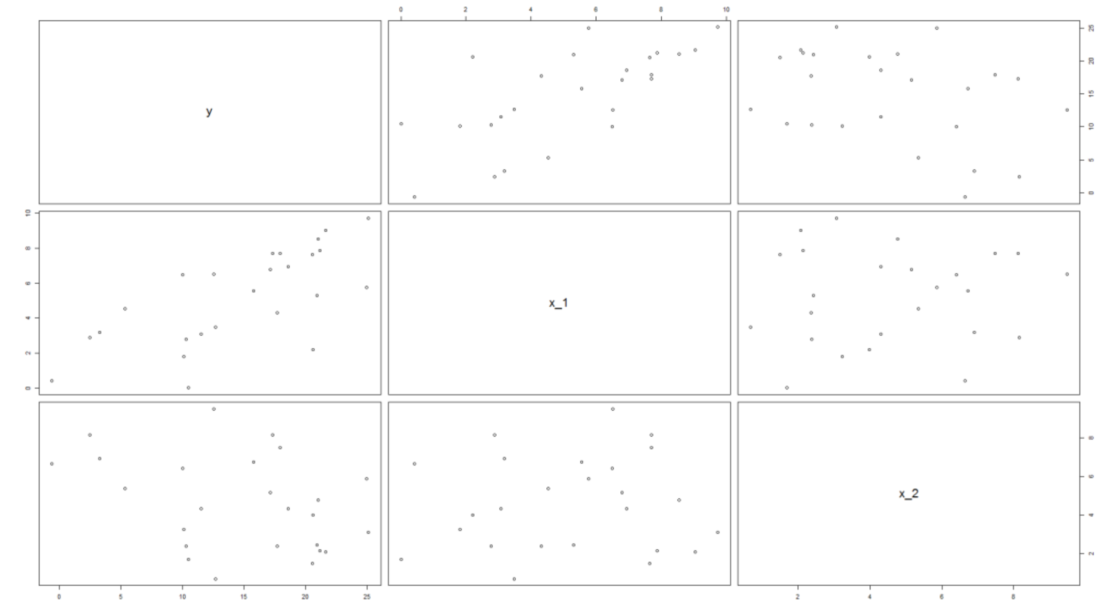
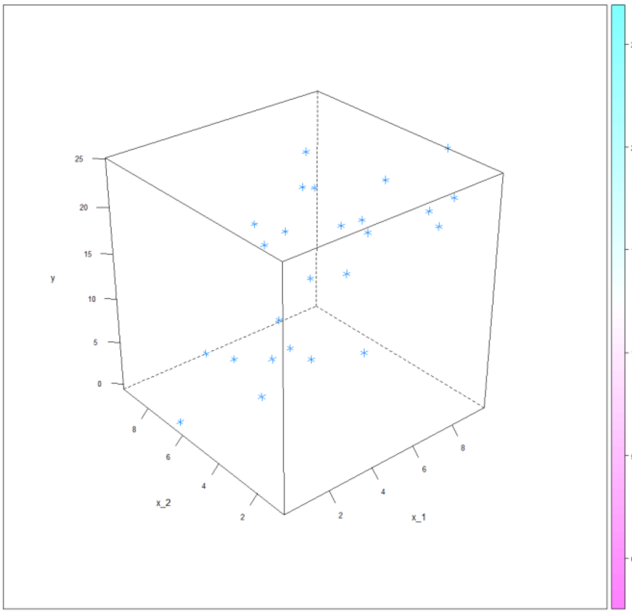
```
#########################################
############# Example 1 ################
####  A Randomly Generated data set ####
```

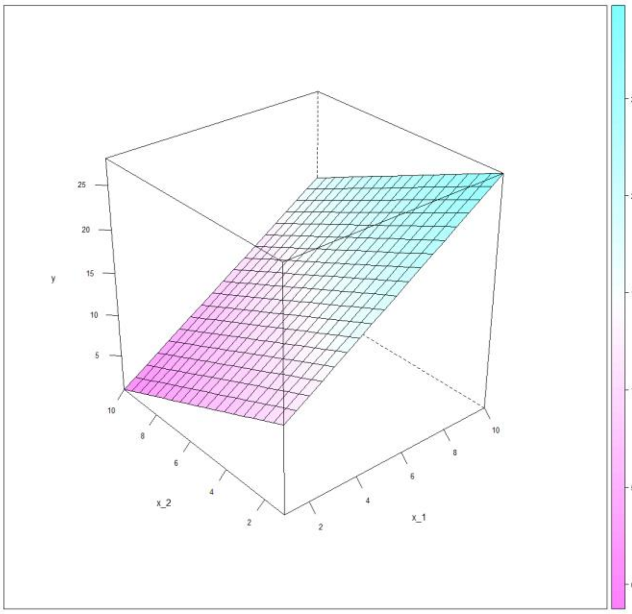
```
> ## Generate the Data
>
> n <- 25
>
> x_1 <- runif(n, 0, 10)
> x_1
 [1] 7.693679861 6.487045861 0.003756292 6.793340724 7.876556476 0.418775880
 [7] 6.934283618 2.774320284 6.502397740 9.734141885 9.034229312 2.879976165
[13] 3.079315228 4.309201655 2.201506943 7.696918347 1.817131375
5.768528963
[19] 4.526201107 8.536848703 5.558272493 3.185938387 3.472364827
5.295939255
[25] 7.648281108

> x_2 <- runif(n, 0, 10)
#expect x_1 and x_2 uncorrelated
> beta_0 <- 10
> beta_1 <- 2
> beta_2 <- -1
>
> eps <- rnorm(n, 0, 5)
> y <- beta_0 + beta_1*x_1 + beta_2*x_2 + eps
#simulated data
> y
 [1] 17.3195915 10.0187280 10.5096328 17.1434380 21.1940116 -0.6405045
 [7] 18.5967179 10.2916488 12.5671848 25.1145341 21.6367746  2.4751151
[13] 11.5243599 17.6999729 20.6003339 17.9276880 10.1307586 24.9784274
[19]  5.3285721 21.0115040 15.7688268  3.2714249 12.6876909 20.9227967
[25] 20.5467094

> ## Plot Data
>
> ## Scatterplot matrix
> pairs(y ~ x_1 + x_2)
>
```




```
>

> ## Correlation matrix
> cor(cbind(y, x_1, x_2))
            y          x_1          x_2
y     1.0000000   0.69329482  -0.35381487
x_1   0.6932948   1.00000000   0.05856667
x_2  -0.3538149   0.05856667   1.00000000

>## 3-D scatterplot

install.packages("lattice")
library(lattice)

cloud(y ~ x_1 + x_2, scales = list(arrows = FALSE), drape = TRUE, colorkey =
TRUE)
```




```
> library(lattice)
> cloud(y ~ x_1 + x_2, scales = list(arrows = FALSE), drape = TRUE, colorkey =
TRUE)
>
> ## 3-D perspective plot of regression surface
> plot.data <- expand.grid(int = 1, x_1 = seq(1,10,by=.5), x_2 = seq(1,10,by=.5))
> plot.data$y <- as.matrix(plot.data) %*% b
> wireframe(y ~ x_1 + x_2, data=plot.data, scales = list(arrows = FALSE), drape =
TRUE, colorkey = TRUE)
>
```




```
#######################################################
>
> ## Fit the Estimated Regression Line
> ##  y ~ x1 +x2 means use the model y = b0 + b1*x1 + b2*x2
>
> myfit1 <- lm(y ~ x_1 + x_2)
```

```
>> myfit1

Call:
lm(formula = y ~ x_1 + x_2)

Coefficients:
(Intercept)         x_1          x_2
     10.294       1.874       -1.152

     > #####################################################
     >
     > ## Hypothesis tests for
     > ##   H_0: beta_0=0 vs. H_a: beta_0 != 0 and
     > ##   H_0: beta_1=0 vs. H_a: beta_1 != 0
     > ##   H_0: beta_2=0 vs. H_a: beta_2 != 0
     >
     > summary(myfit1)

     Call:
     lm(formula = y ~ x_1 + x_2)

     Residuals:
         Min      1Q       Median      3Q     Max
     -7.2808 -3.1847  0.1621     1.9629 10.7786

     Coefficients:
                     Estimate    Std. Error   t value       Pr(>|t|)
     (Intercept)  10.2941     2.5493      4.038      0.00055 ***
     x_1             1.8741      0.3367      5.566      1.35e-05 ***
     x_2            -1.1516      0.3745      -3.075     0.00554 **

     Residual standard error: 4.455 on 22 degrees of freedom
     Multiple R-squared:  0.6368,     Adjusted R-squared:  0.6037
F-statistic: 19.28 on 2 and 22 DF,  p-value: 1.453e-05

> ## Make an ANOVA table and F-test
> ## These p-values correspond to the sequential testing approach
> ## H_0: beta_1=0 vs. H_a: beta_1 != 0  for the model y = b0 + b1*x1 + e
> ## H_0: beta_2=0 vs. H_a: beta_2 != 0  for the model y = b0 + b1*x1 + b2*x2 + e
>
> ## This will become an important concept for model building, but we can sort of
> ## ignore these for now
>
> anova(myfit1)
Analysis of Variance Table

Response: y
               Df   Sum Sq   Mean Sq     F value     Pr(>F)
x_1             1    577.83    577.83        29.1115    2.033e-05 ***
x_2             1    187.66    187.66        9.4544      0.005542 **
```

```
Residuals 22     436.68   19.85



> ####################################################
> ## 95% confidence intervals for beta_0 and beta_1
>
> confint(myfit1, level=.95)
                  2.5 %       97.5 %
(Intercept)  5.007285   15.5810112
x_1          1.175829    2.5722818
x_2         -1.928271   -0.3748674

> ## Calculate estimates using only matrix algebra
> ## This is just for illustration.  Not useful for actual data analysis
> ## Design Matrix
> X <- cbind(rep(1,n), x_1, x_2)
> X
             x_1          x_2
 [1,] 1 7.693679861 8.1244036
 [2,] 1 6.487045861 6.4009168
 [3,] 1 0.003756292 1.6954951
 [4,] 1 6.793340724 5.1630478
 [5,] 1 7.876556476 2.1379373
 …..
> ## estimated coef
> ##   - %*% means matrix multiplication
> ##   - t(X) means transpose X
> ##   - solve(A) means A inverse
> b <- solve(t(X) %*% X) %*% t(X) %*% y
> b
        [,1]
      10.294148
x_1  1.874056
x_2  -1.151569
> ## Compare to lm fit
> myfit1$coef
(Intercept)        x_1        x_2
  10.294148    1.874056   -1.151569
>> ## Calculate t-tests
> y_hat <- X %*% b
> MSE <- sum((y-y_hat)^2)/(n-3)
> s2_b <- MSE*solve(t(X)%*%X)
>
> ## t stat for beta_0, beta_1, and beta_2
> t_0 <- b[1]/sqrt(s2_b[1,1])
> t_1 <- b[2]/sqrt(s2_b[2,2])
> t_2 <- b[3]/sqrt(s2_b[3,3])
> t_0
[1] 4.038076
```

```
> t_1
[1] 5.566325
> t_2
[1] -3.074807

Coefficients:
               Estimate   Std. Error  t value    Pr(>|t|)
(Intercept)    10.2941    2.5493      4.038      0.00055 ***
x_1            1.8741     0.3367      5.566      1.35e-05 ***
x_2            -1.1516    0.3745      -3.075     0.00554 **


>> ## p-val for beta_0, beta_1, and beta_2
> pval_0 <- 2*(1-pt(abs(t_0), n-3))
> pval_1 <- 2*(1-pt(abs(t_1), n-3))
> pval_2 <- 2*(1-pt(abs(t_2), n-3))
> pval_0
[1] 0.0005496706
> pval_1
[1] 1.35295e-05
> pval_2
[1] 0.005541743
>
> ## CI's for beta_0, beta_1, and beta_2
> CI_beta_0 <- c(b[1]-qt(.975, n-3)*sqrt(s2_b[1,1]), b[1]+qt(.975, n-
3)*sqrt(s2_b[1,1]))
> CI_beta_1 <- c(b[2]-qt(.975, n-3)*sqrt(s2_b[2,2]), b[2]+qt(.975, n-
3)*sqrt(s2_b[2,2]))
> CI_beta_2 <- c(b[3]-qt(.975, n-3)*sqrt(s2_b[3,3]), b[3]+qt(.975, n-
3)*sqrt(s2_b[3,3]))
> CI_beta_0
[1]  5.007285 15.581011
> CI_beta_1
[1] 1.175829 2.572282
> CI_beta_2
[1] -1.9282713 -0.3748674

> ##################################################
>
> ## 95% CI's for E(Y) when (x1,x2)=(1,2), (x1,x2)=(2.5,6), (x1,x2)=(4,8)
> newdata <- data.frame(x_1=c(1, 2.5, 4), x_2=c(2, 6, 8))
> predict(myfit1, newdata=newdata, interval="confidence", level=.95)
       fit        lwr         upr
1 9.865065   5.931824   13.79831
2 8.069871   5.172718   10.96702
3 8.577816   5.214601   11.94103


> ## 95% CI's for E(Y) using Bonferroni Correction
> newdata <- data.frame(x_1=c(1, 2.5, 4), x_2=c(2, 6, 8))
```

```
> predict(myfit1, newdata=newdata, interval="confidence", level=1-.05/3)
       fit          lwr          upr
1  9.865065   4.950657    14.77947
2  8.069871   4.450008    11.68973
3  8.577816   4.375629    12.78000

> #####################################################
>
> ## 95% Prediction Intervals for Y_new when (x1,x2)=(1,2), (x1,x2)=(2.5,6),
> ##  (x1,x2)=(4,8)
>
> newdata <- data.frame(x_1=c(1, 2.5, 4), x_2=c(2, 6, 8))
> predict(myfit1, newdata=newdata, interval="prediction", level=.95)
       fit          lwr          upr
1 9.865065  -0.1768171  19.90695
2 8.069871  -1.6132332  17.75298
3 8.577816  -1.2547948  18.41043
>
> ## 95% PI's for Y_new using Bonferroni Correction
> newdata <- data.frame(x_1=c(1, 2.5, 4), x_2=c(2, 6, 8))
> predict(myfit1, newdata=newdata, interval="prediction", level=1-.05/3)
       fit           lwr          upr
1 9.865065  -2.681817  22.41195
2 8.069871  -4.028734  20.16848
3 8.577816  -3.707591  20.86322

#####################################################
#####################################################
############# Example 2 ##########################
####### Multiple Regression on SENIC data ########
```

```
##outliers
##install package car
install.packages("car")
library(car)

rstudent(myfit2)  ##gives rstudent values
outlierTest(myfit2)  ##Reports the Bonferroni p-value for the most extreme
observation.
qt(1-.05/(2*113),113-4-1)

##leverage, x outliers
lev<-hatvalues(myfit2)
lev
xoutliers <- which(lev > 2*4/113))
xoutliers
lev[xoutliers]
plot(lev)
```