```
regression = function(yy, XX, nperm=999)
#
# This function computes a multiple regression and tests
# the coefficient of determination (R-square) by permutation.
#
# yy = response vector
# XX = matrix of explanatory variables
# nperm = number of permutations (ex. 99, 499 or 999)
#
# Output:
# - the subfile  $coefficients  contains the regression coefficients
# - the subfile  $adjusted      contains the adjusted values
# - the subfile  $residuals     contains the regression residuals
#
# Pedagogical objectives --
# - illustrate the structure of an R-language function
# - show how to program a loop
# - show how to program a permutation test
# - show how to write out the results of the function
# - show how to output a file containing subfiles
# - revise some important elements of matrix algebra and statistics
#
# Examples of use of the function:
#
#     res = regression(vector_y, matrix_X, 999)
# or  res = regression(nperm=999, XX=matrix_X, yy=vector_y)
#
# Output elements:
#     res$adjusted   # contains the fitted values
#     res$residuals  # contains the residuals
#
#                          Pierre Legendre, February 2005
{
epsilon <- .Machine$double.eps
n = nrow(XX)
m = ncol(XX)
y = as.matrix(yy)
# Add a first column of '1' to matrix XX
# Proceed in two steps: first, create a vectot of '1'; then,
# combine this vector to the data.frame XX, using function 'cbind'
# The resulting matrix X is of size (n x m+1)
vector1 = rep(1,n)
X = as.matrix(cbind(vector1,XX))
#
# Compute the multiple regression
# See Numerical ecology 1998, p. 79, equation 2.19
XprX = t(X) %*% X
if(det(XprX) < epsilon) stop ('Collinearity detected in the explanatory matrix X')
XprXinv = solve(XprX)
# XprXinv = ginv(XprX)
projX = X %*% XprXinv %*% t(X)
yAdjust = projX %*% y
yRes = y - yAdjust
#
# Compute the regression coefficients
# Print results. We will use function 'cat' instead of 'print'
b = XprXinv %*% t(X) %*% y
cat('\n')
cat("Intercept and regression coefficients",'\n','\n')
for(i in 1:(m+1)) { cat('b(',i-1,') =',b[i],'\n') }
cat('\n')
#
# Write the regression coefficients in a matrix of results
```

```
regress.out = matrix(NA,(m+1),1)
colnames(regress.out) = c("Coeff. de regression")
b.names = paste("b.",0:m,sep="")
rownames(regress.out) = b.names
for(i in 1:(m+1))  regress.out[i,1] = b[i]
#
# Compute R-square
# See Numerical ecology 1998, p. 525, equation 10.19
vary = var(y)
varyAdjust = var(yAdjust)
Rsquare=varyAdjust/vary
#
# Compute the adjusted R-square
# See Numerical ecology 1998, p. 525, equation 10.20
# Find the rank of (X'X) in case it is smaller than the order of the matrix
# See Numerical ecology 1998, p. 72-73 and p. 91, 'Second property'
# Beware: matrix X contains a column of '1', which is used to  estimate
# the intercept. That column is already counted in 'mm'
XprX.eig=eigen(XprX)
mm = length(which(XprX.eig$values > sqrt(epsilon)))
# mm=0
# for(i in 1:(m+1)) { if(XprX.eig$values[i] > 0.00000001) mm=mm+1 }
totalDF=n-1
residualDF=n-mm
adjRsq = 1-((1-Rsquare)*totalDF/residualDF)
cat('No. objects =',n,"  rank(X'X) =",mm,'  total d.f. =',totalDF,
    '  residual d.f. =',residualDF,'\n')
cat('R-square=',Rsquare,'    Adjusted R-square =',adjRsq,'\n','\n')
#
# Compute the F statistic
# See Numerical ecology 1998, eq. 4.40 and 11.19
# mm1 = number of independent explanatory variables, not counting the intercept
mm1=mm-1
if(mm1<m) cat('BEWARE: Collinearite detected in matrix XX','\n','\n')
nu1=mm1
nu2=n-mm1-1
F=(Rsquare/nu1)/((1-Rsquare)/nu2)
probF=pf(F,nu1,nu2,lower.tail=FALSE)
#
# Permutation test of R-square. H0: R-square = 0 in the statistical population
# See Numerical ecology 1998, p. 20-26
nPGE=1
for(i in 1:nperm)
   {
   yPerm = sample(y,n)
   yAdjustPerm = projX %*% yPerm
   varyAdjustPerm = var(yAdjustPerm)
   RsquarePerm = varyAdjustPerm/vary
   FPerm=(RsquarePerm/nu1)/((1-RsquarePerm)/nu2)
   if(FPerm >= F) nPGE=nPGE+1
   }
P=nPGE/(nperm+1)
cat('F =',F,'   Prob (param) =',probF,'  Prob(',nperm,'permutations) =',P,'\n','\n')
#
return(list(coefficients=regress.out, adjusted=yAdjust, residuals=yRes,
Rsquare=Rsquare, F=F, p.param=probF, p.perm=P, nperm=nperm))
}
```