

## Simple Linear Models

### The Function `lm()`

The basic function for fitting ordinary simple/multiple linear models is `lm()`, and a streamlined version of the call is as follows:

```
> fitted.model <- lm(formula, data = data.frame)
```

For example

```
> fm2 <- lm(y ~ x1 + x2, data = production)
```

would fit a multiple regression model of  $y$  on  $x_1$  and  $x_2$  (with implicit intercept term). The important but technically optional parameter `data = production` specifies that any variables needed to construct the model should come first from the production data frame. This is the case regardless of whether data frame production has been attached on the search path or not.

### Generic functions for extracting model information

The value of `lm()` is a fitted model object; technically a list of results of class "lm". Information about the fitted model can then be displayed, extracted, plotted and so on by using generic functions that orient themselves to objects of class "lm". These include

<code>add1</code>	<code>coef</code>	<code>effects</code>	<code>kappa</code>	<code>predict</code>	<code>residuals</code>
<code>alias</code>	<code>deviance</code>	<code>family</code>	<code>labels</code>	<code>print</code>	<code>step</code>
<code>anova</code>	<code>drop1</code>	<code>formula</code>	<code>plot</code>	<code>proj</code>	<code>summary</code>

A brief description of the most commonly used ones is given below.

```
anova(object 1, object 2)
```

Compare a submodel with an outer model and produce an analysis of variance table.

```
coefficients(object)
```

Extract the regression coefficient (matrix).

Short form: `coef(object)`.

```
deviance(object)
```

Residual sum of squares.

```
formula(object)
```

Extract the model formula.

```
plot(object)
```

Produce four plots, showing residuals, fitted values and some diagnostics.

```
predict(object, newdata=data.frame)
```

The data frame supplied must have variables specified with the same labels as the original. The value is a vector or matrix of predicted values corresponding to the determining variable values in `data.frame`.

```
print(object)
```

Print a concise version of the object. Most often used implicitly.

```
residuals(object)
```

Extract the (matrix of) residuals.

Short form: `resid(object)`.

```
step(object)
```

Select a suitable model by adding or dropping terms and preserving hierarchies. The model with the largest value of AIC (Akaike's An Information Criterion) discovered in the stepwise search is returned. This function will be used later on.

```
summary(object)
```

Print a comprehensive summary of the results of the regression analysis.

### Example: (Problem 10.8)

Read the data into R. Here is one way of reading the data.

```
> data <- data.frame(SAL = c(10455, 9680, 7300, 9388, 12496, 11812, 9224, 11725,
11320, 12000, 12500, 13310, 12105, 6200, 11522, 8000, 12548, 7700,
10028, 13176, 13255, 13004, 8000, 8224, 10750, 11669, 12322, 11002,
10666, 10839), CGPA = c(2.58, 2.31, 2.47, 2.52, 3.22, 3.37, 2.43, 3.08,
2.78, 2.98, 3.55, 3.64, 3.72, 2.24, 2.7, 2.3, 2.83, 2.37, 2.52, 3.22,
3.55, 3.55, 2.47, 2.47, 2.78, 2.78, 2.98, 2.58, 2.58, 2.58))
```

```
> summary(data)
```

SAL		CGPA	
Min. :	6200	Min. :	2.240
1st Qu.:	9461	1st Qu.:	2.482
Median :	11161	Median :	2.740
Mean :	10741	Mean :	2.838
3rd Qu.:	12268	3rd Qu.:	3.185
Max. :	13310	Max. :	3.720

Plot SAL (Y) vs. CGPA (X) (see the graph on next page.)

```
> par(mfrow=c(1,1),mar=rep(4,4))
# The above commands line contains options for graph page.
# 1x1 array of figures
# on the page, with 4 lines of margin around each
> plot(data$CGPA, data$SAL, ylab="SAL",
       xlab="CGPA", main="Scatter Plot of the Data" )
```

To compute the correlation coefficient between Y and X

```
> r <- cor(data$CGPA, data$SAL); r
```

```
[1] 0.8273676
```

To fit the LS regression line.

```
> fit <- lm(SAL ~ CGPA, data=data)
> summary(fit)
```

Call:

```
lm(formula = SAL ~ CGPA, data = data)
```

Residuals:

Min	1Q	Median	3Q	Max
-2368.4	-814.9	164.0	861.8	1837.6

Coefficients:

```

              Estimate Std. Error t value Pr(>|t|)
(Intercept)   435.9      1337.9   0.326   0.747
CGPA          3630.6      465.8   7.795 1.72e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1125 on 28 degrees of freedom
Multiple R-Squared: 0.6845,    Adjusted R-squared: 0.6733
F-statistic: 60.76 on 1 and 28 DF,  p-value: 1.72e-008

```

Obviously, the hypothesis  $H_0 : \beta_1 = 0$  is rejected as the corresponding p-value is  $1.72e-08$ . which is really small. However, the p-value corresponding to  $H_0 : \beta_0 = 0$  is 0.747. So we can not reject the null.

The analysis of variance table of the linear fit can be output as follows.

```

> anova(fit)

Analysis of Variance Table

Response: SAL
      Df  Sum Sq Mean Sq F value    Pr(>F)
CGPA    1 76858486 76858486  60.758 1.720e-08 ***
Residuals 28 35419547  1264984
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

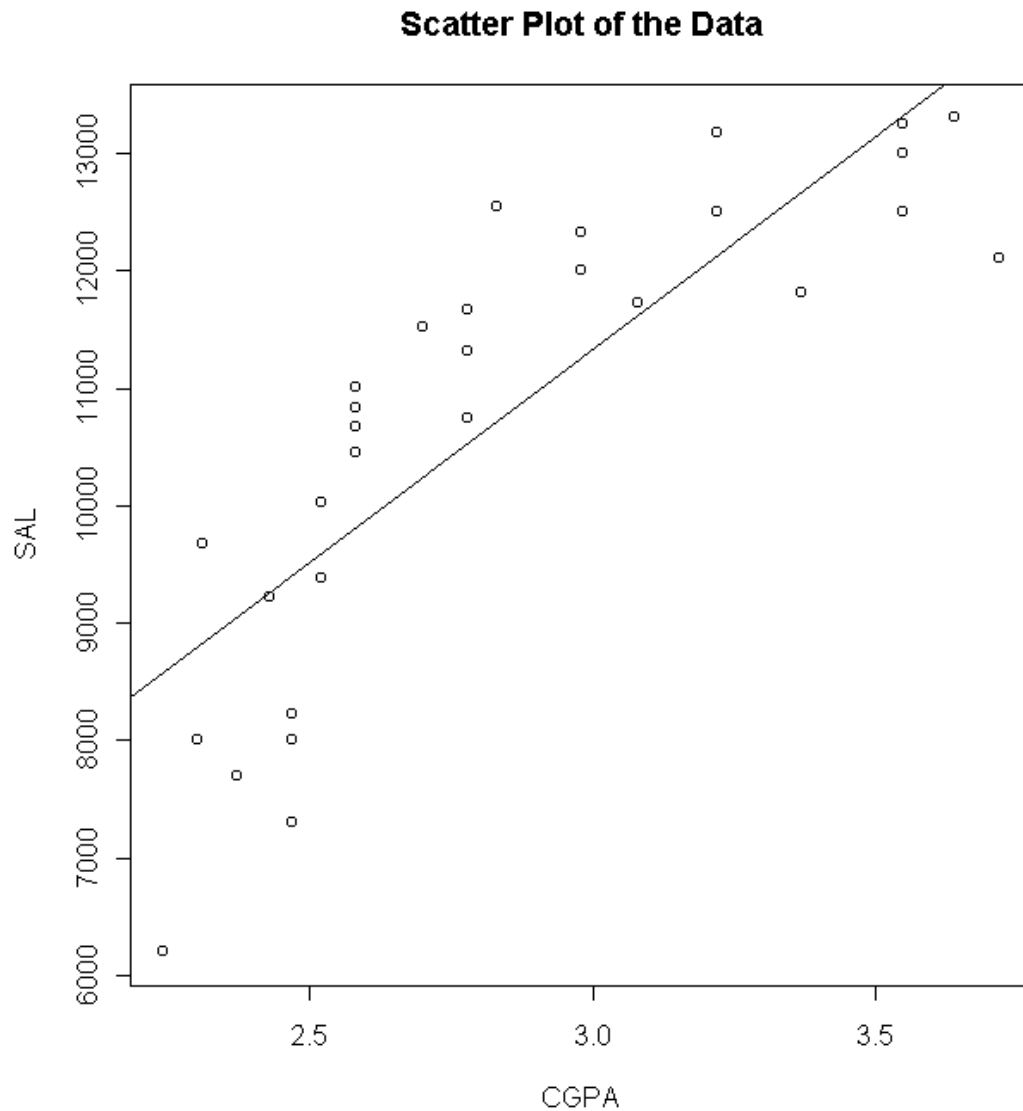
Note that the F-test for  $H_0 : \beta_1 = 0$  has exactly the same p-value as the t-test.

The fitted straight line is  $y = 435.9 + 3630.6 \cdot x$ . To add the fitted line to the scatter plot, use the command `abline`.

```

> abline(a=435.9, b=3630.6)
# a - intercept; b - slope.

```



**Figure1:** The scatter plot of the data, together with the fitted straight line.

To estimate the mean response or predict a new individual response, use the function `predict`. It has the following usage.

```
predict(object, newdata, se.fit = FALSE, scale = NULL, df = Inf,
        interval = c("none", "confidence", "prediction"),
        level = 0.95, type = c("response", "terms"))
```

For example, to estimate the mean response when  $x_0 = 2.5$ , use the following commands.

```
> est.intv <- predict( fit, newdata=data.frame(CGPA=2.5),
                    se.fit=TRUE, interval="confidence", level=0.95)
```

```
> est.intv

      $fit
      fit      lwr      upr
[1,] 9512.327 8982.113 10042.54

      $se.fit
[1] 258.842

      $df
[1] 28

      $residual.scale
[1] 1124.715
```

To predict the response when  $x_0 = 2.5$ , use

```
> pred.intv <- predict( fit, newdata=data.frame(CGPA=2.5),
      se.fit=TRUE, interval="prediction", level=0.95)
> pred.intv

      $fit
      fit      lwr      upr
[1,] 9512.327 7148.228 11876.43

      $se.fit
[1] 258.842

      $df
[1] 28

      $residual.scale
[1] 1124.715
```

Now we would like to plot the *confidence band* and the *prediction band*, the confidence (or prediction) intervals for all values in the range of  $CGPA(X)$

```
> range(data$CGPA)
[1] 2.24 3.72
```

Here, we basically consider, say, 100 values between 2.24 and 3.72 and obtain the intervals for each values.

```
> new <- data.frame(CGPA= seq(2.24, 3.72, length=100))
> new
      CGPA
1  2.240000
2  2.254949
3  2.269899
.....
99 3.705051
100 3.720000
```

We first get the confidence intervals. The object `CI95` has several attributes, which can be listed using commands like `names` or `attributes`.

```
> CI95 <- predict( fit, newdata=new, se.fit=TRUE,
  interval="confidence", level=0.95)
> names(CI95)

[1] "fit"          "se.fit"       "df"          "residual.scale"

> CI95$fit
      fit      lwr      upr
1  8568.381  7859.290  9277.471
2  8622.656  7924.997  9320.315
3  8676.931  7990.597  9363.265
.....
98 13833.061 12917.998 14748.125
99 13887.336 12959.583 14815.090
100 13941.612 13001.123 14882.100
```

Similarly, get the prediction intervals.

```
> PI95 <- predict( fit, newdata=new, se.fit=TRUE,
  interval="prediction", level=0.95)
> PI95$fit
      fit      lwr      upr
1  8568.381  6157.853 10978.91
2  8622.656  6215.466 11029.85
3  8676.931  6272.999 11080.86
4  8731.206  6330.451 11131.96
.....
98 13833.061 11354.115 16312.01
99 13887.336 11403.677 16371.00
100 13941.612 11453.167 16430.06
```

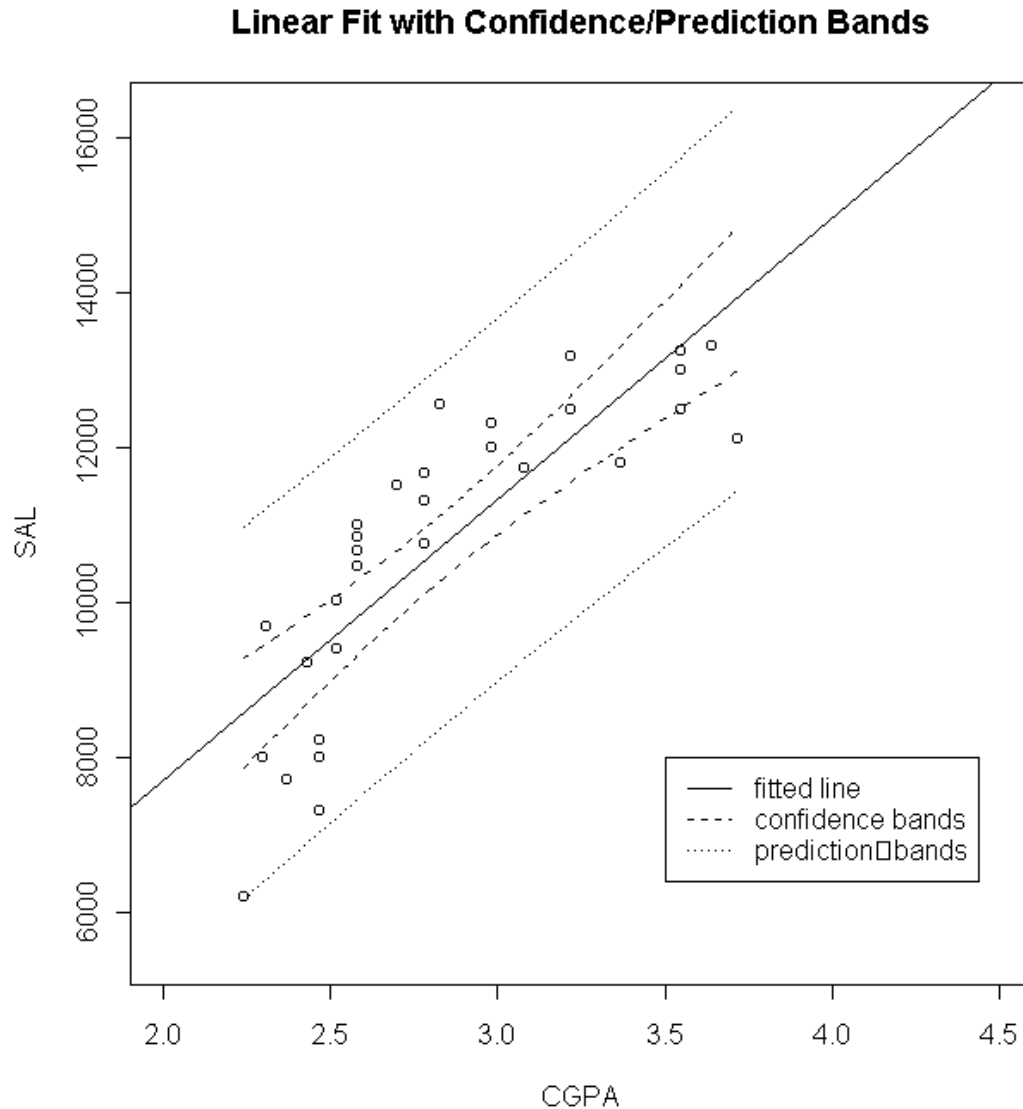
Now we add the confidence/prediction bands to the scatter plot of the data.

```
par(mar=rep(4,4))
# First create a frame for the plot, which is big enough to
# contain the confidence bands
plot(c(2, 4.5), c(5500, 16300), type="n", ylab="SAL",
      xlab="CGPA", main="Linear Fit with Confidence/Prediction Bands" )
# Add the scatter plot of the data
points(data$CGPA, data$SAL)
# Add the fitted line
abline(a=435.9, b=3630.6)
lines(new$CGPA, CI95$fit[,2], lty=2) # The lower CI bond
lines(new$CGPA, CI95$fit[,3], lty=2) # The upper CI bond
lines(new$CGPA, PI95$fit[,2], lty=3) # The lower PI bond
lines(new$CGPA, PI95$fit[,3], lty=3) # The upper PI bond

# To add a legend to the graph
```

```
legend(3.5,8000,c("fitted line", "confidence bands", "prediction
bands"), lty=1:3)
```

The above command lines result in the following plot.



**Figure 2:** Confidence band and prediction band.

If one prefers colored pictures, this can be done using the `col` option.

```
lines(new$age,CI95$fit[,2],lty=2, col="red")
lines(new$age,CI95$fit[,3],lty=2, col="red")
lines(new$age,PI95$fit[,2],lty=3, col="blue")
lines(new$age,PI95$fit[,3],lty=3, col="blue")
legend(10,1500, c("fitted line", "confidence bands", "prediction
bands"), lty=1:3, col=c("black", "red", "blue"))
```