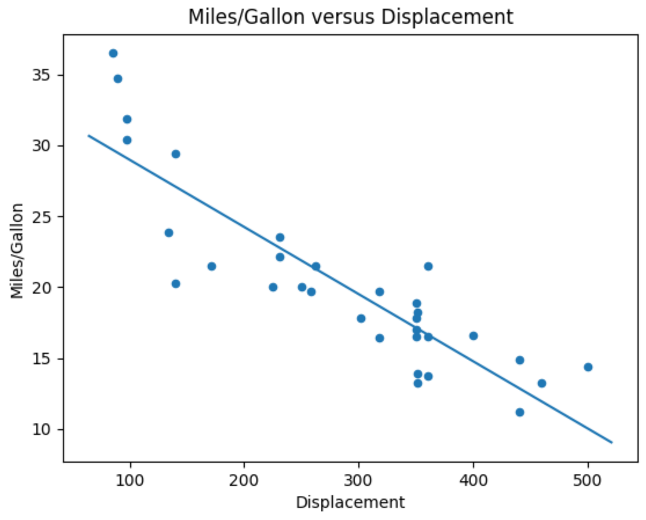
a. Fit a simple linear regression model relating gasoline mileage y (miles per gallon) to engine displacement x l (cubic inches).

b. Draw the scatter plot between gasoline mileage y and engine displacement x1, and draw the fitted the simple linear regression line

```
# fit a simple linear regression model
X = MS(['x1']).fit_transform(auto)
y = auto['y']
model1 = sm.OLS(y, X)
results1 = model1.fit()
print(summarize(results1))
```

	coef	std err	t	P>|t|
intercept	33.7227	1.444	23.355	0.0
x1	-0.0474	0.005	-10.086	0.0

```
def abline(ax, b, m):  1 usage
    "Add a line with slope m and intercept b to ax"
    xlim = ax.get_xlim()
    ylim = [m * xlim[0] + b, m * xlim[1] + b]
    ax.plot(xlim , ylim)

ax = auto.plot.scatter('x1', 'y')
abline(ax, results1.params[0], results1.params[1])
ax.set_xlabel('Displacement')
ax.set_ylabel('Miles/Gallon')
ax.set_title('Miles/Gallon versus Displacement')
plt.show()
```



c. Test the hypothesis H 0 : β 1 = 0.

d. Find 95% Confidence Interval for β_0 *and* β_1

```
# fit a simple linear regression model
X = MS(['x1']).fit_transform(auto)
y = auto['y']
model1 = sm.OLS(y, X)
results1 = model1.fit()
print(summarize(results1))
              coef  std err       t  P>|t|
intercept  33.7227    1.444  23.355    0.0
x1         -0.0474    0.005 -10.086    0.0
```

```
# rejection region
# for t test
import scipy.stats
import pandas as pd
auto = pd.read_csv('chap2_exercise_1.csv')
t_value=scipy.stats.t.ppf(q=1 - 0.05/2, df=auto.shape[0]-2)
print('t critical value:',t_value)
t critical value: 2.0422724563012373
```

```
# CI
CI = results1.conf_int(alpha=0.05)
print(CI)
                   0         1
intercept  30.773834  36.67152
x1         -0.056949  -0.03777
```

Conclusion:

c. Hypothesis test

i. hypothesis $\begin{cases} H_0\colon \beta_1 = 0 \\ H_a\colon \beta_1 \neq 0 \end{cases}$.

ii. Test statistic: $t_{obs} = \dfrac{b_1 - \beta_{10}}{\sqrt{\dfrac{s^2}{S_{xx}}}} = \dfrac{-0.04736-0}{0.004695} = -10.09$

iii. Rejection region: t > $t_{\frac{\alpha}{2},n-2} = 2.042$ *or* t < $-t_{\frac{\alpha}{2},n-2} = -2.042$

iv. Since t_{obs} is in the rejection region, we reject the H_0. It indicates that x1 (displacement) is negatively linear related to y (miles/gallon)

d. Find 95% Confidence Interval for β_0 *and* β_1, and 95% Confidence Interval for β_1

95% CI for β_0 is (30.77, 36.67);

95% CI for β_1 is (-0.057, -0.038);

e. Find **80%** Confidence Interval for β_0 *and* β_1

```
# Confidence Interval (80%)
CI = results1.conf_int(alpha=0.20)
print(CI)
```

```
                   0          1
intercept  31.830565  35.614789
x1         -0.053512  -0.041207
```

80% Confidence Interval for β_0 is (31.83, 35.61)
80% Confidence Interval for β_1 is (-0.054, -0.041)

f. Calculate R^2, and calculate the **correlation** between miles/gallon (y) and engine displacement (x1).

```
# find R^2
X = MS(['x1']).fit_transform(auto)
y = auto['y']
model1 = sm.OLS(y, X)
results1 = model1.fit()
print('The R square r2:', results1.rsquared)
print('The adjusted R square r2_adj:', results1.rsquared_adj)
```

```
The R square r2: 0.772271242320445
The adjusted R square r2_adj: 0.7646802837311264
```

Conclusion: R^2=0.7723, and the correlation between y and x1 is r=-0.8788

```
# Find correlation between x1 and y
corr = np.corrcoef(auto['x1'], auto['y'])
corr2 = corr[0,1]**2
print('The correlation between y and x1:', corr[0,1])
print('The square of correlation:', corr2)
```

```
The correlation between y and x1: -0.8787896462296564
The square of correlation: 0.7722712423204446
```

$$r^2 = (-0.8788)^2 = 0.7723 = R^2$$

g. Calculate **sum of squared errors (SSE)**, **mean square error (MSE)**, and **regression (or residual) standard error (s)**.

```
# Find SSres=SSE, MSE
print('SSres:', results1.ssr)
print('MSE = s^2 = SSres/n-2:', results1.mse_resid)
```

```
SSres: 281.82437762005355
MSE = s^2 = SSres/n-2: 9.394145920668452
```

The LSE estimator of σ^2 is $s^2 = 9.39$.
The residual sum of squares SSres=281.82

h. Find a 95% PI on the **mean (average) gasoline mileage** if the engine displacement is 275 in.

i. Suppose that we wish to predict **the gasoline mileage** obtained from a car with a 275 in engine. Give a point estimate of mileage. Find a 95% prediction interval on the mileage.

j. Compare the two intervals obtained in parts h and i. Explain the difference between them. Which one is wider, and why?

```
# Find C.I and P.I for mean response and new response
design = MS(['x1'])
design = design.fit(auto)
X = design.transform(auto)
print(X[:4])

new_df = pd.DataFrame({'x1':[275]})
newX = design.transform(new_df)
print(newX)

new_predictions = results1.get_prediction(newX)
print(new_predictions.predicted_mean)
CI = new_predictions.conf_int(alpha=0.05)
print('95% CI on the mean gasoline mileage:', CI)

PI = new_predictions.conf_int(obs=True , alpha=0.05)
print('95% prediction interval on the mileage:', PI)
```

```
   intercept     x1
0        1.0  350.0
1        1.0  350.0
2        1.0  250.0
3        1.0  351.0
   intercept   x1
0        1.0  275
[20.69879276]
95% CI on the mean gasoline mileage: [[19.58806865 21.80951687]]
95% prediction interval on the mileage: [[14.34147153 27.05611399]]
```

Conclusion:

h. The 95% confidence interval on the mean mileage is (19.59, 21.81).

i. The point estimate of mileage is 20.69879.
The 95% prediction interval on the mileage is (14.34, 27.06).

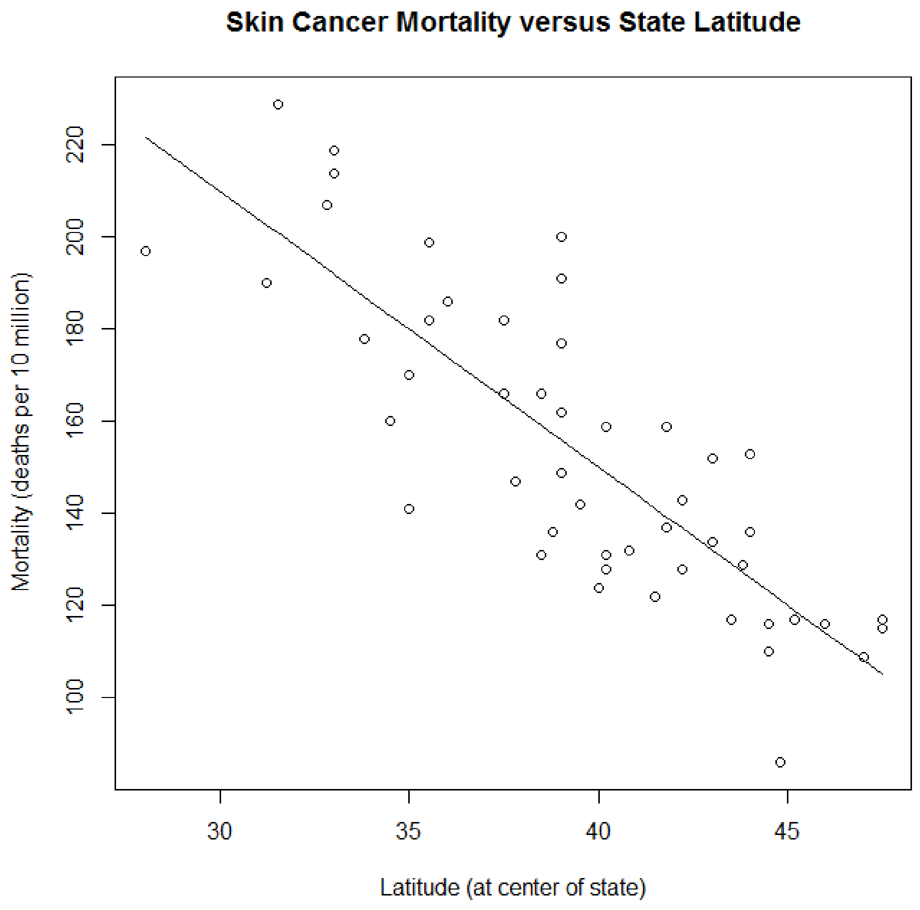
j. Compare the above two intervals obtained in parts h and i, the interval of mean mileage gives us a narrower interval. The PI of mileage gives a wider interval.

1. Skin cancer

- Load the skin cancer data, estimate β_0 *and* β_1, and produce a **scatterplot with a simple linear regression line**
- 95% Confidence Interval for β_0 *and* β_1

```
> setwd('F:\\UMD\\teaching\\STT530\\Lab')
> skincancer <- read.table("skincancer.txt", header=T)
> attach(skincancer)
The following objects are masked from skincancer (pos = 3):

    Lat, Long, Mort, Ocean, State

The following objects are masked from skincancer (pos = 4):

    Lat, Long, Mort, Ocean, State

> dim(skincancer)
[1] 49  5
> skincancer[1:3,]
     State  Lat Mort Ocean  Long
1  Alabama 33.0  219     1  87.0
2  Arizona 34.5  160     0 112.0
3 Arkansas 35.0  170     0  92.5
> model <- lm(Mort ~ Lat)
> summary(model)

Call:
lm(formula = Mort ~ Lat)

Residuals:
    Min      1Q  Median      3Q     Max 
-38.972 -13.185   0.972  12.006  43.938 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 389.1894    23.8123   16.34  < 2e-16 ***
Lat          -5.9776     0.5984   -9.99 3.31e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.12 on 47 degrees of freedom
Multiple R-squared:  0.6798,    Adjusted R-squared:  0.673 
F-statistic:  99.8 on 1 and 47 DF,  p-value: 3.309e-13

> confint(model)
                 2.5 %     97.5 %
(Intercept) 341.285151 437.093552
Lat          -7.181404  -4.773867
```

```
> plot(x=Lat, y=Mort,
+      xlab="Latitude (at center of state)", ylab="Mortality (deaths per 10 million)",
+      main="Skin Cancer Mortality versus State Latitude",
+      panel.last = lines(sort(Lat), fitted(model)[order(Lat)]))
> detach(skincancer)
```



2. Student height and weight

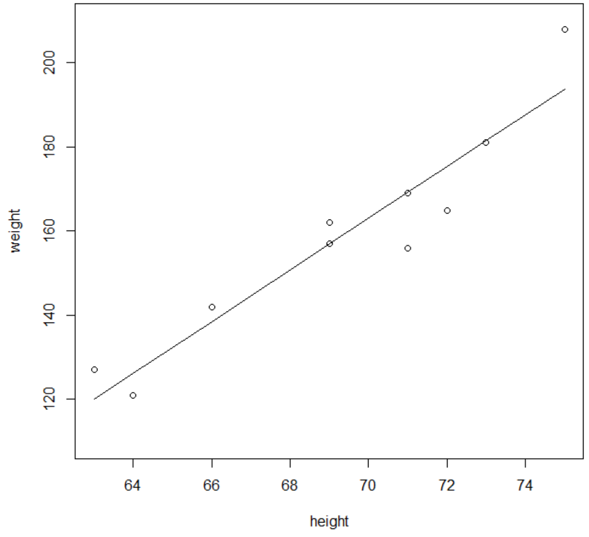
- Load the student height and weight data.
- Fit a **simple linear regression model**.

```
> heightweight <- read.table("student_height_weight.txt", header=T)
> attach(heightweight)
> dim(heightweight)
[1] 10  2
> heightweight
   ht  wt
1  63 127
2  64 121
3  66 142
4  69 157
5  69 162
6  71 156
7  71 169
8  72 165
9  73 181
10 75 208
> model <- lm(wt ~ ht)
> summary(model)

Call:
lm(formula = wt ~ ht)

Residuals:
     Min       1Q   Median       3Q      Max 
-13.2339  -4.0804  -0.0963   4.6445  14.2158 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -266.5344    51.0320  -5.223    8e-04 ***
ht             6.1376     0.7353   8.347 3.21e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.641 on 8 degrees of freedom
Multiple R-squared:  0.897,     Adjusted R-squared:  0.8841 
F-statistic: 69.67 on 1 and 8 DF,  p-value: 3.214e-05
```

2. Student height and weight

- Produce a scatterplot with a simple linear regression line and another line with specified intercept and slope.
- Calculate **sum of squared errors (SSE)**.
- **Predict** weight for height=66 and height=67.
- PIs for E(y|x_0) and y|x_0.

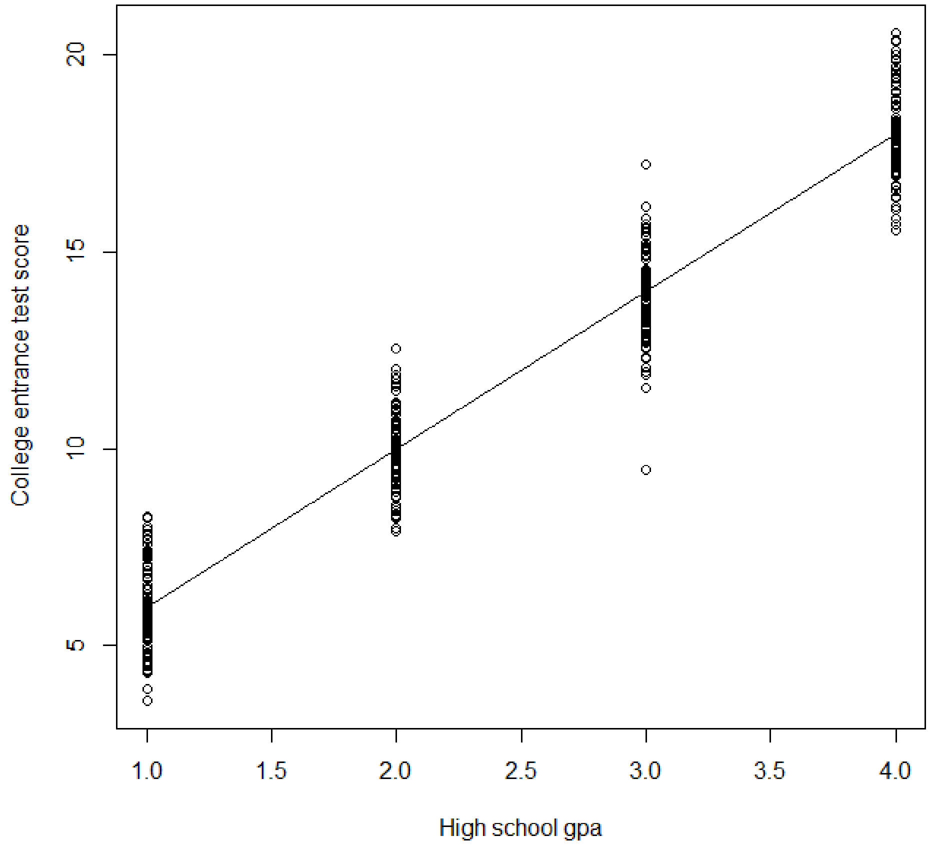
```
> plot(x=ht, y=wt, ylim=c(110,210), xlab="height", ylab="weight",
+       panel.last = c(lines(sort(ht), fitted(model[order(ht)]))))
> sum(residuals(model)^2) # SSE = 597.386
[1] 597.386
> predict(model, newdata=data.frame(ht=c(66, 67))) # 138.5460 144.6836
       1        2
138.5460 144.6836
> predict(model, newdata=data.frame(ht=c(66, 67)), interval="confidence")# PI for E(y|x0)
       fit      lwr      upr
1 138.5460 130.1186 146.9734
2 144.6836 137.2728 152.0943
> predict(model, newdata=data.frame(ht=c(66, 67)), interval="prediction")# PI for y|x0
       fit      lwr      upr
1 138.5460 116.9102 160.1818
2 144.6836 123.4231 165.9440
> detach(heightweight)
```



3. High school GPA and college test scores

- Generate the high school GPA and college test score (population) data.
- Produce a scatterplot of the population data with the population regression line.

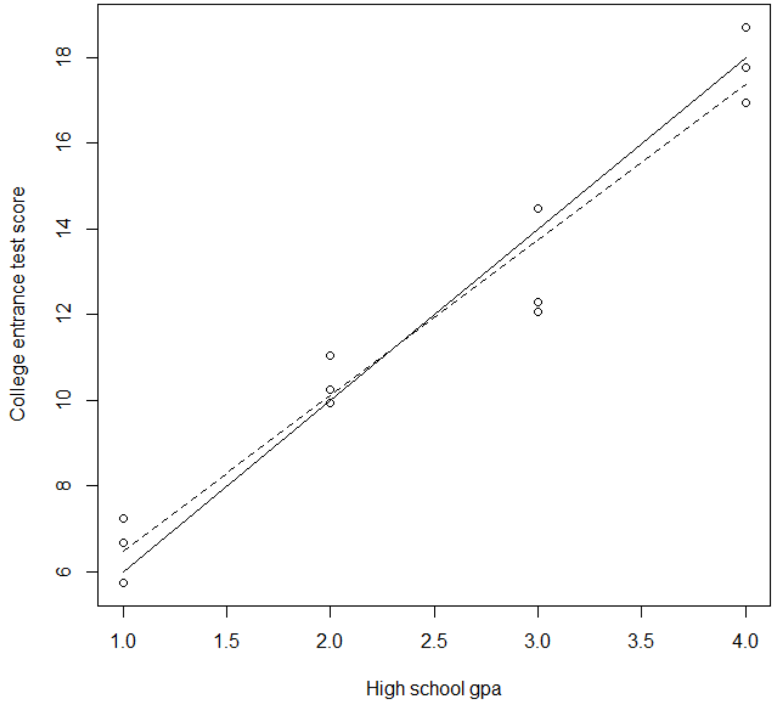
```
> X <- c(rep(1, 100), rep(2, 100), rep(3, 100), rep(4, 100))
> Y <- 2 + 4*X + rnorm(400, 0, 1)
> plot(X, Y, xlab="High school gpa", ylab="College entrance test score",
+      panel.last = lines(X, 2+4*X))
```



3. High school GPA and college test scores

- Sample the data (your results will differ since we're randomly sampling here).
- Produce a scatterplot of the sample data with a simple linear regression line and the population regression line.

```
> Xs <- c(rep(1, 3), rep(2, 3), rep(3, 3), rep(4, 3))
> Ys <- Y[c(rep(0, 3), rep(100, 3), rep(200, 3), rep(300, 3)) + sample.int(100, 12)]
> model <- lm(Ys ~ Xs)
> plot(Xs, Ys, xlab="High school gpa", ylab="College entrance test score",
+      panel.last = c(lines(Xs, 2+4*Xs),
+                        lines(sort(Xs), fitted(model[order(Xs)]),lty=2)))
```



3. High school GPA and college test scores

- Calculate **sum of squared errors (SSE)**, **mean square error (MSE)**, and **regression (or residual) standard error (S)**.

```
> sum(residuals(model)^2) # SSE = 9.669
[1] 9.669046
> sum(residuals(model)^2)/10 # MSE = 0.9669
[1] 0.9669046
> sqrt(sum(residuals(model)^2)/10) # S = 0.9833
[1] 0.9833131
> summary(model) # Residual standard error: 0.9833 on 10 degrees of freedom

Call:
lm(formula = Ys ~ Xs)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.6775 -0.5155  0.1725  0.7420  1.3440 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.8679     0.6953   4.125  0.00206 ** 
Xs            3.6250     0.2539  14.278 5.61e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9833 on 10 degrees of freedom
Multiple R-squared:  0.9532,	Adjusted R-squared:  0.9486 
F-statistic: 203.9 on 1 and 10 DF,  p-value: 5.612e-08
```

4. Skin cancer

- Load the skin cancer data.
- Fit a simple linear regression model with y = Mort and x = Lat and display the **coefficient of determination, R^2**.
- Calculate the **correlation** between Mort and Lat.

```
> skincancer <- read.table("skincancer.txt", header=T)
> attach(skincancer)
The following objects are masked from skincancer (pos = 3):

    Lat, Long, Mort, Ocean, State

> model <- lm(Mort ~ Lat)
> summary(model) # Multiple R-squared:  0.6798

Call:
lm(formula = Mort ~ Lat)

Residuals:
    Min      1Q  Median      3Q     Max 
-38.972 -13.185   0.972  12.006  43.938 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 389.1894    23.8123   16.34  < 2e-16 ***
Lat          -5.9776     0.5984   -9.99 3.31e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 19.12 on 47 degrees of freedom
Multiple R-squared:  0.6798,    Adjusted R-squared:  0.673 
F-statistic:  99.8 on 1 and 47 DF,  p-value: 3.309e-13

> cor(Mort, Lat) # correlation = -0.8245178
[1] -0.8245178
> detach(skincancer)
```

5. Temperature

- Create the temperature data.
- Fit a simple linear regression model with y = F and x = C and display the **coefficient of determination, R^2**.
- Calculate the **correlation** between F and C.

```
> C <- seq(0, 50, by=5)
> F <- (9/5)*C+32
> model <- lm(F ~ C)
> summary(model) # Multiple R-squared:      1

Call:
lm(formula = F ~ C)

Residuals:
       Min         1Q     Median         3Q        Max 
-1.509e-14 -1.580e-16  2.723e-16  1.628e-15  1.180e-14 

Coefficients:
             Estimate Std. Error   t value Pr(>|t|)    
(Intercept) 3.200e+01  3.714e-15 8.616e+15   <2e-16 ***
C           1.800e+00  1.256e-16 1.434e+16   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.584e-15 on 9 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:      1 
F-statistic: 2.055e+32 on 1 and 9 DF,  p-value: < 2.2e-16

Warning message:
In summary.lm(model) : essentially perfect fit: summary may be unreliable
> cor(F, C) # correlation = 1
[1] 1
```

6. Driver's age and distance

- Load the driver's age and distance data.
- Fit a simple linear regression model with y = Distance and x = Age and display the **coefficient of determination, R^2**.
- Calculate the **correlation** between Distance and Age.

```
> signdist <- read.table("signdist.txt", header=T)
> attach(signdist)
> model <- lm(Distance ~ Age)
> summary(model) # Multiple R-squared:  0.642

Call:
lm(formula = Distance ~ Age)

Residuals:
    Min      1Q  Median      3Q     Max 
-78.231 -41.710   7.646  33.552 108.831 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 576.6819    23.4709  24.570  < 2e-16 ***
Age          -3.0068     0.4243  -7.086 1.04e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 49.76 on 28 degrees of freedom
Multiple R-squared:  0.642,     Adjusted R-squared:  0.6292 
F-statistic: 50.21 on 1 and 28 DF,  p-value: 1.041e-07

> cor(Distance, Age) # correlation = -0.8012447
[1] -0.8012447
> detach(signdist)
```

7. Lung function

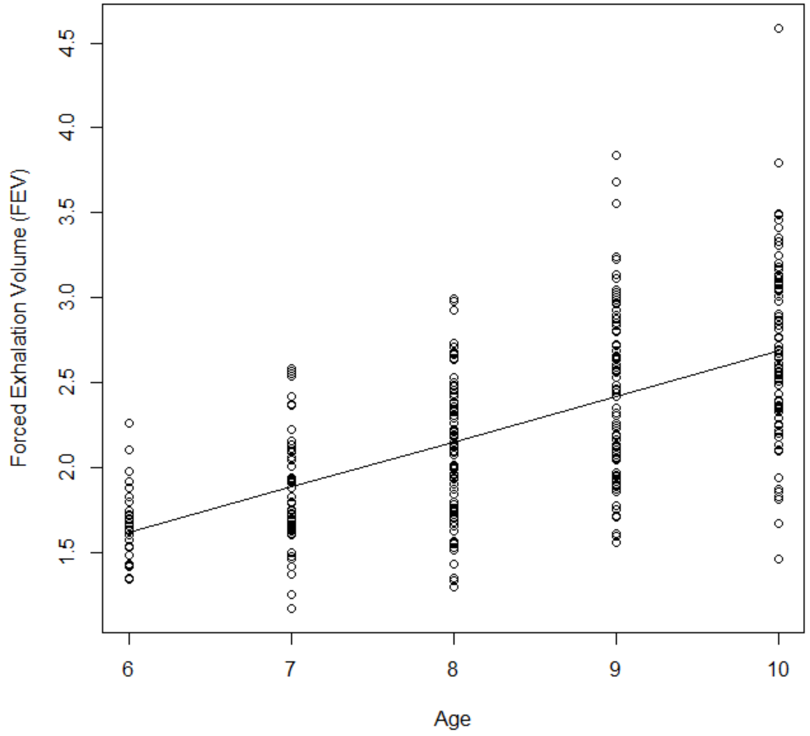
- Load the lung function data.
- Fit a simple linear regression model with y = FEV and x = age for ages 6-10 only and display the model results.

```
> lungfunction <- read.table("fev_dat.txt", header=T)
> attach(lungfunction)
The following objects are masked from lungfunction (pos = 3):

    age, FEV, ht, sex, smoke

> dim(lungfunction)
[1] 654   5
> lungfunction[1:5,]
  age   FEV   ht sex smoke
1   9 1.708 57.0   0     0
2   8 1.724 67.5   0     0
3   7 1.720 54.5   0     0
4   9 1.558 53.0   1     0
5   9 1.895 57.0   1     0
>
> model.1 <- lm(FEV ~ age, subset = age>=6 & age<=10)
> summary(model.1)

Call:
lm(formula = FEV ~ age, subset = age >= 6 & age <= 10)

Residuals:
     Min       1Q   Median       3Q      Max
-1.22576 -0.28855 -0.00534  0.27106  1.90724

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01165    0.15237   0.076    0.939
age          0.26721    0.01801  14.839   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4312 on 349 degrees of freedom
Multiple R-squared:  0.3869,    Adjusted R-squared:  0.3851
F-statistic: 220.2 on 1 and 349 DF,  p-value: < 2.2e-16
```

7. Lung function

- Produce a scatterplot for ages 6-10 only with a simple linear regression line.
- Fit a simple linear regression model with y = FEV and x = age for the full dataset and display the model results.

```
> plot(age[age>=6 & age<=10], FEV[age>=6 & age<=10],
+      xlab="Age", ylab="Forced Exhalation Volume (FEV)",
+      panel.last = lines(sort(age[age>=6 & age<=10]),
+                         fitted(model.1)[order(age[age>=6 & age<=10])]))
>
> model.2 <- lm(FEV ~ age)
> summary(model.2)

Call:
lm(formula = FEV ~ age)

Residuals:
     Min       1Q   Median       3Q      Max
-1.57539 -0.34567 -0.04989  0.32124  2.12786

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.431648   0.077895   5.541 4.36e-08 ***
age         0.222041   0.007518  29.533  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5675 on 652 degrees of freedom
Multiple R-squared:  0.5722,    Adjusted R-squared:  0.5716
F-statistic: 872.2 on 1 and 652 DF,  p-value: < 2.2e-16
```



7. Lung function

- Produce a scatterplot for the full dataset with a simple linear regression line.

```
> plot(age, FEV, xlab="Age", ylab="Forced Exhalation Volume (FEV)",
+       panel.last = lines(sort(age), fitted(model.2)[order(age)]))
> detach(lungfunction)
```