```
### $Id: g.test.R,v 1.0 2001/11/08
###
### G test for R
###
### Copyright 2001 José F. Calvo & José A. Palazon
### <jfcalvo@um.es> <palazon@um.es>
###
### This file is made available under the terms of the GNU General
### Public License, version 2, or at your option, any later version,
### incorporated herein by reference.
###
### This program is distributed in the hope that it will be
### useful, but WITHOUT ANY WARRANTY; without even the implied
### warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
### PURPOSE. See the GNU General Public License for more
### details.
###

g.test <- function(x){

sum2c<-0
sum2r<-0

nr<-nrow(x)
nc<-ncol(x)

sum1<-sum(x*log(x))

for(i in 1:nr){
sum2r<-sum(x[i,])*log(sum(x[i,]))+sum2r
}

for(i in 1:nc){
sum2c<-sum(x[,i])*log(sum(x[,i]))+sum2c
}

sum2<-sum2r+sum2c
sum3<-sum(x)*log(sum(x))

g<-2*(sum1-sum2+sum3)

df<-(nr-1)*(nc-1)

pval<-1-pchisq(g,df)

DNAME <- deparse(substitute(x))
METHOD <-"G test"
names(g) <- "G"
names(df) <- "df"
structure(list(statistic = g, parameter = df, method = METHOD,
       p.value = pval, data.name = DNAME), class = "htest")

}
```

```
### $Id: acb.R,v 1.0 1999/12
###
### Análisis de correspondencias binarias para R
### Correspondence analysis for R
###
```

```
### Copyright 1999 José A. Palazon & José F. Calvo
### <palazon@um.es> <jfcalvo@um.es>
###
### This file is made available under the terms of the GNU General
### Public License, version 2, or at your option, any later version,
### incorporated herein by reference.
###
### This program is distributed in the hope that it will be
### useful, but WITHOUT ANY WARRANTY; without even the implied
### warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
### PURPOSE. See the GNU General Public License for more
### details.
###

acb <- function (x) {

x<-as.matrix(x)

traspuesta<-F
if(nrow(x)>ncol(x)){x<-t(x);traspuesta<-T}

diag(1/sqrt(apply(x,2,sum))) ->C

diag(1/sqrt(apply(x,1,sum)))->R

R %*% x %*% C -> M

eigen(M%*%t(M)) -> vp

sqrt(diag(vp$va))  ->a

sqrt(sum(x))* R %*% vp$ve  %*% a ->V

diag(1/apply(x,2,sum)) ->C

diag(1/sqrt(vp$va))  -> a

C %*% (t(x) %*% V) %*% a -> W

dimension <- min(dim(x))

V <- round(V[,2:dimension],3)
W <- round(W[,2:dimension],3)

if(!traspuesta){y<-V;V<-W;W<-y;x<-t(x)}

valores <- vp$va[2:dimension]
s <- sum(valores)
acumval <- round( valores/s*100,1)
valores <- round(valores,3)

colnames(W)<-paste("Eje",1:dim(W)[2])
colnames(V)<-colnames(W)
rownames(W)<-colnames(x)
rownames(V)<-rownames(x)

list(
valorespropios=valores, #vp$va,
#vectorespropios=vp$ve,
inercia=acumval,
ccolumnas=V,
cfilas=W
)
}
```

```
### $Id: acp.R,v 1.0 1999/12
###
### Análisis de componentes principales para R
### Principal components analysis for R
###
### Copyright 1999 José A. Palazon & José F. Calvo
### <palazon@um.es> <jfcalvo@um.es>
###
### This file is made available under the terms of the GNU General
### Public License, version 2, or at your option, any later version,
### incorporated herein by reference.
###
### This program is distributed in the hope that it will be
### useful, but WITHOUT ANY WARRANTY; without even the implied
### warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
### PURPOSE. See the GNU General Public License for more
### details.
###

acp <- function (x) {

x<-as.matrix(x)

eigen(var(x)) -> vp

diag( sqrt(vp$va), dim(x)[2])  -> a

vp$ve %*% a -> V

(x %*% vp$ve) -> W

V <- round(V,3)
W <- round(W,3)
valores <- vp$va
s <- sum(valores)
acumval <- round( valores/s*100,1)
valores <- round(valores,3)

colnames(W)<-paste("Eje",1:dim(W)[2])
colnames(V)<-colnames(W)
rownames(V)<-colnames(x)
rownames(W)<-rownames(x)

list(
valorespropios=valores, #vp$va,
#vectorespropios=vp$ve,
inercia=acumval,
ccolumnas=V,
cfilas=W
)
}



### Utilidades para la asignatura Ecología Metodológica y Cuantitativa
###
### Copyright 2001 José A. Palazon & José F. Calvo
### <palazon@um.es> <jfcalvo@um.es>
###
### This file is made available under the terms of the GNU General
### Public License, version 2, or at your option, any later version,
### incorporated herein by reference.
```

```
###
### This program is distributed in the hope that it will be
### useful, but WITHOUT ANY WARRANTY; without even the implied
### warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
### PURPOSE. See the GNU General Public License for more
### details.
###


# Dibuja ejes de coordenadas

dejes <- function() {abline(h=0,v=0,col="grey")}

dxycn <- function (x) {
plot (x,type="n")
text(x,rownames(h))
}

rotacion <- function() {
h<-0.866027/2
b<-1/2
m<-c(-b,-h, b,-h, 0,h, -b,-h);dim(m)<-c(2,4);m<-t(m)
a<-0.8
for(alfa in seq(0,2*pi+0.05,0.05))
{r<-c(cos(alfa),sin(alfa),-sin(alfa),cos(alfa));
dim(r)<-c(2,2);
mz<-m%*%r
plot(mz,xlim=c(-a,a),ylim=c(-a,a),type="l")}
}

leefichero<- function(fich,fil,col) {
scan(fich) ->x
dim(x)<-c(col,fil)
t(x)
}



#  Función que determina, dado un punto a, en el plano x,y
#  y un conjunto de puntos (x) en el mismo plano, la distancia
#  del punto al más cercano de ellos (dist) y su número de orden (ind)

distpi <- function (a,x)
{

  (a[1]-x[,1])^2+(a[2]-x[,2])^2->xt;
   min(xt)->minxt;
   i<-1:(dim(x)[1]);
   i[xt==minxt]->o;
   list (dist=sqrt(minxt),ind=o)
}


#  Función que determina, dado un vector de puntos, a, en el plano x,y
#  y un conjunto de puntos (x) en el mismo plano, las distancias de
#  cada uno de los puntos al más cercano y su número de orden (ind)

distpsi <- function (a,x)
{
   dim(a)[1]->n
   rep(0,n)->mvd;
   rep(0,n)->mv;
   for(i in 1:n){
     distpi(c(a[i,1],a[i,2]),x)->mvi;
```

```
     mvi$d[1]->mvd[i];
     mvi$i[1]->mv[i];
     }
  list (dist=mvd,ind=mv)
}



# Dibuja el diagrama de ordenación de un análisis de correspondencias, acb

dibujaacb <- function(x,e1=1,e2=2,c1=3,c2=4,titulo="Análisis de
correspondencias"){

plot(c(x$cf[,e1],x$cc[,e1]),c(x$cf[,e2],x$cc[,e2]),type="n",
       xlab=colnames(x$cc)[e1], ylab=colnames(x$cc)[e2],asp=1,
       main=titulo)

dejes()

text(x$cc[,c(e1,e2)],rownames(x$cc),col=c2)

text(x$cf[,c(e1,e2)],rownames(x$cf),col=c1)
a<-0
}


# Dibuja el diagrama de ordenación de un análisis de componentes principales,
acp

dibujaacp <- function(x,e1=1,e2=2,c1=3,c2=4,titulo="Análisis componentes
principales"){

plot(x$cf[,e1],x$cf[,e2],type="n",
       xlab=colnames(x$cc)[e1], ylab=colnames(x$cc)[e2],asp=1,
       main=titulo)

dejes()

arrows(0,0,x$cc[,e1],x$cc[,e2],col=c2)

text(x$cc[,c(e1,e2)],rownames(x$cc),pos=1,col=c2)

text(x$cf[,c(e1,e2)],rownames(x$cf),col=c1)

a<-0

}

a01 <- function (x,max=1){

 (x - min(x)) / (max(x)-min(x)) * max

#print (x)

}

#ordena la tabla de datos en función de una ordenación
#matriz de datos:       x
#objetos de ordenación: y
#eje para ordenar:      e
ordenamatriz <- function (x,y,e=1){
filas   <-order(y$cf[,e])
columnas<-order(y$cc[,e])
x[filas,columnas]
```

```
}

#mediante stars
#dibuja la matriz de datos                    x
#ubicando los resultados según la ordenación x.ord
#mediante un código de corrección             escala
#proporcionando color mediante                color

dibujadatos<- function (x,x.ord,e1=1,e2=2,escala=10,color=2){
stars(x,locations=x.ord$cf[,1:2]*escala,
      col.stars=color);dejes()
}

# Dibuja la curva normal sobre un histograma de frecuencias

histnorm <- function (x,a) {
   c<-x[!is.na(x)];
   hist(c,a,freq=F);
   i<-seq(min(c),max(c),by=(max(c)- min(c))/100);
   lines(i,dnorm(i,mean(c),sd(c)),col=2)
}



# Calcula diversos índices de distancias binarias

idb<-function(b,method="jaccard",dec=3){

b<-as.matrix(b)

(b) %*% t(b)->ta;
(b) %*% t(!b)->tb;
(!b) %*% t(b)->tc;
(!b) %*% t(!b)->td;

(round(
switch(method,
          jaccard     = (ta)    / (ta+tb+tc),             #Índice de Jaccard
          simmat      = (ta+td) / (ta+tb+tc+td),          #Simple matching
          czekanowski = (2*ta)  / (2*ta+tb+tc+td),        #I Czekanowski
          ochiai      = (2*ta)  / sqrt((ta+tb)*(ta+tc)),  #I Ochiai
          mozley      = ta*(ta+tb+tc+td) / ((ta+tb)*(ta+tc)) )  #I Mozley

,dec))

}


# Dibuja intervalos de confianza para n muestras al azar

limnorm <- function(n=30,nl=100,alfa=0.10){

lim<-1:1
lsm<-1:1

Ialfa<-qnorm(1-alfa/2)

for(i in 1:nl){
x<-rnorm(n);
mx<-mean(x);
lim[i]<-mx-Ialfa/sqrt(n);
lsm[i]<-mx+Ialfa/sqrt(n)
}
```

```
plot(c(0,nl+1),range(c(lim,lsm)),type="n",
     xlab="Muestras",ylab="Límites de confianza")
i<-1:nl;
rect(i,lim,i,lsm);
points(i,lim);
points(i,lsm,col=2);
abline(0,0,col=3)
sum(lim>0 | lsm<0)
}


# Dibuja intervalos de confianza para n muestras al azar con varianza
desconocida

limnormvd <- function(n=30,nl=100,alfa=0.10){

lim<-1:1
lsm<-1:1

talfa<-qt(1-alfa/2,n-1)

for(i in 1:nl){
x<-rnorm(n)
mx<-mean(x)
dx<-sd(x)
lim[i]<-mx-talfa*dx/sqrt(n)
lsm[i]<-mx+talfa*dx/sqrt(n)
}

plot(c(0,nl+1),range(c(lim,lsm)),type="n",
     xlab="Muestras",ylab="Límites de confianza")
i<-1:nl;
rect(i,lim,i,lsm);
points(i,lim);
points(i,lsm,col=2);
abline(0,0,col=3)
sum(lim>0 | lsm<0)
}



# Pruebas de normalidad

normalidad<-function(x){
print(shapiro.test(x))
qqnorm(scale(x));abline(0,1)
}


# Cálculo y representación de una regresión no significativa

reg.norm.indp <- function (n){
x <- rnorm(n)
y <- rnorm(n)


par(mfrow=c(2,2))

plot(x,y,main="X frente Y")

par(ask=T)

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

```
tit<-c("X frente Y:",paste("y = ",reg$coef[2]," * x + ",reg$coef[1]))
plot(x,y,main=tit)
abline(reg$coef[1],reg$coef[2],col=3)

tit<-c(tit,"Valores estimados y residuos")
plot(x,y,main=tit)
abline(reg$coef[1],reg$coef[2],col=3)
points(x,x*reg$coef[2]+reg$coef[1],col=2)
segments(x,y,x,x*reg$coef[2]+reg$coef[1],col=5)


tit<-"Residuos frente a x"

plot(x, scale(y-x*reg$coef[2]+reg$coef[1]),
   main=tit,ylab="Residuos estandarizdos")
abline(0,0)

print(summary(reg))


par(ask=F)

par(mfrow=c(1,1))

}


# Cálculo y representación de una regresión significativa

reg.norm.dp <- function (n,s){
x <- rnorm(n)
y <- rnorm(n)
y <- x + y*s

par(mfrow=c(2,2))

plot(x,y,main="X frente Y")

par(ask=T)

reg<-lm(y~x)

tit<-c("X frente Y:",paste("y = ",reg$coef[2]," * x + ",reg$coef[1]))
plot(x,y,main=tit)
abline(reg$coef[1],reg$coef[2],col=3)

tit<-c(tit,"Valores estimados y residuos")
plot(x,y,main=tit)
abline(reg$coef[1],reg$coef[2],col=3)
points(x,x*reg$coef[2]+reg$coef[1],col=2)
segments(x,y,x,x*reg$coef[2]+reg$coef[1],col=5)


tit<-"Residuos frente a x"

plot(x, scale(y-x*reg$coef[2]+reg$coef[1]),
   main=tit,ylab="Residuos estandarizdos")
abline(0,0)

print(summary(reg))


par(ask=F)
```

```
par(mfrow=c(1,1))

}


##############


### $Id: Tukey.R,v 1.1 2000/05/05 22:31:21 bates Exp $
###
### Tukey multiple comparisons for R
###
### Copyright 2000-2000 Douglas M. Bates <bates@stat.wisc.edu>
###
### This file is made available under the terms of the GNU General
### Public License, version 2, or at your option, any later version,
### incorporated herein by reference.
###
### This program is distributed in the hope that it will be
### useful, but WITHOUT ANY WARRANTY; without even the implied
### warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
### PURPOSE. See the GNU General Public License for more
### details.
###
### You should have received a copy of the GNU General Public
### License along with this program; if not, write to the Free
### Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
### MA 02111-1307, USA

Tukey <- function(x, ...) UseMethod("Tukey")

Tukey.aov <-
    function(x, order = FALSE, conf.level = 0.95, ...)
{
    mm <- model.tables(x, "means")
 tabs <- mm$tables[-1]
    nn <- mm$n
    out <- vector("list", length(tabs))
    names(out) <- names(tabs)
    MSE <- sum(resid(x)^2)/x$df.residual
    for (nm in names(tabs)) {
        means <- as.vector(tabs[[nm]])
        nms <- names(tabs[[nm]])
        n <- nn[[nm]]
        ## expand n to the correct length if necessary
        if (length(n) < length(means)) n <- rep(n, length(means))
        if (as.logical(order)) {
            ord <- order(means)
            means <- means[ord]
            n <- n[ord]
            if (!is.null(nms)) nms <- nms[ord]
        }
        center <- outer(means, means, "-")
        keep <- lower.tri(center)
        center <- center[keep]
        width <- qtukey(conf.level, length(means), x$df.residual) *
            sqrt((MSE/2) * outer(1/n, 1/n, "+"))[keep]
        dnames <- list(NULL, c("diff", "lwr", "upr"))
        if (!is.null(nms)) dnames[[1]] <- outer(nms, nms, paste, sep =
"-")[keep]
        out[[nm]] <- array(c(center, center - width, center + width),
                           c(length(width), 3), dnames)
    }
    class(out) <- "Tukey"
```

```
    out
}



dib.dbinom<-function(n=5,p=0.5,desp=0,color=2,dec=3){
# Permite representar la función de distribución binomial
# variando el valor de p, cambiando desp y color
x<-0:n
Prob<-dbinom(x,n,p)
if(desp==0)plot(x,Prob,col=color,type="h")
if(desp!=0)lines(x+desp,Prob,col=color,type="h")
round(Prob,dec)->Prob
matrix(Prob,n+1)->Prob
colnames(Prob)<-"Prob"
rownames(Prob)<-as.character(x)

(dib.dbinom<-Prob)
}


dib.df<-function(df1=10,df2=10,max=10,desp=0,color=2,...){
# Permite representar la función de distribución binomial
# variando el valor de p, cambiando desp y color
x<-seq(0,max,0.1)
F<-df(x,df1,df2)
if(desp==0)plot(x,F,col=color,type="l",...)
if(desp!=0)lines(x,F,col=color)
abline(v=0,h=0,col="pink")
}


dib.dpois<-function(landa=0.5,desp=0,color=2,dec=3){
# Permite representar la función de distribución binomial
# variando el valor de p, cambiando desp y color
Prob<-dpois(0:1000,landa)
Prob[cumsum(Prob)<0.99]->Prob
n<-length(Prob)-1
x<-0:n
if(desp==0)plot(x,Prob,col=color,type="h")
if(desp!=0)lines(x+desp,Prob,col=color,ylim=c(0,max(max(Prob),0.5)),type="h")
round(Prob,dec)->Prob
matrix(Prob,n+1)->Prob
colnames(Prob)<-"Prob"
rownames(Prob)<-as.character(x)

(dib.dpois<-Prob)
}


dibuja.idb<-function(x)plot(hclust(as.dist(1-x)),hang=-1)


dibuja.kmeans<-function(x,k,n=5){
kmeans(x,k,iter.max=n)->x.cl
plot(x[,1:2],col=x.cl$cl)
points(x.cl$ce,pch=17,col=3)
text(x.cl$ce[,1], x.cl$ce[,2],pos=1,cex=3,col=2)
(dibuja.kmeans<-x.cl)
}

mapa.europa<-function(a=1,b=-1){
cmdscale(eurodist)->mds.ed
plot(a*mds.ed[,1],b*mds.ed[,2],type="n")
```

```
text(a*mds.ed[,1],b*mds.ed[,2],rownames(mds.ed))
}
```