```
rm(list=ls())
# Multiple regression with three predictors

y<-c(3,2,5,7,4,7,7,4,9,5)

n<-length(y)

x0<-rep(1,n)
x1<-c(4,1,8,7,2,5,8,3,7,8)
x2<-c(1,2,3,4,5,10,6,7,8,10)
x3<-c(8,1,9,3,5,3,4,7,9,4)
X<-cbind(x0,x1,x2,x3)
# To visualize the matrix created until now:
cbind(y,X)

# The function dim(matrix) gives as output
# a new matrix containing the dimension of our initial matrix
# In our case, the first row contains
# the number of the rows of X, while the second row
# contains the numbers of columns of X

# It is a way to indicate the number of observed variables
k<-dim(X)[2]-1

# Let's estimate the parameters
XX<-t(X)%*%X
XXinv<-solve(XX)
Xy<-t(X)%*%y
b<-XXinv%*%Xy

round(b,2)
# The equation of the regression line is:
# y=1.23(x0)+0.44(x1)+0.27(x2)+0.05(x3).

# Let's calculate SSreg e SSres,
# by means of the total variance's decomposition
# into (a) its component given by the regression
# and (b) its component given by the residuals , in order to
# verify that all the coefficient are different from zero


# In matrix form, SSreg it is equivalent to:
# b'X'y - [(sum(y))^2/n]
```

```
# or:
# b'X'y - [n*mean(y)^2]
# where b' is the transpose of the coefficients,
# X' is the transpose of matrix X,
# y is the vector of observed responses and n is the number of subjects.

# As said before, (sum(y)) can be defined
# as the element [1,1] of matrix XX,
# namely the element in [the first row - first column].
# The sum of (y) is the the first value of matrix Xy
SSreg <- t(b)%*%Xy - Xy[1,1]^2/n

# Similarly, SSres and Sstot are given by:
SSres <- t(y)%*%y - t(b)%*%Xy
SStot <- SSreg + SSres
cbind(SStot,SSreg,SSres)

# Let's calculate the degrees of freedom
dftot=n-1
dfreg=k
dfres=n-k-1
cbind(dftot,dfreg,dfres)

# The R squared, namely the proportion of explained variance
R2<-SSreg/SStot
adjR2<-1-(1-R2)*(dftot)/(dfres)

# Mean Square
MSreg<-SSreg / dfreg
MSres<-SSres /dfres

# At this point, the F statistic can be calculated
# in two equivalent ways:

# A:
F<-MSreg/MSres
# B:
F<-(R2/dfreg)/((1-R2)/(dfres))

# The function pf(statistic, dofA,dofB)
# gives us back the area under the curve for values between -infinte and F
# for the F distribution
```

```
F.p<-1-pf(F,dfreg,dfres)

# We can use also the further parameter lower.tail=F
# in the way that pf(statistic, dofA,dofB,lower.tail=FALSE)
# gives back the area under the curve for values between +infinite and F

F.p<-pf(F,dfreg,dfres,lower.tail=FALSE)

# In this way, we can find the critical value for  f(dfreg,dfres)

F.crit<- qf(1-.05,dfreg,dfres)
F.crit

# Since the critical value is bigger the our estimated one,
# we cannot (fail to) reject the null hypothesis, stating that
# all the coefficients are equal to zero.

# If we insert all the values in a data frame
# we can summarize all our outputs
data.frame(R2,adjR2,F,dfreg,dfres,F.p)

# Now, let's verify the null hypothesis according to which
# each predictor is equal to zero

# In order to reach this goal, we need the standard error
# of the estimated parameters of the covariance matrix of those parameters.
# Such a matrix can be obtained by multiplying
# the scalar MSres by the inverse of the transpose of X multiplied by X itself

es.b<- sqrt( c(MSres) * diag(XXinv) )

# In this way we insert in a unique matrix all the four different
# values of the standard errors associated to each parameter

# After that, we can calculate the statistic t for each parameter
# and its associated probability
t <- b/es.b

# The function  qt(prob,dof) gives back the t point
# associated to a certain probability with specific dfres

t.crit<-qt(1-.05/2,dfres)
t.crit
```

```
# The function  pt(statistic,dof)
# gives back the area under the curve for values between
# -infinite and t, accordingly with the t-Student distribution.

# For this reason, it is important pass as an argument
# values by which t > 0. As an advice, we can use
# the function abs(x) that calculated absolute values of X.
# The degrees of freedom are the same of those related to MSres,
# since we are facing a bidirectional hypothesis (so, two tails to consider)

t.p<-2*(1-pt(abs(t),dfres))

# Here we see how to calculate the confidence intervals for each b
b.inf <- b - t.crit * es.b
b.sup <- b + t.crit * es.b

# Now we have:
# the values of b,
# the standard error of b,
# the confidence intervals,
# the statistic t and the related probability.

data.frame(b,es.b,b.inf,b.sup,t,t.p)

# As we can see from the three comparisons,
# for each parameter, the statistic t is less the the critical value,
# such as the probability associated to each point is higher than
# the probability indicated by alpha and the null value
# lies inside the proposed interval of confidence


# As usual, let's double-check the results

model<-lm(y~x1+x2+x3)
summary(model)
anova(model)

## to be continued...
```