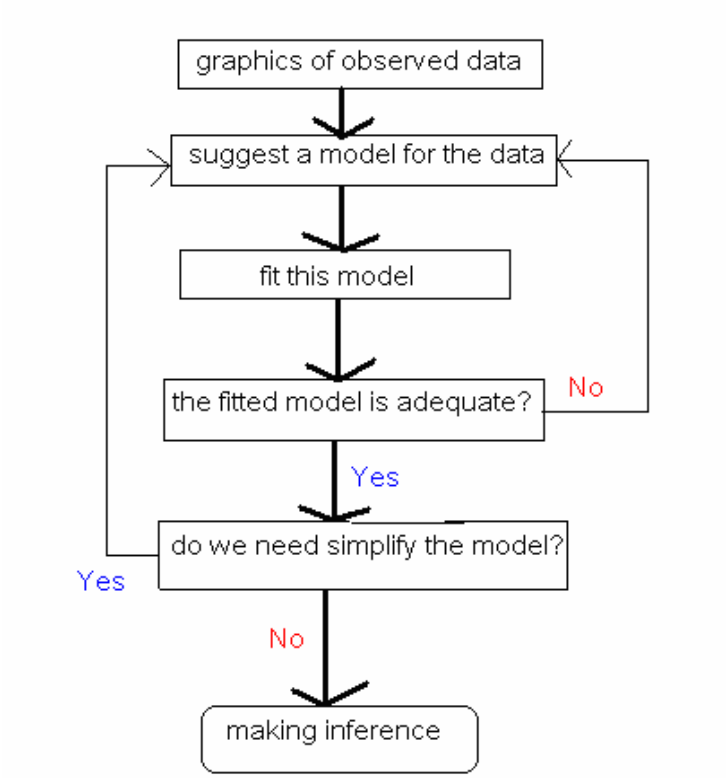
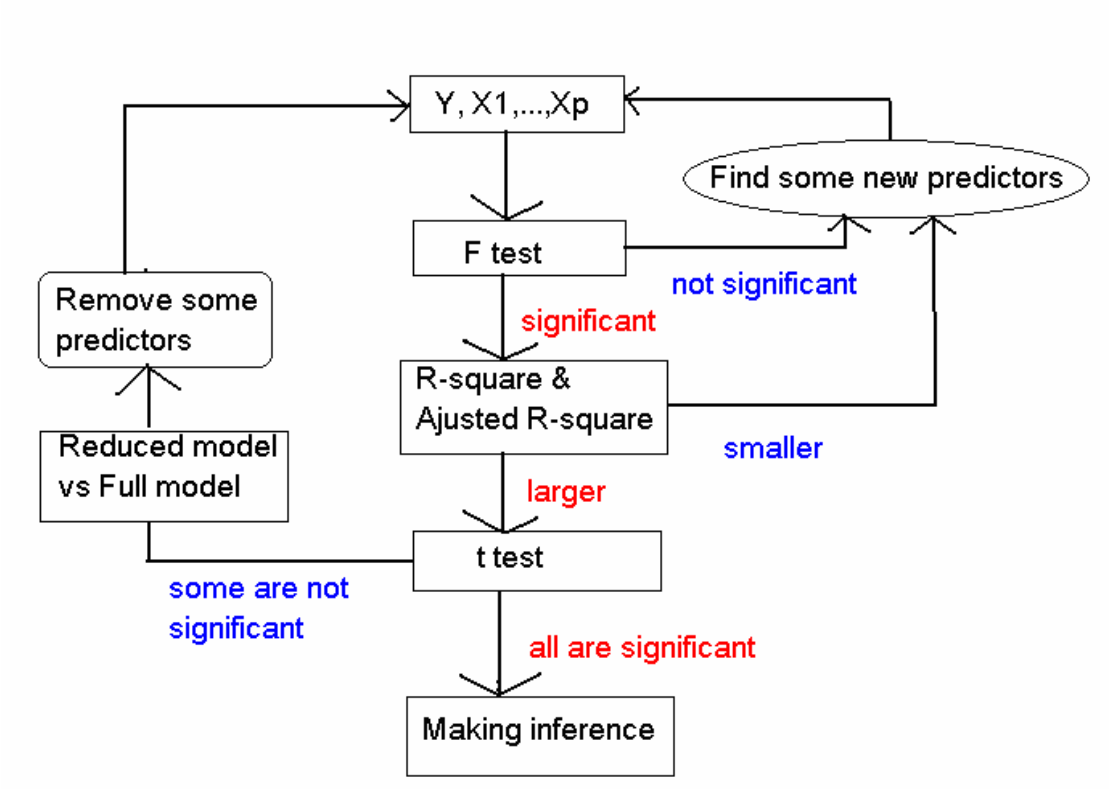
Modeling linear regression model with R

1. General procedure



A more detailed sub-procedure of the above is as follows.



Note that: the first flow chart in the above is the same as the case of SLR. Actually the only difference between SLR and MLR is that we need consider the problem of variable selection since there are more than on predictors involved in MLR. Most procedures of SLR and MLR are the same.

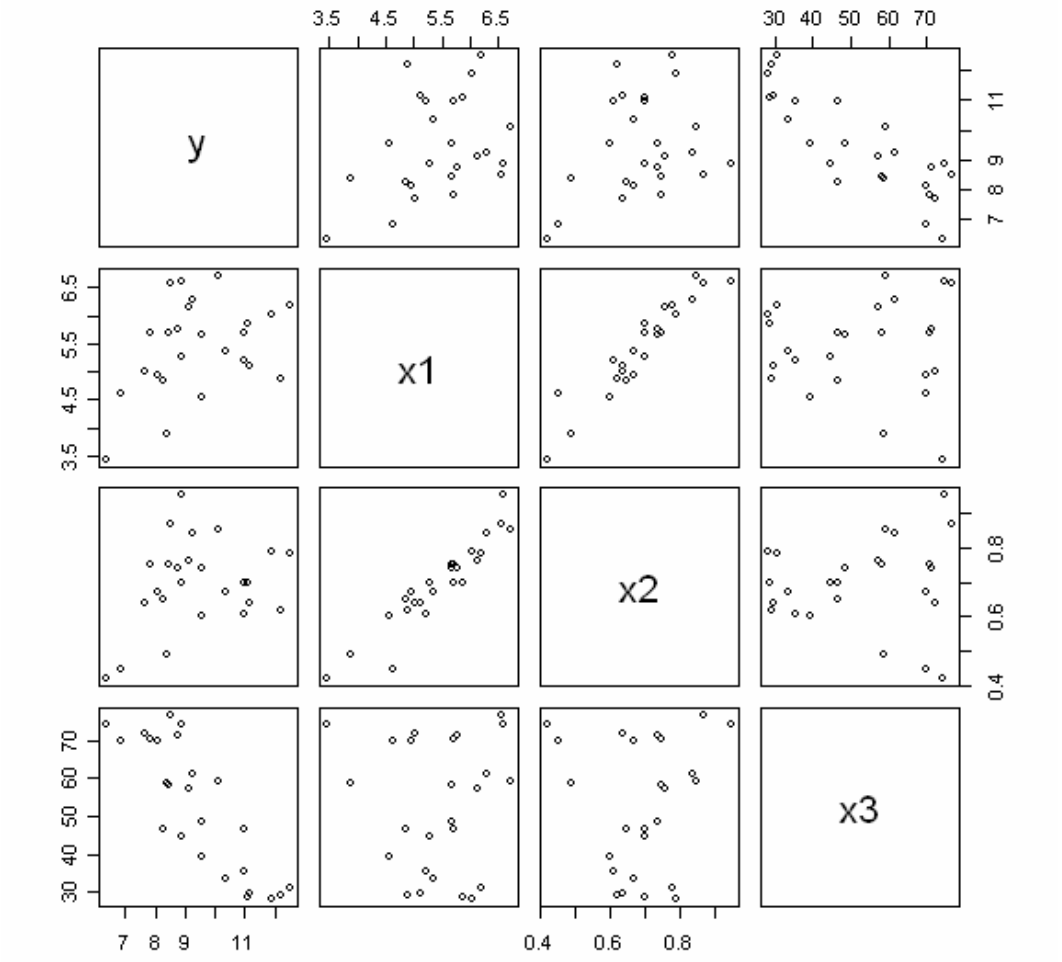
2. Graphic display of the observed data

There are more than one predictors involved in MLR, and it is not suitable to draw only one scatter plot. Hence, here we need a **scatter plot matrix** instead of a **scatter plot**.

Here we still use the steam data, downloadable from the webpage of this course, to demonstrate how to use R to do MLR. Specifically, we will consider variable 1 to be the response and variables, 2, 3 and 8, to be the predictors.

As for the case of SLR, copy these data to a text file, and save this file to be 'c:\steam.txt'. We next use the following program to read it to R and to draw a scatter plot matrix.

```
> # Graphical display of the observed data.
> steam_raw <- read.table('c:/steam.txt',header=FALSE)
> steam <- data.frame(y=steam_raw$V2,x1=steam_raw$V3,x2=steam_raw$V4,x3=steam_raw$V9)
> plot(steam)
>
```



Conclusion: Obviously there exist a strong relationship between X1 and X2, and a relationship between Y and X3.

3. Modeling multiple linear regression with R

We still use the function *lm* to do MLR as follows.

```
> # Fit a multiple linear regressin model.
> mlr <- lm(y ~ x1+x2+x3, data=steam)
> summary(mlr)

Call:
lm(formula = y ~ x1 + x2 + x3, data = steam)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2347 -0.4116  0.1239  0.3744  1.2979

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.514814   1.062969   8.951 1.30e-08 ***
x1           0.713592   0.502297   1.421     0.17
x2           0.330497   3.267694   0.101     0.92
x3          -0.079928   0.007884 -10.138 1.52e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.652 on 21 degrees of freedom
Multiple R-Squared: 0.8601,     Adjusted R-squared: 0.8401
F-statistic: 43.04 on 3 and 21 DF,  p-value: 3.794e-09
```

As to the case of SLR, we can extract the regression quantities we need from the model object. The structure of *mlr* is as follows.

```
> names(mlr)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
> mlrs <- summary(mlr)
> names(mlrs)
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
>
```

The structure of *slr* is as follows.

```
> names(slr)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
> slrs <- summary(slr)
> names(slrs)
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
>
```

4. Adequacy checking (This part is the same as the case of SLR)

4.1 From the viewpoint of the fitted model

There are three types of tests used to check the adequacy of the fitted model from the viewpoint of the model.

- ***t* tests:** they are used to check the significance of the fitted parameters.
  - $H_0 : \beta_0 = 0$ `8.951 1.30e-08 ***`
  - $H_0 : \beta_1 = 0$ `1.421 0.17`
  - $H_0 : \beta_2 = 0$ `0.101 0.92`
  - $H_0 : \beta_3 = 0$ `-10.138 1.52e-09 ***`
- **F-ratio:** they are used to compare two models.

  ```
  F-statistic: 43.04 on 3 and 21 DF,  p-value: 3.794e-09
  ```
- R^2 **Statistic:** they try to answer the question whether the fitted model is good enough.

  ```
  Multiple R-Squared: 0.8601,     Adjusted R-squared: 0.8401
  ```

4.2 From the viewpoint of residuals

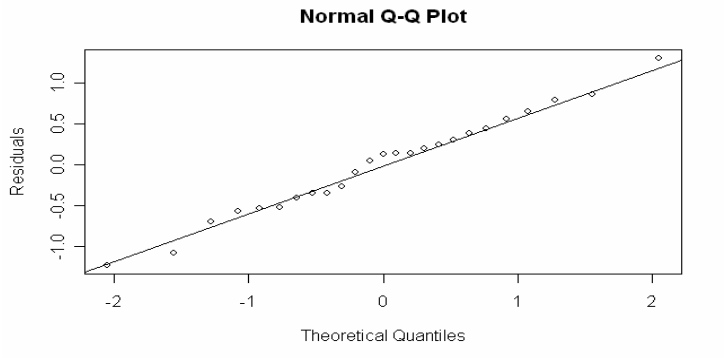
The basic idea of such type methods is to check whether there exists some useful information in the residuals. If yes, we need to adjust the model to improve its adequacy. Those methods include:

- Normality checking;
- Checking for time effects if the time order of the data is known, for non-constant variance to see whether we need take transformation on response, and for curvature of higher order than fitted in the predictors;
- Checking for the sequential dependence of the estimated residuals.

4.2.1 Normality checking

There are many ways to check the normality of a sequence of observed data. Here we only introduce the method of Q-Q plot.

```
> # Normality checking.
> qqnorm(residuals(mlr),ylab='Residuals')
> qqline(residuals(mlr))
>
```



4.2.2 Checking for time effects, non-constant variance and higher order curvatures

We use graphics to do such checks, and all judgments are based on our eyes. In those plots, we put the residuals $\hat{e}_i$ to the Y-axis, and then put the following variables to X-axis in turn:

- The time order of the data, if known;
- The fitted values $\hat{y}_i$;
- The predictor variable x_i. If there are more than one predictor variables, then put them to X-axis one by one.

In all of these cases, a satisfactory plot is one that shows a horizontal band of points giving the impression of the following figure.

Figure 2.5. A satisfactory residuals plot should give this overall impression.

There are many possible unsatisfactory plots, and three typical ones are listed as follows.

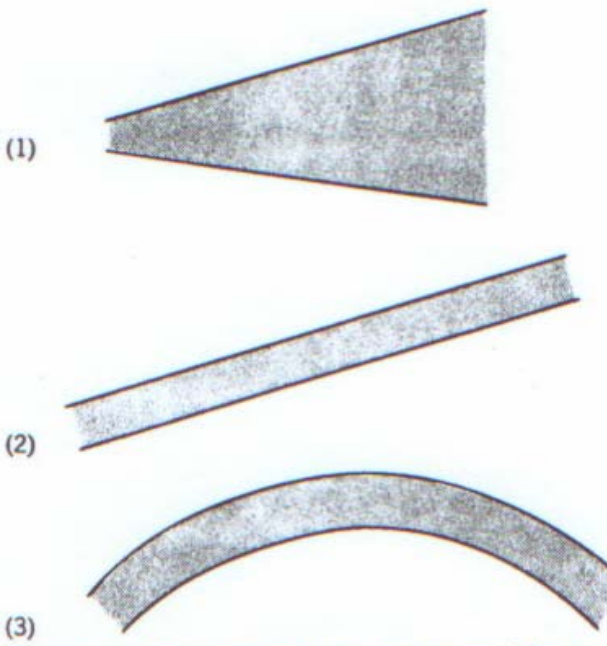
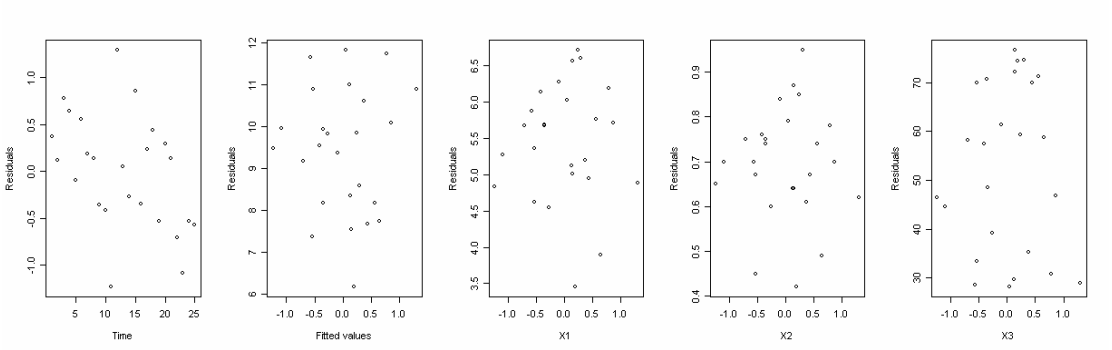


Figure 2.6. Examples of characteristics shown by unsatisfactory residuals behavior.

The program to draw them is listed as follows.

```
> # Draw some plots of residuals.
> par(mfrow=c(1,5))
> plot(residuals(mlr),ylab='Residuals',xlab='Time')
> plot(residuals(mlr),fitted(mlr),ylab='Residuals',xlab='Fitted values')
> plot(residuals(mlr),steam$x1,ylab='Residuals',xlab='X1')
> plot(residuals(mlr),steam$x2,ylab='Residuals',xlab='X2')
> plot(residuals(mlr),steam$x3,ylab='Residuals',xlab='X3')
> par(mfrow=c(1,1))
>
```

The three plots are draw as follows.



Conclusion: All these three plots display horizontal bands of points and, hence, there should be no useful information in the residuals.

Note:

- We can not draw the plot of $\hat{\varepsilon}_i = y_i - \hat{y}_i$ against y_i instead of against $\hat{y}_i$. The reason is that $\hat{\varepsilon}_i = y_i - \hat{y}_i$ is independent of $\hat{y}_i$, but depends on y_i. Specifically, if you regress $\hat{\varepsilon}_i = y_i - \hat{y}_i$ on $\hat{y}_i$, the coefficient will be zero; however it will be $1 - R^2$ for regressing on y_i. Why? Please check it by yourself.
- When we draw the plot of $\hat{\varepsilon}_i = y_i - \hat{y}_i$ against $\hat{y}_i$, there exist some straight lines with slopes of -1. This is a normal phenomenon, and these points on the same line have the same y_i but different x_i.
- Some statistics can be considered instead of just judging by eyes. However, we do not use them here.

4.2.3 Checking for sequential dependence

Durbin-Watson test is used to check the possible sequential dependence, and the test statistic is:

$$d = \sum_{u=2}^{n} (e_u - e_{u-1})^2 \Big/ \sum_{u=1}^{n} e_u^2 .$$

This statistic has the following properties:

- $0 \leqq d \leqq 4$;
- The distribution of d is symmetric about 2;
- If d is near to 0, then successive residuals are **positively** serially correlated;
- If d is near to 4, then successive residuals are **negatively** serially correlated.

Some critical values are given at Table 2.6 in the textbook.

- If $d < d_L$, then there is **positively** serially correlated;
- If $d > 4 - d_L$, then there is **negatively** serially correlated;

- If $d_U < d < 4 - d_U$, then there is no serially correlated.

This test is implemented in the *lmtest* package. You need download this package first. Then use the following program to perform this test.

```
> # Durbin-Watson tests.
> library(lmtest)
载入需要的程辑包：zoo

载入程辑包：'zoo'


        The following object(s) are masked from package:base :

         rapply

> dwtest(y ~ x1+x2+x3, data=steam)

        Durbin-Watson test

data:  y ~ x1 + x2 + x3
DW = 1.857, p-value = 0.3634
alternative hypothesis: true autocorrelation is greater than 0
```

5. F-test for reduced model and full model

5.1 Test for whether some coefficients are zeros

From the results of MLR, we can see that predictors X1 and X2 are not significant, i.e. they may be equal to zero. Hence we may want to test the following hypothesis,

$$H_0 : \beta_1 = \beta_2 = 0 .$$

```
> # Some F-tests.
> mlr1 <- lm(y ~ x3,data=steam)
> anova(mlr1,mlr)
Analysis of Variance Table

Model 1: y ~ x3
Model 2: y ~ x1 + x2 + x3
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     23 18.2234
2     21  8.9270  2    9.2964 10.934 0.0005569 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Conclusion: we have to reject the null hypothesis at the level of 0.01.

5.2 Test for more complicated relationship

From the results of MLR, we get that

$$\hat{\beta}_1 = 0.713592 \quad \text{and} \quad \hat{\beta}_2 = 0.330497 .$$

Hence you may want to

$$H_0 : \beta_1 = 2\beta_2$$

```
> mlr2 <- lm(y ~ I(2*x1+x2)+x3,data=steam)
> summary(mlr2)

Call:
lm(formula = y ~ I(2 * x1 + x2) + x3, data = steam)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2359 -0.4106  0.1239  0.3754  1.2974

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     9.517978   0.953246   9.985 1.24e-09 ***
I(2 * x1 + x2)  0.355000   0.074167   4.786 8.85e-05 ***
x3             -0.079941   0.007531 -10.615 4.03e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.637 on 22 degrees of freedom
Multiple R-Squared: 0.8601,     Adjusted R-squared: 0.8474
F-statistic: 67.63 on 2 and 22 DF,  p-value: 4.014e-10

> anova(mlr2,mlr)
Analysis of Variance Table

Model 1: y ~ I(2 * x1 + x2) + x3
Model 2: y ~ x1 + x2 + x3
  Res.Df   RSS Df Sum of Sq     F Pr(>F)
1     22 8.927
2     21 8.927  1 2.392e-05 1e-04  0.994
>
```

Conclusion: we can not reject the null hypothesis at the level of 0.1.

5.3 Test for whether coefficients are constant

From the results of MLR, we get

$$\hat{\beta}_1 = 0.713592\,.$$

Hence you may want to

$$H_0 : \beta_1 = 0.7$$

```
> mlr3 <- lm(y ~ offset(0.7*x1)+x2+x3,data=steam)
> summary(mlr3)

Call:
lm(formula = y ~ offset(0.7 * x1) + x2 + x3, data = steam)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2394 -0.4073  0.1232  0.3775  1.2955

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.532874   0.808290   11.79 5.54e-11 ***
x2           0.414138   1.035447    0.40    0.693
x3          -0.079971   0.007548  -10.59 4.18e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.637 on 22 degrees of freedom
Multiple R-Squared:  0.86,      Adjusted R-squared: 0.8473
F-statistic: 67.59 on 2 and 22 DF,  p-value: 4.038e-10

> anova(mlr3,mlr)
Analysis of Variance Table

Model 1: y ~ offset(0.7 * x1) + x2 + x3
Model 2: y ~ x1 + x2 + x3
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     22 8.9273
2     21 8.9270  1    0.0003 7e-04 0.9787
```

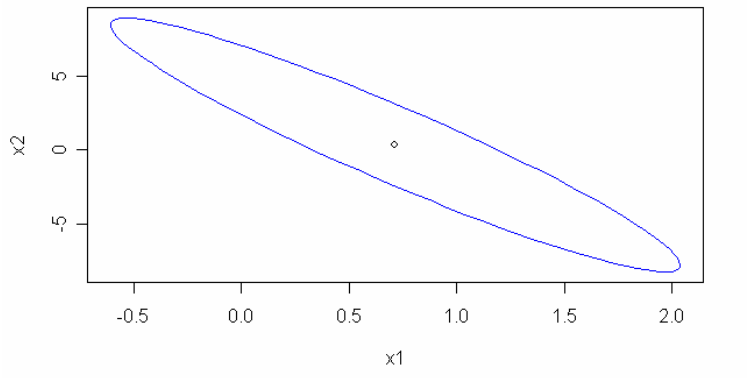
Conclusion: we can not reject the null hypothesis at the level of 0.1.

6. Making inference

6.1 Confidence interval

- **Confidence region best in theory**

```
> # Confidence intervals.
> library(ellipse)
> plot(ellipse(mlr,c(2,3),level=0.95),type='l',col=4)
> points(coef(mlr)[2],coef(mlr)[3])
>
```



- **Bonferroni limit**

```
> bon_level=0.05/4
> confint(mlr,level=1-bon_level)
                0.625 %    99.375 %
(Intercept)  6.6111741 12.41845382
x1          -0.6585005  2.08568384
x2          -8.5956417  9.25663529
x3          -0.1014649 -0.05839168
>
```

Note that: we need set bon_level=0.05/2, if we only consider the Bonferroni limit for X1 and X2.

6.2 Prediction

- For $l = c'\beta$, we use $\hat{l} = c'\hat{\beta}$ to be the estimator, and a $100(1-\alpha)\%$ CI is
  - $\hat{l} \pm t_{n-2}^{\alpha/2} s\sqrt{c'(X'X)^{-1}c}$.
- For $l = c'\beta + \varepsilon$ with $\varepsilon \sim N(0, c_3\sigma^2)$, we use $\hat{l} = c'\hat{\beta}$ to be the estimator, and a $100(1-\alpha)\%$ CI is

$$\hat{l} \pm t_{n-2}^{\alpha/2} s\sqrt{c_3 + c'(X'X)^{-1}c}.$$

```
> # Prediction
> con <- c(1,0.7,0.5,0.3)
> lhat <- sum(con*coef(mlr))
> lhat
[1] 10.15560
> t05 <- qt(0.975,21)
> bm <- t05*mlrs$sigma*sqrt(con%*%mlrs$cov.unscaled%*%con)
> c(lhat-bm,lhat+bm)
[1]  6.169092 14.142104
> c3 <- 1
> bm <- t05*mlrs$sigma*sqrt(con%*%mlrs$cov.unscaled%*%con+c3)
> c(lhat-bm,lhat+bm)
[1]  5.944817 14.366379
> con <- data.frame(x1=0.7,x2=0.5,x3=0.3)
> predict(mlr,con,interval='confidence',level=0.95)
          fit      lwr      upr
[1,] 10.15560 6.169092 14.14210
> predict(mlr,con,interval='prediction',level=0.95)
          fit      lwr      upr
[1,] 10.15560 5.944817 14.36638
>
```

7. The complete R program

```
# Graphical display of the observed data.
steam_raw <- read.table('c:/steam.txt',header=FALSE)
steam <-
data.frame(y=steam_raw$V2,x1=steam_raw$V3,x2=steam_raw$V4,x3=steam_raw$V9)
plot(steam)

# Fit a multiple linear regressin model.
mlr <- lm(y ~ x1+x2+x3, data=steam)
summary(mlr)
names(mlr)
mlrs <- summary(mlr)
names(mlrs)

# Normality checking.
qqnorm(residuals(mlr),ylab='Residuals')
qqline(residuals(mlr))

# Draw some plots of residuals.
par(mfrow=c(1,3))
plot(residuals(mlr),ylab='Residuals',xlab='Time')
plot(residuals(mlr),fitted(mlr),ylab='Residuals',xlab='Fitted values')
plot(residuals(mlr),steam$predictor,ylab='Residuals',xlab='Predictor variable')
par(mfrow=c(1,1))

# Durbin-Watson tests.
library(lmtest)
dwtest(y ~ x1+x2+x3, data=steam)

# Some F-tests.
mlr1 <- lm(y ~ x3,data=steam)
anova(mlr1,mlr)
mlr2 <- lm(y ~ I(2*x1+x2)+x3,data=steam)
summary(mlr2)
anova(mlr2,mlr)
mlr3 <- lm(y ~ offset(0.7*x1)+x2+x3,data=steam)
summary(mlr3)
anova(mlr3,mlr)

# Confidence intervals.
library(ellipse)
plot(ellipse(mlr,c(2,3),level=0.95),type='l',col=4)
points(coef(mlr)[2],coef(mlr)[3])
```

```
bon_level=0.05/4
confint(mlr,level=1-bon_level)

# Prediction
con <- c(1,0.7,0.5,0.3)
lhat <- sum(con*coef(mlr))
lhat
t05 <- qt(0.975,21)
bm <- t05*mlrs$sigma*sqrt(con%*%mlrs$cov.unscaled%*%con)
c(lhat-bm,lhat+bm)
c3 <- 1
bm <- t05*mlrs$sigma*sqrt(con%*%mlrs$cov.unscaled%*%con+c3)
c(lhat-bm,lhat+bm)
con <- data.frame(x1=0.7,x2=0.5,x3=0.3)
predict(mlr,con,interval='confidence',level=0.95)
predict(mlr,con,interval='prediction',level=0.95)
```