

```

#COST Action ES0601 - HOME. Patras R training course (7-10 Nov., 2009).
# R notes. To be complemented with the HTML help, with examples and 'see also'.

#Example data files:
#data1.txt Monthly maximum mean temperatures (1961-2002)
#data2.txt Daily maximum temperatures of tree years

#===== Installation and documentation of R:

http://cran.r-project.org/
http://wiki.r-project.org/

#===== Starting R =====

#Interactive mode:
R

#Batch mode (put --save if we want to save the state at exit):
R --no-save < FICH_ORDENES > FICH_SALIDA

#In both cases the file .Rprofile is read if it exists, and if there is a
#function .First in it, it is executed. Example of .Rprofile:
.First <- function() {
  source("initfunctions.R")
  cat("\n initfunctions.R file has been read\n")
}

q() #exit from R.

#===== General functions =====

ls() #list memory objets (variables, functions, etc)
rm(VAR) #deletes variable VAR
rm(list=ls(all=TRUE)) #deletes all variables de la memoria (even the hidden
#ones, beginning by '.')

x <- c(-3:2,7,5,2:-1) #create a vector
x #displaying it
length(x)
plot(x) #basic plot
#Other plot arguments:
# xlim, ylim, main, xlab, ylab, col,
# type="T" with T= p(points), l(lines), b(both), c(as b, but only the lines),
# o(as b, but 'overplotted'), h(similar to a histogram), s(steps), S(forward
# steps), n(don't plot: compute the axis only)
# bty="o|l|7|c|u||n" border type (as the capital leters, or none)
barplot(x)
boxplot(x,col="green")
grid()
grid(col=grey(.4))
boxplot(x,col="green",add=TRUE)

m <- matrix(rnorm(5*18,10,2),18,5) #matrix of normally distr. radom numbers
m
round(m,2)
boxplot(m,col=5) #only 1 box!
boxplot(as.data.frame(m),col=5) #5 boxes
attributes(m)
t <- read.table("data1.txt") #reading from a file
t
attributes(t)
boxplot(t,col=6,main="Average maximum temperatures",ylab="°C",xlab="Months")
grid(col=grey(.5))

#===== Object manipulations

```

```

#Conversion: as.matrix, as.vector, etc
#Identification: is.na, is.character, etc
#Manipulation:
length(x)
sort(x)
rank(x)
order(x)
rev(x) #reverse order
t(m) #matrix transposition
range(x)
max(x)
min(x)
which.max(x) #first occurrence
which.min(x) #first occurrence
which(x==max(x)) #all occurrences
which(x<0)
which(x==1)

y <- c(1:4,1:2,-3,2,7,-1) #another vector

union(x, y)
intersect(x, y)
setdiff(x, y) #elements of x that are also in y
setdiff(y, x) #elements of y that are also in x
setequal(x,y) #are x and y equal?

a <- round(m) #copy the matrix
attr(a,"dim") #dimensions of a
attr(a,"dim") <- c(5,6,3) #change to a 3-dimensional array

a <- array(1:24,c(2,4,3)) #or we can create it directly
a
apply(a,1,sum)
apply(a,c(1,3),sum)

#factors can be very useful to stratify our data:
z <- as.vector(as.matrix(t[,6:8])) #summer monthly maximum mean temperatures
f <- factor(z>28)
tapply(z,f,mean)
f <- factor(cut(z,breaks=24:33))
tapply(z,f,mean)

#subsetting data. Besides the subscripting procedures, you can use subset():
subset(t,t$Aug>=30)
subset(t,t$Aug>=30,select=3:5) #id., selecting the spring months

#Selection of records of a data frame by means of regular expressions:
words <- c("Flea","Floor","Lion","Flower","Failed")
grep('^F',words)
grep('^Fl',words)
grep('^..o',words)
#grep can then be used for the selection. E.g.: If we had a data frame MyDat
#containing a column named WORD with character strings, we could select all
#records with the character 'o' in the third place by using:
# MyDat[grep('^..o', MyDat$WORD),]

#===== Defining functions

#Basic model, with parameters and their default values:
myfunct <- function(par1=defv1, par2=defv2, ...) {
  ... #put here your loops, computations, etc
  ...
}

#Example: Fourier decomposition of a time series. It returns the residuals
#of the adjustment, and creates 'fourier.fitted' with the fitted values.

```

```

fourier <- function(y,na=5,nc=1,detrend=TRUE) {
  #na: number of harmonics; nc=number of cycles in the series
  n <- length(y)
  f <- 2*pi/n
  x <- matrix(0,n,2*na)
  for(j in 1:na) {
    for(i in 1:n) {
      x[i,j*2-1] <- sin(i*j*nc*f)
      x[i,j*2] <- cos(i*j*nc*f)
    }
  }
  if(detrend) x <- cbind(x,1:n)
  aj <- lm(y~x,na.action="na.exclude")
  print(summary(aj))
  #very important! All objects created inside this function are local, and
  #they will not be preserved after the function completion, unless you
  #assign them with '<<-':
  fourier.fitted <- fitted.values(aj)
  return(residuals(aj))
}

tx <- scan("data2.txt") # 3 year daily maximum temperatures
plot(tx,type="l")

rtx <- fourier(tx,nc=3)
lines(mean(tx,na.rm=TRUE)+rtx,col="red") #add the mean value!
lines(fourier.fitted,col="blue",lwd=2)

#Exercise: build a function to compute the Euclidean distance between two points

#Possible solution:
eudist <- function(p1,p2) {
  dx <- p1[1]-p2[1]
  dy <- p1[2]-p2[2]
  sqrt(dx*dx+dy*dy)
}
#Application:
c1 <- c(0,1) #coordinates of point 1
c2 <- c(1,0) #coordinates of point 2
eudist(c1,c2)

#==== Statistical functions

#Basic functions:
mean(z)
max(z)
min(z)
var(z)
sd(z)
summary(z)
stem(z) #stem & leaf output

#Count numbers of occurrences of each value:
summary(as.factor(x))

h <- hist(z) #histogram (plot)
h #if we store the results of hist, we have access to many info as, e.g.:
h$counts #frequencies

#Available probability distributions:
# beta, binom, cauchy, chisq, f, gamma, geom, hyper, lnorm, logis, nbinom,
# norm, pois, t, unif, weibull, wilcox
#We can prefix each one with any of the letters d (prob. density),
# p (distribution function), q (quantiles), and r (random values)

pnorm(1:4)

```

```

qnorm(.75)
plot(.1*-30:30,dnorm(.1*-30:30),type="l")
plot(.1*0:60,dweibull(.1*0:60,1.5,1.5),type="l")

#quantiles:
quantile(tx,na.rm=TRUE)
quantile(tx,probs=c(.25,.5,.7,.9,.95),na.rm=TRUE) #choosing the prob. levels
#empirical cumulative distribution function:
dtx <- ecdf(tx)
plot(dtx, do.points=FALSE, verticals=TRUE)

#Maximum likelihood adjustment to a probability distribution:
library(MASS)
fitdistr(t[,3],"weibull")

t[,3]
sample(t[,3],14,replace=T) #sampling

#===== Linear models

#General form: lm(y~x)

#let's simulate 2 correlated variables:
x <- rnorm(50,30,3)
y <- x+rnorm(50)+2
#simple regression:
aj <- lm(y~x)
aj
summary(aj)
plot(aj)
#look at the cloud of points (x,y):
plot(x,y,col="blue")
abline(aj,col="red") #add the fitted line
#other ways of adding straight lines:
abline(0,1,lty=2) #identity line
abline(h=26,lty=2,col="green") #horizontal line
abline(v=27,lty=2,col="green") #vertical line

confint(aj,level=.90) #confidence interval
residuals(aj)
fitted(aj)

#model with no intercept:
aj <- lm(y~0+x)
summary(aj)
#It appears to be better, but notice this note found in the Internet:
* For technical reasons the R-squared statistic for regressions
through the origin is not comparable with R-squared values
obtained from a regression with intercept. The same
comment applies to the p-value for overall model fit.
* So, be very wary about fitting regressions through the origin
unless you have a very strong scientific reason for doing so.

#multiple regression:
aj <- lm(t$Aug~as.matrix(t[,1:5]))
summary(aj)

#Several linear models at a time:
aj <- lm(as.matrix(t[,6:8])~as.matrix(t[,1:5]))
aj
summary(aj)

#let's simulate a random variable depending on 10 correlated variables:
y <- rnorm(50,30,3) #dependent var.
z <- matrix(0,50,10)

```

```

for(i in 1:10) z[,i] <- y+rnorm(50,0,i/10)
#correlation matrix:
round(cor(cbind(y,z)),2)
#multiple regression:
aj <- lm(y~z)
summary(aj)
#stepwise selection of the significant variables:
aj2 <- step(aj,scope=list(lower=y~1,upper=y~.),direction="both")
#(doesn't seem to work here!)

#===== More on graphics

#plotting data frames:
plot(t[,3:5])

#Other graphic parameters, set inside par():
las=0|1|2|3 #label orientation: paralel to axis, horiz., perpendicular, vert.
oma=c(d,l,u,r) #outer margins, in lines (down, left, upper, right)
omd=c(d,l,u,r) #id., in fractions (0 to 1)
pty="s|m" #plot region: s=square, m=maximal (def.)
xaxp=c(x1,x2,n) #extrem x tics, and number of intervals
xaxs="r|i" #x axis size: regular, or internal (without a 4% increase)
xaxt="n"... #x axis type: don't draw the axis, ...(etc: see help page)

#Colors and color palettes: colors, rgb, hsv, gray, heat.colors,
#terrain.colors, topo.colors, rainbow, palette.

#Guide for hue values:
# 0=red, 1/6=yellow, 2/6=green, 3/6=cyan, 4/6=blue, 5/6=magenta

#Other examples of 2D plots: matplot, points, poligon, pie, stars, ...,
# rosavent, diagw1.

#Copy your graphic to a PDF file
grabpdf <- function(file) {
  if(dev.next()==dev.cur()) {
    pdf(file,onfile=TRUE)
    dev.set(dev.prev())
  }
  dev.copy(which=dev.next())
  dev.off()
}

#Exercise:
# 1.- Compute the quantiles of t for the period 1971-2000.
# 2.- Print them on screen with the function print.
# 3.- Plot them with lines, and overplot the 2002 montly values to visually
# estimate if the temperatures of that year were extremelly warm, very warm,
# warm, normal, cold, very cold or extremelly cold.
qt <- matrix(0,12,6)
for(i in 1:12) qt[i,] <- quantile(t[11:40,i],probs=c(.05,.2,.4,.6,.8,.95))
qt <- as.data.frame(qt)
names(qt) <- as.character(c(.05,.2,.4,.6,.8,.95))
attr(qt,"row.names") <- names(t)
print(qt)
matplot(qt,type="l",lty=1,col=rainbow(6,start=.7,end=0),xlab="Months",ylab="Maximum
average temperatures (°C)",main="Characterization of 2002 monthly maximum average
temperatures",lab=c(12,5,1), xaxt='n')
axis(1, 1:length(names(t)), as.character(names(t)))
lines(1:12,t[42,],lwd=2,lty=2)
grid(col=grey(.5))

#3D plots: contour, image, persp.

#Example (from the internet) of orography with colors varying with height:

```

```

data(volcano)
z <- 2 * volcano          # Exaggerate the relief
x <- 10 * (1:nrow(z))    # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z))    # 10 meter spacing (E to W)

z0 <- min(z) - 20
z <- rbind(z0, cbind(z0, z, z0), z0)
x <- c(min(x) - 1e-10, x, max(x) + 1e-10)
y <- c(min(y) - 1e-10, y, max(y) + 1e-10)

fill <- matrix("green3", nr = nrow(z)-1, nc = ncol(z)-1)
fill[ , i2 <- c(1,ncol(fill))] <- "gray"
fill[i1 <- c(1,nrow(fill)) , ] <- "gray"

fcol <- fill
zi <- volcano[ -1,-1] + volcano[ -1,-61] +
      volcano[-87,-1] + volcano[-87,-61] # / 4
fcol[-i1,-i2] <-
  terrain.colors(20)[cut(zi, quantile(zi, seq(0,1, len = 21)),
                        include.lowest = TRUE)]

par(mar=rep(.5,4))
persp(x, y, 2*z, theta = 110, phi = 40, col = fcol, scale = FALSE,
      ltheta = -120, shade = 0.4, border = NA, box = FALSE)

# Generate a grid from irregularly distributed data:
xyz <- matrix(c(runif(10,0,10),runif(10,0,10),rnorm(10,20,5)),10,3)
xyz
plot(xyz[,1:2])
xyz <- as.data.frame(xyz)
names(xyz) <- c("x","y","z") #columns MUST have these names!
library(spatial)
t.kr <- surf.gls(4, expcov, xyz, d=0.7)
t.surf <- prmat(t.kr,0,10,0,10,10) #xl,xu,yl,yu,n ##ERROR!
contour(t.surf)

#===== Statistical tests

#Chi2 test for a contingency table:
fr <- matrix(c(25,37,12,47,52,17),2,3,byrow=TRUE)
fr
chisq.test(fr)

z <- t$May

t.test(z[1:21],z[22:42]) #Diference of means (Student's t-test)
var.test(z[1:21],z[22:42]) #Diference of variances (Snedecor's F)
cor.test(t$May,t$Aug) #Correlation test ("pearson", "kendall", and "spearman")
#Don't trust significant correlations if the series have trends!:
x <- rnorm(100); y <- rnorm(100)
matplot(cbind(x,y),type="l")
cor.test(x,y)
plot(x,y)
x2 <- x-1:100/20; y2 <- y+1:100/30
matplot(cbind(x2,y2),type="l")
cor.test(x2,y2)
plot(x2,y2)

#Other tests:
binom.test
fisher.test
friedman.test
kruskal.test
mantelhaen.test
mcnemar.test
prop.test
wilcox.test

```

```

chisq.gof
ks.gof #Kolgomorov-Smirnov goodness of fit

#===== Time series analysis

plot(tx,type="l")
ftx <- filter(tx,rep(1/10,10)) # 10 terms running means
lines(ftx,col="red")
#filter giving more weight to the central terms:
filter(tx,c(.1,.2,.4,.2,.1))

#Autocorrelation function (with graphics):
acf(tx,col="blue",ci.col="red",lwd=3,,na.action=na.pass)
#Partial autocorrelation function (again with graphics):
pacf(tx,col="blue",ci.col="red",lwd=3,na.action=na.pass)

#Test of independence of a temporal series:
Box.test(as.ts(tx))

#Cross-correlation function (with graphics):
ccf(t$May,t$Aug,col="blue",ci.col="red",lwd=3)

cpgram(tx) #cumulative periodogram:

#frequency spectrum:
spectrum(tx) #with logarithmic scale
spectrum(tx,log="no") #without logarithmic scale

#Adjusting an autoregressive model:
art<-ar(tx)
art
#Generate 5 terms forecasted by the model:
tpred<-predict(art,n.ahead=5)
tpred$pred #display the forecasted values
#Plot the last 50 observations with the forecasted values:
n <- 50
n1 <- length(tx); n2 <- length(tpred$pred)
plot(tx[(1+n1-n):(n1+n2)],type="l",ylab="Max. average temp.
(°C)",main="Observations+predictions")
lines(n:(n+n2),c(tx[n],tpred$pred),col="red")
grid(col=grey(.5))

#ARIMA models ajustement:
arima(tx)

#Discrete differenciation and integration:
z <- t$Mar[1:12] # 12 first March temperatures
z #display the values
dz <- diff(z) #first difference series
dz #display the values
diffinv(dz) #By default, it beguins by 0. Provide the initial value:
diffinv(dz, xi=17) #now the original z series has been reconstructed

#Seasonal Decomposition of Time Series by Loess:
stl(as.vector(as.matrix(t)),"per") #ERROR!
#Use the fourier() function builded before...

#===== Multivariate analysis

#Principal Component Analysis:
c<-princomp(t) #covariance matrix is used by default
summary(c)
c<-princomp(t,cor=TRUE) #now we use the correlation matrix
summary(c,loadings=TRUE)
plot(c) #A "scree plot" is displayed
plot(c,npcs=12) #id., but with all the components

```

```

plot(c,npcs=12,type="l") #less appealing, but more classic

#A more elaborate version:
nv <- length(t) #number of variables
var=(c$sdev*c$sdev)*100/nv #variance explained by each component (in %)
vara <- var #cumulated variances:
for(i in 2:nv) vara[i] <- vara[i]+vara[i-1]
plot(1:nv,var,type="l",lwd=2,col="blue",ylim=c(0,100),xlab="Components",ylab="Variance (%)",main="Explained variance") #Scree plot
lines(vara,lwd=2,col="blue")
abline(100/nv,0,col="red") #add initial variance as a reference
grid()
points(var)
points(vara)

biplot(c) #'biplot', with cases (points) and variables (axis)
biplot(c,xlabs=rep("+",length(t[,1]))) #more neat, though less informative
abline(h=0)
abline(v=0)
biplot(c,choices=c(2,3),xlabs=rep("+",length(t[,1]))) #id., for comp. 2 and 3
abline(h=0)
abline(v=0)

matplot(c$loadings[,1:4],xlab="Original variables",main="Loadings of PC 1-4")
abline(h=0,lty=2)
v <- varimax(c$loadings[,1:4]) #axis rotation retaining 4 components
matplot(v$loadings[,1:4],xlab="Original variables",main="Loadings of rotated PC 1-4")
abline(h=0,lty=2)

#See also Canonical Correlation Analysis (cancor function) and alike.

#==== Cluster analysis

di <- dist(t) #disimilarity matrix by Euclidean distance
round(di,1) #display
h <- hclust(di,method="ward") #hierarchical classification by Ward method
plot(h) #dendrogram of years grouped by their monthly values similarities
#Another example: Grouping months by their correlation similarity
di <- 1-cor(t,use="p") #(high correlations mean more similarity!)
hc <- hclust(as.dist(di),method="ward")
plot(hc,xlab="Stations",sub="",ylab="Disimilarity (1-r)",hang=-1)

#Visualization of the differences between years by means of the stars function:
stars(t)

#==== End of these notes. See provided and web documentation, and enjoy... :-)
```