# PRÀCTICA: REGRESSIÓ LINEAL

## PREPARAR L'ENTORN R I LES DADES

Engegar el programa R

Menú: `Archivo->Cambiar dir->Browse->D:\AAMD` (si no existeix el directori `D:\AAMD` el crearem de nou)

De la pàgina de l'assignatura al Campus Virtual UB, baixeu els fitxers

`UScensus.r, Polynomial.Regression.r, Quantitats.Regressio.r.`

`Acetylene.dat.txt, Acetylene.txt, Steam.dat.txt, Steam.txt, Steam1.txt.`

i poseu-los al directori `D:\AAMD`.

Apart d'aquests fitxers de dades externs, carregarem dos fitxers de dades que vénen en el package datasets del R:

```
data(cars)
```

```
data(longley)
```

Aquestes instruccions creen dos `data.frame` amb els mateixos noms que els datasets.

Feu `?cars, ?longley,` per veure el help sobre aquests fitxers. Feu `str(cars), str(longley)` per comprovar que s'han creat correctament els dos dataframes i veure la seva estructura.

Carregarem el package DAAG fent

```
library(DAAG)
```

Aquest és el paquet de funcions i jocs de dades corresponent al llibre: Maindonald, J., Braun, J. (2003) *Data Analysis and Graphics Using R*. Cambridge University Press.

També farem servir els package MASS i ISLR, corresponents, respectivament, als llibres: Venables, W. N., Ripley, B. D. (2002), *Modern Applied Statistics with S,* James, G., Witten, D., Hastie, T., Tibshirani, R. (2013), *An Introduction to Statistical Learning with Applications in R.* Els carreguem amb:

```
library(MASS), library(ISLR)
```

Carregueu les dades `hills` i mireu el seu help:

```
data(hills)
```

```
?hills
```

## REGRESSIÓ SIMPLE I MÚLTIPLE

La funció lm() serveix per ajustar els models lineals amb R.

Provem en primer lloc una regressió simple:

```
x<-cars$speed
```

```
y<-cars$dist
```

```
l1<-lm(y~x)
```

produeix `l1`, un objecte de la classe lm, que conté tota la informació sobre la regressió de y sobre x. Una sintaxi millor, sense definir variables globals, és:

```
l2    <-lm(dist~speed,data=cars)
```

A partir d'un objecte de la classe lm es poden extreure els resultats numèrics i gràfics que en necessitin de la regressió. Un resum s'obté fent:

```
l2
```

O amb més detall:

```
summary(l2)
```

Fem el diagrama de punts, i li afegim la recta de regressió:

```
plot(dist~speed,data=cars)
abline(l2)
```

Ara llegim les dades dels fixers:

```
Acetylene<-read.table("Acetylene.txt",header=TRUE)
Steam<-read.table("Steam1.txt",header=TRUE)
```

Per a fer regressió múltiple:

```
l2<-lm(Y~X1+X2+X3,data=Acetylene)
l3<-lm(y~.,data=Steam)
```

Per a fer predicció de noves dades, per exemple:

```
newy<-
predict(l2,newdata=list(X1=1100,X2=8,X3=0.02),
type="response")
```

## REGRESSIÓ POLINÒMICA, NÚMERO DE CONDICIÓ

La funció `kappa()` calcula el número de condició d'una matriu.

Comproveu el número de condició de la matriu de la regressió amb les dades `Acetylene`, resultat d'afegir una columna de uns a les X1, X2, X3:

```
Xa<-Acetylene[,-4]
na<-nrow(Xa)
Xa1<-cbind(rep(1,n),Xa)
kappa(Xa1)
```

Aquestes dades tenen mala condició, estan preparades per un exemple de regressió ridge. També, amb les dades `longley`.

Feu `File->SourceRcode` amb els fitxers `UScensus.r` i `Polynomial.Regression.r`. La primera operació crea les variables `t, p` de l'exemple del cens US.

Amb la funció `Polynomial.Regression()` podeu preparar la matriu de regressió d'una regressió polinòmica de grau `k` a partir d'un vector (per exemple la `t` de les dades `UScensus`). Calculeu el número de condició i intenteu fer la regressió per a valors creixents de `k`.

## QUANTITATS IMPORTANTS DE LA REGRESSIÓ

Per motius del model físic del problema és més oportú treballar amb els logaritmes de les dades `hills`. Ho implementem calculant un nou `data.frame` amb els log, posant noms adequats a les noves variables.

```
loghills<-log(hills)
```

```
names(loghills)<-c("logdist","logclimb","logtime")
```

Es guanya gaire de cara a la regressió? Compareu:

```
plot(hills)
```

```
plot(loghills)
```

```
cor(hills)
```

```
cor(loghills)
```

```
l1 <- lm(time ~ dist + climb, data = hills)
```

```
logl1 <- lm(logtime ~ logdist + logclimb, data = loghills)
```

Anem a calcular quantitats importants de la regressió:

```
y<-hills$time
```

```
y0<-y-mean(y)
```

```
TotalSS0<-sum(y0^2)

n<-nrow(hills)

Totaldf<-n

Totaldf0<-n-1
```

Amb les dades ajustades:

```
yhat<-as.numeric(l1$fitted.values)

yhat0<-yhat-mean(yhat)

RegSS0<-sum(yhat0^2)

Regdf<-3

Regdf0<-2
```

Comproveu que mean(y) és igual a mean(yhat). Els residus de la regressió els podeu extreure:

```
ytilde<-as.numeric(l1$residuals)
```

També com `y-yhat` o com `y0-yhat0`. Els residus ja venen centrats.

```
ResSS<-sum(ytilde^2)

Resdf<-Totaldf0-Regdf0
```

Les mitjanes dels quadrats s'obtenen dividint cada suma de quadrats pel seu nombre de graus de llibertat.

```
TotalMeanS0<-TotalSS0/Totaldf0

RegMeanS0<-RegSS0/Regdf0

ResMeanSS<-ResSS/Resdf
```

Ara podeu verificar que el coeficient de determinació de la regressió (Multiple R-squared) és igual a:

```
R2<-RegSS0/TotalSS0
```

I que l'estadístic F que surt al summary és:

```
RegMeanS0/ResMeanS
```

En principi aquest quocient és una mesura de quan gran és la mitjana dels quadrats de la regressió comparada amb la mitjana dels quadrats residuals. Si la regressió és normal Gauss-Màrkov, aquest estadístic segueix una distribució F de Fisher-Snedecor i el p-valor que surt està calculat amb aquesta regressió.

La funció

```
anova(l1)
```

treu aquestes sumes i mitjanes de quadrats, descomposades per a cada predictor individual.

## COMPARACIÓ DE REGRESSIONS D'UNA MATEIXA RESPOSTA AMB MÉS O MENYS PREDICTORS

Les tres quantitats de què disposem de moment per comparar diferents regressions són: La suma de quadrats residual ResSS, el coeficient de determinació R2 i l'estadístic F.

A aquestes hi podeu afegir l'estadístic AIC (Akaike Information Criterion). El podeu obtenir amb:

```
extractAIC(l1)
```

Comproveu que:

```
n*log(ResSS/n)+2*Regdf
```

És a dir, es tracta d'una versió penalitzada del logaritme de la suma de quadrats residual.

Ara, amb les dades `Steam`, calculeu diverses regressions, començant amb tots els predictors:

```
ls1<-lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9, data=Steam)
```

I seguint, amb menys predictors, comparant les quantitats citades més amunt.

Les quantitats de què disposem per comparar diferents regressions són:

1. La suma de quadrats residual ResSS,
2. El coeficient de determinació R2
3. L'estadístic F.
4. L'estadístic AIC (Akaike Information Criterion).

La suma de quadrats residual d'un model és una mesura d'ajust: més petita com millor l'ajust del model. Ara bé, per comparar dos models per a una mateixa resposta, si el segon model s'ha obtingut afegint un o més predictors al primer model, inevitablement la suma de quadrats residual del segon model és més petita, per la geometria (estem projectant sobre un subespai més gran, que conté el primer). El mateix es pot dir del coeficient de determinació. Des d'un punt de vista estadístic, el que es tracta de fer és decidir si aquesta disminució de la suma de quadrats residual, o l'augment del coeficient de determinació és significatiu. Això és el que es proposa fer l'estadístic F.

Si, partint d'un model L0, arribem a un model L1 afegint-hi un predictor, l'estadístic F de comparació dels dos models és el quocient:

```
F01<-(ResSS0-ResSS1)/(ResSS1/df1)
```

on `ResSS0` i `ResSS1` són les respectives sumes de quadrats residuals i `df1` és el nombre de graus de llibertat de `ResSS1`. `F01` és un estadístic de test de la hipòtesis nul·la

H0 = "El nou predictor no millora l'ajust del model".

Si el model és normal Gauss-Màrkov, `F01` segueix una distribució F amb 1 i df1 graus de llibertat.

L'estadístic AIC *(Akaike Information Criterion)* respon a un altre problema: en afegir un predictor estem afegint un paràmetre i, en general, l'augment de paràmetres d'un model estadístic tendeix a ajustar millor la mostra d'aprenentatge, en detriment de l'ajust a la població d'on s'ha extret aquesta mostra. Un exemple més del conflicte biaix/variància. El AIC és el (–log) d'una versemblança

penalitzada, que permet comparar dos models “aniuats” (terme incorrecte, volent dir un inclòs dins de l’altre). El model millor és el que té el AIC més petit, cosa que equival a un màxim de la versemblança penalitzada. El AIC d’un model L el podeu obtenir amb:

```
extractAIC(L)
```

Comproveu la fórmula de la pàgina 153 de DAAG:

```
n*log(ResSS/n)+2*Regdf
```

Recuperem les dades per a regressió de la secció anterior:

```
loghills<-log(hills)
names(loghills)<-c("logdist","logclimb","logtime")
```

Calculem les regressions:

```
l0 <- lm(logtime~logdist,data=loghills)
l1 <- lm(logtime~logdist+logclimb,data=loghills)
```

Amb la funció `Quantitats.Regressio`, que prepareu amb:

```
source("Quantitats.Regressio.r")
```

podeu calcular les quantitats rellevants de totes dues regressions:

```
Q0<-Quantitats.Regressio(l0)
Q1<-Quantitats.Regressio(l1)
```

Decidiu quin dels dos models és el millor, segons els criteris que hem vist. El package `stats` té dues funcions, `add1` i `drop1`, que permeten fer aquesta operación de manera semiautomàtica:

```
add1(l0,"logclimb")
add1(l0,"logclimb",test="F")
add1(l0,"logclimb",test="Chisq")
drop1(l1,"logclimb")
drop1(l1,"logclimb",test="F")
```

```
drop1(l1,"logdist")

drop1(l1,"logdist",test=”F”)
```

Amb dades com les `Steam`, amb molts predictors, hi ha moltes possibles regressions –exactament $2^p$, si hi ha p predictors, tantes com subconjunts té el conjunt de predictors- que es poden provar aplicant repetidament les funcions anteriors, sigui començant amb el model que té només intercept:

```
ls0<-lm(y~1, data=Steam)
```

i anant afegint predictors, o bé  començant amb tots els predictors:

```
ls1<-lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9, data=Steam)
```

I anar suprimint-los. Per exemple, afegim, successivament cadascuna de les variables a `ls0`:

```
add1(ls0,"x1",test="F")

   ...

add1(ls0,"x9",test="F")
```

Comprovem que `x7` és la millor addició. Ens quedem amb aquesta variable i ara anem a afegir una segona, pel mateix procediment, i així successivament, fins a obtenir un subconjunt satisfactori. Aquest és el procediment “forward” de selecció de variables. El procediment “backward” parteix de `ls1` i consisteix a anar treient variables de forma anàloga.

Cal tenir en compte que si, per exemple, en certa etapa del procediment “forward”, hem arribat a un subconjunt A de predictors, no hem provat tots els subconjunts de A, o sigui que partint de A hauríem d’aplicar “backward” i així successivament. El package stats té la funció step que permet automatitzar aquest procés:

```
step(ls0,"~x1+x2+x3+x4+x5+x6+x7+x8+x9")

step(ls1)
```

Aquest procediment a passes individuals s’anomena, lògicament, “stepwise”.

En el package MASS hi ha `addterm`, `dropterm`, i `stepAIC`, que tenen la mateixa funcionalitat que les que hem vist aquí. La funció `step` és una versió simplificada de `stepAIC`.

Finalment, hi ha la possibilitat de calcular totes les $2^p$ regressions possibles, amb tots els subconjunts de 0 al total de p predictors, comparar directament tots els AIC obtinguts, quedant-nos, ara sí, amb el millor subconjunt. Això és el que s'anomena "All Subsets Regression". Naturalment, aquest càlcul és impracticable mitjançant força bruta, a partir de no gaires predictors. Hi ha però algorismes que permeten limitar el nombre de càlculs reals que cal fer, ordenant parcialment el conjunt de totes les regressions potencials en una estructura en forma d'arbre, del qual és suficient avaluar alguns nodes. Aquest procediment el teniu implementat al package `leaps` (el nom vé de l'expressió "leaps and bounds" ).